Игорь Гайдышев

Решение научных и инженерных задач средствами Excel, VBA и C/C++

+ CD-ROM

- Обработка и анализ данных
- Статистический контроль качества
- Математическое моделирование
- Распознавание образов
- Создание макросов XLM и VBA
- Разработка DLL и XLL для MS Excel

МАСТЕР_РЕШЕНИЙ_

Игорь Гайдышев

Решение научных и инженерных задач средствами Excel, VBA и C/C++

Санкт-Петербург
«БХВ-Петербург»
2004

УДК 681.3.068+800VBA,C++
ББК 32.973.26-018.1
Г14

Гайдышев И. П.

Г14 Решение научных и инженерных задач средствами Excel, VBA и C/C++. — СПб.: БХВ-Петербург, 2004. — 512 с.: ил.

ISBN 5-94157-477-0

Книга посвящена практическим вопросам программирования. Подробно рассматривается технология создания недорогого и максимально адаптированного для пользователей всех квалификаций программного обеспечения для анализа данных и математического моделирования. Изложение ведется как в применении к официальным, так и — впервые в литературе — альтернативным средствам разработки, причем последним уделяется большое внимание. На прилагаемом компакт-диске даются полные исходные тексты программ, проекты для различных средств разработки и дистрибутивы программ и дополнительных модулей для электронных таблиц. Все программы тщательно протестированы. Книга будет полезна в качестве практического пособия преподавателям и студентам, программистам, научным работникам и инженерам-исследователям.

Для программистов

УДК 681.3.068+800VBA,C++
ББК 32.973.26-018.1

Группа подготовки издания:

Главный редактор	*Екатерина Кондукова*
Зам. главного редактора	*Игорь Шишигин*
Зав. редакцией	*Григорий Добин*
Редактор	*Елена Кашлакова*
Компьютерная верстка	*Татьяны Валерьяновой*
Корректор	*Евгений Камский*
Оформление серии	*Via Design*
Дизайн обложки	*Игоря Цырульникова*
Зав. производством	*Николай Тверских*

Лицензия ИД № 02429 от 24.07.00. Подписано в печать 19.03.04.
Формат 70×100 1/16. Печать офсетная. Усл. печ. л. 41,28.
Тираж 3000 экз. Заказ №
"БХВ-Петербург", 190005, Санкт-Петербург, Измайловский пр., 29.

Гигиеническое заключение на продукцию, товар № 77.99.02.953.Д.001537.03.02
от 13.03.2002 г. выдано Департаментом ГСЭН Минздрава России.

Отпечатано с готовых диапозитивов
в Академической типографии "Наука" РАН
199034, Санкт-Петербург, 9 линия, 12.

ISBN 5-94157-477-0

© Гайдышев И. П., 2004
© Оформление, издательство "БХВ-Петербург", 2004

Содержание

От автора **7**
Введение **8**
Глава 1. Разработка элементов пользовательского интерфейса на языке VBA **18**
Решаемая проблема 18
Редактор Visual Basic 20
Создание меню средствами VBA 22
Создание диалоговых панелей 29
Ввод, верификация данных средствами VBA и обработка аварийных ситуаций 38
Рекомендуемая литература 48
Глава 2. Разработка дополнительных модулей для Microsoft Excel **49**
Решаемая проблема 50
Разработка надстроек для Microsoft Excel 50
Дополнительные модули 58
Стандартные DLL 65
Организация работы Microsoft Excel с программами и пользовательскими функциями 68
Особенности отладки и тестирования расчетных программ 72
Ресурсы Интернет 75
Рекомендуемая литература 76
Глава 3. Предварительная обработка данных на VBA и C/C++ **78**
Решаемая проблема 78
Проверка типа распределения эмпирических данных 79
Обработка пропущенных данных 111
Обработка выбросов 118
Руководство пользователя 127
Тестирование и пример применения 135
Ресурсы Интернет 136
Рекомендуемая литература 136
Глава 4. Информационный анализ на «чистом» VBA **138**
Решаемая проблема 138
Особенности решения 138

Научные алгоритмы ... 139
Руководство пользователя и сообщения об ошибках ... 144
Рекомендуемая литература ... 147

Глава 5. Язык макросов Microsoft Excel и VBA: статистическая проверка гипотез ... 148

Решаемая проблема ... 148
Особенности решения ... 149
Научные алгоритмы ... 150
Руководство пользователя ... 173
Пример применения ... 177
Рекомендуемая литература ... 178

Глава 6. Многомерный анализ: факторный анализ в VBA и C ... 180

Решаемая проблема ... 180
Особенности решения ... 182
Научные алгоритмы ... 183
Руководство пользователя ... 208
Дальнейшие исследования ... 211
Рекомендуемая литература ... 211

Глава 7. Распознавание образов на VBA и C ... 213

Решаемая проблема ... 213
Распознавание образов без обучения ... 214
Распознавание образов с обучением ... 245
Пример применения ... 296
Рекомендуемая литература ... 296

Глава 8. Корреляционный анализ в кодах VBA и C ... 298

Решаемая проблема ... 298
Особенности решения ... 299
Научные алгоритмы ... 299
Руководство пользователя ... 309
Дальнейшие исследования ... 312
Пример практического применения ... 312
Рекомендуемая литература ... 313

Глава 9. Методы математического моделирования в программах VBA и C/C++ ... 315

Решаемая проблема ... 315
Особенности решения ... 317
Научные алгоритмы ... 318
Численное решение дифференциальных уравнений ... 331

Идентификация параметров дифференциальных уравнений 367
О приложении дифференциальных уравнений к вычислению статистических распределений 370
Ошибки численных схем 372
Руководство пользователя и пример применения 373
Дальнейшие исследования 377
Рекомендуемая литература 377

Глава 10. Рекурсии на примере многочленов в виде XLL на XLM и C 380

Решаемая проблема 380
Особенности решения 381
Научные алгоритмы 381
Руководство пользователя 389
Рекомендуемая литература 390

Глава 11. Статистический контроль качества и графические средства VBA 392

Решаемая проблема 392
Научные алгоритмы 393
Руководство пользователя 403
Ресурсы Интернет 405
Рекомендуемая литература 405

Глава 12. Разработка справочной системы и генерация дистрибутива.....406

Решаемая проблема 406
Разработка справочной системы 407
Генерация дистрибутива 426
Рекомендуемая литература 434

Приложение 1. Разработка дополнительных модулей для StarOffice и OpenOffice.org 435

Решаемая проблема 435
История StarOffice 436
Средства разработки для StarCalc/Calc 437
Особенности разработки дополнительных модулей для StarCalc/Calc 438
Использование add-in для StarCalc/Calc 444
Рекомендуемая литература 445

Приложение 2. Статистические распределения 446

Вычисление распределений 446
Нормальное распределение 456
Хи-квадрат - распределение 458

Распределение Стьюдента 459
Рекомендуемая литература 461

Приложение 3. Разработка консольных приложений на C/C++ 463

Решаемая проблема 463
Рекомендуемая литература 465

Приложение 4. Средства программирования (обзор) 466

Операционные системы 467
Языки научного программирования 481
Ресурсы Интернет 489
Рекомендуемая литература 489

Приложение 5. Работа с прилагаемым компакт-диском 491

Технические требования 491
Установка программного обеспечения 491
Запуск программ и работа с ними 496
Техническая поддержка 496
Правовые вопросы использования программного обеспечения 496
Содержимое прилагаемого компакт-диска 501

Заключение 502

От автора

Первая наша книга восполнила только половину широкого зазора между теорией и практикой научного анализа данных и математического моделирования — она представила современные, надежные и быстрые реализации проверенных алгоритмов. На ее создание, вместе с программной отработкой алгоритмов и практическими исследованиями, ушло полтора десятка лет. Благодаря длительному тестированию реализация программных продуктов занимает, как это и планировалось, минимальное время на базе любой программной платформы, поддерживающей применяемые нами средства разработки на языках C и C++.

Нынешняя книга, благодаря качественной предварительной работе, создавалась гораздо быстрее. Она завершает проект превращения формул в работающие программы. Мы постарались сделать это прямым и понятным путем. Все рассмотренные в книге программы работают и уже приносят практическую выгоду тысячам пользователей. После завершения работы мы, с некоторым удивлением для себя, поняли, что, по всей видимости, нам посчастливилось найти оптимальное решение по доставке научных алгоритмов анализа массовому, в первую очередь, отечественному пользователю.

Наш молодой коллега однажды спросил, ознакомившись с версткой книги: «Неужели это все из головы?» Это из головы, хотя и базируется на исследованиях, сделанных и другими авторами, и нами. Так было принято в науке: по крупицам собирать факты, делать выводы и, в конце концов, становиться базой для дальнейших исследований других авторов. В нашей книге читатель найдет только результаты оригинальных исследований и полные тексты оригинальных работающих программ. Здесь нет компиляции найденных в сети материалов, которые так любят некоторые студенты - представители поколения Интернета — предъявлять в качестве рефератов и курсовых работ. Здесь нет заимствований размером в полглавы и, надеемся, нет слишком уж грубых ошибок.

Надеемся, читатель благожелательно оценит наш труд.

Игорь Гайдышев

Введение

«Компьютерный мир захвачен потребительским образом мышления, для которого важна не культурная ценность той или иной вещи, а лишь то, сколько эта вещь стоит».
Т. Полгар
«Когда машина станет универсальной, что будут делать люди?»
К. Ривз

Проблема

В нашей книге изложение практики программирования мы ведем в применении к научно-техническим программам, что совсем не исключает применение рассмотренных алгоритмов для любых других прикладных задач. Методики с успехом применялись, кроме решения научных и инженерных задач, при разработке различных программ экономического и хозяйственного назначения. Это было доказано нашими коллегами по работе в Российском научном центре «Восстановительная травматология и ортопедия» имени академика Г.А. Илизарова (Генеральный директор профессор В.И. Шевцов).

Итак, к научно-техническим программам можно отнести:

- программы математического моделирования;
- программы статистической обработки экспериментальной информации, анализа данных различной природы; программы статистического контроля качества;
- системы распознавания образов;
- программы аналитических вычислений;
- экспертные системы;
- программы управления и автоматизации эксперимента и производственных процессов; программы управления различными устройствами как гражданского, так и военного назначения[1].

Эти программы в реальной жизни могут исполняться на различных типах компьютеров, существовать в виде бинарных модулей (software) или быть

---

[1] *По словам Уинстона Черчилля, «что бы ученые ни делали, у них всегда получается оружие...».*

встроенными в различные устройства, а также пересекаться. Программы последнего типа часто являются заказными и имеют узкое специальное применение, не позволяющее не только описать их в доступной литературе, но даже предположить об их существовании. В нашей книге мы остановимся на создании программ, в основном относящихся к первым трем разделам научно-технического программирования.

Несмотря на обширную номенклатуру программ математического моделирования, анализа данных и распознавания образов, слишком оптимистично утверждать, что это программное обеспечение распространено в России настолько широко, насколько это необходимо для обеспечения статистической обработки данных, анализа, прогнозирования и математического моделирования. Можно предположить, что основные причины сложившейся ситуации:

- Высокая стоимость программ. Научная сложность, продолжительность процесса разработки и отладки прикладных программных продуктов рассматриваемого класса и малый тираж никогда не смогут снизить их цену до приемлемых для массового потребителя величин.
- Необходимость изучения новых программных продуктов. Некоторые программы имеют нестандартные или слишком запутанные интерфейсы, и выполнить нужную функцию, не уделив длительного времени обучению, нелегко даже специалисту, профессионально разбирающемуся в методах анализа.

Положение несколько сглаживается распространением так называемых «пиратских» копий программ. Отсутствие гарантий правильной работы программы[2], документации и технической поддержки, юридическое преследование – только некоторые из бед, подстерегающих пользователя на этом опасном пути. Противоправные действия по использованию «пиратского» программного обеспечения пользователи пытаются оправдать благородством целей. Но согласно рекомендациям по представлению результатов научных изысканий в печати, в любой научной работе следует указать, какими программными средствами научного анализа пользовался автор. Ссылка на «пиратскую» копию программы будет нонсенсом настолько же, насколько сообщение о том, что научное исследование выполнено с помощью украденного прибора.

Назрела необходимость создания максимально доступного пользователям разной квалификации программного анализа данных. Доступные для изучения, тщательно прокомментированные, простые и понятные тексты программ дают возможность проверить правильность реализации алгоритмов и информационную безопасность программного обеспечения. Публикация алгоритмов и кодов

---

[2] *О соотношении принципа «As is» и прав покупателя получить за свои деньги качественный программный продукт см. статью Г. Григоренко в «КомпьютерПресс», № 1, 2003 г., с.179.*

в книге позволяет сделать это, не подвергая опасности свою интеллектуальную собственность. Не основной, но как надеется автор, важной целью книги, является демонстрация сложности решения вычислительных задач, не пугающей, а вполне преодолимой.

Предмет книги

Программный анализ данных может быть выполнен несколькими способами:

- Stand-alone программа (package) или приложение (application). Для анализа данных с помощью рассматриваемой категории программ, в принципе, не требуется другое программное обеспечение, кроме операционной системы.
- Web-based решения: апплет (applet) или сценарий (script). Выполнение производится под управлением браузера (browser), независимо от аппаратно-программной платформы.
- Надстройка (дополнительный модуль, add-in) к другой программе. Нужно иметь базовую программу, позволяющую работать с надстройками (это может быть текстовый процессор, электронные таблицы, программа обработки векторной или растровой графики, программа анализа данных и другие программы), а надстройка призвана увеличить ее функциональность.

Книга посвящена надстройкам. Рассмотрены разнообразные возможности для разработки модулей анализа и математического моделирования, значительно усиливающих аналитические способности электронных таблиц. Не говорится об инструментарии программиста и библиотеках разработчика, главным вопросом при использовании которых является решение дилеммы: применить их на свой страх и риск (в том числе финансовый) или создать свои собственные библиотеки, что не так уж и сложно. Не рассматриваются также проблемы анализа данных на примерах известных программных продуктов.

В соответствии с нашими представлениями, в книге рассматривается процесс разработки программы как наукоемкого изделия, от строгого теоретического обоснования до практически готового программного продукта. При желании, читатель может пройти весь этот путь либо подключиться к решению на любом этапе.

Популярные электронные таблицы Microsoft Excel обладают набором тщательно протестированных статистических функций (параметрические тесты, стандартные распределения, регрессионный анализ). Пакет анализа Excel

содержит такие методы, как:

- описательная статистика,
- параметрические методы проверки гипотез,

- дисперсионный анализ,
- вычисление коэффициента корреляции.

Не хватает, как минимум, следующих методов:

- проверка типа эмпирического распределения, в частности, проверка нормальности распределения,
- обработка пропущенных значений,
- обработка выбросов,
- информационный анализ,
- непараметрические методы проверки гипотез,
- факторный анализ,
- распознавание образов без обучения и с обучением,
- корреляционный анализ,
- статистический контроль качества.

В нашей книге в качестве примеров рассмотрены проекты, реализующие все перечисленные методы анализа Microsoft Excel, а в *приложении 1* приводится пример альтернативного решения для электронных таблиц офисных пакетов OpenOffice.org и Sun StarOffice. Большинство программных решений представляют собой модули разработанных нами программных продуктов AtteStat и ME.com. Часть модулей распространяется на коммерческой основе, часть свободно распространяется. Исходные тексты всех модулей доступны.

В показанных листингах применяется ряд функций, обычно без каких-либо модернизаций, улучшений и исправлений, на языках C и C++ (операции с массивами, линейная алгебра, статистические распределения) из нашей первой книги, но исходные тексты даны в приложении на компакт-диске. Если было удобно для реализации конкретных программ, когда применение языка C казалось излишним, составлялись аналоги на языке VBA.

Структура книги

Книга состоит из двенадцати глав и пяти приложений.

Глава 1 посвящена избранным вопросам создания элементов пользовательского интерфейса на языке VBA. Глава не может служить справочником и руководством по VBA. В главе описаны только минимальные сведения, которых достаточно, чтобы создать работающий интерфейс к модулям программы анализа данных, представленным в главах 3–8. Автор работает с VBA не очень продолжительное время, однако мощность и простота данного языка программирования позволила добиться устойчивых успехов.

Глава 2 посвящена вопросам программирования дополнительных модулей для Microsoft Excel. В ней нет теоретических сведений. Однако в ней нет и лишних сведений, втягивающих в обсуждение сложных вопросов системного программирования и, как огромный рекламный щит, отвлекающих от прямой дороги быстрого практического создания нормально функционирующей программы. Приведенная информация тщательно проверена и автор полагает, что все предлагаемые решения действительно работают. Техническая информация, без сомнения, обратит внимание программистов на удобство и простоту создания дополнительных модулей для Microsoft Excel (да и других программ) и повернет многих программистов, которых отпугнули неполные материалы источников и электронной документации, к данной технологии. На разработку материалов главы было потрачено определенное время, т. к. ряд ключевых сведений в литературе и в сопроводительной программной документации совершенно не представлен. Автор надеется, что его труд по исследованию не пропал даром, и многим программистам сэкономит немало драгоценного времени.

Главы 3–8 посвящены различным особенностям создания научных программ для Microsoft Excel. В качестве примеров в главах рассмотрены реально работающие проекты конкретных разработок по статистическому анализу данных. Рассмотренные подходы могут быть использованы не только для научного программирования, но и для решения любых других прикладных задач. При написании глав использован многолетний опыт работы автора по составлению и применению в практической деятельности различных статистических программ.

Глава 9 посвящена одному из важнейших разделов математического моделирования и научного программирования — решению обыкновенных дифференциальных уравнений. Просто и естественно, но в то же время научно строго, обращается внимание на важнейшие проблемы, часто оставляемые без внимания авторами популярных практических руководств. При написании главы использован опыт работы автора с математическими моделями сложных инженерных механических систем.

Глава 10 посвящена разработке рекуррентных алгоритмов на примере программирования алгоритмов вычисления многочленов.

Глава 11 посвящена наиболее яркому практическому применению элементарных методов анализа данных — некоторым методам статистического контроля качества.

Глава 12 посвящена разработке справочной системы программы и генерации дистрибутива (установочного пакета). Рассмотрены наиболее современные, на момент написания книги, и удобные, с точки зрения автора, средства для решения данных задач.

Приложение 1 посвящено решениям по анализу данных для альтернативных электронных таблиц. Дан функциональный аналог решения из *главы 5*, но

для электронных таблиц других типов. Приводятся уникальные сведения о разработке дополнительных модулей, не затронутые в литературе и отсутствующие в документации.

В приложении 2 рассмотрены стандартные статистические распределения, применяемые в книге.

В приложении 3 рассмотрены вопросы создания консольных приложений, необходимых для отладки алгоритмов.

В приложении 4 дан краткий обзор средств программирования с интересным фактическим материалом.

В приложении 5 подробно описана работа с прилагаемым программным обеспечением. Несмотря на кажущуюся тривиальность, представленные сведения могут оказаться полезными и начинающему пользователю компьютера, и опытному программисту.

Каждая глава заканчивается списком литературы, причем ссылки на источники в тексте глав не делаются, как не делаются и какие-либо объемные заимствования из источников. Это сделано намеренно. Приводимой в книге информации достаточно для самостоятельной работы с прилагаемыми методами расчета и программным обеспечением. Если же приведенной в книге информации все же покажется мало, читатель всегда может собрать набор литературы, рекомендуемый нами, и изучить проблему более подробно. Часть источников давно стала библиографической редкостью, поэтому автор особенно старался сделать ссылки на книги, изданные в последние годы.

Содержание прилагаемого компакт-диска

Неотъемлемой частью книги является компакт-диск, который содержит полные тексты и проекты всех модулей программного обеспечения, описанного в книге. Благодаря наличию законченных проектов книгу в комплекте с компакт-диском можно логическим продолжением первой книги автора.

На компакт-диске находятся также дистрибутивы всех описанных в книге модулей учебных версий программного обеспечения анализа данных и математического моделирования. Отличия учебной версии от полной коммерческой версии заключаются только в отсутствии справочной системы и печатного руководства пользователя, поставляемого по специальному заказу. Учебные версии поставляются только вместе с книгой и сопровождаются исходными кодами (для коммерческих версий они доступны по особому запросу), поэтому книга в комплекте с компакт-диском заменяет собой справочную систему. Работа с прилагаемым компакт-диском описана в приложении 4.

Лицензионное программное обеспечение

Для установки и использования программного обеспечения, дистрибутивы которого содержатся на прилагаемом компакт-диске, необходимы лицензионные программы:

- Операционная система из семейства Microsoft Windows 95/98/98SE/NT4/2000/XP.
- Электронные таблицы Microsoft Excel, входящие в пакет офисных приложений Microsoft Office 97, 2000, XP или 2003 любой редакции, которые мы используем в качестве платформы для представленных в книге разработок.
- Пакет офисных приложений Sun StarOffice 5.2 и выше или пакет офисных приложений с открытыми исходными кодами OpenOffice.org версии 1.0 и выше.

Для работы с книгой в плане изучения, адаптации и совершенствования модулей представленного программного обеспечения дополнительно, в зависимости от принятого решения, может потребоваться следующее лицензионное программное обеспечение:

- Microsoft Excel 97 Developer's Kit — это компонент официального набора электронной документации MSDN Library от Microsoft Corporation, содержащий почти полную информацию по всем аспектам программирования для Microsoft Excel. Для получения некоторых материалов, которые должны присутствовать в данном пакете, читателю придется посетить сайт MSDN.
- Система разработки приложений Microsoft Visual C++ 5.0 и выше любой редакции предназначена для разработки проектов из глав 3–10, кроме проекта из приложения 1, для разработки которого пришлось применить Bloodshed Dev-C++ 4.01. Рекомендации по установке и использованию Microsoft Visual C++ см. в главе 2.
- Система разработки приложений Borland C++ версии 5.5 предназначена для разработки проектов из глав 3–10, как альтернатива дорогостоящему решению Microsoft Visual C++.
- Система разработки приложений Bloodshed Dev-C++ версии 4.01 и выше[3] предназначена для разработки проектов из глав 3–5, 9 и 10, конкурируя с системами разработки Microsoft Visual C++ и Borland C++, и разработки проекта из приложения 1, для которой Microsoft Visual C++ не оказался полезным. Для работы с проектами из глав 6–8 Bloodshed Dev-C++ применить не удалось.

Дополнительные инструментальные средства, необходимые для создания программного продукта, такие, как компиляторы справочной системы или генера-

[3] *На момент данной публикации все проблемы с версией 5 решены не были.*

торы установочных пакетов (дистрибутивов), работа с которыми рассмотрена в *главе 12*, можно получить в комплекте со средствами разработки, приобрести отдельно или из сети Интернет, обратившись по указанным ссылкам.

Ресурсы Интернет

С большинством программных продуктов по анализу данных всех типов можно ознакомиться на сайте **www.statistics.com**[4], где даны подробные ссылки на web-based ресурсы, но не делается различий между stand–alone программами и надстройками. Поэтому перечислим наиболее известные научные надстройки для популярных электронных таблиц.

Надстройки, выполняющие научные расчеты в среде электронных таблиц Excel, поставляют: Addinsoft SARL (**www.xlstat.com**), Alpha System Ltd (**www.user.cityline.ru/~alphasys**), AnalyCorp (**www.analycorp.com**), Analyse–it Software Ltd (**www.analyse-it.com**), BaRaN Systems LLC (**www.baran–systems.com**), Digital Computations Inc. (**www.sigmazone.com**), JWalk & Associates Inc. (**j-walk.com/ss**), Numerical Algorithms Group Limited (**www.nag.co.uk**), NeuroDimension Inc. (**www.nd.com**), NeurOK LLC (**www.neurok.ru**), statistiXL (StatistiXL, **www.statistixl.com**), UNISTAT Ltd (**www.unistat.com**), Vector One Corporation Ltd (**www.vstat.km.ru**) и другие.

Вы можете ознакомиться с предлагаемыми решениями и почерпнуть из них некоторые идеи по усовершенствованию номенклатуры методов и интерфейса своих собственных программ, а иногда и посмотреть, как не нужно делать, если целью разработки поставлено создание полезной программы.

Благодарности

Автор искренне благодарит:

- руководителя АСУ областной больницы д.м.н. В.А. Уткина (г. Курган), за полезное обсуждение проблем программной реализации научных алгоритмов статистического анализа данных,
- кандидата наук В.А. Биллига (Тверской госуниверситет), за проявленное внимание и подробное пояснение механизма разработки надстроек к программному обеспечению Microsoft Office (в основном касательно Microsoft Word),
- Али Абубакирова, генерального директора ООО «Физтех-софт», за консультации по истории операционных систем отечественной разработки,

---

[4] *Здесь и далее все ссылки актуальны на момент составления информации и могут быть без предупреждения изменены.*

- студента-химика Курганского государственного университета и системного администратора РНЦ «ВТО» им. академика Г.А. Илизарова, своего сына А.И. Гайдышева, за тестирование разработанного нами программного обеспечения, его апробацию в своей учебной и научной работе и поддержку корпоративной и домашней компьютерных сетей на базе Microsoft Windows 2000 Server и ALT Linux,
- самого увлеченного компьютерами человека, свою жену С.А. Гайдышеву, за терпение, поддержку, тестирование программ и консультации по использованию электронных коммуникаций,
- руководителя рекламного подразделения М.А. Уланову (РНЦ «ВТО», г. Курган), за название прилагаемого к книге программного обеспечения анализа данных AtteStat, как за определенный лингвистический смысл в немецком и русском языках, так и за классическую для статистических программ аббревиатуру,
- старшего эксперта по выставкам Н.А. Березину (РНЦ «ВТО»), за помощь в работе с англо- и франкоязычными материалами, полезное обсуждение применения математических алгоритмов в гуманитарной сфере, предложенный оригинальный девиз к книге.

Кроме того, автор благодарит всех коллег по прошлой и настоящей работе в стенах РНЦ «ВТО», которые явились инициаторами написания того или иного модуля представленного в книге программного обеспечения:

- руководителя отдела телемедицины д.м.н. М.М. Щудло, за предложение создания модуля информационного анализа и тестирование его на своих научно-медицинских экспериментальных данных,
- научного сотрудника к.м.н. А.Н. Ерохина, за идею точного расчета непараметрического критерия Колмогорова–Смирнова (и сделавшего это «вручную») в электронных таблицах Microsoft Excel, с которого началась разработка модуля по непараметрической статистике,
- руководителя подразделения к.м.н. А.Д. Наумова и научного сотрудника к.м.н. Н.А. Гордиевских, за предложение выполнить факторный анализ своих научно-медицинских экспериментальных данных,
- заведующего отделением к.м.н. Д.А. Попкова, за интересную идею сделать кластерный анализ средствами Microsoft Excel.

Автор особо благодарит Российское представительство корпорации Microsoft за предоставленное в комплекте с лучшим на тот момент компьютером R&K Wiener 4 лицензионное программное обеспечение — операционную систему Microsoft Windows XP Home Edition и стандартный выпуск офисного пакета программ Microsoft Office XP. Это позволило протестировать все разработанные программы на совместимость с самым современным системным и прикладным программным обеспечением.

Рекомендуемая литература

1. Bailar J.C., Mosteller F. Guidelines for Statistical Reporting in Articles for Medical Journals. Amplifications and Explanations // Annals of Internal Medicine, 15 February 1988, 108:266–273. Available from World Wide Web: < **http://www.acponline.org/journals/resource/guidelines.htm** >.

1. Боровиков В.П. STATISTICA: искусство анализа данных на компьютере. Для профессионалов. — СПб.: Питер, 2001.

2. Бююль А., Цефель П. SPSS: искусство обработки информации. Анализ статистических данных и восстановление скрытых закономерностей. — СПб.: ООО «ДиаСофтЮП», 2001.

3. Васильков Ю.В., Василькова Н.Н. Компьютерные технологии вычислений в математическом моделировании. — М.: Финансы и статистика, 2002.

4. Дюк В. Обработка данных на ПК в примерах. – СПб.: Питер, 1997.

5. Кулаичев А.П. Методы и средства анализа данных в среде Windows. STADIA. — М.: Информатика и компьютеры, 1999.

6. Макарова Н.В., Трофимец В.Я. Статистика в Excel: Учебное пособие. — М.: Финансы и статистика, 2002.

7. Тюрин Ю.Н., Макаров А.А. Статистический анализ данных на компьютере. — М.: ИНФРА-М, 1999.[5]

8. Орвис В. Дж. Excel для ученых, инженеров и студентов. — Киев: Юниор, 1999.

[5] *В 2003 г. выпущено дополненное переиздание.*

ГЛАВА 1

Разработка элементов пользовательского интерфейса на языке VBA

«Поскольку сегодняшний пользователь ЭВМ обычно имеет профессиональные интересы, выходящие за рамки программирования, то пользовательский интерфейс должен стимулировать эти интересы, а не сдерживать их».
А. Стивенс
«У Вас есть выбор, на каком языке писать ту или иную часть приложения. Так, пользовательский интерфейс приложения Вы, скорее всего, будете создавать на Microsoft Visual Basic, но прикладную логику лучше всего реализовать на C++.»
Дж. Рихтер

Для работы с материалами главы понадобятся лицензионные средства разработки и тестирования программ:

- Microsoft Excel 97 или выше;
- Microsoft Excel Developer's Kit версии 8 или выше.

Потребуется знание VBA.

Решаемая проблема

Современная программа, предназначенная для обработки научных, финансовых, коммерческих или иных данных, предоставляет возможности для

хранения массивов числовой информации (учитывая возможность преобразования в числовой других видов информации) и программные средства для их обработки. Такой концепции наиболее полно отвечают электронные таблицы, оснащенные дополнительными аналитическими инструментами. Без потери общности, мы можем говорить о научных программах, потому что результаты наших исследований применимы для решения широкого круга задач.

Лучшие специализированные программы статистического анализа данных:

- NCSS — Number Cruncher Statistical Systems (NCSS, **www.ncss.com**),
- SPSS (SPSS Inc., **www.spss.com**),
- STADIA (НПО «Информатика и компьютеры», **statsoft.msu.ru**)
- STATISTICA (StatSoft Inc., **www.statsoft.com**),
- SYSTAT (SYSTAT Software Inc., **www.systat.com**)

и другие — построены также на концепции электронных таблиц. Однако средства манипуляции данными в перечисленных программах несравнимо более ограничены, чем в любых электронных таблицах. Это и понятно — производителям лучших программ анализа данных вряд ли по силам конкурировать в части интерфейса с производителями любых электронных таблиц. У этих программных продуктов — разные задачи и группы пользователей, пересекающиеся незначительно. Однако мы покажем, что вполне возможно создать программный продукт, сосредоточившись на реализации алгоритмов анализа данных и воспользовавшись всеми доступными средствами, предлагаемыми лучшими в мире электронными таблицами.

Остановимся кратко на истории электронных таблиц производства Microsoft, с ними нам предстоит работать на протяжении всей книги. Под руководством Ч. Саймони (Charles Simonyi), перешедшим в Microsoft из исследовательского центра Xerox PARC, в 1981 г. были разработаны электронные таблицы Multiplan для операционной системы MS–DOS, как, годом позднее, и текстовый процессор Word для MS–DOS. Программный продукт Microsoft Multiplan для Apple Macintosh был анонсирован в 1984 г. Программа Multiplan была написана на языке программирования C, чем была обеспечена ее легкая переносимость на микрокомпьютеры с различной архитектурой. Однако эти электронные таблицы оказались менее совершенными, чем продукты конкурентов.

Электронные таблицы (иначе, табличный процессор) Microsoft Excel были разработаны в 1985 г. для платформы Macintosh, и это было стратегически правильным решением. Графические возможности Apple Macintosh и Microsoft Excel обеспечили успех и тому, и другому продукту. В 1987 г. появилась версия Microsoft Excel 2 для операционной системы Microsoft Windows.

Электронные таблицы Microsoft Excel 4 были впервые оснащены специальной версией языка программирования Visual Basic Applications Edition.

Первой 32-разрядной версией стали электронные таблицы Microsoft Excel 7 (другое название Microsoft Excel 95), выпущенные в 1995 г. Версия 8 (Microsoft Excel 97) была разработана в 1997 г. и получила новый интерфейс Visual Basic for Application (VBA), единый для всех компонентов Microsoft Office 97. Благодаря Microsoft Excel и VBA, практически на каждом современном компьютере в распоряжении пользователя имеется мощная система программирования. Компонентом пакета офисных приложений Microsoft Office 2000 стала версия 9 (Microsoft Excel 2000), Microsoft Office XP — версия 10 (Microsoft Excel 2002), Microsoft Office 2003 — версия 11 (Microsoft Excel 2003). Пакет Microsoft Excel доступен и в качестве отдельного продукта для Microsoft Windows и Apple Macintosh.

Представленные в книге дополнительные модули для электронных таблиц Microsoft Excel корректно работают в локализованных версиях Microsoft Excel 97, 2000, 2002 и 2003.

Мы хотим, чтобы с помощью сведений данной главы читатель любой квалификации мог разобраться (или использовать как готовый рецепт, не вникая в подробности) в VBA и легко, как и предполагает VBA, создавать программные проекты любой сложности. Здесь представлен надежно функционирующий каркас полезного приложения, который можно модифицировать или наращивать, по своему усмотрению.

Вывод графиков средствами VBA в настоящей главе не упоминается. Он отдельно рассмотрен в *главе* 11.

Редактор Visual Basic

Разработка программы на языке VBA начинается одинаково для всех представленных в книге проектов, использующих VBA. Все примеры приводятся для русских версий Excel из комплекта офисных приложений Microsoft Office 2000 Professional, под управлением операционной системы MS Windows 98 Second Edition[1], или из комплекта офисных приложений Microsoft Office XP Standard под управлением операционной системы MS Windows XP Home Edition.

После запуска электронных таблиц Microsoft Excel, нажмите комбинацию клавиш <Alt>+<F11> либо выберите из меню Excel последовательность пунктов **Сервис ▸ Макрос ▸ Редактор Visual Basic**. Появится окно интегрированной среды разработки приложений Visual Basic for Application, как на рис. 1.1.

---

[1] *О типах операционных систем см. приложение 4.*

Рис. 1.1. Интегрированная среда разработки приложений Visual Basic for Application

Нажмите правую кнопку мыши в любом месте окна **Project** и выберите из появившегося меню **Insert ▸ Module**. К проекту будет добавлен новый модуль **Module1** (потом можно переименовать), а в появившееся окно модуля введите исходные тексты программы (макроса).

Следует предусмотреть систему меню для легкого доступа к прикладным алгоритмам и диалоговые панели (формы) для выбора или ввода различных параметров, управляющих работой этих прикладных алгоритмов. В *главах 3* и *5* дополнительно приводятся столь же удобные, хотя и функционально ограниченные, решения, позволяющие обойтись не только без отдельной системы меню, но иногда и вообще без кода VBA, воспользовавшись языком макросов Microsoft Excel.

Чтобы придать нашим разработкам вид серьезных программ, в дальнейшем, мы предполагаем, что разработанные программы будут называться AtteStat и ME.com. Разработка будет модульной, подобно настоящей программе анализа данных. Это, на самом деле, недалеко от истины — после добавления всех модулей Microsoft Excel «превратится» в настоящую программу анализа данных.

Создание меню средствами VBA

Добавление пользовательских меню, обеспечивающих удобный доступ к прикладным алгоритмам модульной программы анализа данных, работающей в среде электронных таблиц Microsoft Excel, средствами VBA не является трудной задачей. Язык визуального программирования VBA как и любой современный развитый язык программирования, предлагает несколько решений. На рис. 1.2 показано, что, примерно, мы хотим получить.

Рис. 1.2. Рабочая книга с пользовательским меню

Наша программа должна добавлять пункт меню к существующему меню Microsoft Excel. Так как программа модульная, будет хорошо, если каждый установленный модуль станет добавлять свое меню к общему меню нашей программы. Это сэкономит место на экране и обеспечит удобный доступ к функциям каждого из установленных модулей.

Программные манипуляции с пользовательскими меню удобно делать в функции `Auto_Open`. Имя функции `Auto_Open` зарезервировано Microsoft Excel в качестве стандартного имени. Это значит, что при загрузке рабочей книги или шаблона, содержащих эту функцию, Microsoft Excel автоматиче-

ски исполняет ее код. В конце главы даются рекомендации по управлению выполнением, в обход любого установленного уровня безопасности Microsoft Excel, что вызвано не пренебрежением к информационной безопасности, а обеспечением требуемой функциональности прикладной программы.

Объекты меню

На листинге показана начальная часть функции `Auto_Open` с описаниями компонентов меню, общих для всех модулей программы AtteStat.

Листинг 1.1

```
Sub Auto_Open ()
 '
 ' Загрузка меню
 '
 Dim WorksheetMenuBar As CommandBar    ' Основная панель
 Dim CstmCtrl As CommandBarControl     ' Пункт основного меню
 Dim AtteStatMenu As CommandBarPopup   ' Выпадающее меню AtteStat
 Dim AtteStat As CommandBarControl     ' Пункт меню AtteStat
 Dim MainControl As CommandBarControl  ' Пункт меню модуля
 Dim Exist As Boolean                  ' Флаг существования AtteStat
```

Для построения меню используются следующие стандартные средства Microsoft Office из коллекции `CommandBarControls`:

- объект `CommandBar` — панель меню (панель команд);
- объект `CommandBarPopup` — панель выпадающего меню;
- объекты класса `CommandBarControl` — отдельные пункты меню.

Нам необходимо создать общее меню для всех модулей нашей прикладной программы, затем добавить к нему меню конкретного модуля. Идея достаточно проста: каждый модуль, если он загружается первым, создает пункт меню **AtteStat** (или **ME.com**) и добавляет свое собственное меню. Если он загружается не первым, то пункт меню **AtteStat** уже существует, его создавать не нужно, а только добавить меню модуля к существующему меню **AtteStat**.

Поэтому, перед добавлением нового меню модуля, следует решить важную задачу. Необходимо проверить, существует ли в меню Excel пункт меню **AtteStat**, который мы собираемся добавить, так как Microsoft Excel допускает существование в меню двух и более одинаковых пунктов. Для проверки существования

конкретного пункта меню как раз предназначен флаг `Exist` (см. листинг 1.1), а сама проверка выполняется, как показано на листинге 1.2.

Листинг 1.2

```
'
' Проверка наличия меню AtteStat
'
Set WorksheetMenuBar = CommandBars.ActiveMenuBar
Exist = False
For Each CstmCtrl In WorksheetMenuBar.Controls
  If CstmCtrl.Caption = "&AtteStat" Then
    Exist = True
    Set AtteStat = CstmCtrl
    Exit For
  End If
Next CstmCtrl
```

Данный фрагмент программы установит (поднимет) флаг `Exist` в положение `True`, если пункт меню `&AtteStat` уже существует (знак `&` обозначает последующий символ «горячей» клавиши, если таковые используются в меню программы; использование знака `&` в поиске пункта меню обязательно). Для нас это будет означать, что создавать новый пункт меню не нужно, а следует добавить пункт загружаемого в данный момент модуля в уже существующее меню. Если флаг `Exist` оставлен в положении по умолчанию `False`, то создадим сначала новое меню **AtteStat**. Свойства меню описаны в следующем разделе.

Как принято в современном программировании, минимальный набор пунктов меню должен содержать, помимо вызова прикладной части программы, также вызов справочной системы и информационного окна (рис. 1.3), содержащего сведения о названии программы, номере версии, обладателе авторских прав и юридических последствиях их коварного нарушения.

Рис. 1.3. Информационное окно интегратора модульной программы ME.com

Свойства меню

На листинге 1.3 показан фрагмент программы, формирующий меню модуля кластерного анализа, логика которого описана в *главе 7*, в зависимости от того, существует ли меню AtteStat на момент исполнения данного кода. Содержательные комментарии объясняют суть выполняемых операторов.

Листинг 1.3

```
 '
 ' Формирование меню модуля в составе AtteStat
 '
 Dim ClaMenuBar As CommandBar        ' Панель меню модуля
 Dim ClaMenu As CommandBarPopup      ' Выпадающее меню модуля
 Dim ClaControl As CommandBarControl ' Пункт меню модуля
 If Not Exist Then
   '
   ' Формирование нового меню AtteStat и добавление к нему меню модуля
   '
   Set AtteStatMenu =
WorksheetMenuBar.Controls.Add(Type:=msoControlPopup, _
                      Temporary:=True)
   AtteStatMenu.Caption = "&AtteStat"
   Set ClaMenu = AtteStatMenu.Controls.Add(Type:=msoControlPopup, _
                 Temporary:=True)

   Set MainControl = AtteStatMenu.Controls.Add(Type:=msoControlButton)
   MainControl.BeginGroup = True
   MainControl.Caption = "&О программе"
   MainControl.OnAction = "Cla.MakeAboutMain"
 Else
   '
   ' Добавление меню модуля к существующему меню AtteStat
   '
   Set ClaMenu = AtteStat.Controls.Add(Type:=msoControlPopup, _
                 Temporary:=True, Before:=1)
 End If
 ClaMenu.Caption = "Модуль &Cla"
```

```
 '
 ' Формирование меню модуля
 '
 Set ClaControl = ClaMenu.Controls.Add(Type:=msoControlButton)
 ClaControl.FaceId = 983
 ClaControl.Caption = "&Справка по Cla"
 ClaControl.OnAction = "Cla.MakeHelp"

 Set ClaControl = ClaMenu.Controls.Add(Type:=msoControlButton)
 ClaControl.FaceId = 82
 ClaControl.Caption = "&Кластерный анализ"
 ClaControl.OnAction = "Cla.RunSolution"

 Set ClaControl = ClaMenu.Controls.Add(Type:=msoControlButton)
 ClaControl.BeginGroup = True
 ClaControl.Caption = "&О программе"
 ClaControl.OnAction = "Cla.MakeAbout"
End Sub
```

Подробные описания методов и свойств даны во встроенной документации VBA и в указанных в конце главы источниках. В данном фрагменте используются свойства:

- `FaceId` — номер изображения, появляющегося перед пунктом меню;
- `Caption` — заголовок пункта меню;
- `OnAction` — метод, исполняемый при выборе пункта меню;
- `BeginGroup` — разделитель.

Сделаем подробные пояснения относительно свойства `FaceId`. Полный список изображений и соответствующих им номеров не является одинаковым для каждого компьютера, а зависит от установленных на нем программ. Соответствие номера конкретному изображению поможет получить показанная на листинге 1.4 компактная утилита (вспомогательная программа) в коде VBA. Макрос создает два пункта меню (можно изменить их количество, по своему усмотрению), которые в листинге 1.4. мы назвали **1–300** и **301–600** (в примере на компакт-диске пунктов меню немного больше). В каждом пункте меню выводится по 300 изображений с номерами, которые им соответствуют. Программисту остается выбрать понравившееся изображение и указать его номер в свойстве `FaceId`.

Листинг 1.4

```
Sub Auto_Open()
 Dim TestIDMenuBar As Object
 Dim TestIDMenu As Object
 Dim TestIDControl As Object

 Set TestIDMenuBar = CommandBars.ActiveMenuBar
 Set TestIDMenu = TestIDMenuBar.Controls.Add(Type:=msoControlPopup, _
                  Temporary:=True)
 TestIDMenu.Caption = "1-300"
 For I = 1 To 300
   Set TestIDControl = TestIDMenu.Controls.Add(Type:=msoControlButton)
   TestIDControl.FaceId = I
   TestIDControl.Caption = Str(I)
 Next I

 Set TestIDMenuBar = CommandBars.ActiveMenuBar
 Set TestIDMenu = TestIDMenuBar.Controls.Add(Type:=msoControlPopup, _
                  Temporary:=True)
 TestIDMenu.Caption = "301-600"
 For I = 301 To 600
   Set TestIDControl = TestIDMenu.Controls.Add(Type:=msoControlButton)
   TestIDControl.FaceId = I
   TestIDControl.Caption = Str(I)
 Next I
End Sub
```

Рабочая книга, содержащая данный макрос, может медленно загружаться на компьютерах с низкой мощностью процессора. В этом случае, можно уменьшить значения счетчика I и создать, например, не 2 пункта меню: от 1 до 300 и от 301 до 600, как в показанном листинге, а один — от 1 до 100. Затем можно проверить, опять однократно, вариант от 101 до 200, и т. д.

Методы меню

Свойства `OnAction` в листинге 1.3 указывают имена методов, которые должны быть исполнены при выборе пользователем того или иного пункта меню. Это могут быть стандартные методы класса `Application` или специально составленные пользовательские функции, как показано на листинге 1.5.

Листинг 1.5

```
Private Sub MakeHelp()
 Application.Help "Cla.hlp"
End Sub
Private Sub MakeAboutMain()
 MainUserForm.Show
End Sub
Private Sub RunSolution()
 ClaUserForm1.Show
End Sub
Private Sub MakeAbout()
 ClaUserForm2.Show
End Sub
```

Метод `Application.Help` заданием параметров вызывает соответсвующую справочную систему, с указанием полного пути к тому каталогу, где находится файл справки. Программисту нужно заранее позаботиться о том, чтобы файл справки уже находился в указанном месте (или в каталоге по умолчанию) до того, как пользователь решит им воспользоваться. Метод одинаково хорошо работает и со справочной системой в формате HLP, и в формате HTML. О создании справочных систем см. *главу 12*. Там же подробно описано местонахождение файлов различных типов, применяемых в наших разработках.

Метод `MainUserForm.Show` вызывает форму (об их проектировании см. следующий раздел) с информацией о проекте AtteStat. Данная форма — общая для всех модулей, составляющих одну программу.

Метод `ClaUserForm1.Show` вызывает диалоговую панель, предназначенную для выбора параметров расчета и запуска прикладных алгоритмов (рис. 7.2). В нашем проекте, помимо общей формы с информацией о программном обеспечении AtteStat, каждый модуль имеет свою собственную информационную форму.

Метод `ClaUserForm2.Show` вызывает форму с информацией о модуле (рис. 1.3). Данная форма — специфическая для каждого разработанного в рамках нашего проекта модуля.

Все представленные в данном модуле функции описаны как `Private`, чтобы исключить их случайный вызов из листа рабочей книги или других модулей и исключить конфликты имен в случае, если в других модулях окажутся функции с такими же именами. Если функции специально предназначены для вызова за пределами содержащего их модуля, они должны быть описаны как `Public`.

Создание диалоговых панелей

Для обеспечения удобства пользователей и упрощения работы программиста, в VBA предусмотрена возможность быстрого создания диалоговых панелей (форм). На рис. 1.4 показано окно редактора Visual Basic в режиме проектирования формы. Неопытным в программировании читателям не следует пугаться слишком сложной картины. Во-первых, редактор весьма удобен. Во-вторых, сложность формы зависит от сложности решаемой задачи. В-третьих, на прилагаемом к книге компакт-диске содержатся готовые коды всех рассмотренных модулей, которые можно использовать в качестве основы для собственных разработок.

Рис. 1.4. Окно редактора Visual Basic в режиме проектирования формы

Основные элементы управления VBA

Для создания новой формы нажмите правую кнопку мыши в любом месте окна **Project** и выберите из появившегося меню **Insert ▸ UserForm**. В проект будет добавлена новая форма UserForm1. Суть визуального проектирования формы заключается в помещении на поле формы путем перенесения мышью элементов управления из палитры элементов управления **Toolbox** редактора VBA (рис. 1.5), определении свойств этих элементов (в специальном окне) и привязке к ним кода VBA, для обработки событий, возникающих при выборе пользователем тех или иных элементов управления.

Элементы управления подробно описаны во встроенной справочной системе и в литературе. Чтобы не пересказывать содержание справочной системы, которая всегда под рукой у VBA-программиста, отметим только те элементы управления, которые используются в наших расчетных программах, (в алфавитном порядке):

- `CheckBox` — флажок (кнопка независимого выбора);
- `ComboBox` — элемент управления, отображающий список величин и позволяющий выбрать одну их них;
- `CommandButton` — командная кнопка;
- `Frame` — рамка;
- `Image` — изображение;
- `Label` — метка (надпись, статический текст);
- `MultiPage` — многостраничный элемент управления;
- `OptionButton` — переключатель (кнопка зависимого выбора, радиокнопка);
- `RefEdit` — поле ссылки на ячейки листа рабочей книги электронной таблицы;
- `TextBox` — поле ввода (окно редактирования, «текстовое» поле).

Рис. 1.5. Палитра элементов управления Toolbox редактора VBA электронных таблиц Microsoft Excel

Флажок `CheckBox` позволяет пользователю выбрать ту или иную опцию, связанную с данным флажком. Он имеет два состояния: `False` — не установлен (это значение по умолчанию), `True` — флажок установлен. Состояние флажка передается в пользовательскую программу через свойство `Value`.

Элемент управления `ComboBox` отображает список величин заданного типа и позволяет пользователю выбрать только одну из этих величин. Дальнейшая обработка выбранного варианта зависит от логики программы. Пример применения `ComboBox` см. в *главе 3*.

Командная кнопка `CommandButton` служит для запуска той или иной процедуры, например, расчетного модуля программы. К командной кнопке можно привязать и другие события, например, установку всех флажков формы в значение `True` или `False` (см. *главу 4*). Можно привязать к кнопке вызов справочной системы или возможность выхода из формы.

Рамка `Frame` обычно применяется для визуального объединения некоторых элементов управления в логически связанную группу. Помимо графического оформления, рамка может выполнять еще одну задачу. Так, элементы управления `OptionButton`, визуально объединенные в рамку, не требуют обязательного составления программистом функций `OptionButton_Click` обработки события типа `Click` для каждого элемента управления `OptionButton`. Достаточно только инициализации формы. Обработка (установка значения `True` только для выбранной кнопки и `False` для всех остальных) будет выполнена автоматически. Если форма имеет несколько рамок, и выбор `OptionButton` должен производиться только для одного элемента из всех рамок, данное преимущество теряется, и программисту придется составлять подпрограммы обработки для каждого элемента.

Элемент управления `Image` служит для включения в форму прямоугольного графического изображения. Данный элемент управления может служить как для улучшения внешнего вида формы, так и для отображения на экране различной графической информации.

Метка `Label` служит для включения в форму надписей (статического текста), служащих для различных целей. Например, с помощью метки `Label` можно ввести в форму различные пояснения относительно элементов управления формы, а также информацию о владельце авторских прав на программу.

Элемент управления `MultiPage` в наших проектах применяется в *главе 7* при создании модуля распознавания образов с обучением. Данный элемент позволяет отобразить в компактной форме большое число элементов управления. В упомянутой главе элемент `MultiPage` позволяет в удобной и компактной форме отобразить как обучающие, так и распознающие алгоритмы.

Переключатель `OptionButton` обеспечивает зависимый выбор (обычно одного из нескольких возможных). Например, переключатель можно применять для выбора одного из нескольких методов расчета либо для выбора техники расчета (см. проекты, представленные в *главах 6—9* и *11*).

Основным элементом управления, обеспечивающем передачу данных из листа рабочей книги электронных таблиц в программу VBA, является поле `RefEdit`. Использование данного поля открывает доступ к широким возможностям Microsoft Excel. С помощью поля `RefEdit` пользователь получает возможность ввести или указать методом протаскивания курсора интервал ячеек Microsoft Excel, содержащих данные для расчета. Указанный в поле `RefEdit` интервал ячеек может находиться не обязательно в активном в данным момент рабочем листе, а в любом листе рабочей книги. Данная приятная особенность делает ненужным дополнительное указание листа (как сделано в стандартном пакете анализа Microsoft Excel), если, например, вывод результатов расчета предполагается в листе, отличном от листа, содержащего исходные данные, или исходные данные берутся из разных листов рабочей книги.

В большинстве известных программ статистического анализа, вместо конструкции, подобной `RefEdit`, для выбора параметров или переменных анализа применяется совершенно другой прием. На рис. 1.6 показано окно выбора параметров программы статистического анализа данных SYSTAT. Пользователю предлагается из списка доступных выбрать переменные для анализа, причем обоснование данного выбора может вызвать у пользователя массу вопросов.

Рис. 1.6. Выбор переменных для расчета коэффициента корреляции в программе SYSTAT

Удобнее ли такой способ, чем типичный для электронных таблиц метод протаскивания курсора — решать пользователю. Отметим только, что способ протаскивания, хотя и малоэффективный в некоторых случаях (например, при большом объеме данных интервалы проще ввести с клавиатуры), отличается гораздо большей гибкостью. А недостатки типичного для программ статистического анализа данных подхода очевидны. Это, прежде всего, выбор только столбца как области исходных данных. Во вторых, столбец берется для расчета целиком — нельзя выделить даже его часть. Примитивность и ограниченность решений выбора переменных для анализа в большинстве стандартных программ статистического анализа, по сравнению с удобством и гибкостью Microsoft Excel и других электронных таблиц, у нас сомнений не вызывает.

В поле ввода `TextBox`, вопреки его названию, вводятся не только данные текстового типа, но и числовые данные различных типов. Это эффективно и удобно. Любые данные вводятся как символьные, а преобразование этих данных в нужный тип производится функцией их обработки. Одновременно можно выполнить и верификацию введенных в данное поле данных.

Свойства каждого элемента управления, помещенного в форму, можно изменять, по усмотрению программиста, с помощью панели свойств, активизируемой при выборе того или иного элемента управления, находящейся, по умолчанию, в левом нижнем углу. В число изменяемых свойств входят наименование, координаты относительно левого верхнего угла формы, абсолютные размеры, цвет, шрифт, индекс пункта контекстно-зависимой помощи и т. д. Для изучения свойств элементов управления проще всего варьировать значения параметров, оценивая результат.

Существуют и другие способы передачи массива данных в модуль VBA, однако они, по нашему мнению, не так удобны и делают решения пользовательской программы менее похожими на стандартные решения Microsoft Excel.

Рекомендации разработчика Microsoft Excel

Очень важно при разработке дополнительных модулей Microsoft Excel создать хороший код, тщательно протестировать его и обеспечить поддержку эффективных методов кодирования. VBA не ограничен применением стандартных разработок, но использование накопленного разработчиками опыта может сделать более легкой разработку и поддержку программ.

Данные рекомендации не имеют целью ограничить творчество программиста. Microsoft Excel — хорошая программа, в смысле удобного обращения и качества реализации представленных в ней методов расчета. Она оставляет программисту место и для творчества, благодаря документированному API

(Application Programming Interface — интерфейс прикладного программирования) и явно недостаточному количеству стандартных методов анализа.

Очень важно, чтобы проект легко интегрировался в Microsoft Excel. Основная рекомендация, при проектировании формы, заключается в требовании как можно более походить на стандартные формы Microsoft Excel, в функциональности и стиле оформления (размер, цвет, шрифт, взаимное расположение элементов управления, содержание надписей и т. п.). Меню дополнительного модуля должны подражать встроенным меню Microsoft Excel. Панели инструментов должны выглядеть так, как будто их разработала корпорация Microsoft. Также должен быть обеспечен доступ к функциям программы с помощью горячих клавиш (называемых еще акселераторами) в диалоговых панелях.

Интерфейс проекта должен выглядеть так, как будто это — встроенная часть Microsoft Excel. Для обеспечения данной рекомендации, предлагается найти готовую диалоговую панель, чтобы использовать ее как модель. Например, кнопки **OK** и **Cancel** в программах Microsoft Office принято располагать в правом нижнем углу формы. Следует убедиться, что управляющие элементы проекта как можно больше походят на встроенные управляющие элементы.

Эффективность программного кода оценивается только через пользовательский интерфейс. Пользователь должен воспринимать внешний вид формы (как и программы в целом) естественно. Все вопросы (если они действительно необходимы) должны быть краткими, но информативными, а ответы на них, в отличие от законодательных актов известного государства, трактоваться однозначно и не нуждаться в пространных комментариях. К тому же, четкое решение составляет хорошую основу для дальнейшего развития проекта.

Обработка событий формы средствами VBA

Теперь рассмотрим код VBA, ответственный за обработку событий, возникающих во время работы пользователя с формой. После двойного щелчка левой кнопкой мыши на имени формы в окне редактора появится окно **Code** с текстом программы VBA, предназначенной для обработки данной формы. Иначе доступ к коду можно получить после нажатия правой кнопки мыши на имени формы с последующим выбором из контекстного меню **View Code**.

Код VBA, прежде всего, содержит функцию `UserForm_Initialize`, исполняемую при вызове формы из меню или из другой формы. В данной функции обычно присваиваются начальные значения свойствам `Value` всех элементов управления формы, в зависимости от их типа. Листинг 1.6 содержит код инициализации следующих объектов:

- 11 кнопок `LinkButton` (их имена введены в соответствующем окне свойств в левом нижнем углу окна редактора) типа `OptionButton`, предназначенных для выбора меры различия для кластерного анализа (*глава 7*);
- 3 кнопок `MethodButton` также типа `OptionButton`, ответственных за выбор метода решения;
- 2 кнопок `ObjectButton` типа `OptionButton`, управляющих порядком ввода исходных данных из таблицы.

Листинг 1.6

```
Private Sub UserForm_Initialize()
 LinkButton1.Value = True
 LinkButton2.Value = False
 LinkButton3.Value = False
 LinkButton4.Value = False
 LinkButton5.Value = False
 LinkButton6.Value = False
 LinkButton7.Value = False
 LinkButton8.Value = False
 LinkButton9.Value = False
 LinkButton10.Value = False
 LinkButton11.Value = False
 MethodButton1.Value = True
 MethodButton2.Value = False
 MethodButton3.Value = False
 MethodButton4.Value = False
 ObjectButton1.Value = True
 ObjectButton2.Value = False
End Sub
```

Структуры функций обработки нажатий перечисленных кнопок одинаковы. Пример такой функции показан на листинге 1.7.

Листинг 1.7

```
Private Sub MethodButton2_Click()
 MethodButton1.Value = False
 MethodButton2.Value = True
 MethodButton3.Value = False
 MethodButton4.Value = False
End Sub
```

Если для логического объединения нескольких кнопок зависимого выбора применяется элемент управления Frame, то, как было указано выше, в применении функций обработки нажатия кнопок, подобной функции, показанной на листинге 1.7, необходимость отпадает. Данные события будут обработаны автоматически.

Более подробного рассмотрения заслуживает функция обработки нажатия командной кнопки, ответственной за запуск программы расчета. Задачами данной функции являются:

- формирование набора параметров расчета, в соответствии с выбором пользователя,
- верификация введенных или выбранных пользователем параметров расчета (не всегда),
- вызов функции, выполняющей верификацию исходных данных (если это не было сделано заранее), производящей расчет и осуществляющей вывод результатов.

В итоге получается то, что во времена перфокарт на электронно-вычислительных машинах (ЭВМ) называлось пакетной обработкой данных, а колода перфокарт для расчета конструкции методом конечных элементов была высотой от плоскости стола до потолка.

Код одной из возможных реализаций функции показан на листинге 1.8.

Листинг 1.8

```
Private Sub CommandButton1_Click()
 Dim Link As Integer
 Dim Met As Integer
 Dim Obj As Integer
 If LinkButton1.Value = True Then
   Link = 0
 ElseIf LinkButton2.Value = True Then
   Link = 1
 ElseIf LinkButton3.Value = True Then
   Link = 2
 ElseIf LinkButton4.Value = True Then
   Link = 3
 ElseIf LinkButton5.Value = True Then
   Link = 4
 ElseIf LinkButton6.Value = True Then
   Link = 5
```

```
 ElseIf LinkButton7.Value = True Then
   Link = 6
 ElseIf LinkButton8.Value = True Then
   Link = 7
 ElseIf LinkButton9.Value = True Then
   Link = 8
 ElseIf LinkButton10.Value = True Then
   Link = 9
 Else
   Link = 10
 End If
 If MethodButton1.Value = True Then
   Met = 1
 ElseIf MethodButton2.Value = True Then
   Met = 2
 ElseIf MethodButton3.Value = True Then
   Met = 3
 Else
   Met = 4
 End If
 If ObjectButton1.Value = True Then
   Obj = 1
 Else
   Obj = 2
 End If
 Call MakeCalc(RefEdit1.Value, RefEdit2.Value, Link, Met, _
               Obj, TextBox1.Text)
 Unload ClaUserForm1
End Sub
```

Функция обработки нажатия командной кнопки, ответственной за запуск программы расчета, заканчивается разрушением (закрытием) формы. Точно так же заканчивается и функция обработки другой командной кнопки, ответственной за разрушение формы, если пользователь решил не выполнять расчет. Листинг данной функции показан ниже.

Листинг 1.9

```
Private Sub CommandButton2_Click()
 Unload ClaUserForm1
End Sub
```

Нами выбрано одно из возможных решений, заключающееся в том, что верификацию введенных данных, вывод результатов расчета и сообщений об ошибках производит функция с именем `MakeCalc`. Такие функции имеются в каждом из представленных в книге модулей, но они — специальные для каждого модуля. Примеры функций `MakeCalc`, особенности их разработки и разнообразные возможные подходы к решению будут рассмотрены в следующих главах книги.

Альтернативное решение может заключаться в том, что верификация исходных данных будет производиться непосредственно в модуле обработки данной формы. Возможно, это немного облегчит функцию `MakeCalc`, но сделает громоздким модуль обработки формы. Решения, в принципе, равноценные. Однако хотелось бы предостеречь от одной ошибки. Если пользователь неправильно ввел или указал неверные исходные данные в предназначенных для этого полях, он должен иметь возможность покинуть данную форму в любом случае, выход из формы должен быть предусмотрен программистом. Возможно, пользователь захочет разобраться с правилами ввода данных вдали от компьютера, изучая печатное руководство пользователя. Он имеет право сделать это в удобное для него время. Выход из программы, используя комбинацию клавиш <Ctrl>+<Alt>+<Delete> — самое худшее из решений, свидетельствующее о низкой квалификации программиста, хотя у Д. Брэдли (David Bradley) были благие намерения, когда он придумал это магическое сочетание клавиш.

Ввод, верификация данных средствами VBA и обработка аварийных ситуаций

На листинге показан полный исходный текст типичной для наших проектов функции `MakeCalc`. Функция, показанная в примере, используется в разработке, посвященной проверке нормальности распределения и описанной в *главе 3*. Данная функция выполняет верификацию исходных данных, организует выполнение расчета и выводит результаты. Выдача диагностических сообщений и сообщений об ошибках также производится функцией `MakeCalc`. Исходные данные, представляющие собой содержимое ячеек рабочего листа, принимаются в функции `MakeCalc` как тип `Variant`. Остальные параметры ответственны за выбор метода расчета.

Листинг 1.10

```
Sub MakeCalc(Data As Variant, _
             Box1 As Boolean, Box2 As Boolean, _
             Box3 As Boolean, Box4 As Boolean, _
             Box5 As Boolean, Box6 As Boolean, _
```

```
            Box7 As Boolean, Box8 As Boolean, _
            Box9 As Boolean, Box10 As Boolean, _
            Alpha As String)
'
' Обозначения:
'  Data  - интервал выборки
'  Box1  - флаг использования критерия Колмогорова
'  Box2  - флаг использования критерия Крамера-Мизеса
'  Box3  - флаг использования критерия Андерсона-Дарлинга
'  Box4  - флаг использования критерия Шапиро-Уилка
'  Box5  - флаг использования критерия Эппса-Палли
'  Box6  - флаг использования критерия асимметрии
'  Box7  - флаг использования критерия эксцесса
'  Box8  - флаг использования критерия асимметрии Мардиа
'  Box9  - флаг использования критерия эксцесса Мардиа
'  Box10 - флаг использования критерия Хенце-Цирклера
'  Alpha - порог принятия решения
'
  Dim N As Integer          ' Численность выборки
  Dim K As Integer          ' Размерность выборки
  Dim Value As Variant      ' Рабочая переменная
  Dim Result As Double      ' P-значение
  Dim MsgMethod As String   ' Название метода
  Dim MsgResult As String   ' Результат анализа
  Dim ErrorTitle As String  ' Заголовок сообщения об ошибке
  Dim ErrorStyle As String ' Стиль окна сообщения об ошибке
  Dim MsgTitle As String    ' Заголовок сообщения об ошибке
  Dim MsgStyle As String    ' Стиль окна сообщения об ошибке
  Dim MsgNotDone As String  ' Сообщение об ошибке
  Dim MsgTooFew As String   ' Сообщение об ошибке
  Dim MsgTooMuch As String  ' Сообщение об ошибке
  ErrorTitle = "Ошибка в исходных данных"
  ErrorStyle = vbOKOnly + vbCritical
  MsgTitle = "Результат анализа"
  MsgStyle = vbOKOnly + vbInformation
  MsgNotDone = " Расчет не выполнен."
  MsgTooFew = "Мало данных для выбранного метода." & MsgNotDone
  MsgTooMuch = "Много данных для выбранного метода." & MsgNotDone
```

Анализ (верификация) выборок исходных данных, переданных из рабочего листа, помогает, в первую очередь, защитить программу от краха из-за неверно введенных данных. Тщательно выверенные числовые данные позволят избежать проблем при вычислении и последующего аварийного завершения программы.

Распространенные проблемы могут возникнуть из-за пустых ячеек (алгоритмы проверки нормальности распределения самостоятельно не могут их обрабатывать) и неверных типов данных, случайно отмеченных пользователем в качестве диапазона исходных данных. Для верификации исходных данных используются стандартные функции, примененные к переменной `Value`, в которой может храниться любой тип данных, извлеченный из переданного в функцию массива `Data` с помощью метода `Range`. Производится контроль:

1. данных, не относящихся к числовым типам. Представленные в книге расчетные алгоритмы, за исключением некоторых специальных случаев, работают только с числовыми (количественными или порядковыми) данными. Для тестирования каждого значения применяется стандартная функция `IsNumeric`;
2. пустых ячеек. Лучшим является такое решение, когда сам пользователь определяет, присваивать ли нулевое значение пустой ячейке. Если это так, нулевое значение должно быть введено явно. Для тестирования каждого значения применяется стандартная функция `IsEmpty`;
3. числовых данных, неприемлемых для применяемых алгоритмов. Например, данных может быть слишком мало или слишком много для того, чтобы они могли быть корректно обработаны применяемым алгоритмом. Для тестирования применяется метод сравнения полученного числового значения с некоторым допустимым значением, зависящим от конкретного применяемого алгоритма.

В главах, посвященных анализу данных методами, использующими сразу всю переданную матрицу исходных данных, входной контроль имеет особенности, отмеченные при описании соответствующих методов. Для некоторых алгоритмов требуется, помимо проверки данных, переданных из рабочего листа, осуществлять также контроль параметров, введенных пользователем в процессе выбора параметров расчета. Параметры должны быть допустимыми в применяемых алгоритмах.

Листинг 1.10 (продолжение)

```
'
' Анализ выборки
'
```

```
  On Error GoTo ErrHandler1
    Range (Data).Select
  On Error GoTo 0
  For Each Value In ActiveSheet.Range (Data)
    If IsEmpty (Value) Then
      MsgBox "Пустая ячейка в области данных." & MsgNotDone, ErrorStyle,
ErrorTitle
      Exit Sub
    End If
    If Not IsNumeric (Value) Then
      MsgBox "Нечисловой тип данных." & MsgNotDone, ErrorStyle,
ErrorTitle
      Exit Sub
    End If
  Next Value
  Set Value = Range (Data)
  N = Value.Rows.Count
  K = Value.Columns.Count
  If Box8 = True Or Box9 = True Or Box10 = True Then
    If K < 2 Then
      MsgBox "Мала размерность выборки." & MsgNotDone, ErrorStyle,
ErrorTitle
      Exit Sub
    End If
  Else
    N = N * K
  End If
  '
  ' Производство расчета
  '
  If Box1 = True Then
    Result = Application.Run("Kolmogorov", Range(Data))
    MsgMethod = "Критерий Колмогорова." & Chr(13) & "P = "
  ElseIf Box2 = True Then
    Result = Application.Run("CramerVonMises", Range (Data))
    MsgMethod = "Критерий Крамера-Мизеса." & Chr(13) & "P = "
  ElseIf Box3 = True Then
    Result = Application.Run("AndersonDarling", Range (Data))
    MsgMethod = "Критерий Андерсона-Дарлинга." & Chr(13) & "P = "
```

```
ElseIf Box4 = True Then
  If N < 3 Then
    MsgBox MsgTooFew, ErrorStyle, ErrorTitle
    Exit Sub
  End If
  If N > 50 Then
    MsgBox MsgTooMuch, ErrorStyle, ErrorTitle
    Exit Sub
  End If
  Result = Application.Run("ShapiroWilk", Range(Data))
  MsgMethod = "Критерий Шапиро-Уилка." & Chr(13) & "P = "
ElseIf Box5 = True Then
  If N < 4 Then
    MsgBox MsgTooFew, ErrorStyle, ErrorTitle
    Exit Sub
  End If
  Result = Application.Run("EppsPulley", Range (Data))
  MsgMethod = "Критерий Эппса-Палли." & Chr(13) & "P = "
ElseIf Box6 = True Then
  If N < 3 Then
    MsgBox MsgTooFew, ErrorStyle, ErrorTitle
    Exit Sub
  End If
  Result = Application.Run("SkewnessTest", Range (Data))
  MsgMethod = "Критерий асимметрии." & Chr(13) & "P = "
ElseIf Box7 = True Then
  If N < 4 Then
    MsgBox MsgTooFew, ErrorStyle, ErrorTitle
    Exit Sub
  End If
  Result = Application.Run("KurtosisTest", Range (Data))
  MsgMethod = "Критерий эксцесса." & Chr(13) & "P = "
ElseIf Box8 = True Then
  Result = Application.Run("MardiaKurtosisTest", Range (Data))
  MsgMethod = "Критерий асимметрии Мардиа." & Chr(13) & "P = "
ElseIf Box9 = True Then
  Result = Application.Run("MardiaSkewnessTest", Range (Data))
  MsgMethod = "Критерий эксцесса Мардиа." & Chr(13) & "P = "
```

```
  Else
    Result = Application.Run("HenzeZirkler", Range (Data))
    MsgMethod = "Критерий Хенце-Цирклера." & Chr(13) & "P = "
  End If
  '
  ' Формулирование вывода по результатам расчета
  '
  If Result >= CDbl(Alpha) Then
    MsgResult = Chr(13) & "Распределение нормальное."
  Else
    MsgResult = Chr(13) & "Распределение отличается от нормального."
  End If
  MsgBox MsgMethod & Result & MsgResult, MsgStyle, MsgTitle
  Exit Sub
```

Следующий фрагмент осуществляет выход из функции в распространенной аварийной ситуации — когда пользователь забыл выбрать или ввести интервал данных для анализа.

Листинг 1.10 (продолжение)

```
ErrHandler1:
  MsgBox "Не определен интервал переменной. Расчет невозможен.", _
         ErrorStyle, ErrorTitle
  Exit Sub
End Sub
```

Следующий фрагмент функции `MakeCalc` из *главы 9* показывает, как наиболее легким способом проверить правильность (т. е. соответствие требованием Microsoft Excel) формулы правой части дифференциального уравнения, введенной пользователем в ячейку рабочего листа. Наиболее простое решение — произвести фиктивный вызов функции. Если введенная формула неверна — выдается соответствующая диагностика, и выполнение программы завершается, под контролем программиста.

Листинг 1.11

```
  '
  ' Контроль правильности формулы
  '
  On Error GoTo ErrorHandler2
    Save = Right(X, Y, RangeX, RangeY, RangeRight)
  On Error GoTo 0
```

Особенность решения из *главы 9* также определяется повышенной трудоемкостью и, следовательно, продолжительным временем решения задачи интегрирования дифференциального уравнения. К тому же, скорость решения, как и сама возможность получения решения, зависит от введенных пользователем параметров. Если функция VBA начала выполняться, прервать ее выполнение, в принципе, можно нажатием комбинации клавиш <Ctrl>+<Break>. Происходит переход в то место кода VBA-функции, в котором ее выполнение было прервано пользователем. Выглядит неуклюже. Чтобы так не происходило, мы переопределим системную реакцию на нажатие пользователем клавиш Esc и <Ctrl>+<Break>, выдадим собственную диагностику и аккуратно завершим процесс вычислений. К тому же, такое переопределение не замедляет решение (не требуются ни вывод специального окна, ни проверка нажатия клавиши на каждом шаге решения), и без того трудоемкое. Как это сделать, показано на фрагменте листинга.

Листинг 1.12

```
On Error GoTo CancelHandler
  '
  ' Разрешить пользователю прервать решение
  '
  Application.EnableCancelKey = xlErrorHandler
  MsgBox "Решение уравнения может потребовать длительного времени." _
         & Chr(13) & _
         "Для прерывания вычислений Вы можете нажать клавишу Esc" _
         & Chr(13) & _
         "или комбинацию клавиш Ctrl-Break.", MsgStyleCancel, _
         MsgTitleCancel
  ActiveCell.FormulaR1C1 = _
                     "Решение уравнения (аргумент x, функция y=y(x))"
  If Method = 1 Then
    ActiveCell.Offset(1, 0).Select
    ActiveCell.FormulaR1C1 = "Метод Рунге-Кутта"
    For I = 1 To N
      Range (RangeOutput).Select
      ActiveCell.Offset(I + 1, 0).Select
      ActiveCell.FormulaR1C1 = X
      ActiveCell.Offset(0, 1).Select
      ActiveCell.FormulaR1C1 = Y
```

```
      ActiveCell.Offset(0, -1).Select
      Y = RungeKuttaFV(X, Y, X + Hout, Hbeg, Hmin, RangeX, RangeY, _
                       RangeRight)
    Next I
  ElseIf Method = 2 Then
    ActiveCell.Offset(1, 0).Select
    ActiveCell.FormulaR1C1 = "Метод Мерсона"
    For I = 1 To N
      Range (RangeOutput).Select
      ActiveCell.Offset(I + 1, 0).Select
      ActiveCell.FormulaR1C1 = X
      ActiveCell.Offset(0, 1).Select
      ActiveCell.FormulaR1C1 = Y
      ActiveCell.Offset(0, -1).Select
      Y = Merson(X, Y, X + Hout, Hbeg, Hmin, Epsilon, RangeX, RangeY, _
                 RangeRight)
    Next I
  ElseIf Method = 3 Then
    ActiveCell.Offset(1, 0).Select
    ActiveCell.FormulaR1C1 = "Метод Хаммера-Холлингсуорта"
    For I = 1 To N
      Range (RangeOutput).Select
      ActiveCell.Offset(I + 1, 0).Select
      ActiveCell.FormulaR1C1 = X
      ActiveCell.Offset(0, 1).Select
      ActiveCell.FormulaR1C1 = Y
      ActiveCell.Offset(0, -1).Select
      Y = HammerHollinsworth(X, Y, X + Hout, Hbeg, Hmin, Iter, Epsilon,
_
                             RangeX, RangeY, RangeRight)
    Next I
  ElseIf Method = | Then
    ActiveCell.Offset(1, 0).Select
    ActiveCell.FormulaR1C1 = "Метод Адамса"
    For I = 1 To N
      Range (RangeOutput).Select
      ActiveCell.Offset(I + 1, 0).Select
      ActiveCell.FormulaR1C1 = X
```

```
      ActiveCell.Offset(0, 1).Select
      ActiveCell.FormulaR1C1 = Y
      ActiveCell.Offset(0, -1).Select
      Y = Adams(X, Y, X + Hout, Node, RangeX, RangeY, RangeRight)
    Next I
  Else
    ActiveCell.Offset(1, 0).Select
    ActiveCell.FormulaR1C1 = "Метод Гира"
    For I = 1 To N
      Range (RangeOutput).Select
      ActiveCell.Offset(I + 1, 0).Select
      ActiveCell.FormulaR1C1 = X
      ActiveCell.Offset(0, 1).Select
      ActiveCell.FormulaR1C1 = Y
      ActiveCell.Offset(0, -1).Select
      Y = Gear(X, Y, X + Hout, Node, RangeX, RangeY, RangeRight)
    Next I
  End If
  '
  ' Вывод последней точки
  '
  Range (RangeOutput).Select
  ActiveCell.Offset(N + 2, 0).Select
  ActiveCell.FormulaR1C1 = X
  ActiveCell.Offset(0, 1).Select
  ActiveCell.FormulaR1C1 = Y
  ActiveCell.Offset(0, -1).Select
On Error GoTo 0
'
' Восстановление начальных значений
'
Range (RangeX).Select
ActiveCell.FormulaR1C1 = Xsave
Range (RangeY).Select
ActiveCell.FormulaR1C1 = Ysave
Exit Sub
```

На следующем листинге показан блок реакций на аварийные ситуации и фрагмент восстановления некоторых исходных значений. Полезный побочный эффект — перехватываются также сообщения о других ошибках выполнения. Например, пользователь не застрахован от таких ошибок при решении дифференциальных уравнений (см. *главу 9*) — для реальных задач нормальная ситуация. Блок оператора `If` служит для отделения реакции на действия пользователя, пожелавшего прервать решение нажатием клавиши <Esc> или комбинации <Ctrl>+<Break>, от алгоритмической ошибки периода выполнения. Таким образом, решение дифференциального уравнения может быть прервано только по двум причинам — по желанию пользователя или из-за ошибки в вычислениях (это могут быть: переполнение, потеря значимости, деление на нуль или выход параметров математической функции за пределы допустимых значений).

Листинг 1.13

```
ErrorHandler1:
  MsgBox "Не определена ячейка. Расчет невозможен.", MsgStyle, MsgTitle
  Exit Sub
ErrorHandler2:
  MsgBox "Неверно задана формула.", MsgStyle, MsgTitle
  Exit Sub
CancelHandler:
  Range (RangeX).Select
  ActiveCell.FormulaR1C1 = Xsave
  Range (RangeY).Select
  ActiveCell.FormulaR1C1 = Ysave
  If Err = 18 Then
    MsgBox "Решение прервано по желанию пользователя.", MsgStyle, _
           MsgTitleBreak
  Else
    MsgBox "Произошла ошибка в вычислениях.", MsgStyle, MsgTitleRun
  End If
End Sub
```

Все ошибки эффективно обрабатываются без краха прикладной программы или операционной системы. В некоторых случаях, ошибки вычислений могут быть интерпретированы как ограничения применимости конкретных методов анализа.

Рекомендуемая литература

1. Биллиг В.А. VBA в Office 2000. Офисное программирование. — М.: Издательско-торговый дом «Русская редакция», 1999.
2. Биллиг В.А. Средства разработки VBA-программиста. Офисное программирование. Том 1. — М.: Издательско-торговый дом «Русская редакция», 2001.
3. Бунин Э. Excel Visual Basic для приложений. — М.: Восточная книжная компания, 1996.
4. Орвис В.Дж. Visual Basic for Application на примерах. — М.: БИНОМ, 1995.

ГЛАВА 2

Разработка дополнительных модулей для Microsoft Excel

«Сколь бы широкое распространение ни получили за последнее время CASE–средства высокого уровня и графические инструменты, подобные Visual Basic, но для написания программ всегда будут нужны программисты, а серьезные программы надо писать на C++».
Г. Шилдт

«Программа на Visual Basic может загружать DLL, написанные на C++, Коболе, Фортране и др.»
Дж. Рихтер

Для работы с материалами главы понадобятся лицензионные средства разработки и тестирования программ:

- Borland C++ 5.5 или выше;
- Bloodshed Dev–C++ 4.01 или выше;
- Microsoft Excel 97 или выше;
- Microsoft Excel Developer's Kit версии 8 или выше;
- Microsoft Visual C++ 5.0 или выше.

Потребуется знание C, XLM, VBA.

Решаемая проблема

Для комплексного решения задачи надстройки какой-либо программы обычно используется встроенный язык программирования. Если программисту не хватает выразительных возможностей встроенного языка или требуется особая функциональность (например, повышение скорости обработки данных при использовании сложных вычислительных алгоритмов), внешние функции надстроек могут компилироваться в динамически связываемые библиотеки (Dynamic Link Library — DLL) и вызываться из программы, разработчик которой предусмотрел работу с DLL. Напомним, что надстройкой называется программа, служащая для добавления в основную программу специальных команд или возможностей. Средствами работы с надстройками обладают многие современные прикладные программы, а самой распространенной из них является табличный процессор (электронные таблицы) Microsoft Excel.

Разработка надстроек для Microsoft Excel

На представленной схеме показаны всевозможные варианты и технологии практической реализации надстроек Microsoft Excel. Обозначенные на схеме методики описаны в данной главе. Сплошными линиями на схеме показаны обязательные компоненты. Пунктиром показаны компоненты, которые могут использоваться или могут не использоваться в данной технологии.

Рис. 2.1. Реализация надстроек Microsoft Excel

Основные преимущества подхода для пользователей:

- аппаратные и системные требования разработанных программ (тип процессора, его быстродействие, емкость оперативной и дисковой памяти, требования к видеосистеме, тип операционной системы), как правило, совпадают с системными требованиями базовой программы;
- не приходится изучать интерфейс нового незнакомого программного продукта. Используется интерфейс всесторонне освоенного стандартного программного продукта;
- используются богатые графические возможности базовой программы: графическое меню, построение стандартных графиков, хотя некоторые важные типы графиков, к сожалению, доступны только в специализированных программах анализа данных;
- не нужно заботиться о целостном, без потерь, переносе пользовательских данных, возможно, накопленных в течение многих лет в базах данных других распространенных типов (MS Access, dBase, Paradox) и электронных таблицах (1–2–3, Quattro Pro), под управлением различных операционных систем, в специализированную программу анализа данных.

Это очень сильные аргументы в пользу рассматриваемого типа программного обеспечения. Рассмотрим основные преимущества подхода для программистов и менеджеров программных проектов.

- При правильном использовании ресурсов компьютера, стабильность разработанного программного обеспечения не хуже, чем у базовых программных продуктов.
- Заботу о манипуляциях с данными (ввод и вывод, преобразования, экспорт и импорт данных в стандартные форматы, межпрограммные коммуникации в рамках возможностей операционной системы) берет на себя базовая программа.
- Программист получает возможность сосредоточить основное внимание не на рутинных, но важных операциях создания пользовательского интерфейса, заготовка которого формируется визуально, а на оригинальных методах. Программист освобождается от основной части труда по созданию каркаса программы и получает больше времени для размышлений и анализа.
- Мощная вычислительная программа становится реализуемой в минимальные сроки. Большое трудолюбие, при отсутствии плодотворных идей, вместо пользы приносит вред. Оригинальный девиз книги «Less means, more power» можно перевести как «Меньше затрат, больше возможностей». Он наиболее точно отражает рассматриваемую технологию решения.

- Снижается потребность в большом количестве программистов низкой и средней квалификации, занятых разработкой интерфейса и утилит. Необоснованное увеличение количества исполнителей увеличивает сложность межличностных коммуникаций, количество внутренней документации и усложняет тестирование версий.

Существует два подхода к разработке надстроек Microsoft Excel, в зависимости от способа регистрации функций DLL в электронных таблицах. Это add-ins (дополнительные модули, надстройки), при разработке которых может применяться язык макросов Microsoft Excel, сокращенно называемый XLM, или язык C, и стандартные DLL. Применяются и комбинации решений, наиболее эффективные в конкретном проекте.

Программные средства разработки DLL для электронных таблиц Microsoft Excel представлены в таблице. Системы разработки приложений на языках C/C++ поставляются в номенклатуре, обеспечивающей программным инструментарием пользователя с любыми финансовыми возможностями, поэтому существенное влияние на выбор пользователя окажет колонка «Условия лицензирования».

***Таблица 2.1**. Средства разработки DLL для Microsoft Excel*

Система программиро-вания	**Условия лицензирования**	**Наличие IDE**	**Локализация**	**Область применения***
Microsoft Visual C++ версий 5.0, 6.0	Коммерческая программа	Да	Нет	Проекты в главах 3, 5, 6, 7, 8, 9, 10
Borland C++ версии 5.5	Бесплатная версия коммерческой программы	Нет**	Нет	Проекты в главах 3, 5, 6, 7, 8, 9
Digital Mars C++***	Бесплатная версия коммерческой программы	Нет**	Нет	Проекты в главах 3, 5, 6, 7, 8, 9
Bloodshed Dev–C++ версии 4.01	Свободно распространяемая**** программа	Да	Нет	Проекты в главах 3, 5, 9
Bloodshed Dev–C++ версии 5	Свободно распространяемая**** программа	Да	Да	Проекты в главах 3, 5, 9

Примечания: * Указанная область применения зависит от технических возможностей системы программирования. ** Бесплатно поставляются IDE разработки третьих фирм. *** По информации прессы и документации. Автором книги данная система не тестировалась. **** «Свободное» программное обеспечение не тождественно бесплатному ПО и так называемому общественному достоянию (public domain).

Объясним, почему для наших проектов мы выбрали именно это инструменты. Итак:

- Microsoft Visual C++ — единственная «официальная» система программирования для всех версий Microsoft Windows;
- Borland C++ и Digital Mars C++ — альтернативные решения, способные для поставленных нами задач заменить Microsoft Visual C++;
- Bloodshed Dev–C++ — доступная локализованная (в версии 5 и выше) свободно распространяемая система программирования, основанная на передовых решениях в области разработки компиляторов.

Существуют и другие системы программирования на C/C++. Они уступают названным системам по ряду параметров. Например, некоторые бесплатные версии Borland C++ и Digital Mars C++ не оснащены IDE, а работают только из командной строки. Другие системы программирования, такие, как Open Watcom C/C++, в текущей комплектации поставки оказались непригодны для создания DLL.

Система программирования Microsoft Visual C++ подробно описана в литературе. Вместе с Borland C++ и Borland Delphi, она — одна из самых «модных» в изданиях на русском языке. Для справки по технической информации можно использовать библиотеку MSDN (**msdn.microsoft.com**). Структура представления технической информации в справочных системах Microsoft Visual C++, Microsoft Excel и VBA достаточно удобна, хотя некоторые авторы и разработчики полагают, что она излишне фрагментарна, а примеры программного кода хотя и содержательны, но не всегда полны, а потому совершенно бесполезны. Справочные системы упомянутых выше систем разработки приложений Borland C++, Digital Mars C++, Open Watcom C/C++ или Bloodshed Dev-C++ также могут оказать большую помощь в получении необходимой информации.

Кратко рассмотрим основные моменты программирования DLL на C/C++ для Microsoft Excel. Конкретные проекты описаны в следующих главах, а здесь мы рассмотрим только общие вопросы программирования и особенности использования DLL для Microsoft Excel.

Все применяемые системы программирования представлены в обзоре (*приложение 4*), включая ссылки на ресурсы Интернет.

Разработка DLL в Microsoft Visual C++

Система разработки приложений Microsoft Visual C++ (версии 5.0 и выше, более ранние версии нами не тестировались), должна быть предварительно установлена на компьютере разработчика. Для работы с рассматриваемыми в книге примерами конфигурация установки может быть выбрана минимальной. Не потребуются компоненты: встроенная справочная система, электронные книги, поддержка баз данных, библиотека MFC и другие.

Для создания проекта DLL запустите Microsoft Visual C++. Выберите из меню **File ▸ New ▸ Projects ▸ Win32 Dynamic-Link Library**. Затем выберите из меню **Project ▸ Add To Project ▸ New ▸ Files ▸ C++ Source File** и введите исходный код (текст) программы, как показано на рисунке 2.2.

Рис. 2.2. Окно Microsoft Visual C++ с введенными исходными текстами DLL

Возможно, понадобятся и другие файлы, как в проекте на рис. 2.2.

Разработка DLL в Bloodshed Dev-C++

Система разработки приложений Bloodshed Dev-C++ версии 4.01 или выше должна быть предварительно установлена на компьютере. На сайте **www.bloodshed.net** помещена последняя версия данного программного обеспечения.

Если вы решили воспользоваться Bloodshed Dev-C++ для создания проекта DLL, запустите его. Выберите из меню **File ▸ New Project ▸ DLL ▸ OK**. Укажите необходимые характеристики проекта в форме **New DLL project** и нажмите кнопку **OK**. Затем отредактируйте исходный текст заготовки программы, предлагаемой Bloodshed Dev-C++, как показано на рисунке 2.3.

Рис. 2.3. Окно локализованной среды Bloodshed Dev-C++ предварительной версии 5 с введенными исходными текстами DLL

Разработка DLL в Borland C++

Описанный в предыдущем разделе компилятор Bloodshed Dev-C++, из–за особенностей реализации DEF-файла, не удалось применить для проектов из *глав 6, 7* и *8*. Для читателей, которые не имеют возможности или желания приобрести Microsoft Visual C++, для компиляции проектов, представленных в указанных главах книги, можно порекомендовать загрузить и установить бесплатный компилятор Borland C++ версии 5.5. Данный компилятор работает только из командной строки, однако разработка стандартных DLL с использованием Borland C++ 5.5 не вызвала каких-либо трудностей при соблюдении некоторых условий:

В файл autoexec.bat, находящийся в корневом каталоге загрузочного диска компьютера, следует добавить путь к исполняемым файлам Borland C++ 5.5. Например, на нашем компьютере Borland C++ 5.5 установлен на диске C: в каталоге \Borland\Bcc55, поэтому файл autoexec.bat имеет вид, как на листинге 2.1:

Листинг 2.1

```
@echo off

path %path%;c:\Borland\Bcc55\bin

mode con codepage prepare=((866) C:\WINDOWS\COMMAND\ega3.cpi)

mode con codepage select=866

keyb ru,,C:\WINDOWS\COMMAND\keybrd3.sys
```

В каталоге проекта нужно создать файл конфигурации, показанный на листинге 2.2 и названный bcc32.cfg, содержащий опции компилятора (назначение ключей компилятора подробно описано в справочной системе Borland C++ 5.5; ключи на листинге 2.2 — это пути к соответствующим служебным каталогам include и lib, а также указание компилировать DLL для Microsoft Windows).

Листинг 2.2

```
-I"c:\Borland\Bcc55\include"

-L"c:\Borland\Bcc55\lib"

-WD
```

Предполагается, что компиляция и сборка производятся одновременно. Если требуется выполнить только компиляцию, а сборку планируется осуществить позднее, можно использовать соответствующий ключ в опциях компилятора. Тогда после компиляции для сборки DLL придется воспользоваться вызовом линкера ilink32.exe, входящего в комплект поставки Borland C++ 5.5. Назначение ключей программы ilink32.exe и содержимое файла конфигурации ilink32.cfg подробно описаны в справочной системе, прилагаемой к пакету Borland C++ 5.5, и в файле readme.txt.

В каталоге проекта должен находиться DEF-файл, содержащий секцию EXPORTS, в которой следует перечислить неискаженные имена всех экспортируемых функций. Рекомендуется, чтобы DEF-файл имел такое же имя(source.def), что и файл с исходными кодами(source.cpp). В этом случае, не нужно будет явно указывать его имя.

После выполнения указанных действий, для создания файла DLL следует только ввести в командной строке bcc32 source.cpp. Из исходного файла source.cpp и DEF-файла source.def будет получена DLL по имени source.dll.

Для ввода исходного текста программы удобно использовать специализированную программу-редактор, поддерживающую выделение цветом син-

таксических конструкций применяемого языка программирования. Рекомендуем редактор Crimson Editor, разработанный И. Кангом (Ingyu Kang) на **www.crimsoneditor.com**. Окно **Crimson Editor** с введенными исходными текстами на C++ показано на рисунке 2.4.

Рис. 2.4. Окно редактора Crimson Editor с введенным текстом на языке C++

Очевидно, возможно создать графическую оболочку, обеспечивающую выбор ключей компилятора и ввод исходных текстов, запускающую процесс создания программы и выдающую результаты в текстовый файл (например, путем переадресации вывода), который затем будет показан в окне выдачи результатов. Данная технология давно и с успехом применяется в различных отраслях программирования. Она получила собственное наименование Front-end (передняя панель). Программа, оснащенная графическим интерфейсом, не выполняет команды самостоятельно, а передает их консольной программе. Результаты, возвращаемые в консоли, принимаются графической программой и выводятся на экран.

На рисунке 2.5 изображено окно доступного на **www.jmcreations.ca** текстового редактора ProEdit 2000, разработанного Дж. Маркесом (Joe Marques). Данный текстовый редактор фактически представляет собой несложную и удобную оболочку для Borland C++ 5.5.

Рис. 2.5. Окно редактора ProEdit 2000 с введенным текстом на языке C++

Дополнительные модули

Дополнительные модули могут быть выполнены двумя способами. Могут применяться: язык XLM совместно с DLL на языках C/C++ либо так называемые stand–alone DLL (XLL), разрабатываемые на языке C с помощью инструментария Microsoft Excel C API. Возможны и различные комбинированные решения. Необходимость всех этих решений вызвана тем, что высокой скорости выполнения алгоритмов, использующих интенсивные вычисления, в Microsoft Excel нельзя добиться иначе как использованием скомпилированных модулей на языке C.

Функции, экспортируемые DLL в стиле Excel 4, нельзя зарегистрировать оператором `Declare` в модулях VBA электронных таблиц Microsoft Excel и в проектах на языке Microsoft Visual Basic. Регистрировать функции, содержащиеся в данной DLL, придется с помощью функции (макроса) `РЕГИСТРАТОР` языка XLM на листе макросов Microsoft Excel или специальными функциями на языке C (это и есть так называемый стиль Excel 4).

Однако, после аккуратной регистрации функций, они, во-первых, для своей работы в минимальной конфигурации не нуждаются в каком-то дополнительном VBA-коде, а, во-вторых, с равным успехом вызываются и из рабочего листа электронных таблиц Microsoft Excel, и из модулей VBA.

Microsoft Excel C API и XLL

Разработка модулей, рассмотренных в данном разделе, ведется с помощью системы программирования Microsoft Visual C++. Инструментарий Microsoft Excel C API, доступный на сайте **msdn.microsoft.com**, предназначен для создания stand-alone DLL, представляющих собой особые DLL, переименованные после окончания компиляции в файлы с расширением XLL.

Хотя технология XLL считается устаревшей, она поддерживается во всех новых версиях Microsoft Excel, а модули XLL активно используются многими разработчиками. Их немалым достоинством является более надежное, в отличие от модулей VBA, сокрытие программного кода, а также высокая скорость выполнения содержащихся в них функций. Наряду с достоинствами, разработчик указал и на неудобства инструментария C API:

- C API оптимизирован для использования вместо модулей XLM, это не лучший механизм для совместного использования с VBA;
- C API не является объектно-ориентированным, т. к. базируется на XLM; нельзя использовать C API для доступа к объектам, свойствам и методам VBA;
- внешние приложения не могут использовать функции C API (но не пользовательские функции, содержащиеся в XLL); нельзя, да в этом и нет необходимости, вызывать их из рабочего листа Microsoft Excel.

Рассмотрим программирование XLL более подробно. Итак, XLL включают, кроме пользовательских, еще ряд функций, обеспечивающих регистрацию функций DLL. Данные функции включены в инструментарий Microsoft Excel C API в составе, как минимум, следующих файлов:

- Framewrk.cpp — библиотека разработчика в исходных кодах,
- Framewrk.h — заголовочный файл для библиотеки разработчика,
- Xlcall.h — заголовочный файл для специальных функций Microsoft Excel,
- Xlcall32.lib — библиотека специальных функций Microsoft Excel.

Файлы Framewrk.cpp и Xlcall32.lib должны быть включены в проект. В проект включается также пользовательский модуль (файл с расширением .cpp), содержащий составленные нами функции, и стандартный файл определе-

ний (с расширением .def), в котором перечисляются имена экспортируемых пользовательских функций.

Пользовательский модуль обязательно должен включать в себя вызовы функций:

- `xlAutoOpen` — функция инициализации, в которой производится вызов специальной функции `Excel ()` или `Excel4 ()`, содержащихся в библиотеке Xlcall32.lib и обеспечивающих регистрацию пользовательских функций, это аналоги макроса `РЕГИСТРАТОР`, а параметры функции `Excel ()` и, в большей степени, `Excel4 ()`, почти аналогичны параметрам макроса `РЕГИСТРАТОР`;
- `xlAutoClose` — функция закрытия, это аналог макроса `ВОЗВРАТ`.

Пользовательский модуль может включать функцию `xlAddInManagerInfo`, для вывода информации о событиях, происходящих при регистрации и выполнении пользовательских функций, и другие функции, применение которых не является обязательным. Функция `xlAddInManagerInfo` удобна при отладке взаимодействия Microsoft Excel и XLL.

В проектах DLL на языках C или C++, экспортируемые из DLL функции при использовании компилятора Microsoft Visual C++ не должны иметь никаких модификаторов.

Практическое решение с использованием C API приводится в *главе 10*, а фрагмент возможной реализации перечисленных выше обязательных функций показан на листинге 2.3.

Листинг 2.3

```
#include <windows.h>
#include <math.h>

#include "xlcall.h"
#include "framewrk.h"

//////////////////////////////////
// Прототипы экспортируемых функций
double Bernoulli (long int,double);
double ChebyshevHermite (long int,double);
double Hermite (long int,double);
double Laguerre (long int,long int,double);
double Legendre (long int,double);
```

```
//////////////////////////////////
// Прототипы "внутренних" функций
double Combination (long int,long int);
double LogFactorial (long int);

//////////////////////////////
// Таблица регистрации функций
// содержит для каждой функции:
// Имя процедуры, Типы параметров, Имя функции, Список аргументов,
// Тип макроса, Имя категории, "Горячая клавиша", Тема помощи,
// Функция помощи
//
static LPSTR func[][9] =
{
 {" Bernoulli"," BJB"," Bernoulli"," "," 1"," Многочлены"," "," "," "},
 {" ChebyshevHermite"," BJB"," ChebyshevHermite"," "," 1"," Многочлены"," 
"," "," "},
 {" Hermite"," BJB"," Hermite"," "," 1"," Многочлены"," "," "," "},
 {" Laguerre"," BJJB"," Laguerre"," "," 1"," Многочлены"," "," "," "},
 {" Legendre"," BJB"," Legendre"," "," 1"," Многочлены"," "," "," "},
 {0,0,0,0,0,0,0}
};

BOOL __stdcall xlAutoOpen (void)
//
// Функция инициализации XLL
//
{
 int i,              // Счетчик
     nFuncs;         // Число экпотрируемых функций
 static XLOPER xDll; // Имя файла XLL

 ////////////////////////
 // Получить имя файла XLL
 Excel (xlGetName, &xDll, 0);

 for(nFuncs = 0; func[nFuncs][0]; nFuncs++)
   for(i = 0; i < 9; i++)
     func[nFuncs][i][0] = (BYTE) strlen (func[nFuncs][i]+1);
```

```
 //////////////////////
 // Регистрация функций
 for(i = 0; i < nFuncs; i++)
   Excel (xlfRegister, 0, 9, (LPXLOPER)&xDll,
         (LPXLOPER)TempStr(func[i][0]),(LPXLOPER)TempStr(func[i][1]),
         (LPXLOPER)TempStr(func[i][2]),(LPXLOPER)TempStr(func[i][3]),
         (LPXLOPER)TempStr(func[i][4]),(LPXLOPER)TempStr(func[i][5]),
         (LPXLOPER)TempStr(func[i][6]),(LPXLOPER)TempStr(func[i][7]),
         (LPXLOPER)TempStr(func[i][8]));

 ///////////////////////////
 // Освободить имя файла XLL
 Excel (xlFree, 0, 1, (LPXLOPER)&xDll);

 return 1;
}

BOOL __stdcall xlAutoClose (void)
//
// Функция завершения XLL
//
{
 return 1;
}
```

Язык макросов Microsoft Excel

В качестве альтернативы рассмотренному в предыдущем разделе, для регистрации функций, содержащихся в разработанных DLL, может применяться язык макросов Microsoft Excel.

Все применяемые нами средства разработки могут создавать DLL описываемого в настоящем разделе класса.

В проектах DLL на языках C или C++ экспортируемые DLL функции должны иметь модификаторы:

- `__declspec(dllexport)` для каждой экспортируемой из DLL функции, необходим, если не используется DEF-файл;
- `extern "C"` означает, что функция написана на языке C, и ее имя нельзя использовать как имя C++, модификатор нужен только в модулях на языке C++, но не C.

Полная техническая информация о регистрации функций содержится в справочных материалах под названием Microsoft Excel 97 Developer's Kit на компакт-диске или в сети Интернет на сайте **msdn.microsoft.com**, а также на **www.microsoft.com,** в файле «Справки по макрофункциям для Microsoft Excel 97». Поэтому на описании всех параметров макроса `РЕГИСТРАТОР` останавливаться не будем, а отметим только необходимые для решения наших задач. В *главах 3* и *5* содержатся подробные работающие примеры.

Данные из Microsoft Excel в функцию C/C++ передаются через указатель на структуру `FP`. Первыми членами структуры являются число строк и число столбцов передаваемой области. Затем следует массив типа `double*`, как показано на листинге 2.4.

Листинг 2.4

```
typedef struct
{
 unsigned short int rows;
 unsigned short int columns;
 double Array[1];
} FP;
```

Число элементов массива равно числу строк, умноженному на число столбцов. На листинге 2.5 показан примерный вид листа макросов Microsoft Excel. Первый параметр функции — имя (без расширения) файла DLL, экспортирующей функцию. О местоположении данного файла на диске компьютера мы поговорим чуть позднее. Второй параметр — имя функции в том виде, как оно экспортируется DLL. Третий параметр функции, `BKK`, означает: `B` — возвращаемое функцией значение имеет тип `double`, `KK` — импортируемая функция имеет два указателя на структуру `FP` (см. листинг 2.3). В наших проектах упоминается также тип `J`, соответствующий `longint` в функциях C/C++. Несогласованность типов формальных параметров, указанных в макросе `РЕГИСТРАТОР`, с фактическими параметрами функции C/C++ может повлечь тяжелые последствия. Отметим, что перечисленные типы параметров — почти те же самые, что при вызове функций `Excel` или `Excel4`, упомянутых в предыдущем разделе.

Листинг 2.5

```
=РЕГИСТРАТОР("EX_dll";"WilcoxonTest";"BKK";"WilcoxonTest";"Массив; Мас-
сив";1;"Непараметрическая статистика";;;"W-критерий Вилкоксона для неза-
висимых выборок";"Первая выборка";"Вторая выборка";)
=ВОЗВРАТ()
```

Помимо собственно регистрации функции DLL, макрос РЕГИСТРАТОР сообщает необходимую информацию о содержании подсказок пользователю. Макрос РЕГИСТРАТОР вводится для каждой импортируемой из DLL функции. Заканчивается макрос вызовом стандартной XLM- функции ВОЗВРАТ.

Данный макрос необходимо выполнить, после чего зарегистрированные функции будут доступны для выполнения. Проще всего это сделать так. Находясь в листе макросов (курсор на первой ячейке, содержащей вызов функции РЕГИСТРАТОР) с вызовами функций РЕГИСТРАТОР и обязательным вызовом функции ВОЗВРАТ, выберите из меню Microsoft Excel команду **Вставка ▸ Имя ▸ Присвоить**. В появившемся диалоговом окне в поле **Имя** введите какое–либо уникальное имя, например, RegDLL, а в поле **Макрос** отметьте пункт **Функция**. Остальные параметры не меняйте и нажмите кнопку **OK**, как показано на рисунке 2.6.

Рис. 2.6. Создание макроса регистрации функций DLL в Microsoft Excel 2000

В результате описанных действий, в категории «Определенные пользователем» появится новая функция (макрос) RegDLL. Она должна быть выполнена до вызова функций, регистрируемых данным макросом. Если макрос RegDLL не выполнить, регистрируемые функции будут видны в указанной категории, например «Непараметрическая статистика» (см. *главу 3*), но выполнить их будет нельзя.

Макрос RegDLL может быть выполнен различными способами:

- находясь на листе макросов (курсор на первой ячейке), нажать правую кнопку мыши, из появившегося меню выбрать пункт **Выполнить** и нажать кнопку **Выполнить;**

- вызвать функцию RegDLL из любой ячейки рабочего листа путем выбора из меню Microsoft Excel пункта **Вставка ▸ Функция**;
- выполнить `RegDLL` автоматически при открытии рабочей книги или шаблона, в произвольном месте функции `Auto_Open`, но до вызова функций, зарегистрированных макросом `RegDLL`. Функция `Auto_Open`, если она имеется в модуле VBA, всегда выполняется автоматически при открытии содержащей ее рабочей книги или шаблона.

Здесь и далее пользовательскими мы будем называть все самостоятельно разработанные нами компоненты – остальные средства Microsoft Excel, входящие в комплект поставки данных электронных таблиц, мы будем называть стандартными. К стандартным мы не относим компоненты, не входящие в комплект поставки Microsoft Excel, их список дан во введении к данной книге.

В качестве последнего штриха, для сокрытия листа макросов, находясь в нем, выберите из меню Microsoft Excel команду **Формат ▸ Лист ▸ Скрыть**. Пользователю совсем необязательно навязывать лишние знания о подробностях нашей работы, а получить информацию о них он сможет, выполнив команду **Формат ▸ Лист ▸ Отобразить**.

Вызов зарегистрированной функции `RandomIndependent` (см. *главу 3*) из VBA-функции в модуле VBA показан на листинге 2.6. Функция `RandomIndependent` вызывается с помощью метода `Application.Run`.

Листинг 2.6

```
Result = Application.Run("RandomIndependent", Range(Data1), Range(Data2))
```

Стандартные DLL

Стандартные DLL могут быть использованы в проектах как на языке VBA, так и на алгоритмическом языке Visual Basic, однако функции стандартных DLL нельзя напрямую вызвать из рабочего листа или из листа макросов Microsoft Excel (для этого можно составить специальные интерфейсные функции). Регистрация функций стандартных DLL производится оператором `Declare`.

В наших примерах стандартные DLL создаются с помощью компиляторов Microsoft Visual C++ версии 5.0 и выше или Borland C++ 5.5. Для разработки стандартных DLL в языках C или C++ необходимо использовать:

- модификатор `__stdcall` для каждой экспортируемой функции (или WINAPI в Microsoft Visual C++);

- `extern "C"` для Borland C++ означает, что функция написана на языке C, и ее имя нельзя использовать как имя C++, модификатор нужен только в модулях на языке C++, но не C;

- DEF-файл, содержащий секцию `EXPORTS`, в которой следует перечислить неискаженные имена всех экспортируемых функций. Это — один из способов компенсации искажений имен экспортируемых DLL. Для Microsoft Visual C++ DEF-файл следует включить в проект. В Borland C++ рекомендуется, чтобы DEF-файл имел то же имя, что и файл с исходными кодами, и находился в том же каталоге проекта. В этом случае его не нужно будет явно указывать.

Возможность использования в проекте DEF-файла — это очень удобное свойство. Отсутствие данной возможности в Bloodshed Dev–C++ создает проблемы для разработки стандартных DLL. Дело в том, что Bloodshed Dev-C++ не позволяет включать в проект пользовательский DEF-файл по усмотрению программиста, а создает DEF-файл только для целей информирования об именах экспортированных функций. О возможностях и проблемах создания DLL средствами компилятора MinGW см. раздел «Частые вопросы и ответы» на сайте **www.mingw.org**.

Как отмечено выше, регистрация функций DLL в программах (функциях) VBA/VB производится с помощью оператора `Declare`. О существовании данной возможности узнают с удивлением даже VBA-программисты со стажем. Ниже рассмотрены основные особенности данного типа регистрации.

Передача скалярных типов данных из программы (функции) VBA/VB в функцию C/C++ подробно описана в документации, и рассматривать ее здесь нет необходимости. Однако в нетривиальной расчетной программе, помимо передачи скаляров, требуется передача массивов из VBA/VB в C/C++ и обратно. На самом деле, нет передачи «туда и обратно». Функции VBA и функции DLL в наших проектах работают с общими областями памяти.

Массив из функции C/C++ в функцию C/C++ передается указателем на его первый элемент. В совместном использовании массивов, при вызове из VBA/VB программы нужно передать в функцию C/C++ ссылку на скаляр (`ByRef`), указав первый элемент массива VBA/VB. Поэтому в VBA-операторах `Declare` вместо массивов, формальных параметров функций, созданных в C/C++, указываются скалярные величины по ссылке, как показано на листинге 2.7. Для примера показана регистрация функции King, содержащейся в динамически связываемой библиотеке Cla.dll (см. *главу 7*). Назовем данный способ передачи массивов в VBA *передачей в стиле C/C++*.

Листинг 2.7

```
Private Declare Function King Lib "Cla.dll" (ByRef Z As Double,_
ByVal K As Integer, ByVal M As Integer, ByVal L As Integer, _
ByRef PH As Integer, ByRef PL As Integer, ByRef S As Double) _
As Integer
```

Чтобы предотвратить случайный вызов зарегистрированной функции `King` из рабочего листа, в VBA она описана как `Private`. Случайный вызов неминуемо приведет к фатальной ошибке, т. к. из рабочего листа Microsoft Excel передаются не стандартные VBA-массивы, а структуры, показанные на листинге 2.4. Вызов можно осуществить с помощью несложной специально составленной интерфейсной функции VBA/VB.

Вызов зарегистрированной функции `King` из VBA-функции в модуле VBA показан на листинге 2.8. Хорошо видно, что функция King вызывается точно так же, как пользовательская или стандартная функция VBA. Другие зарегистрированные функции вызываются аналогично.

Листинг 2.8

```
Result = King (X(1), N, K, Mode, Y1(1), Y2(1), S(1))
```

Первый и три последних параметра в операторе вызова функции `King` являются массивами, передаваемыми из функции VBA через ссылку на первый элемент массива.

Функцией `King` возвращается код ошибки `Result`.

Существуют стандартные рекомендации по числовым значениям кодов ошибки. В рамках данной книги, при успешном завершении функции, код ошибки всегда нулевой. Исключение составляют функции с фиксированной областью определения параметров. Допустимость параметров следует проверять до вызова функции, для значений, отличных от нуля, нами зарезервирована только величина «−1», означающая недостаточность памяти для динамических массивов. Если функция не возвращает значения, то это означает, что ошибки при выполнении функции не ожидаются.

О типе данных Variant

Этот универсальный тип данных удобен, и даже незаменим в некоторых ситуациях, однако программисту вычислительной программы полезно знать тип каждого объекта в программе. Широкое применение тип `Variant` находит при передаче данных из ячеек электронной таблицы в функ-

цию VBA. Использование типа `Variant`, особенно при передаче переменных или массивов этого типа в модули, составленные на других языках программирования, чревато стабильным возникновением некоторых специфических ошибок и неэкономным расходованием ресурсов компьютера. Предложенная нами схема передачи массивов в стиле C/C++, т.е. через ссылку на скаляр `ByRef`, нас ни разу еще не подвела, чего не скажешь о некоторых решениях с использованием типа `Variant`. Лучше обеспечить для электронных таблиц Microsoft Excel надежную работу, чем аварийно завершать их выполнение.

Организация работы Microsoft Excel с программами и пользовательскими функциями

Нами составлены все необходимые компоненты: рабочие книги с листами макросов и модулями VBA, некоторые функции при необходимости помещены в DLL. В данном параграфе мы рассмотрим, как организовать работу со всеми этими компонентами, какие использовать каталоги для их размещения и как осуществлять запуск функций надстроек, чтобы работа с ними была максимально удобна и похожа на работу со стандартными средствами Microsoft Excel.

Организация работы Microsoft Excel с DLL, XLL и модулями VBA

Файл DLL в конкретном проекте скопируем в системный каталог ОС Microsoft Windows. В зависимости от используемой версии Microsoft Windows, системными считаются каталоги:

- C:\Windows\System и C:\Windows\System32 для Windows 95/98/Me/XP;
- C:\WINNT\System32 для Windows NT/2000.

В одном из этих каталогов, по умолчанию, может находиться файл DLL.

Можно поместить файл DLL в другой каталог. При этом только придется позаботиться о том, чтобы макрос или функция VBA смогла его найти. Это решение позволит избежать возможного конфликта имен файлов DLL и будет соответствовать рекомендации разработчика Microsoft Windows избегать помещения в системные каталоги пользовательских DLL.

Напомним, что расширения файлов Microsoft Excel по умолчанию: xls — рабочая книга, xlt — шаблон, xla — надстройка.

Шаблон, по умолчанию, сохраняется в каталоге C:\WINDOWS\Application Data\Microsoft\Шаблоны. Рисунок 2.7 иллюстрирует возможность создания

документа из шаблона Inform.xlt (см. *главу 4*) после выбора из меню **Файл ▸ Создать**. При выборе шаблона Inform будет создана новая рабочая книга, причем будет выполнен программный код листа макросов или модуля VBA, содержащийся в данном шаблоне. Программный код, содержащийся в шаблоне, будет скопирован в данную рабочую книгу.

Рис. 2.7. Окно создания нового документа

Основной недостаток рассмотренного способа — пользователь при каждом открытии рабочей книги или шаблона будет получать предупреждение, подобное показанному на рисунке 2.8. Для запуска программы, пользователю необходимо выбрать опцию **Не отключать макросы**. Многим пользователям после ознакомления с текстом грозного предупреждения требуются дополнительные объяснения, почему следует сделать именно так, а не иначе. На самом деле ничего сложного в этом нет, но выглядит такой запуск программы неуклюже.

Рис. 2.8. Окно предупреждения об опасности макросов Microsoft Excel 2000

Одно из решений — сохранить рабочую книгу как надстройку путем выбора **Файл ▸ Сохранить как ▸ Тип файла: Надстройка Microsoft Excel ▸ Сохранить**. Это — лучшее из рассмотренных решений, пользователь получает чис-

тую рабочую книгу и функциональность установленных в системе надстроек. Чтобы надстройка автоматически открывалась при каждом запуске Microsoft Excel, ее следует поместить в стартовый каталог Microsoft Excel, например:

- c:\Program Files\Microsoft Office\Office\XLStart для Microsoft Excel 2000 — компонента Microsoft Office 2000;
- c:\Program Files\Microsoft Office\Office10\XLStart для Microsoft Excel 2002 — компонента Microsoft Office XP;
- c:\WINDOWS\Application Data\Microsoft\Excel\XLSTART для ОС Microsoft Windows 95/98/Me;
- c:\WINNT\Application Data\Microsoft\Excel\XLSTART для ОС Microsoft Windows NT/2000.

Возможны также варианты сохранения в каталогах надстроек:

- c:\Windows\Application Data\Microsoft\AddIns для ОС Microsoft Windows 95/98/Me;
- c:\WINNT\Application Data\Microsoft\AddIns для ОС Microsoft Windows NT/2000;
- c:\Documents and Settings\[Имя пользователя]\Application Data\Microsoft\AddIns для ОС Windows XP.

В последних трех случаях, при запуске Microsoft Excel после инсталляции надстройки, необходимо выбрать из меню Microsoft Excel пункт **Сервис ▸ Надстройки** и отметить появившуюся в меню. Этот способ работы с надстройками XLL, по нашему мнению, менее удобен, чем предыдущие, т. к. при удалении надстройки пользователю придется также вручную деактивировать ее. Надстройка может также быть сохранена в произвольном каталоге, а при активации найдена стандартным образом через кнопку **Обзор**, как показано на рисунке 2.9.

Рис. 2.9. Активация надстройки Microsoft Excel 2002, содержащейся в произвольном каталоге

Мы изучили, в какие каталоги на компьютере пользователя следует помещать файлы DLL и файлы надстроек XLA и XLL. Другие файлы, например, файлы справочной системы, различные текстовые файлы с информацией о программном продукте, можно помещать в произвольные каталоги, учитывая это при обращении к ним из надстройки или из меню Windows. Удобным является помещение файлов Справочной системы в каталоги:

- c:\Windows\Help для ОС Microsoft Windows 95/98/Me/XP;
- c:\WINNT\Help для ОС Microsoft Windows NT/2000.

В указанных каталогах будет производиться поиск файла справочной системы по умолчанию. В этом случае отпадает необходимость явного указания пути к данным файлам в тексте XLM, VBA или помещенной в XLL программы.

Организация работы пользователей с функциями Microsoft Excel

Работа со стандартными и пользовательскими функциями организуется тремя стандартными способами.

- Вызов функции осуществляется путем непосредственного ввода имени функции в ячейку рабочего листа с указанием всех необходимых для данной функции параметров, как в примере программы AtteStat в *главах 3* и *5*. Если пользователь ошибся в написании имени функции или данная функция не существует, в ячейку будет выдано стандартное сообщение об ошибке, означающее, что введенное имя не определено.
- Вызов функции выполнятся с помощью Мастера функций. Вызов функции производится путем выбора из меню пункта **Вставка ▸ Функция** нужной категории функций, затем имени функции и заполнения всех необходимых полей, соответствующих параметрам функции. Примеры программы AtteStat приведены в *главах 3* и *5*, модуля программы ME.com— в *главе 10*.
- Функция вызывается через специальный интерфейс надстройки, как, например, Пакет анализа Microsoft Excel. Вызов функции производится выбором **Сервис ▸ Анализ данных**. Примеры программы AtteStat рассмотрены в *главах 3–8*.

Другие способы, типа работы функций надстройки только с заранее выделенной областью ячеек рабочей книги электронных таблиц мы не рекомендуем.

Организация работы с функциями Microsoft Excel

Существует несколько способов вызова программой Microsoft Excel пользовательских функций, содержащихся в DLL или в модулях VBA.

Обязанностью программиста является разработка удобного интерфейса к разработанным им функциям, максимально похожего на стандартные пользовательские интерфейсы Microsoft Excel.

Содержащиеся в DLL функции могут быть вызваны:

- из рабочего листа, при этом функция зарегистрирована в рабочем листе;
- из программы на VBA, при этом функция зарегистрирована в листе макросов Microsoft Excel или в модуле VBA;
- из рабочего листа или из макроса, при этом функция зарегистрирована в листе макросов. Функция может быть зарегистрирована также и в модуле VBA. В этом случае вызвать ее из рабочего листа можно либо через специальную интерфейсную функцию, обеспечивающую формирование стандартных VBA-массивов из структур данных, переданных из рабочего листа, либо в качестве ее формальных параметров должны быть введены не стандартные VBA-массивы, а структуры, аналогичные структурам данных, переданных из рабочего листа.

Первый способ вызова функций из DLL был доступен в версии Microsoft Excel 97. Начиная с версии Microsoft Excel 2000, регистрация функций DLL в рабочем листе упразднена, а в справочной системе Microsoft Excel 2002 появилась информация о том, что в рабочем листе функции DLL больше не регистрируются. Более подробно работа программистов с функциями DLL и функциями VBA рассмотрена в *главах 3–8*.

Второй способ вызова функций DLL подробно представлен в *главах 3, 5* и *10*.

Третий способ иллюстрируется и примерами *глав 3, 6, 7* и *8*.

Особенности отладки и тестирования расчетных программ

На этапе разработки и отладки расчетных и аналитических программ (научных, технических, бухгалтерских) рекомендуется применять минимум возможностей системы программирования и операционной среды. Лучшим является использование консольных приложений для отладки отдельных модулей (см. *приложение 3*). Если система разработки приложений для Windows не позволяет создавать консольные приложения на платформе Windows, она не слишком годится для программирования сложных расчетных алгоритмов в данной операционной среде.

При разработке расчетных программ полная отладка каждого модуля производится индивидуально. Это повышает надежность структурных единиц, составляющих большую вычислительную программу. Преимуществом такой отладки является то, что качество одного модуля не зависит от другого модуля. Отлаженный модуль мы можем включать в программу, проверяя его совместимость с остальными модулями и работоспособность лишь в ограниченном наборе тестов или вообще не проверяя до тестирования готового программного продукта.

Особенно важно требование индивидуальной отладки каждого модуля при создании DLL для Microsoft Excel или другой работающей с DLL программы, т. к. отладить сложный алгоритм при вызове функций из DLL трудно. Функции, содержащиеся в DLL, надо отладить индивидуально, причем с использованием консольных приложений — нет почти никакой разницы, из какого приложения вызывается та или иная расчетная функция, отличия в форматах принимаемых данных легко корректируются.

Для каждой функции важно подобрать или придумать набор входных и выходных данных, на которых и выполняется отладка. В готовую DLL функцию следует включать только после того, как она будет полностью отлажена, и программист будет уверен в качестве ее работы.

Разложение алгоритма решения задачи по нескольким функциям должно быть разумным. Это совсем не похоже на разложение производственного технологического процесса по операциям, исходя из того принципа, чтобы каждую операцию любой индивидуум мог выполнять с минимальными умственными затратами. Трудно оправдать наличие функции С, состоящей из вызова единственной другой функции С, если, конечно, оплата труда программиста не находится в прямой зависимости от количества составленных им функций.

Усложнения интерфейса функций раньше диктовались унификацией включенных в ту или иную библиотеку ППП (пакет прикладных программ). В настоящее время программисты часто идут на усложнение интерфейса, чтобы совместить привлекательность программы для покупателя с недоступностью исходных кодов. Сложность интерфейса функции должна определяться только требованиями решаемой задачи. У потребителя всегда есть выбор: воспользоваться функцией из библиотеки с недоступными исходными кодами (например, NAG или IMSL), функцией с доступными исходными кодами (NR) или составить функцию самостоятельно.

Следует избегать применения рекурсивных процедур. Считается, что рекурсии дают компактные (см., например, функцию вычисления критерия рандомизации для независимых выборок, представленную в *главе 5*, или методы *главы 10*), но не слишком эффективные решения. При большом числе вложенных вызовов поведение таких функций трудно контролировать, а возросшее время расчета не позволяет широко применять рекур-

сии для задач произвольного размера. Сравните рекурсивную функцию вычисления определителя `Det`, представленную на с. 492 нашей первой книги с гораздо более эффективной процедурой `Krauth`, представленной на с. 511 той же книги, для матрицы 12 порядка. Дождаться завершения работы функции `Det` вряд ли хватит терпения, хотя реализация вполне изящна. В то же время, чтобы точно оценить время решения функцией `Krauth`, из-за высочайшего быстродействия потребуется вызвать ее несколько сотен раз. По поводу компактности решения — тоже спорный вопрос: листинг функции `Krauth`, если убрать комментарии, почти равен объему листинга функции `Det`.

Научная программа исполняется не в компьютере. Прежде, чем попасть к пользователю, она выполняется в уме программиста. Лучшие программисты давно заметили, что иногда трудно объяснить, как была составлена та или иная функция — функция просто была записана. Бывает даже трудно понять, почему функция вообще работает. Вопрос стоит только о том, правильно ли она работает. Или более тонко – всегда ли правильно она работает.

Для некоторых программ, особенно образовательных, лингвистических и развлекательных, то, что система не зависает при их запуске или закрытии — уже большое достижение. От расчетных научных программ требуется больше. Для них правильность вычислений и надежность работы являются основными параметрами, по которым судят о качестве программ.

Проверка правильности вычислений производится путем сопоставления с заранее известными (лучше даже опубликованными) решениями, к которым иногда следует относиться критически. Заранее известные решения могут быть выполнены как вручную, так и с использованием других программ, решающих аналогичные задачи.

Повышенная надежность научным программам часто необходима еще и оттого, что, возможно, выполняться программа будет в условиях, когда нажать клавишу Reset на системном блоке компьютера будет просто некому.

Основным объектом при проверке качества научной программы является исходный код. Без анализа исходного кода доказать, что, предположим, деструктивно настроенный программист не решил отформатировать винчестер пользователя, невозможно. Надуманная «проблема 2000 года», потребовавшая для своего решения больших финансовых затрат, относится именно к проблемам такого сорта. Пользователям иногда важнее иметь возможность изучить исходный код как жест откровенности со стороны разработчика, чем действительно заняться его изучением. И очень немногие будут это делать. Описаний алгоритмов и честных заверений авторов программного обеспечения недостаточно. Если исходный код научной программы не предоставляется общественности, об оценке ее ка-

чества не может быть речи. Для общества дешевле полагать, что расчетной научной программы, исходного кода которой никто не видел, не существует. Правда, есть альтернативы, особенно заметные своими последствиями — это аварии на атомных электростанциях и транспортных коммуникациях, столкновения авиалайнеров, отказы систем жизнеобеспечения, потери спутников.

Ресурсы Интернет

Программирование для Microsoft Excel — задача увлекательная и несложная, если программист обладает необходимой технической информацией. Однако как раз здесь его могут подстерегать неожиданные трудности, разрешить которые без доступа к сети Интернет нельзя.

Большое число ценных материалов на немецком языке по программированию надстроек Microsoft Excel можно найти на превосходном сайте «Thomas Risi Softwareentwicklung» (Thomas Risi, **rtsoftwaredevelopment.de**), и на французском языке на сайте «Complements Excel 97–2002» (Laurent Longre, **longre.free.fr**).

Материалы с сайта MSDN (**msdn.microsoft.com**), которые помогут разобраться с API Microsoft Excel:

- Q152152 «Sample: FRMWRK32.EXE: Updated Generic.xll Template for Excel»,
- Q178474 «HOWTO: Build an Add–in (XLL) for Excel Using Visual C++»,
- Q211563 «XL2000: How to Create an Add–in File. Deploying Office Templates and Application–Specific Add–ins»,
- Q238228 «HOWTO: Build an Office 2000 COM Add–In in Visual Basic»,
- Q256624 «HOWTO: Use a COM Add–In Function as an Excel Worksheet Function».

На сайте «TA–LIB» (Mario Fortier, **ta–lib.org**) предлагаются материалы по программному обеспечению технического анализа и библиотека TA–LIB для C/C++/Excel. Данные материалы можно рекомендовать для изучения построения законченного программного продукта с использованием описываемой технологии.

Компания Planatech Solutions Ltd (**www.as–ltd.co.uk/xllplus–online**) поставляет специализированный инструментарий для разработки XLL под названием XLL+.

Рекомендуемая литература

1. Numerical Recipes in C: The Art of Scientific Computing / W.H. Press, S.A. Teukolsky, W.T. Vetterling, Brian P. Flannery, New York: Cambridge University Press, 2002. Available from World Wide Web: < **http://www.nr.com** >.
2. Stevens A. Turbo C: memory resident utilities, screen I/O and programming techniques. — Management Information Source Inc. Press. (Неофициальный перевод: Стивенс А. Техника программирования на Турбо С. — AcademySoft, 1989)
3. Биллиг В.А. VBA в Office 2000. Офисное программирование. — М.: Издательско-торговый дом «Русская редакция», 1999.
4. Брэдли Х. Дополнительные модули для Excel // PC Magazine / Russian Edition CD-ROM, №7, 2003.
5. Васильков Ю.В., Василькова Н.Н. Компьютерные технологии вычислений в математическом моделировании. — М.: Финансы и статистика, 2002.
6. Керниган Б., Ритчи Д. Язык программирования Си. — М.: Финансы и статистика, 1992. Available from World Wide Web: < **http://lib.ru/ctotor/kernigan.txt** >.
7. Кормен Т., Лейзерсон Ч., Ривест Р. Алгоритмы: построение и анализ. — М.: МЦНМО, 2000.
8. Круглински Д., Уингоу С., Шеферд Дж. Программирование на Microsoft Visual C++ для профессионалов. — СПб: Питер, 2001.
9. Макарова Н.В., Трофимец В.Я. Статистика в Excel: Учебное пособие. — М.: Финансы и статистика, 2002.
10. Мак–Кракен Д., Дорн У. Численные методы и программирование на ФОРТРАНе. — М.: Мир, 1977.
11. Орвис В.Дж. Excel для ученых, инженеров и студентов. — Киев: Юниор, 1999.
12. Орвис В.Дж. Visual Basic for Application на примерах. — М.: БИНОМ.
13. Рихтер Дж. Windows для профессионалов: создание эффективных Win32-приложений с учетом специфики 64–разрядной версии Windows. — СПб: Питер; М.: Издательский дом «Русская редакция», 2001.
14. Страуструп Б. Язык программирования Си++. — М.: Радио и связь, 1991. Available from World Wide Web: < **http://lib.ru/cpphb/cpptut.txt** >. См. также «Введение в язык С++». Available from World Wide Web: < **http://www.citforum.ru/programming/cpp/aglav.shtml** >.

15. Тюлин А., Жуков И., Ефанов Д. На страже конфиденциальной информации // Открытые системы, №10, 2001.
16. Уокенбах Дж. Подробное руководство по созданию формул в Excel 2002. — М.: Издательский дом «Вильямс», 2002.
17. Уэйт М., Прата С., Мартин Д. Язык Си. Руководство для начинающих. — М.: Мир, 1988.
18. Шаргин А. Отладка в среде Visual C++ // RSDN Magazine, #0, 2002, с. 38–54.
19. Шилдт Г. Теория и практика C++. — СПб.: BHV–Санкт-Петербург, 1996.

ГЛАВА 3

Предварительная обработка данных на VBA и C/C++

«Изучая истину, можно иметь троякую цель: открыть истину, когда ищем ее; доказать ее, когда нашли; наконец, отличить ото лжи, когда ее рассматриваем».
Б. Паскаль

Для работы с материалами главы понадобятся лицензионные средства разработки и тестирования программ:

- Microsoft Excel 97 или выше;
- Microsoft Visual C++ 5.0 или выше;
- Borland C++ 5.5 или выше;
- Bloodshed Dev–C++ 4.01 или выше.

Потребуется знание VBA, C.

Решаемая проблема

Предварительная обработка предназначена для подготовки «сырых» исходных данных к статистической обработке. К сожалению, методы предварительной обработки отсутствуют в стандартных электронных таблицах и многих программах анализа данных. Например, проверка нормальности распределения должна обязательно предшествовать любой статистической обработке: описательной статистике, проверке гипотез (см. *главу 5*

и *приложение 1*), факторному анализу (см. *главу 6*) и другим методам. Данная процедура очень важна, т. к. определяет, каким методом статистической обработки данных можно пользоваться, а каким нет. Многие из алгоритмов статистической обработки данных основаны на предположении о нормальности распределения. Если это предположение не соблюдается, методы обработки могут не быть адекватными природе исходных данных.

Из других методов предварительной обработки в главе рассматриваются обработка пропущенных значений и обработка выбросов.

В пределах каждого раздела обозначения в формулах, как правило, соответствуют друг другу при описании разных методов. Если это не так, мы особо даем пояснения. Например, при описании одномерных и многомерных процедур выборка может быть обозначена одним и тем же символом, однако смысл эти обозначения будут иметь различный. Между разделами одни и те же переменные или параметры могут иметь различные буквенные обозначения. Это вызвано традициями той или иной отрасли анализа.

Проверка типа распределения эмпирических данных

Задача установления типа распределения неизбежно возникает при подготовке к применению методов проверки гипотез и дисперсионного анализа с целью выбора адекватной статистической модели. Проверка соответствия эмпирического распределения заданному теоретическому распределению может быть выполнена различными способами. В данном разделе рассмотрены почти все основные классические и современные методы, которые охватывают выборки любой численности, встречающейся в практических задачах. Применение предлагаемых методик снимает проблему проверки нормальности распределения малых выборок. Ограничение на максимальную численность выборки накладывает Microsoft Excel.

Особенности решения

Регистрация функций DLL, реализующих расчетные методы, выполнена в листе макросов. Макрос выполняется автоматически стандартной VBA-функцией `Run`, при открытии рабочей книги или шаблона, т. к. помещен в тело VBA-функции `Auto_Open`. Более подробно о регистрации функций рассказано в *главе 5*.

Нам нет необходимости протоколировать результаты вычислений в рабочем листе, поэтому вывод осуществляем только в окно сообщений с помощью стандартной VBA-функции `MsgBox`, сохраняя возможность вызова функций из рабочего листа Microsoft Excel. В зависимости от результата расчета, пользователь будет делать выбор метода анализа. Например, в задаче проверки статистической гипотезы (см. *главу 5*), соответствие исходных данных нормальному распределению позволит использовать параметрические методы пакета анализа Microsoft Excel. Если нет, пользователь должен будет применить непараметрические методы.

Все представленные в разделе функции составлены на алгоритмическом языке C, с целью получения максимальной скорости вычислений. Для отладки логики алгоритмов использовалась система программирования Bloodshed Dev-C++ 4.01. Составлялись консольные приложения, т. к. только они позволяют создать минимальное приложение, без использования интерфейсных библиотек, что очень важно при отладке математических алгоритмов для произвольной целевой платформы. Применение операционной системы Windows XP позволило минимизировать накладные расходы, неизбежные при нетривиальной разработке.

После отладки, экспортируемые функции были адаптированы к Microsoft Excel. Разработка и отладка выполнялись отдельно, поэтому адаптация была поэтапной.

- Помещение функций в DLL потребовало изменить их интерфейсную часть, в соответствии с требованиями API Microsoft Excel. Это не отразилось на большом числе внутренних функций, которыми оперирует Microsoft Excel. Внутренние функции DLL из внешних приложений вызываться не могут, а пользователю нет необходимости знать об их существовании.

- Все целые данные типа `int` получили тип `short`. Это касается и внутренних функций, не экспортируемых DLL. С формальной точки зрения, в Win32 целые типы `int` и `short` эквивалентны, поэтому никаких дополнительных проблем не возникло.

- Вызовы процедур, предназначенных для обработки одной выборки, или же матрицы данных, в отличие от программ на алгоритмическом языке C или C++, в VBA будут выглядеть одинаково[1]. Это вызвано той особенностью API Microsoft Excel, что в передаваемой структуре `*FP` уже присутствует информация о числе столбцов и строк. В дополнительной передаче тех же сведений в функцию нет необходимости.

---

[1] *Путем дополнительных усилий и в C/C++ вызовы можно сделать аналогичными VBA. Это, правда, увеличит объем кода и ухудшит читабельность программ.*

- Чтобы изменения затрагивали минимальную часть кода, были введены стандартные массивы. Из переданных в функции структур `*FP` стандартным образом извлекаются как исходные данные, так и информация о числе столбцов и строк области ячеек рабочего листа, переданной из электронных таблиц. Обычно информация о столбцах и строках передается через формальные параметры. При необходимости, читателю не составит труда просто опустить данные фрагменты и переписать интерфейс функции, чтобы получить стандартные функции C.

Научные алгоритмы

Не всегда обоснованно считается, что эмпирические данные имеют нормальное распределение тем вернее, чем больше численность выборки. Численность выборки считается достаточно большой, если она превышает 100, выборки численностью менее 30 почти всегда считаются малыми. На самом деле вряд ли можно избежать статистической проверки нормальности распределения даже при больших выборках, постулировав нормальность ввиду большой численности (пример неоднородной выборки в *главе 11*).

Полные исходные тексты модуля проверки нормальности распределения находятся на прилагаемом носителе. Применяются семь методов проверки нормальности распределения одномерной случайной величины, представленной в виде обычной эмпирической выборки:

- критерий Колмогорова;
- критерий Крамера–Мизеса;
- критерий Андерсона–Дарлинга;
- критерий Эппса–Палли;
- критерий Шапиро–Уилка;
- критерий коэффициента асимметрии;
- критерий эксцесса.

Методы реализуют все современные подходы к решению проблемы. Они охватывают выборки практически любой численности. Данные методы необходимо применять, например, перед проверкой статистических гипотез (пакет анализа Microsoft Excel, а также методы *главы 5*).

Рассмотрены три метода проверки нормальности распределения (произвольной размерности):

- критерий коэффициента асимметрии Мардиа;
- критерий эксцесса Мардиа;
- критерий Хенце–Цирклера.

Данные методы необходимо использовать, например, перед применением факторного анализа *(глава 6)* и части методов кластерного анализа *(глава 7)*, перед исследованием корреляции *(глава 8)*. В этих случаях необходимо исследовать нормальность распределения двумерных эмпирических выборок.

При проверке согласия эмпирического и заданного теоретического распределения различают простые и сложные гипотезы. Простой гипотеза будет в том случае, если теоретическое распределение задано всеми своими параметрами. Сложной гипотеза будет, если все или некоторые параметры теоретического распределения оцениваются по выборке. Так, в случае нормального распределения, по выборке могут оцениваться среднее значение и дисперсия. Поэтому для нормального распределения сложная гипотеза может быть трех видов:

- по выборке оценивается среднее, дисперсия задана;
- по выборке оценивается дисперсия, среднее задано;
- по выборке оцениваются и среднее, и дисперсия.

Для исследователя сказанное означает, что, хотя статистика критерия вычисляется во всех четырех (1 простая гипотеза и 3 сложных) случаях по одним и тем же алгоритмам, необходимо наличие статистических таблиц (или алгоритмов вычисления) критических значений, особых как для каждого типа распределения, так и для каждого случая сложной гипотезы. Некоторые авторы программ применяют алгоритмы вычисления критических значений в случае простой гипотезы для сложных гипотез. В литературе опубликованы и даже стандартизованы формулы и таблицы используемых критериев для различных гипотез, поэтому при внимательном подходе такие ошибки допустить трудно.

Нами рассматриваются алгоритмы только для сложных гипотез. Для нормального распределения это означает, что среднее значение и дисперсию там, где это необходимо, мы оцениваем по эмпирической выборке. Поэтому критические значения статистик критериев берутся из формул и таблиц для соответствующей сложной гипотезы.

Некоторые критерии согласия изначально, по замыслу авторов алгоритмов, не предполагают различие простой и сложной гипотез. Все параметры оцениваются по эмпирической выборке, поэтому данные критерии предназначены только для сложных гипотез.

Попутно укажем также на возможности проверки близости закона распределения к равномерному распределению. Для этого могут применяться относительная энтропия и избыточность, рассмотренные в *главе 4*.

Гистограмма распределения, включая вывод диаграммы средствами VBA, может быть построена с помощью вычислительных алгоритмов, рассмотренных в *главе 11*.

Критерий Колмогорова

Критерий Колмогорова (Колмогорова – Смирнова) является распространенным и обоснованным методом проверки согласия распределения. Он предназначен для проверки согласия эмпирической и теоретической функций распределения. Статистика критерия Колмогорова задается формулой:

$$D_n = \sup_{|x|<\infty} \left| F_n(x) - F(x) \right|,$$

где $F_n(.)$ — эмпирическая функция распределения,

$F(.)$ — теоретическая функция распределения. В случае сложной гипотезы это функция нормального распределения, определенная параметрами, вычисленными по эмпирической выборке,

n — численность эмпирической выборки.

Функция распределения может строиться непосредственно по эмпирической выборке произвольного объема. В этом случае вычисление статистики критерия производится по формуле

$$D_n = \max(D_n^+, D_n^-),$$

где

$$D_n^+ = \max_{1 \le m \le n} \left(\frac{m}{n} - F(\eta_m) \right), \quad D_n^- = \max_{1 \le m \le n} \left(F(\eta_m) - \frac{m-1}{n} \right).$$

Здесь $\eta_m, m = 1,2,...,n$ — это эмпирическая выборка, отсортированная в порядке возрастания значений вариант.

Начинающие исследователи часто задают вопрос, где же в формулах вычисления статистик D^+ и D^- содержатся эмпирические функции распределения. Выражение $\frac{m}{n}, m = 1,2,...,n$ как раз и представляет собой эмпирическую функцию распределения — неубывающий ступенчатый график со значениями ординат в диапазоне от нуля до единицы. Такой график может быть равномерным по оси абсцисс (неконструктивный подход) или по оси ординат. Во втором случае ступеньки графика, соот-

ветствующие вариантам, располагаются по оси абсцисс в соответствии с отсортированной выборкой, а ординаты ступенек — равномерно. Иногда пытаются строить статистику критерия Колмогорова через частоты распределения. Во-первых, данный подход применим только для больших выборок. Во-вторых, снижается точность алгоритма[2] за счет приведения количественной шкалы измерения исходной эмпирической выборки к порядковой шкале частот распределения.

Примечание

Признаки, в зависимости от шкалы измерения, бывают: номинальными (например, велосипед, мотоцикл, автомобиль), порядковыми (отлично, хорошо, удовлетворительно), количественными (год рождения, средняя заработная плата). Номинальные признаки с двумя градациями называются качественными или дихотомическими.

В листинге 3.1 приведен текст функции точного вычисления критерия Колмогорова. P-значение вычислено путем аппроксимации критических значений, полученных методом компьютерного моделирования.

Листинг 3.1

```
double __declspec(dllexport) Kolmogorov (FP *xx)
//
// Функция вычисления критерия Колмогорова для проверки нормальности
// распределения в случае сложной гипотезы.
// Обозначения:
//  xx — эмпирическая выборка, разрушается.
// Возвращаемое значение:
//  P-значение статистики критерия.
//
{
 short i,                          // Счетчик цикла
       n = xx->rows * xx->columns; // Численность выборки
 double r = 0,                     // Статистика критерия
        *x,                        // Копия выборки
        m = 0,                     // Параметры распределения,
        s = 0,                     // затем рабочие переменные
        d = 0;                     // Вспомогательная переменная
```

[2] *Если, конечно, данные для анализа изначально не были получены в виде частот.*

```
////////////////
// Копия выборки
x = new double[n];
for (i = 0; i < n; i++)
  x[i] = xx->Array[i];

/////////////
// Сортировка
SortArrayUp (x,n);

//////////////////////////////////
// Оценка параметров распределения
for (i = 0; i < n; i++)
  {
   m += x[i];
   s += x[i] * x[i];
  }
m /= n;
s = sqrt((s - n * m * m) / (n - 1));
if (s == 0)
  {
   delete [] x;
   return 0;
  }

///////////////
// Нормирование
for (i = 0; i < n; i++)
  x[i] = (x[i] - m) / s;

//////////////////////////////////
// Вычисление статистики критерия
for (i = 0; i < n; i++)
  {
   m = (1.0 + i) / n - NormalDistribution(x[i]);
   if (m > r)
     r = m;
   s = NormalDistribution(x[i]) - double(i) / n;
```

```
    if (s > d)
      d = s;
   }
 if (r < d)
   r = d;

 delete [] x;

 ////////////////////////
 // Вычисление P-значения
 return 501.258437895150 / exp (10.398547901867 * r *
       (sqrt(n) - 0.01 + 0.85 / sqrt(n)));
}
```

Критерии типа омега-квадрат

Критерии типа омега-квадрат основаны на идее сравнения эмпирического и теоретического распределений в квадратичной метрике

$$\omega^2 = \int_{-\infty}^{\infty} [F_n(x) - F(x)]^2 \psi[F(x)] dF(x),$$

где $F_n(.)$ — эмпирическая функция распределения,

$F(.)$ — теоретическая функция распределения. В случае сложной гипотезы это функция нормального распределения, определенная параметрами, вычисленными по эмпирической выборке,

$\psi(.)$ — некоторая весовая функция,

n — численность эмпирической выборки.

При выборе весовой функции в виде $\psi(t) = 1$ получается критерий Крамера–Мизеса (Крамера–Фон Мизеса, Крамера–Мизеса–Смирнова).

Как и в алгоритме вычисления критерия Колмогорова, функция распределения может строиться непосредственно по эмпирической выборке, без разнесения вариант по классам, поэтому практическое вычисление статистики критерия Крамера–Мизеса производится по формуле

$$n\omega^2 = \frac{1}{12n} + \sum_{j=1}^{n} \left[F(\eta_j) - \frac{2j-1}{2n} \right]^2,$$

где $\eta_m, m = 1,2,...,n$ —эмпирическая выборка, отсортированная в порядке возрастания значений вариант.

При выборе весовой функции в виде $\psi(t) = \dfrac{1}{t(1-t)}$ получается популярный критерий Андерсона–Дарлинга. Практические вычисления статистики критерия производятся по формуле

$$n\Omega^2 = -n - 2\sum_{j=1}^{n}\left\{\frac{2j-1}{2n}\ln F\left(\eta_j\right) + \left(1 - \frac{2j-1}{2n}\right)\ln\left[1 - F\left(\eta_j\right)\right]\right\}.$$

В листинге приведены исходные тексты функции вычисления критериев Крамера–Мизеса и Андерсона–Дарлинга. P-значения вычислены путем аппроксимации критических значений, полученных методом компьютерного моделирования.

Листинг 3.2

```
double __declspec(dllexport) CramerVonMises (FP *xx)
//
// Функция вычисления критерия омега-квадрат Крамера-Мизеса
// для проверки нормальности распределения в случае сложной гипотезы.
// Обозначения:
//  xx — эмпирическая выборка, разрушается.
// Возвращаемое значение:
//  P-значение статистики критерия.
//
{
 short i,                          // Счетчик цикла
       n = xx->rows * xx->columns; // Численность выборки
 double r = 1.0 /12 / n,           // Статистика критерия
        *x,                        // Копия выборки
        m = 0,s = 0;               // Параметры распределения,
                                   // затем рабочие переменные

 ///////////////
 // Копия выборки
 x = new double[n];
 for (i = 0; i < n; i++)
   x[i] = xx->Array[i];

 /////////////
 // Сортировка
```

```
 SortArrayUp (x,n);

 ////////////////////////////////
 // Оценка параметров распределения
 for (i = 0, m = 0, s = 0; i < n; i++)
   {
    m += x[i];
    s += x[i] * x[i];
   }
 m /= n;
 s = sqrt((s - n * m * m) / (n - 1));
 if (s == 0)
    {
     delete [] x;
     return 0;
    }

 ///////////////
 // Нормирование
 for (i = 0; i < n; i++)
   x[i] = (x[i] - m) / s;

 ////////////////////////////////
 // Вычисление статистики критерия
 for (i = 0; i < n; i++)
   {
    m = NormalDistribution(x[i]) - (2.0 * i + 1) / 2 / n;
    r += m * m;
   }

 delete [] x;

 ////////////////////////
 // Вычисление P-значения
 return 2.552990990559 / exp (31.183639935070 * r * (0.5 / n + 1.0));
}

double __declspec(dllexport) AndersonDarling (FP *xx)
```

```
//
// Функция вычисления критерия омега-квадрат Андерсона-Дарлинга
// для проверки нормальности распределения в случае сложной гипотезы.
// Обозначения:
//  xx — эмпирическая выборка, разрушается.
// Возвращаемое значение:
//  P-значение статистики критерия.
//
{
 short i,                          // Счетчик цикла
       n = xx->rows * xx->columns; // Численность выборки
 double r = 0,                     // Статистика критерия
        *x,                        // Копия выборки
        m = 0,s = 0;               // Параметры распределения,
                                   // затем рабочие переменные

 ////////////////
 // Копия выборки
 x = new double[n];
 for (i = 0; i < n; i++)
   x[i] = xx->Array[i];

 /////////////
 // Сортировка
 SortArrayUp (x,n);

 ////////////////////////////////
 // Оценка параметров распределения
 for (i = 0; i < n; i++)
   {
    m += x[i];
    s += x[i] * x[i];
   }
 m /= n;
 s = sqrt((s — n * m * m) / (n — 1));
 if (s == 0)
   {
    delete [] x;
```

```
    return 0;
   }

 ///////////////
 // Нормирование
 for (i = 0; i < n; i++)
   x[i] = (x[i] — m) / s;

 ////////////////////////////////
 // Вычисление статистики критерия
 for (i = 0; i < n; i++)
   {
    m = (2.0 * i + 1) / 2 / n;
    s = NormalDistribution(x[i]);
    r += m * log (s) + (1.0 — m) * log (1.0 — s);
   }
  r = — r * 2 — n;

 delete [] x;

 ////////////////////////
 // Вычисление P-значения
 return 2.514302687951 / exp (5.198361440790 * r);
}
```

Критерии типа Эппса–Палли

Критерий Эппса–Палли основан на идее измерения расстояния между характеристической функцией эмпирического распределения и модельной (теоретической) функцией распределения

$$T_n = n \int_{-\infty}^{\infty} \left| \psi_n(t) - e^{-t^2/2} \right|^2 \varphi(t) dt,$$

где эмпирическая характеристическая функция распределения имеет вид

$$\psi_n(t) = \frac{1}{n} \sum_{j=1}^{n} e^{it\left(X_j - \overline{X}\right)/S},$$

n — численность эмпирической выборки,

i — мнимая единица, т. е. число, квадрат которого равен отрицательной единице; таким образом, $i = \sqrt{-1}$,

$X_j, j = 1,2,...,n$ — эмпирическая выборка,

$\overline{X} = \frac{1}{n}\sum_{j=1}^{n} X_j$ — выборочное среднее значение,

S^2 — дисперсия, оцениваемая по выборке.

выражение $|.|^2$ означает квадрат модуля комплексного выражения.

Для нормального распределения формула практического вычисления статистики критерия Эппса–Палли имеет вид

$$T_n = 1 + \frac{n}{\sqrt{3}} + \frac{2}{n}\sum_{j=2}^{n}\sum_{k=1}^{j-1} e^{-\frac{1}{2}(X_j - X_k)^2/S^2} - \sqrt{2}\sum_{j=1}^{n} e^{-\frac{1}{4}(X_j - \overline{X})^2/S^2}.$$

P-значение для малых выборок берется по таблице, составленной методом компьютерного моделирования, а для выборок численностью от 10 и выше вычисляется по формуле

$P = \Phi(z)$,

где $\Phi(z)$ — функция стандартного нормального распределения,

величина $z = z(T_n^*)$ рассчитывается как

$z = \gamma + \delta\log((T_n^* - \xi)/(\xi + \lambda - T_n^*)$,

где $\gamma, \delta, \xi, \lambda$ — константы,

а $T_n^* = (T_n - 0{,}365/n + 1{,}34/n^2)(1 + 1{,}3/n)$.

Существует аналог критерия Эппса–Палли, предназначенный для проверки нормальности многомерного распределения. Вычисление критерия Хенце–Цирклера (инвариантного теста Хенце–Цирклера) производится по формуле

$$D_{n,\beta} = \frac{1}{n^2}\sum_{j=1}^{n}\sum_{k=1}^{n} e^{-\frac{\beta^2}{2}|Y_j - Y_k|^2} - 2(1+\beta^2)^{-d/2}\frac{1}{n}\sum_{j=1}^{n} e^{-\frac{\beta^2}{2(1+\beta^2)}|Y_j|^2} + \left(1 + 2\beta^2\right)^{-d/2},$$

где β — вычисляемый или задаваемый параметр,

d — размерность многомерной (d-мерной) выборки $X_j, j = 1,2,...,n,$.

Многомерность эмпирической выборки для программиста и исследователя означает, что в рабочем листе электронных таблиц она представлена таблицей чисел, строки которой являются вариантами (в данном слу-

чае — векторными) d-мерной выборки, число строк равно численности многомерной выборки n, а число столбцов равно размерности выборки d.

В качестве примера многомерной выборки может служить вектор измерений d характеристик испытуемого. При этом d характеристик одного испытуемого будут записаны в одной строке электронной таблицы и занимать d столбцов. Если число испытуемых (или, например, число измерений вектора параметров одного испытуемого) обозначить через n, то вся матрица данных будет занимать область из d столбцов и n строк.

Остальные входящие в формулу параметры вычисляются как

$$\left|Y_j - Y_k\right|^2 = \left(X_j - X_k\right)' S^{-1}\left(X_j - X_k\right),$$

$$\left|Y_j\right|^2 = \left(X_j - \overline{X}\right)' S^{-1}\left(X_j - \overline{X}\right),$$

где S^{-1} — матрица, обратная дисперсионно-ковариационной матрице (*см. след. раздел*),

$\overline{X}$ — d-мерный вектор среднего значения, вычисленный по d-мерной выборке,

штрих означает операцию транспонирования.

Распределение статистики критерия $D_{n,\beta}$ асимптотически нормально (по другим источникам — асимптотически логарифмически нормально).

В листинге приведены тексты функций вычисления критериев Эппса–Палли и Хенце–Цирклера.

Листинг 3.3

```
double __declspec(dllexport) EppsPulley (FP *xx)
//
// Функция вычисления критерия Эппса-Палли для проверки
// нормальности распределения в случае сложной гипотезы.
// Обозначения:
//  xx – эмпирическая выборка.
// Возвращаемое значение:
//  P-значение критерия Эппса-Палли.
//
{
 short k,j,                        // Счетчик цикла
       n = xx->rows * xx->columns; // Численность выборки
```

```
double m = 0,                    // Выборочное среднее
       d = 0,                    // Стандарт
       *x,                       // Копия выборки
       b = 0,                    // Рабочая переменная
       t = n / sqrt(3) + 1;      // Статистика критерия

////////////////
// Копия выборки
x = new double[n];
for (j = 0; j < n; j++)
  x[j] = xx->Array[j];

//////////////////////////////////
// Оценка параметров распределения
for (j = 0; j < n; j++)
  {
   m += x[j];
   d += x[j] * x[j];
  }
m /= n;
d = (d — n * m * m) / (n — 1);
if (d == 0)
  return 0;

//////////////////////////////
// Расчет статистики критерия
for (k = 1; k < n; k++)
  for (j = 0; j < k; j++)
    b += exp(-(x[j] — x[k]) *(x[j] — x[k]) / 2 / d);
t += b / n * 2;

for (j = 0, b = 0; j < n; j++)
  b += exp(-(x[j] — m) * (x[j] — m) / 4 / d);
t -= b * sqrt(2);

delete [] x;

//////////////////////////////
```

```
 // Расчет P-значения критерия
 if (n == 4)
   return 0.000004220828 / PowerOf(t,7.369420999768);
 else if (n == 5)
   return 0.000636023436 / PowerOf(t,3.690587754895);
 else if (n == 6)
   return 0.000907082768 / PowerOf(t,3.560934199857);
 else if (n == 7)
   return 0.001049523019 / PowerOf(t,3.503440054279);
 else if (n == 8)
   return 0.001196973326 / PowerOf(t,3.450785535968);
 else if (n == 9)
   return 0.001264701758 / PowerOf(t,3.437060244217);
 else
   ////////////////////////////////////
   // Параметр нормальной аппроксимации
   t = (t - 0.365 / n + 1.34 / n / n) * (1 + 1.3 / n);

 ////////////////////////////////////////////////
 // Вычисление нормальной аппроксимации P-значения
 return 1 - NormalDistribution(fabs(3.55295 + 1.23062 *
           log ((t + 0.020682) / (2.245958 - t))));
}

double __declspec(dllexport) HenzeZirkler (FP *xx)
//
// Функция вычисления критерия Хенце-Цирклера для проверки
// нормальности многомерного распределения.
// Обозначения:
//  xx - многомерная эмпирическая выборка.
// Возвращаемое значение:
//  P-значение статистики критерия.
//
{
 short i,j,l,           // Счетчик цикла
       m1,m2,           // Индекс
       n = xx->rows,    // Численность выборки
       k = xx->columns; // Размерность выборки
```

```
double *m,              // Вектор выборочного среднего
       *s,              // Ковариационная матрица
       *t,              // Обратная ковариационная матрица
       *x,              // Копия выборки
       *mi,*mj,*t1,     // Рабочие массивы
       g = 0,           // Статистика критерия
       g1 = 0,b;        // Рабочая переменная

///////////////
// Копия выборки
x = new double[n * k];
for (i = 0; i < n * k; i++)
  x[i] = xx->Array[i];

//////////////////////////////////////////
// Динамическая память для рабочих массивов
// (результат операции не проверяется)
s = new double[k * k];
t = new double[k * k];

//////////////////////////////////
// Оценка параметров распределения
CovMatrix (x,s,n,k);
if(InverseMatrix (s,t,k) == -2)
  {
   delete [] x;
   delete [] s;
   delete [] t;
   return 0;
  }

//////////////////////////////////////////
// Динамическая память для рабочих массивов
// (результат операции не проверяется)
m = new double[k];
mi = new double[k];
mj = new double[k];
t1 = new double[1];
```

```
 ////////////////////////////////
// Оценка параметров распределения
MeanVector (x,m,n,k);

//////////////////////////////////////////
// Параметр, зависящий от размерности выборки
b = PowerOf(0.25 * (2 * k + 1),1.0 / (k + 4)) *
    PowerOf(n,1.0 / (k + 4)) / sqrt(2);
b *= b;

//////////////////////////////
// Вычисление статистики критерия
for (i = 0; i < n; i++)
  {
   for (l = 0, m2 = i; l < k; l++)
     {
      mi[l] = x[m2] - m[l];
      m2 += n;
     }
   MatrixTMultiplyMatrix (mi,t,s,1,k,k);
   for (j = 0; j < n; j++)
     {
      for (l = 0, m1 = j; l < k; l++)
        {
         mj[l] = x[m1] - m[l];
         m1 += n;
        }
      MatrixMultiplyMatrix (s,mj,t1,1,k,1);
      g += exp(- b * t1[0] / 2);
     }
   MatrixMultiplyMatrix (s,mi,t1,1,k,1);
   g1 += exp(-b * t1[0] / 2 / (b + 1));
  }
b = PowerOf(b + 1,-0.5 * k);

//////////////////////////////////
// Освобождение динамической памяти
delete [] x;
```

```
 delete [] s;
 delete [] t;
 delete [] m;
 delete [] mi;
 delete [] mj;
 delete [] t1;

 ////////////////////////
 // Вычисление P-значения
 return 1 - NormalDistribution (fabs(g / n / n - g1 / n * 2 * b + b));
}
```

Критерий Шапиро — Уилка

В ряде опытов, особенно в медицинских исследованиях, численность выборки мала. Специально для проверки нормальности распределения малых, численностью от 3 до 50 вариант, выборок Шапиро и Уилк разработали критерий W.

Вычисления статистики критерия производятся по формулам:

$$W = b^2 / S^2,$$

где $S^2 = \sum_{i=1}^{n} \left(x_i - \bar{x}\right)^2$,

$$b = \sum_{i=1}^{k} a_{n-i+1} \left(x_{n-i+1} - x_i\right),$$

а $x_i, i = 1,2,...,n$ — отсортированная в порядке возрастания выборка,

n — численность эмпирической выборки,

$k = n / 2$, если n — четное,

$k = (n - 1) / 2$, если n — нечетное,

$a_{n-i+1}; i = 1,2,...,k; n = 3,4,...,50$ — константы.

Для вычисления P–значения применяется нормальная аппроксимация.

Листинг 3.4

```
double __declspec(dllexport) ShapiroWilk (FP *xx)
//
// Функция вычисления критерия W Шапиро-Уилка.
```

```
// Обозначения:
//  xx — эмпирическая выборка, разрушается.
// Возвращаемое значение:
//  P-значение статистики критерия.
//
{
 short i,j,l = 0,                    // Счетчики цикла
       s = 0,                        // Начало отсчета массива коэффициен-
тов
       n = xx->rows * xx->columns; // Численность выборки
 double d = 0,b = 0,                 // Рабочие переменные
        *x,                          // Копия выборки
        w,                           // Статистика критерия
        a[] = {                      // Массив коэффициентов
 0.7071,
 0.6872,0.1677,
 0.6646,0.2413,
 0.6431,0.2806,0.0875,
 0.6233,0.3031,0.1401,
 0.6052,0.3164,0.1743,0.0561,
 0.5888,0.3244,0.1976,0.0947,
 0.5739,0.3291,0.2141,0.1224,0.0399,
 0.5601,0.3315,0.2260,0.1429,0.0695,
 0.5475,0.3325,0.2347,0.1586,0.0922,0.0303,
 0.5359,0.3325,0.2412,0.1707,0.1099,0.0539,
 0.5251,0.3318,0.2460,0.1802,0.1240,0.0727,0.0240,
 0.5150,0.3306,0.2495,0.1878,0.1353,0.0880,0.0433,
 0.5056,0.3290,0.2521,0.1939,0.1447,0.1005,0.0593,0.0196,
 0.4968,0.3273,0.2540,0.1988,0.1524,0.1109,0.0725,0.0359,
 0.4886,0.3253,0.2553,0.2027,0.1587,0.1197,0.0837,0.0496,0.0163,
 0.4808,0.3232,0.2561,0.2059,0.1641,0.1271,0.0932,0.0612,0.0303,
 0.4734,0.3211,0.2565,0.2085,0.1686,0.1334,0.1013,0.0711,0.0422,0.0140,
 0.4643,0.3185,0.2578,0.2119,0.1736,0.1399,0.1092,0.0804,0.0530,0.0263,
 0.4590,0.3156,0.2571,0.2131,0.1764,0.1443,0.1150,0.0878,0.0618,0.0368,
 0.0122,
 0.4542,0.3126,0.2563,0.2139,0.1787,0.1480,0.1201,0.0941,0.0696,0.0459,
 0.0228,
 0.4493,0.3098,0.2554,0.2145,0.1807,0.1512,0.1245,0.0997,0.0764,0.0539,
 0.0321,0.0107,
```

0.4450,0.3069,0.2543,0.2148,0.1822,0.1539,0.1283,0.1046,0.0823,0.0610,
0.0403,0.02,
0.4407,0.3043,0.2533,0.2151,0.1836,0.1563,0.1316,0.1089,0.0876,0.0672,
0.0476,0.0284,0.0094,
0.4366,0.3018,0.2522,0.2152,0.1848,0.1584,0.1346,0.1128,0.0923,0.0728,
0.0540,0.0358,0.0178,
0.4328,0.2992,0.2510,0.2151,0.1857,0.1601,0.1372,0.1162,0.0965,0.0778,
0.0598,0.0424,0.0253,0.0084,
0.4291,0.2968,0.2499,0.2150,0.1864,0.1616,0.1395,0.1192,0.1002,0.0822,
0.0650,0.0483,0.0320,0.0159,
0.4254,0.2944,0.2487,0.2148,0.1870,0.1630,0.1415,0.1219,0.1036,0.0862,
0.0697,0.0537,0.0381,0.0227,0.0076,
0.4220,0.2921,0.2475,0.2145,0.1874,0.1641,0.1433,0.1243,0.1066,0.0899,
0.0739,0.0585,0.0435,0.0289,0.0144,
0.4188,0.2898,0.2463,0.2141,0.1878,0.1651,0.1449,0.1265,0.1093,0.0931,
0.0777,0.0629,0.0485,0.0344,0.0206,0.0068,
0.4156,0.2876,0.2451,0.2137,0.1880,0.1660,0.1463,0.1284,0.1118,0.0961,
0.0812,0.0669,0.0530,0.0395,0.0262,0.0131,
0.4127,0.2854,0.2439,0.2132,0.1882,0.1667,0.1475,0.1301,0.1140,0.0988,
0.0844,0.0706,0.0572,0.0441,0.0314,0.0187,0.0062,
0.4096,0.2834,0.2427,0.2127,0.1883,0.1673,0.1487,0.1317,0.1160,0.1013,
0.0873,0.0739,0.0610,0.0484,0.0361,0.0239,0.0119,
0.4068,0.2813,0.2415,0.2121,0.1883,0.1678,0.1496,0.1331,0.1179,0.1036,
0.0900,0.0770,0.0645,0.0523,0.0404,0.0287,0.0172,0.0057,
0.4040,0.2794,0.2403,0.2116,0.1883,0.1683,0.1505,0.1344,0.1196,0.1056,
0.0924,0.0798,0.0677,0.0559,0.0444,0.0331,0.0220,0.0110,
0.4015,0.2774,0.2391,0.2110,0.1881,0.1686,0.1513,0.1356,0.1211,0.1075,
0.0947,0.0824,0.0706,0.0592,0.0481,0.0372,0.0264,0.0158,0.0053,
0.3989,0.2755,0.2380,0.2104,0.1880,0.1689,0.1520,0.1366,0.1225,0.1092,
0.0967,0.0848,0.0733,0.0622,0.0515,0.0409,0.0305,0.0203,0.0101,
0.3964,0.2737,0.2368,0.2098,0.1878,0.1691,0.1526,0.1376,0.1237,0.1108,
0.0986,0.0870,0.0759,0.0651,0.0546,0.0444,0.0343,0.0244,0.0146,0.0049,
0.3940,0.2719,0.2357,0.2091,0.1876,0.1693,0.1531,0.1384,0.1249,0.1123,
0.1004,0.0891,0.0782,0.0677,0.0575,0.0476,0.0379,0.0283,0.0188,0.0094,
0.3917,0.2701,0.2345,0.2085,0.1874,0.1694,0.1535,0.1392,0.1259,0.1136,
0.1020,0.0909,0.0804,0.0701,0.0602,0.0506,0.0411,0.0318,0.0227,0.0136,
0.0045,
0.3894,0.2684,0.2334,0.2078,0.1871,0.1695,0.1539,0.1398,0.1269,0.1149,

```
0.1035,0.0927,0.0824,0.0724,0.0628,0.0534,0.0442,0.0352,0.0263,0.0175,
0.0087,
0.3872,0.2667,0.2323,0.2072,0.1868,0.1695,0.1542,0.1405,0.1278,0.1160,
0.1049,0.0943,0.0842,0.0745,0.0651,0.0560,0.0471,0.0383,0.0296,0.0211,
0.0126,0.0042,
0.3850,0.2651,0.2313,0.2065,0.1865,0.1695,0.1545,0.1410,0.1286,0.1170,
0.1062,0.0959,0.0860,0.0765,0.0673,0.0584,0.0497,0.0412,0.0328,0.0245,
0.0163,0.0081,
0.3830,0.2635,0.2302,0.2058,0.1862,0.1695,0.1548,0.1415,0.1293,0.1180,
0.1073,0.0972,0.0876,0.0783,0.0694,0.0607,0.0522,0.0439,0.0357,0.0277,
0.0197,0.0118,0.0039,
0.3808,0.2620,0.2291,0.2052,0.1859,0.1695,0.1550,0.1420,0.1300,0.1189,
0.1085,0.0986,0.0892,0.0801,0.0713,0.0628,0.0546,0.0465,0.0385,0.0307,
0.0229,0.0153,0.0076,
0.3789,0.2604,0.2281,0.2045,0.1855,0.1693,0.1551,0.1423,0.1306,0.1197,
0.1095,0.0998,0.0906,0.0817,0.0731,0.0648,0.0568,0.0489,0.0411,0.0335,
0.0259,0.0185,0.0111,0.0037,
0.3770,0.2589,0.2271,0.2038,0.1851,0.1692,0.1553,0.1427,0.1312,0.1205,
0.1105,0.1010,0.0919,0.0832,0.0748,0.0667,0.0588,0.0511,0.0436,0.0361,
0.0288,0.0215,0.0143,0.0071,
0.3751,0.2574,0.2260,0.2032,0.1847,0.1691,0.1554,0.1430,0.1317,0.1212,
0.1113,0.1020,0.0932,0.0846,0.0764,0.0685,0.0608,0.0532,0.0459,0.0386,
0.0314,0.0244,0.0174,0.0104,0.0035},
      Gamma[] = {
  0.625,1.107,1.530,2.010,2.356,2.696,2.968,3.262,3.485,3.731,3.936,
  4.155,4.373,4.567,4.713,4.885,5.018,5.153,5.291,5.413,5.508,5.605,
  5.704,5.803,5.905,5.988,6.074,6.160,6.248,6.324,6.402,6.480,6.559,
  6.640,6.721,6.803,6.887,6.961,7.035,7.111,7.188,7.266,7.345,7.414,
  7.484,7.555,7.615,7.677},
      Eta[] = {
  0.386,0.714,0.935,1.138,1.245,1.333,1.400,1.471,1.515,1.571,1.613,
  1.655,1.695,1.724,1.739,1.770,1.786,1.802,1.818,1.835,1.848,1.862,
  1.876,1.890,1.905,1.915,1.934,1.949,1.965,1.976,1.988,2.000,2.012,
  2.024,2.037,2.049,2.062,2.075,2.088,2.101,2.114,2.128,2.141,2.155,
  2.169,2.183,2.198,2.212},
      Epsylon[] = {
  0.7500,0.6297,0.5521,0.4963,0.4533,0.4186,0.3900,0.3660,0.3451,0.3270,
  0.3111,0.2969,0.2842,0.2727,0.2622,0.2528,0.2440,0.2359,0.2264,0.2207,
```

```
  0.2157,0.2106,0.2063,0.2020,0.1980,0.1943,0.1907,0.1872,0.1840,0.1811,
  0.1781,0.1755,0.1727,0.1702,0.1677,0.1656,0.1633,0.1612,0.1591,0.1572,
  0.1552,0.1534,0.1516,0.1499,0.1482,0.1466,0.1451,0.1436};

///////////////
// Копия выборки
x = new double[n];
for (i = 0; i < n; i++)
  x[i] = xx->Array[i];

////////////////////
// Сортировка выборки
SortArrayUp (x,n);

//////////////////////////
// Определение знаменателя
for (i = 0; i < n; i++)
  {
   b += x[i];
   d += x[i] * x[i];
  }
d -= b * b / n;
if (d == 0)
  {
   delete [] x;
   return 0;
  }

//////////////////////
// Определение числителя
for (i = 1; i < n - 2; i++)
  for (j = 0; j <= i / 2; j++)
    s++;
for (i = 0, j = n - 1, b = 0; i < n / 2; i++)
  b += a[s++] * (x[j--] - x[l++]);

//////////////////////////////
// Вычисление статистики критерия
```

```
 w = b * b / d;

 delete [] x;

 ////////////////////////////////////////////////
 // Вычисление нормальной аппроксимации P-значения
 return 1 - NormalDistribution(fabs(-Gamma[n - 3] + Eta[n - 3] *
        log ((w - Epsylon[n - 3]) / (1.0 - w))));
}
```

Критерии отклонения от нормального распределения

Существует группа критериев, которые позволяют оценить отклонение некоторых параметров эмпирического распределения от параметров нормального распределения. Наиболее распространены критерий коэффициента асимметрии (третий нормированный центральный момент) и критерий эксцесса (четвертый нормированный центральный момент). Асимметрия характеризует несимметричность распределения случайной величины. Эксцесс характеризует частоту появления удаленных от среднего значений. Для нормального распределения асимметрия равна нулю, эксцесс — трем. При вычислении эксцесса от полученного значения часто отнимают число три. Коэффициент асимметрии и эксцесс — величины, не зависящие от выбора начала отсчета и от единиц измерения случайной величины. Они вычисляются по формулам, соответственно,

$$k_3 = \frac{M(X-\mu)^3}{\sigma^3},$$

$$k_4 = \frac{M(X-\mu)^4}{\sigma^4} - 3,$$

где μ — математическое ожидание случайной величины X,

σ^2 — дисперсия.

Выборочные коэффициент асимметрии и эксцесс вычисляются по формулам выборочных моментных отношений, соответственно,

$$b_1 = \frac{k_3}{S^3},$$

и $b_2 = \dfrac{k_4}{S^4}$,

где S^2 — выборочная дисперсия, вычисляемая по формуле

$$S^2 = \frac{1}{n-1}\sum_{i=1}^{n}(X_i - \overline{X})^2 ,$$

n — численность эмпирической выборки,

$\overline{X}$ — выборочное среднее,

$$k_3 = \frac{n}{(n-1)(n-2)}\sum_{i=1}^{n}(X_i - \overline{X})^3 ,$$

$$k_4 = \frac{n(n+1)\sum_{i=1}^{n}(X_i - \overline{X})^4 - 3(n-1)\left\{\sum_{i=1}^{n}(X_i - \overline{X})^2\right\}^2}{(n-1)(n-2)(n-3)} .$$

Статистики b_1 и b_2 для большой численности выборки распределены асимптотически нормально с нулевым средним и дисперсиями, соответственно:

$$Db_1 = \frac{6n(n-1)}{(n-2)(n+1)(n+3)} \text{ и } Db_2 = \frac{24n(n-1)^2}{(n-3)(n-2)(n+3)(n+5)} .$$

Применение критерия коэффициента асимметрии и критерия эксцесса рекомендуется для проверки отклонения от нормальности, например, при решении вопроса о применении критерия Стьюдента, который можно рассчитать с помощью пакета анализа Microsoft Excel. Для выборок, отличающихся от нормальных выборок, результаты применения критерия Стьюдента будут близки к верным результатам, если эксцесс и коэффициент асимметрии анализируемых выборок — такие же, как у нормальных выборок.

Листинг 3.5

```
double __declspec(dllexport) SkewnessTest (FP *xx)
//
// Функция вычисления критерия коэффициента асимметрии для проверки
// нормальности распределения в случае сложной гипотезы.
// Обозначения:
//  xx — эмпирическая выборка.
// Возвращаемое значение:
//  P-значение статистики критерия.
//
{
 short i,                          // Счетчик цикла
```

```
      n = xx->rows * xx->columns; // Численность выборки
 double m = 0,                    // Выборочное среднее
        s = 0,                    // Стандарт
        *x,                       // Копия выборки
        g = 0;                    // Статистика критерия

 ////////////////////////////////
 // Оценка параметров распределения
 for (i = 0; i < n; i++)
   {
    m += x[i];
    s += x[i] * x[i];
   }
 m /= n;
 s = (s - n * m * m) / (n - 1);
 if (s == 0)
   {
    delete [] x;
    return 0;
   }

 ////////////////////////////////
 // Вычисление статистики критерия
 for (i = 0; i < n; i++)
   g += (x[i] - m) * (x[i] - m) * (x[i] - m);
 g = g * n / (n - 1) / (n - 2) / s / sqrt (s);

 delete [] x;

 ////////////////////////////////////////////////
 // Вычисление нормальной аппроксимации P-значения
 return 1 - NormalDistribution (fabs(g) / sqrt (6.0 * n *
        (n - 1) / (n - 2) / (n + 1) / (n + 3)));
}

double __declspec(dllexport) KurtosisTest (FP *xx)
//
// Функция вычисления критерия коэффициента эксцесса для проверки
```

```
// нормальности распределения в случае сложной гипотезы.
// Обозначения:
//  xx — эмпирическая выборка.
// Возвращаемое значение:
//  P-значение статистики критерия.
//
{
 short i,                          // Счетчик цикла
       n = xx->rows * xx->columns; // Численность выборки
 double m = 0,                     // Выборочное среднее
        s = 0,                     // Стандарт
        *x,                        // Копия выборки
        b = 0,                     // Статистика критерия
        d = 0;                     // Рабочая переменная

 ///////////////
 // Копия выборки
 x = new double[n];
 for (i = 0; i < n; i++)
   x[i] = xx->Array[i];

 ////////////////////////////////
 // Оценка параметров распределения
 for (i = 0; i < n; i++)
   {
    m += x[i];
    s += x[i] * x[i];
   }
 m /= n;
 s = (s — n * m * m) / (n — 1);
 if (s == 0)
   {
    delete [] x;
    return 0;
   }

 ////////////////////////////////
 // Вычисление статистики критерия
```

```
 for (i = 0; i < n; i++)
   {
    b += (x[i] — m) * (x[i] — m) * (x[i] — m) * (x[i] — m);
    d += (x[i] — m) * (x[i] — m);
   }
 b = (b * n * (n + 1) — d * d * 3 * (n — 1)) / (n — 1) / (n — 2) /
     (n — 3) / s / s;

 delete [] x;

 ////////////////////////////////////////////////
 // Вычисление нормальной аппроксимации P-значения
 return 1 — NormalDistribution (fabs(b) /  sqrt (24.0 * n *
        (n — 1) * (n — 1) / (n — 3) / (n — 2) / (n + 3) / (n + 5)));
}
```

Многомерные аналоги критериев коэффициента асимметрии и эксцесса предложены Мардиа. Вычисления производятся, соответственно, по формулам

$$b_{1.d} = \frac{1}{n^2}\sum_{i=1}^{n}\sum_{j=1}^{n}\left[(X_i - \overline{X})' S^{-1} (X_j - \overline{X})\right]^3$$

и $b_{2.d} = \frac{1}{n}\sum_{i=1}^{n}\left[(X_i - \overline{X})' S^{-1} (X_i - \overline{X})\right]^2$,

n — численность эмпирической выборки,

d — размерность многомерной (d-мерной) выборки $X_j, j = 1,2,...,n,$.

S^{-1} — матрица, обратная дисперсионно-ковариационной матрице,

$\overline{X}$ — d–мерный вектор среднего значения, вычисленный по d-мерной выборке,

штрих означает операцию транспонирования.

Статистика $\frac{n}{6} b_{1,d}$ распределена асимптотически как хи-квадрат (см. *приложение 2*) с параметром $d(d+1)(d+2)/6$. Статистика $b_{2,d}$ распределена асимптотически нормально со средним $d(d+2)$ и дисперсией $8d(d+2)/n$.

Листинг 3.6

```
double __declspec(dllexport) MardiaSkewnessTest (FP *xx)
//
// Функция вычисления критерия коэффициента асимметрии для проверки
// нормальности многомерного распределения.
// Обозначения:
//  xx — многомерная эмпирическая выборка.
// Возвращаемое значение:
//  P-значение статистики критерия.
//
{
 short i,j,l,           // Счетчик цикла
       m1,m2,           // Индекс
       n = xx->rows,    // Численность выборки
       k = xx->columns; // Размерность выборки
 double *m,             // Вектор выборочного среднего
        *s,             // Ковариационная матрица
        *t,             // Обратная ковариационная матрица
        *x,             // Копия выборки
        *mi,*mj,*t1,    // Рабочие массивы
        g = 0;          // Статистика критерия

 ///////////////
 // Копия выборки
 x = new double[n * k];
 for (i = 0; i < n * k; i++)
   x[i] = xx->Array[i];

 ////////////////////////////////////////////
 // Динамическая память для рабочих массивов
 // (результат операции не проверяется)
 s = new double[k * k];
 t = new double[k * k];

 ////////////////////////////////
 // Оценка параметров распределения
 CovMatrix (x,s,n,k);
 if (InverseMatrix (s,t,k) == -2)
```

```
  {
   delete [] x;
   delete [] s;
   delete [] t;
   return 0;
  }

////////////////////////////////////////////
// Динамическая память для рабочих массивов
// (результат операции не проверяется)
m = new double[k];
mi = new double[k];
mj = new double[k];
t1 = new double[1];

//////////////////////////////////
// Оценка параметров распределения
MeanVector (x,m,n,k);

////////////////////////////////
// Вычисление статистики критерия
for (i = 0; i < n; i++)
  {
   for (l = 0, m2 = i; l < k; l++)
     {
      mi[l] = x[m2] - m[l];
      m2 += n;
     }
   MatrixTMultiplyMatrix (mi,t,s,1,k,k);
   for (j = 0; j < n; j++)
     {
      for (l = 0, m1 = j; l < k; l++)
        {
         mj[l] = x[m1] - m[l];
         m1 += n;
        }
      MatrixMultiplyMatrix (s,mj,t1,1,k,1);
      g += t1[0] * t1[0] * t1[0];
     }
  }
```

```
  ////////////////////////////////////
 // Освобождение динамической памяти
 delete [] x;
 delete [] s;
 delete [] t;
 delete [] m;
 delete [] mi;
 delete [] mj;
 delete [] t1;

 ////////////////////////
 // Вычисление P-значения
 g = ChiSquareDistribution (g / 6 / n, k * (k + 1) * (k + 2) / 6);
 if (g > 0.5)
   return 1 — g;
 else
   return g;
}

double __declspec(dllexport) MardiaKurtosisTest (FP *xx)
//
// Функция вычисления критерия эксцесса для проверки нормальности
// многомерного распределения.
// Обозначения:
//  xx — многомерная эмпирическая выборка.
// Возвращаемое значение:
//  P-значение статистики критерия.
//
{
 short i,j,              // Счетчик цикла
       m1,               // Индекс
       n = xx->rows,     // Численность выборки
       k = xx->columns;  // Размерность выборки
 double *m,              // Вектор выборочного среднего
        *s,              // Ковариационная матрица
        *t,              // Обратная ковариационная матрица
        *x,              // Копия выборки
        *mi,*t1,         // Рабочие массивы
        g = 0;           // Статистика критерия
```

```
////////////////
// Копия выборки
x = new double[n * k];
for (i = 0; i < n * k; i++)
  x[i] = xx->Array[i];

//////////////////////////////////////////
// Динамическая память для рабочих массивов
// (результат операции не проверяется)
s = new double[k * k];
t = new double[k * k];

//////////////////////////////////
// Оценка параметров распределения
CovMatrix (x,s,n,k);
if(InverseMatrix (s,t,k) == -2)
  {
   delete [] x;
   delete [] s;
   delete [] t;
   return 0;
  }

//////////////////////////////////////////
// Динамическая память для рабочих массивов
// (результат операции не проверяется)
m = new double[k];
mi = new double[k];
t1 = new double[1];

//////////////////////////////////
// Оценка параметров распределения
MeanVector (x,m,n,k);

//////////////////////////////////
// Вычисление статистики критерия
```

```
 for (i = 0; i < n; i++)
   {
    for (j = 0, m1 = i; j < k; j++)
      {
       mi[j] = x[m1] — m[j];
       m1 += n;
      }
    MatrixTMultiplyMatrix (mi,t,s,1,k,k);
    MatrixMultiplyMatrix (s,mi,t1,1,k,1);
    g += t1[0] * t1[0];
   }

 ////////////////////////////////////
 // Освобождение динамической памяти
 delete [] x;
 delete [] s;
 delete [] t;
 delete [] m;
 delete [] mi;
 delete [] t1;

 ////////////////////////
 // Вычисление P-значения
 return 1 — NormalDistribution (fabs(g / n — k * (k + 2)) /
            sqrt (8.0 * k * (k + 2) / n));
}
```

Обработка пропущенных данных

В практической деятельности возникают ситуации, когда часть экспериментальных статистических данных оказывается утраченной. Существуют ряд методов специальной обработки пропущенных значений.

Особенности решения

Применяемые алгоритмы просты, поэтому программа написана на языке VBA.

Научные алгоритмы

Перед применением методов обработки пропущенных данных исследователь должен точно знать, что данные на местах пропусков обязательно должны быть, но отсутствуют. Речь не идет о пропуске, вызванном условиями эксперимента.

Обработку таких данных производят с помощью методов анализа цензурированных выборок. Совместный анализ матрицы исходных данных и матрицы пропусков вряд ли уместен, т. к. предполагает связь изучаемого процесса с причиной возникновения пропусков. Мы же полагаем, что эти явления независимы.

Обработка пропущенных значений производится методами:

- игнорирования пропусков;
- заполнения средним значением;
- заполнения регрессионными значениями;
- заполнения случайными значениями.

Игнорирование пропусков

Самым простым способом обработки пропущенных значений является их игнорирование. На месте пропущенных значений, если не делать никаких дополнительных предположений, могут стоять *любые* значения.

Метод игнорирования пропусков в нашей программе работает с одним столбцом данных, осуществляя сдвиг ячеек вверх. В результате обработки происходит уменьшение численности выборки на число пропущенных ячеек.

Метод рекомендуется для выборок с малым числом пропусков.

Заполнение средним значением

Заполнение пропущенных значений некоторыми допустимыми значениями является распространенным способом их восстановления. В качестве допустимого значения обычно выбирается выборочное среднее значение, вычисляемое по формуле

$$\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i .$$

В данной формуле в суммировании участвуют только присутствующие варианты выборки, а n является разностью между численностью выборки и числом пропущенных значений.

Метод удобен тем, что в результате его применения важнейшая статистическая мера положения, а именно среднее значение, в выборке с заполненными пропусками не изменяется, по сравнению со средним значением, вычисленным для выборки с пропущенными значениями (исходной выборки). Однако метод приводит к заниженной выборочной оценке дисперсии с заполненными значениями относительно дисперсии исходной выборки. Другим недостатком является искажение эмпирического распределения выборки, независимо от типа эмпирического распределения исходной выборки.

Метод заполнения средним значением рекомендуется применять для больших выборок с малым числом пропусков. Выборка должна принадлежать количественной шкале. Данный метод представляет собой частный случай заполнения регрессионными значениями.

Заполнение регрессионными значениями

Заполнение регрессионными значениями базируется на идее заполнения пропусков, основываясь на информации о связи выборки $x_i, i = 1,2,...,n$ с другой выборкой $y_i, i = 1,2,...,n$. При коэффициенте корреляции между двумя выборками, близком к единице, можно предположить существующую тесную регрессионную зависимость между *x* и *y*. Коэффициент корреляции Пирсона вычисляется по формуле

$$r = \frac{Cov(x, y)}{\sqrt{Cov(x,x)Cov(y,y)}},$$

где $Cov(.,.)$ — выборочная ковариация, вычисляемая по формуле

$$Cov(x, y) = \frac{1}{n-1}\sum_{i=1}^{n}\left(x_i - \bar{x}\right)\sum_{i=1}^{n}\left(y_i - \bar{y}\right),$$

n — численность каждой эмпирической выборки,

$\bar{x}$ — среднее значение выборки $x_i, i = 1,2,...,n$,

$\bar{y}$ — среднее значение выборки $y_i, i = 1,2,...,n$.

После вычисления коэффициента корреляции пользователю предлагается принять решение, выполнять дальнейшие вычисления или нет в зависимости от величины коэффициента корреляции. Если принято решение продолжить, производится вычисление линейной регрессии. В простейшем случае регрессионную зависимость можно представить как

$$y = ax + b,$$

т.е линейную.

Коэффициенты линейной регрессии вычисляются по формулам

$$a = \frac{\sum_{i=1}^{n} x_i y_i - \bar{x}\sum_{i=1}^{n} y_i}{\sum_{i=1}^{n} x_i^2 - \bar{x}\sum_{i=1}^{n} x_i},$$

$$b = \bar{y} - a\bar{x}.$$

Если установлена линейная зависимость между переменными, можно восстановить отсутствующие значения, решив обратную задачу:

$$x = \frac{y - b}{a}.$$

Данные вычисления подходят, конечно, если в парах значений y известны, а x утрачены. Если утраченными оказываются оба значения, заполнение пропусков производится средним значением.

В листинге 3.7 представлены необходимые для реализации алгоритма функции.

Листинг 3.7

```
Private Function Pearson(X() As Double, Y() As Double, N As Integer) As
Double
'
' Функция вычисляет значение коэффициента корреляции Пирсона.
' Обозначения:
'  X — первая выборка,
'  Y — вторая выборка,
'  n — численность каждой выборки.
' Возвращаемое значение:
'  значение выборочного коэффициента корреляции.
'
 Pearson = Cov(X(), X(), N) * Cov(Y(), Y(), N)
 If Not Pearson = 0 Then
   Pearson = Cov(X(), Y(), N) / Sqr(Pearson)
 End If
End Function
Private Function Cov(X() As Double, Y() As Double, N As Integer) As
Double
'
' Функция вычисляет значение выборочной ковариации.
```

```
' Обозначения:
'  X — первая сравниваемая выборка,
'  Y — вторая сравниваемая выборка,
'  N — численность каждой выборки.
' Возвращаемое значение:
'  значение выборочной ковариации.
'
 Dim I As Integer ' Счетчик
 Dim Xm As Double ' Среднее первой выборки
 Dim Ym As Double ' Среднее первой выборки
 Xm = VectorSum(X(), N) / N
 Ym = VectorSum(Y(), N) / N
 Cov = 0
 For I = 1 To N
  Cov = Cov + (X(I) — Xm) * (Y(I) — Ym)
 Next I
 Cov = Cov / (N — 1)
End Function
Private Function VectorSum(X() As Double, N As Integer) As Double
'
' Функция вычисления суммы элементов массива.
' Обозначения:
'  X — массив,
'  Y — длина массива.
' Возвращаемое значение:
'  сумма элементов массива.
'
 Dim I As Integer ' Счетчик
 VectorSum = 0
 For I = 1 To N
  VectorSum = VectorSum + X(I)
 Next I
End Function
```

Заполнение случайными значениями

Метод заполнения случайными значениями, вопреки наименованию, производит наиболее корректное заполнение пропусков в смысле сохранения несмещенности статистических параметров выборки. В основе метода лежит

предположение, что меры положения (средние значения) и меры разброса (средние квадратические отклонения) в присутствующей и в пропущенной частях выборки равны. Реализация заключается в определении типа теоретического распределения эмпирической выборки и генерации случайных значений отсутствующей части выборки с тем же теоретическим распределением.

Нами решается более частная задача: предполагается, что распределение является нормальным. Генерация выборки, распределенной по нормальному закону $N(\bar{x},\sigma^2)$, производится с помощью формулы

$$y_j = \bar{x} + \sigma x_j,\ j = 1,2,\ldots,m,$$

где $\bar{x}$ — выборочное среднее значение присутствующей части выборки,

σ — выборочное среднее квадратичное отклонение, квадратный корень из дисперсии присутствующей части выборки,

$x_j, j = 1,2,...,m,$ — выборка, сгенерированная по стандартному нормальному закону $N(0,1)$,

m — число пропущенных значений.

Выборочное среднее определяется по формуле

$$\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i.$$

Выборочная дисперсия вычисляется по формуле

$$\sigma^2 = \frac{1}{n-1}\sum_{i=1}^{n}\left(x_i - \bar{x}\right)^2.$$

В суммировании участвуют только присутствующие варианты выборки, а n является разностью между численностью выборки и числом пропущенных значений.

Обратная функция стандартного нормального распределения представляет собой решение задачи, обратной

$$\Phi(x) = \frac{1}{\sqrt{2\pi}}\int_{-\infty}^{x} e^{-y^2/2}dy,$$

т. е. по известному $\Phi(x)$ определяется x.

Рассмотренный метод рекомендуется для выборок с большим числом пропусков, причем выборка должна принадлежать количественной шкале в случае непрерывных признаков.

Основой реализации является функция, представленная в листинге 3.8.

Листинг 3.8

```
Private Function InvNormal(P As Double) As Double
'
' Функция вычисления обратной функции стандартного
' нормального распределения.
' Обозначения:
'  p — доверительный уровень.
' Возвращаемое значение:
'  функция распределения.
'
 Dim Flag As Integer ' Параметр учета симметрии распределения
 Dim T As Double     ' Рабочая переменная

 Flag = 1

 If P < 0.5 Then
   P = 1 — P
 ElseIf P = 0.5 Then
   InvNormal = 0
   Exit Function
 Else
   Flag = 0
 End If
 '
 ' Разложение в ряд
 '
 T = Sqr(-2 * Log(1 — P))
 T = T — (2.515517 + 0.802853 * T + 0.010328 * T * T) / _
 (1 + 1.432788 * T + 0.189269 * T * T + 0.001308 * T * T * T)

 If Flag = 0 Then
   InvNormal = T
 Else
   InvNormal = -T
 End If
End Function
```

Обработка выбросов

В практической деятельности экспериментальные данные оказываются «засоренными» резко выделяющимися значениями (выбросами). Выбросы обычно означают грубые ошибки измерений, возникающие в результате просчета, неправильного чтения показаний прибора и т. п.

Существуют ряд методов, помимо непосредственной проверки результатов наблюдений, позволяющих специальным образом обработать выбросы. Данные методы основаны на критериях исключения минимального (максимального) наблюдения.

Мы полагаем, что, несмотря на достаточную теоретическую проработку методов обработки выбросов, каждый выброс, прежде чем будет исключен из массива экспериментальных данных, должен получить объяснение.

Особенности решения

Применяемые алгоритмы позволяют полностью решить задачу в рамках VBA.

Научные алгоритмы

Предположим нормальное распределение исходных данных. Кроме того, в наших реализациях мы рассматриваем случаи, когда основные статистические параметры совокупности (мера положения — среднее значение, а также мера разброса — дисперсия) неизвестны и вычисляются по выборке.

Обработка выбросов производится методами:

- критерий Смирнова–Граббса;
- критерий Титьена–Мура;
- правило Томпсона.

Основные функции, реализующие прикладную логику программы, показаны далее, при описании соответствующих методов. Различные вспомогательные функции не показаны. Их полные тексты имеются на прилагаемом компакт-диске.

Критерий Смирнова–Граббса

Критерий Смирнова–Граббса предназначен для исключения одного максимального или минимального наблюдения. Статистика критерия основана, соответственно, на величине

$$T_N = \frac{\max\limits_i x_i - \bar{x}}{s} \text{ или } T_N = \frac{\bar{x} - \min\limits_i x_i}{s},$$

где $x_1, x_2, ..., x_N$ — результаты N-наблюдений,

Выборочное среднее значение рассчитывается по формуле

$$\bar{x} = \frac{1}{N}\sum_{i=1}^{N} x_i,$$

а выборочная дисперсия

$$s^2 = \frac{1}{N}\sum_{i=1}^{N}\left(x_i - \bar{x}\right)^2.$$

Величина статистики критерия сравнивается с табличным значением. При значении, большем табличного, наблюдение исключается.

Находит применение еще один метод исключения максимального или минимального наблюдения — критерий Граббса, связанный с представленным здесь критерием простой формулой

$$G_N = 1 - \frac{1}{N-1}T_N^2,$$

поэтому дающий аналогичные результаты.

Примечание

Казалось бы, для исключения нескольких выбросов можно применить представленный критерий несколько раз. Это не так. Критерий Смирнова–Граббса, как и критерий Граббса, в общем случае не годится для исключения k ($k > 1$) выбросов из–за так называемого маскирующего эффекта. Для исключения нескольких выбросов применяется критерий Титьена–Мура, описанный далее.

Фрагмент, представленный на листинге, вычисляет критерий Смирнова–Граббса.

Листинг 3.9

```
'
' Критерий Смирнова-Граббса
'
Set RData = Range(Data)
Value = RData.Cells(1, 1)
Row = 1
Col = 1
```

```
    If Box4 = True Then
      For I = 1 To RData.Rows.Count
        For J = 1 To RData.Columns.Count
          If RData.Cells(I, J) > Value Then
            Value = RData.Cells(I, J)
            Row = I
            Col = J
          End If
        Next J
      Next I
    Else
      For I = 1 To RData.Rows.Count
        For J = 1 To RData.Rows.Count
          If RData.Cells(I, J) < Value Then
            Value = RData.Cells(I, J)
            Row = I
            Col = J
          End If
        Next J
      Next I
    End If
    T = Abs(Value - Mean) / Sqr(Var)
    If T > SmirnovGrubbsDistribution(N) Then
      RData.Cells(Row, Col).ClearContents
    End If
```

Функция, представленная на следующем листинге, вычисляет критическое значение критерия Смирнова–Граббса. Аппроксимация статистических таблиц выполнена в Microsoft Excel.

Листинг 3.9 (продолжение)

```
Private Function SmirnovGrubbsDistribution(N As Integer)
'
' Функция вычисляет критическое значение критерия
' Смирнова-Граббса.
' Обозначения:
'  N — численность выборки.
' Возвращаемое значение:
'  критическое значение для уровня значимости 0,05.
```

```
'
 Dim Y() As Variant ' Таблица критических значений
 Dim U As Double    ' Параметр

 If N > 50 Then
   U = InvNormal(1 - 0.05 / N)
   SmirnovGrubbsDistribution = U * Sqr(2 * (N - 1) / _
   (2 * N - 5 + U * U + (3 + U * U + 2 * U * U * U * U) / 6 / _
   (2 * N - 5)))
 Else
   Y = Array(1.412, 1.689, 1.869, 1.996, 2.093, 2.172, 2.237, _
             2.294, 2.343, 2.387, 2.426, 2.461, 2.493, 2.523, _
             2.551, 2.577, 2.6, 2.623, 2.644, 2.664, 2.683, _
             2.701, 2.718, 2.734, 2.749, 2.764, 2.778, 2.792, _
             2.805, 2.818, 2.83, 2.842, 2.853, 2.864, 2.874, _
             2.885, 2.894, 2.904, 2.913, 2.922, 2.931, 2.94, _
             2.948, 2.956, 2.964, 2.972, 2.98, 2.987)
   SmirnovGrubbsDistribution = Y(N - 2)
 End If
End Function
```

Критерий Титьена–Мура

Критерий Титьена–Мура является обобщением критерия Граббса на случай нескольких выбросов. Критерий Титьена–Мура позволяет бороться с маскирующим эффектом. Статистика критерия исключения k наибольших или наименьших значений основана, соответственно, на величине

$$L_k = \frac{\sum_{i=1}^{N-k}\left(x_i - \bar{x}_k\right)^2}{\sum_{i=1}^{N}\left(x_i - \bar{x}\right)^2} \text{ или } \widetilde{L}_k = \frac{\sum_{i=k+1}^{N}\left(x_i - \widetilde{x}_k\right)^2}{\sum_{i=1}^{N}\left(x_i - \bar{x}\right)^2},$$

где $x_1, x_2, ..., x_N$ — упорядоченные по возрастанию результаты N наблюдений.

Выборочное среднее значение (для всей выборки) рассчитывается как в предыдущем разделе, а средние значения после отбрасывания k наибольших или наименьших значений рассчитываются, соответственно, по формулам

$$\bar{x}_k = \frac{1}{N-k}\sum_{i=1}^{N-k} x_i \quad \text{или} \quad \tilde{x}_k = \frac{1}{N-k}\sum_{i=k+1}^{N} x_i .$$

Величина статистики критерия сравнивается с табличным значением. При величине статистики, меньшей табличной, наблюдение исключается.

Фрагмент, представленный на листинге, вычисляет критерий Титьена–Мура.

Листинг 3.10

```
'
' Критерий Титьена-Мура
'
K = Tex1
If K > N / 2 Then
  K = N / 2
End If
If Box4 = True Then
  Call SortArrayUp(X(), N, Ix(), 1)
Else
  Call SortArrayDown(X(), N, Ix(), 1)
End If
Mean = 0
For I = 1 To N - K
  Mean = Mean + X(I)
Next I
Mean = Mean / (N - K)
T = 0
For I = 1 To N - K
  T = T + (X(I) - Mean) * (X(I) - Mean)
Next I
T = T / Var / N
If T < TietjenMooreDistribution(N, K) Then
  For I = 1 To K
    X2(Ix(N + 1 - I)) = ""
  Next I
End If
S = 1
For I = 1 To RData.Rows.Count
```

```
      For J = 1 To RData.Columns.Count
        RData.Cells(I, J) = X2(S)
        S = S + 1
      Next J
    Next I
```

Функция, представленная на следующем листинге, вычисляет критическое значение критерия Титьена–Мура. Аппроксимация статистических таблиц выполнена в Microsoft Excel.

Листинг 3.10 (продолжение)

```
Private Function TietjenMooreDistribution(N As Integer, K As Integer)
'
' Функция вычисляет критическое значение критерия
' Титьена-Мура.
' Обозначения:
'  N — численность выборки,
'  K — число выбросов.
' Возвращаемое значение:
'  критическое значение для уровня значимости 0,05.
'
 Dim A As Double     ' Коэффициент аппроксимации
 Dim B As Double     ' Коэффициент аппроксимации
 Dim Y1() As Variant ' Таблица критических значений для K = 1
 Dim Y2() As Variant ' Таблица критических значений для K = 2
 Dim Y3() As Variant ' Таблица критических значений для K = 3
 Dim Y4() As Variant ' Таблица критических значений для K = 4
 Dim Y5() As Variant ' Таблица критических значений для K = 5
 Dim Y6() As Variant ' Таблица критических значений для K = 6
 Dim Y7() As Variant ' Таблица критических значений для K = 7
 Dim Y8() As Variant ' Таблица критических значений для K = 8
 Dim Y9() As Variant ' Таблица критических значений для K = 9

 Y1 = Array(0.003, 0.051, 0.125, 0.203, 0.273, 0.326, 0.372, 0.418, _
            0.454, 0.489, 0.517, 0.54, 0.556, 0.575, 0.594, 0.608, 0.624)
 Y2 = Array(0.001, 0.018, 0.055, 0.106, 0.146, 0.194, 0.233, 0.27, _
            0.305, 0.337, 0.363, 0.387, 0.41, 0.427, 0.447, 0.462)
 Y3 = Array(0.01, 0.032, 0.064, 0.099, 0.129, 0.162, 0.196, 0.224, _
```

```
            0.25, 0.276, 0.3, 0.322, 0.337, 0.354)
Y4 = Array(0.022, 0.045, 0.07, 0.098, 0.125, 0.15, 0.174, 0.197, _
            0.219, 0.24, 0.259, 0.277)
Y5 = Array(0.034, 0.054, 0.076, 0.098, 0.122, 0.14, 0.159, 0.181, _
            0.2, 0.209)
Y6 = Array(0.042, 0.06, 0.079, 0.097, 0.115, 0.136, 0.154, 0.168)
Y7 = Array(0.05, 0.066, 0.082, 0.1, 0.116, 0.13)
Y8 = Array(0.055, 0.072, 0.086, 0.099)
Y9 = Array(0.062, 0.074)

If N < 20 Then
  '
  ' Точные табличные значения
  '
  If K = 1 Then
    TietjenMooreDistribution = Y1(N - 2)
  ElseIf K = 2 Then
    TietjenMooreDistribution = Y2(N - 3)
  ElseIf K = 3 Then
    TietjenMooreDistribution = Y3(N - 5)
  ElseIf K = 4 Then
    TietjenMooreDistribution = Y4(N - 7)
  ElseIf K = 5 Then
    TietjenMooreDistribution = Y5(N - 9)
  ElseIf K = 6 Then
    TietjenMooreDistribution = Y6(N - 11)
  ElseIf K = 7 Then
    TietjenMooreDistribution = Y7(N - 13)
  ElseIf K = 8 Then
    TietjenMooreDistribution = Y8(N - 15)
  Else
    TietjenMooreDistribution = Y9(N - 17)
  End If
Else
  '
  ' Аппроксимация логарифмической функцией
  '
  If K = 1 Then
```

```
    A = 0.19418112
    B = -0.06575383
  ElseIf K = 2 Then
    A = 0.25812829
    B = 0.28278038
  ElseIf K = 3 Then
    A = 0.29275863
    B = 0.49380456
  ElseIf K = 4 Then
    A = 0.31458311
    B = 0.63943566
  ElseIf K = 5 Then
    A = 0.3245631
    B = 0.73176359
  ElseIf K = 6 Then
    A = 0.33029662
    B = 0.79975254
  ElseIf K = 7 Then
    A = 0.32887508
    B = 0.83561828
  ElseIf K = 8 Then
    A = 0.32971362
    B = 0.87499117
  ElseIf K = 9 Then
    A = 0.32553272
    B = 0.89145385
  Else
    A = 0.31999176
    B = 0.89985535
  End If
  TietjenMooreDistribution = A * Log(N) - B
 End If
End Function
```

Правило Томпсона

В правиле Томпсона для исключения резко выделяющихся наблюдений (выбросов) используется статистика

$$t_i = \frac{|x_i - \bar{x}|}{s},\ i = 1,2,...,N,$$

где $x_1, x_2, ..., x_N$ — результаты N наблюдений.

Выборочная дисперсия (может применяться также несмещенная оценка, отличающаяся делителем $N - 1$) вычисляется по формуле

$$s^2 = \frac{1}{N}\sum_{i=1}^{N}(x_i - \bar{x})^2.$$

Величина статистики критерия подсчитывается для каждого наблюдения и сравнивается с табличным значением. Аппроксимация таблиц выполнена в Microsoft Excel. При значении статистики, большем табличного, наблюдение исключается.

В общем случае правило Томпсона не годится для исключения всех или нескольких выбросов из-за маскирующего эффекта. Для исключения нескольких выбросов применяется критерий Титьена–Мура.

Фрагмент, представленный на листинге, выполняет исключение выбросов по правилу Томпсона.

Листинг 3.11

```
'
' Правило Томпсона
'
Set RData = Range(Data)
Row = 1
Col = 1
For I = 1 To RData.Rows.Count
  For J = 1 To RData.Columns.Count
    T = Abs(RData.Cells(I, J) - Mean) / Sqr(Var)
    If T > ThompsonDistribution(N) Then
      RData.Cells(I, J).ClearContents
    End If
  Next J
Next I
```

Функция, представленная на следующем листинге, вычисляет критическое значение правила Томпсона. Аппроксимация статистических таблиц выполнена в Microsoft Excel.

Листинг 3.11 (продолжение)

```
Private Function ThompsonDistribution(N As Integer)
'
' Функция вычисляет критическое значение правила
' Томпсона.
' Обозначения:
'  N — численность выборки.
' Возвращаемое значение:
'  критическое значение для уровня значимости 0,05.
'
 Dim Y() As Variant ' Таблица критических значений
 Dim M As Integer   ' Параметр

 M = N — 2

 If M > 50 Then
   ThompsonDistribution = 0.00645052 * Log(M) + 1.92620846
 Else
   Y = Array(1.41, 1.645, 1.757, 1.814, 1.848, 1.87, 1.885, 1.896, _
             1.904, 1.91, 1.915, 1.92, 1.923, 1.926, 1.929, 1.931, _
             1.933, 1.934, 1.936, 1.937, 1.938, 1.939, 1, 940, 1.941, _
             1.942, 1.943, 1.943, 1.944, 1.945, 1.945, 1.946, 1.946, _
             1.947, 1.947, 1.947, 1.948, 1.948, 1.948, 1.949, 1.949, _
             1.949, 1.95, 1.95, 1.95, 1.95, 1.951, 1.951, 1.951, _
             1.951, 1.951)
   ThompsonDistribution = Y(M)
 End If
End Function
```

Руководство пользователя

Модуль проверки нормальности распределения

Для работы с модулем проверки нормальности распределения (NDC) выберите из меню программы Microsoft Excel добавленный программным обеспечением AtteStat новый пункт **AtteStat ▸ Модуль NDC ▸ Проверка нормальности**. На экране появится окно, изображенное на рис. 3.1.

Рис. 3.1. Окно модуля проверки нормальности.

1. Выберите или введите интервал одномерной или многомерной эмпирической выборки.
2. Выберите метод проверки нормальности. Для расчета может быть выбран только один из предлагаемых методов.
3. Выберите или оставьте по умолчанию стандартное пороговое значение уровня значимости для принятия решения. В конце главы в примере представлена информация о том, как пользоваться пороговым значением.
4. Нажмите кнопку **Выполнить** расчет либо **Отмена**, если не выполнять расчет.

Имеется альтернативный способ запуска функций NDC для опытных пользователей. Функции, содержащиеся в NDC, будут доступны для вызова в пункте **Проверка нормальности**. Работа с этими функциями показана на рисунке 3.2, во время работы мастера функций из меню **Вставка ▸ Функция**.

Рис. 3.2. Окно мастера функций

Альтернативный метод позволяет организовать сложные, в том числе циклические, алгоритмы проверки нормальности наборов данных. При этом диагностика ошибок аналогична стандартной и подробно описана в справочной системе электронных таблиц.

Сообщения об ошибках

При ошибках ввода могут выдаваться сообщения следующих типов, как показано на снимках с экрана.

Рис. 3.3. Окно сообщения об ошибке

На рис. 3.3 неверно введен интервал эмпирической выборки. Лучшим способом избежать ошибки является не ввод, а выделение интервала протаскиванием курсора.

Рис. 3.4. Окно сообщения об ошибке

На рис. 3.4 следует проверить исходные данные и заполнить все ячейки, если в ячейке не должно быть данных условиям, введите в эту ячейку нуль.

Рис. 3.5. Окно сообщения об ошибке

На рис. 3.5 тип может быть только числовым. Проще всего выделить интервал ячеек и явно определить их тип как числовой.

Рис. 3.6. Окно сообщения об ошибке

Выбранный на рис. 3.6 интервал эмпирической выборки содержит слишком мало значений. Минимальные численности выборок указаны при описании соответствующих критериев.

Рис. 3.7. Окно сообщения об ошибке

Выбранный на рис. 3.7 интервал эмпирической выборки содержит слишком много значений. Максимальные численности выборок указаны при описании соответствующих критериев.

Модуль обработки пропущенных данных

Для работы с модулем обработки пропущенных данных (MDP) запустите электронные таблицы Microsoft Excel. Выберите из меню программы Microsoft Excel добавленный программным обеспечением AtteStat новый пункт **AtteStat ▸ Модуль MDP ▸ Обработка пропущенных данных**. На экране появится диалоговое окно, изображенное на рис. 3.8.

1. Выберите или введите интервал переменной. Исходным для расчета является эмпирический ряд в виде столбца (но не строки таблицы).

2. Выберите или введите интервал регрессионной переменной, записанной в виде столбца. Данный интервал необходим только если выбран метод **Заполнение регрессионными значениями**.

3. Выберите метод обработки пропущенных значений. Для расчета может быть выбран только один из предлагаемых методов.

4. Нажмите кнопку **Выполнить** расчет либо **Отмена**, если не выполнять расчет.

Рис. 3.8. Окно модуля обработки пропущенных данных

После нажатия кнопки **Выполнить расчет** появится информационное окно (рис. 3.9) с указанием числового значения коэффициента корреляции. Если модуль коэффициента корреляции близок к единице, следует продолжить расчет, нажав кнопку **Да**. В противном случае, от продолжения расчета необходимо воздержаться и применить другой метод расчета.

Рис. 3.9. Информационное окно

Примечание

Вычисления с помощью MDP необратимы. Стандартная команда **Правка ▸ Отменить** не поддерживается. Поэтому перед выполнением вычислений убедитесь, что сохранили данные.

Сообщения об ошибках

При ошибках ввода могут выдаваться диагностические сообщения следующих типов, как показано на снимках с экрана.

Рис. 3.10. Окно сообщения об ошибке

На рис 3.10 неверно введен входной интервал. Лучшим способом избежать ошибки является не ввод, а выделение интервала протаскиванием курсора.

Рис. 3.11. Окно сообщения об ошибке

На рис 3.11 неверно введен интервал регрессии. Лучшим способом избежать ошибки является не ввод, а выделение интервала протаскиванием курсора.

Рис. 3.12. Окно сообщения об ошибке

На рис.3.12 для расчета следует выбрать или ввести входной интервал, а также интервал регрессии размером более одной числовой ячейки (иначе нельзя построить регрессию). Если выбран метод заполнения случайными значениями, необходим входной интервал, содержащий не менее двух числовых ячеек (иначе нельзя вычислить выборочную оценку дисперсии).

Рис. 3.13. Окно сообщения об ошибке

На рис.3.13 необходимо выбрать интервал, содержащий, по меньшей мере, одну ячейку с числовым значением. Некоторые применяемые методы требуют более одной числовой ячейки.

Модуль обработки выбросов

Для работы с модулем обработки выбросов (ОР) выберите из меню программы Microsoft Excel добавленный программным обеспечением AtteStat новый пункт **AtteStat ▸ Модуль ОР ▸ Обработка выбросов**. На экране появится диалоговое окно, изображенное на рис. 3.14.

Рис. 3.14. Окно модуля обработки выбросов

1. Выберите или введите интервал переменной. Исходным для расчета является эмпирический ряд в виде столбца, строки или прямоугольной области электронной таблицы.
2. Выберите метод обработки выбросов. Для расчета может быть выбран только один из предлагаемых методов.
3. Выберите дополнительные параметры: **Проверить максимум** или **Проверить минимум** (действительно для критериев Смирнова–Граббса и Титьена–Мура), а также укажите количество выбросов (действительно для критерия Титьена–Мура).
4. Нажмите кнопку **Выполнить расчет**.

При ошибках, вызванных неверными действиями пользователя, выдаются сообщения об ошибках.

Примечание

Вычисления с помощью ОР необратимы для ваших данных. Стандартная команда **Правка ▸ Отменить** не поддерживается. Поэтому перед выполнением вычислений убедитесь, что сохранили данные.

Сообщения об ошибках

При ошибках ввода могут выдаваться диагностические сообщения следующих типов, как показано на снимках с экрана.

Рис. 3.15. Окно сообщения об ошибке

На рис.3.15 неверно введен входной интервал. Лучшим способом избежать ошибки является не ввод, а выделение интервала протаскиванием курсора.

Рис. 3.16. Окно сообщения об ошибке

На рис. 3.16 необходимо выбрать интервал, содержащий только ячейки с числовым значением. Пропуски также недопустимы.

Рис. 3.17. Окно сообщения об ошибке

Интервал исходных данных на рис. 3.17 должен содержать, по меньшей мере, 3 числовых ячейки.

Рис. 3.18. Окно сообщения об ошибке

Для метода Титьена–Мура на рис. 3.18 необходимо ввести число выбросов, которые предполагается обработать, в интервале от 1 до 10.

Тестирование и пример применения

Для тестирования алгоритмов и демонстрации предложенных решений рассмотрим пример контроля эмпирических и специально сгенерированных выборок на нормальность распределения. Это — крайне важное исследование в любых задачах по обработке экспериментальных данных. Статистически доказанная нормальность распределения позволяет применять большое число алгоритмов обработки, которые требуют нормальности. В то же время недоказанная нормальность заставляет обращаться к непараметрическим методам.

Для тестирования нами будут применены 2 выборки:

- первая выборка — данные о расходе бензина (в галлонах) при поездке из Нью–Йорка в Сан–Франциско десятью автомобилями средних размеров (данные из монографии Хана и Шапиро): 303, 338, 406, 457, 461, 469, 474, 489, 515, 583. Мы, конечно, могли бы сгенерировать случайную выборку любой численности, но нас сейчас интересует подтверждение заранее известного ответа классической задачи;
- вторая выборка — специально подобранный ряд чисел, распределение которых явно нельзя отнести к нормальному распределению: 1; 1; 1; 1; 1; 1; 12; 12; 12; 12; 12; 12.

Результаты расчета сведены в таблицу.

***Таблица 3.1**. Анализ нормальности распределения*

Номер выборки	**P–значение статистики критерия**				
	Критерий Колмогорова	**Критерий Крамера–Мизеса**	**Критерий Андерсона–Дарлинга**	**Критерий Шапиро–Уилка**	**Критерий Эппса–Палли**
1	0,11161	0,24309	0,35323	0,44269	0,40141
2	0,00041	0,00002	0,00003	0,00003	0,02441

К нормальным выборкам на принятом уровне значимости более 0,05 можно отнести выборки, для которых P-значение статистики критерия находится в интервале от 0,05 до 0,95. С учетом симметричности функции нормального распределения, учтенной при составлении программного обеспечения, вывод будет таким: выборки, для которых P-значение менее 0,05, не являются нормальными. Результат расчета всеми применяемыми методами подтверждает заранее известные ответы.

Ресурсы Интернет

В процессе работы над данной главой нами использованы материалы (рабочие заметки, статьи, монографии, учебные методические разработки) сайтов:

- Departement de mathematiques et de statistique, Universite de Montreal, Canada, **www.dms.umontreal.ca**;
- Department of Mathematical Sciences, Montana State University, **www.montana.edu**;
- Department of Physics of Complex Systems, Eotvos University, Budapest, **galahad.elte.hu**;
- Finnish IT center for Science (CSC), Ministry of Education, Finland, **www.csc.fi;**
- Факультет прикладной математики и информатики, Новосибирский государственный технический университет, **www.ami.nstu.ru**.

Рекомендуемая литература

1. Epps T.W., Pully L.B. A test for normality based on the empirical characteristic function // Biometrica, vol. 70, 1983, pp.723–726.
2. Henze N. An Approximation to the limit distribution of the Epps–Pulley test statistic for normality // Metrica, vol. 37, 1990, pp.7–18.
3. Большев Л.Н., Смирнов Н.В. Таблицы математической статистики. — М.: Наука, 1983.
4. Брандт З. Анализ данных. Статистические и вычислительные методы для научных работников и инженеров. — М.: Мир, ООО «Издательство АСТ», 2003.
5. Гайдышев И. Анализ и обработка данных: специальный справочник. — СПб: Питер, 2001.
6. ГОСТ Р ИСО 5479–2002. Статистические методы. Проверка отклонения распределения вероятностей от нормального распределения. — М.: Издательство стандартов. 2002.
7. Кендалл М.Дж., Стьюарт А. Статистические выводы и связи. — М.: Наука, 1973.
8. Кулаичев А.П. Полное собрание сочинений в трех томах. Том 1. Методы и средства анализа данных в среде Windows. STADIA. — М.: Информатика и компьютеры, 1999.

9. Лисенков А.Н. Математические методы планирования многофакторных медико–биологических экспериментов. — М.: Медицина, 1979.
10. Литтл Р.Дж.А., Рубин Д.Б. Статистический анализ данных с пропусками. — М.: Финансы и статистика, 1990.
11. Рекомендации по стандартизации Р 50.1.037–2002. Прикладная статистика. Правила проверки согласия опытного распределения с теоретическим. Часть II. Непараметрические критерии. — М.: Госстандарт России, 2002.
12. Романовский В.И. Математическая статистика. Кн.2. Оперативные методы математической статистики. — Ташкент: Издательство Академии наук УзССР, 1963.
13. Смоляк С.А., Титаренко Б.П. Устойчивые методы оценивания: статистическая обработка неоднородных совокупностей. — М.: Статистика, 1980.
14. Справочник по прикладной статистике. В 2–х т. Т. 1 / Под ред. Э. Ллойда, У. Ледермана, Ю.Н. Тюрина. — М.: Финансы и статистика, 1989.
15. Справочник по прикладной статистике. В 2–х т. Т. 2 / Под ред. Э. Ллойда, У. Ледермана, С.А. Айвазяна, Ю.Н. Тюрина. — М.: Финансы и статистика, 1990.
16. Статистические методы повышения качества / Под ред. Х. Кумэ. — М.: Финансы и статистика, 1990.
17. Тюрин Ю.Н., Макаров А.А. Статистический анализ данных на компьютере. — М.: ИНФРА–М, 1999.
18. Уилкс С. Математическая статистика. — М.: Наука, 1967.
19. Хан Г., Шапиро С. Статистические модели в инженерных задачах. — М.: Мир, 1969.

ГЛАВА 4

Информационный анализ на «чистом» VBA

«Объяснить, как рождается мысль, мы, конечно, не в состоянии. Поэтому сконцентрируемся на том, как использовать в этом процессе информационные технологии».
«Компьютерра» № 21, 2002 г.

Для работы с материалами главы понадобятся лицензионные средства разработки и тестирования программ:

- Microsoft Excel 97 или выше.

Потребуется знание VBA.

Решаемая проблема

Методы теории информации находят широкое применение в научных исследованиях, например, для определения таксономической ценности признаков, при анализе изображений, при выборе адекватной математической модели. Вычисление и интерпретация информационных параметров называется разведочным информационным анализом.

Особенности решения

Применяемые в данной главе научные алгоритмы не являются громоздкими в вычислительном отношении, поэтому в выборе средств было принято решение ограничиться VBA. В прикладном статистическом анализе имеется не слишком

много разделов, требующих больших объемов или высокой скорости вычислений (к наиболее трудным относится вычисление статистических распределений, комбинаторные алгоритмы, методы линейной алгебры и некоторые другие), поэтому Visual Basic хорошо зарекомендовал себя при решении статистических задач. Дополнительным преимуществом является то, что рабочие книги или шаблоны, содержащие код VBA, могут поставляться заказчикам или коллегам, не имеющим на своих компьютерах программного обеспечения анализа данных. Все аналитическое программное обеспечение уже содержится в рабочих книгах или шаблонах.

Научные алгоритмы

Основные VBA-функции, реализующие прикладную логику программы, показаны при описании соответствующих методов.

Информация по Шеннону

В теории информации в качестве меры количества информации, возможности выбора (количества разнообразия) и неопределенности применяется величина

$$H = -\sum_{i=1}^{n} p_i \log_a p_i ,$$

где p_i — вероятность появления дискретного события i, $i = 1,2,,n$; величины p_i, $i=1,2,,n$, образуют множество вероятностей для практических вычислений вероятности допустимо заменить частотами распределений),

a — основание логарифма для оценки величины энтропии, обычно равное 2 или числу e. При вычислении условились считать, что $0 \cdot \log_a 0 = 0$.

Величина H называется *энтропией дискретного множества вероятностей* (энтропией дискретной случайной величины, средней собственной информацией) и иногда обозначается как I, чтобы отличить от энтропии непрерывного распределения. Энтропия представляет собой количественную меру степени неопределенности исхода случайного опыта, зависящую не от индивидуальных свойств результатов опыта, а от соответствующих вероятностей. В дискретном случае энтропия равна нулю, когда одна из вероятностей равна 1, а остальные нулю.

Энтропия непрерывного распределения с функцией плотности распределения $p(x)$ определяется как

$$H = -\int_{-\infty}^{\infty} p(x) \log_a p(x) dx$$

и называется также относительной или дифференциальной энтропией. В такой форме определение энтропии в наших расчетах не применяется.

Если в качестве основания логарифма *a* выбрано число 2, энтропия вычисляется в битах, если число *e* — в нитах. Если энтропия вычислена в нитах, для вычисления энтропии в битах нужно разделить на ln2 значение в нитах.

Исходные данные для вычисления энтропии системы представляют собой дискретный или интервальный вариационный ряд. Дискретный и интервальный вариационные ряды представляют собой таблицы распределения количеств вариант по классам и различаются тем, что в первом случае количества относятся к определенным, возможно нечисловым, значениям признаков, а во втором случае — к интервалам изменения признака (классовым интервалам). Построить вариационный ряд можно вручную или с помощью пакета анализа Microsoft Excel, выбрав из меню **Сервис ▸ Анализ данных ▸ Гистограмма** и следуя указаниям программы.

Некоторыми авторами энтропия называется количеством информации или двоичной энтропией. Так называемая энтропия Колмогорова, характеризующая хаотическое движение в фазовом пространстве (см. *главу 9*) произвольной размерности, также определяется по формуле Шеннона.

Листинг 4.1

```
Private Function Entropie(F() As Double, N As Integer) As Double
'
' Функция вычисления неопределенности (энтропии) системы
' для дискретных процессов по Шеннону.
' Обозначения:
'  F — дискретный вариационный ряд,
'  N — число классов.
' Возвращаемое значение:
'  вычисленное значение энтропии (в битах).
'
  Dim i As Integer ' Счетчик цикла
  Dim H As Double  ' Энтропия в нитах
  H = 0
  For i = 1 To N
    If F(i) <> 0 Then
      H = H + F(i) * Log(F(i))
    End If
  Next i
  Entropie = -H / Log(2)
End Function
```

Максимальное разнообразие системы вычисляется по формуле Хартли

$$H_{\max} = \log_2 n.$$

Таким образом, справедливо

$$0 \le H \le H_{\max}.$$

Максимум энтропии соответствует наибольшей неопределенности или равенству вероятностей всех возможностей. Опыт имеет наибольшую энтропию при k равновероятных исходах с вероятностями $1 / k$. Степень неопределенности опыта тем больше, чем больше число k его исходов.

Листинг 4.2

```
Private Function MaxEntropie(N As Integer) As Double
'
' Функция вычисления максимальной неопределенности (энтропии) системы
' для дискретных процессов по Шеннону.
' Обозначения:
'  N — число разрядов позиций кода.
' Возвращаемое значение:
'  значение энтропии (в битах).
'
  MaxEntropie = Log(N) / Log(2)
End Function
```

Энтропия графического изображения зависит от количества уровней, а при одинаковом числе уровней — от закона распределения. Равномерный закон распределения соответствует полной хаотичности. В этом случае энтропия достигает максимума, который зависит только от количества уровней

$$H_{\max} = \log_a (h_{\max} - h_{\min} + 1),$$

где $h_{\max}$ — максимальный уровень изображения,

$h_{\min}$ — минимальный уровень изображения,

значение в скобках представляет собой размах дискретной случайной величины.

Для сравнения систем, различающихся по количеству элементов кода, сопоставление энтропий не будет корректным. Для решения задачи применяется относительная энтропия (коэффициент сжатия информации), определяемая как

$$h = H / H_{\max}.$$

Относительная энтропия определяет относительную степень информационной загруженности системы по отношению к возможной максимальной нагрузке. Кроме того, относительная энтропия, как и рассмотренная ниже избыточность, может характеризовать степень близости закона распределения к равномерному распределению.

Листинг 4.3

```
Private Function RelEntropie(Hmax As Double, H As Double) As Double
'
' Функция вычисления относительной энтропии системы.
' Обозначения:
'  Hmax — максимальная энтропия,
'     H — текущая энтропия.
' Возвращаемое значение:
'  значение организации.
'
  RelEntropie = H / Hmax
End Function
```

Под организацией системы понимают реализованную в ней неопределенность. Абсолютная организация системы вычисляется по формуле

$$O = H_{\max} - H.$$

При организации, равной максимальной энтропии, система становится детерминированной (см. табл. 4.1).

Листинг 4.4

```
Private Function Organization(Hmax As Double, H As Double) As Double
'
' Функция вычисления абсолютной организации системы.
' Обозначения:
'  Hmax - максимальная энтропия,
'     H - текущая энтропия.
' Возвращаемое значение:
'  значение организации (в битах).
```

```
'
  Organization = Hmax - H
End Function
```

Разведочный информационный анализ

В разведочном информационном анализе применяется понятие избыточности, передаваемой информации. Она дает соотношение между полным количеством информации, шумом и сохранившейся упорядоченностью системы. Избыточность (в источниках обозначается также как R) вычисляется по формуле

$$D = 1 - H / H_{\max}.$$

Избыточность характеризует степень близости закона распределения к равномерному распределению и может быть выражена в долях, как в показанной формуле, либо в процентах. Например, для полутонового изображения с 256 уровнями энтропия не может превышать 8, а избыточность при равномерном законе распределения равна нулю.

Листинг 4.5

```
Private Function Surplus(Hmax As Double, H As Double) As Double
'
' Функция вычисления избыточности системы.
' Обозначения:
'  Hmax — максимальная энтропия,
'     H — текущая энтропия.
' Возвращаемое значение:
'  значение избыточности (в процентах).
'
  Surplus = (1 — H / Hmax) * 100
End Function
```

При математическом моделировании (см. *главу 9*) важно точно обосновать возможность применения тех или иных методов анализа и адекватность модели. В первом приближении проверку адекватности можно выполнить путем вычисления информационных показателей и выбрать тип модели, в зависимости от информационных показателей, представленных в таблице 4.1.

***Таблица 4.1**. Выбор адекватной модели*

Величина избыточности, %	Характеристика системы	Тип модели	Пример из биологии
0–10	Вероятностная	Вероятностная (статистическая)	Взаимодействие организма с окружающей средой
10–30	Вероятностно-детерминированная (квази-детерминированная)	Дифференциальные уравнения с коэффициентами, подчиняющимися непрерывным законам распределения	Системы управления внутренними органами с компонентами нервной регуляции
30–100	Детерминированная	Дифференциальные или интегральные уравнения динамики входных, промежуточных и выходных показателей	Модель биологического нейрона, элементы сердечно-сосудистой системы

Разность избыточности в норме и при патологии приводит к понятию ненадежности (эквивокации) передачи информации, что дает количественную характеристику структурной перестройки исследуемого объекта (системы). Вычисление эквивокации производится по формуле

$$D = R_{norm} - R_{pat} = \frac{H_{pat} - H_{norm}}{H_{\max}},$$

где R_{norm} — избыточность в норме,

R_{pat} — избыточность в патологии,

H_{pat} — энтропия в патологии,

H_{norm} — энтропия в норме.

Руководство пользователя и сообщения об ошибках

Для работы с модулем информационного анализа (IA) выберите из меню программы Microsoft Excel добавленный программным обеспечением AtteStat новый пункт **AtteStat ▸ Модуль IA ▸ Информационный анализ**. На экране появится диалоговое окно, изображенное на рис. 4.1.

Рис. 4.1. Окно модуля информационного анализа

Затем выберите или введите интервал переменной. Исходным для расчета является вариационный ряд, то есть таблица распределения количеств вариант по классам. В некоторых опытах вариационный ряд может быть получен экспериментально. В других опытах вариационный ряд строится на основании первичных экспериментальных данных. Тогда он может быть построен вручную или средствами пакета анализа Microsoft Excel. В последнем случае выберите из меню **Сервис ▸ Анализ данных ▸ Гистограмма**.

Выберите или введите выходной интервал. Начиная с первой ячейки выходного интервала (остальные ячейки интервала игнорируются), будут выведены вычисленные параметры информационного анализа.

Отметьте необходимые информационные параметры расчета при помощи кнопок **Отметить все** или **Очистить все**. Нажмите кнопку **Выполнить**.

При ошибках, вызванных неверными действиями пользователя, выдаются сообщения об ошибках. При ошибках ввода могут выдаваться диагностические сообщения следующих типов.

Рис. 4.2. Окно сообщения об ошибке

Проверьте исходные данные и заполните все ячейки входного интервала. Для избежания ошибок расчета, вызванных разногласиями в интерпретации значений пустых ячеек, пакет информационного анализа требует заполнения всех ячеек. Пустые ячейки интервала заполните нулями.

Рис. 4.3. Окно сообщения об ошибке

Проверьте типы ячеек входного интервала. Тип может быть только числовым. Проще всего выделить интервал ячеек и явно задать их тип как числовой.

Рис. 4.4. Окно сообщения об ошибке

Вы не выбрали или неверно ввели входной интервал. Лучшим способом избежать ошибки является не ввод с клавиатуры, а выделение интервала с помощью мыши, методом протаскивания курсора.

Рис. 4.5. Окно сообщения об ошибке

Вы не выбрали или неверно ввели входной интервал. Лучшим способом избежать ошибки является не ввод с клавиатуры, а выделение интервала с помощью мыши, методом протаскивания курсора.

Рис. 4.6. Окно сообщения об ошибке

Для расчета необходимо выбрать интервал, содержащий хотя бы две ячейки с числовым значением.

Рекомендуемая литература

1. MacKay D.J.C. Information Theory, Inference, and Learning Algorithms [online]. — Cambridge University Press, 1999. Available from World Wide Web: < **http:// www.cs.toronto.edu/~mackay/itprnn** >.
2. Вентцель Е.С. Теория вероятностей. — М.: Высшая школа, 1999.
3. Вероятность и математическая статистика: Энциклопедия. — М.: Научное издательство «Большая Российская энциклопедия», 1999.
4. Кадыров Х.К., Антомонов Ю.Г. Синтез математических моделей биологических и медицинских систем. — Киев: Наукова думка, 1974.
5. Куликовский Л.Ф., Мотов В.В. Теоретические основы информационных процессов. — М.: Высшая школа, 1987.
6. Кульбак С. Теория информации и статистика. — М.: Наука, 1967.
7. Леонтюк А.С., Леонтюк Л.А., Сыкало А.И. Информационный анализ в морфологических исследованиях. — Минск: Наука и техника, 1981.
8. Лоскутов А.Ю., Михайлов А.С. Введение в синергетику: Учебное руководство. — М.: Наука, 1990.
9. Макарова Н.В., Трофимец В.Я. Статистика в Excel: Учебное пособие. — М.: Финансы и статистика, 2002.
10. Теннант-Смит Дж. Бейсик для статистиков. — М.: Мир, 1988.
11. Шеннон К. Работы по теории информации и кибернетике. — М.: Издательство иностранной литературы, 1963.
12. Шустер Г. Детерминированный хаос: Введение. — М.: Мир, 1988.
13. Яглом А.М., Яглом И.М. Вероятность и информация. — М.: Наука, 1973.
14. Яншин В.В., Калинин Г.А. Обработка изображений на языке Си для IBM PC: Алгоритмы и программы. — М.: Мир, 1994.

ГЛАВА 5

Язык макросов Microsoft Excel и VBA: статистическая проверка гипотез

«Проверка — это столкновение, по крайней мере, трех сторон: соперничающих теорий и эксперимента».
И. Лакатос

Для работы с материалами главы понадобятся лицензионные средства разработки и тестирования программ:

- Microsoft Excel 97 или выше;
- Microsoft Visual C++ 5.0 или выше;
- Borland C++ 5.5 или выше;
- Bloodshed Dev-C++ 4.01 или выше.

Потребуется знание VBA, XLM, C.

Решаемая проблема

Закономерности в экспериментальных данных можно обнаружить путем статистической проверки результатов многих опытов, принимая или отклоняя определенные гипотезы. Процедура статистической проверки начинается с формулировки *нулевой* гипотезы H_0 (например, «нет статистически значимого различия») и *альтернативной* (*конкурирующей*) гипотезы H_1 («имеет место статистически значимое различие»). Затем, на основании выборок

экспериментальных данных, производится проверка нулевой гипотезы относительно альтернативной с помощью соответствующего теста (критерия). Результатом статистической проверки является вывод о том, в скольких случаях, например, на каждые 100 проведенных испытаний, в предположении определенной модели, отклонения от модели можно считать случайными. Таким образом, на заданном стандартном уровне значимости исследователь может остановиться на одной из гипотез.

Критерии (тесты), при помощи которых могут быть сравнены статистические совокупности, разделяются на две группы: параметрические и непараметрические. Параметрические критерии достаточно полно представлены в электронных таблицах Microsoft Excel. Особенностью параметрических критериев является предположение о том, что распределение признака в генеральной совокупности подчиняется некоторому известному, в данном случае нормальному, закону. Нормальность эмпирического распределения выборки должна быть статистически доказана до применения любого параметрического теста, но существует большое количество опытных данных, которые не показывают нормальности своего распределения, поэтому применение параметрических критериев не может быть обоснованным для таких данных.

Непараметрические критерии не требуют предварительных предположений относительно вида исходного распределения и являются более робастными[1], чем их параметрические аналоги. Их называют также критериями значимости, независимыми от типа распределения.

Особенности решения

Регистрация функций DLL может производиться на листе макросов. Для удобства, макрос, регистрирующий функции DLL, выполнен стандартной функцией `Run` при открытии рабочей книги, аналогично первому проекту из *главы 3*.

Комбинированное решение обеспечило компактность и функциональность приложения. Функции можно:

1. вызывать как стандартные Microsoft Excel;
2. использовать как в стандартных надстройках;
3. вызвать из меню **AtteStat**.

Для реализации пунктов 1 и 2 достаточно VBA-функции `Auto_Open`. Остальной код VBA реализует пункт 3.

---

[1] *Слабая чувствительность к отклонениям от стандартных условий называется робастностью, а методы, применимые в широком диапазоне реальных условий, называют робастными.*

Функции, представленные в данной главе, выполнены на алгоритмическом языке C++. В *приложении 1* приводится альтернативное решение на языке C. Конверсия между вариантами функций может быть выполнена достаточно просто.

Потребительские качества программы[2] оценены на конкурсе Microsoft Office Extensions 2001–2002 гг. Десятки положительных отзывов пользователей доказали полезность и своевременность решения.

Научные алгоритмы

Рассмотрим несколько типов непараметрических тестов в рассмотренном модуле, но сначала остановимся на общей методике их применения.

Утверждение о том, что некоторая статистическая величина θ лежит в доверительном интервале (t_0, t_1) означает, что равенство

$$P(t_0 \le \theta \le t_1) = 1 - \alpha,$$

где α — уровень значимости (significance level),

справедливо в $(1-\alpha)$ случаях.

Для проверки гипотезы выборочное пространство W разделяется на две области:

- w — критическая; при попадании выборочной точки в область гипотеза отвергается;
- $(W - w)$ — область принятия; при попадании выборочной точки в область гипотеза принимается.

Если известно распределение вероятностей наблюдений, соответствующее H_0, можно определить w так, чтобы при выполнении H_0 вероятность отвергнуть эту гипотезу была равна заранее заданной величине — уровню значимости:

$$P\{x \in w \mid H_o\} = p.$$

Для уровня значимости может вычисляться точное значение. Такой подход применяется во многих коммерческих программных продуктах и называется вычислением P-значения. Оно получается в результате подстановки значения статистики критерия в ее функцию распределения, поэтому иногда называется обратной вероятностью. Вывод при этом может быть таким: сред-

---

[2] *MegaStat for Excel версия 1.0 beta. Программа доступна в приложении на компакт-диске к журналу «PC Magazine / Russian Edition», №11 за 2002 г.*

ние значения двух выборок статистически различаются на уровне значимости 0,012. В публикациях принято приводить ближайшее к вычисленному стандартное значение уровня значимости. Существует несколько стандартных оценок:

1. $p > 0,05$ — различия недостоверны;
2. $p < 0,05$ — различия достоверны на уровне значимости 0,05;
3. $p < 0,01$ — различия достоверны на уровне значимости 0,01;
4. $p < 0,005$ — различия достоверны на уровне значимости 0,005;
5. $p < 0,001$ — различия достоверны на уровне значимости 0,001, причем данную оценку можно применять и при $p = 0$.

В случае 1 может быть принята нулевая гипотеза об отсутствии различия, в случаях 2, 3, 4 и 5 может быть принята альтернативная гипотеза о различии на указанном уровне значимости.

Рассмотрим понятия односторонней (one sided) и двусторонней (two sided) гипотез, которым соответствуют односторонний и двусторонний критерии значимости. Когда исследователь имеет достаточное количество данных, позволяющих предсказать в альтернативной гипотезе направление различий (доля желательных эффектов в опытной группе превышает долю эффектов в контрольной группе), используется односторонний критерий, в противном случае (опытная группа просто отличается от контрольной группы) используется двусторонний критерий. Уровень значимости двустороннего критерия в два раза выше одностороннего.

Данные, полученные в реальных экспериментах, могут быть представлены независимыми либо попарно сопряженными выборками.

Критерии для независимых выборок применяются, чтобы выявить статистическую значимость различий двух различных групп. Примерами независимых выборок могут служить:

- параметры двух групп испытуемых, к которым применялись различные методики лечения, с целью изучения значимости различий между методиками;
- параметры двух групп, к одной из которых (опытная группа) применялось воздействие методики, а к другой (контрольной) не применялось, с целью изучения значимости влияния данной методики на результат лечения.

Критерии, применяемые к выборкам с попарно сопряженными вариантами, называются парными критериями, либо критериями для связанных или сопряженных выборок. Примеры сопряженных выборок:

- параметры одной и той же испытуемой группы до и после воздействия какого-либо фактора, например методики лечения;

- параметры одного объекта испытаний, относящиеся к разным его частям, одна из которых подвергается воздействию, а другая – нет.

Критерии рандомизации

Ниже рассмотрены несколько критериев, объединенных общей идеей рандомизации компонент выборок или их рангов. Данные критерии называют также критериями перестановок, а при их вычислении применяются комбинаторные алгоритмы. Комбинаторные алгоритмы могут быть очень трудоемкими в вычислениях, особенно при больших численностях выборок.

Критерий рандомизации для независимых выборок

Критерий рандомизации компонент Фишера применяется для проверки нулевой гипотезы о том, отобраны ли две исследуемые выборки из генеральных совокупностей с одинаковыми средними значениями.

Методика теста базируется на идее перебора всех комбинаций наблюдаемых отметок. Пусть даны две выборки: $x_i, i = 1,2,...,n_x$, и $y_j, j = 1,2,...,n_y$, где n_x и n_y — численности выборок. Сумма вариант выборки, меньшая из наблюдаемых:

$$S = \min\left(\sum_{i=1}^{n_x} x_i, \sum_{i=1}^{n_y} y_i\right).$$

Число благоприятных исходов вычисляется по формуле

$$N = 2\sum_{i=1}^{C_n^m} n_i, n_i = \left[\begin{array}{l} 0, s_i < S \\ 1, s_i \geq S \end{array}\right.,$$

где n_i –оценка i–го исхода,

C_n^m — общее число исходов,

$n = n_x + n_y$ — численность объединенной выборки,

m — численность выборки, соответствующей минимальной сумме

$$s_i = \sum_{j=1}^{m} z_j, i = 1,...,C_n^m,$$

где z_j, j = 1,...,m — массив сочетаний из объединенной выборки.

Одностороннее P-значение вычисляется по формуле

$$p = N / C_n^m$$

и сравнивается с заданным уровнем значимости. Например, проверяется нулевая гипотеза о равенстве средних двух выборок на уровне значимости 0,05. Предположим, вычислено $p = 0{,}019048 < 0{,}05$ или $p = 0{,}988095 > 0{,}95$, что вытекает из требования при оценке различий признавать значимыми как малые, так и большие величины статистики критерия. Нулевая гипотеза отклоняется, и принимается альтернативная гипотеза о статистически значимом различии средних значений на уровне значимости менее 0,05.

Хотя критерий разработан для анализа выборок малой численности, в литературе даны минимальные значения выборок, позволяющие отклонить нулевую гипотезу (см. табл. 2.1). При больших численностях выборок вместо описанного здесь рекомендуется применять W-критерий ранговой рандомизации Вилкоксона.

***Таблица 2.1**. Минимально допустимые численности выборок для обоснованного применения критериев рандомизации компонент*

	Односторонний критерий		**Двухсторонний критерий**	
Уровень значимости	0,01	0,05	0,01	0,05
Минимальная численность выборки	5	6	7	8

Представленная функция очень компактна вследствие применения рекурсивного алгоритма при вычислении благоприятных сочетаний.

Листинг 5.1

```
double __declspec(dllexport) RandomIndependent (FP *x,FP *y)
//
// Функция вычисляет критерий рандомизации для несвязанных
// выборок для односторонней гипотезы.
// Обозначения:
// x - первая выборка,
// y - вторая выборка.
// Возвращаемое значение:
// p-значение для односторонней гипотезы.
//
{
 short j,                          // Счетчик цикла
       nx = x->rows * x->columns, // Численность первой выборки
       ny = y->rows * y->columns; // Численность первой выборки
```

```
short n = nx + ny,                // Численность суммарного массива
      k = 0,                      // Счетчик
      m;                          // Численность массива сочетаний

double nSUCCESS = 1,              // Число благоприятных исходов
       *dat,                      // Суммарный массив
       sum,                       // Сумма вариант
       sumx = 0,                  // Сумма вариант массива x
       sumy = 0;                  // Сумма вариант массива y

dat = new double[n];

//////////////////////////////
// Выбор массива с меньшей суммой
for (j = 0; j < nx; j++)
  sumx += x->Array[j];
for (j = 0; j < ny; j++)
  sumy += y->Array[j];
if (sumx >= sumy)
  {
   m = ny;
   sum = sumy;
  }
else
  {
   m = nx;
   sum = sumx;
  }

//////////////////////////////
// Построение суммарного массива
for (j = 0; j < nx; j++)
  dat[k++] = x->Array[j];
for (j = 0; j < ny; j++)
  dat[k++] = y->Array[j];

//////////////////////////////
// Вычисление значения критерия
```

```
 Combination (dat,n,m,sum,&nSUCCESS);
 delete [] dat;

 return 0.5 * nSUCCESS /
        exp(LogFactorial (n) - LogFactorial (nx) - LogFactorial (ny));
}
```

Критерий рандомизации для связанных выборок

Критерий рандомизации компонент для связанных выборок применяется для проверки нулевой гипотезы о равенстве средних значений двух связанных совокупностей, в качестве которых взяты параметры некоторой группы испытуемых объектов до и после воздействия какого-либо фактора.

Основным моментом в реализации критерия является перебор возможных исходов, построенных из разностных отметок. Пусть даны две выборки: $x_i, y_i, i = 1,2,...,n,$ где n — число пар экспериментальных значений. Тогда сумма массива разностных отметок будет

$$S = \sum_{i=1}^{2^n} s_i \ .$$

Определим значения разностных отметок:

$$s_i = \sum_{j=1}^{n} a_{ij} \left| x_j - y_j \right|, i = 1,...,2^n \ ,$$

где $a_{ij}, i = 1,...,2n; j = 1,2,...,n$ — элементы матрицы возможных исходов.

Систематизацию перебора всех возможных исходов мы проведем в соответствии с ортогональным планом эксперимента первого порядка. Размер полного ортогонального плана составляет $2n$ строк на n столбцов, причем j-й столбец размером $2n$ представляет собой величины +1 и −1, чередующиеся с шагом $2j^{-1}, j = 1,2,...,n.$.

Число благоприятных исходов вычисляется по формуле:

$$N = \sum_{i=1}^{2^n} n_i, n_i = \left[\begin{array}{l} 0, s_i < S \\ 1, s_i \geq S \end{array} \right. .$$

Одностороннее P-значение, вычисленное по формуле

$$p = \frac{N}{2n},$$

сравнивается с заданным уровнем значимости. Например, проверяется нулевая гипотеза о равенстве средних двух выборок на уровне значимости 0,05. Пусть вычислено P-значение, равное 0,168. Нулевая гипотеза об отсутствии статистически значимого различия средних значений принимается на уровне значимости > 0,05.

В показанной реализации ортогональный план не хранится в памяти компьютера во время вычислений — каждая очередная строка насчитывается при необходимости При больших численностях выборок вместо описанного здесь рекомендуется применять T-критерий ранговой рандомизации Вилкоксона.

Листинг 5.2

```
double __declspec(dllexport) Random (FP *x,FP *y)
//
// Функция вычисляет критерий рандомизации для связанных выборок.
// Обозначения:
//  x - первая выборка,
//  y - вторая выборка.
// Возвращаемое значение:
//  p-значение для односторонней гипотезы.
//
{
 short i,                        // Счетчик цикла
       *step,                    // Вектор шагов чередования
       *a,                       // Строка матрицы планирования
       n = x->rows * x->columns, // Численность первой выборки
       m = y->rows * y->columns; // Численность второй выборки

 long j,                         // Счетчик цикла
      n2 = 2 << (n - 1);         // Число возможных исходов

 double n1 = 0,                  // Число благоприятных исходов
        *dat,                    // Массив модулей разностей
        sum = 0,                 // Пороговое значение
        sum1;                    // Текущая сумма

 ////////////////////////////////////////////////////////
```

```
 // Корректировка для избежания выхода за границу массива
 n = MinimumOf (n,m);

 dat = new double[n];
 step = new short[n];
 a = new short[n];

 for (i = 0; i < n; i++)
   {
    dat[i] = x->Array[i] - y->Array[i];
    sum += dat[i];
    dat[i] = fabs (dat[i]);
    step[i] = TwoDegree (i);
    a[i] = -1;
   }

 ////////////////////////////////
 // Перечисление возможных исходов
 for (j = 0; j < n2; j++)
   {
    for (i = 0, sum1 = 0; i < n; i++)
      {
       if (j / step[i] * step[i] == j)
         a[i] *= -1;
       sum1 += dat[i] * a[i];
      }
    ////////////////////
    // Благоприятный исход
    if (sum1 >= sum)
      n1++;
   }

 delete [] dat; delete [] step; delete [] a;

 return n1 / n2;
}
```

W-критерий Вилкоксона

W-критерий Вилкоксона (критерий ранговых сумм Вилкоксона, двухвыборочный критерий Вилкоксона, статистика ранговой суммы Уилкоксона, Wilcoxon signed-rank test, Wilcoxon sum-of-ranks test for comparing two unmatched samples) применяется для сравнения двух независимых совокупностей одинаковой или разной численности по их центральной тенденции. Иначе говоря, по критерию Вилкоксона сравнивают центры двух эмпирических распределений, или эмпирические функции распределений. Этот критерий является ранговым, так как он оперирует не численными значениями вариант, а их рангами, то есть местами в рядах, упорядоченных по возрастанию или убыванию вариант. Он является критерием ранговой рандомизации — процедура точного вычисления критерия Вилкоксона практически не отличается от вычисления критерия рандомизации компонент, за исключением того, что манипуляции производятся не с самими вариантами выборок, а с их рангами.

Ранжирование сравниваемых выборок производят совместно. Данная процедура может быть организована различными способами, предпочтительным является объединение двух выборок, их сортировка, ранжирование и последующее разнесение рангов на места соответствующих им вариант в обеих выборках. Если имеются совпадающие значения, совпавшим наблюдениям условились назначать средний ранг. Затем вычисление W-статистики критерия производится по формуле:

$$W = \min\left(\sum_{i=1}^{n_1} R_i, \sum_{i=1}^{n_2} S_i\right),$$

где $R_i, i = 1,2,...,n_1,$ — ранги выборки, имеющей наименьшую сумму рангов,

$S_i, i = 1,2,...,n_2,$ — ранги выборки, имеющей наибольшую сумму рангов.

Если меньшая сумма рангов меньше верхнего критического значения, различие между рядами статистически достоверно на заданном стандартном доверительном уровне. Для практического вычисления критерия нами применяется другой подход, основанный на том факте, что статистика

$$\frac{W - E[W]}{\sqrt{D[W]}},$$

где $E[W] = n_1(N+1)/2$ — математическое ожидание,

$D[W] = n_1 n_2 (N+1)/12$ — дисперсия, которая при наличии совпадающих вариант корректируется,

$N = n_1 + n_2$ — суммарная численность выборок,

распределена по стандартному нормальному закону.

Если полученное значение статистики превышает 0,02, в формулу вводится поправка на непрерывность: считается, что новое значение наименьшей суммы рангов равно $W + 0{,}5$.

Приведем пример. Пусть вычислено P–значение, равное 0,033097. На уровне значимости < 0,05 принимается конкурирующая гипотеза о различии выборок.

Листинг 5.3

```
double __declspec(dllexport) WilcoxonTest (FP *x,FP *y)
//
// Функция вычисления W-критерия Вилкоксона.
// Обозначения:
//  x - первая выборка,
//  y - вторая выборка.
// Возвращаемое значение:
//  p-значение для односторонней гипотезы.
//
{
 short i,                          // Счетчик
       nx = x->rows * x->columns,  // Численность первой выборки
       ny = y->rows * y->columns,  // Численность первой выборки
       *ix;                        // Информация о перестановках

 short n = nx + ny,                // Общая численность
       save;                       // Буфер

 double *xx,*xy,                   // Рабочий массив
        xn = 0,yn = 0,             // Суммы рангов
        tt,                        // Поправка на объединение рангов
        criterion,                 // Статистика критерия
        t;                         // Параметр нормальной аппроксимации

 ix = new short[n];
 xy = new double[n];
```

```
xx = new double[n];
//////////////////////////
// Слияние исходных массивов
for (i = 0; i < nx; i++)
  xy[i] = x->Array[i];
for (i = 0; i < ny; i++)
  xy[i + nx] = y->Array[i];

//////////////////////////
// Сортировка по возрастанию
SortArrayUp (xy,n,ix,1);

///////////////
// Ранжирование
tt = Rank (xy,xx,n);

//////////////////////////////
// Восстановление порядка рангов
UnSortArray (xx,xy,n,ix);

////////////////////////
// Вычисление сумм рангов
for (i = 0; i < nx; i++)
  xn += xy[i];
for (i = 0; i < ny; i++)
  yn += xy[i + nx];

//////////////////////////////////////////////
// Параметры выборки с наименьшей суммой рангов
if (xn <= yn)
  criterion = xn;
else
  {
   criterion = yn;
   save = ny;
   ny = nx;
   nx = save;
```

```
    }
  ////////////////////////////////////
  // Параметр нормальной аппроксимации
  t = fabs ((criterion) - 0.5 * nx * (n + 1)) /
      sqrt (0.0833333333 * nx * ny * (n + 1 - tt / n / (n - 1)));

  ////////////////////////////////////////
  // Расчет с поправкой на непрерывность
  if ((1.0 - NormalDistribution (t)) > 0.02)
    t = fabs ((criterion + 0.5) - 0.5 * nx * (n + 1)) /
        sqrt (0.0833333333 * nx * ny * (n + 1 - tt / n / (n - 1)));

  delete [] ix; delete [] xy; delete [] xx;

  return (1.0 - NormalDistribution (t));
}
```

T-критерий Вилкоксона для связанных выборок

T-критерий Вилкоксона (одновыборочный критерий Вилкоксона, знаковый ранговый критерий Уилкоксона, критерий знаковых рангов Уилкоксона, критерий Вилкоксона для сопряженных пар, T-дельта-критерий, W-критерий Вилкоксона либо просто критерий Вилкоксона, Wilcoxon signed-ranks test for matched pairs) применяется для сравнения выборок с попарно сопряженными вариантами. Проверяется статистическая значимость нулевой гипотезы о том, что распределение случайных величин[3] симметрично относительно нуля. Например, критерием сравниваются выборки, представляющие собой параметры одной и той же совокупности до и после воздействия некоторого фактора, с целью выявления эффекта воздействия данного фактора.

Методика точного вычисления критерия полностью совпадает с критерием рандомизации компонент для связанных выборок, с тем исключением, что манипуляции производятся не с вариантами выборок, а с их рангами. Методика приближенных вычислений похожа на процедуру вычисления W-критерия Вилкоксона, однако здесь мы оперируем абсолютными величинами разностей вариант. Массив разностей ранжируется. Затем рангам добавляются знаки разностей, и вычисляется наименьшая из сумм положительных W^+ рангов, которая сравнивается с критическим значением.

---

[3] *Эти случайные величины в рассматриваемом случае представляют собой разности случайных величин, соответствующих двум другим выборкам.*

Для практического вычисления критерия мы применяем подход, учитывающий факт, что статистика

$$\frac{W^+ - E[W^+]}{\sqrt{D[W^+]}},$$

где $E[W^+] = N(N+1)/4$ — математическое ожидание,

$D[W^+] = N(N+1)(2N+1)/24$ — дисперсия, которая в случае совпадений рангов корректируется, как показано в следующем разделе,

N — численность каждой выборки,

распределена по стандартному нормальному закону.

Пример. Предположим, вычислено P-значение, равное 0,086476. Следовательно, различия можно признать значимыми на уровне < 0,1.

Листинг 5.4

```
double __declspec(dllexport) WilcoxonSignedTest (FP *x,FP *y)
//
// Функция вычисления T-критерия Вилкоксона для связанных выборок.
// Обозначения:
//  x - первая выборка,
//  y - вторая выборка.
// Возвращаемое значение:
//  p-значение для односторонней гипотезы.
//
{
 short i,j,                      // Счетчик
       *ix,                      // Информация о сортировке
       n = x->rows * x->columns, // Численность первой выборки
       m = y->rows * y->columns; // Численность второй выборки

 double *spx,*spy,               // Копии исходных массивов
        *xx,                     // Массив рангов
        save,                    // Буфер, затем поправка
                                 // на объединение рангов
        criterion = 0,           // Статистика критерия
        t;                       // Параметр нормальной аппроксимации
```

```
//////////////////////////////////////////////////////////
// Корректировка для избежания выхода за границу массива
n = MinimumOf (n,m);

spx = new double[n];
spy = new double[n];
xx = new double[n];
ix = new short[n];

//////////////////////////////////////
// Работаем с копиями исходных массивов
for (i = 0; i < n; i++)
  {
   spx[i] = x->Array[i];
   spy[i] = y->Array[i];
  }

////////////////////
// Массивы разностей
for (i = 0; i < n; i++)
  {
   save = spx [i];
   spx[i] = fabs (spx[i] - spy[i]); // Разности без знаков
   spy[i] = save - spy[i];          // Разности со знаками
  }

////////////////////////////
// Сортировка по возрастанию
SortArrayUp (spx,n,ix,1);

///////////////
// Ранжирование
save = Rank (spx,xx,n);

//////////////////////////////////////////////////////
// Восстановление порядка рангов массива со знаками
UnSortArray (xx,spx,n,ix);
```

```
 ////////////////////////////////////////
 // Сумма рангов положительных разностей
 for (i = 0; i < n; i++)
   if (spy[i] >= 0)
     criterion += spx[i];

 delete [] spx; delete [] spy; delete [] ix; delete [] xx;

 ////////////////////////////////////////
 // Расчет с поправкой на непрерывность
 return (1.0 - NormalDistribution (fabs (criterion - 0.25 * n * (n + 1))
/
     sqrt (0.0416666667 * n * (n + 1) * (2 * n + 1) - 0.0208333333 *
save)));
}
```

Проблема совпадений в критериях Вилкоксона

Рассмотрим две проблемы, возникающие при использовании критериев типа критерия Вилкоксона.

Проблема равных рангов не имеет места при точном вычислении статистик критериев, а возникает только при вычислении аппроксимации критериев. При наличии совпадений в паре анализируемых выборок совпадающим наблюдениям приписывается средний ранг. При этом формулы для вычислений нормальных аппроксимаций статистик критериев остаются совершенно теми же, а формулы для дисперсий, применяемых для их вычисления, корректируются.

Для W-критерия Вилкоксона соотношения для дисперсии запишутся как

$$D[W] = \frac{n_1 n_2}{12}\left[N + 1 - \frac{T}{N(N-1)}\right],$$

где T — поправка на объединение рангов, вычисляемая в процессе ранжирования,

$N = n_1 + n_2$ — суммарная численность выборок.

Для T-критерия Вилкоксона новая дисперсия будет считаться по формуле

$$D[W^+] = \frac{N(N+1)(2N+1)}{24} - \frac{T}{48},$$

где N — численность каждой выборки.

В составленных программах совпадения учитываются. Легко заметить, что модифицированные соотношения при числе связок, равном нулю, обращаются в обычные соотношения. Следовательно, достаточно использовать только формулы для модифицированных соотношений.

Другая проблема, возникающая только в алгоритме вычисления T-критерия Вилкоксона, — наличие совпадений одновременно в двух анализируемых выборках, что приводит к нулевым разностям. Применяется прием вычеркивания равных сопряженных вариант из сравниваемых выборок. При этом уменьшаются объемы выборок на количество выброшенных вариант.

Критерий Колмогорова–Смирнова

По критерию Колмогорова–Смирнова сравнивают функции распределения двух эмпирических рядов. Проверяется нулевая гипотеза о том, являются ли одинаковыми непрерывные функции распределения генеральных совокупностей, из которых взяты выборки. Иначе, проверяется принадлежность двух выборок одной и той же генеральной совокупности, при условии непрерывности ее функции распределения.

Статистика критерия имеет вид

$$D_{m,n} = \sup_{-\infty<x<\infty} \left| F_n(x') - G_m(x) \right|,$$

где $D_{m,n}$ — максимальная разность между частостями рядов x' и x,

m и n — численности вариационных рядов.

Возможен непосредственный расчет по данной формуле, однако при этом нужно учесть, что функции распределения должны быть сопоставимы по оси абсцисс, что гарантируется, если они явились продуктом построения не по исходной выборке, а по частотам распределения, с сопоставимыми классовыми интервалами. Данный способ удобен только при анализе больших выборок.

В случае построения массивов функций распределения непосредственно по выборкам (без разнесения вариант по классовым интервалам) для двустороннего критерия вычисления производятся по формулам:

$$D_{m,n} = \max\left(D^{+}_{m,n}, D^{-}_{m,n}\right),$$

$$D^{+}_{m,n} = \max_{1\le r\le m}\left(\frac{r}{m} - F_n(x'_r)\right) = \max_{1\le s\le n}\left(G_m(x_s) - \frac{s-1}{n}\right),$$

$$D^{-}_{m,n} = \max_{1\le r\le m}\left(F_n(x'_r) - \frac{r-1}{m}\right) = \max_{1\le s\le n}\left(\frac{s}{n} - G_m(x_s)\right).$$

Для одностороннего критерия

$$D_{m,n} = D^{+}_{m,n}.$$

При данном способе построения массивов функций распределения, они будут равномерны по оси ординат, чем и определяется вид формул.

Функция распределения статистики критерия $D_{m,n}\sqrt{\frac{mn}{m+n}}$ при $mn/(m+n) \to \infty$ сходится к функции распределения Колмогорова.

При данном способе построения функций распределений мы используем их равномерность по оси ординат, так что значение в текущей точке легко вычисляется через номер варианты в исходной выборке.

Листинг 5.5

```
double __declspec(dllexport) SmirnovExact (FP *x,FP *y)
//
// Функция точного вычисления критерия Колмогорова-Смирнова.
// Обозначения:
//  x - первая выборка,
//  y - вторая выборка.
// Возвращаемое значение:
//  p-значение для односторонней гипотезы.
//
{
 short i,j = 0,                    // Счетчики
       nx = x->rows * x->columns, // Численность первой выборки
       ny = y->rows * y->columns, // Численность первой выборки
       *ix;                       // Фиктивный указатель

 double s,                         // Рабочая переменная
        criterion = 0,             // Статистика критерия
        *xx,*yy;                   // Копии выборок

 xx = new double[nx];
 yy = new double[ny];

 ///////////////
 // Копии выборок
```

```
for (i = 0; i < nx; i++)
  xx[i] = x->Array[i];
for (i = 0; i < ny; i++)
  yy[i] = y->Array[i];

////////////////////
// Сортировка выборок
SortArrayUp (xx,nx,ix,0);
SortArrayUp (yy,ny,ix,0);

//////////////////////////////////////////////////
// Перебрать все варианты 2-й выборки относительно
// всех вариант 1-й выборки
for (i = 0; i < ny; i++)

  ////////////////////////////////////////////////////////////
  // Текущая варианта 2-й выборки меньше или равна 1-й варианте
  // 1-й выборки
  if (yy[i] <= xx[0])
    {
     s = (double)i / ny;
     if (s > criterion)
       criterion = s;
    }

  ////////////////////////////////////////////////////////
  // Текущая варианта 2-й выборки больше или равна последней
  // варианте 1-й выборки
  else if (yy[i] >= xx[nx - 1])
    {
     s = 1 - (double) i / ny;
     if (s > criterion)
       criterion = s;

     ////////////////////////////////////////////////
     // Дальше вычисления можно не проводить - максимум
     // уже был достигнут в процессе вычислений
     break;
```

```
    }

  ////////////////////////////////////////////////////
  // Текущая варианта 2-й выборки в пределах 1-й выборки
  else
    {
     ////////////////////////////////////////
     // Уточним, между какими вариантами именно
     for (; j < nx - 1; j++)
       if (yy[i] >= xx[j] && yy[i] < xx[j + 1])
         {
          s = fabs ((double)i / ny - (double)(j + 1) / nx);

          ////////////////////////////////////////////////////////
          // Прервать цикл с сохранением достигнутого номера текущей
          // варианты 2-й выборки - так уменьшим объем вычислений
          break;
         }
     if (s > criterion)
       criterion = s;
    }

////////////////////////////////////////////////
// Перебрать все варианты 1-й выборки относительно
// всех вариант 2-й выборки
for (i = 0, j = 0; i < nx; i++)

  ////////////////////////////////////////////////////////////
  // Текущая варианта 1-й выборки меньше или равна 1-й варианте
  // 2-й выборки
  if (xx[i] <= yy[0])
    {
     s = (double)i / nx;
     if (s > criterion)
       criterion = s;
    }

  ////////////////////////////////////////////////////////
```

```
   // Текущая варианта 1-й выборки больше или равна последней
   // варианте 2-й выборки
   else if (xx[i] >= yy[ny - 1])
     {
      s = 1 - (double) i / nx;
      if (s > criterion)
        criterion = s;

      ////////////////////////////////////////////////
      // Дальше вычисления можно не проводить - максимум
      // уже был достигнут в процессе вычислений
      break;
     }

   ////////////////////////////////////////////////////
   // Текущая варианта 1-й выборки в пределах 2-й выборки
   else
     {
      ////////////////////////////////////////
      // Уточним, между какими вариантами именно
      for (; j < ny - 1; j++)
        if (xx[i] >= yy[j] && xx[i] < yy[j + 1])
          {
           s = fabs ((double)i / nx - (double)(j + 1) / ny);

           ////////////////////////////////////////////////////
           // Прервать цикл с сохранением достигнутого номера текущей
           // варианты 1-й выборки - так уменьшим объем вычислений
           break;
          }
      if (s > criterion)
        criterion = s;
     }

 delete [] xx; delete [] yy;

 return (1 - LambdaDistribution (criterion * sqrt (nx * ny / (nx + ny))))
/ 2;
}
```

Точный метод Фишера — Ирвина

Точный метод Фишера — Ирвина (точный метод Фишера, критерий Фишера — Ирвина) применяется для проверки нулевой гипотезы о том, отобраны ли две исследуемые бинарные (дихотомические) выборки из генеральных совокупностей с одинаковой частотой встречаемости изучаемого эффекта. Рассматриваемый метод предназначен для обработки так называемых четырехпольных (четырехклеточных) таблиц, или таблиц «2 × 2». В результате вычислений получается точное значение уровня значимости нулевой гипотезы. Вычисление производится по формуле

$$\alpha = \frac{(a+b)!(c+d)!(a+c)!(b+d)!}{n!a!b!c!d!},$$

где a — число наблюдений с эффектом A в первой выборке,

b — число наблюдений без эффекта A в первой выборке,

c — число наблюдений с эффектом A во второй выборке,

d — число наблюдений без эффекта A во второй выборке,

$n = a + b + c + d.$

При вычислении получается заниженное значение вероятности, поэтому алгоритм вычисления статистики критерия корректируется следующим образом:

1. По приведенной выше формуле вычисляется вероятность получения исходной таблицы.

2. Рассматриваются все остальные варианты заполнения таблицы, имеющие такие же маргинальные частоты. Для этого находится минимальное значение из всех четырех ячеек таблицы (диагональ, на которой находится ячейка, не должна содержать нулевых значений). Затем в выбранной ячейке проставляются все значения от нуля до заданного. Соответственно корректируются значения в остальных ячейках таблицы, чтобы суммы строк и столбцов были неизменными.

3. Вычисляются вероятности всех вариантов таблицы, построенных в пункте 2.

4. Вычисленные в пункте 3 вероятности, если они не превышают вероятности получения исходной таблицы, суммируются с вероятностью получения исходной таблицы. Сумма представляет собой искомое значение статистики критерия.

Пример. Пусть вычислено значение 0,012 < 0,05. На уровне значимости < 0,05 (ближайшее стандартное значение), нулевая гипотеза отклоняется и принимается альтернативная гипотеза о различии средних значений.

Листинг 5.6

```
double __declspec(dllexport) FisherIrwin (FP *x,FP *y)
//
// Функция вычисления критерия значимости различий
// точным методом Фишера-Ирвина.
// Обозначения:
//  x - первая выборка,
//  y - вторая выборка.
// Возвращаемое значение:
//  p-значение для односторонней гипотезы.
//
{
 short i = 0,                      // Счетчик
       nx = x->rows * x->columns, // Численность первой выборки
       ny = y->rows * y->columns, // Численность первой выборки
       a = 0, // Число наблюдений с эффектом А в первой выборке
       b = 0, // Число наблюдений без эффекта А в первой выборке
       c = 0, // Число наблюдений с эффектом А во второй выборке
       d = 0; // Число наблюдений без эффекта А во второй выборке

 for (i = 0; i < nx; i++)
   if (x->Array[i] == 1)
     a++;
   else
     b++;
 for (i = 0; i < ny; i++)
   if (y->Array[i] == 1)
     c++;
   else
     d++;

 ////////////////////////////////////////
 // Вероятность получения текущей таблицы
 double Buf,

 ////////////////////////////
 // Постоянная часть формулы
```

```
 Const = LogFactorial (a + b) + LogFactorial (c + d) +
         LogFactorial (a + c) + LogFactorial (b + d) -
         LogFactorial (a + b + c + d);

 ////////////////////////////////////////
 // Вероятность получения исходной таблицы
 double Ret = exp (Const - LogFactorial (a) - LogFactorial (b) -
        LogFactorial (c) - LogFactorial (d));
 double Save = Ret;

 ////////////////////////////////////////////////////
 // Перебор всевозможных вариантов заполнения таблицы
 if (a != 0 && d != 0)
   {
    if (a <= d)
      while (a)
        {
         if ((Buf = exp (Const - LogFactorial (--a) - LogFactorial (++b)
-
              LogFactorial (++c) - LogFactorial (--d))) <= Save)
           Ret += Buf;
        }
    else
      while (d)
        {
         if ((Buf = exp (Const - LogFactorial (--a) - LogFactorial (++b)
-
              LogFactorial (++c) - LogFactorial (--d))) <= Save)
           Ret += Buf;
        }
   }
 else if (b != 0 && c != 0)
   {
    if (b <= c)
      while (b)
        {
         if ((Buf = exp (Const - LogFactorial (++a) - LogFactorial (--b)
-
              LogFactorial (--c) - LogFactorial (++d))) <= Save)
```

```
            Ret += Buf;
          }
      else
        while (c)
          {
           if ((Buf = exp (Const - LogFactorial (++a) - LogFactorial (--b)
_
                LogFactorial (--c) - LogFactorial (++d))) <= Save)
            Ret += Buf;
          }
    }

 return Ret;
}
```

Руководство пользователя

Модуль непараметрической статистики

Для работы с модулем непараметрической статистики (NP) выберите из меню Microsoft Excel добавленный программным обеспечением NP новый пункт **AtteStat ▸ Модуль NP ▸ Непараметрическая статистика**. На экране появится диалоговое окно, изображенное на рис. 5.1.

Рис. 5.1. Окно модуля непараметрической статистики

Выберите или введите интервалы сравниваемых выборок, выходной интервал. Начиная с первой ячейки выходного интервала, будут выведены результаты вычислений. Выберите критерий для проведения статистического расчета. Нажмите кнопку **Выполнить расчет**.

Начиная с первой ячейки выходного интервала, будет выведено название статистического критерия, P-значение и вывод о результате проверки статистической гипотезы.

В данном варианте модуль непараметрической статистики берет на себя верификацию исходных данных, выдавая подробную диагностику. Второе достоинство — результаты расчета можно передать коллегам, не имеющим на своих компьютерах программного обеспечения NP.

Недостатки варианта вытекают из преимуществ. Неудобством является более длительный путь получения результатов. К тому же, для каждого набора данных необходимо повторить всю процедуру. Просто скопировать формулу и поместить ее в нужную ячейку, изменив интервалы исходных данных, не удастся. Для этого служит вариант для опытных пользователей, описанный в следующем разделе.

При ошибках ввода могут выдаваться диагностические сообщения следующих типов:

Рис. 5.2. Окно сообщения об ошибке

На рис.5.2 вы не выбрали или неверно ввели входной интервал. Лучшим способом избежать ошибки является не ввод, а выделение интервала протаскиванием курсора.

Рис. 5.3. Окно сообщения об ошибке

На рис.5.3 проверьте исходные данные и заполните все ячейки входного интервала. Если в ячейке не должно быть данных по физической природе исследуемого процесса, введите нуль.

Рис. 5.4. Окно сообщения об ошибке

Проверьте типы ячеек входного интервала на рис.5.4. Тип может быть только числовым. Проще всего выделить интервал ячеек и явно определить их тип как числовой стандартными средствами Microsoft Excel.

Рис. 5.5. Окно сообщения об ошибке

На рис.5.5 не выбран или неверно введен выходной интервал. Лучшим способом избежать ошибки является не ввод, а выделение интервала протаскиванием курсора.

Рис. 5.6. Окно сообщения об ошибке

На рис.5.6 выборки разной численности, следует выбирать попарно сопряженные. Отметим, что при попытке сравнения выборок разной численности парным методом (см. следующий раздел),вместо такого диагностического сообщения ожидает сюрприз: с целью предупреждения выхода за границы массивов, численности обеих выборок берутся равными численности наименьшей выборки.

Рис. 5.7. Окно сообщения об ошибке

На рис.5.7 выбраны слишком большие (например, более 12 для критериев рандомизации) численности для данного метода расчета. Из-за особенностей реализации, вычисление превысит допустимую продолжительность расчета. Кроме того, без потери точности выводов, вполне может быть выбран метод не столь трудоемкий (для рассмотренного примера — соответствующие критерии Вилкоксона). Рекомендации по минимальным и максимальным численностям анализируемых выборок даны при описании соответствующих критериев.

Модуль непараметрической статистики (для опытных пользователей)

Работа с функциями модуля непараметрической статистики не отличается от работы со стандартными функциями, Мастер функций запускается выбором пункта **Вставка ▸ Функция**. На экране появится окно, как на рис. 5.8.

Рис. 5.8. Окно Мастера функций при работе с пакетом NP в Microsoft Excel 2000

Следует выбрать категорию **Непараметрическая статистика** и нужную функцию, после чего нажать кнопку **OK** для перехода к следующему шагу. На шаге 2 из 2 ввести или выбрать все необходимые параметры в соответствующих полях окна Мастера функций. В данном случае, это будут интервалы ячеек массивов сравниваемых выборок. После ввода или выбора нажать кнопку **OK**. Будет выполнен требуемый расчет, а результат помещен в ячейку, в которую была введена функция. Дополнительные сведения о порядке вызова функций можно получить из справочной системы Microsoft Excel.

Сообщения об ошибках в рассматриваемом варианте для опытных пользователей не выдаются, как и выводы по результатам расчета.

Пример применения

Применим методы математической статистики для ответа на вопрос, влияет ли уровень социально-экономического развития регионов России на предпочтения электората, по данным Центральной избирательной комиссии России на **www.izbirkom.ru**.

Для анализа выберем два региона (Москва и Курганская область), с различиями в минимальном прожиточном уровне, среднем уровне заработной платы, в медицинском и социальном обеспечении, экономическом и финансовом развитии. Анализируя исходные данные, можно выделить в Курганской области активность протестного электората, а в Москве — выбирающего стабильность. Посмотрим, так ли это на самом деле.

Введем в Microsoft Excel (с установленным модулем непараметрической статистики) результаты из сводной таблицы итогов голосования по Московскому и Курганскому избирательному округу. Для компенсации различной численности электората будем исследовать исходные данные в процентах (рис.5.9).

	A	B	C	D	E	F
1	Федеральный список	Москва	Курган			
2	Единение	1,17	0,94	Т-критерий Вилкоксона для связанных выборок		
3	Союз правых сил	1,84	1,4	p-значение	0,437562424	
4	Российская партия пенсионеров и партия социальной справедливости	2,37	2,84	Различия недостоверны		
5	Российская демократическая партия «Яблоко»	2,68	3,06			
6	За Русь святую	0,37	0,29	Парный двухвыборочный t-тест для средних		
7	Объединенная Российская партия «Русь»	0,21	0,24			
8	Новый курс – автомобильная Россия	0,57	0,56		*Переменная 1*	*Переменная 2*
9	Народно-республиканская партия России	0,14	0,12	Среднее	4,087916667	4,120416667
10	Российская экологическая партия «Зеленые»	0,35	0,24	Дисперсия	139,7164694	63,04521286
11	Аграрная партия России	0,75	6,19	Наблюдения	24	24
12	Истинные патриоты России	0,19	0,12	Корреляция Пирсона	0,872052898	
13	Народная партия Российской Федерации	0,7	1,84	Гипотетическая разнос	0	
14	Демократическая партия России	0,15	0,23	df	23	
15	Великая Россия – Евразийский Союз	0,19	0,17	t-статистика	-0,025471766	
16	Партия СЛОН	0,27	0,1	P(T<=t) одностороннее	0,489949191	
17	Родина (народно-патриотический союз)	3,84	5,47	t критическое одностор	1,713870006	
18	Партия Мира и Единства (ПМЕ)	0,24	0,23	P(T<=t) двухстороннее	0,979898383	
19	ЛДПР	5,47	19,17	t критическое двухстор	2,068654794	
20	Партия Возрождения России – Российская партия жизни	2,55	2,05			
21	Политическая партия «Единая Россия»	58,65	33,59			
22	Российская конституционно-демократическая партия	0,26	0,17			
23	Развитие предпринимательства	1,71	0,23			
24	Коммунистическая партия Российской Федерации (КПРФ)	9,04	15,82			
25	Против всех	4,4	3,82			

Рис. 5.9. Статистический контроль результатов выборов в Государственную думу

Выполним анализ различий между выборками парным критерием Вилкоксона. В результате расчета получим Р-значение, равное 0,4376. Различия статистически не доказаны при $p > 0,05$. Для контроля используем парный двухвыборочный t-тест для средних из пакета анализа Microsoft Excel. Различия также статистически не доказаны.

Итак, уровень социально-экономического развития регионов России на предпочтения электората не влияет.

Рекомендуемая литература

1. Бейли Н. Математика в биологии и медицине. — М.: Мир, 1970. Partial available from World Wide Web: < **http://www.statsoft.ru/home/portal/applications/medicine/ methods.htm** >.
2. Бейли Н. Статистические методы в биологии. — М.: Мир, 1963.
3. Биллиг В.А. Мир объектов Excel 2000. — М.: Издательско-торговый дом «Русская редакция», 2001.
4. Большев Л.Н., Смирнов Н.В. Таблицы математической статистики. — М.: Наука, 1983.
5. Браунли К.А. Статистическая теория и методология в науке и технике. — М.: Наука, 1977.
6. Вероятность и математическая статистика: Энциклопедия. — М.: Научное издательство «Большая Российская энциклопедия», 1999.
7. Гайдышев И. Анализ и обработка данных: специальный справочник. — СПб: Питер, 2001.
8. Макарова Н.В., Трофимец В.Я. Статистика в Excel: Учебное пособие. — М.: Финансы и статистика, 2002.
9. Медик В.А., Токмачев М.С., Фишман Б.Б. Статистика в медицине и биологии: Руководство. В 2–х томах / Под ред. Ю.М. Комарова. Т.1. Теоретическая статистика. — М.: Медицина, 2000.
10. Мюллер П., Нойман П., Шторм Р. Таблицы по математической статистике. — М.: Финансы и статистика, 1982.
11. Поллард Дж. Справочник по вычислительным методам статистики. — М.: Финансы и статистика, 1982.
12. Романовский В.И. Математическая статистика. Кн.2. Оперативные методы математической статистики. — Ташкент: Издательство Академии наук УзССР, 1963.
13. Рунион Р. Справочник по непараметрической статистике. — М.: Финансы и статистика, 1982.

14. Сергиенко В.И., Бондарева И.Б. Математическая статистика в клинических исследованиях — М.: ГЭОТАР–МЕД, 2001.
15. Тюрин Ю.Н., Макаров А.А. Статистический анализ данных на компьютере. — М.: ИНФРА-М, 1999.
16. Уткин В.А. Статистические технологии в медицинских исследованиях: Монография. — Пятигорск: ГНИИК, 2002.
17. Флейс Дж. Статистические методы для изучения таблиц долей и пропорций. — М.: Финансы и статистика, 1989.
18. Хеттманспергер Т. Статистические выводы, основанные на рангах. — М.: Финансы и статистика, 1987.
19. Холлендер М., Вулф Д. Непараметрические методы статистики. — М.: Финансы и статистика, 1983.

ГЛАВА 6

Многомерный анализ: факторный анализ в VBA и C

«Открытие какого-либо порядка вещей непросто, но если он найден, нет никаких трудностей в его понимании».
Р. Декарт

Для работы с материалами главы понадобятся лицензионные средства разработки и тестирования программ:

- Microsoft Excel 97 или выше;
- Microsoft Visual C++ 5.0 или выше;
- Borland C++ 5.5 или выше.

Потребуется знание VBA, C.

Решаемая проблема

Методами факторного анализа решаются основные задачи:

- отыскание скрытых, но предполагаемых закономерностей воздействия внутренних или внешних причин (факторов) на изучаемый процесс;
- выявление и изучение статистической связи признаков с факторами и главными компонентами;
- сжатие информации путем интерполяции общими факторами или главными компонентами, с той или иной степенью точности воспроизводящими корреляционную матрицу.

Применение факторного анализа оказалось особенно успешным при анализе научных исследовательских психологических, медицинских и санитарных данных. Для успешного применения методик лечения всегда важно знать о существовании факторов, определяющих причину заболеваний.

Основным объектом исследования является корреляционная матрица, построенная с использованием коэффициента корреляции Пирсона для количественных признаков (см. *главу 8*). Существуют другие коэффициенты типа корреляции, предназначенные для порядковых, качественных и смешанных признаков. Основным требованием является положительная полуопределенность матрицы, и как следствие, неотрицательность всех собственных значений. Требование обычно нарушается в двух случаях: при некорректной редукции и в случае использования коэффициентов корреляции для признаков, измеренных в разных шкалах. Матрица, основанная на коэффициенте корреляции Пирсона, по условиям своего построения всегда положительно полуопределена. Для редуцированной корреляционной матрицы, данное утверждение не всегда является верным. Поэтому вначале всегда проверяется положительная полуопределенность матрицы.

Коэффициенты корреляции в матрице, по умолчанию, вычисляются между параметрами. Это *R*-факторный анализ. Если исследуется корреляция между объектами, анализ называется техникой *Q*, когда предполагается, что состояния объектов могут иметь общую побудительную причину. Существует также техника *P*, предполагающая анализ исследований, выполненных на одном и том же индивидууме в различные промежутки времени (объекты — один и тот же индивидуум в различные промежутки времени), причем изучаются корреляции между состояниями индивидуума. Аналог техники *Q* для последнего случая составляет предмет исследования техники *O*.

Применение техники *R* (*P*), *Q* (*O*), *R* (*Q*) или *P* (*O*) происходит в пределах одной программы изменением исходных данных. За выбор отвечает флаг использования техники 1 или 2 в листинге 6.1 (см. *рис. 6.2*).

Примечание

Выбор исходных данных производится в зависимости от техники факторного анализа, повышая функциональность программы.

Рассмотрим методы факторного анализа: метод главных факторов и метод максимального правдоподобия, предположив, что изучаемая зависимость носит линейный характер.

Основное требование к исходным данным, если производится факторный анализ на основе корреляционной матрицы, составленной из коэф-

фициентов корреляции Пирсона (см. *главу 8*), — это то, что они должны подчиняться многомерному нормальному распределению. Т. е., для каждой пары выборок, по которой вычисляется коэффициент корреляции, должна быть проверена нормальность распределения двухмерной выборки. Для проверки нормальности многомерной выборки применяются критерии, представленные в *главе 3* и в соответствующем модуле программного обеспечения AtteStat.

Получение матрицы факторного отображения, в принципе, является целью факторного анализа. Ее строки представляют собой координаты векторов в *r*-мерном факторном пространстве. Каждый вектор несет информацию о процессе. Взаимная ориентация векторов дает представление о взаимной зависимости переменных.

Если число выделенных факторов больше единицы, обычно производится вращение матрицы факторного отображения с целью получения так называемой простой структуры.

В процессе решения задачи факторного анализа нужно быть готовым к тому, что иногда решение получить не удается. Данное замечание в большей степени относится к методу максимума правдоподобия, в схеме которого фигурирует матрица, обратная корреляционной. Это вызвано сложностью решаемой проблемы собственных значений корреляционной матрицы. Например, матрица может оказаться вырожденной, что может быть вызвано совпадением или полной линейной корреляцией параметров. Для матриц высокого порядка может произойти потеря значимости в процессе вычислений. Поэтому теоретически нельзя исключить ситуацию, когда методы факторного анализа окажутся неприменимы.

Особенности решения

Модуль факторного анализа является одним из самых объемных и насыщенных сложными алгоритмами.

Регистрация функций DLL выполнена в модуле VBA. В *главе 5*, например, результатом вычислений любым методом всегда является одна скалярная величина — P-значение. Результатом работы программ факторного анализа являются не скалярные значения, а массивы. Даже один массив нельзя без дополнительных усилий передать с помощью макросов. Поэтому регистрация экспортируемых DLL функций VBA-оператором Declare была логичным решением. Ниже дано комплектное решение, которое одинаково работает и в модулях VBA, и, при необходимости, в программе VB. Хотелось бы обратить внимание читателей на данную возможность использование разработанных нами DLL.

Научные алгоритмы

Листинги основных функций расчета даны далее. Полные исходные тексты всех функций имеются на прилагаемом компакт-диске.

Метод главных факторов

Рассмотрим подробнее метод главных компонент — вариант метода главных факторов.

Основная модель метода главных компонент (компонентного анализа) записывается в матричном виде следующим образом:

$Z = AP$,

где Z — матрица стандартизованных исходных данных,

A — матрица факторного отображения,

P — матрица значений факторов.

Матрица Z имеет размер $m \times n$, матрица A — размер $m \times r$, матрица P — размер $r \times n$, где

m — количество переменных (векторов данных),

n — количество индивидуумов (элементов одного вектора),

r — количество выделенных факторов.

Примечание

Обратим внимание читателей, что в матричных выражениях мы специально указываем размеры матриц. Данный простой прием помогает контролировать правильность записи выражений и программного кода.

Модель компонентного анализа содержит только общие для имеющихся векторов факторы.

Матрица стандартизованных исходных данных определяется из матрицы исходных данных Y (ее размер $m \times n$) по формуле

$$z_{ij} = \frac{y_{ij} - \overline{y}_i}{s_i}, i = 1,2,...,m, j = 1,2,...,n,$$

где y_{ij} — элемент матрицы исходных данных,

$\overline{y}_{ij}$ — среднее значение,

s_i — стандартное отклонение.

Для вычисления корреляционной матрицы — основного элемента факторного анализа — имеет место простое соотношение:

$$\frac{1}{n-1}ZZ' = R,$$

где R — корреляционная матрица; она имеет размер $m \times m$,

$'$ — символ транспонирования.

В программе корреляционная матрица вычисляется с помощью функций, показанных на листинге 6.1. В *главе 8* можно посмотреть альтернативные решения на языке C++. Обратите внимание на существенные различия в вызовах из VBA программы функций, составленных на VBA, и составленных на C/C++ и помещенных в DLL. Особенности вызова функций из DLL рассмотрены в *главе 2*.

Листинг 6.1

```
Private Function Pearson(X() As Double, Y() As Double, N As Integer) As
Double
'
' Функция вычисляет значение коэффициента корреляции Пирсона.
' Обозначения:
'  X — первая выборка,
'  Y — вторая выборка,
'  N — численность каждой выборки.
' Возвращаемое значение:
'  значение выборочного коэффициента корреляции.
'
 Pearson = Cov(X(), X(), N) * Cov(Y(), Y(), N)
 If Not Pearson = 0 Then
   Pearson = Cov(X(), Y(), N) / Sqr(Pearson)
 End If
End Function


Private Function Cov(X() As Double, Y() As Double, N As Integer) As
Double
'
' Функция вычисляет значение выборочной ковариации.
' Обозначения:
'  X — первая сравниваемая выборка,
```

```
'  Y — вторая сравниваемая выборка,
'  N — численность каждой выборки.
' Возвращаемое значение:
'  значение выборочной ковариации.
'
 Dim I As Integer ' Счетчик
 Dim Xm As Double ' Среднее первой выборки
 Dim Ym As Double ' Среднее первой выборки
 Xm = VectorSum(X(1), N) / N
 Ym = VectorSum(Y(1), N) / N
 Cov = 0
 For I = 1 To N
  Cov = Cov + (X(I) — Xm) * (Y(I) — Ym)
 Next I
 Cov = Cov / (N — 1)
End Function
```

На главной диагонали матрицы R значения равны 1. Эти значения называются общностями и обозначаются как h_i^2, являясь мерой полной дисперсии переменной. Для метода главных факторов, общности отличны от 1 и вычисляются определенным образом, рассматриваемым далее.

Неизвестными являются матрицы A и P. Матрица A может быть найдена из основной теоремы факторного анализа

$$R = ACA',$$

где C — корреляционная матрица, отражающая связь между факторами.

Если $C = I$, то говорят об ортогональных факторах, если $C \neq I$, говорят о косоугольных факторах. Здесь I — единичная матрица.

Для матрицы С справедливо соотношение:

$$\frac{1}{n-1} PP' = C .$$

Нами рассматривается только случай ортогональных факторов, для которых

$$R = AA'.$$

Модель классического факторного анализа содержит ряд общих факторов и по одному характерному фактору на каждую переменную.

Первая из приведенных в разделе формул является основной моделью факторного анализа для метода главных компонент. Число главных компонент всегда меньше либо равно числу переменных.

Для метода главных факторов основная модель записывается в виде

$$Z = FP^{+},$$

где F — полная факторная матрица,

P^{+} — матрица значений факторов, включая значения характерных факторов.

Матрица F имеет размер m x $(r + m)$, матрица P^{+} имеет размер $(r + m) \times n$.

Матрица F может быть представлена в виде суммы двух матриц

$$F = A + U,$$

где A — матрица нагрузок общих факторов,

U — матрица нагрузок характерных факторов.

Очевидно, что матрицы A и U имеют размер матрицы F.

Полная дисперсия переменной складывается из общности h_i^2, значение которой меньше либо равно 1 и означает часть полной дисперсии переменной, приходящейся на главные факторы, и характерности, обозначаемой как u_i^2, приходящейся на характерные факторы. Следовательно,

$$u_i^2 = 1 - h_i^2 .$$

Блок матрицы U является нулевым, другой блок (размером $m \times m$) представляет собой диагональную матрицу квадратных корней из характерностей, которые уже вычислены из общностей. Таким образом, может быть определена матрица U и, следовательно, матрица F, если известна матрица A.

В случае ортогональных факторов, можно записать

$$R = FF'.$$

Развернув выражение, получим:

$$R = R_h + U^2,$$

где R — корреляционная матрица с единицами на главной диагонали,

R_h — корреляционная матрица с общностями на главной диагонали, определяемая выражением

$$R_h = AA',$$

U^2 означает UU'.

На листинге 6.2 представлена функция факторного анализа методом главных факторов, адаптированная к вызову из программ на различных языках программирования, включая VB и VBA.

Листинг 6.2

```
short __stdcall PrincipalFactor (double z[],short k,double r[],short *m)
//
// Функция производит факторный анализ методом главных факторов.
// Обозначения:
//   z - корреляционная матрица,
//   k - порядок корреляционной матрицы,
//   r - матрица факторного отображения размером не менее k * k,
//  *m - число факторов.
// Возвращаемое значение:
//   0 при нормальном окончании счета,
//  -1 при недостатке памяти для рабочих массивов,
//  -2 при ошибке в вычислениях.
//
{
 short i,j; // Счетчики

 short m1,Code;      // Вспомогательная переменная

 double *z1,       // Рабочая копия корреляционной матрицы
        save;      // Буфер

 *m = 0;

 if ((z1 = new double[k * k]) == 0)
   return -1;

 ////////////////////////////////////
 // Рабочая копия корреляционной матрицы
 ArrayToArray (z,z1,k * k);

 ////////////////////////////
 // Решение проблемы факторов
 if ((Code = Jacobi (z1,k,0.000001,&m1,r)) != 0)
   {
    delete [] z1;
```

```
   return Code;
  }

////////////////////////////////////////////////
// Упорядочивание собственных значений и векторов
for (i = 1; i < k; i++)
  z1[i] = z1[i * (k + 1)];  // Первый столбец -
                            // собственные значения

for (i = 0; i < k; i++)
  {
   Code = i;
   save = z1[i];
   for (j = i + 1; j < k; j++)
     if (save < z1[j])
       {
        Code = j;
        save = z1[j];
       }

   ColInterchange (z1,1,i,Code);
   ColInterchange (r,k,i,Code);
  }

////////////////////////////
// Отбор собственных значений
for (i = 0, m1 = 0; i < k; i++)
  if (z1[i] < 0)
    break;
  else
    {
     (*m)++;

     ////////////////////////////////////////////
     // Вычисление матрицы факторного отображения
     save = sqrt (z1[i]);
     for (j = 0; j < k; j++, m1++)
```

```
        r[m1] *= save;
    }

 delete [] z1;

 return 0;
}
```

Проблема общности

Редуцированием (редукцией) корреляционной матрицы называется процесс оценки общностей, то есть замены единиц на главной диагонали корреляционной матрицы некоторыми величинами, называемыми общностями. Общность — это сумма квадратов факторных нагрузок. Иначе, та часть дисперсии, которая обусловлена общими факторами. Это вытекает из предположения, что полная дисперсия складывается из общей дисперсии и специфичной дисперсии, обусловленной факторами для данной переменной, и дисперсии, обусловленной ошибкой.

Для метода главных факторов имеет место проблема общности, то есть на главной диагонали корреляционной матрицы, в отличие от метода главных компонент, необходимо проставить значения общностей, чтобы получить редуцированную корреляционную матрицу R_h. Без редукции, то есть с единицами на главной диагонали корреляционной матрицы, мы получаем метод главных компонент. Были рассмотрены четыре основных способа оценки общностей:

- способ наибольшей корреляции;
- коэффициент множественной корреляции, при этом общности вычисляются с помощью выражения

  $$h_i^2 = 1 - \frac{1}{r_{ii}},$$

  где в знаменателе стоит диагональный элемент матрицы, обратной к матрице R_h;
- средние по столбцу корреляционной матрицы коэффициенты корреляции;
- метод триад.

На листинге 6.3 представлена функция редукции корреляционной матрицы, адаптированная к вызову из программ на различных языках программирования, включая VB и VBA.

Листинг 6.3

```
short __stdcall Reduction (double z[], double r[],short k,short Mode)
//
// Функция производит редукцию корреляционной матрицы.
// Обозначения:
//     z - корреляционная матрица,
//     r - редуцированная корреляционная матрица,
//     k - размерность корреляционной матрицы,
//  Mode - метод вычисления общностей:
//       = 0 - анализ главных компонент,
//       = 1 - способ наибольшей корреляции,
//       = 2 - коэффициент множественной корреляции,
//       = 3 - средние коэффициенты корреляции,
//       = 4 - метод триад.
// Возвращаемое значение:
//   0 при нормальном окончании расчета.
//  -1 при недостатке памяти,
//  -2 при ошибке в параметрах, например, k >= 3 для метода триад.
//
{
 short i,j;        // Счетчики

 short m,m1,k1,k2; // Счетчики одномерного массива

 double *t,        // Вспомогательный массив
        s,s1,s2;   // Вспомогательные переменные

 if ((t = new double[k * k]) == 0)
   return -1;

 ////////////////////////////////
 // Копия корреляционной матрицы
 ArrayToArray (z,r,k * k);

 ////////////////////////
 // Определение общностей
 switch (Mode)
```

```
{
 case 0:
   ////////////////////////////////////////
   // Нет редукции - анализ главных компонент
   break;
 case 1:
   ////////////////////////////////
   // Способ наибольшей корреляции
   for (i = 0, m = 0, m1 = 0; i < k; i++)
     {
      for (j = 0; j < k; j++, m++)
        t[j] = i == j ? 0 : fabs (z[m]);
      r[m1] = MaximumOf (t,k);
      m1 += k + 1;
     }
   break;
 case 2:
   ////////////////////////////////////////
   // Коэффициент множественной корреляции
   if (InverseMatrix (r,t,k) == 0)
     for (i = 0, m = 0; i < k; i++,m += k + 1)
       if (t[m] != 0 || t[m] != 1)
         r[m] = 1 - 1 / t[m];
   break;
 case 3:
   ////////////////////////////////////
   // Средние коэффициенты корреляции
   for (i = 0, m = 0, m1 = 0; i < k; i++,m1 += k + 1)
     {
      for (j = 0, r[m1] = 0; j < k; j++, m++)
        r[m1] += i == j ? 0 : fabs (z[m]);
      r[m1] /= k - 1;
     }
   break;
 default:
   //////////////
   // Метод триад
   if (k < 3)
```

```
        {
         delete [] t;
         return -2;
        }

      for (i = 0, m = 0, m1 = 0; i < k; i++,m1 += k + 1)
        {
         ////////////////////////////////
         // Поиск двух максимумов в столбце
         for (j = 0; j < k; j++)
           t[j] = fabs (z[m++]);
         t[i] = 0;
         s1 = MaximumOf (t,k,&k1);
         t[k1] = 0;
         s2 = MaximumOf (t,k,&k2);

         ////////////////////////////////////////////
         // Корреляция между максимальными параметрами
         s = fabs (z[k1 + k2 * k]);

         if (s != 0)
           r[m1] = s1 * s2 / s;
        }
    }

 delete [] t;

 return 0;
}
```

Вычисление матрицы факторного отображения

В методах главных факторов и главных компонент матрица факторного отображения определяется из решения проблемы собственных значений корреляционной матрицы R или, соответственно, редуцированной корреляцион-

ной матрицы R_h. Для простоты записи значок h далее опускаем, имея в виду R или R_h в зависимости от применяемой разновидности метода факторного анализа.

Факторы пропорциональны собственным векторам матрицы R. Стандартная проблема собственных значений матрицы R записывается в виде:

$$(R - \lambda_l I) = 0,$$

где $\lambda_l, l = 1,2,...,m$, — собственные значения матрицы R,

l — номер собственного значения.

Соответствующий l-му собственному значению собственный вектор обозначается a_l.

Задача упрощается тем, что матрица R является действительной и симметрической, поэтому для решения проблемы собственных значений применимы эффективные устойчивые алгоритмы. К таким алгоритмам относятся, прежде всего, метод Якоби и QR-алгоритм Франсиса и Кублановской.

Метод максимума правдоподобия

Метод главных факторов уступает методу максимума правдоподобия в том, что он не позволяет получить точной оценки общности и добиться, таким образом, полной воспроизводимости корреляционной матрицы.

Поясним, что мы понимаем под полной воспроизводимостью. Если количество факторов равно числу переменных, справедливо тождество

$$R = AA',$$

где R — корреляционная матрица с единицами на главной диагонали,

A — вычисленная тем или иным методом матрица факторного отображения.

В этом случае говорят, что получена полная воспроизводимость корреляционной матрицы. Термин не следует понимать в том смысле, что по корреляционной матрице, не привлекая дополнительной информации, можно точно восстановить матрицу исходных данных или матрицу стандартизованных исходных данных.

Если количество факторов меньше числа переменных, тождество

$$R_h = AA'$$

где R_h — корреляционная матрица с общностями на главной диагонали,

будет справедливым только в случае, если мы получим точные оценки общностей. Именно поэтому в методе главных факторов пытаются, чаще всего безуспешно, получить точные оценки общностей. Ведь если общности не

будут вычислены точно, тождество будет нарушено (если не сильно, то результатами факторного анализа можно пользоваться и в этом случае).

Схема алгоритма метода максимума правдоподобия заключается в следующем. Сначала делается произвольная оценка матрицы факторного отображения, обозначенная как $A_{1/2}$, например, полученная методом главных компонент. Затем запускается итерационный процесс:

1. вычисляется диагональная матрица характерностей

$$D_i^2 = diag(I - A_{i-1/2}A'_{i-1/2}),$$

где i — здесь и далее номер итерации;

2. вычисляется вспомогательная матрица

$$J_{i-1/2} = A'_{i-1/2}D_i^{-2}A_{i-1/2};$$

3. вычисляется диагональная матрица собственных значений матрицы $J_{i-1/2}$ по формуле

$$J_i = Q'_i J_{i-1/2} Q_i,$$

где Q_i — матрица, столбцы которой суть нормализованные собственные векторы матрицы $J_{i-1/2}$, вычисленной на шаге 2;

4. выполняется вращение матрицы факторного отображения по формуле

$$A_i = A_{i-1/2}Q_i;$$

5. вычисляется новая матрица факторного отображения

$$A_{i+1/2} = (RD_i^{-2} - I)A_i J_i^{-1};$$

6. производится проверка сходимости

$$\left|A_{i+1/2} - A_{i-1/2}\right| < \varepsilon,$$

где ε — малое число, обычно равное 0,001.

Шаги 1–5 повторяются, пока проверка сходимости на шаге 6 не будет успешно выполнена.

В методе максимума правдоподобия предварительная оценка общностей не производится. Общности находятся в результате вычислений, из условия полной воспроизводимости корреляционной матрицы в указанном выше смысле. Вычисленные в результате применения метода максимума правдоподобия общности определяют 100% дисперсии, приходящейся на главные факторы. В этом заключается основное преимущество метода максимума правдоподобия перед методом главных факторов. Корреляционная матрица (при числе факторов, равном числу переменных — с единицами на главной

диагонали; при числе факторов, меньших числа переменных — с общностями на главной диагонали) воспроизводится с точностью до ошибки вычислений.

Недостаток метода максимума правдоподобия — неустойчивость к данным, содержащим совпадающие или линейно зависимые векторы переменных. С точки зрения многомерной статистики, в данном случае, даже многомерное нормальное распределение будет вырожденным. С точки зрения факторного анализа, будет вырожденной матрица характерностей. Вряд ли можно избежать численной неустойчивости. Если же это произошло, следует воспользоваться методом главных факторов либо одним из способов выделения и последующего исключения из рассмотрения линейно зависимых переменных.

На листинге 6.4 представлена функция факторного анализа методом максимума правдоподобия, адаптированная к вызову из программ на различных языках программирования, включая VB и VBA.

Листинг 6.4

```
short __stdcall MaximumLikelihood (double z[],short k,double r[],double
b[],short mz,short *ms)
//
// Функция производит факторный анализ методом максимума
// правдоподобия.
// Обозначения:
//    z - нередуцированная корреляционная матрица,
//    k - размер корреляционной матрицы,
//    r - матрица факторного отображения размером не менее k * k,
//    b - вычисленный вектор общностей,
//   mz - заданное число факторов,
//  *ms - вычисленное число факторов.
// Примечание:
//  если заданное число факторов велико, то оно корректируется.
// Возвращаемое значение:
//   0 - при нормальном окончании счета,
//  -1 - при недостатке памяти для рабочих массивов,
//  -2 - при ошибке в вычислениях.
//
{
 short i,        // Счетчик цикла
       m,        // Число факторов
```

```
      Code,     // Код ошибки
      m1;       // Счетчик одномерного массива

double *a,*am, // Факторная матрица
       *d,*dm, // Матрица характерностей и обратная ей
       *t,     // Вспомогательная матрица
       buf;    // Сумма для проверки сходимости

//////////////////////////////
// Вычисление m главных компонент
if ((Code = PrincipalFactor (z,k,r,&m)) != 0)
  return Code;

if ((a = new double[k * k]) == 0)
  return -1;
if ((am = new double[k * k]) == 0)
  {
   delete [] a;
   return -1;
  }
if ((d = new double[k * k]) == 0)
  {
   delete [] a; delete [] am;
   return -1;
  }
if ((dm = new double[k * k]) == 0)
  {
   delete [] a; delete [] d; delete [] am;
   return -1;
  }
if ((t = new double[k * k]) == 0)
  {
   delete [] a; delete [] d; delete [] dm; delete [] am;
   return -1;
  }

//////////////////////////////
// Корректировка числа факторов
```

```
if (mz < m && mz > 0)
  {
   ////////////////////////////
   // Заданное количество факторов
   *ms = mz;
   m = mz;
  }
else
  ////////////////////////////////
  // Вычисленное количество факторов
  *ms = m;

FillUp (b,k,1);

//////////////////////////////////////////////
// Начало итераций - матрица главных компонент
ArrayToArray (r,am,k * m);

do
  {
   ////////////////////////////////////////
   // Матрица A для следующего шага итераций
   // Здесь a - матрица A(i - 1/2), как альтернатива,
   // можно применить функцию ArrayToArray
   for (i = 0; i < k * m; i++)
     a[i] = am[i];

   ////////////////////////////////
   // Определение матрицы общностей
   MatrixMultiplyMatrixT (a,a,d,k,m,k);

   for (i = 0, m1 = 0; i < k; i++, m1 += k + 1)
     {
      ////////////////////////////
      // Вектор общностей для вывода
      b[i] = d[m1];

      //////////////////////////
```

```
    // Матрица характерностей D
    d[m1] = 1 - d[m1];
   }

////////////////////////////////////////////////
// Определение обратной матрицы характерностей DM
if ((Code = InverseMatrix (d,dm,k)) != 0)
   {
    delete [] d; delete [] t; delete [] dm; delete [] am;
    delete [] a;
    return Code;
   }

//////////////////////////////////
// Определение матрицы J(i - 1/2)

// Здесь d - матрица J(i - 1/2)
MatrixTMultiplyMatrix (a,dm,t,m,k,k);
MatrixMultiplyMatrix (t,a,d,m,k,m);

//////////////////////////////////////////////////////////
// Вращение матрицы J(i - 1/2) - определение матрицы J(i)
// Здесь t - матрица вращения Q
//       d - матрица J(i)
if ((Code = Jacobi (d,m,0.000001,&m1,t)) != 0)
   {
    delete [] d; delete [] t; delete [] dm; delete [] am;
    delete [] a;
    return Code;
   }

////////////////////////////////////////
// Определение факторной матрицы A(i)
MatrixMultiplyMatrix (a,t,r,k,m,m);

////////////////////////////////////////
// Определение обратной матрицы J(i)
// Здесь t - обратная матрица J(i)
```

```
    if ((Code = InverseMatrix (d,t,m)) != 0)
      {
       delete [] d; delete [] t; delete [] dm; delete [] am;
       delete [] a;
       return Code;
      }

    ////////////////////////////////
    // Определение матрицы A(i + 1/2)
    // Здесь am - матрица A(i + 1/2)
    MatrixMultiplyMatrix (z,dm,am,k,k,k);
    for (i = 0, m1 = 0; i < k; i++, m1 += k + 1)
      am[m1] -= 1;
    MatrixMultiplyMatrix (am,r,d,k,k,m);
    MatrixMultiplyMatrix (d,t,am,k,m,m);

    ////////////////////////////
    // Условие окончания расчета
    for (i = 0, buf = 0; i < k * m; i++)
      buf += fabs (am[i] - a[i]);

   } while (buf < 0.001);

 delete [] d; delete [] t; delete [] dm; delete [] am; delete [] a;

 return 0;
}
```

Вращение матрицы факторного отображения

Оси координат выделенных факторов ортогональны, их направления устанавливаются последовательно, по максимуму оставшейся дисперсии. Но полученные таким образом координатные оси, большей частью, содержательно не интерпретируются. Поэтому предпочтительны системы координат, повернутые вокруг ее начала. Пространственная конфигурация векторов в результате применения этой процедуры остается неизменной. Цель вращения — получение так называемой простой факторной структуры. Мы применяем популярный метод вращения варимакс (VARIMAX), алгоритм которого заключается в следующем.

Пусть нормализованные (деленные на соответствующие общности) факторные нагрузки параметра z_j, $j = 1,,n$, при данной паре факторов p и q обозначены через x_j и y_j, а факторные нагрузки при повернутых факторах — через X_j и Y_j. Тогда соответствующее преобразование имеет вид

$$(X_j, Y_j) = (x_j, y_j)\begin{bmatrix} \cos\varphi & -\sin\varphi \\ \sin\varphi & \cos\varphi \end{bmatrix},$$

где φ — угол вращения в плоскости факторов p и q.

Тангенс учетверенного искомого угла вращения равен

$$tg4\varphi = \frac{D - 2\dfrac{AB}{n}}{C - \dfrac{(A^2 - B^2)}{n}},$$

где $A = \sum_{j=1}^{n} u_j$, $B = \sum_{j=1}^{n} v_j$, $C = \sum_{j=1}^{n}(u_j^2 - v_j^2)$, $D = 2\sum_{j=1}^{n} u_j v_j$,

$u_j = x_j^2 - y_j^2$, $v_j = 2x_j y_j$.

Следует брать такие из решений последнего выражения, чтобы угол φ находился в интервале $[-\pi/4;+\pi/4]$. Для этого вычисляются отдельно числитель и знаменатель формулы, а затем по таблице 6.1 ищется необходимый интервал.

Таблица 6.1. *Угол варимакс-вращения*

Числитель	**Знаменатель**	**Квадрант угла** 4φ		**Интервал** φ
+	+	I:	$0 < 4\varphi < \pi/2$	$0 < \varphi < \pi/8$
+	-	II:	$\pi/2 < 4\varphi < \pi$	$\pi/8 < \varphi < \pi/4$
-	-	III:	$-\pi < 4\varphi < -\pi/2$	$-\pi/4 < \varphi < -\pi/8$
-	+	IV:	$-\pi/2 < 4\varphi < 0$	$-\pi/8 < \varphi < 0$

Итерационный процесс продолжается, пока не будет получена сходимость по улучшенному варианту варимакс-критерия, предложенного Кайзером, так чтобы финальные факторные нагрузки отвечали максимуму функции

$$V = n\sum_{p=1}^{m}\sum_{j=1}^{n}\left(\frac{b_{jp}}{h_j}\right)^4 - \sum_{p=1}^{m}\left(\sum_{j=1}^{n}\frac{b_{jp}^2}{h_j^2}\right)^2.$$

На листинге 6.5 представлена функция варимакс-вращения, адаптированная к вызову из программ на различных языках программирования, включая VB и VBA. Отметим, что популярный квартимакс-метод является упрощенным вариантом описанного критерия. При необходимости, функция квартимакс-вращения может быть легко получена, некоторым упрощением представленной функции.

Листинг 6.5

```
short __stdcall Varimax (double r[],short k,short m)
//
// Функция производит варимакс-вращение матрицы факторного
// отображения.
// Обозначения:
//   r - матрица факторного отображения размером k * m,
//   k - размерность корреляционной матрицы, k > 2,
//   m - число факторов.
// Возвращаемое значение:
//   0 при нормальном окончании счета,
//  -1 при недостатке памяти для рабочих массивов.
//
{
 short i,j,j1; // Счетчики цикла

 short m1,m2,m3,m4;     // Счетчик одномерного массива

 double *x,           // Вращаемая пара векторов
        *y,           // Повернутая пара векторов
        *h,           // Суммарная нагрузка
        u,v,          // Вспомогательные переменные
        A,B,C,D,
        vc = 0,vc1,   // Величина для проверки окончания
                      // процедуры
        num,          // Числитель и знаменатель
        cs[4];        // Матрица вращения

 if ((h = new double[k]) == 0)
   return -1;
```

```
//////////////////////////////////
// Нормализация факторных нагрузок
for (i = 0; i < k; i++)
  {
   for (j = 0, h[i] = 0, m1 = i; j < m; j++, m1 += k)
     h[i] += r[m1] * r[m1];
   h[i] = sqrt (h[i]);
  }

for (i = 0,m1 = 0; i < m; i++)
  for (j = 0; j < k; j++)
    r[m1++] /= h[j];

if ((x = new double[k * 2]) == 0)
  {
   delete [] h;
   return -1;
  }
if ((y = new double[k * 2]) == 0)
  {
   delete [] x;
   return -1;
  }

do
  {
   vc1 = vc;

   ///////////////
   // Цикл вращения
   for (i = 0; i < m - 1; i++)
     {
      for (j = i + 1,m3 = j * k,m4 = m3; j < m; j++)
        {
         ///////////
         // Вектор x
         for (j1 = 0,m1 = i * k,m2 = m1; j1 < k; j1++)
           x[j1] = r[m1++];
```

```
///////////
// Вектор y
for (j1 = k; j1 < k + k; j1++)
  x[j1] = r[m3++];

////////////////////////////////////
// Вычисление вспомогательных величин
for (j1 = 0,A = 0,B = 0,C = 0,D = 0; j1 < k; j1++)
  {
   u = x[j1] * x[j1] - x[j1 + k] * x[j1 + k];
   v = 2 * x[j1] * x[j1 + k];
   A += u;
   B += v;
   C += u * u - v * v;
   D += u * v;
  }

////////////////////////////////////
// Тангенс учетверенного угла вращения
num = 2 * (D - A * B / k);
u = num / (C - (A * A - B * B) / k);

/////////////////////////////////////////
// Выбор угла из условия максимума критерия
if (u > 0 && num < 0)
  u = atan (u) - M_PI;
else if (u < 0 && num > 0)
  u = atan (u) + M_PI;
else
  u = atan (u);

u /= 4;

//////////////////////////
// Если вращение имеет смысл
if (fabs (u) >= 0.000001)
  {
   ////////////////////////////
```

```
          // Вычисление матрицы вращения
          cs[0] = cos (u);
          cs[1] = sin (u);
          cs[2] = -cs[1];
          cs[3] = cs[0];

          ////////////////////////
          // Вращение пары векторов
          MatrixMultiplyMatrix (x,cs,y,k,2,2);

          //////////////////////
          // Вектор x - на место
          for (j1 = 0; j1 < k; j1++)
            r[m2++] = y[j1];

          //////////////////////
          // Вектор y - на место
          for (j1 = k; j1 < k + k; j1++)
            r[m4++] = y[j1];
         }
      }
   }

 ////////////////////////////////////////////
 // Проверка сходимости по варимакс-критерию
 for (i = 0,vc = 0,m1 = 0; i < m; i++)
   {
    for (j = 0,u = 0,v = 0; j < k; j++,m1++)
      {
       A = r[m1] * r[m1];
       u += A * A;
       v += A;
      }
    vc += k * u - v * v;
   }

} while (fabs (vc1 - vc) > 0.000001);
```

```
 delete [] x; delete [] y;

 ////////////////////////////////////
 // Денормализация факторных нагрузок
 for (i = 0,m1 = 0; i < m; i++)
   for (j = 0; j < k; j++)
     r[m1++] *= h[j];

 delete [] h;

 return 0;
}
```

Критерии оценки числа удерживаемых факторов

Существует несколько критериев оценки максимального числа удерживаемых факторов. Критерии, основанные на анализе определителей (детерминантов) исходной и воспроизведенной корреляционной матриц, не показывают стабильности, хотя сама проблема вычисления детерминанта не представляет особой сложности.

Критерии, основанные на величине собственных значений корреляционной матрицы, приводят к анализу процента дисперсии, выделенной факторами. Все общие факторы, число которых равно числу параметров, выделяют 100% дисперсии. Если сумма процентов дисперсии превышает величину 100%, то это означает: при вычислении собственных значений корреляционной матрицы были получены отрицательные собственные значения и, как следствие, комплексные собственные векторы. Причиной данного катаклизма, скорее всего, является некорректная редукция исходной корреляционной матрицы, либо применение коэффициентов корреляции различных типов.

Таким образом, максимальное число удерживаемых факторов в методах главных компонент и главных факторов можно приблизительно установить из анализа процента дисперсии, выдаваемой программой для каждого фактора. Резюмируя сказанное выше, мы рекомендуем такой порядок действий: сначала пользователь проводит разведочный факторный анализ, без указания максимального числа факторов, затем по величине дисперсии оценивает число факторов, задает его и проводит повторный анализ, для окончательных результатов.

В программе метода максимума правдоподобия по результатам вычислений необходимо выдать оценки общностей, вычисленные из условия полной воспроизводимости корреляционной матрицы. Если задать число факторов

равным числу параметров, то оценки общности будут совпадать с общностями исходной корреляционной матрицы, то есть будут равны единице. Максимальное число удерживаемых факторов можно оценить из анализа процента дисперсии, выдаваемой программой для каждого фактора.

На листинге 6.6 представлена функция вычисления выделенного факторами процента дисперсии, адаптированная к вызову из программ на различных языках программирования, включая VB и VBA.

Листинг 6.6

```
void __stdcall PercentOfVariance (double z[],short k,double r[],double
s[],short ms)
//
// Функция вычисляет процент дисперсии, выделенной факторами.
// Обозначения:
//   z - корреляционная матрица,
//   k - размерность корреляционной матрицы,
//   r - матрица факторного отображения,
//   s - вектор процентов дисперсии,
//  ms - число факторов.
// Примечание:
//  функция применяется после успешного завершения факторного
//  анализа, поэтому ошибка типа нулевой суммарной дисперсии
//  не отслеживается.
// Возвращаемое значение:
//  нет.
//
{
 short i,j; // Счетчики

 short m;            // Счетчик одномерного массива

 double sd = 0;    // Суммарная дисперсия

 for (i = 0, m = 0; i < k; i++)
   {
    sd += z[m];
    m += k + 1;
   }

 for (i = 0, m = 0; i < ms; i++)
   {
```

```
    for (j = 0, s[i] = 0; j < k; j++, m++)
      s[i] += r[m] * r[m];

    s[i] /= sd / 100;
   }
}
```

Визуализация результатов факторного анализа и пример применения

Для наглядности, результаты факторного анализа можно изобразить графически. Обычно дают изображение многомерного факторного пространства в двумерных срезах. Соответствующая диаграмма может быть построена прямо из VBA, с помощью объектов Chart. В нашем случае для построения диаграмм можно ограничиться стандартными средствами Microsoft Excel.

Рассмотрим пример интерпретации результатов анализа. Пусть в эксперименте получены некоторые опытные данные, измерения четырех параметров. В результате проведенных расчетов было выделено два фактора (две главных компоненты), обозначенные буквами A и B.

Рис. 6.1. Изученные параметры 1, 2, 3 и 4 в пространстве главных компонент, которым соответствуют фактор A и фактор B

По результатам расчета модулем Faa программы анализа данных AtteStat (см. следующий раздел) построим диаграмму. На рис. 6.1 показана диаграмма, построенная стандартными средствами Microsoft Excel путем выбора из меню **Вставка ▸ Диаграмма ▸ Стандартные ▸ Точечная**. После выбора типа диаграммы следует нажать кнопку **Далее**. В качестве данных нужно стандартным образом выбрать вычисленную матрицу факторного отображения или повернутую матрицу факторного отображения, как на рис. 6.1. Менять другие параметры нет необходимости. Нажать кнопку **Далее**. При желании, можно сделать надписи в диаграмме.

Векторы данных распадаются на две группы: одну с параметрами 1 и 2, другую с параметрами 3 и 4. Таким образом, по результатам расчета можно выдвинуть гипотезы:

- параметр 1 и параметр 2 имеют взаимную линейную корреляцию;
- параметр 3 и параметр 4 также имеют сильную взаимную линейную корреляцию.

Практический вывод: в эксперименте достаточно измерить не четыре параметра, а только два: параметр 1 (или 2) и параметр 3 (или 4).

Эффект при вставке диаграммы: пункт меню AtteStat временно исчезает (как и несколько стандартных пунктов меню, не имеющих отношения к построению диаграммы), когда пользователь редактирует диаграмму (рис. 6.1), но появляется вновь, как только пользователь возвращается в рабочее поле листа.

Руководство пользователя

Для работы с модулем факторного анализа (Faa) выберите из меню Microsoft Excel добавленный программным обеспечением AtteStat новый пункт **AtteStat ▸ Модуль Faa ▸ Факторный анализ**. На экране появится диалоговое окно, изображенное на рис. 6.2.

Рис. 6.2. Окно модуля факторного анализа

1. Выберите или введите интервалы матрицы исходных данных.
2. Выберите или введите выходной интервал. Начиная с первой ячейки выходного интервала, будут выведены результаты вычислений.
3. Выберите или оставьте по умолчанию метод факторного анализа.
4. Выберите или оставьте по умолчанию метод оценки общностей.
5. Выберите или оставьте по умолчанию тип связи.
6. Укажите или оставьте по умолчанию, как расположены выборки.
7. Введите, по желанию, число факторов. Оно не должно превышать число параметров.
8. Нажмите кнопку **Выполнить расчет**.

Начиная с первой ячейки выходного интервала, будет выведено название метода и результаты расчета.

Модуль факторного анализа берет на себя верификацию исходных данных, выдавая подробную диагностику. Обратим внимание, что не все сочетания параметров допустимы. Если встречается недопустимое сочетание параметров, расчет может иметь отклонения, например, редукцию матрицы дисперсий ковариаций, модуль факторного анализа пропустит без всякой диагностики.

Сообщения об ошибках

При ошибках ввода данных могут выдаваться диагностические сообщения следующих типов.

Рис. 6.3. Окно сообщения об ошибке

На рис. 6.3 не выбран или неверно введен входной интервал. Лучшим способом избежать ошибки является не ввод, а выделение интервала протаскиванием курсора.

Рис. 6.4. Окно сообщения об ошибке

Проверьте исходные данные на рис. 6.4 и заполните все ячейки входного интервала. Если в ячейке не должно быть данных, по физической природе исследуемого процесса, введите нуль.

Рис. 6.5. Окно сообщения об ошибке

Проверьте типы ячеек входного интервала на рис.6.5. Тип может быть только числовым. Проще выделить интервал ячеек и явно определить их тип как числовой средствами Microsoft Excel.

Рис. 6.6. Окно сообщения об ошибке

На рис.6.6 не выбран или неверно введен выходной интервал. Лучшим способом избежать ошибки является не ввод, а выделение интервала стандартным образом.

В процессе факторного анализа нужно быть готовым к тому, что иногда решение получить не удается. Это вызвано сложностью решаемой проблемы собственных значений корреляционной матрицы. Для матриц высокого порядка может произойти потеря значимости в процессе вычислений.

Теоретически нельзя исключить ситуацию, когда методы факторного анализа неприменимы, без дополнительного исправления. Выявите линейно зависимые параметры с помощью, например, метода корреляционного анализа из пакета анализа Microsoft Excel, и оставьте в исходных данных только один из группы линейно зависимых параметров.

Корреляционная матрица может оказаться вырожденной. Применяемый для определения собственных значений метод дает решение и в этом случае. При этом некоторые собственные значения будут нулевыми, что делает метод главных факторов более устойчивым, чем метод максимума правдоподобия. Однако метод главных факторов не позволяет получить точной оценки общности и добиться, таким образом, полной воспроизводимости корреляционной матрицы с общностями на главной диагонали.

Дальнейшие исследования

Дальнейшее развитие модуля факторного анализа в двух направлениях:

1. увеличение функциональных возможностей за счет расширения спектра используемых алгоритмов;
2. улучшение возможностей по наглядному представлению результатов расчета.

На первом из направлений насущным видится дополнение вычислительного модуля Faa возможностями анализа признаков в различных шкалах.

Наглядное представление дает, прежде всего, использование графических возможностей Microsoft Excel, автоматический вывод, всех двумерных срезов пространства факторов. Для этого могут быть применены методы, рассмотренные в *главе 11*. Количество s таких двумерных срезов, равное количеству выводимых графиков, пространства $r > 1$ выделенных факторов легко подсчитать по формуле:

$$s = \sum_{i=1}^{r-1} i .$$

Рекомендуемая литература

1. Биллиг В.А. Мир объектов Excel 2000. — М.: Издательско-торговый дом «Русская редакция», 2001.
2. Брандт З. Анализ данных. Статистические и вычислительные методы для научных работников и инженеров. — М.: Мир, ООО «Издательство АСТ», 2003.
3. Гайдышев И. Анализ и обработка данных: специальный справочник. — СПб: Питер, 2001.
4. Гантмахер Ф.Р. Теория матриц. — М.: Наука, 1988.
5. Голуб Дж., Ван Лоун Ч. Матричные вычисления. — М.: Мир, 1999.
6. Деммель Дж. Вычислительная линейная алгебра. Теория и приложения. — М.: Мир, 2001.
7. Додж М., Кината К., Стинсон К. Эффективная работа с Excel 7.0 для Windows 95. — СПб: Питер, 1996.
8. Дюк В. Обработка данных на ПК в примерах. — СПб: Питер, 1997.
9. Лоули Д., Максвелл А. Факторный анализ как статистический метод. — М.: Мир, 1967.
10. Окунь Я. Факторный анализ. — М.: Статистика, 1974.

11. Петренко В., Сергеева М. Фактор рекламы в представлении курильщиком рынка табачных изделий // Лаборатория рекламы. — 2003. — №1. — С.30–33.

12. Сборник научных программ на Фортране. Вып. 1. Статистика. — М.: 1974.

13. Статистические методы для ЭВМ / Под ред. К. Энслейна, Э. Рэлстона, Г.С. Уилфа. — М.: Наука, 1986.

14. Уилкинсон Дж. Х., Алгебраическая проблема собственных значений. — М.: Наука, 1970.

15. Уилкинсон, Райнш. Справочник алгоритмов на языке АЛГОЛ. Линейная алгебра. — М.: Машиностроение, 1976.

16. Уткин В.А. Статистические технологии в медицинских исследованиях: Монография. — Пятигорск: ГНИИК, 2002.

17. Уткин В.А., Гайдышев И.П., Кобазева О.М. О возможном единообразии приложений и условий реализации факторного анализа // Наука и образование Зауралья. — 2001. — №1. — С.3338.

18. Факторный, дискриминантный и кластерный анализ / Дж.О. Ким, Ч.У. Мюллер, У.Р. Клекка и др. — М.: Финансы и статистика, 1989.

19. Харман Г. Современный факторный анализ. — М.: Статистика, 1972.

ГЛАВА 7

Распознавание образов на VBA и C

«Не в совокупности ищи единства, но более — в единении разделения».
К. Прутков

Для работы с материалами главы понадобятся лицензионные средства разработки и тестирования программ:

- Microsoft Excel 97 или выше;
- Microsoft Visual C++ 5.0 или выше;
- Borland C++ 5.5 или выше.

Потребуется знание VBA, C.

Решаемая проблема

Распознавание образов — это научное направление, связанное с разработкой теоретической основы и практической реализацией методов, предназначенных для распознавания объектов, явлений, процессов, установления диагнозов. Распознавание образов включает в себя два больших раздела — распознавание без обучения (без учителя) и распознавание с обучением (с учителем). Из рассмотренных нами методов к числу первых относится кластерный анализ, к числу вторых — дискриминантный анализ и множественный регрессионный анализ. Мы постараемся отобразить основные алгоритмы.

Из новых методов распознавания применяются нейронные сети различных архитектур. Данные методы нами здесь не рассматриваются, а краткое введение в нейронные сети изложено в нашей первой книге. Там же даны тексты программ и полезные ссылки.

Распознавание образов без обучения

Классическими методами распознавания образов без обучения являются методы кластерного анализа (кластер-анализа). Методами кластерного анализа решается задача разбиения (классификации, кластеризации, группировки) множества объектов таким образом, чтобы все объекты, принадлежащие одному кластеру (классу, группе) были более похожи друг на друга, чем на объекты других кластеров.

При использовании результатов кластерного анализа в других методах, например распознавании образов с обучением, порядок следования кластеров может быть любым удобным для исследователя.

Классифицируемыми могут быть как параметры, так и объекты, поэтому по ходу изложения там, где идет речь о классификации объектов, можно говорить о классификации параметров, и наоборот.

Особенности решения

Регистрация функций DLL выполнена в модуле VBA. Программное решение, в основном, повторяет изложенное в *главе 6*, так как результатом работы программы кластерного анализа являются массивы чисел. Их нельзя без дополнительных усилий (немногие даже опытные пользователи знают, как принять в Microsoft Excel массив, возвращаемый VBA–функцией) передать с помощью макросов, поэтому регистрация VBA–оператором `Declare` была удобным решением. Основное отличие в более компактной форме передачи управляющих параметров, обусловленных разнообразием вариантов решения.

Научные алгоритмы

Листинги некоторых вызываемых функций не показаны, т. к. опубликованы в нашей первой книге, однако есть одно существенное отличие. Построение матрицы мер, по сравнению с упомянутым источником, оптимизировано, благодаря чему функция построения матрицы мер стала в 2 раза более компактной (нужно было сделать это раньше). Исходные тексты данных функций приводятся на прилагаемом компакт-диске.

Меры различия

Меры для кластерного анализа могут быть следующих видов:

- мера типа расстояния (функции расстояния), называемая также мерой различия. В этом случае объекты считаются тем более похожими, чем меньше расстояние между ними, поэтому некоторые авторы называют меры сходства типа расстояния мерами различия;
- мера типа корреляции, называемая связью, определяющая похожесть объектов (мера сходства). В этом случае объекты считаются тем более похожими, чем больше связь между ними. Меры посредством элементарных преобразований могут быть сведены к мерам различия, что и сделано в данном программном обеспечении, с целью единообразия.

Нами применяются меры различия, в зависимости от принадлежности той или иной шкале измерения описывающих объекты параметров:

для количественных признаков:

- евклидово расстояние;
- манхеттенское расстояние;
- супремум-норма;
- расстояние Махаланобиса;
- расстояние Пирсона;

для порядковых или количественных признаков:

- расстояние Спирмэна;
- расстояние Кендалла;

для номинальных признаков:

- расстояние Жаккара;
- расстояние Рассела–Рао;
- расстояние Бравайса;
- расстояние Юла.

Рассмотренными мерами могут оперировать различные методы кластерного анализа.

На листинге 7.1 показана функция построения матрицы мер для кластерного анализа. Данная функция строит матрицу мер для любой из мер, по выбору пользователя.

Листинг 7.1

```
short LikenessMatrix (double data[],double z[],double s[],short n,short
k,short Mode)
//
// Функция строит матрицу мер для кластерного анализа.
// Обозначения:
//  data — массив исходных данных об объектах (k векторов друг за дру-
гом),
//  z    — вычисленный массив мер сходства размером n x n,
//  s    — заранее вычисленная обратная дисперсионно-корреляционная
//         матрица (только для вычисления расстояния Махалонобиса),
//  n    — число классифицируемых объектов,
//  k    — число параметров, описывающих каждый объект,
//  Mode — вариант меры сходства:
//       =  0 — евклидово расстояние,
//       =  1 — манхеттенское расстояние,
//       =  2 — расстояние Махаланобиса,
//       =  3 — сюпремум-норма,
//       =  4 — расстояние Пирсона,
//       =  5 — расстояние Спирмэна,
//       =  6 — расстояние Кендалла,
//       =  7 — расстояние Жаккара,
//       =  8 — расстояние Рассела-Рао,
//       =  9 — расстояние Бравайса,
//       = 10 — расстояние Юла.
// Возвращаемое значение:
//   0 — при нормальном окончании расчета,
//  -1 — при недостатке памяти.
//
{
 short i,j,l,m,m1,m2; // Счетчики

 double *dat1,*dat2,  // Копии векторов данных
        temp;         // Вспомогательная величина

 if (!(dat1 = new double[k]))
   return -1;
 if (!(dat2 = new double[k]))
   {
```

```
    delete [] dat1;
    return -1;
   }

////////////////////////////
// Построение матрицы сходства
for (i = 0, m2 = 0; i < n; i++)
  for (j = 0; j < n; j++)
    {
     ////////////////////////////////////////////////////
     // Копия делается в текущем цикле для восстановления
     // после разрушения dat1 функцией Mahalanobis
     for (l = 0, m = i; l < k; l++, m += n)
       dat1[l] = data[m];

     for (l = 0, m1 = j; l < k; l++, m1 += n)
       dat2[l] = data[m1];

     switch (Mode)
       {
        case 0:
          z[m2++] = Euclid (dat1,dat2,k);
          break;
        case 1:
          z[m2++] = CityBlock (dat1,dat2,k);
          break;
        case 2:
          z[m2++] = Mahalanobis (dat1,dat2,s,k);
          break;
        case 3 :
          z[m2++] = Sup (dat1,dat2,k);
          break;
        case 4:
          z[m2++] = 1 — Pearson (dat1,dat2,n);
          break;
        case 5:
          if (Spearman (dat1,dat2,n,&temp) == -1)
            {
             delete [] dat1; delete [] dat2;
```

```
            return -1;
           }
          z[m2++] = 1 — temp;
          break;
        case 6:
           if (Kendall (dat1,dat2,n,&temp) == -1)
             {
              delete [] dat1; delete [] dat2;
              return -1;
             }
          z[m2++] = 1 — temp;
          break;
        case 7:
          z[m2++] = 1 — Jaccard (dat1,dat2,k);
          break;
        case 8:
          z[m2++] = 1 — Rao (dat1,dat2,k);
          break;
        case 9:
          z[m2++] = 1 — Bravais (dat1,dat2,k);
          break;
        default:
          z[m2++] = 1 — Yule (dat1,dat2,k);
          break;
       }
    }

 delete [] dat1; delete [] dat2;

 return 0;
}
```

Метрика Минковского

Наиболее общей мерой является метрика Минковского

$$d_{ij} = \sqrt[r]{\sum_{k=1}^{n} \left| x_{ik} - x_{jk} \right|^{r}} .$$

При $r = 1$ метрика Минковского дает манхеттенское расстояние (метрику города, city block, Manhattan distance)

$$d_{ij} = \sum_{k=1}^{n} \left| x_{ik} - x_{jk} \right| .$$

Если в метрике Минковского положить $r = 2$, получим стандартное евклидово расстояние (евклидову метрику)

$$d_{ij} = \sqrt{\sum_{k=1}^{n} \left(x_{ik} - x_{jk} \right)^2} .$$

При $r \to \infty$ метрика Минковского дает метрику доминирования

$$d_{ij} = \max_k \left| x_{ij} - x_{jk} \right|, k = 1,2,...,n,$$

что совпадает с супремум-нормой (∞ -нормой, сюпремум-нормой)

$$d_{ij} = \sup \left\{ \left| x_{ik} - x_{jk} \right| \right\}, k = 1,2,\ldots,n.$$

Расстояние Махалонобиса

При анализе двух совокупностей для различных ковариационных матриц, в случае произвольного распределения совокупностей, дивергенция, оценивающая расхождение между распределениями, выражается формулой

$$J_{ij} = \int_x \left[p_i(x) - p_j(x) \right] \ln \frac{p_i(x)}{p_j(x)} dx .$$

Иначе дивергенция называется полной средней информационной мерой различия двух классов или, более коротко, средней различающей информацией.

Дивергенция может быть вычислена по формуле

$$J_{ij} = \frac{1}{2} tr\left[\left(C_i - C_j \right) \left(C_j^{-1} - C_i^{-1} \right) \right] + \frac{1}{2} tr\left[\left(C_i^{-1} + C_j^{-1} \right) \left(m_i - m_j \right) \left(m_i - m_j \right)' \right],$$

где C_i, C_j — дисперсионно-ковариационные матрицы совокупностей i и j,

m_i, m_j — вектора средних совокупностей i и j,

штрих означает операцию транспонирования.

Иначе дивергенция называется информационной мерой Кульбака. В практических исследованиях мера Кульбака применяется для изучения расхождения между двумя эмпирическими распределениями.

Мера Махаланобиса (расстояние Махалонобиса, обобщенное Евклидово расстояние) является дивергенцией, в предположении, что ковариационные матрицы классов равны

$$C_i = C_j = \Sigma,$$

где Σ — общая внутригрупповая дисперсионно–ковариационная матрица,

а совокупности подчиняются многомерному нормальному распределению. Последние требования накладывают определенные ограничения на применение рассматриваемой меры различия, что часто не принимается во внимание исследователями. Проверка многомерной нормальности произвольного многомерного распределения — пока нерешенная задача (задача решена для двумерного распределения), поэтому единственное, что остается исследователю — предположить, что распределение опытных данных соответствует многомерному нормальному распределению. Такое предположение — весьма слабое основание для применения метода.

После должных преобразований получим, что практически мера Махаланобиса вычисляется как

$$d_{ij} = \left(X_i - X_j\right)' \Sigma^{-1} \left(X_i - X_j\right),$$

где $X_{\cdot}$ — вектора сравниваемых совокупностей,

Σ — общая внутригрупповая дисперсионно-ковариационная матрица.

Расстояние Пирсона

Коэффициент корреляционного отношения Пирсона (коэффициент корреляции, выборочный коэффициент корреляции, коэффициент корреляции Бравайса–Пирсона) измеряет силу линейной корреляционной связи количественных признаков. Коэффициент корреляции вычисляется по известной формуле

$$r = \frac{\sum_{i=1}^{n}\left(x_i - \bar{x}\right)\left(y_i - \bar{y}\right)}{\sqrt{\sum_{i=1}^{n}\left(x_i - \bar{x}\right)^2}\sqrt{\sum_{i=1}^{n}\left(y_i - \bar{y}\right)^2}},$$

где n — численность каждой выборки,

$x_i, i = 1,2,...,n$ — первая выборка, $\bar{x}$ – соответствующее среднее значение,

$y_i, i = 1,2,...,n$ — вторая выборка, $\bar{y}$ – соответствующее среднее значение.

Использование коэффициента корреляции в качестве меры связи оправдано лишь тогда, когда совместное распределение пары признаков нормально или приблизительно нормально.

Расстояние Пирсона как мера сходства, может быть получено из рассмотренного коэффициента путем вычитания последнего из единицы.

Расстояние Спирмэна

Показатель ранговой корреляции Спирмэна (показатель корреляции рангов Спирмэна, коэффициент корреляции рангов, коэффициент корреляции Спирмэна, коэффициент ранговой корреляции ρ, Spearman rank correlation) применяется в случае линейной связи между рядами, представленными в количественной или порядковой шкале. При анализе количественных признаков применять показатель Спирмэна вместо коэффициента корреляционного отношения Пирсона не следует, так как при его вычислении происходит понижение количественной шкалы до порядковой. Широкое применение показатель Спирмэна нашел при анализе корреляции порядковых признаков. Расчет ведется по формуле

$$\rho_S = 1 - \frac{6(S_\rho + B_x + B_y)}{n^3 - n}; S_\rho = \sum_{i=1}^{n} (r_i - s_i)^2 ,$$

где $r_i, s_i, i = 1,2,...,n$ — массивы рангов,

n — число пар вариант исследуемых рядов,

B_x, B_y — поправки на объединение рангов в соответствующих рядах, вычисляемые по формуле

$$B_{\cdot} = \frac{1}{12} \sum_{i=1}^{m} n_i (n_i^2 - 1) ,$$

где m — число групп объединенных рангов в ряду,

$n_i, i = 1,2,...,m$ — число рангов в i-ой группе.

Расстояние Спирмэна, как мера сходства, может быть получено из рассмотренного коэффициента путем вычитания последнего из единицы.

Расстояние Кендалла

Коэффициент ранговой корреляции Кендалла (коэффициент корреляции рангов, ранговый коэффициент корреляции, коэффициент корреляции Кендэла, коэффициент τ Кендалла, Kendall rank correlation) предназначен для вычисления силы корреляционной связи между двумя рядами. Коэф-

фициент применяется при тех же условиях, что и рассмотренный ранее показатель Спирмэна, но считается более строгой оценкой.

Все основные положения и замечания, данные при описании показателя Спирмэна, справедливы и в отношении коэффициента Кендалла. Расчет ведется по формуле:

$$\tau = \frac{S_\tau}{\sqrt{\left(\frac{n(n-1)}{2} - B_x\right)\left(\frac{n(n-1)}{2} - B_y\right)}}, S_\tau = \sum_{i=1}^{n}\sum_{j=i+1}^{n} sign(r_j - s_i),$$

где $r_i, s_i, i = 1,2,...,n$ — массивы рангов анализируемых рядов,

n — число пар вариант исследуемых рядов,

B_x, B_y — поправки на объединение рангов в соответствующих рядах, вычисляемые по формуле

$$B_{\cdot} = \frac{1}{2}\sum_{i=1}^{m} n_i(n_i - 1),$$

где m — число групп объединенных рангов в ряду,

$n_i, i = 1,2,...,m$ — число рангов в i-ой группе.

Расстояние Кендалла как мера сходства может быть получено из рассмотренного коэффициента путем вычитания последнего из единицы.

Расстояние Жаккара

Показатель подобия Жаккара (коэффициент Жаккара) для таблиц сопряженности 2 × 2 вычисляется по формуле

$$t_J = \frac{a}{a+b+c},$$

где a — число наблюдений с эффектом А в первой выборке,

b — число наблюдений без эффекта А в первой выборке,

c — число наблюдений с эффектом А во второй выборке.

Расстояние Жаккара как мера сходства может быть получено из рассмотренного коэффициента путем вычитания последнего из единицы.

Расстояние Рассела–Рао

Показатель подобия Рассела и Рао для таблиц сопряженности 2 × 2 вычисляется по формуле

$$t_R = \frac{a}{a+b+c+d},$$

где a — число наблюдений с эффектом А в первой выборке,

b — число наблюдений без эффекта А в первой выборке,

c — число наблюдений с эффектом А во второй выборке,

d — число наблюдений без эффекта А во второй выборке.

Расстояние Рассела–Рао как мера сходства может быть получено из рассмотренного коэффициента путем вычитания последнего из единицы.

Подробно показатель подобия Рассела-Рао рассмотрен в *главе 8*.

Расстояние Бравайса

Специальная форма коэффициента корреляции для таблиц сопряженности 2×2 — коэффициент сопряженности Бравайса (φ-коэффициент Пирсона, коэффициент контингенции Пирсона) — рассчитывается по формуле

$$\varphi = \frac{ad - bc}{\sqrt{(a+b)(a+c)(d+b)(d+c))}},$$

где a — число наблюдений с эффектом А в первой выборке,

b — число наблюдений без эффекта А в первой выборке,

c — число наблюдений с эффектом А во второй выборке,

d — число наблюдений без эффекта А во второй выборке.

Расстояние Бравайса как мера сходства может быть получено из рассмотренного коэффициента путем вычитания последнего из единицы.

Расстояние Юла

Ориентировочную оценку корреляционной связи в случае исследования таблиц сопряженности 2×2 может дать коэффициент ассоциации Юла. Вычисления производятся по формуле

$$t_Y = \frac{ad - bc}{ad + bc},$$

где a — число наблюдений с эффектом А в первой выборке,

b — число наблюдений без эффекта А в первой выборке,

c — число наблюдений с эффектом А во второй выборке,

d — число наблюдений без эффекта А во второй выборке.

Расстояние Юла как мера сходства может быть получено из рассмотренного коэффициента путем вычитания последнего из единицы.

Методы кластерного анализа

В рассматриваемом модуле реализованы следующие методы кластерного анализа:

- иерархические методы группировки:
  - метод ближней связи;
  - метод средней связи Кинга;
  - метод Уорда.
- итеративные методы группировки:
  - метод k-средних Мак-Куина.

Метод ближней связи

Метод является самым простым из иерархических агломеративных методов кластерного анализа. Метод начинает процесс классификации с поиска и объединения двух наиболее похожих объектов в матрице сходства. На следующем этапе находятся два очередных наиболее похожих объекта, и процедура повторяется до полного исчерпания матрицы сходства.

В процессе кластеризации методом ближней связи явно прослеживается образование цепочек объектов. Таким образом, для выделения кластеров после окончания процесса кластеризации требуется задаться некоторым пороговым уровнем сходства, на котором выделяется число кластеров, больше одного. Процедура не всегда ведет к образованию одного большого кластера на последнем этапе кластеризации и часто заканчивается явным разделением всех предъявленных объектов на кластеры.

После проведения классификации рекомендуется визуализировать результаты кластеризации путем построения дендрограммы (дендограммы).

Листинг 7.2

```
short __stdcall NearLink (double data[],short n,short k,short Mode,short
ph[],short pl[],double s[])
//
// Функция производит кластерный анализ методом ближней связи.
// Обозначения:
//   data — матрица исходных данных размером n * k
//          (k векторов длиной n друг за другом),
```

```
//      n — число объектов, подлежащих классификации,
//      k — число параметров,
//   Mode — вариант меры сходства,
//  ph,pl — на выходе — массивы номеров каждый длиной (n — 1),
//          содержащие информацию о группировке объектов,
//      s — уровень объединения пары [ph,pl].
// Возвращаемое значение:
//   0 при нормальном окончании расчета,
//  -1 при недостатке памяти.
//
{
 short i,j,       // Счетчик цикла
       k1 = 0,    // Счетчик пар номеров
       Code,      // Код ошибки
       m;         // Счетчик одномерного массива

 double buf,buf1, // Рабочая переменная
        *t,       // Матрица, обратная дисперсионно-ковариационной
        *z;       // Матрица сходства

 if (!(z = new double[MaximumOf (k,n) * MaximumOf (k,n)]))
   return -1;
 if (!(t = new double[k * k]))
   {
    delete [] z;
    return -1;
   }

 ////////////////////////////////////////////////
 // Построение обратной дисперсионно-ковариационной
 // матрицы для вычисления расстояния Махаланобиса
 if (Mode == 2)
   {
    ////////////////////////////////////////////////
    // Построение дисперсионно-ковариационной матрицы
    Code = CovMatrix (data,z,n,k);

    if (Code != 0)
```

```
      {
       delete [] t; delete [] z;
       return Code;
      }

   ////////////////////////////////////////////////
   // Матрица, обратная дисперсионно-ковариационной
   Code = InverseMatrix (z,t,k);

   if (Code != 0)
      {
       delete [] t; delete [] z;
       return Code;
      }
  }

if ((Code = LikenessMatrix (data,z,t,n,k,Mode)) != 0)
  {
   delete [] t; delete [] z;
   return Code;
  }

delete [] t;

//////////////////
// Сдвиг максимума
buf1 = MaximumOf (z,n * n) + 1;

////////////////////////////////////
// Фактическое исключение диагонали
for (i = 0, m = 0; i < n; i++, m += n + 1)
  z[m] = buf1;

do
  {
   buf = MinimumOf (z,n * n);

   for (i = 0, m = 0; i < n; i++)
```

```
      for (j = 0; j < n; j++, m++)
        if (z[m] == buf)
          {
           //////////////////////
           // Запомнить соединение
           ph[k1] = i;
           pl[k1] = j;
           s[k1++] = buf;

           ///////////////
           // Убрать строку
           FillUpCol (z,n,i,buf1);

           ////////////////////////////
           // Убрать симметричный элемент
           z[j * n + i] = buf1;
          }

   } while (k1 < n);

 delete [] z;

 return 0;
}
```

Метод средней связи Кинга

Метод средней связи подобен методу ближней связи. Разница в том, что на каком-либо этапе ранее объединенные в один кластер объекты считаются одним объектом с усредненными по кластеру параметрами.

По поводу алгоритма показанной далее функции поясним, что на каждом $(i + 1)$-м шаге кластер, объединивший объекты (кластеры) на шаге i, имеет номер меньшего из сравниваемых номеров объектов, остальные объекты перенумеровываются, и количество объектов уменьшается, таким образом, на единицу.

Листинг 7.3

```
short __stdcall King (double data[],short n,short k,short Mode,short
ph[],short pl[],double s[])
```

```
//
// Функция производит кластерный анализ методом средней связи Кинга.
// Обозначения:
//   data — матрица исходных данных размером n * k
//          (k векторов длиной n друг за другом),
//      n — число объектов, подлежащих классификации,
//      k — число параметров,
//   Mode — вариант меры сходства:
//        =  0 — евклидово расстояние,
//        =  1 — манхеттенское расстояние,
//        =  2 — расстояние Махаланобиса,
//        =  3 — сюпремум-норма,
//        =  4 — расстояние Пирсона,
//        =  5 — расстояние Спирмэна,
//        =  6 — расстояние Кендалла,
//        =  7 — расстояние Жаккара,
//        =  8 — расстояние Рассела-Рао,
//        =  9 — расстояние Бравайса,
//        = 10 — расстояние Юла,
//  ph,pl — на выходе — массивы номеров каждый длиной (n — 1),
//          содержащие информацию о группировке объектов,
//      s — уровень объединения пары [ph,pl].
// Возвращаемое значение:
//   0 при нормальном окончании расчета,
//  -1 при недостатке памяти,
//  -2 при потере значимости.
//
{
 short i,j,l,    // Счетчик
       m,m1,     // Счетчик одномерного массива
       r,        // Число объектов на текущем шаге
       Code = 0; // Возвращаемый код ошибки

 double *z,*dat, // Рабочая матрица
        *t;      // Матрица, обратная дисперсионно-ковариационной

 if (!(t = new double[k * k]))
   return -1;
```

```
//////////////////////////////////////////////////
// Построение обратной дисперсионно-ковариационной
// матрицы для вычисления расстояния Махаланобиса
if (Mode == 2)
  {
   ////////////////////////////////////////////
   // Здесь z временно используется для хранения
   // дисперсионно-ковариационной матрицы
   if (!(z = new double[k * k]))
     {
      delete [] t;
      return -1;
     }

   ////////////////////////////////////////////////
   // Построение дисперсионно-ковариационной матрицы
   Code = CovMatrix (data,z,n,k);

   if (Code != 0)
     {
      delete [] t; delete [] z;
      return Code;
     }

   ////////////////////////////////////////////////
   // Матрица, обратная дисперсионно-ковариационной
   Code = InverseMatrix (z,t,k);

   //////////////////////////////////////
   // Теперь память для массива z свободна
   delete [] z;

   if (Code != 0)
     {
      delete [] t;
      return Code;
     }
```

```
  }
if (!(dat = new double[n * k]))
  {
   delete [] t;
   return -1;
  }

//////////////////////////////////
// Копия исходного массива данных
ArrayToArray  (data,dat,n * k);

////////////////
// Основной цикл
for (l = 0, r = n; l < n — 1; l++, r--)
  {
   if (!(z = new double[r * r]))
     {
      delete [] t; delete [] dat;
      return -1;
     }

   ////////////////////////////////
   // Построение матрицы сходства
   if ((Code = LikenessMatrix (dat,z,t,r,k,Mode)) != 0)
     {
      delete [] t; delete [] z; delete [] dat;
      return Code;
     }

   ////////////////////////////////////////////////
   // Поиск двух наиболее похожих объектов путем
   // анализа нижнего треугольника матрицы сходства
   for (i = 0; i < r; i++)
     for (j = i + 1, m = i * r + i + 1; j < r; j++, m++)
       if (m == 1 || z[m] < s[l])
         {
          ph[l] = i;   // Номер первого элемента пары объектов
```

```
        pl[l] = j;   // Номер второго элемента пары объектов
        s[l] = z[m]; // Расстояние между объектами кластера
       }

////////////////////////////////////////////////////
// Теперь массив z используется для хранения массива
// данных, полученного на предыдущем шаге
delete [] z;
if (!(z = new double[r * k]))
  {
   delete [] t; delete [] dat;
   return -1;
  }

//////////////////////////////////
// Копия исходного массива данных
ArrayToArray (dat,z,r * k);

////////////////////////////////////////
// Уменьшенный на 1 строку массив данных
delete [] dat;
dat = new double[(r - 1) * k];

//////////////////////////////////////////////
// Центр тяжести нового кластера на месте ph[l]
for (j = 0, m1 = ph[l], m = pl[l]; j < k;
     j++, m1 += r, m += r)
  z[m1] = (z[m1] + z[m]) / 2;

////////////////////////////////////////////
// Сдвинуть строки на место pl[l] и далее
for (i = pl[l]; i < r - 1; i++)
  for (j = 0, m = i; j < k; j++, m += r)
    z[m] = z[m + 1];

//////////////////////////////
// Копия уменьшенного массива
for (i = 0; i < r - 1; i++)
```

```
      for (j = 0, m = i, m1 = i; j < k;
           j++, m += r - 1, m1 += r)
        dat[m] = z[m1];

    ////////////////////////////////////////////////
    // Теперь массив z свободен и будет использован для
    // хранения матрицы сходства
    delete [] z;
   }

 delete [] t; delete [] dat;

 return Code;
}
```

Метод Уорда

Данный метод напоминает метод средней связи Кинга. Особенность состоит в том, что основанием для помещения объекта в кластер является не близость двух объектов в каком-либо смысле, в зависимости от меры сходства, а минимум дисперсии внутри кластера при помещении в него текущего классифицируемого объекта.

Листинг 7.4

```
short __stdcall Ward (double data[],short n,short k,short ph[],short
pl[],double s[])
//
// Функция производит кластерный анализ методом Уорда.
// Обозначения:
//   data — матрица исходных данных размером n * k
//          (k векторов длиной n друг за другом),
//      n — число объектов, подлежащих классификации,
//      k — число параметров,
//  ph,pl — на выходе — массивы номеров каждый длиной (n — 1),
//          содержащие информацию о группировке объектов,
//      s — уровень объединения пары [ph,pl].
// Возвращаемое значение:
//   0 при нормальном окончании расчета,
//  -1 при недостатке памяти,
```

```
//  -2 при потере значимости.
//
{
 short i,j,l1,        // Счетчик цикла
       l,             // Счетчик кластеров
       m,m1,m2,       // Счетчик одномерного массива
       r,             // Число объектов на текущем шаге
       Code = 0,      // Код ошибки
       *inf;          // Количество элементов в кластерах

 double *z,*dat,      // Рабочая матрица
        *dat1,*dat2; // Копии векторов данных

 if (!(dat = new double[n * k]))
   return -1;
 if (!(inf = new short[n]))
   {
    delete [] dat;
    return -1;
   }
 if (!(dat1 = new double[n]))
   {
    delete [] dat; delete [] inf;
    return -1;
   }
 if (!(dat2 = new double[n]))
   {
    delete [] dat1; delete [] inf; delete [] dat1;
    return -1;
   }

 ////////////////////////////////////////////
 // На первом шаге — по 1 элементу в кластере
 FillUp (inf,n,1);

 ////////////////////////////////
 // Копия исходного массива данных
 ArrayToArray (data,dat,n * k);
```

```
////////////////
// Основной цикл
for (l = 0, r = n; l < n — 1; l++, r--)
  {
   if (!(z = new double[r * r]))
     {
      delete [] dat; delete [] inf; delete [] dat1;
      delete [] dat2;
      return -1;
     }

   ////////////////////////////////////////////
   // Построение внутригрупповых сумм квадратов
   for (i = 0, m2 = 0; i < r; i++)
     {
      for (l1 = 0, m = i; l1 < k; l1++, m += r)
        dat1[l1] = dat[m];
      for (j = 0; j < r; j++)
        {
         for (l1 = 0, m1 = j; l1 < k; l1++, m1 += r)
           dat2[l1] = dat[m1];
         z[m2++] = Euclid (dat1,dat2,k) *
                   inf[i] * inf[j] / (inf[i] + inf[j]);
        }
     }

   ////////////////////////////////////////////
   // Поиск двух наиболее похожих объектов путем
   // анализа нижнего треугольника матрицы сходства
   for (i = 0; i < r; i++)
     for (j = i + 1, m = i * r + i + 1; j < r; j++, m++)
       if (m == 1 || z[m] < s[l])
         {
          ph[l] = i;   // Номер первого элемента пары объектов
          pl[l] = j;   // Номер второго элемента пары объектов
          s[l] = z[m]; // Расстояние между объектами кластера
         }
```

```
//////////////////////////////////////////////////
// Теперь массив z используется для хранения массива
// данных, полученного на предыдущем шаге
delete [] z;
if (!(z = new double[r * k]))
  {
   delete [] dat; delete [] inf; delete [] dat1;
   delete [] dat2;
   return -1;
  }

////////////////////////////////
// Копия исходного массива данных
ArrayToArray (dat,z,r * k);

//////////////////////////////////////
// Уменьшенный на 1 строку массив данных
delete [] dat;
dat = new double[(r — 1) * k];

/////////////////////////////////////////
// Центр тяжести нового кластера на месте ph[l]
for (j = 0, m1 = ph[l], m = pl[l]; j < k;
     j++, m1 += r, m += r)
  z[m1] = (z[m1] + z[m]) / 2;

//////////////////////////////////
// Добавился элемент в кластер ph[l]
inf[ph[l]] += 1;

//////////////////////////////////////
// Сдвинуть строки на место pl[l] и далее
for (i = pl[l]; i < r — 1; i++)
  {
   inf[i] = inf[i + 1];
   for (j = 0, m = i; j < k; j++, m += r)
     z[m] = z[m + 1];
```

```
      }
    ////////////////////////////
    // Копия уменьшенного массива
    for (i = 0; i < r - 1; i++)
      for (j = 0, m = i, m1 = i; j < k;
           j++, m += r - 1, m1 += r)
        dat[m] = z[m1];

    ////////////////////////////////////////////////
    // Теперь массив z свободен и будет использован для
    // хранения матрицы сходства
    delete [] z;
   }

 delete [] dat; delete [] inf; delete [] dat1; delete [] dat2;

 return Code;
}
```

Метод k-средних Мак-Куина

Теоретическое обоснование метода k-средних (k-внутригрупповых средних) сравнительно просто, логично и может быть найдено во многих источниках. Принцип классификации сводится к некоторому, возможно, случайному, исходному разбиению множества объектов на заданное число кластеров (классов, групп, популяций), последующее отнесение остальных объектов к ближайшим кластерам, пересчету новых «центров тяжести» кластеров и продолжении описанной процедуры, пока не будет получено некоторое оптимальное разбиение. Кластеризация обычно продолжается, пока новые «центры тяжести» кластеров не перестанут отличаться от старых «центров тяжести». Особенностью метода является то, что выделенные в результате расчетов кластеры не будут пересекаться — гарантируется, что каждый классифицированный объект будет отнесен только к одному кластеру.

Метод имеет одну интересную особенность. Если исходные данные имеют такую структуру, что может быть выделено, к примеру, 2 кластера, а пользователь задал, что классификация объектов должна производиться на 4 кластера, то в результате вычислений окажется, что получено два «настоящих» кластера и два кластера с нулевым количеством объектов и нулевыми центрами тяжести.

В визуализации результатов кластеризации методом k-средних, по нашему мнению, нет необходимости, хотя может оказаться наглядным и красивым изображение пространственных эллипсоидов (только для пространств размерностью не более 3, для большей же размерности используются двумерные срезы пространства), содержащих классифицированные объекты.

Листинг 7.5

```
short __stdcall McQueen (double data[],short n,short k,short m,short
Mode,short inf[],double c[])
//
// Функция производит кластерный анализ методом k средних Мак-Куина.
// Обозначения:
//  data — матрица исходных данных размером n * k (k n-мерных векторов
//         друг за другом),
//     n — число объектов, подлежащих классификации,
//     k — число параметров,
//     m — заданное число кластеров,
//  Mode — вариант меры сходства:
//       =  0 — евклидово расстояние,
//       =  1 — манхеттенское расстояние,
//       =  2 — расстояние Махаланобиса,
//       =  3 — сюпремум-норма,
//       =  4 — расстояние Пирсона,
//       =  5 — расстояние Спирмэна,
//       =  6 — расстояние Кендалла,
//       =  7 — расстояние Жаккара,
//       =  8 — расстояние Рассела-Рао,
//       =  9 — расстояние Бравайса,
//       = 10 — расстояние Юла.
//   inf — на выходе — массив номеров длиной n, содержащий информацию
//         о принадлежности объектов тому или иному кластеру,
//     c — массив центров тяжести кластеров размером m * k (k m-мерных
//         векторов друг за другом),
//  *sum — функция качества классификации.
// Возвращаемое значение:
//   0 — при нормальном окончании расчета,
//  -1 — при недостатке памяти,
//  -2 — при потере значимости.
```

```
//
{
 short i,j,l,    // Счетчик
       m1,m2,    // Счетчик одномерного массива
       Count,    // Счетчик элементов кластера
       Code = 0; // Код ошибки

 double *z,      // Расширенная матрица сходства
    *dat,        // Расширенная матрица данных
    *t,          // Матрица, обратная дисперсионно-ковариационной
    Crit1 = 0,   // Функция качества
    Crit;        // Функция качества на предыдущем шаге

 ////////////////////////////////////////////////////
 // Построение обратной дисперсионно-ковариационной
 // матрицы для вычисления расстояния Махалонобиса
 if (Mode == 2)
   {
    ////////////////////////////////////////////////////
    // Для экономии памяти массив t фактически определен
    // при Mode = 2, в остальный случаях — фиктивный массив
    if (!(t = new double[k * k]))
      return -1;

    ////////////////////////////////////////////////
    // Здесь z временно используется для хранения
    // дисперсионно-ковариационной матрицы
    if (!(z = new double[k * k]))
      {
       delete [] t;
       return -1;
      }

    ////////////////////////////////////////////////
    // Построение дисперсионно-ковариационной матрицы
    Code = CovMatrix (data,z,n,k);

    if (Code != 0)
```

```
      {
       delete [] t; delete [] z;
       return Code;
      }

    for (i = 0; i < k * k; i++)
      z[i] = fabs (z[i]);

    ////////////////////////////////////////////////
    // Матрица, обратная дисперсионно-ковариационной
    Code = InverseMatrix (z,t,k);

    ////////////////////////////////////
    // Теперь память для массива z свободна
    delete [] z;

    if (Code != 0)
      {
       delete [] t;
       return Code;
      }
  }
else
  if (!(t = new double[1]))
    return -1;

if (!(z = new double[(n + m) * (n + m)]))
  {
   delete [] t;
   return -1;
  }
if (!(dat = new double[(n + m) * k]))
  {
   delete [] t; delete [] z;
   return -1;
  }

//////////////////////////////////////////////
```

```
// Первые n строк расширенной матрицы данных
// совпадают с исходной матрицей данных
for (i = 0, m2 = 0; i < k; i++)
  for (j = 0, m1 = i * (n + m); j < n; j++)
    dat[m1++] = data[m2++];

//////////////////////////////////////////////
// Остальные m строк — центры тяжести кластеров
// Возьмем их совпадающими с первыми m объектами
for (i = 0; i < k; i++)
  for (j = 0, m1 = i * (n + m) + n, m2 = i * n; j < m; j++)
    dat[m1++] = data[m2++];

do
  {
   Crit = Crit1;

   ////////////////////////////////////////
   // Построение расширенной матрицы сходства
   if ((Code = LikenessMatrix (dat,z,t,n + m,k,Mode)) != 0)
     {
      delete [] t; delete [] z; delete [] dat;
      return Code;
     }

   ////////////////////////////////////////////////////
   // Анализ матрицы сходства на предмет разнесения объектов
   // по кластерам. Используется тот факт, что последние m строк
   // (или m столбцов) расширенной матрицы сходства — это
   // расстояния между объектами и центрами тяжести кластеров
   for (i = 0, m1 = n * (m + n); i < n; i++, m1++)
     for (j = 1, m2 = m1 + m + n, inf[i] = 0; j < m; j++, m2 += m + n)
   if (z[m2] < z[m1])
     inf[i] = j;

   ////////////////////////////////////////
   // Вычисление новых центров тяжести кластеров
```

```
    for (i = 0; i < k; i++)
      for (j = 0, m1 = i * (n + m) + n; j < m; j++, m1++)
    {
     ////////////////////////////////////////
     // Выбор объектов, включенных в кластер j
     for (l = 0, Count = 0, dat[m1] = 0, m2 = i * n; l < n; l++, m2++)
       if (j == inf[l])
         {
          dat[m1] += data[m2];
          Count++;
         }
     ////////////////////////
     // Если кластер не пустой
     if (Count != 0)
       dat[m1] /= Count;
    }

    ////////////////////////////////////////////
    // Вычисление функции качества классификации
    for (i = 0, Crit1 = 0; i < k; i++)
      for (j = 0, m1 = i * (n + m) + n; j < m; j++,m1++)
    for (l = 0, m2 = i * n; l < n; l++, m2++)
      if (j == inf[l])
        Crit1 += fabs (dat[m1] — data[m2]);

   } while (Crit1 != Crit);

 //////////////////////////
 // Центры тяжести кластеров
 for (i = 0, m1 = 0; i < k; i++)
   for (j = 0, m2 = i * (n + m) + n; j < m; j++)
     c[m1++] = dat[m2++];

 delete [] z; delete [] dat; delete [] t;

 return Code;
}
```

Визуализация результатов кластерного анализа

Хотя от графического представления результатов расчета методами кластерного анализа можно отказаться, иерархические методы кластерного анализа становятся более наглядными, если результаты расчета представить в виде специального графика, называемого дендрограммой (дендограммой).

Рассмотрим построение дендрограммы по результатам расчета, а также выделение заданного числа кластеров по построенной дендрограмме в иерархических методах кластерного анализа (из рассмотренных нами это методы ближней связи, Уорда и средней связи Кинга).

Предположим, после применения одного из иерархических методов получены результаты классификации в виде величин связи для пар объектов. Идея построения дендрограммы проста — пары объектов, отмеченные по оси абсцисс, соединяются в соответствии с уровнем связи, отложенным по оси ординат (рис. 7.1).

Рис. 7.1. Дендрограмма иерархического метода

Теперь, когда дендрограмма построена, поясним, каким образом выделяется заданное число кластеров. Например, задав число кластеров, равное 3, мы смотрим, на каком уровне связи число пересечений горизонтальной линии, соответствующей уровню связи, и вертикальных линий, соответствующих объектам, равно трем. Находим, что такому уровню соответствует связь 60 и ниже. Разматывая деревья объединения объектов от точки пересечения го-

ризонтали вплоть до уровня связи, равного нулю, находим все объекты, соответствующие кластерам 1, 2 и 3. Первому кластеру в нашем примере принадлежат объекты 1, 2, 3 и 4. Второму кластеру — объекты 5 и 6. Третьему — объекты 7 и 8. Конечно, в данном искусственном примере конфигурация объектов и нумерация кластеров являются произвольными.

Руководство пользователя

Для работы с модулем кластерного анализа (Cla) выберите из меню Microsoft Excel пункт **AtteStat ▸ Модуль Cla ▸ Кластерный анализ**. На экране появится диалоговое окно, изображенное на рис. 7.2. Оно довольно густо насыщено параметрами, что вызвано спецификой решаемой задачи. Выбор вариантов сочетаний мер и методов придает дополнительную гибкость программе.

Рис. 7.2. Окно модуля кластерного анализа

1. Выберите или введите интервалы матрицы исходных данных.
2. Выберите или введите выходной интервал. Начиная с первой ячейки выходного интервала, будут выведены результаты вычислений.
3. Выберите или оставьте по умолчанию меру различия (кроме метода Уорда, использующего только Евклидово расстояние).
4. Выберите или оставьте по умолчанию метод анализа.
5. Введите число кластеров (действительно только для метода *k*-средних, для иерархических методов число кластеров выделяется вручную).

6. Укажите или оставьте по умолчанию, как расположены классифицируемые объекты. Обычно принимают, что объекты расположены в строках, свойства объектов — в столбцах. Данная опция, кроме удобства по классификации объектов, позволяет выбрать, что классифицировать — объекты или параметры.
7. Нажмите кнопку **Выполнить расчет**.

После выполнения вычислений будет, начиная с первой ячейки выходного интервала, выведено название метода и результаты расчета.

За выбор адекватного исходным данным метода расчета несет ответственность пользователь. Cla берет на себя верификацию исходных данных, выдавая подробную диагностику. При неверных действиях пользователя выдаются сообщения об ошибках.

Сообщения об ошибках

При ошибках ввода данных могут выдаваться диагностические сообщения следующих указанных типов.

Рис. 7.3. Окно сообщения об ошибке

На рис 7.3 вы не выбрали или неверно ввели входной интервал. Лучшим способом избежать ошибки является не ввод, а выделение интервала протаскиванием курсора.

Рис. 7.4. Окно сообщения об ошибке

Проверьте исходные данные на рис.7.4 и заполните все ячейки входного интервала. Если в ячейке не должно быть данных, по физической природе исследуемого процесса, введите нуль.

Рис. 7.5. Окно сообщения об ошибке

Проверьте типы ячеек входного интервала на рис. 7.5. Тип может быть только числовым. Проще всего выделить интервал ячеек и явно определить их тип как числовой средствами Microsoft Excel.

Рис. 7.6. Окно сообщения об ошибке

На рис. 7.6 не задано число кластеров для метода k-средних. Введите число кластеров.

Рис. 7.7. Окно сообщения об ошибке

На рис. 7.7 не выбран или неверно введен выходной интервал. Лучшим способом избежать ошибки является не ввод, а выделение интервала стандартным образом.

При решении тестовых задач вычислительный модуль показал стабильность.

Распознавание образов с обучением

Методы распознавания образов с обучением (с учителем) предназначены для отнесения неклассифицированных объектов (явлений, ситуаций, процессов) к заранее описанным кластерам (классам, обучающим группам). В

этом случае программа (или устройство) распознавания называется классификатором. Классическими классификаторами являются методы дискриминантного анализа.

Если устройство распознавания выдает ответ в виде действительного числа или вектора действительных чисел, оно называется, соответственно, предиктором, или локализатором. Метод множественного регрессионного анализа является классическим локализатором.

В соответсвии с результатами, выдаваемыми классификаторами и локализаторами, обучающие данные должны быть подготовлены так, чтобы использовать все преимущества того или иного метода. Например, мы изучаем влияние группы параметров на доход населения. Если используются методы дискриминантного анализа, то обучающие классы допустимо описать как, скажем, группы с низким, средним и высоким доходом на одного члена семьи. Им могут соответствовать классы 1, 2, 3. Однако если используется множественный регрессионный анализ, такое представление фактически приведет к потере ценной информации, содержащейся в исходных данных, т. к. понизит количественную шкалу, с которой работает данный медод, до шкалы порядковой[1]. Для множественного регрессионного анализа ответ лучше представить в виде суммы дохода в расчете на одного члена семьи.

Особенности решения

В представленном решении применен элемент управления MultiPage. Разработанная программа, чтобы быть максимально полезной пользователю, должна выполнять и функции классификатора, и функции предиктора. Как раз элемент MultiPage позволяет в удобной и компактной форме отобразить как обучающие, так и распознающие алгоритмы в пределах одной формы.

Научные алгоритмы

Дискриминантный анализ представляет собой линейный (линейным он является потому, что модель метода линейна относительно дискриминантных функций) метод распознавания данных с обучением. Пользователь должен задать некоторое число объектов, указав их принадлежность к так называемым обучающим группам (классам, кластерам, популяциям). Поэтому применению методов распознавания обязательно должны

---

[1] *Подробно о шкалах измерений см.: Гайдышев И. Анализ и обработка данных: специальный справочник. — СПб: Питер, 2001.*

предшествовать исследования методами классификации без обучения (кластерного анализа или эмпирической классификации). В задачах, построенных на реальных экспериментальных данных, классы могут пересекаться, особенно если обучение производится на основании эмпирической классификации. Пересечение классов обычно ухудшает качество классификации, а наилучшие для данного набора данных результаты распознавания получаются в случае непересекающихся классов.

Методы распознавания образов с обучением используют в качестве обучающих выборок объекты, заранее классифицированные тем или иным способом. Качество процедуры дискриминации определяется вероятностью правильной классификации. Очень хорошие результаты для распознавания образов с обучением дает предварительное применение метода ближней связи, а применение метода k-средних для предварительного отнесения объектов классам дает практически 100% качество распознавания для любого метода дискриминантного анализа. Это обусловлено тем, что из рассмотренных нами методов распознавания без обучения только метод k-средних на основе выбранной метрики гарантированно строит непересекающиеся кластеры. Данные методы подробно описаны в разделе «Кластерный анализ» настоящей главы.

Методы распознавания образов с обучением вырабатывают некоторые решающие правила, позволяющие отнести предлагаемые объекты к заданным классам. Решающие правила могут быть получены:

1. в виде простых классифицирующих функций, как это сделано в линейном дискриминантном анализе Фишера;
2. в виде дискриминантных функций, как это сделано в каноническом дискриминантном анализе;
3. в виде некоторых характеристик (групповая ковариационная матрица, групповой вектор средних и определитель ковариационной матрицы), как это сделано в линейном дискриминантном анализе;
4. в виде набора коэффициентов регрессии (без свободного члена или со свободным членом), как это сделано в методе линейного множественного регрессионного анализа.

Качество классификации методами дискриминантного анализа в литературе оценивается различными способами, в том числе особыми статистическими критериями. Наиболее простым и понятным способом, которым мы пользуемся, является оценка качества дискриминации относительным процентным содержанием, объектов обучающей выборки в объеме выборки.

Качество построения множественной регрессионной модели удобно оценивать средним квадратичным отклонением.

Линейный дискриминантный анализ Фишера

Метод линейного дискриминантного анализа (линейная дискриминация Фишера, дискриминаторный анализ) Фишера предполагает, что классификация должна проводиться с помощью линейной комбинации дискриминантных (различающих) переменных. Основанием отнесения объекта к кластеру (классу, популяции) является наибольшее значение так называемой простой классифицирующей функции h_k для k-го класса, являющейся линейной комбинацией дискриминантных переменных:

$$h_k = b_{k0} + \sum_{i=1}^{p} b_{ki} X_i \ ,$$

где p — число дискриминантных переменных,

b_{ki} — коэффициент для i-й переменной k-го класса, определяемый как

$$b_{ki} = (n_{.} - g) \sum_{j=1}^{p} a_{ij} X_{jk.},$$

где $n_{.}$ — общее число наблюдений по всем классам,

a_{ij} — элементы матрицы, обратной к матрице W — матрице разброса внутри классов (внутригрупповая матрица сумм попарных произведений), вычисляемой по формуле

$$w_{ij} = \sum_{k=1}^{g} \sum_{m=1}^{n_k} \left(X_{ikm} - X_{ik.} \right) \left(X_{jkm} - X_{jk.} \right),$$

где g — число классов,

n_k — число наблюдений в k-м классе,

X_{ikm} — значение i-ой дискриминантной переменной (величина i-й переменной m-го наблюдения k-го класса),

$X_{ik.}$ — среднее i-й переменной k-го класса.

В основе метода лежат предположения:

1. популяции, среди которых производится дискриминация, подчиняются многомерному нормальному распределению;
2. популяции, среди которых производится дискриминация, имеют статистически неразличимые ковариационные матрицы.

При искусственном объявлении ковариационных матриц статистически неразличимыми могут оказаться отброшенными важные индивидуальные черты, имеющие большое значение для хорошей дискриминации. Однако введенное предположение позволяет получить решение и в случае, когда количество обучающих выборок в кластере оказывается меньшим количества дискриминантных функций.

Результаты линейного дискриминантного анализа Фишера совпадают, в смысле качества классификации, с результатами более сложного в реализации канонического дискриминантного анализа, что наблюдалось во всех сделанных нами расчетах.

Представленная на листинге функция выполнена на языке C и предназначена для вычисления параметров обучения (коэффициентов простых классифицирующих функций).

На листинге 7.6 также представлена функция распознавания одного объекта на основе вычисленных параметров. Данная функция может быть использована и для вычисления качества распознавания, и для распознавания вновь введенных объектов, не участвовавших в обучении.

Листинг 7.6

```
short __stdcall FDA (double z[],short n,short k,short l[],short g,short
h[],double p[])
//
// Функция производит линейный дискриминантный анализ по Фишеру.
// Обозначения:
//   z - матрица исходных данных размером n * k
//       (k n-мерных векторов друг за другом),
//   n - общее число объектов, включает в себя и обучающие выборки,
//   k - число параметров,
//   l - массив длиной g размеров k-мерных выборок (числа объектов)
//       в каждом из обучающих классов,
//   g - число обучающих групп, оно же число дискриминантных функций,
//   h - вектор, содержащий номера объектов, составляющих обучающие клас-
сы,
//       номера расположены в соответствии с l друг за другом,
//   p - массив длиной g * (k + 1) коэффициентов g дискриминантных
//       (простых классифицирующих) функций, расположенных друг за дру-
гом,
//       начиная со свободного члена.
// Возвращаемое значение:
//   0 при нормальном окончании расчета,
//  -1 при недостатке памяти,
//  -2 при потере значимости.
//
{
 short i,j,i1,          // Счетчики цикла
```

```
      m1,m2,m3,m4,m5, // Индексы
      Code;           // Код ошибки

double *x,            // Матрица групповых средних
       *t,*d,         // Рабочий массив
       *s,            // Текущая групповая дисперсионная матрица
       buf;           // Рабочая переменная

if ((d = new double[k * k]) == 0)
  return -1;
if ((s = new double[k * k]) == 0)
  {
   delete [] d;
   return -1;
  }
if ((x = new double[k * g]) == 0)
  {
   delete [] s; delete [] d;
   return -1;
  }

//////////////////////////////////////////////////////
// Начальные значения объединенной дисперсионной матрицы
FillUp (d,k * k,0);

//////////////////////////////////////////////
// Построение объединенной дисперсионной матрицы
// и матрицы групповых средних
for (i = 0, m3 = 0, m5 = 0; i < g; i++)
  {
   ////////////////////////////////////
   // Групповая матрица исходных данных
   if ((t = new double[l[i] * k]) == 0)
     {
      delete [] s; delete [] d; delete [] x;
      return -1;
     }
   for (i1 = 0, m1 = 0, m4 = 0; i1 < k; i1++, m4 += n)
```

```
      for (j = 0, m2 = m3; j < l[i]; j++)
        t[m1++] = z[h[m2++] + m4 - 1];
    m3 += l[i];

    ///////////////////////////
    // Матрица групповых средних
    for (i1 = 0, m2 = 0; i1 < k; i1++)
      {
       for (j = 0, x[m5] = 0; j < l[i]; j++)
         x[m5] += t[m2++];
       x[m5++] /= l[i];
      }

    ////////////////////////////////////////////////
    // Групповая дисперсионно-ковариационная матрица
    if (Likeness (t,s,l[i],k,1) != 0)
      {
       delete [] s; delete [] d; delete [] x; delete [] t;
       return -1;
      }

    ////////////////////////////////////////
    // Групповая матрица взаимных произведений
    // отклонений от средних
    MatrixMultiplyNumber (s,k * k,l[i] - 1);

    ////////////////////////////////////////////
    // Объединенная матрица взаимных произведений
    // отклонений от средних
    MatrixPlusMatrix (d,s,k * k);

    delete [] t;
   }

delete [] s;

////////////////////////////////////////////////
// Объединенная дисперсионно-ковариационная матрица
```

```
MatrixMultiplyNumber (d,k * k,1.0 / (m3 - g));

//////////////////////////////////////////////
// Матрица, обратная дисперсионно-ковариационной
if ((t = new double[k * k]) == 0)
  {
   delete [] d; delete [] x;
   return -1;
  }

if ((Code = InverseMatrix (d,t,k)) != 0)
  {
   delete [] d; delete [] x; delete [] t;
   return Code;
  }

delete [] d;

//////////////////////
// Вектор общих средних
if ((d = new double[k]) == 0)
  {
   delete [] x; delete [] t;
   return -1;
  }
for (j = 0; j < k; j++)
  {
   for (i1 = 0, m2 = j, d[j] = 0; i1 < g; i1++, m2 += k)
     d[j] += l[i1] * x[m2];
   d[j] /= m3;
  }

//////////////////////////////////////////////////////////
// Вычисление коэффициентов простых классифицирующих функций
for (i1 = 0, m1 = 0; i1 < g; i1++)
  {
   for (j = 0, p[m1] = 0; j < k; j++)
     for (i = 0; i < k; i++)
```

```
        p[m1] += t[j + i * k] * x[j + i1 * k] * x[i + i1 * k];
    p[m1++] /= (-2);

    for (i = 0; i < k; i++, m1++)
      for (j = 0, p[m1] = 0; j < k; j++)
        p[m1] += t[i + j * k] * x[j + i1 * k];
   }

 delete [] d; delete [] x; delete [] t;

 return 0;
}

short __stdcall FDAClass (double x[],double w[],short k,short g,short
*c,double *p)
//
// Функция производит классификацию объекта по данным
// линейного дискриминантного анализа по Фишеру.
// Обозначения:
//   x - вектор исходных данных,
//   w - массив простых классифицирующих функций,
//   k - число параметров,
//   g - число классов,
//  *c - номер класса,
//  *p - соответствующая вероятность.
// Возвращаемое значение:
//   0 при нормальном окончании расчета,
//  -1 при недостатке памяти.
//
{
 short i,j,        // Счетчик
       s = 0;      // Индекс элемента массива
 double Value = 0, // Величина дискриминантной функции
        *d;        // Вектор дискриминантных функций объекта

 if ((d = new double[g]) == 0)
   return -1;

 //////////////////////////
```

```
 // Вычисление номера класса
 for (i = 0, *c = 0; i < g; i++)
   {
    d[i] = w[s++];
    for (j = 0; j < k; j++)
     d[i] += w[s++] * x[j];
    if (d[i] > Value)
      {
       Value = d[i];
       *c = i + 1;
      }
   }

 ////////////////////////////////////////////////////
 // Расчет вероятности принадлежности объекта классу
 for (i = 0, *p = 0; i < g; i++)
   *p += (*p) + exp(d[i] - Value);

 *p = 1.0 - 1.0 / (*p);

 delete [] d;

 return 0;
}
```

На листинге 7.7 представлена интерфейсная функция к функции распознавания одного объекта. Данная функция составлена на языке VBA и предназначена для верификации введенных данных, вызова функции распознавания и вывода результатов распознавания.

Листинг 7.7

```
Sub ReconFisher(Data1 As Variant, Data2 As Variant)
'
' Функция производит классификацию объекта по данным
' линейного дискриминантного анализа Фишера.
' Обозначения:
'  Data1 - интервал объекта
'  Data2 - интервал простых классифицирующих функций
'
```

```
  Dim I As Integer          ' Счетчик
  Dim J As Integer          ' Счетчик
  Dim S As Integer          ' Счетчик
  Dim K As Integer          ' Число параметров
  Dim N As Integer          ' Рабочая переменная
  Dim G As Integer          ' Число классов
  Dim C As Integer          ' Номер наиболее вероятного класса
  Dim P As Double           ' Вероятность принадлежности объекта классу
  Dim X() As Double         ' Вектор исходных данных k
  Dim W() As Double         ' Массив длиной g * k классифицирующих
                            ' функций
  Dim Value As Variant      ' Рабочая переменная
  Dim RData As Range        ' Вспомогательный объект
  Dim Result As Integer     ' Код ошибки
  Dim MsgTitle As String    ' Заголовок сообщения
  Dim MsgTitleEr As String  ' Заголовок сообщения об ошибке
  Dim MsgStyle As Integer   ' Стиль окна сообщения
  Dim MsgStyleEr As Integer ' Стиль окна сообщения об ошибке
  MsgTitle = "Результат распознавания"
  MsgTitleEr = "Ошибка в исходных данных"
  MsgStyle = vbOKOnly + vbInformation
  MsgStyleEr = vbOKOnly + vbCritical
  '
  ' Анализ исходных данных
  '
  On Error GoTo ErrHandler1
    Range(Data1).Select
  On Error GoTo 0
  K = 0
  For Each Value In ActiveSheet.Range(Data1)
    If IsEmpty(Value) Then
      MsgBox "Пустая ячейка в области данных. Расчет невозможен.",
MsgStyleEr, MsgTitleEr
      Exit Sub
    End If
    If Not IsNumeric(Value) Then
      MsgBox "Нечисловой тип данных. Расчет невозможен.", MsgStyleEr,
MsgTitleEr
      Exit Sub
```

```
    End If
    K = K + 1
  Next Value
  On Error GoTo ErrHandler1
    Range(Data2).Select
  On Error GoTo 0
  For Each Value In ActiveSheet.Range(Data2)
    If IsEmpty(Value) Then
      MsgBox "Пустая ячейка в области данных. Расчет невозможен.",
MsgStyleEr, MsgTitleEr
      Exit Sub
    End If
    If Not IsNumeric(Value) Then
      MsgBox "Нечисловой тип данных. Расчет невозможен.", MsgStyleEr,
MsgTitleEr
      Exit Sub
    End If
  Next Value
  On Error GoTo ErrHandler1
    Range(Data2).Select
  On Error GoTo 0
  For Each Value In ActiveSheet.Range(Data3)
    If IsEmpty(Value) Then
      MsgBox "Пустая ячейка в области данных. Расчет невозможен.",
MsgStyleEr, MsgTitleEr
      Exit Sub
    End If
    If Not IsNumeric(Value) Then
      MsgBox "Нечисловой тип данных. Расчет невозможен.", MsgStyleEr,
MsgTitleEr
      Exit Sub
    End If
  Next Value
  '
  ' Получение и верификация исходных данных
  '
  ReDim X(K)
  I = 1
  For Each Value In ActiveSheet.Range(Data1)
    X(I) = Value
```

```
    I = I + 1
  Next Value
  Set RData = Range(Data2)
  N = RData.Rows.Count
  G = RData.Columns.Count
  If Not K + 1 = N Then
    MsgBox "Несоответствие числа параметров. Расчет невозможен.",
MsgStyleEr, MsgTitleEr
    Exit Sub
  End If
  If G < 2 Then
    MsgBox "Мало классов.", MsgStyleEr, MsgTitleEr
    Exit Sub
  End If
  ReDim W(G * N)
  S = 1
  For I = 1 To G
    For J = 1 To N
      W(S) = RData.Cells(J, I)
      S = S + 1
    Next J
  Next I
  '
  ' Вычисление номера класса и соответствующей вероятности
  '
  Result = FDAClass(X(1), W(1), K, G, C, P)
  If Result = -1 Then
    MsgBox "Недостаточно памяти.", MsgStyleEr, MsgTitleEr
    Exit Sub
  Else
    MsgBox "Объект принадлежит классу " & C & Chr(13) & _
           "с вероятностью " & P, MsgStyle, MsgTitle
  End If
Exit Sub
ErrHandler1:
  MsgBox "Не определен интервал переменной. Расчет невозможен.",
MsgStyleEr, MsgTitleEr
  Exit Sub
End Sub
```

Канонический дискриминантный анализ

Канонический дискриминантный анализ основан на определении так называемых дискриминантных функций, количество которых меньше либо равно числу параметров объектов:

$$f_{km} = u_o + \sum_{i=1}^{p} u_i X_{ikm} ,$$

где f_{km} — значение канонической дискриминантной функции для m-го объекта k-го класса,

u_i — коэффициенты, определяемые по формуле

$$u_i = v_i \sqrt{n_{.} - g}, u_0 = -\sum_{i=1}^{p} u_i X_{i..} ,$$

где $X_{i..}$ — среднее *i*-й переменной по всем классам,

$v_i, i = 1,2,...,p,$ — коэффициенты, вычисляемые как компоненты собственных векторов решения обобщенной проблемы собственных значений:

$$Bv = \lambda Wv,$$

где B — межгрупповая сумма квадратов отклонений,

v — собственный вектор,

остальные обозначения те же, что и в предыдущем разделе.

Матрица B определяется как

$$B = T - W,$$

где T — матрица сумм квадратов и попарных произведений, элементы которой вычисляются как

$$t_{ij} = \sum_{k=1}^{g} \sum_{m=1}^{n_k} \left(X_{ikm} - X_{i..}\right)\left(X_{jkm} - X_{j..}\right).$$

Представленная на листинге 7.8 функция выполнена на языке C и предназначена для вычисления параметров обучения (коэффициентов канонических дискриминантных функций и массива центроидов).

На листинге также представлена функция распознавания одного объекта на основе вычисленных параметров. Данная функция может быть использована и для вычисления качества распознавания, и для распознавания вновь введенных объектов, не участвовавших в обучении.

Листинг 7.8

```
short __stdcall CDA (double z[],short n,short k,short l[],short g,short
h[],double p[],double c[])
//
// Функция производит канонический дискриминантный анализ.
// Обозначения:
//   z - матрица исходных данных размером n * k (k n-мерных векторов друг
за другом),
//   n - общее число объектов, включает в себя и обучающие выборки,
//   k - число параметров, оно же число канонических функций,
//   l - массив длиной g размеров k-мерных выборок (числа объектов) в ка-
ждом из обучающих классов,
//   g - число обучающих классов,
//   h - вектор, содержащий номера объектов, составляющих обучающие клас-
сы,
//       номера расположены в соответствии с l друг за другом,
//   p - массив длиной (k + 1) * k коэффициентов канонических
//       дискриминантных функций, расположенных друг за другом,
//       начиная со свободного члена,
//   c - массив центроидов обучающих классов размером k * g.
// Возвращаемое значение:
//   0 при нормальном окончании расчета,
//  -1 при недостатке памяти,
//  -2 при потере значимости.
//
{
 short i,j,i1,       // Счетчики цикла
       m1,m2,m4,m5,  // Индексы
       m3,           // Общее число обучающих выборок (объектов)
       Code;         // Рабочая переменная

 double *t,*d,       // Рабочий массив
        *x,          // Вектор общих средних
        buf,         // Рабочая переменная
        *u,          // Вектор собственных значений
        *r,          // Общая корреляционная матрица
        *w;          // Внутригрупповая корреляционная матрица
```

```
if ((d = new double[k * k]) == 0)
  return -1;
if ((r = new double[k * k]) == 0)
  {
   delete [] d;
   return -1;
  }
if ((w = new double[k * k]) == 0)
  {
   delete [] d; delete [] r;
   return -1;
  }
if ((u = new double[k]) == 0)
  {
   delete [] d; delete [] r; delete [] w;
   return -1;
  }

////////////////////
// Начальные значения
FillUp (d,k * k,0);

for (i = 0, m3 = 0, m5 = 0; i < g; i++)
  {
   ////////////////////////////////////
   // Групповая матрица исходных данных
   if ((t = new double[l[i] * k]) == 0)
     {
      delete [] d; delete [] u; delete [] r; delete [] w;
      return -1;
     }
   for (i1 = 0, m1 = 0, m4 = 0; i1 < k; i1++, m4 += n)
     for (j = 0, m2 = m3; j < l[i]; j++)
       t[m1++] = z[h[m2++] + m4 - 1];
   m3 += l[i];

   ////////////////////////////////////
   // Массив средних обучающих классов
```

```
    for (j = 0, m1 = 0; j < k; j++, m5++)
      {
       for (i1 = 0, c[m5] = 0; i1 < l[i]; i1++)
         c[m5] += t[m1++];
       c[m5] /= l[i];
      }

    ////////////////////////////////////////////////////
    // Внутригрупповая дисперсионно-ковариационная матрица
    if (Likeness (t,p,l[i],k,1) != 0)
      {
       delete [] d; delete [] t; delete [] u; delete [] r; delete [] w;
       return -1;
      }

    ////////////////////////////////////////
    // Групповая матрица взаимных произведений
    // отклонений от средних
    MatrixMultiplyNumber (p,k * k,l[i] - 1);

    ////////////////////////////////////////////////
    // Внутригрупповая матрица взаимных произведений
    // отклонений от средних
    MatrixPlusMatrix (d,p,k * k);

    delete [] t;
   }

////////////////////////////////////////
// Внутригрупповая корреляционная матрица
for (i = 0, m1 = 0, m2 = 0; i < k; i++, m2 += k + 1)
  for (j = 0, m4 = 0; j < k; j++, m1++, m4 += k + 1)
    w[m1] = d[m1] / sqrt (d[m2] * d[m4]);

//////////////////////////
// Общая матрица наблюдений
if ((t = new double[m3 * k]) == 0)
  {
```

```
    delete [] d; delete [] u; delete [] r; delete [] w;
    return -1;
   }
for (j = 0; j < m3; j++)
  for (i = 0, m1 = j, m2 = h[j] - 1; i < k; i++, m1 += m3, m2 += n)
    t[m1] = z[m2];

//////////////////////
// Вектор общих средних
if ((x = new double[k]) == 0)
  {
   delete [] d; delete [] t; delete [] u; delete [] r; delete [] w;
   return -1;
  }
for (j = 0; j < k; j++)
  {
   for (i = 0, m1 = j, x[j] = 0; i < g; i++, m1 += k)
     x[j] += c[m1];
   x[j] /= g;
  }

//////////////////////////////////////////
// Общая дисперсионно-ковариационная матрица
if (Likeness (t,p,m3,k,1) == -1)
  {
   delete [] d; delete [] t; delete [] x; delete [] u; delete [] r;
   delete [] w;
   return -1;
  }

//////////////////////////////
// Общая корреляционная матрица
if (Likeness (t,r,m3,k,0) == -1)
  {
   delete [] d; delete [] t; delete [] x; delete [] u; delete [] r;
   delete [] w;
   return -1;
  }
```

```
//////////////////////////////////////////////////////////
// Общая матрица взаимных произведений отклонений от средних
MatrixMultiplyNumber (p,k * k,m3 - 1);

//////////////////////////////////////////////////
// Межгрупповая матрица сумм квадратов и попарных
// произведений
MatrixMinusMatrix (p,d,k * k);

////////////////////////////////////////
// Решение проблемы собственных значений
if (Eigen (p,d,k,0.000001,&m1) != 0)
  {
   delete [] d; delete [] t; delete [] x; delete [] u; delete [] r;
   delete [] w;
   return Code;
  }

//////////////////////////////////////////////
// Сортировка собственных значений и векторов
for (i = 0; i < k; i++)
  u[i] = p[i * (k + 1)];

for (i = 0; i < k; i++)
  {
   Code = i;
   buf = u[i];
   for (j = i + 1; j < k; j++)
     if (buf < u[j])
       {
        Code = j;
        buf = u[j];
       }
   ColInterchange (u,1,i,Code);
   ColInterchange (d,k,i,Code);
  }
```

```
 ////////////////////////////////////////////////
 // Вычисление коэффициентов канонических функций
 ArrayToArray (d,t,k * k);
 MatrixMultiplyNumber (t,k * k,sqrt (m3 - g));
 for (i = 0, m2 = 0, m4 = 0; i < k; i++, m4 += k + 1)
   for (j = 0, m1 = i * (k + 1) + 1, p[m4] = 0; j < k; j++, m2++)
     {
      p[m1++] = t[m2];
      p[m4] -= t[m2] * x[j];
     }

 delete [] x; delete [] t; delete [] d;

 ////////////////////
 // Массив центроидов
 if ((t = new double[k * g]) == 0)
   return -1;
 ArrayToArray (c,t,k * g);
 for (i = 0, m1 = 0; i < g; i++)
   for (j = 0, m4 = 0; j < k; j++, m1++)
     {
      c[m1] = p[m4++];
      for (i1 = 0, m2 = i * k; i1 < k; i1++)
        c[m1] += p[m4++] * t[m2++];
     }

 delete [] t; delete [] u; delete [] r; delete [] w;

 return 0;
}

short __stdcall CDAClass (double x[],double w[],double u[],short k,short
g,short *c)
//
// Функция производит классификацию объектов по данным
// канонического дискриминантного анализа.
// Обозначения:
//   x - вектор исходных данных,
```

```
//   w - массив центроидов длиной g * k,
//   u - массив длиной k * (k + 1) коэффициентов k дискриминантных
//       функций,
//   k - число параметров,
//   g - число классов,
//  *c - номер класса.
// Возвращаемое значение:
//   0 при нормальном окончании расчета,
//  -1 при недостатке памяти.
//
{
 short i,j,    // Счетчики цикла
       m1,m2,  // Счетчик одномерного массива
       y = k;  // Число дискриминантных функций, <= k

 double r,r1,  // Расстояние между объектом и кластером
        *s1,   // Координаты объекта
        *s2;   // Координаты центра кластера

 if ((s1 = new double[y]) == 0)
   return -1;
 if ((s2 = new double[y]) == 0)
   {
    delete [] s1;
    return -1;
   }

 ////////////////////
 // Координаты объекта
 for (j = 0, m1 = 0; j < y; j++)
   {
    s1[j] = u[m1++];
    for (i = 0; i < k; i++)
      s1[j] += u[m1++] * x[i];
   }

 ////////////////////////////
 // Координаты центра кластера
```

```
 for (j = 0, m2 = 0; j < y; j++)
   s2[j] = w[m2++];

 ////////////////////////////////////
 // Пропустить дискриминантные функции,
 // соответствующие отброшенным собственным значениям
 for (j = y; j < k; j++)
   m2++;

 ////////////////////
 // Начальные значения
 r = Euclid (s1,s2,y);
 *c = 1;

 ////////////////////////////////
 // Проверка всех центров кластеров
 for (j = 1; j < g; j++)
   {
    for (i = 0; i < y; i++)
      s2[i] = w[m2++];
    for (i = y; i < k; i++)
      m2++;
    r1 = Euclid (s1,s2,y);
    if (r1 < r)
      {
       r = r1;
       *c = j + 1;
      }
   }

 delete [] s1; delete [] s2;

 return 0;
}
```

На следующем листинге представлен интерфейс к функции распознавания одного объекта. Данная функция составлена на языке VBA и предназначена для верификации введенных данных, вызова функции распознавания и вывода результатов распознавания.

Листинг 7.9

```
Sub ReconCanon(Data1 As Variant, Data2 As Variant, Data3 As Variant)
'
' Функция производит классификацию объекта по данным
' канонического дискриминантного анализа.
' Обозначения:
'  Data1 - интервал объекта
'  Data2 - интервал канонических функций функций
'  Data3 - интервал центроидов
'
  Dim I As Integer          ' Счетчик
  Dim J As Integer          ' Счетчик
  Dim S As Integer          ' Счетчик
  Dim K As Integer          ' Число параметров
  Dim N As Integer          ' Рабочая переменная
  Dim G As Integer          ' Число классов
  Dim C As Integer          ' Номер наиболее вероятного класса
  Dim P As Double           ' Вероятность принадлежности объекта классу
  Dim X() As Double         ' Вектор исходных данных k
  Dim V() As Double         ' Массив длиной (k + 1) * k канонических
                            ' функций
  Dim W() As Double         ' Массив длиной g * k центроидов
  Dim Value As Variant      ' Рабочая переменная
  Dim RData As Range        ' Вспомогательный объект
  Dim Result As Integer     ' Код ошибки
  Dim MsgTitle As String    ' Заголовок сообщения
  Dim MsgTitleEr As String  ' Заголовок сообщения об ошибке
  Dim MsgStyle As Integer   ' Стиль окна сообщения
  Dim MsgStyleEr As Integer ' Стиль окна сообщения об ошибке
  MsgTitle = "Результат распознавания"
  MsgTitleEr = "Ошибка в исходных данных"
  MsgStyle = vbOKOnly + vbInformation
  MsgStyleEr = vbOKOnly + vbCritical
  '
  ' Анализ исходных данных
  '
  On Error GoTo ErrHandler1
```

```
    Range(Data1).Select
  On Error GoTo 0
  K = 0
  For Each Value In ActiveSheet.Range(Data1)
    If IsEmpty(Value) Then
      MsgBox "Пустая ячейка в области данных. Расчет невозможен.",
MsgStyleEr, MsgTitleEr
      Exit Sub
    End If
    If Not IsNumeric(Value) Then
      MsgBox "Нечисловой тип данных. Расчет невозможен.", MsgStyleEr,
MsgTitleEr
      Exit Sub
    End If
    K = K + 1
  Next Value
  On Error GoTo ErrHandler1
    Range(Data2).Select
  On Error GoTo 0
  For Each Value In ActiveSheet.Range(Data2)
    If IsEmpty(Value) Then
      MsgBox "Пустая ячейка в области данных. Расчет невозможен.",
MsgStyleEr, MsgTitleEr
      Exit Sub
    End If
    If Not IsNumeric(Value) Then
      MsgBox "Нечисловой тип данных. Расчет невозможен.", MsgStyleEr,
MsgTitleEr
      Exit Sub
    End If
  Next Value
  On Error GoTo ErrHandler1
    Range(Data3).Select
  On Error GoTo 0
  For Each Value In ActiveSheet.Range(Data3)
    If IsEmpty(Value) Then
      MsgBox "Пустая ячейка в области данных. Расчет невозможен.",
MsgStyleEr, MsgTitleEr
      Exit Sub
    End If
```

```
    If Not IsNumeric(Value) Then
      MsgBox "Нечисловой тип данных. Расчет невозможен.", MsgStyleEr,
MsgTitleEr
      Exit Sub
    End If
  Next Value
  '
  ' Получение и верификация исходных данных
  '
  ReDim X(K)
  I = 1
  For Each Value In ActiveSheet.Range(Data1)
    X(I) = Value
    I = I + 1
  Next Value
  Set RData = Range(Data2)
  G = RData.Rows.Count
  N = RData.Columns.Count
  If Not K = N Then
    MsgBox "Несоответствие числа параметров. Расчет невозможен.",
MsgStyleEr, MsgTitleEr
    Exit Sub
  End If
  If G < 2 Then
    MsgBox "Мало классов.", MsgStyleEr, MsgTitleEr
    Exit Sub
  End If
  ReDim W(G * K)
  S = 1
  For I = 1 To G
    For J = 1 To K
      W(S) = RData.Cells(J, I)
      S = S + 1
    Next J
  Next I
  Set RData = Range(Data2)
  N = RData.Rows.Count
  If Not K + 1 = N Then
```

```
    MsgBox "Несоответствие числа параметров. Расчет невозможен.",
MsgStyleEr, MsgTitleEr
    Exit Sub
  End If
  N = RData.Columns.Count
  If Not K = N Then
    MsgBox "Несоответствие числа параметров. Расчет невозможен.",
MsgStyleEr, MsgTitleEr
    Exit Sub
  End If
  ReDim V((K + 1) * K)
  S = 1
  For I = 1 To K
    For J = 1 To K + 1
      V(S) = RData.Cells(J, I)
      S = S + 1
    Next J
  Next I
  '
  ' Вычисление номера класса и соответствующей вероятности
  '
  Result = CDAClass(X(1), W(1), V(1), K, G, C)
  If Result = -1 Then
    MsgBox "Недостаточно памяти.", MsgStyleEr, MsgTitleEr
    Exit Sub
  Else
    MsgBox "Объект принадлежит классу " & C, MsgStyle, MsgTitle
  End If
Exit Sub
ErrHandler1:
  MsgBox "Не определен интервал переменной. Расчет невозможен.",
MsgStyleEr, MsgTitleEr
  Exit Sub
End Sub
```

Линейный дискриминантный анализ

Недостатком метода линейной дискриминации Фишера является предположение о равенстве ковариационных матриц рассматриваемых выборок, вследствие чего могут оказаться отброшенными важные индивидуальные

черты, имеющие большое значение для хорошей дискриминации. В методе линейного дискриминантного анализа, напротив, ковариационные матрицы для различных классов считаются различными.

В случае, когда количество обучающих выборок в кластере оказывается меньше количества дискриминантных функций, программа выдаст сообщение об ошибке при вычислении, а пользователю придется применить линейный дискриминантный анализ Фишера или канонический дискриминантный анализ.

В рассматриваемом методе основанием отнесения объекта к классу является наибольшее значение для данного объекта функции плотности нормального распределения среди всех классов. Формула плотности многомерного нормального распределения приводится в *приложении 2*. Вектор средних значений, входящих в данную формулу, а также дисперсионно-ковариационная матрица для каждого обучающего класса, оцениваются по исходным данным на этапе обучения.

Отсутствие простых решающих правил (для получения результата нужно проделать довольно объемные вычисления) было некоторым препятствием для широкого применения этого мощного метода в период, предшествовавший распространению персональных компьютеров. Метод использовался фактически редко, но здесь введен нами, как серьезная альтернатива линейному дискриминантному анализу Фишера. В наших расчетах метод линейного дискриминантного анализа показал более высокое качество распознавания по сравнению с линейной дискриминацией Фишера и каноническим дискриминантным анализом.

Представленная на листинге 7.10 функция выполнена на языке C и предназначена для вычисления параметров обучения (ковариационных матриц, средних и корней из определителей ковариационных матриц).

На листинге также представлена функция распознавания одного объекта на основе вычисленных параметров. Данная функция может быть использована и для вычисления качества распознавания, и для распознавания вновь введенных объектов, не участвовавших в обучении.

Листинг 7.10

```
short __stdcall LDA (double z[],short n,short k,short l[],short g,short
h[],double p[],double x[],double v[])
//
// Функция производит линейный дискриминантный анализ.
// Обозначения:
//   z - матрица исходных данных размером n * k
//       (k n-мерных векторов друг за другом),
```

```
//   n - общее число объектов, включает в себя и обучающие выборки,
//   k - число параметров,
//   l - массив длиной g размеров k-мерных выборок (числа объектов) в ка-
ждом из обучающих классов,
//   g - число обучающих классов,
//   h - вектор, содержащий номера объектов, составляющих обучающие
//       классы,
//       номера расположены в соответствии с l друг за другом,
//   p - массив длиной g * k * k ковариационных матриц, расположенных
//       друг за другом,
//   x - массив длиной g * k средних,
//   v - массив длиной g корней квадратных из определителей
//       ковариационных матриц.
// Возвращаемое значение:
//   0 при нормальном окончании расчета,
//  -1 при недостатке памяти,
//  -2 при потере значимости.
//
{
 short i,j,i1,      // Счетчики цикла
       m1,m2,m3,m4, // Индексы
       m5 = 0,      // Счетчик массива ковариационных матриц
       m6 = 0;      // Счетчик массива средних

 double *t,         // Групповая матрица исходных данных
        *s;         // Групповая дисперсионная матрица

 if ((s = new double[k * k]) == 0)
   return -1;

 ////////////////////////////////////
 // Построение дисперсионной матрицы
 for (i = 0, m3 = 0; i < g; i++)
   {
    if ((t = new double[l[i] * k]) == 0)
      {
       delete [] s;
       return -1;
      }
```

```
for (i1 = 0, m1 = 0, m4 = 0; i1 < k; i1++, m4 += n)
  for (j = 0, m2 = m3; j < l[i]; j++)
    t[m1++] = z[h[m2++] + m4 - 1];
m3 += l[i];

//////////////////////////
// Матрица групповых средних
for (i1 = 0, m2 = 0; i1 < k; i1++)
  {
   for (j = 0, x[m6] = 0; j < l[i]; j++)
     x[m6] += t[m2++];
   x[m6++] /= l[i];
  }

//////////////////////////////////////////////
// Групповая дисперсионно-ковариационная матрица
if (Likeness (t,s,l[i],k,1) != 0)
  {
   delete [] s; delete [] t;
   return -1;
  }
for (i1 = 0, m1 = 0; i1 < k; i1++)
  for (j = 0; j < k; j++)
    p[m5++] = s[m1++];

v[i] = Krauth (s,k);

delete [] t;

//////////////////////
// Проверка значимости
if (v[i] <= 0.000001)
  {
   delete [] s;
   return -2;
  }

v[i] = sqrt (v[i]);
```

```
   }

 delete [] s;

 return 0;
}

short __stdcall LDAClass (double x[],double u[],double d[],double
v[],short k,short g,short *c,double *p)
//
// Функция производит классификацию объекта по данным линейного
// дискриминантного анализа.
// Обозначения:
//   x - вектор исходных данных,
//   u - массив длиной g * k * k ковариационных матриц
//       (g матриц размером k * k),
//   d - массив длиной k * g средних,
//   v - массив длиной g корней квадратных из определителей
//       ковариационных матриц.
//   k - число параметров,
//   g - число классов,
//  *c - номер класса,
//  *p - соответствующая вероятность.
// Возвращаемое значение:
//   0 при нормальном окончании расчета,
//  -1 при недостатке памяти,
//  -2 при ошибке в вычислениях.
//
{
 short i,j,count, // Счетчики цикла
       Code,      // Код ошибки
       buf,       // Номер максимального элемента
       m1,        // Счетчик средних
       m2;        // Счетчик ковариационной матрицы

 double buf1,     // Рабочая переменная
        *s,       // Рабочий массив
        *u1,      // Групповая ковариационная матрица
        *u2,      // Матрица, обратная к ковариационной
```

```
       *z;       // Рабочий массив

 if ((s = new double[g]) == 0)
   return -1;
 if ((u1 = new double[k * k]) == 0)
   {
    delete [] s;
    return -1;
   }
 if ((u2 = new double[k * k]) == 0)
   {
    delete [] s; delete [] u1;
    return -1;
   }
 if ((z = new double[k]) == 0)
   {
    delete [] s; delete [] u1; delete [] u2;
    return -1;
   }

 ////////////////////////////////////////
 // Расчет массива плотностей для объекта
 for (i = 0, m1 = 0, m2 = 0; i < g; i++)
   {
    ////////////////////////
    // Ковариационная матрица
    for (j = 0; j < k * k; j++)
      u1[j] = u[m1++];

    ////////////////////////////////////
    // Матрица, обратная к ковариационной
    if ((Code = InverseMatrix (u1,u2,k)) != 0)
      {
       delete [] s; delete [] u1; delete [] u2; delete [] z;
       return Code;
      }

    ////////////////
```

```
    // Вектор средних
    for (j = 0; j < k; j++)
      z[j] = x[j] - d[m2++];

    MatrixMultiplyMatrix (u2,z,u1,k,k,1);
    MatrixTMultiplyMatrix (z,u1,u2,1,k,1);

    s[i] = exp (- 0.5 * u2[0]) / v[i];
   }

 buf1 = MaximumOf (s,g,&buf);
 *c = buf + 1;
 *p = buf1 / VectorSum (s,g);

 delete [] s; delete [] u1; delete [] u2; delete [] z;

 return 0;
}
```

На листинге 7.11 представлена интерфейсная функция к функции распознавания одного объекта. Данная функция составлена на языке VBA и предназначена для верификации введенных данных, вызова функции распознавания и вывода результатов распознавания.

Листинг 7.11

```
Sub ReconLinear(Data1 As Variant, Data2 As Variant, Data3 As Variant,
Data4 As Variant)
'
' Функция производит классификацию объекта по данным
' линейного дискриминантного анализа.
' Обозначения:
'  Data1 - интервал объекта
'  Data2 - ковариационные матрицы
'  Data3 - массив средних
'  Data4 - корни определителей
'
  Dim I As Integer          ' Счетчик
  Dim J As Integer          ' Счетчик
  Dim S As Integer          ' Счетчик
```

```
Dim K As Integer          ' Число параметров
Dim N As Integer          ' Рабочая переменная
Dim G As Integer          ' Число классов
Dim C As Integer          ' Номер наиболее вероятного класса
Dim P As Double           ' Вероятность принадлежности объекта классу
Dim X() As Double         ' Вектор исходных данных k
Dim V() As Double         ' Массив длиной g * k * k ковариационных
                          ' матриц
Dim W() As Double         ' Массив длиной k * g средних
Dim D() As Double         ' Вектор длиной g корней определителей
Dim Value As Variant      ' Рабочая переменная
Dim RData As Range        ' Вспомогательный объект
Dim Result As Integer     ' Код ошибки
Dim MsgTitle As String    ' Заголовок сообщения
Dim MsgTitleEr As String  ' Заголовок сообщения об ошибке
Dim MsgStyle As Integer   ' Стиль окна сообщения
Dim MsgStyleEr As Integer ' Стиль окна сообщения об ошибке
MsgTitle = "Результат распознавания"
MsgTitleEr = "Ошибка в исходных данных"
MsgStyle = vbOKOnly + vbInformation
MsgStyleEr = vbOKOnly + vbCritical
'
' Анализ исходных данных
'
On Error GoTo ErrHandler1
  Range(Data1).Select
On Error GoTo 0
K = 0
For Each Value In ActiveSheet.Range(Data1)
  If IsEmpty(Value) Then
    MsgBox "Пустая ячейка в области данных. Расчет невозможен.", _
           MsgStyleEr, MsgTitleEr
    Exit Sub
  End If
  If Not IsNumeric(Value) Then
    MsgBox "Нечисловой тип данных. Расчет невозможен.", _
           MsgStyleEr, MsgTitleEr
    Exit Sub
```

```
  End If
  K = K + 1
Next Value
On Error GoTo ErrHandler1
  Range(Data2).Select
On Error GoTo 0
For Each Value In ActiveSheet.Range(Data2)
  If IsEmpty(Value) Then
    MsgBox "Пустая ячейка в области данных. Расчет невозможен.", _
           MsgStyleEr, MsgTitleEr
    Exit Sub
  End If
  If Not IsNumeric(Value) Then
    MsgBox "Нечисловой тип данных. Расчет невозможен.", _
           MsgStyleEr, MsgTitleEr
    Exit Sub
  End If
Next Value
On Error GoTo ErrHandler1
  Range(Data3).Select
On Error GoTo 0
For Each Value In ActiveSheet.Range(Data3)
  If IsEmpty(Value) Then
    MsgBox "Пустая ячейка в области данных. Расчет невозможен.", _
           MsgStyleEr, MsgTitleEr
    Exit Sub
  End If
  If Not IsNumeric(Value) Then
    MsgBox "Нечисловой тип данных. Расчет невозможен.", _
           MsgStyleEr, MsgTitleEr
    Exit Sub
  End If
Next Value
On Error GoTo ErrHandler1
  Range(Data4).Select
On Error GoTo 0
G = 0
For Each Value In ActiveSheet.Range(Data4)
```

```
  If IsEmpty(Value) Then
    MsgBox "Пустая ячейка в области данных. Расчет невозможен.", _
           MsgStyleEr, MsgTitleEr
    Exit Sub
  End If
  If Not IsNumeric(Value) Then
    MsgBox "Нечисловой тип данных. Расчет невозможен.", _
           MsgStyleEr, MsgTitleEr
    Exit Sub
  End If
  G = G + 1
Next Value
If G < 2 Then
  MsgBox "Мало классов. Расчет невозможен.", MsgStyleEr, MsgTitleEr
  Exit Sub
End If
'
' Получение и верификация исходных данных
'
ReDim X(K)
I = 1
For Each Value In ActiveSheet.Range(Data1)
  X(I) = Value
  I = I + 1
Next Value
ReDim D(G)
I = 1
For Each Value In ActiveSheet.Range(Data4)
  D(I) = Value
  I = I + 1
Next Value
Set RData = Range(Data2)
N = RData.Rows.Count
If Not N = G * K Then
  MsgBox "Несоответствие числа параметров. Расчет невозможен.", _
         MsgStyleEr, MsgTitleEr
  Exit Sub
End If
```

```
N = RData.Columns.Count
If Not N = K Then
  MsgBox "Несоответствие числа параметров. Расчет невозможен.", _
         MsgStyleEr, MsgTitleEr
  Exit Sub
End If
ReDim V(G * K * K)
S = 1
For I = 1 To K
  For J = 1 To G * K
    V(S) = RData.Cells(J, I)
    S = S + 1
  Next J
Next I
Set RData = Range(Data3)
N = RData.Rows.Count
If Not N = K Then
  MsgBox "Несоответствие числа параметров. Расчет невозможен.", _
         MsgStyleEr, MsgTitleEr
  Exit Sub
End If
N = RData.Columns.Count
If Not N = G Then
  MsgBox "Несоответствие числа параметров. Расчет невозможен.", _
         MsgStyleEr, MsgTitleEr
  Exit Sub
End If
ReDim W(G * K)
S = 1
For I = 1 To G
  For J = 1 To K
    W(S) = RData.Cells(J, I)
    S = S + 1
  Next J
Next I
'
' Вычисление номера класса и соответствующей вероятности
'
```

```
  Result = LDAClass(X(1), V(1), W(1), D(1), K, G, C, P)
  If Result = -1 Then
    MsgBox "Недостаточно памяти.", MsgStyleEr, MsgTitleEr
    Exit Sub
  ElseIf Result = -2 Then
    MsgBox "Ошибка в вычислениях.", MsgStyleEr, MsgTitleEr
    Exit Sub
  Else
    MsgBox "Объект принадлежит классу " & C & Chr(13) & _
           "с вероятностью " & P, MsgStyle, MsgTitle
  End If
Exit Sub
ErrHandler1:
  MsgBox "Не определен интервал переменной. Расчет невозможен.",
MsgStyleEr, MsgTitleEr
  Exit Sub
End Sub
```

Линейный множественный регрессионный анализ

В ходе нетривиального эксперимента абсолютно точные измерения параметров, как правило, невозможны. Чтобы уменьшить влияние ошибок, производится большое число измерений. Каждое измерение дает нам уравнение с известной структурой, с точностью до коэффициентов, подлежащих определению. При числе измерений большем, чем число параметров, мы приходим к необходимости решения системы, число уравнений которой превышает число неизвестных параметров. В общем случае, полученный вектор решения не будет точно удовлетворять ни одному уравнению системы, однако будет наилучшим.

Для решения задачи в первом приближении предположим линейную зависимость результата эксперимента от параметров модели. Требуется решить систему линейных уравнений:

$$\sum_{j=1}^{m} a_{ij} x_j \approx b_i, i = 1,2,...,n,$$

где a_{ij} — элемент матрицы экспериментальных данных,

x_j — элемент подлежащего определению вектора весовых коэффициентов,

b_i — элемент вектора оценок результатов эксперимента,

m — количество измеряемых в эксперименте параметров,

n — количество опытов.

При m = n система в случае, если матрица системы не вырождена, имеет одно решение, поиск которого не вызывает трудностей и может быть осуществлен методом Гаусса или одним из итерационных методов.

При m > n система имеет бесчисленное множество решений. Число линейно независимых уравнений меньше количества неизвестных, поэтому возникает неопределенная (неопределенная) система линейных уравнений порядка *m*, в матричной записи имеющая вид

Ax = b.

Ранг n-матрицы системы A меньше порядка системы. Можно выделить одно решение, наложив на систему дополнительные условия, например:

$$\|b - Ax\|_2 = \min.$$

Размерность матрицы A суть n × m, причем n < m, как уже было отмечено выше.

После элементарных выкладок для матрицы, псевдообратной к A^T, получаем формулу решения, обладающего минимальной нормой, в виде

$$x^T = b^T \left(A^T\right)^+ .$$

Решение выполняет функция в листинге 7.12.

Листинг 7.12

```
short __stdcall Solves (double a[],double b[],short n,short m)
//
// Функция решения неопределенной системы линейных уравнений.
// Обозначения:
//  a - исходная матрица системы размером m x n, разрушается,
//  b - столбец свободных членов длиной n, на выходе - решение,
//      на входе используется только m <= n ячеек,
//  n - порядок системы (число неизвестных),
//  m - ранг матрицы системы <= n.
// Возвращаемое значение:
//   0 при нормальном окончании расчета,
//  -1 при недостатке памяти для вычислений,
//  -2 при потере значимости или ошибке в вычислениях.
//
{
 short Code;  // Возвращаемое значение

 double *t;   // Рабочий массив
```

```
 if ((t = new double[m * n]) == 0)
   return -1;

 MatrixT (a,t,m,n);

 if ((Code = PseudoInverseMatrix (t,a,n,m)) == 0)
   {
    MatrixTMultiplyMatrix (b,a,t,1,m,n);
    ArrayToArray (t,b,n);
   }

 delete [] t;

 return Code;
}
```

При m < n система является несовместной и может быть решена приближенно. В формуле знак ≈ применен вместо знака равенства, чтобы подчеркнуть неточность определения результата эксперимента. Процедура данного решения описана ниже.

Введем вектор ошибок $\varepsilon_i, i = 1,2,...,n$. Тогда исходную систему можно переписать:

$$\sum_{j=1}^{m} a_{ij} x_j + \varepsilon_i = b_i, i = 1,2,...,n.$$

Обычно применяют один из двух методов: метод, основанный на критерии минимакса, для чего минимизируют максимальную ошибку из $\varepsilon_i, i = 1,2,...,n$, либо метод наименьших квадратов, в котором минимизируют сумму квадратов ошибок $\varepsilon_i, i = 1,2,...,n$. Воспользуемся методом наименьших квадратов. Составим квадратичный функционал

$$I(x_1, x_2, ..., x_m) = \sum_{i=1}^{n} \varepsilon_i^2 = \sum_{i=1}^{n} \left[b_i - \sum_{j=1}^{m} a_{ij} x_j \right]^2 .$$

Потребуем минимума функционала $I = I(x_1, x_2, ..., x_m)$ по элементам вектора $x_j, j = 1,2,...,m$:

$$I(x_1, x_2, ..., x_m) \underset{x_j}{\rightarrow} \min, j = 1,2,...,m.$$

Требование означает равенство нулю всех частных производных функционала I по элементам $x_j, j = 1,2,...,m$, что приводит к системе m линейных алгебраических уравнений, решив которые мы найдем m неизвестных $x_j, j = 1,2,...,m$:

$$\frac{\partial I}{\partial x_k} = 0, k = 1,2,...,m.$$

Распишем подробнее. Подставив выражение для функционала, после элементарных преобразований получим:

$$\sum_{j=1}^{m}\left[\sum_{i=1}^{n} a_{ij}a_{ik}\right]x_j = \sum_{i=1}^{n} b_i a_{ik}, k = 1,2,...,m.$$

Это выражение является системой m уравнений для m неизвестных. Выражение в квадратных скобках представляет собой запись элемента матрицы системы уравнений, а правая часть формулы – элемента столбца свободных членов.

Лучшие результаты в смысле более точного описания реального объекта исследований часто можно получить, если использовать уравнения со свободным членом:

$$x_0 + \sum_{j=1}^{m} a_{ij}x_j + \varepsilon_i = b_i, i = 1,2,...,n.$$

Тогда остальные уравнения примут, соответственно, вид:

$$I(x_0, x_1, x_2,..., x_m) = \sum_{i=1}^{n} \varepsilon_i^2 = \sum_{i=1}^{n}\left[b_i - x_0 - \sum_{j=1}^{m} a_{ij}x_j\right]^2,$$

$$I(x_0, x_1, x_2,..., x_m) \underset{x_j}{\to} \min, j = 0,1,...,m,$$

$$\frac{\partial I}{\partial x_k} = 0, k = 0,1,...,m,$$

$$nx_0 + \sum_{j=1}^{m}\left[\sum_{i=1}^{n} a_{ij}\right]x_j = \sum_{i=1}^{n} b_i,$$

$$\sum_{i=1}^{n} a_{ik}x_0 + \sum_{j=1}^{m}\left[\sum_{i=1}^{n} a_{ij}a_{ik}\right]x_j = \sum_{i=1}^{n} b_i a_{ik}, k = 1,2,...,m.$$

Последнее выражение является системой $(m+1)$ уравнений для $(m+1)$ неизвестных. На практике, в применении последней формулы нет необходимости. При расчете достаточно ввести дополнительный $(m+1)$-й параметр, которому будет соответствовать число 1 в каждом векторе данных.

Листинг 7.13

```
short __stdcall LinearRegression (double t[],double z[],double
data[],short m,short n)
//
// Функция построения множественной линейной регрессионной модели.
// Обозначения:
//  z    - матрица системы размером m x n,
//  t    - вектор свободных членов длиной m,
//  data - вектор вычисленных коэффициентов длиной m,
//  m    - число параметров,
//  n    - число опытов.
// Возвращаемое значение:
//  0 - при нормальном окончании счета,
// -1 при недостатке памяти,
// -2 при потере значимости.
// Примечание: m <= n.
//
{
 short i,j,k,mm,mm1,mm2, // Счетчики
       Code;             // Код ошибки

 double *zz,             // Массив левых частей
        *bb;             // Массив правых частей

 if ((zz = new double[m * m]) == 0)
   return -1;
 if ((bb = new double[m]) == 0)
   {
    delete [] zz;
    return -1;
   }

 ////////////////////////////////////////////////////
```

```
 // Присвоение нулевых значений накапливаемым элементам
 FillUp (zz,m * m,0);
 FillUp (bb,m,0);

 ////////////////////////////////////////
 // Построение системы линейных уравнений
 for (k = 0, mm2 = 0; k < m; k++)
   {
    for (i = 0, mm = k; i < n; i++)
      {
       bb[k] += t[i] * z[mm];
       mm += m;
      }

    for (j = 0; j < m; j++)
      {
       for (i = 0, mm1 = j, mm = k; i < n; i++)
         {
          zz[mm2] += z[mm1] * z[mm];
          mm += m;
          mm1 += m;
         }
       mm2++;
      }
   }

 ////////////////////////////////////////
 // Решение системы линейных уравнений
 if ((Code = Gauss (bb,zz,m,1,0.0001)) == 0)
   for (i = 0; i < m; i++)
     data[i] = bb[i];

 delete [] zz; delete [] bb;

 return Code;
}

short __stdcall LinearRegression0 (double data[],double z[],double
t[],short numbers,short m)
```

```
//
// Функция построения множественной линейной регрессионной модели со
// свободным членом.
// Обозначения:
//   z       - матрица системы размером numbers x m,
//   data    - вектор свободных членов длиной m,
//   t       - вектор решения длиной m + 1,
//   m       - число опытов,
//   numbers - число параметров.
// Возвращаемое значение:
//   0 при нормальном окончании счета,
//  -1 при недостатке памяти,
//  -2 при потере значимости.
// Примечание: m <= n.
//
{
 short i,count,mm1,mm2, // Счетчики
       Code;            // Код ошибки

 double *zz;            // Массив левых частей

 if ((zz = new double[(numbers + 1) * m]) == 0)
   return -1;

 ////////////////////////////////////////
 // Преобразовать матрицу исходных данных
 for (i = 0, mm1 = 0; i < m; i++, mm1 += numbers + 1)
   zz[mm1] = 1;     // Единичный вектор - свободный член
 for (i = 0; i < numbers; i++)
   for (count = 0, mm1 = i + 1, mm2 = i; count < m; count++)
     {
      zz[mm1] = z[mm2];
      mm1 += numbers + 1;
      mm2 += numbers;
     }

 Code = LinearRegression (data,zz,t,numbers + 1,m);
```

```
 delete [] zz;

 return Code;
}
```

На листинге 7.14 представлен интерфейс к функции распознавания одного объекта и сама функция распознавания. Она выполнена в VBA и предназначена для верификации введенных данных и вывода результатов распознавания.

Листинг 7.14

```
Sub ReconMulti(Data1 As Variant, Data2 As Variant)
'
' Функция производит распознавание объекта по данным
' множественного линейного регрессионного анализа.
' Обозначения:
'  Data1 - интервал объекта
'  Data2 - интервал коэффициентов регрессии
'
  Dim I As Integer          ' Счетчик
  Dim K As Integer          ' Число параметров
  Dim N As Integer          ' Число коэффициентов регрессии
  Dim C As Double           ' Оценка
  Dim X() As Double         ' Вектор исходных данных длиной k
  Dim W() As Double         ' Вектор коэффициентов регрессии длиной
                            ' k или k + 1
  Dim Value As Variant      ' Рабочая переменная
  Dim MsgTitle As String    ' Заголовок сообщения
  Dim MsgTitleEr As String  ' Заголовок сообщения об ошибке
  Dim MsgStyle As Integer   ' Стиль окна сообщения
  Dim MsgStyleEr As Integer ' Стиль окна сообщения об ошибке
  MsgTitle = "Результат распознавания"
  MsgTitleEr = "Ошибка в исходных данных"
  MsgStyle = vbOKOnly + vbInformation
  MsgStyleEr = vbOKOnly + vbCritical
  '
  ' Анализ исходных данных
  '
```

```
  On Error GoTo ErrHandler1
    Range(Data1).Select
  On Error GoTo 0
  K = 0
  For Each Value In ActiveSheet.Range(Data1)
    If IsEmpty(Value) Then
      MsgBox "Пустая ячейка в области данных. Расчет невозможен.",
MsgStyleEr, MsgTitleEr
      Exit Sub
    End If
    If Not IsNumeric(Value) Then
      MsgBox "Нечисловой тип данных. Расчет невозможен.", MsgStyleEr,
MsgTitleEr
      Exit Sub
    End If
    K = K + 1
  Next Value
  On Error GoTo ErrHandler1
    Range(Data2).Select
  On Error GoTo 0
  N = 0
  For Each Value In ActiveSheet.Range(Data2)
    If IsEmpty(Value) Then
      MsgBox "Пустая ячейка в области данных. Расчет невозможен.",
MsgStyleEr, MsgTitleEr
      Exit Sub
    End If
    If Not IsNumeric(Value) Then
      MsgBox "Нечисловой тип данных. Расчет невозможен.", MsgStyleEr,
MsgTitleEr
      Exit Sub
    End If
    N = N + 1
  Next Value
  '
  ' Получение и верификация исходных данных
  '
  If N < K Then
    MsgBox "Мало коэффициентов. Расчет невозможен.", MsgStyleEr,
MsgTitleEr
```

```
    Exit Sub
  End If
  If N > K + 1 Then
    MsgBox "Много коэффициентов. Расчет невозможен.", MsgStyleEr,
MsgTitleEr
    Exit Sub
  End If
  ReDim X(K)
  ReDim W(K + 1)
  I = 1
  For Each Value In ActiveSheet.Range(Data1)
    X(I) = Value
    I = I + 1
  Next Value
  I = 1
  For Each Value In ActiveSheet.Range(Data2)
    W(I) = Value
    I = I + 1
  Next Value
  '
  ' Распознавание
  '
  MsgBox "Оценка равна " & MultiRecon(X(), N, W(), K), MsgStyle, MsgTitle
Exit Sub
ErrHandler1:
  MsgBox "Не определен интервал переменной. Расчет невозможен.",
MsgStyleEr, MsgTitleEr
  Exit Sub
End Sub

Private Function MultiRecon(X() As Double, N As Integer, W() As Double, K
As Integer) As Double
'
' Функция производит распознавание объекта по данным
' множественного линейного регрессионного анализа.
' Обозначения:
'  X - параметры объекта
'  N - число параметров
'  W - коэффициенты регрессии
```

```
'  K - число коэффициентов
' Возвращаемое значение:
'  вычисленное значение оценки.
'
 Dim I As Integer ' Счетчик

 If N = K Then
   MultiRecon = 0
   For I = 1 To N
     MultiRecon = MultiRecon + X(I) * W(I)
   Next I
 Else
   MultiRecon = W(1)
   For I = 1 To N
     MultiRecon = MultiRecon + X(I) * W(I + 1)
   Next I
 End If
End Function
```

Руководство пользователя

Для работы с модулем распознавания образов с обучением (Pr) выберите из меню Microsoft Excel пункт **AtteStat ▸ Модуль Pr ▸ Распознавание образов с обучением**. На экране появится диалоговое окно, изображенное на рис. 7.8. Оно содержит 5 закладок, что вызвано спецификой решаемой задачи и обеспечивает как обучение на основе обучающей выборки, так распознавание вновь введенных объектов всеми применяемыми методами.

Рис. 7.8. Окно модуля распознавания образов с обучением, закладка «Обучение»

Для решения задачи обучения:

1. выберите или введите интервалы обучающей выборки. Для определенности объекты располагаются по вертикали (в строках), а параметры по горизонтали (в столбцах). Например, один объект, описываемый 4 параметрами, занимает 4 ячейки строки электронной таблицы;
2. выберите или введите интервалы номеров классов (для методов дискриминантного анализа) или оценок (для множественной регрессии). Численность данного массива должна равняться числу объектов и не содержать пропусков. Например, если имеется в наличии 6 классов, то каждый объект будет принадлежать одному из 6-ти классов: 1, 2, 3, 4, 5 или 6. Ситуация, когда какой-либо класс не содержит объектов (например, класс 2 не присутствует в списке) недопустима;
3. выберите или введите интервал вывода результатов. Начиная с первой ячейки выходного интервала, будут выведены результаты вычислений;
4. выберите или оставьте по умолчанию метод анализа;
5. нажмите кнопку **Расчет**.

После выполнения вычислений будет, начиная с первой ячейки выходного интервала, выведено название метода и результаты обучения (см. *рис.7.9*).

Рис. 7.9. Окно модуля распознавания образов с обучением, закладка «Распознавание» методом линейного дискриминантного анализа по Фишеру

Для решения задачи распознавания:

1. выберите или введите интервал параметров объекта. Объект описывается своими параметрами, число которых должно быть тем же, что и при обучении;
2. выберите или введите интервалы простых классифицирующих функций, вычисленных на этапе обучения;
3. нажмите кнопку **Расчет**.

После выполнения расчета будет выдано информационное окно, содержащее наиболее вероятный номер класса, которому принадлежит объект, и вероятность принадлежности объекта этому классу (см. *рис. 7.10*).

Модуль Pr берет на себя верификацию исходных данных, включая проверку соответствия всех параметров друг другу, выдавая подробную диагностику. При неверных действиях пользователя выдаются сообщения об ошибках.

Рис. 7.10. Окно модуля распознавания образов с обучением, закладка «Распознавание» методом линейного канонического дискриминантного анализа

Сообщения об ошибках

При ошибках ввода данных могут выдаваться диагностические сообщения следующих указанных типов.

Рис. 7.11. Окно сообщения об ошибке

На рис. 7.11 не выбран или неверно введен входной интервал. Лучшим способом избежать ошибки является не ввод, а выделение интервала протаскиванием курсора.

Рис. 7.12. Окно сообщения об ошибке

Проверьте исходные данные на рис. 7.12 и заполните все ячейки входного интервала. Если в ячейке не должно быть данных, по физической природе исследуемого процесса, введите нуль.

Рис. 7.13. Окно сообщения об ошибке

Проверьте типы ячеек входного интервала На рис. 7.13. Тип может быть только числовым. Проще всего выделить интервал ячеек и явно определить их тип как числовой средствами Microsoft Excel.

Рис. 7.14. Окно сообщения об ошибке

На рис.7.14 не выбран или неверно введен выходной интервал. Лучшим способом избежать ошибки является не ввод, а выделение интервала стандартным образом.

Рис. 7.15. Окно сообщения об ошибке

Число объектов обучающей выборки на рис.7.15 не соответствует численности массива классов.

Рис. 7.16. Окно сообщения об ошибке

Номера классов на рис. 7.16 содержат один или несколько пропущенных номеров. Введите номера классов без пропусков, начиная с 1. Например, классы могут быть следующие: 1, 2, 3, 4, 5. Но не 1, 2, 3, 5, 6. В последнем случае пропущен класс номер 4. Иначе — класс номер 4 не содержит объектов.

Рис. 7.17. Окно сообщения об ошибке

Число параметров распознаваемого объекта на рис.7.17 не соответствует параметрам распознавания, вычисленным на этапе обучения.. Кроме того, все массивы, вычисленные при обучении, должны быть выбраны точно. Для минимизации ошибок при выборе интервалов ячеек электронной таблицы рекомендуется метод протаскивания курсора.

При решении тестовых задач вычислительный модуль показал стабильность.

Пример применения

Для примера применения рассмотренных методов распознавания образов введем исходные данные, взятые из работы: Уткин В.А., Скляр Е.И. Классификация и диагностика с участием и без участия врача // Материалы XXVI научно-практической конференции врачей Курганской области, г. Курган, 1993, с. 28-35. Читатели могут обратиться к упомянутой работе и получить оттуда все числовые материал для расчета (см. *рис. 7.18*).

Рис. 7.18. Исходные данные и результаты расчета

Канонический дискриминантный анализ показал качество распознавания 77,27273%, линейный дисриминантный анализ по Фишеру — также 77,27273%. Линейный дисриминантный анализ выполнить не удалось в связи с особенностями алгоритма.

Точность распознавания может быть повышена увеличением численности обучающей группы и более точным выбором номенклатуры параметров для распознавания. Это, конечно же, потребует дополнительных исследований.

Однако уже в данном виде реализация рассмотренных алгоритмов на базе Microsoft Excel позволяет применять представленные математические алгоритмы не только в исследовательских целях, но и в практической работе.

Рекомендуемая литература

1. Гайдышев И. Анализ и обработка данных: специальный справочник. — СПб: Питер, 2001.
2. Гантмахер Ф.Р. Теория матриц. — М.: Наука, 1988.
3. Голуб Дж., Ван Лоун Ч. Матричные вычисления. — М.: Мир, 1999.
4. Деммель Дж. Вычислительная линейная алгебра. Теория и приложения. — М.: Мир, 2001.
5. Додж М., Кината К., Стинсон К. Эффективная работа с Excel 7.0 для Windows 95. — СПб: Питер, 1996.
6. Дюк В. Обработка данных на ПК в примерах. — СПб: Питер, 1997.
7. Дюран Б., Оделл П. Кластерный анализ. — М.: Финансы и статистика, 1977.

8. Жамбю М. Иерархический кластер–анализ и соответствия. — М.: Финансы и статистика, 1988.
9. Классификация и кластер. — М.: Мир, 1980.
10. Кульбак С. Теория информации и статистика. — М.: Наука, 1967.
11. Мандель И.Д. Кластерный анализ. — М.: Финансы и статистика, 1988.
12. Сборник научных программ на Фортране. Вып. 1. Статистика. — М.: Статистика, 1974.
13. Статистические методы для ЭВМ / Под ред. К. Энслейна, Э. Рэлстона, Г.С. Уилфа. — М.: Наука, 1986.
14. Уилкинсон Дж. Х., Алгебраическая проблема собственных значений. — М.: Наука, 1970.
15. Уилкинсон, Райнш. Справочник алгоритмов на языке АЛГОЛ. Линейная алгебра. — М.: Машиностроение, 1976.
16. Уткин В.А. Статистические технологии в медицинских исследованиях: Монография. — Пятигорск: ГНИИК, 2002.
17. Факторный, дискриминантный и кластерный анализ / Дж.О. Ким, Ч.У. Мюллер, У.Р. Клекка и др. — М.: Финансы и статистика, 1989.

ГЛАВА 8

Корреляционный анализ в кодах VBA и C

«Знание некоторых принципов нередко возмещает незнание некоторых фактов».
Гельвеций

Для работы с материалами главы понадобятся лицензионные средства разработки и тестирования программ:

- Microsoft Excel 97 или выше;
- Microsoft Visual C++ 5.0 или выше;
- Borland C++ 5.5 или выше.

Потребуется знание VBA, C.

Решаемая проблема

В практических наблюдениях часто бывают случаи, когда зависимости не имеют функционального характера: равномерному изменению одного признака соответствует изменение величины другого признака в среднем. Такой вид соотношений называется корреляционной зависимостью, или корреляцией. Корреляционным анализом называется совокупность методов обнаружения корреляционной зависимости между случайными величинами или признаками. Считается, что исследование взаимной зависимости приводит к теории корреляции, тогда как изучение зависимости ведет к теории регрессии.

Выделяется также случай функциональной зависимости между величинами, измерения которых, возможно, подвержены ошибкам наблюдений или измерений. Функциональная зависимость может описываться методами математического моделирования. Наличие статистической корреляционной зависимости не влечет зависимости причинной, поэтому статистическое моделирование не является моделированием математическим, а статистическую модель нельзя называть математической моделью[1]. О математическом моделировании см. *главу* 9.

Особенности решения

Регистрация функций DLL выполнена в модуле VBA, так как результатом работы методов корреляционного анализа являются массивы чисел. Их нельзя передать с помощью макросов, поэтому регистрация экспортируемых DLL функций VBA-оператором `Declare` была удобным решением.

Основное отличие методов корреляционного анализа в оперировании:

- двумя выборками исходных данных;
- двумя матрицами, состоящими из групп выборок исходных данных;
- одной матрицей (группой выборок) исходных данных.

Одни и те же поля и методы ввода данных имеют различный смысл, в зависимости от применяемого анализа. При этом информация, переданная из ячеек электронной таблицы, извлекается в том виде, который необходим для данного метода анализа, с помощью одной и той же функции.

Научные алгоритмы

Листинги основных функций, реализующих методы расчета, приведены.

Рассмотрены методы исследования корреляции, когда оба признака измерены в одной и той же шкале — количественной, порядковой или номинальной.

Использование коэффициентов корреляции для построения мер связи в кластерном анализе рассмотрено в *главе 7*.

Корреляция количественных признаков

В данном разделе представлены методы исследования корреляции количественных признаков: коэффициент корреляционного отношения Пир-

[1] *Обратное неверно. Математическая модель может включать в себя стохастические элементы.*

сона, применяемый для измерения тесноты связи при прямолинейной корреляции, и коэффициент корреляции Фехнера. Коэффициенты ранговой корреляции, которые исследуют корреляцию порядковых признаков (рангов), полученных из признаков количественных (путем применения операции присвоения рангов), помещены в раздел «Корреляция порядковых признаков».

Полученные в результате применения линейных методов корреляционного анализа выводы могут подтвердить или опровергнуть гипотезу о существовании линейной зависимости между рядами, но не связи другого типа. Вывод в этом случае такой: чем ближе вычисленная величина корреляционного отношения к нулю, тем слабее сила линейной связи между рядами, чем ближе вычисленная величина к значению +1 (полная положительная корреляция) или к значению −1 (полная отрицательная корреляция), тем сильнее сила линейной связи.

Коэффициент корреляционного отношения Пирсона

Коэффициент корреляционного отношения Пирсона (коэффициент корреляции, выборочный коэффициент корреляции, коэффициент корреляции Бравайса − Пирсона) измеряет силу линейной корреляционной связи количественных признаков. Выборочный коэффициент корреляции вычисляется по известной формуле

$$r = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2}\sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2}},$$

где n — численность каждой выборки,

$x_i, i = 1,2,...,n$ — первая выборка, $\bar{x}$ − соответствующее среднее значение,

$y_i, i = 1,2,...,n$ — вторая выборка, $\bar{y}$ − соответствующее среднее значение.

Использование коэффициента корреляции оправдано лишь тогда, когда совместное распределение пары количественных признаков нормально или приближенно нормально. Частой ошибкой в публикациях является игнорирование этого требования, поэтому перед применением алгоритма вычисления коэффициента Пирсона следует проверить нормальность двумерной выборки (см. *главу* 3).

При исследовании многомерной совокупности случайных величин из коэффициентов корреляции, вычисленных попарно между случайными величинами, составляется квадратная симметрическая корреляционная матрица с единицами на главной диагонали, которая служит основным элементом

при построении многих алгоритмов многомерной статистики, например, в факторном анализе (см. *главу* 6).

Доверительный интервал выборочной оценки коэффициента корреляции нормальной двумерной генеральной совокупности вычисляется как

$$r \in \left[\tanh\left(z(r) - \frac{N_{(1+p)/2}}{\sqrt{n-3}} \right); \tanh\left(z(r) + \frac{N_{(1+p)/2}}{\sqrt{n-3}} \right) \right],$$

где $N_{(1+p)/2}$ — квантиль нормального распределения,

p — стандартное значение доверительного уровня,

$z(r)$ — z-преобразование выборочного коэффициента корреляции.

Листинг 8.1

```
Private Sub Interval(Z As Double, N As Integer, P As Double, Low As
Double, High As Double)
'
' Функция вычисляет доверительный интервал коэффициента корреляции.
' Обозначения:
'     z — преобразование Фишера,
'     n — размерность каждой из выборок,
'     p — доверительный уровень,
'   Low — нижняя граница,
'  High — верхняя граница.
' Возвращаемое значение:
'  нет.
'
 Dim B As Double ' Вспомогательная переменная
 B = InverseNormalDistribution((1 + P) / 2) / Sqr(N — 3)
 Low = Tanh(Z — B)
 High = Tanh(Z + B)
End Sub


Private Function Tanh(X As Double) As Double
'
' Функция вычисляет значение гиперболического тангенса.
```

```
' Обозначения:
'  P — параметр.
' Возвращаемое значение:
'  значение гиперболического тангенса.
'
 Tanh = (Exp(X) — Exp(-X)) / (Exp(X) + Exp(-X))
End Function
```

Нормализующее z-преобразование выборочного коэффициента корреляции вычисляется как гиперболический арктангенс по формуле

$$z(r) = Arth(r) = \frac{1}{2}\ln\frac{1+r}{1-r} \text{ при } |r| < 1.$$

Листинг 8.2

```
Private Function Atanh(R As Double) As Double
'
' Функция вычисляет значение гиперболического арктангенса.
' Обозначения:
'  R — параметр.
' Возвращаемое значение:
'  значение гиперболического арктангенса.
'
  Atanh = Log((1 + R) / (1 — R)) / 2
End Function
```

Коэффициент корреляции может применяться для проверки гипотезы независимости признаков (некоррелированности, значимости связи) следующим образом. Гипотеза о независимости признаков отвергается на выбранном уровне значимости, если вычисленное по опытным данным значение коэффициента корреляции превосходит (по модулю) критическое. В случае нормального распределения исходных данных величина выборочного коэффициента корреляции считается значимо отличной от нуля, если выполняется неравенство

$$r^2 > \left[1 + (n-2)/t_\alpha^2\right]^{-1},$$

где t_α — критическое значение t-распределения с $(n - 2)$ степенями свободы (см. *приложение 2*).

Иначе говоря, величина

$$t_r = r\sqrt{\frac{n-2}{1-r^2}}$$

имеет t-распределение с $(n - 2)$ степенями свободы.

Кроме того, распределение величины $z(r)$ уже при небольших значениях n приближается нормальным распределением с математическим ожиданием, равным

$$Mz = \frac{1}{2}\ln\frac{1+r}{1-r} + \frac{r}{2(n-1)} + \ldots$$

и дисперсией

$$Dz = \frac{1}{n-3} + \ldots,$$

где опущены слагаемые, малые по сравнению с оставленными слагаемыми.

Таким образом, случайная величина

$$\frac{z(r) - Mz}{\sqrt{Dz}}$$

распределена приближенно по стандартному нормальному закону $N(0,1)$.

Коэффициент корреляции Фехнера

Кратко опишем коэффициент корреляции Фехнера (фехнеровский коэффициент корреляции), который был предложен для изучения корреляции количественных признаков. При вычислении коэффициента происходит понижение количественной шкалы до номинальной шкалы. В расчетах участвуют только количественные признаки (по ним вычисляются средние значения), поэтому метод представлен в разделе, посвященном количественным признакам. Вычисления производятся по формуле

$$r_F = \frac{C - H}{C + H},$$

где C — число совпадений знаков отклонений вариант от соответствующих средних значений,

H — число несовпадающих знаков.

Коэффициент корреляции Фехнера с успехом применяется также и для изучения корреляции только номинальных признаков. В этом случае, в соответствующих обозначениях, приведенная формула может быть записана как

$$r_F = \frac{a+d-b-c}{a+b+c+d},$$

где a — число наблюдений с эффектом А в первой выборке,

b — число наблюдений без эффекта А в первой выборке,

c — число наблюдений с эффектом А во второй выборке,

d — число наблюдений без эффекта А во второй выборке.

Коэффициент корреляции Фехнера может применяться для проверки гипотезы независимости признаков (некоррелированности, значимости связи). В этом случае можно произвести вычисление по формуле

$$t_F = \frac{a+d-b-c-1}{\sqrt{a+b+c+d}} = \frac{r_F n - 1}{\sqrt{n}},$$

где $n = a + b + c + d$.

Критические значения статистики t_F приближенно распределены по стандартному нормальному закону $N(0,1)$.

Корреляция порядковых признаков

В данном разделе рассмотрены методы исследования корреляции признаков, измеренных в порядковой шкале, либо приведенных к ней.

Показатель ранговой корреляции Спирмэна

Показатель ранговой корреляции Спирмэна (показатель корреляции рангов Спирмэна, коэффициент корреляции рангов, коэффициент корреляции Спирмэна, коэффициент ранговой корреляции ρ, Spearman rank correlation) применяется в случае, если изучается линейная связь между рядами. При анализе количественных признаков применять показатель Спирмэна вместо коэффициента Пирсона не следует, так как при его вычислении происходит понижение количественной шкалы до порядковой. При анализе корреляции порядковых признаков расчет ведется по формуле

$$\rho_S = 1 - \frac{6(S_\rho + B_x + B_y)}{n^3 - n}; S_\rho = \sum_{i=1}^{n} (r_i - s_i)^2 ,$$

где $r_i, s_i, i = 1,2,...,n$ — массивы рангов,

n — число пар вариант исследуемых рядов,

B_x, B_y — поправки на объединение рангов в соответствующих рядах, вычисляемые по формуле

$$B_{\cdot} = \frac{1}{12} \sum_{i=1}^{m} n_i (n_i^2 - 1) ,$$

где m — число групп объединенных рангов в ряду,

$n_i, i = 1,2,...,m$ — число рангов в i–ой группе.

Показатель ранговой корреляции Спирмэна может применяться для проверки гипотезы независимости признаков. Гипотеза о независимости признаков отвергается на выбранном уровне значимости, если вычисленное по опытным данным значение коэффициента корреляции превосходит (по модулю) критическое. Можно также произвести вычисление по формуле

$$t_\rho = \rho_S \sqrt{\frac{n-2}{1-\rho_S^2}} ,$$

где критические значения статистики t_ρ имеют t-распределение с (n — 2) степенями свободы.

Для больших выборок (n > 50) статистика $\sqrt{n-1}\rho_S$ распределена приближенно по стандартному нормальному закону $N(0,1)$.

Коэффициент ранговой корреляции Кендалла

Коэффициент ранговой корреляции Кендалла (коэффициент корреляции рангов, ранговый коэффициент корреляции, коэффициент корреляции Кендэла, Kendall rank correlation) предназначен для вычисления силы корреляционной связи между двумя рядами при тех же условиях, что показатель Спирмэна, он считается более строгой оценкой.

Расчет ведется по формуле

$$\tau = \frac{S_\tau}{\sqrt{\left(\frac{n(n-1)}{2} - B_x\right)\left(\frac{n(n-1)}{2} - B_y\right)}} , S_\tau = \sum_{i=1}^{n} \sum_{j=i+1}^{n} sign(r_j - s_i) ,$$

где $r_i, s_i, i = 1,2,...,n$ — массивы рангов анализируемых рядов,

n — число пар вариант исследуемых рядов,

B_x, B_y — поправки на объединение рангов в соответствующих рядах, вычисляемые по формуле

$$B_{\cdot} = \frac{1}{2}\sum_{i=1}^{m} n_i(n_i - 1),$$

где m — число групп объединенных рангов в ряду,

$n_i, i = 1,2,...,m$ — число рангов в i–ой группе.

Коэффициент ранговой корреляции Кендалла может применяться для проверки гипотезы независимости признаков. Гипотеза о независимости признаков отвергается на выбранном уровне значимости, если вычисленное по опытным данным значение коэффициента корреляции превосходит (по модулю) критическое. В случае больших выборок можно произвести вычисление по формуле

$$t_\tau = \tau\sqrt{\frac{9n(n-1)}{2(2n+5)}},$$

где критические значения статистики t_τ приближенно распределены по стандартному нормальному закону $N(0,1)$.

Корреляция номинальных признаков

В данном разделе представлены методы исследования корреляции признаков, измеренных в номинальной шкале либо приведенных к номинальной шкале.

Рассмотрены коэффициенты (показатели подобия) Рассела-Рао и Бравайса, предназначенные для оценки связи между дихотомическими (номинальными, качественными) признаками. Эти и другие показатели находят широкое применение в кластерном анализе (см. *главу 7*), где они именуются также мерами сходства *типа корреляции*.

Таблица 2×2 (при исследовании показателей подобия называемая также таблицей ассоциативности) строится по тому же принципу, что и таблица сопряженности для номинальных признаков, рассмотренная в разделе «Точный метод Фишера – Ирвина» из *главы 3*.

Значение варианты выборки, равное нулю, указывает на отсутствие переменной или признака, значение, равное единице — на наличие. Например, в ячейку a записано число пар элементов двух массивов, одно-

временно имеющих признак, равный единице. В ячейку c записано число пар элементов тех же массивов, в которых значение элемента первого массива равно единице, а значение элемента второго равно нулю и т. д.

Коэффициент Рассела-Рао

Коэффициент Рассела-Рао (показатель подобия Рассела-Рао) вычисляется по формуле

$$t_R = \frac{a}{a+b+c+d},$$

где a — число наблюдений с эффектом А в первой выборке,

b — число наблюдений без эффекта А в первой выборке,

c — число наблюдений с эффектом А во второй выборке,

d — число наблюдений без эффекта А во второй выборке.

Коэффициент Рассела-Рао может применяться для проверки гипотезы независимости признаков. В этом случае можно произвести вычисление по формуле

$$t_R = \frac{a-1}{\sqrt{a+b+c+d}} = \frac{t_R n - 1}{\sqrt{n}},$$

где критические значения статистики t_R приближенно распределены по стандартному нормальному закону $N(0,1)$.

Коэффициент сопряженности Бравайса

Специальная форма коэффициента корреляции — коэффициент сопряженности Бравайса (φ-коэффициент Пирсона, коэффициент контингенции Пирсона) — рассчитывается по формуле

$$\varphi = \frac{ad - bc}{\sqrt{(a+b)(a+c)(d+b)(d+c)}},$$

где a — число наблюдений с эффектом А в первой выборке,

b — число наблюдений без эффекта А в первой выборке,

c — число наблюдений с эффектом А во второй выборке,

d — число наблюдений без эффекта А во второй выборке.

Коэффициент сопряженности Бравайса может применяться для проверки гипотезы независимости признаков. Гипотеза о независимости признаков отвергается на выбранном уровне значимости, если вычисленное по опытным данным значение коэффициента корреляции превосходит (по модулю) критическое. Критические значения коэффициента имеют t-распределение с $(n - 2)$ степенями свободы, где

$$n = a + b + c + d.$$

Канонический корреляционный анализ

Канонический корреляционный анализ выполняется между двумя совокупностями (группами) выборок и предназначен для определения линейной функции от первых p компонент и линейной функции от остальных q компонент так, чтобы коэффициент корреляции между этими линейными функциями принял наибольшее из возможных значений. Численности групп (количество выборок в первой и второй группах, p и q) могут различаться, однако необходимым требованием является равное количество вариант во всех выборках, составляющих обе группы. Матрица взаимной корреляции двух групп выборок имеет вид

$$R = \begin{bmatrix} R_{11} & R_{12} \\ R_{12}^T & R_{22} \end{bmatrix},$$

где R_{11} — матрица взаимной корреляции p переменных 1–й группы, размер p x p,

R_{22} — матрица взаимной корреляции q переменных 2–й группы, размер q x q,

R_{12} — матрица взаимной корреляции переменных 1–й и 2–й группы, размер p x q.

Решение задачи сводится к обобщенной проблеме собственных значений

$$R_{12}^T R_{11}^{-1} R_{12} \cdot v = \lambda \cdot R_{22} \cdot v\ ,$$

где λ — вектор q собственных значений.

Так называемые канонические корреляции представляют собой квадратные корни из собственных значений. Представленной функцией выводятся: значения критерия χ^2 (массив длиной q) и соответствующие степени свободы (массив длиной q), а также коэффициенты правой (массив размером $q \times q$) и левой (массив размером $q \times p$) стороны.

Руководство пользователя

Для работы с модулем корреляционного анализа (Cora) выберите из меню Microsoft Excel пункт **AtteStat ▸ Модуль Cora ▸ Корреляционный анализ**. На экране появится диалоговое окно, изображенное на рис. 8.1. Выбор многочисленных вариантов сочетаний коэффициентов корреляции и методов анализа придаёт дополнительную гибкость программе, отсутствующую в других программных продуктах анализа данных.

Рис. 8.1. Окно модуля корреляционного анализа

1. Выберите или введите интервалы вектора или матрицы исходных данных (выборки 1).
2. Выберите или введите интервалы вектора или матрицы исходных данных (выборки 2) для методов «Исследование корреляции» и «Канонический анализ».
3. Выберите или введите выходной интервал. Начиная с первой ячейки выходного интервала, будут выведены результаты вычислений.
4. Выберите или оставьте по умолчанию коэффициент корреляции, в зависимости от характера исходных данных. Модуль Cora сам не определяет тип исходных данных, поэтому пользователь должен сам выбрать коэффициент корреляции, адекватный исходным данным. Обратите внимание, что для качественных (дихотомических) признаков предусмотрены

только два значения вариант выборок: 0 при отсутствии признака, 1 при наличии признака.

5. Выберите или оставьте по умолчанию метод анализа.
6. Нажмите кнопку **Выполнить расчет**.

После выполнения вычислений будет, начиная с первой ячейки выходного интервала, выведено название метода и результаты расчета.

Модуль Cora берет на себя верификацию исходных данных, выдавая подробную диагностику. При неверных действиях пользователя выдаются сообщения об ошибках.

Сообщения об ошибках

При ошибках ввода данных могут выдаваться диагностические сообщения следующих указанных типов.

Рис. 8.2. Окно сообщения об ошибке

На рис.8.2 не выбрн или неверно введен входной интервал. Лучшим способом избежать ошибки является не ввод, а выделение интервала протаскиванием курсора. Ошибка может быть вызвана также тем, что для метода «Исследование корреляции» требуется указание двух выборок, а для метода «Канонический анализ» требуется указание двух матриц исходных данных.

Рис. 8.3. Окно сообщения об ошибке

На рис.8.3 проверьте исходные данные и заполните все ячейки входного интервала. Если в ячейке не должно быть данных, введите нуль.

Рис. 8.4. Окно сообщения об ошибке

Проверьте типы ячеек входного интервала на рис.8.4. Тип может быть только числовым. Проще всего выделить интервал ячеек и явно определить их тип как числовой средствами Microsoft Excel.

Рис. 8.5. Окно сообщения об ошибке

Методы «Исследование корреляции» и «Канонический анализ» оперируют, соответственно, векторами или матрицами данных. Вектора или матрицы состоят из выборок, содержащих равные количества наблюдений. Убедитесь, что все выборки исходных данных на рис.8.5 содержат равные количества наблюдений.

Рис. 8.6. Окно сообщения об ошибке

На рис.8.6 не выбран или неверно введен выходной интервал. Лучшим способом избежать ошибки является не ввод, а выделение интервала стандартным образом.

Рис. 8.7. Окно сообщения об ошибке

Аналогичное сообщение может быть выдано и для группы 2 на рис.8.7. Для метода «Канонический анализ» число выборок в матрице данных не должно быть менее двух. Если это не так, используйте метод «Исследование корреляции» для признаков соответствующего типа.

Рис. 8.8. Окно сообщения об ошибке

Сообщение на рис.8.8 выдается в случае, если число вариант (наблюдений) анализируемой исходной выборки меньше двух. Для применения любых методов корреляционного анализа число вариант выборки не должно быть менее двух. Максимальные численности выборок ограничены только требованиями Microsoft Excel.

При решении тестовых вычислительный модуль показал стабильность.

Дальнейшие исследования

Дополнение вычислительного модуля Cora возможностями анализа корреляции признаков, измеренных в различных шкалах.

Пример практического применения

Для примера рассмотрим, как можно применить алгоритмы, рассмотренные в данной главе, при анализе корректности построения калибровочного графика стандартного раствора неорганического фосфата. Данный график применяется для определения концентрации неорганического фосфата по оптической плотности раствора (данные предоставлены А.И. Гайдышевым, работа выполнена под руководством А.Н. Накоскина).

Для решения задачи введем в ячейки рабочего листа две выборки. В данном случае неважно, будут это столбцы или строки. Итак, первая выборка — оптическая плотность. Ей соответствует вторая выборка — концентрация стандартного раствора фосфат-иона в моль/л, как показано на рис. 8.9.

	A	B
1	оптическая плотность	концентрация стандартного раствора фосфат-иона
2	0,425	0,1
3	0,509	0,2
4	0,532	0,4
5	0,576	0,6
6	0,594	0,8
7	0,62	1
8	Коэффициент Пирсона	
9	0,941059714	
10	Значимость корреляции	
11	0,002625693	
12	Доверительный 95% интервал	
13	0,547915535	
14	0,993705874	
15		
16		

Рис. 8.9. Корреляционный анализ калибровочных данных

Подсчитаем коэффициент Пирсона. Он равен 0,94106 при значимости 0,002626 и 95% доверительном интервале 0,547916 – 0,993706.

Высокий коэффициент корреляции при очень высокой значимости свидетельствует о корректности калибровочного графика и возможности его использования в практическом научном исследовании.

Рекомендуемая литература

1. Афифи А., Эйзен С. Статистический анализ: Подход с использованием ЭВМ. — М.: Мир, 1982.
2. Большев Л.Н., Смирнов Н.В. Таблицы математической статистики. — М.: Наука, 1983.
3. Браунли К.А. Статистическая теория и методология в науке и технике. — М.: Наука, 1977.
4. Гайдышев И. Анализ и обработка данных: специальный справочник. — СПб: Питер, 2001.

5. Гантмахер Ф.Р. Теория матриц. — М.: Наука, 1988.
6. Гласс Дж., Стэнли Дж. Статистические методы в педагогике и психологии. — М.: Прогресс, 1976.
7. Голуб Дж., Ван Лоун Ч. Матричные вычисления. — М.: Мир, 1999.
8. Деммель Дж. Вычислительная линейная алгебра. Теория и приложения. — М.: Мир, 2001.
9. Дюк В. Обработка данных на ПК в примерах. — СПб.: Питер, 1997.
10. Лакин Г.Ф. Биометрия. — М.: Высшая школа, 1990.
11. Медик В.А., Токмачев М.С., Фишман Б.Б. Статистика в медицине и биологии: Руководство. В 2-х томах / Под ред. Ю.М. Комарова. Т. 1. Теоретическая статистика. — М.: Медицина, 2000.
12. Мюллер П., Нойман П., Шторм Р. Таблицы по математической статистике. — М.: Финансы и статистика, 1982.
13. Сборник научных программ на Фортране. Вып. 1. Статистика. — М.: 1974.
14. Уилкинсон Дж. Х., Алгебраическая проблема собственных значений. — М.: Наука, 1970.
15. Уилкинсон, Райнш. Справочник алгоритмов на языке АЛГОЛ. Линейная алгебра. — М.: Машиностроение, 1976.
16. Уилкс С. Математическая статистика. — М.: Наука, 1967.
17. Уткин В.А. Статистические технологии в медицинских исследованиях: Монография. — Пятигорск: ГНИИК, 2002.
18. Фукс В. По всем правилам искусства. Точные методы в исследованиях литературы, музыки и изобразительного искусства // В кн.: Моль А., Фукс В., Касслер М. Искусство и ЭВМ. — М.: Мир, 1975.

ГЛАВА 9

Методы математического моделирования в программах VBA и C/C++

«Человеческий мозг содержит слишком много частиц для того, чтобы решить все уравнения, необходимые для того, чтобы предсказывать чье-то поведение. Мы разве только когда-нибудь в обозримом будущем научимся предсказывать поведение червя нематоды».
С.У. Хокинг

Для работы с материалами главы понадобятся лицензионные средства разработки и тестирования программ:

- Microsoft Excel 97 или выше;
- Microsoft Visual C++ 5.0 или выше;
- Borland C++ 5.5 или выше;
- Bloodshed Dev-C++ 4.01 или выше.

Потребуется знание VBA, C и C++.

Решаемая проблема

Материалы данной главы несколько выбиваются из проходящей красной нитью через всю книгу темы обработки данных, рассмотренной в главах 3—8. Мы посчитали, что необходимо включить главу о математическом моде-

лировании в книгу, посвященную практическому программированию для ученых и инженеров. Мы также хотели продемонстрировать методику составления и отладки научных алгоритмов, которой пользуемся на протяжении всей книги, и которая осталась как бы скрытой в предыдущих главах, т. к. в них применялись уже отлаженные функции, а также показать, насколько простыми, при верных исходных предположениях, могут быть решения сложнейших задач.

Математической моделью называют приближенное описание какого-либо класса явлений окружающего мира, выраженное с помощью математической символики.

В математической модели переменные и константы моделируют реальные сущности, но не могут самостоятельно их порождать, представляя собой только символы на бумаге или на экране компьютера. Математическими формулами можно описать, но нельзя объяснить физическую картину явлений. При попытках более детального исследования или получении новых экспериментальных данных приходится создавать новые математические модели или совершенствовать старые. Поэтому главное в математическом моделировании — выделение свойств исследуемых объектов или явлений, существенных для решаемой в данный момент конкретной практической задачи. Пытаясь описать всевозможные характеристики явления, которые стали известны исследователю из эксперимента, он рискует запутаться в частностях и не описать их вовсе. Только опыт и здравый смысл помогут отличить влияние существенной части влияния на исследуемый процесс от совокупного влияния малозначительных черт.

Исследование корректно составленных моделей позволяет выявить совершенно новые, часто неподдающиеся прямому исследованию в эксперименте, характеристики явлений, дополняя и даже иногда частично заменяя экспериментальные методы исследований, и породить новые неожиданные идеи.

Список областей математического моделирования настолько широк, что может охватывать абсолютно все области знаний, от механики и химии до биологии и социологии. Например, путем исследования математической модели в виде системы обыкновенных дифференциальных уравнений можно рассчитать критическую скорость шимми (автоколебаний шасси, приводящих конструкцию к разрушению) самолета, железнодорожного локомотива или автомобиля. А можно показать, что государство, основанное на совокупности национальных территориальных образований, не является устойчивым — оно или разрушится, или будет перманентно находиться в состоянии хаоса.

Напомним, что дифференциальным называется уравнение, содержащее искомую функцию, ее производные различных порядков и независимые переменные. Дифференциальные уравнения являются мощным инструментом

математического моделирования явлений, процессов, поведения объектов. Составление дифференциальных уравнений настолько тесно связано с математическим моделированием, что, говоря о математической модели, часто имеют в виду модель в виде системы дифференциальных уравнений.

Сложность составленных уравнений, моделирующих реальные системы, обычно исключает возможность их аналитического решения, поэтому для решения применяются различные численные методы.

Схема решения тем или иным методом может быть простой, даже казаться примитивной, но быть эффективной. Точность и скорость решения зависят не только от реализации схемы численного решения. Качество решения математической модели в виде дифференциальных уравнений зависит:

- от адекватности составленных уравнений физике реального процесса,
- от предварительно проведенных исследований уравнений.

Существует множество приемов составления уравнений математических моделей. Техника составления уравнений заслуживает даже не одного тома, а целой библиотеки, и не является предметом настоящей книги. Некоторые сведения о методике составления дифференциальных уравнений (вместе с подробными примерами из различных областей науки) можно почерпнуть из источников, указанных в конце главы.

Иногда для решения задачи вообще нет необходимости интегрировать систему. Так, если стоит вопрос, будет ли та или иная система устойчивой при заданных значениях параметров или при каких значениях параметров — неустойчивой, нужно выполнить не интегрирование системы дифференциальных уравнений, а исследовать ее устойчивость.

Метод интегрирования выбирается, исходя из предварительно выполненных исследований жесткости задачи.

Особенности решения

Решение поставленных в данной главе задач интегрирования дифференциальных уравнений с помощью программ на языке VBA в Microsoft Excel потребовало привлечения приемов, не применявшихся нами в предыдущих проектах.

Решение дифференциальных уравнений в электронных таблицах имеет особенность, заключающейся в необходимости передачи в прорамму VBA кода функции правых частей дифференциального уравнения. Электронные таблицы Microsoft Excel предлагают несколько возможных решений, самое сложное из которых требует написания функции синтаксического разбора введенных пользователем выражений. Можно также использовать метод

Evaluate, предварив его вызов нетривиальным синтаксическим разбором и стандартными операциями по конкатенации строк. Однако мы поступим совершенно иначе. Наиболее изящное, хотя и не самое эффективное в смысле продолжительности счета, решение заключается в том, что функция правых частей вообще не передается в VBA-модуль. Она вычисляется непосредственно в ячейке таблицы. Функция VBA помещает в ячейки электронной таблицы значения параметров, от которых зависит функция правых частей, стандартным образом введенная пользователем в другую ячейку и считанная оттуда функцией VBA.

Научные алгоритмы

Порядком дифференциального уравнения называют наибольший порядок входящих в него производных. Дифференциальное уравнение первого порядка называется также обыкновенным дифференциальным уравнением и записывается как

$$F(x, y, y') = 0,$$

где y — скалярная или s-компонентная векторная функция, подлежащая определению,

x — независимая переменная (при исследовании динамики протяженных во времени процессов обычно берется время t),

y' — полная производная y по x, может также иметь обозначение dy/dx.

Примечание

В математическом моделировании, когда модель записана в виде системы обыкновенных дифференциальных уравнений, векторная функция y или комбинация ее компонент часто называется выходом модели. В механике y называется вектором обобщенных координат.

Если в явном виде независимая переменная в уравнения не входит, система уравнений называется автономной. Если система уравнений не автономна, то можно выбрать в качестве времени новую независимую переменную τ, а затем, добавив к системе еще одно уравнение

$$d\tau/dt = 1,$$

превратить исходную систему уравнений в систему автономную, хотя явной выгоды от данного процесса можно и не ощутить. Выгода может быть получена при вычислениях, если составлять все алгоритмы только для автономных систем уравнений. Это несколько упростит интерфейс составленных функций. Можно также заранее предусмотреть автоматическое преобразование неавтономной системы уравнений к автономной системе.

Степенью дифференциального уравнения называют высший показатель степени, в которой старшая производная входит в уравнение после того, как уравнение приведено к рациональному виду. Если в уравнение искомая функция и все ее производные входят в первой степени, уравнение называется линейным, в противном случае уравнение будет нелинейным.

Если исходное уравнение разрешимо относительно производной, то получается уравнение

$$y' = f(x, y),$$

где $f(x, y)$ — функция правой части (правая часть).

Общее решение дифференциального уравнения определяется с точностью до постоянных C, определяющих семейство кривых

$$y = y(x) + C.$$

При задании начального условия и в случае выполнения перечисленных требований, решение существует, и оно единственное. Задача Коши формулируется как аналитическое или численное решение системы s обыкновенных дифференциальных уравнений, в общем случае нелинейных (с нелинейной функцией правых частей), на интервале $x \in [a,b]$ с начальными условиями

$$y(x_0) = y_0,$$

где $x_0 = a$.

Иногда, чтобы отличить одно дифференциальное уравнение от системы дифференциальных уравнений, его называют скалярным дифференциальным уравнением. Уравнение высшего порядка может быть сведено к системе уравнений первого порядка простым изменением обозначений и введением новых переменных. Размерность системы — это число уравнений, содержащих производные порядка не выше первого. Все выводы, сделанные относительно одного дифференциального уравнения, естественно обобщаются на систему, поэтому в теоретических выкладках, приведенных в настоящей главе, не делается разницы, является y скалярной или s-компонентной векторной функцией. Все показанные формулы могут быть использованы как для исследования одного уравнения, так и для исследования системы уравнений векторных переменных.

Модернизация приведенных функций состоит в замене векторами (массивами) функции и входящие в формулы скалярные параметры. Кроме того, там, где производится проверка точности, нужно в цикле производить проверку каждого элемента вектора, ответственного за точность решения.

Особенность системы уравнений относительно одного уравнения заключается в том, что принятие решения, в случае недостаточной точности, следу-

ет делать сразу же, как только один из элементов вектора даст для этого повод. Принятие же решения в случае чрезмерной точности делать только после того, как проверки всех элементов покажут, что точность чрезмерна. Для систем, соответственно, должны быть изменены также краевые — начальные и граничные — условия задачи.

Отметим один научный факт, который будет использован в дальнейших рассуждениях. Система нелинейных дифференциальных уравнений может быть представлена в виде линеаризованной системы, которая получена с использованием только первых двух членов разложения решения в ряд Тейлора

$$y' = J(x)y + (**),$$

где J — постоянная матрица Якоби системы — матрица частных производных правых частей $\partial f / \partial y$,

(**) — нелинейные члены.

Предположения

Введем предположения относительно свойств исследуемой системы дифференциальных уравнений.

Основное предположение относительно функции $f(x,y)$ состоит в том, что она удовлетворяет условию Липшица

$$\|f(x, y_1) - f(x, y_2)\| \le L\|y_1 - y_2\|,$$

где L — константа Липшица,

для всех значений $x \in [a,b]$ и, в общем случае, всех соответствующих компонент векторов y_1 и y_2.

Константа Липшица играет важную роль в теории численных методов, в частности, при исследовании их численной устойчивости, и может быть вычислена как

$$L = \|\partial f / \partial y\|,$$

в чем нетрудно заметить норму матрицы Якоби.

При выводе методов решения, основанных на разложении в ряд Тейлора, предполагается надлежащая дифференцируемость y. При этом говорят, что функция y должна обладать той степенью гладкости, которая требуется в конкретном частном случае. Проблема может возникнуть, если искомые функции имеют разрыв. Кроме того, разрыв могут иметь правые части дифференциальных уравнений. Такая ситуация может

быть обусловлена конструктивными особенностями моделируемой физической системы.

Напомним, что точкой разрыва (особой точкой) функции называется такая точка, в которой функция не является непрерывной. Точка разрыва будет первого рода, если существуют пределы

$$\lim_{x \to x_0 + 0} f(x) = f(x_0 + 0) \text{ и } \lim_{x \to x_0 - 0} f(x) = f(x_0 - 0).$$

Величина $f(x_0 + 0) - f(x_0 - 0)$ называется скачком функции.

Точка разрыва будет второго рода, если функция определена в окрестности особой точки за исключением, быть может, самой точки, и один из пределов не существует.

Разрыв первого рода в правых частях дифференциальных уравнений обычно легко преодолевается, без какой-либо модернизации алгоритма.

Устойчивость методов решения

Применительно к дифференциальным уравнениям, различают численную устойчивость методов решения и устойчивость системы дифференциальных уравнений. Таким образом, различают устойчивость метода и устойчивость уравнений как математической модели некоторого физического явления. При сходстве методов анализа совершенно различаются объекты его приложения.

Анализ устойчивости метода сводится к изучению глобальной ошибки решения, складывающейся из ошибки усечения и ошибки распространения. Выводятся рекомендации по выбору минимально допустимого шага интегрирования либо, в общем смысле, интервала устойчивости.

Другое определение устойчивого алгоритма проще: устойчивым называется алгоритм, в котором не накапливаются ошибки округления.

Жесткие задачи

Жесткой называется задача, моделирующая процесс, компоненты которого обладают несоразмеримыми характерными временами, или же процесс, характерное время которого намного меньше отрезка интегрирования.

О высокой жесткости системы свидетельствует большое значение константы Липшица.

Есть и другой показатель. Согласно Лэмберту, задача Коши для устойчивой системы называется жесткой, если локальный коэффициент жесткости задачи

$$S(x) = \max_{i=1,2,\ldots,n} \mathrm{Re}(-\lambda_i) / \min_{i=1,2,\ldots,n} \mathrm{Re}(-\lambda_i) >> 1,$$

где $\lambda_i, i = 1,2,\ldots,n$ — собственные значения матрицы Якоби системы.

Жесткой считается задача уже при значении $S(x) = 10$, хотя, на практике, могут встречаться значения до величин $S(x) = 10^6$.

Жесткие задачи решаются с помощью специально разработанных методов, некоторые из которых представлены далее.

Устойчивость решения

Устойчивость решения системы дифференциальных уравнений соответствует устойчивости движения моделируемой физической системы. Описываемое системой дифференциальных уравнений движение (под которым можно понимать любой изменяющийся во времени[1] процесс) называется устойчивым асимптотически, если достаточно малым возмущениям будет соответствовать предварительно заданная малость возмущенного движения. Для устойчивого движения имеет место также стремление амплитуды этого возмущенного движения к нулю, при неограниченном росте времени.

Практически неустойчивое поведение физической системы ведет к ее разрушению, к выходу из интервала допустимых эксплуатационных параметров либо (в случае физической системы, описываемой системой нелинейных дифференциальных уравнений) к хаосу (беспорядку, нерегулярности) того или иного типа. Критерием перехода регулярной организованной структуры к хаосу может служить устойчивость структур по отношению к малым возмущениям. Если такая устойчивость отсутствует, детерминированное описание структур теряет смысл, и необходимо использовать статистические методы.

Для анализа поведения объекта или явления, описываемого системой дифференциальных уравнений, не всегда требуется решать эту систему. Если исследователя, например, интересуют значения некоторых параметров системы или начальных условий задачи, определяющих, будут ли в системе затухать возмущения, или, наоборот, малые возмущения будут приводить к нарастанию амплитуды возмущенного движения, значит, исследователю нужно исследовать устойчивость системы (интегрировать систему дифференциальных уравнений нет необходимости). Начальные возмущения для реализации такого движения всегда найдутся.

---

[1] *В качестве времени может выступать некоторый параметр.*

Критерий Ляпунова

Локальное поведение решения нелинейной системы в первом приближении будет определяться решением линеаризованной системы. Согласно критерию Ляпунова, решение системы при любых членах (**) порядка выше первого (см. приведенную общую форму линеаризованного уравнения) является устойчивым, если все характеристические числа матрицы Якоби имеют отрицательные вещественные (действительные) части:

$$\mathrm{Re}(\lambda_i) < 0, i = 1,2,\ldots,n.$$

Задача сводится, таким образом, к анализу уравнения

$$\Delta(\lambda) = \left|\lambda I - J\right| = 0.$$

Раскрытие определителя дает характеристическое (вековое) уравнение — вещественный многочлен (характеристический полином) с комплексными корнями

$$\sum_{i=0}^{n} a_i \lambda^{n-i} = 0,$$

где a_i, i =1,2,,n — коэффициенты полинома, причем $a_0 \neq 0$.

Функция вычисления коэффициентов характеристического полинома, полученного для матрицы в верхней почти треугольной форме Хессенберга, к которой матрица Якоби линеаризованной системы дифференциальных уравнений может быть приведена тем или иным методом, дана в листинге 9.1.

Листинг 9.1

```
int PolynomCoeff (double h[],int n,double b[])
//
// Функция вычисления коэффициентов характеристического полинома
// действительной матрицы в верхней почти треугольной форме
// Хессенберга либо, как частный случай, в форме трехдиагональной
// матрицы.
// Обозначения:
//  h — матрица Хессенберга,
//  n — порядок матрицы h,
//  b — полученный вектор длиной (n + 1) коэффициентов
//      характеристического полинома, расположенных начиная с
//      коэффициента при младшей степени переменной.
// Возвращаемое значение:
```

```
//   0 при нормальном окончании счета,
//  -1 при недостатке памяти для вычислений.
//
{
 register int i,is,l, // Счетчики
              ir;     // Номер основного шага вычислений

 int ij,              // Начальный индекс для произведения
     il,              // Начало первой строки рабочего массива
     in,              // Рабочий индекс для суммирования
     ik = 0,          // Начало ir-й строки рабочего массива
     kl,              // Текущий индекс для суммирования
     j,               // Текущий индекс для произведения
     im,              // Рабочий индекс для сдвига
     kn = 0,          // Начальные индексы для суммирования
     km = 0;

 double *s,           // Рабочий массив
        sr,           // Буфер
        sig,          // В формуле: (-1) в степени (ir-1)
        sig1 = -1;    // Знак последнего слагаемого

 ////////////////////////////////////////////////////////////
 // Рабочий массив — нижняя треугольная матрица коэффициентов
 // — хранится в виде вектора по строкам без последней строки
 if ((s = new double[n * (n + 1) / 2]) == 0)
   return -1;

 s[0] = 1;
 for (i = 1; i < n * (n + 1) / 2; i++)
   s[i] = 0;
 for (i = 0; i <= n; i++)
   b[i] = 0;
 for (ir = 1; ir <= n; ir++)
   {
    ///////////////
    // Суммирование
    il = 0;
```

```
ik += ir;
ij = 1;
kl = km;
sig1 = -sig1;
sig = sig1;
for (is = 0; is <= ir - 1; is++)
  {
   if (is != ir - 1)
     {
      sr = 1;
      j = ij;
      for (l = is + 2; l <= ir; l++)
        {
         sr *= h[j];
         j += n + 1;
        }
      sr *= h[kl] * sig;
      sig = -sig;
     }
   else
     sr = h[kn];
   kl++;
   in = ik;
   for (i = 0; i <= is; i++)
     {
      if (ir != n)
        s[in] = s[in] + sr * s[il];
      else
        b[i] = b[i] + sr * s[il];
      in++;
      il++;
     }
   ij += n + 1;
  }
kn += n + 1;
km += n;

////////
```

```
    // Сдвиг
    in = ik;        // Далее in — индекс для сдвига
    im = in — ir;
    for (i = 0; i <= ir — 1; i++)
      {
       if (ir != n)
         s[in + 1] = s[in + 1] — s[im];
       else
         b[i + 1] = b[i + 1] — s[im];
       in++;
       im++;
      }
   }

 delete [] s;

 return 0;
}
```

Для исследования устойчивости нет необходимости в поиске корней многочлена. Предложено использовать специальные критерии: критерий Шура, критерий Рауса, критерий Рауса—Гурвица (критерий Гурвица), критерий Льенара—Шипара. Целью анализа по любому из названных алгоритмов является выяснение, попадает ли хотя бы один комплексный корень вещественного многочлена в правую полуплоскость (имеет положительную действительную часть). Наличие таких корней или хотя бы одного корня свидетельствует о неустойчивости системы: ее решение при любых возмущениях параметров или начальных условий с течением времени не затухает. Это можно оценить также и визуально, решив систему аналитически или одним из численных методов, и для наглядности изобразить результаты графически. На графике при устойчивости системы будут затухающие колебания, при неустойчивости — колебания с увеличивающимся размахом (амплитудой). При каких-то критических значениях параметров будут равномерные колебания с постоянной амплитудой. Есть задачи, в которых именно такие колебания являются целью при конструировании технического устройства. Для реальной системы, например, механической, в случае неустойчивости, амплитуда колебаний будет увеличиваться, пока позволит прочность конструкции. Затем произойдет разрушение конструкции (пример — отрыв фрагмента летательного аппарата при флаттере).

С помощью метода деления отрезка пополам или аналогичного, производится уточнение корня (в данном случае, конечно же, — не корня много-

члена, а так называемого управляющего параметра, от величины которого зависит устойчивость системы), а интервал его локализации в простейшем случае ищется в предполагаемых из физических соображений пределах простым пошаговым поиском. Могут быть применены также методы безусловной нелинейной оптимизации.

При наличии адекватной математической модели расчеты по представленным алгоритмам показывают поразительное совпадение с опытными данными.

Критерий Рауса

Критерий Рауса считается быстрее, чем другие рассмотренные критерии, т. к. в других схемах надо считать определители Гурвица, тем более что результаты, полученные с помощью всех упомянутых методов, совпадают.

Пусть характеристический многочлен (полином) представлен в виде

$$a_0x^n + b_0x^{n-1} + a_1x^{n-2} + b_1x^{n-3} + \ldots (a_0 \neq 0).$$

Коэффициенты данного многочлена могут быть вычислены с помощью функции, показанной на листинге 9.1.

Схему Рауса многочлена представляет система чисел

$a_0 \quad a_1 \quad a_2 \quad \ldots$

$b_0 \quad b_1 \quad b_2 \quad \ldots$

$c_0 \quad c_1 \quad c_2 \quad \ldots$

$d_0 \quad d_1 \quad d_2 \quad \ldots$

...

где, например,

$$c_i = a_{i+1} - b_{i+1}a_0 / b_0, i = 0,1,2,\ldots.$$

Чтобы все корни вещественного многочлена имели отрицательные вещественные части, необходимо и достаточно, чтобы при выполнении алгоритма Рауса все элементы первого столбца схемы Рауса получались отличными от нуля и одного знака.

Функция, показанная на листинге 9.2, реализует алгоритм Рауса. При построении программы следует учесть, что функция `PolynomCoeff` выдает коэффициенты характеристического полинома, начиная с коэффициента при младшей степени переменной, а функция `Routh` требует массив коэффициентов, начиная с коэффициента при старшей степени. Необходимо инвертировать массив коэффициентов, вычисленный функцией `PolynomCoeff`.

Листинг 9.2

```
int Routh (double a[],int n)
//
// Функция определения числа корней с положительными
// действительными частями характеристического полинома с
// действительными коэффициентами по алгоритму Рауса.
// Обозначения:
//  a — массив коэффициентов полинома длиной (n + 1),
//      расположенных начиная с коэффициента при старшей
//      степени переменной, a[0] не равно нулю,
//  n — степень полинома,
// Возвращаемое значение:
//  число корней с положительными действительными частями,
//  -1 при недостатке памяти.
//
{
 register int i,j,k,l;  // Счетчики

 int Ret;               // Возвращаемое значение

 double Temp,           // Рабочая переменная
        Eps = 0.000001, // Уровень малости
        *s;             // Рабочий массив

 if (!(s = new double[n + 2]))
   return -1;

 ////////////////////
 // Начальные значения
 for (i = 0; i <= n; i += 2)
   {
    s[i] = a[i];
    j = i + 1;
    s[j] = j <= n ? a[j] : 0;
   }

 Ret = s[0] * s[1] < 0 ? 1 : 0;
 //////////////
```

```
// Схема Рауса
for (i = n — 1; i >= 1; i--)
  {
   l = (i + 1) / 2 * 2;

   ////////////////////
   // Перестановка строк
   for (j = 0; j <= l; j += 2)
     {
      Temp = s[j];
      s[j] = s[j + 1];
      s[j + 1] = Temp;
     }

   //////////////////////
   // Проверка на малость
   if (fabs (s[0]) < Eps)
     {
      s[0] = Eps * Sign (s[1]);

      for (j = 2, k = 0; j <= l; j += 2)
        if (fabs(s[j]) < Eps)
          k += 2;

      if (k == i)
        for (j = 0; j == i; j += 2)
          s[j] = s[j + 1] * (i — j);
     }

   ///////////////////////
   // Преобразование строк
   for (j = 1, Temp = s[1]; j <= l; j += 2)
     s[j] = s[j + 2] — Temp * s[j + 1] / s[0];

   s[l + 1] = 0;

   if (s[0] * s[1] < 0)
     Ret++;
  }
```

```
 delete [] s;

 return Ret;
}

double Sign (double k)
//
// Функция определения знака числа.
// Обозначения:
//  k — число.
// Возвращаемое значение:
//   1, если число положительное,
//   0, если число - нуль,
//  -1, если число отрицательное.
//
{
 if (k > 0)
   return 1;
 else if (k < 0)
   return -1;
 else
   return 0;
}
```

Критерий Рауса–Гурвица

Для пояснения применения критерия Рауса—Гурвица представим многочлен как

$$a_0x^n + a_1x^{n-1} + \ldots + a_{n-1}x + a_n\,(a_0 > 0).$$

Для выполнения критерия Рауса—Гурвица необходимо и достаточно, чтобы были положительными все n определителей Гурвица, составленных из коэффициентов многочлена по схеме

$$\Delta_k = \begin{vmatrix} a_1 & a_3 & a_5 & \dots & a_{2k-1} \\ a_0 & a_2 & a_4 & \dots & a_{2k-2} \\ 0 & a_1 & a_3 & \dots & a_{2k-3} \\ 0 & a_0 & a_2 & \dots & a_{2k-4} \\ . & . & . & . & . \\ 0 & 0 & 0 & \dots & a_k \end{vmatrix}, k = 1,2,\dots,n.$$

На главной диагонали определителя записаны коэффициенты, начиная с a_1, а строки заполнены коэффициентами через один.

Примечание

Скорость алгоритма Рауса-Гурвица существенно зависит от алгоритма вычисления определетеля. Значение определителя является побочным продуктом некоторых видов разложения матриц.

Критерий Льенара–Шипара

Критерий Льенара—Шипара популярен среди исследователей флаттера. Согласно схеме критерия, для нахождения всех корней характеристического полинома слева от мнимой оси, необходимым и достаточным является совместное выполнение трех условий:

$a_n > 0$;

$a_1 > 0$, $a_3 > 0$,, $a_{2k-1} > 0$, либо $a_2 > 0$, $a_4 > 0$,, $a_{2k} > 0$,;

$\Delta_1 > 0$, $\Delta_3 > 0$,, $\Delta_{2k-1} > 0$, либо $\Delta_2 > 0$, $\Delta_4 > 0$,, $\Delta_{2k} > 0$, .

Недостатком данного алгоритма, как и предыдущего, является необходимость вычисления определителей.

Численное решение дифференциальных уравнений

Если решается уравнение порядка выше первого, количество начальных условий будет равно порядку уравнения, т. к. начальные условия накладываются как на функцию y, так и на все ее производные.

Среди численных методов наиболее употребительны разностные методы, использующие значения функции в нескольких последовательных точках (многошаговые методы) и одношаговые методы (например, метод Рунге—Кутта), со значением функции в одной предыдущей точке.

Все рассмотренные численные методы являются дискретными, т. е. с их помощью ищется последовательность значений искомой функции y в N заданных точках

$$y_n \approx y(x_n),$$

где $x_{n+1} = x_n + h_n$, $n = 0,1,...,N-1$, $x_0 = a$, $x_N = b$ — заданное множество точек интегрирования,

$h_n > 0$ — шаг сетки.

Чаще всего рассматривается случай $h_n = h$, где h — постоянная величина.

Примечание

Следует отличать шаг сетки решения от шага вывода решения, т. е. шага, с которым программное решение должно выдаваться пользователю. Хотя эти величины могут совпадать, в общем случае они различны. Удобно, чтобы данные величины были кратными во избежание чрезмерно малого значения шага сетки в конце интервала. Это справедливо для всех методов, особенно для методов с автоматическим выбором длины шага, при вызове которых как начальный шаг решения, так и минимальный шаг должны быть кратны шагу вывода. Впрочем, за счет некоторого усложнения программ во всех приведенных листингах данное требование не обязательно.

Наиболее распространенные схемы решения обыкновенных дифференциальных уравнений, без потери общности, рассмотрим на примере решения одного дифференциального уравнения.

Нами даны функции решения дифференциальных уравнений, составленные на VBA. Дополнительно приводятся аналоги на языке C++. Функция, описывающая правую часть дифференциального уравнения, в проектах на языке C++ имеет имя `UserFunc`. В проектах на языке VBA данная функция вообще не содержит кода, описывающего правую часть дифференциального уравнения. Данная VBA функция нужна только в качестве интерфейса к ячейкам электронной таблицы, которые описывают правую часть уравнения. Подробнее на всех этих вопросах мы остановимся далее.

Для решения систем дифференциальных уравнений, функция правых частей `UserFunc` будет иметь вид, отличающийся от приведенного вида тем, что y будет вектором.

Исходные коды функций `UserFunc` на используемых алгоритмических языках содержатся в представленных листингах, реализующих метод Рунге—Кутта на языках VBA и C++. Предполагается, что для всех методов решения обыкновенных дифференциальных уравнений интерфейс данной функции будет аналогичным.

Одношаговые методы

Решение обыкновенного дифференциального уравнения в точке x_{n+1}, если оно известно в точке x_n, на основе разложения в ряд Тейлора находится по формуле

$$y(x_{n+1}) = y(x_n) + h\Delta(x_n, y_n, h),$$

где

$$\Delta(x_n, y_n, h) = \sum_{k=1}^{\infty} \frac{h^{k-1}}{k!} y^{(k)}(x).$$

Если бесконечный ряд оборвать на некотором числе членов и точное значение $y(x_n)$ заменить приближенным значением y_n, получится формула

$$y_{n+1} = y_n + h\varphi(x_n, y_n, h),$$

где

$$\varphi(x, y, h) = \sum_{k=1}^{p} \frac{h^{k-1}}{k!} f^{(k-1)}(x_n, y_n),$$

где p — степень метода решения.

Считается, что формулы более высоких степеней дают не только более высокую точность, но и уменьшают количество вычислительной работы, т. к. допускают большие величины шага. При $p = 1$ имеем известный метод Эйлера.

С целью исключения производных из формулы для φ при построении методов со степенями $p > 1$ Рунге, Хойн и Кутта предложили процесс интерполяции рядов Тейлора выражениями вида

$$\varphi(x, y, h) = \sum_{r=1}^{m} c_r k_r,$$

где m — количество этапов решения.

В конкретном методе фигурирует как число этапов, означающее количество вычислений правой части, так и степень метода, означающее число удерживаемых членов разложения и характеризующего теоретическую точность метода. Для степени $p > 4$ в литературных источниках показано, что количество этапов $m > p$.

В источниках все одношаговые методы часто традиционно называют методами Рунге—Кутта.

Явные схемы

Предположим, коэффициенты в предыдущей формуле вычисляются как

$$k_1 = f(x,y),\ k_r = f(x + \alpha_r h, y + h\sum_{s=1}^{r-1}\beta_{r,s}k_s),\ r = 2,3,...,m,$$

$\alpha_r, \beta_{r,s}$ — константы.

Данная формула определяет общий вид так называемых явных схем одношаговых методов. Подробный вывод громоздок и здесь не приводится.

Основное применение явные схемы находят при решении нежестких систем.

Метод Рунге–Кутта

Расчетные формулы для схемы классического метода Рунге—Кутта имеют вид

$y_{n+1} = y_n + (k_1 + 2k_2 + 2k_3 + k_4) / 6,$

где

$k_1 = hf(x_n,y_n),$

$k_2 = hf(x_n + h / 2, y_n + k_1 / 2),$

$k_3 = hf(x_n + h / 2, y_n + k_2 / 2),$

$k_4 = hf(x_n + h, y_n + k_3).$

Наилучший прием проверки точности решения состоит в том, чтобы после окончания решения задачи методом Рунге—Кутта уменьшить шаг решения h в два раза и повторить решение с новым шагом. Если разница между двумя решениями устраивает исследователя, можно удовлетвориться первым шагом. В противном случае, алгоритм уменьшения шага решения следует повторить.

Классический метод Рунге—Кутта может рассматриваться, как обобщение для дифференциальных уравнений квадратурной формулы Симпсона вычисления определенных интегралов.

Листинг 9.3

```
Private Function RungeKuttaFV(X As Double, Y As Double, Xout As Double, _
Hbeg As Double, Hmin As Double, RangeX As Variant, RangeY As Variant, _
RangeRight As Variant) As Double
'
' Функция интегрирования обыкновенного дифференциального уравнения
```

```
' первого порядка методом Рунге-Кутта (полнофункциональная версия)
' Обозначения:
'  X — начало интервала интегрирования
'  Y — значение функции в точке X
'  Xout — конец интервала интегрирования (точка вывода)
'  Hbeg — заданный шаг интегрирования
'  Hmin — минимально допустимый шаг интегрирования
'  RangeX — ячейка аргумента
'  RangeY — ячейка функции
'  RangeRight — ячейка формулы правой части
' Возвращаемое значение:
'  значение функции в точке Xout.
'
 Dim H As Double  ' Шаг решения
 Dim Xs As Double ' Сохраненное значение
 Dim Ys As Double ' Сохраненное значение
 '
 ' Начальные значения параметров
 '
 H = Hbeg
 Xs = X
 Ys = Y
 '
 ' Выполнять решение, пока не будет достигнута точка вывода
 '
 Do While X < Xout
   '
   ' Коррекция длины шага, при необходимости
   '
   If X + H > Xout Then
     H = Xout — X
     If H < Hmin Then
       X = Xs
       Y = Ys
       H = Xout — X
     End If
   End If
   Ys = Y
```

```
    '
    ' Интегрирование уравнения
    '
    Y = RungeKutta(X, Y, H, RangeX, RangeY, RangeRight)
    Xs = X
    X = X + H
  Loop
  RungeKuttaFV = Y
End Function
```

Здесь остановимся и обратим внимание на функцию вычисления правой части дифференциального уравнения. На самом деле, в VBA никакой функции вычисления правой части дифференциального уравнения нет. Используется аппарат вычислений Microsoft Excel. Это удобное и универсальное решение, освобождающее программиста от рутинного труда по составлению программы синтаксического анализа выражений и верификации введенной формулы. Недостаток подхода — длительное время решения, ввиду продолжительности операций вывода чисел в ячейки рабочего листа и считывания из ячейки вычисленного значения формулы.

Листинг 9.3 (продолжение)

```
Private Function Right(X As Double, Y As Double, RangeX As Variant, _
RangeY As Variant, RangeRight As Variant) As Double
'
' Функция вычисления правой части обыкновенного дифференциального уравне-
ния
'  X — значение аргумента
'  Y — значение функции
'  RangeX — ячейка аргумента
'  RangeY — ячейка функции
'  RangeRight — ячейка формулы правой части  дифференциального уравнения
' Возвращаемое значение:
'  значение правой части дифференциального уравнения.
'
 Range(RangeX).Select
 ActiveCell.FormulaR1C1 = X
 Range(RangeY).Select
 ActiveCell.FormulaR1C1 = Y
 Right = ActiveSheet.Range(RangeRight).Value
End Function
```

Особенностью представленной ниже функции является то, что блок контроля над достижением точки вывода решения выполнен не в теле самой функции, а вынесен за ее пределы. У представленных в следующих разделах методов с контролем погрешности процесс удобнее осуществлять в теле функций. Данная в листинге 9.4 функция предназначена не для непосредственного решения уравнений. Она используется во вспомогательных целях, например, для старта многошаговых методов, как будет показано при их описании.

Листинг 9.4

```
Private Function RungeKutta(X As Double, Y As Double, H As Double, _
RangeX As Variant, RangeY As Variant, RangeRight As Variant) As Double
'
' Функция интегрирования обыкновенного дифференциального уравнения
' первого порядка методом Рунге-Кутта (упрощенная версия)
' Обозначения:
'  X — начало интервала интегрирования
'  Y — значение функции в точке X
'  H — шаг интегрирования
'  RangeX — ячейка аргумента
'  RangeY — ячейка функции
'  RangeRight — ячейка формулы правой части
' Возвращаемое значение:
'  значение функции в точке X + H.
'
 Dim K1 As Double ' Вспомогательная величина
 Dim K2 As Double ' Вспомогательная величина
 Dim K3 As Double ' Вспомогательная величина
 Dim K4 As Double ' Вспомогательная величина
 '
 ' Интегрирование уравнения
 '
 K1 = H * Right(X, Y, RangeX, RangeY, RangeRight)
 K2 = H * Right(X + H / 2, Y + K1 / 2, RangeX, RangeY, RangeRight)
 K3 = H * Right(X + H / 2, Y + K2 / 2, RangeX, RangeY, RangeRight)
 K4 = H * Right(X + H, Y + K3, RangeX, RangeY, RangeRight)
 RungeKutta = Y + (K1 + (K2 + K3) * 2 + K4) / 6
End Function
```

Аналог показанной выше VBA-функции, выполненный на языке C++, дан в листинге 9.5. Дополнительно показана вся процедура вызова функции, упомянутая в комментарии к предыдущему листингу. Тестовая программа составлена в виде консольного приложения, разработка которого в различных системах программирования рассмотрена в *приложении 3*.

Листинг 9.5

```
#include <fstream.h>
#include <iomanip.h>
#include <math.h>

typedef double (*userFuncOne) (double,double);

double UserFunc (double,double);
double RungeKutta (double,double,double,userFuncOne);

void main (void)
{
 double x = 0,        // Начальное значение абсциссы
                      // и текущая точка решения
        xend = 3,     // Конечная точка интервала решения
        xout = x,     // Текущая точка вывода решения
        y = 1,        // Начальное значение ординаты
                      // и значение функции в текущей точке
        xs,ys,        // Сохраненные координаты предыдущей точки
        h,            // Текущий шаг решения
        hbeg = 0.1,   // Заданный шаг решения
        hmin = 0.001, // Минимально допустимый шаг решения
        hout = 0.2;   // Шаг вывода решения

 //////////////////////////
 // Вывод начальных условий
 cout << "Начальные значения: x0, y0\n"
      << "--------------------------\n"
      << setw (10) << xout << setw (10) << y << endl;

 ////////////////////////////////////////////
 // Интегрирование дифференциального уравнения
```

```
// в заданных интервалах
cout << "Решение: x, y (вычисленное), y (известное)\n"
     << "----------------------------------------\n";
do
  {
   ////////////////////////////////////////////////
   // Начальные установки для текущего шага решения
   h = hbeg;
   xout += hout;

   ////////////////////////////////////////////////////////////
   // Выполнять решение, пока не будет достигнута точка вывода
   while (x < xout)
     {
      ////////////////////////////////////////
      // Если шаг вывода не кратен шагу решения,
      // может возникнуть ситуация, что
      if (x + h > xout)
        {
         h = xout - x;

         ////////////////////////////////////////////
         // Ограничение с целью избежать переполнения
         // или потери значимости
         if (h < hmin)
           {
            ////////////////////////////////
            // Восстановить предыдущую точку
            x = xs;
            y = ys;

            ////////////////////////////////
            // Принять шаг большим заданного
            // на величину не более hmin
            h = xout - x;
           }
        }
      ////////////////////////////////////
```

```
        // Сохранить ординату текущей точки
        ys = y;

        ////////////////////////
        // Выполнить шаг решения
        y = RungeKutta (x,y,h,UserFunc);

        ////////////////////////////////////
        // Сохранить абсциссу текущей точки
        xs = x;

        ////////////////////////////////////
        // Вычисление абсциссы следующей точки
        x += h;
       }

    ////////////////////////////////////////////////////////
    // Вывод численного и известного аналитического решения
    cout << setw (10) << xout << setw (10) << y
         << setw (10) << (exp (-xout)) << endl;
   } while (xout <= xend);
}
double RungeKutta (double x,double y, double h,userFuncOne fff)
//
// Функция интегрирования методом Рунге-Кутта одного
// обыкновенного дифференциального уравнения.
// Обозначения:
//    x — начало интервала интегрирования,
//    y — значение функции в точке x,
//    h — шаг интегрирования,
//  fff — имя функции правой части уравнения.
// Возвращаемое значение:
//  значение функции в точке x + h.
//
{
 double k1 = h * fff (x,y),
        k2 = h * fff (x + h / 2,y + k1 / 2),
```

```
        k3 = h * fff (x + h / 2,y + k2 / 2),
        k4 = h * fff (x + h,y + k3);

 return y + (k1 + (k2 + k3) * 2 + k4) / 6;
}
```

Обратим внимание на функцию вычисления правой части дифференциального уравнения. Предполагается, что для всех составленных на алгоритмическом языке C++ функция должна иметь точно такую же интерфейсную часть.

Листинг 9.5 (продолжение)

```
double UserFunc (double x,double y)
//
// Пример обыкновенного дифференциального уравнения y' = -y
// (известное аналитическое решение: семейство y = exp (-x) + C)
// для тестирования численных методов решения.
//
{
 return -y;
}
```

Метод Гилла

Существует несколько вариантов формул четвертого порядка, имеющих преимущество по сравнению с классической формулой Рунге—Кутта, в смысле минимизации количества вычислений. Популярен метод Рунге—Кутта в модификации, предложенной Гиллом:

$y_{n+1} = y_n + (k_1 + 3k_2 + k_3 + k_4) / 6,$

где

$k_1 = hf(x_n,y_n),$

$k_2 = hf(x_n + h / 2,y_n + k_1 / 2),$

$k_3 = hf(x_n + h / 2,y_n - k_1 / 2 + k_2),$

$k_4 = hf(x_n + h,y_n + k_1 / 2 + k_3 / 2).$

Методика вызова, как и результаты расчета, с помощью функции `Gill` полностью совпадают с реализацией классической формулы Рунге—Кутта, поэтому пример применения не приводится. Не приводится и VBA-функция. Ее модификация на основе функции `RungeKutta` не вызовет совершенно никаких сложностей.

Листинг 9.6

```
double Gill (double x,double y, double h,userFuncOne fff)
//
// Функция интегрирования методом Рунге-Кутта-Гилла одного
// обыкновенного дифференциального уравнения.
// Обозначения:
//    x — начало интервала интегрирования,
//    y — значение функции в точке x,
//    h — шаг интегрирования,
//  fff — имя функции правой части уравнения.
// Возвращаемое значение:
//  значение функции в точке x + h
//
{
 double k1 = h * fff (x,y),
        k2 = h * fff (x + h / 2,y + k1 / 2),
        k3 = h * fff (x + h / 2,y — k1 / 2 + k2),
        k4 = h * fff (x + h,y + k2 / 2 + k3 / 2);

 return y + (k1 + k2 * 3 + k3 + k4) / 6;
}
```

Метод Мерсона

Рассматриваемые далее формулы получены на основе идеи, что, если схема решения дифференциального уравнения, наряду с приращением функции, будет содержать некоторое приближение более высокого порядка, последнее можно использовать для управления погрешностью решения и длиной шага интегрирования.

Созданная Мерсоном оригинальная модификация метода Рунге—Кутта предоставляет возможность автоматического выбора шага h для достижения заданной точности решения. Метод Мерсона (Кутты—Мерсона, Рунге—Кутта—Мерсона), являющийся пятиэтапным методом четвертого порядка, дает оценку погрешности решения в общем случае (т. е. для нелинейных уравнений) сверху, что позволяет управлять шагом сетки решения, добиваясь точности, не хуже заданной. Формулы метода Мерсона могут быть представлены в виде:

$y_{n+1} = y_n + y_1,$

где

$y_1 = k_1 / 6 + 2k_4 / 3 + k_5 / 6,$

$k_1 = hf(x_n,y_n),$

$k_2 = hf(x_n + h / 3,y_n + k_1 / 3),$

$k_3 = hf(x_n + h / 3,y_n + k_1 / 6 + k_2 / 6),$

$k_4 = hf(x_n + h / 2,y_n + k_1 / 8 + 3k_3 / 8),$

$k_5 = hf(x_n + h,y_n + k_1 / 2 — 3k_3 / 2 + 2k_4).$

Кроме того, подсчитывается

$y_2 = k_1 / 2 — 3k_3 / 2 + 2k_4,$

после чего производится вычисление величины

$\varepsilon' = |y_1 - y_2|,$

которая сравнивается с заданной абсолютной погрешностью ε. Если $\varepsilon' > \varepsilon$, шаг интегрирования уменьшается вдвое и вычисление повторяется с предыдущей точки. Недостатком подхода является то, что не предусмотрено увеличение шага интегрирования для слишком малой погрешности. Достоинством же представленного программного решения является то, что на его основе может быть легко построен любой метод семейства.

Представленная в листинге 9.7 функция `Merson` имеет приятную особенность, по сравнению с реализациями других методов, в виде высокой скорости работы. Некоторое замедление ощутимо только на начальном этапе подстройки шага решения. Применение данной функции мы считаем предпочтительным.

Листинг 9.7

```
Private Function Merson(X As Double, Y As Double, Xout As Double, _
Hbeg As Double, Hmin As Double, Epsilon As Double, RangeX As Variant, _
RangeY As Variant, RangeRight As Variant) As Double
'
' Функция интегрирования обыкновенного дифференциального уравнения
' первого порядка методом Мерсона
' Обозначения:
'  X — начало интервала интегрирования
'  Y — значение функции в точке X
'  Xout — конец интервала интегрирования (точка вывода)
'  Hbeg — заданный шаг интегрирования
'  Hmin — минимально допустимый шаг интегрирования
```

```
'  Epsilon — абсолютная погрешность
'  RangeX — ячейка аргумента
'  RangeY — ячейка функции
'  RangeRight — ячейка формулы правой части
' Возвращаемое значение:
'  значение функции в точке Xout.
'
 Dim K1 As Double   ' Вспомогательная величина
 Dim K2 As Double   ' Вспомогательная величина
 Dim K3 As Double   ' Вспомогательная величина
 Dim K4 As Double   ' Вспомогательная величина
 Dim K5 As Double   ' Вспомогательная величина
 Dim H As Double    ' Шаг решения
 Dim Xs As Double   ' Сохраненное значение
 Dim Ys As Double   ' Сохраненное значение
 Dim D As Double    ' Вспомогательная величина
 Dim Err As Integer ' Код ошибки
 '
 ' Начальные значения параметров
 '
 H = Hbeg
 Xs = X
 Ys = Y
 Err = 0
 '
 ' Выполнять решение, пока не будет достигнута точка вывода
 '
 Do
   '
   ' Контроль конца участка интегрирования
   '
   D = Xout — X
   '
   ' Контроль минимального шага
   '
   If D < H Then
     If D <= Hmin Then
       X = Xs
```

```
      Y = Ys
      H = H + D
    Else
      H = D
    End If
  End If
  '
  ' Сохранение информации о предыдущей точке
  '
  Xs = X
  Ys = Y
  '
  ' Один шаг интегрирования
  '
  K1 = H * Right(X, Y, RangeX, RangeY, RangeRight)
  K2 = H * Right(X + H / 3, Y + K1 / 3, RangeX, RangeY, RangeRight)
  K3 = H * Right(X + H / 3, Y + K1 / 6 + K2 / 6, _
                 RangeX, RangeY, RangeRight)
  K4 = H * Right(X + H / 2, Y + 0.125 * K1 + 0.375 * K2, _
                 RangeX, RangeY, RangeRight)
  K5 = H * Right(X + H, Y + K1 / 2 - 1.5 * K3 + K4 * 2, _
                 RangeX, RangeY, RangeRight)
  '
  ' Контроль погрешности
  '
  D = K1 / 6 + K4 * 2 / 3 + K5 / 6
  If Not Err = 0 And Abs(K1 / 2 - 1.5 * K3 + K4 * 2 - D) > Epsilon Then
     '
     ' Уменьшить длину шага вдвое
     '
     H = H / 2
     '
     ' Контроль минимального шага
     '
     If H < Hmin Then
       H = Hmin
       Err = -2
     End If
```

```
      '
      ' Выполнить шаг решения заново
      '
      X = Xs
      Y = Ys
   Else
      '
      ' Продолжить решение с допустимыми параметрами
      '
      X = X + H
      Y = Y + D
   End If
 Loop While X < Xout
 Merson = Y
End Function
```

В листинге 9.8 показан аналог на C++, снабженный полным примером применения.

Листинг 9.8

```
#include <fstream.h>
#include <iomanip.h>
#include <math.h>

typedef double (*userFuncOne) (double,double);

double UserFunc (double,double);
double Merson
(double,double,double,double,double,double,userFuncOne,int*);

void main (void)
{
 int err;              // Код ошибки

 double x = 0,         // Начальная (затем текущая) точка интегрирования
        y = 1,         // Значение функции в точке x
        xend = 3,      // Конечная точка интегрирования
        h = 0.01,      // Шаг интегрирования
```

```
        hmin = 0.001, // Минимально допустимый шаг интегрирования
        hout = 0.2,   // Шаг вывода решения
        eps = 0.0001; // Абсолютная погрешность

 //////////////////////////
 // Вывод начальных условий
 cout << "Начальные значения: x0, y0\n"
      << "--------------------------\n"
      << setw (10) << x << setw (10) << y << endl;

 ////////////////////////////////////////////
 // Интегрирование дифференциального уравнения
 cout << "Решение: x, y (вычисленное), y (известное)\n"
      << "-----------------------------------------\n";
 do
   {
    y = Merson (x,x + hout,y,h,hmin,eps,UserFunc,&err);

    x += hout;

    ////////////////////////////////////////////////////
    // Вывод численного и известного аналитического решения
    cout << setw (10) << x
         << setw (10) << y
         << setw (10) << (exp (-x)) << endl;
   } while (x <= xend);

 ////////////////////
 // Анализ результата
 if (!err)
   cout << "Решение завершено с допустимыми параметрами\n"
        << "Гарантируется погрешность " << eps << endl;
 else
   cout << "Решение завершено с минимальным шагом\n"
        << "Погрешность " << eps << "не гарантируется\n";
}

double Merson (double x,double xend,double y, double h,double hmin,double
eps,userFuncOne fff,int *err)
```

```
//
// Функция интегрирования методом Мерсона одного обыкновенного
// дифференциального уравнения.
// Обозначения:
//     x — начало интервала интегрирования,
//  xend — конец интервала интегрирования,
//     y — значение функции в точке x,
//     h — заданный шаг интегрирования,
//  hmin — минимально допустимый шаг,
//   eps — заданная абсолютная погрешность,
//   fff — имя функции правой части уравнения,
//  *err — параметр, равный 0 при нормальном окончании расчета,
//         в этом случае заданная погрешность гарантируется;
//         равный -2 при достижении минимально допустимого  шага,
//         в этом случае заданная погрешность не гарантируется.
// Возвращаемое значение:
//  значение функции в точке xend
//
{
 double k1,k2,k3,k4,k5, // Параметры схемы решения
        xn = x,yn = y,  // Параметры предыдущей точки
        d;              // Рабочая переменная

 *err = 0;

do
  {
   ////////////////////////////////////////
   // Контроль конца участка интегрирования
   d = xend — x;
   if (d < h)
     {
      ////////////////////////////
      // Контроль минимального шага
      if (d <= hmin)
        {
         ////////////////////////////////
         // Восстановить предыдущую точку
```

```
        x = xn;
        y = yn;

        ////////////////////////////////////////
        // Взять шаг, равный рабочему + добавок,
        // не превышающий hmin
        h += d;
      }
    else
      h = d;
  }

////////////////////////////////////////////
// Сохранение информации о предыдущей точке
xn = x;
yn = y;

//////////////////////////
// Один шаг интегрирования
k1 = h * fff (x,y);
k2 = h * fff (x + h / 3,y + k1 / 3);
k3 = h * fff (x + h / 3,y + k1 / 6 + k2 / 6);
k4 = h * fff (x + h / 2,y + 0.125 * k1 + 0.375 * k3);
k5 = h * fff (x + h,y + k1 / 2 — 1.5 * k3 + k4 * 2);

//////////////////////
// Контроль погрешности
d = k1 / 6 + k4 * 2 / 3 + k5 / 6;
if (!(*err) && fabs (k1 / 2 — 1.5 * k3 + k4 * 2 — d) > eps)
  {
   ////////////////////////////
   // Уменьшить длину шага вдвое
   h /= 2;

   ////////////////////////////
   // Контроль минимального шага
   if (h < hmin)
```

```
        {
         h = hmin;
         *err = -2;
        }

      ////////////////////////////////
      // Выполнить шаг решения заново
      x = xn;
      y = yn;
     }
   else
     {
      ////////////////////////////////////////////////
      // Продолжить решение с допустимыми параметрами
      x += h;
      y += d;
     }

  } while (x < xend);

 return y;
}
```

Формулы метода Мерсона, предусматривающие увеличение длины шага интегрирования в случае слишком малой вычисленной погрешности, имеют вид:

$y_{n+1} = y_n + (k_1 + 4k_4 + k_5) / 2,$

где

$k_1 = hf(x_n, y_n) / 3,$

$k_2 = hf(x_n + h / 3, y_n + k_1) / 3,$

$k_3 = hf(x_n + h / 3, y_n + k_1 / 2 + k_2 / 2) / 3,$

$k_4 = hf(x_n + h / 2, y_n + 3k_1 / 8 + 9k_3 / 8) / 3,$

$k_5 = hf(x_n + h, y_n + 3k_1 / 2 — 9k_3 / 2 + 6k_4) / 3.$

Шаг решения выбирается автоматически, в зависимости от оценки абсолютной погрешности решения, данной формулой

$\varepsilon' = (k_1 - 9k_3 / 2 + 4k_4 - k_5 / 2) / 5.$

Возможны три случая:

1. если $\varepsilon' > \varepsilon$, точность неудовлетворительная, шаг уменьшается в два раза, и вычисление *повторяется* с новым шагом;
2. если $\varepsilon' \le \varepsilon / 32$, точность чрезмерно высока, шаг должен быть увеличен в два раза, и вычисление *продолжается* с новым шагом;
3. если $\varepsilon / 32 < \varepsilon' \le \varepsilon$, точность в заданных пределах, шаг выбран верно, и вычисление *продолжается* с текущим шагом.

Представленная в листинге 9.9 функция иллюстрирует рассмотренный подход. Достоинством функции является то, что найденный ею оптимальный шаг интегрирования включен в число выдаваемых функцией параметров. Следовательно, он может быть использован в последующих вызовах, чем способствовать сокращению времени счета. Подстройка шага осуществляется на начальном этапе решения, а последующее решение производится с фиксированным шагом, используя лишь незначительное число подстроек. Это актуально при решении таких объемных вычислительных задач, как идентификация параметров математических моделей. Выполнять задачу не рекомендуется на VBA, поэтому покажем только вариант функции на алгоритмическом языке C++.

Листинг 9.9

```
#include <fstream.h>
#include <iomanip.h>
#include <math.h>

typedef double (*userFuncOne) (double,double);

double UserFunc (double,double);
double RungeKuttaMerson
(double,double,double,double*,double,double,userFuncOne,int*);

void main (void)
{
 int err;              // Код ошибки

 double x = 0,         // Текущая точка вывода решения
        y = 1,         // Значение функции в текущей точке
        xend = 3,      // Конечная точка решения
        h = 0.1,       // Текущий шаг решения
        hmin = 0.001,  // Минимально допустимый шаг решения
```

```
        hout = 0.2,   // Шаг вывода решения
        eps = 0.0001; // Заданная погрешность

 //////////////////////////
 // Вывод начальных условий
 cout << "Начальные значения: x0, y0\n"
      << "--------------------------\n"
      << setw (10) << x << setw (10) << y << endl;

 ////////////////////////////////////////////
 // Интегрирование дифференциального уравнения
 // в заданных интервалах
 cout << "Решение: x, y (вычисленное), y (известное)\n"
      << "-----------------------------------------\n";
 do
   {
    ////////////////////////////////////////////////
    // Начальные установки для текущего шага решения
    y = Merson (x,x + hout,y,&h,hmin,eps,UserFunc,&err);

    x += hout;

    ////////////////////////////////////////////////////
    // Вывод численного и известного аналитического решения
    cout << setw (10) << x
         << setw (10) << y
         << setw (10) << (exp (-x)) << endl;
   } while (x <= xend);

 ////////////////////////////////////////
 // Вывод оптимального шага и кода ошибки
 cout << "Оптимальный шаг = " << h << "\tКод ошибки = " << err << endl;
}

double RungeKuttaMerson (double x,double xend,double y, double* h,double
hmin,double eps,userFuncOne fff,int *err)
//
// Функция интегрирования методом Рунге-Кутта-Мерсона одного
// обыкновенного дифференциального уравнения первого порядка.
```

```
//     x — начало интервала интегрирования,
//  xend — конец интервала интегрирования,
//     y — значение функции в точке x,
//    *h — при входе начальный шаг интегрирования,
//         при выходе — оптимальный с точки зрения точности шаг,
//  hmin — минимально допустимая величина шага,
//   eps — заданная погрешность,
//   fff — определяемая пользователем функция правой части уравнения,
//  *err — параметр, равный 0 при нормальном окончании  расчета, равный -
2
//         при достижении минимально допустимого  шага итераций, в этом
//         случае заданная погрешность не гарантируется.
// Возвращаемое значение:
//  значение функции в точке xend.
//
{
 double k1,k2,k3,k4,k5,
        eps5 = eps * 5,         // Максимальная погрешность
        eps532 = 0.15625 * eps, // Минимальная погрешность
        epsx,xn;                // Рабочие переменные

 *err = 0;

label1:

 if (xend — x < *h)
   *h = xend — x;

 xn = x;

 k1 = *h * fff (x,y) / 3;
 x += *h / 3;
 k2 = *h * fff (x,y + k1) / 3;
 k3 = *h * fff (x,y + k1 / 2 + k2 / 2) / 3;
 x += *h / 6;
 k4 = *h * fff (x,y + 0.375 * k1 + 1.125 * k3) / 3;
 x += *h / 2;
 k5 = *h * fff (x,y + 1.5 * k1 — 4.5 * k3 + k4 * 6) / 3;
```

```
 epsx = k1 — 4.5 * k3 + k4 * 4 — k5 / 2;

 if (*err == 0)
   {
    if (epsx > eps5)
      {
       *h /= 2;
       if (*h < hmin)
         {
          *h = hmin;
          *err = -2;
         }
       x = xn;
       goto label1;
      }
   if (epsx < eps532)
     *h *= 2;
  }

 y += (k1 + k4 * 4 + k5) / 2;

 if (x < xend)
   goto label1;

 return  y;
}
```

Популярны и другие методы решения с контролем погрешности, самый известный из которых — шестиэтапный метод четвертого порядка Фельберга. По опыту наших расчетов, метод Фельберга не показал преимуществ перед методом Мерсона.

Неявные схемы

Для неявных схем отличие от явных схем в вычислении коэффициентов, входящих в общую формулу одношаговых методов, следующее

$$k_r = f(x + \alpha_r h, y + h\sum_{s=1}^{m}\beta_{r,s}k_s), r = 1,3,\ldots,m,$$

где $\alpha_{..}, \beta_{..}$ — константы.

Неявные методы ориентированы на решение жестких систем. Они дают хорошие результаты и для нежестких систем.

Метод Хаммера–Холлингсуорта

Рассмотрим двухэтапный неявный метод четвертого порядка Хаммера—Холлингсуорта (Хаммера—Холлингсворта):

$y_{n+1} = y_n + (k_1 + k_2) / 2,$

где

$k_1 = hf(x_n + (1/2 - \sqrt{3}/6)h, y_n + k_1 / 4 + (1/4 - \sqrt{3}/6)k_2)$

$k_2 = hf(x_n + (1/2 + \sqrt{3}/6)h, y_n + (1/4 + \sqrt{3}/6)k_1 + k_2 / 4).$

Две последних формулы определяют итерационный процесс, для нежестких уравнений сходящийся очень быстро. Для таких систем представленный неявный алгоритм не эффективнее классического метода Рунге—Кутта: хотя здесь и требуется в два раза меньше вычислений правых частей дифференциальных уравнений, однако производиться эти вычисления должны в цикле. В рассматриваемом примере для достижения заданной точности потребовалось три итерации. Таким образом, количество вычислений правых частей увеличилось в 1,5 раза относительно явного метода Рунге—Кутта. Однако в случае жестких систем, по данным литературы, явные одношаговые методы не работают вообще, а с помощью неявных одношаговых методов решение получить иногда удается.

Методика вызова, как и результаты расчета того же тестового примера, с помощью функции `HammerHollingsworth` полностью совпадают с реализацией классической формулы Рунге—Кутта, поэтому пример применения не приводится.

Листинг 9.10

```
Private Function HammerHollinsworth(X As Double, Y As Double, _
Xout As Double, Hbeg As Double, Hmin As Double, Iter As Integer, _
Epsilon As Double, RangeX As Variant, RangeY As Variant, _
RangeRight As Variant) As Double
'
' Функция интегрирования обыкновенного дифференциального уравнения
' первого порядка методом Хаммера-Холлингсуорта
' Обозначения:
'  X — начало интервала интегрирования
```

```
'  Y — значение функции в точке X
'  Xout — конец интервала интегрирования (точка вывода)
'  Hbeg — заданный шаг интегрирования
'  Hmin — минимально допустимый шаг интегрирования
'  Iter — число итераций
'  Epsilon — абсолютная погрешность
'  RangeX — ячейка аргумента
'  RangeY — ячейка функции
'  RangeRight — ячейка формулы правой части
' Возвращаемое значение:
'  значение функции в точке Xout.
'
 Dim K1 As Double ' Вспомогательная величина
 Dim K2 As Double ' Вспомогательная величина
 Dim S1 As Double ' Величина для контроля сходимости
 Dim S2 As Double ' Величина для контроля сходимости
 Dim SS As Double ' Вспомогательная константа
 Dim H As Double  ' Шаг решения
 Dim Xs As Double ' Сохраненное значение
 Dim Ys As Double ' Сохраненное значение
 Dim I As Integer ' Счетчик
 '
 ' Начальные значения параметров
 '
 K1 = 0
 K2 = 0
 SS = Sqr(3) / 6
 H = Hbeg
 Xs = X
 Ys = Y
 '
 ' Выполнять решение, пока не будет достигнута точка вывода
 '
 Do While X < Xout
   '
   ' Коррекция длины шага, при необходимости
   '
   If X + H > Xout Then
     H = Xout — X
```

```
      If H < Hmin Then
        X = Xs
        Y = Ys
        H = Xout - X
      End If
    End If
    Ys = Y
    '
    ' Интегрирование уравнения
    '
    For I = 0 To Iter
     S1 = K1
     S2 = K2
     K1 = H * Right(X + H * (0.5 - SS), Y + K1 * 0.25 + K2 * (0.25 - SS),
_
          RangeX, RangeY, RangeRight)
     K2 = H * Right(X + H * (0.5 + SS), Y + K1 * (0.25 + SS) + K2 * 0.25,
_
          RangeX, RangeY, RangeRight)
     '
     ' Сходимость достигнута
     '
     If Abs(S1 + S2 - K1 - K2) < Epsilon Then
       Exit For
     End If
    Next I
    Y = Y + (K1 + K2) / 2
    Xs = X
    X = X + H
  Loop
  HammerHollinsworth = Y
End Function
```

В листинге 9.11 представлен функциональный аналог на алгоритмическом языке C++. В реализации производится 10 итераций. Значение числа итераций может быть заменено либо вынесено в формальные параметры функции, как это сделано в VBA, однако, по опыту расчетов, сходимость всегда достигалась задолго до исчерпания данной величины.

Листинг 9.11

```
double HammerHollingsworth (double x,double y, double h,userFuncOne fff)
//
// Функция интегрирования методом Хаммера-Холлингсуорта одного
// обыкновенного дифференциального уравнения.
// Обозначения:
//  x — начало интервала интегрирования,
//  y — значение функции в точке x,
//  h — шаг интегрирования.
// Возвращаемое значение:
//  значение функции в точке x + h.
//
{
 double k1 = 0,k2 = 0,      // Рабочие переменные
        s1,s2,              // Переменные для контроля сходимости
        ss = sqrt (3) / 6; // Вспомогательная константа

 ////////////////////////////////////////
 // Выполнить заданное число (10) итераций
 for (register int i = 0; i < 10; i++)
   {
    s1 = k1;
    s2 = k2;

    k1 = h * fff (x + h * (0.5 — ss),y + k1 * 0.25 + k2 * (0.25 — ss));
    k2 = h * fff (x + h * (0.5 + ss),y + k1 * (0.25 + ss) + k2 * 0.25);

    ///////////////////////
    // Сходимость достигнута
    if (fabs (s1 + s2 — k1 — k2) < 0.000001)
      break;
   }

 return y + (k1 + k2) / 2;
}
```

Многошаговые методы

Многошаговые методы оперируют значениями функции в нескольких предыдущих точках. Общая формула линейного k-шагового метода имеет вид

$$y_{n+1} = \sum_{i=1}^{k} \alpha_i y_{n+1-i} + h \sum_{i=0}^{k} \beta_i f_{n+1-i} ,$$

где α_i — коэффициенты перед значениями функции,

β_i — коэффициенты перед значениями правых частей.

Для вычисления y_{n+1} должны быть известны все или некоторые (в зависимости от метода) величины y_n, y_{n-1}, ... ,y_{n-k+1}, f_n, f_{n-1}, f_{n-k+1}.

Бывают явные и неявные многошаговые методы. Некоторые многошаговые методы включают в себя два шага: явный, называемый предиктором, и неявный (или несколько неявных), называемый корректором.

Многошаговые методы показали свою эффективность при решении жестких систем.

Явные схемы

При $\beta_0 = 0$ метод называется явным. Рассмотрен классический метод Адамса. Другие методы широко представлены в литературе и могут быть запрограммированы на основе функции `Adams`.

Метод Адамса

Формула метода Адамса четвертого порядка имеет вид

$y_{n+1} = y_n + h(55f_n - 59f_{n-1} + 37f_{n-2} - 9f_{n-3}) / 24.$

В варианте предиктор-корректор данная формула дополняется выражением

$y_{n+1} = y_n + h(9f_{n+1} + 19f_n - 5f_{n-1} + f_{n-2}) / 24.$

В листинге 9.12 показана функция, производящая интегрирование одного дифференциального уравнения методом Адамса. Первые четыре точки решения получены методом Рунге—Кутта (может быть использован и другой метод). По сравнению с одношаговыми методами, возвращение на один шаг назад, которое необходимо сделать в конце интервала решения, если шаг вывода решения не кратен шагу сетки решения, в многошаговых методах потребовало не то что бы больших усилий, но увеличило бы объем программного кода. Поэтому здесь вместо шага сетки задается число узлов сетки, а нужный шаг сетки вычисляется программой автоматически. Соответственно, отпадает необходимость контроля конца интервала — достаточно выполнить заданное число циклов, равное числу узлов разностной сетки.

Листинг 9.12

```
Private Function Adams (X As Double, Y As Double, Xend As Double, _
Node As Integer, RangeX As Variant, RangeY As Variant, _
RangeRight As Variant) As Double
'
' Функция интегрирования обыкновенного дифференциального уравнения
' первого порядка методом Адамса
' Обозначения:
'  X — начало интервала интегрирования
'  Y — значение функции в точке X
'  Xend — конец интервала интегрирования (точка вывода)
'  Node — число узлов сетки
'  RangeX — ячейка аргумента
'  RangeY — ячейка функции
'  RangeRight — ячейка формулы правой части
' Возвращаемое значение:
'  значение функции в точке Xend.
'
 Dim I As Integer   ' Счетчик
 Dim F(4) As Double ' Параметры
 Dim H As Double    ' Шаг сетки
 '
 ' Вычисление параметров решения
 '
 H = (Xend — X) / Node
 '
 ' Старт метода
 '
 For I = 0 To 3
   F(3 — I) = Right(X, Y, RangeX, RangeY, RangeRight)
   Y = RungeKutta(X, Y, H, RangeX, RangeY, RangeRight)
   X = X + H
 Next I
 '
 ' Интегрирование уравнения
```

```
 '
 For I = 4 To Node - 1
   F(0) = Right(X, Y, RangeX, RangeY, RangeRight)
   X = X + H
   Y = Y + H * (55 * F(0) - 59 * F(1) + 37 * F(2) - 9 * F(3)) / 24
   F(3) = F(2)
   F(2) = F(1)
   F(1) = F(0)
 Next I
 Adams = Y
End Function
```

Аналог на C++ с примером применения показан в листинге 9.13.

Листинг 9.13

```
#include <fstream.h>
#include <iomanip.h>
#include <math.h>

typedef double (*userFuncOne) (double,double);

double UserFunc (double,double);
double Adams (double,double,double,int,userFuncOne);

void main (void)
{
 int n = 200;       // Число узлов сетки

 double x = 0,      // Начальная (затем текущая) точка интегрирования
        y = 1,      // Значение функции в точке x
        xend = 3,   // Конечная точка интегрирования
        hout = 0.2; // Шаг вывода решения

 ////////////////////////
 // Вывод начальных условий
 cout << "Начальные значения: x0, y0\n"
      << "-------------------------\n"
      << setw (10) << x << setw (10) << y << endl;
```

```
 ////////////////////////////////////////////
 // Интегрирование дифференциального уравнения
 cout << "Решение: x, y (вычисленное), y (известное)\n"
      << "----------------------------------------\n";
 do
   {
    y = Adams (x,x + hout,y,n,UserFunc);

    x += hout;

    ////////////////////////////////////////////////////
    // Вывод численного и известного аналитического решения
    cout << setw (10) << x << setw (10) << y
         << setw (10) << (exp (-x)) << endl;
   } while (x <= xend);
}

double Adams (double x,double xend,double y,int n,userFuncOne fff)
//
// Функция интегрирования методом Адамса одного обыкновенного
// дифференциального уравнения.
// Обозначения:
//     x — начало интервала интегрирования,
//  xend — конец интервала интегрирования,
//     y — значение функции в точке x,
//     n — число узлов сетки,
//   fff — имя функции правой части уравнения.
// Возвращаемое значение:
//  значение функции в точке xend
//
{
 double f[4],                // Параметры
        a[] = {55,59,37,9}, // Коэффициенты
        h = (xend — x) / n; // Шаг сетки

 ///////////////
```

```
 // Старт метода
 for (register int i = 0; i <= 3; i++)
   {
    f[3 — i] = fff(x,y);
    y = RungeKutta (x,y,h,fff);
    x += h;
   }

 ///////////////////////////
 // Интегрирование уравнения
 for (; i < n; i++)
   {
    f[0] = fff (x,y);
    x += h;
    y += h * (a[0] * f[0] — a[1] * f[1] + a[2] * f[2] — a[3] * f[3]) /
24;
    f[3] = f[2];
    f[2] = f[1];
    f[1] = f[0];
   }

 return y;
}
```

Неявные схемы

Если $\beta_0 \neq 0$, то метод будет называться неявным. Рассмотрен метод Гира четвертого порядка.

Методы Гира

Для частных случаев рассматриваемый метод введен Кертисом и Хиршфельдером (Гиршфельдером). Общий случай рассмотрел Гир. Метод Гира использует идею так называемого обратного дифференцирования. Для рассматриваемого случая формула имеет вид

$$25/12\ y_{n+1} - 4y_n + 3y_{n-1} - 4/3\ y_{n-2} + 1/4\ y_{n-3} = hf_{n+1}.$$

Процесс, входящий в схему метода, быстро сходится.

Методы Гира применяются для интегрирования жестких дифференциальных уравнений. Для нежестких систем получены также хорошие результаты.

В листинге 9.14 решение начинается с помощью метода Рунге—Кутта. Это не лучший старт, особенно, если система является жесткой в начале интервала. В тестовых задачах, впрочем, каких-либо проблем не возникало.

Листинг 9.14

```
Private Function Gear(X As Double, Y As Double, Xend As Double, _
Node As Integer, RangeX As Variant, RangeY As Variant, _
RangeRight As Variant) As Double
'
' Функция интегрирования обыкновенного дифференциального уравнения
' первого порядка методом Гира 4 порядка
' Обозначения:
'  X — начало интервала интегрирования
'  Y — значение функции в точке X
'  Xend — конец интервала интегрирования (точка вывода)
'  Node — число узлов сетки
'  RangeX — ячейка аргумента
'  RangeY — ячейка функции
'  RangeRight — ячейка формулы правой части
' Возвращаемое значение:
'  значение функции в точке Xend.
'
 Dim I As Integer    ' Счетчик
 Dim YY(3) As Double ' Решения в предыдущих 3-х точках сетки
 Dim H As Double     ' Шаг сетки
 Dim T As Double     ' Решение в 5-й точке
 '
 ' Вычисление параметров решения
 '
 H = (Xend — X) / Node
 '
 ' Старт метода
 '
 For I = 0 To 2
   YY(I) = Y
   Y = RungeKutta(X, Y, H, RangeX, RangeY, RangeRight)
   X = X + H
 Next I
```

```
 T = RungeKutta(X, Y, H, RangeX, RangeY, RangeRight)
 '
 ' Интегрирование уравнения
 '
 For I = 3 To Node - 1
   X = X + H
   '
   ' Выполнить 2 итерации
   '
   T = 1.92 * Y + 0.48 * H * Right(X - H, T, RangeX, RangeY,
RangeRight) - _
       1.44 * YY(2) + 0.64 * YY(1) - 0.12 * YY(0)
   T = 1.92 * Y + 0.48 * H * Right(X - H, T, RangeX, RangeY,
RangeRight) - _
       1.44 * YY(2) + 0.64 * YY(1) - 0.12 * YY(0)
   YY(0) = YY(1)
   YY(1) = YY(2)
   YY(2) = Y
   Y = T
 Next I
 Gear = Y
End Function
```

Далее показана функция `Gear`, аналогичная представленной, но составленная на языке C++.

Листинг 9.15

```
double Gear (double x,double xend,double y,int n,userFuncOne fff)
//
// Функция интегрирования методом Гира 4 порядка одного обыкновенного
// дифференциального уравнения.
// Обозначения:
//     x - начало интервала интегрирования,
//  xend - конец интервала интегрирования,
//     y - значение функции в точке x,
//     n - число узлов сетки,
//   fff - имя функции правой части уравнения.
```

```
// Возвращаемое значение:
//  значение функции в точке xend
//
{
 double yy[3],              // Решения в предыдущих точках сетки
        h = (xend — x) / n, // Шаг сетки
        t;                  // Решение в (n+1)-й точке

 ///////////////
 // Старт метода
 for (register int i = 0; i < 3; i++)
   {
    ////////////////////////////////
    // Решение в первых трех точках
    yy[i] = y;

    //////////////////////////
    // Решение в текущей точке
    y = RungeKutta (x,y,h,fff);
    x += h;
   }

 ////////////////////////////////////
 // Приближение к решению в 5-й точке
 t = RungeKutta (x,y,h,fff);

 ////////////////////////////
 // Интегрирование уравнения
 for (; i < n; i++)
   {
    x += h;

    ///////////////////////
```

```
    // Выполнить 2 итерации
    t = 1.92 * y + 0.48 * h * fff (x,t) – 1.44 * yy[2] +
        0.64 * yy[1] – 0.12 * yy[0];
    t = 1.92 * y + 0.48 * h * fff (x,t) – 1.44 * yy[2] +
        0.64 * yy[1] – 0.12 * yy[0];

    yy[0] = yy[1];
    yy[1] = yy[2];
    yy[2] = y;
    y = t;
   }

 return y;
}
```

Идентификация параметров дифференциальных уравнений

Математическое моделирование может включать в себя этап подстройки (идентификации) неизвестных параметров и неизвестных начальных условий в соответствии с данными эксперимента.

Рассмотрим случай, когда уравнения математической модели записаны в виде системы m обыкновенных дифференциальных уравнений

$y' = f(x,y,p)$,

где p — вектор параметров.

Задача решается с начальными условиями: при $x = x_0$ полагают $y = y_0$, и похожа на задачу Коши, однако все или некоторые начальные условия могут быть неизвестны, как могут быть неизвестны все или некоторые параметры, составляющие вектор p.

Идентификацией модели назовем определение по данным эксперимента компонент вектора параметров и начальных условий. Для этого применим модификацию метода Ньютона, известную под названием алгоритма чувствительности. Функции чувствительности четкий физический смысл. Они показывают, насколько чувствительно решение системы уравнений к изменению параметров системы. Уравнения для вычисления функций чувствительности запишем в виде:

$$\frac{dW_P}{dx} = \frac{\partial f(x,y,p)}{\partial y} W_P + \frac{\partial f(x,y,p)}{\partial p},$$

$$\frac{dW_\theta}{dx} = \frac{\partial f(x,y,p)}{\partial y} W_\theta,$$

где W_P — матрица функций чувствительности системы к параметрам,

W_θ — матрица функций чувствительности системы к начальным условиям,

θ — вектор идентифицируемых начальных условий, в него могут быть включены все или некоторые неизвестные начальные условия задачи.

Показанная система дифференциальных уравнений решается совместно с решением дифференциальных уравнений математической модели, например, методом Рунге—Кутта, с учетом начальных условий для функций чувствительности при $x = x_0$:

- для неидентифицируемых начальных условий $W_P = 0$, $W_\theta = 0$;
- для идентифицируемых начальных условий $W_\theta = 1$.

Обозначим, для простоты записи, через β вектор, составленный из двух векторов: p и θ.

Квадратичный функционал качества идентификации запишем в виде

$$I_N = \sum_{k=1}^{N} (\eta^*(x_k) - \eta^l(x_k))^2,$$

где η^* и η — векторы выходов эксперимента и математической модели,

N — количество опытных точек и точек интегрирования системы.

Минимизация функционала качества идентификации

$$I_N \to \min$$

дает систему N линейных алгебраических уравнений для определения вектора приращения $\Delta\beta^{l+1}$, полученного на l-м шаге итераций:

$$\left\{ \sum_{k=1}^{N} \left(\frac{d\eta^l(x_k)}{d\beta}\right)^T V \frac{d\eta^l(x_k)}{d\beta} \right\} \Delta\beta^{l+1} = \sum_{k=1}^{N} \left(\frac{d\eta^l(x_k)}{d\beta}\right)^T V (\eta^*(x_k) - \eta^l(x_k)),$$

где $\frac{d\eta^l}{d\beta} = \left(\frac{\partial \eta^l}{\partial x} W_P^l; \frac{\partial \eta^l}{\partial x} W_\theta^l\right)$,

V — диагональная весовая матрица.

Используемые в формулах матрицы частных производных могут быть получены методом центральных разностей или эффективным методом матрицы планирования эксперимента, представленным в нашей первой книге.

Весовая матрица предназначается для искусственного уравнивания чувствительности системы к параметрам. Элементы матрицы $diagV = [\nu_1, \nu_2, \ldots, \nu_m]$ предлагается определять автоматически на первом шаге итерации по формуле:

$$\nu_i = 1/\eta_i^{*\max},$$

где $\eta_i^{*\max}, i = 1,2,\ldots,m,$ — максимальные значения выходов эксперимента.

Решение производится до тех пор, пока не будет достигнута заданная исследователем точность. Вектор β^{l+1} на *(l*+1*)*-м шаге итераций определяется как

$$\beta^{l+1} = \beta^l + \Gamma^{l+1}\Delta\beta^{l+1},$$

где Γ^{l+1} — диагональная матрица длины шага итераций.

Элементы матрицы $diag\Gamma^{l+1} = [\gamma_1^{l+1}, \gamma_2^{l+1}, \ldots, \gamma_n^{l+1}]$ определим следующим образом. Введем величины

$$\Delta_i^{l+1} = \left|\Delta\beta_i^{l+1}/\Delta\beta_i^l\right|, i = 1,2,\ldots,n,$$

а также постоянные $\Delta_i^{\max}, i = 1,2,\ldots,n,$ определяемые эмпирически для конкретной системы. Тогда:

$$\gamma_i^{l+1} = \Delta_i^{\max}/\Delta_i^{l+1}, \text{ если } \Delta_i^{\max}/\Delta_i^{l+1} \le 1,$$

$$\gamma_i^{l+1} = 1, \text{ если } \Delta_i^{\max}/\Delta_i^{l+1} > 1.$$

Введение величин $\gamma_i^{l+1}, i = 1,2,\ldots,n,$ вместо скаляра, а также постоянных $\Delta_i^{\max}, i = 1,2,\ldots,n,$ обусловлено разной чувствительностью системы к изменению различных параметров. Так, для слабо чувствительных параметров приращения $\Delta\beta^{l+1}$ оказываются чрезмерно большими, и, для обеспечения сходимости процесса идентификации параметров, требуется уменьшать шаг итераций, что приводит к уменьшению скорости сходимости.

Рассмотренный метод позволяет вычислять неизвестные параметры дифференциальных уравнений, моделирующих реальную физическую систему, что позволяет по выходу эксперимента делать вывод о значении параметров в реальной системе. Например, при использовании адекватной математиче-

ской модели могут быть оперативно вычислены инерционные параметры объекта и спрогнозирована дальнейшая траектория, по нескольким имеющимся предшествующим измерениям траектории его движения.

О приложении дифференциальных уравнений к вычислению статистических распределений

Методы решения обыкновенных дифференциальных уравнений могут применяться в качестве альтернативы методам численного интегрирования.

При решении статистических задач часто приходится вычислять статистические распределения (см. *приложение 2*). В качестве примера рассмотрим решение задачи вычисления функции стандартного нормального распределения с помощью методов решения обыкновенных дифференциальных уравнений.

В случае прямого интегрирования, методы вычисления определенных интегралов могут иметь ряд трудностей:

- Прямое интегрирование не может обеспечить оперативный контроль погрешности вычислений. Чтобы получить оценку погрешности, необходимо, как минимум дважды, вычислить значение интеграла. Такой контроль могут обеспечить методы решения обыкновенных дифференциальных уравнений с контролем погрешности.
- Методы вычисления определенных интегралов неудовлетворительно работают для некоторых типов жестких систем. Например, это t-распределение и F-распределение при малых значениях параметров. Полезными для вычисления данных распределений могут оказаться методы обыкновенных дифференциальных уравнений.

Трудности разрешимы путем применения методов решения обыкновенных дифференциальных уравнений. Показанный в листинге 9.16 пример показывает вычисление стандартного нормального распределения двумя методами:

- методом Симпсона (функция `NormalDistribution1`, вызывающая функцию вычисления определенного интеграла методом Симсона);
- методом Рунге—Кутта—Мерсона (функция `NormalDistribution2`, вызывающая функцию решения обыкновенного дифференциального уравнения методом Мерсона).

Для стандартного вызова функции `RungeKuttaMerson` требуется ввести измененную (введен фиктивный параметр) функцию `NormalDensity` вычисления плотности вероятности стандартизированной случайной величины.

Чтобы обеспечить заданную точность, нижний предел интегрирования и число узлов интегрирования для метода Симпсона подобраны эмпирически. Для метода Рунге—Кутта—Мерсона просто указана требуемая погрешность. Некоторые функции на листинге не показаны, но даны на прилагаемом компакт-диске.

Листинг 9.16

```
#include <fstream.h>
#include <iomanip.h>
#include <math.h>

#define M_PI 3.14159265358979323846 // Число "пи"

typedef double (*userFunc) (double);
typedef double (*userFuncOne) (double,double);

double NormalDensity (double);
double NormalDensity (double,double);
double NormalDistribution1 (double);
double NormalDistribution2 (double);
double RungeKuttaMerson
(double,double,double,double*,double,double,userFuncOne,int*);
double Simpson (double,double,int,userFunc);
int Parity (int);

void main (void)
{
 double t = 1.837117;

 cout << "Simpson: " <<  NormalDistribution1 (t)
      << "\nMerson: " <<  NormalDistribution2 (t) << endl;
}

double NormalDistribution2 (double x)
//
// Функция вычисления функции стандартного нормального распределения
// методом Рунге-Кутта-Мерсона.
```

```
// Обозначения:
//  x — стандартизированная случайная величина.
// Возвращаемое значение:
//  уровень значимости.
//
{
 int err;

 double h = 0.001;

 return RungeKuttaMerson (0,x,0,&h,0.0001,0.000001,NormalDensity,&err) /
sqrt (M_PI * 2) + 0.5;
}

double NormalDensity (double x,double t)
//
// Функция вычисления плотности вероятности стандартизированной
// случайной величины.
// Обозначения:
//  x — стандартизированная случайная величина,
//  t — фиктивный параметр.
// Возвращаемое значение:
//  значение плотности вероятности.
//
{
 return 1.0 / exp (x * x / 2);
}
```

Ошибки численных схем

Разностные формулы характеризуются ошибками двух типов: ошибками отбрасывания и ошибками округления. Ошибки отбрасывания принято характеризовать порядком ошибки $O(h^n)$. Все разностные схемы вычисления первых и вторых производных, построенные на правых или левых разностях, имеют ошибку $O(h)$. Схемы, построенные на центральных разностях, имеют ошибку $O(h^2)$. Схемы методов Рунге—Кутта четвертого порядка имеют ошибку $O(h^5)$.

Методы с теоретической ошибкой более высокого порядка малости при практических вычислениях не всегда лучше методов с ошибкой меньшего порядка.

Было бы неверно думать, что чем меньше шаг решения h, тем меньше ошибка. Одновременно с уменьшением ошибки отбрасывания, при уменьшении шага решения h, увеличивается ошибка округления, связанная с представлением данных в компьютере. При создании программ следует учитывать, что общая ошибка достигает минимума при некотором оптимальном значении шага решения.

Руководство пользователя и пример применения

Для работы с модулем дифференциальных уравнений (DEQ) выберите из меню Microsoft Excel пункт **ME.com ▸ Модуль DEQ ▸ Дифференциальные уравнения**. На экране появится диалоговое окно, изображенное на рис. 9.1.

Рис. 9.1. Окно модуля дифференциальных уравнений

1. Выберите или введите ячейку, содержащую начальное значение аргумента.
2. Выберите или введите ячейку, содержащую начальное значение функции.
3. Выберите или введите ячейку, содержащую формулу правой части уравнения. Контроль допустимости введенной правой части и ее соответствие правилам записи формул Microsoft Excel осуществляется стандартным образом. Рисунок 9.2 показывает, каким образом вводится формула правой части. Так, в ячейку B3 введено начальное значение аргумента X,

в ячейку B4 введено начальное значение функции *Y*. В ячейку B5 введено выражение правой части решенного в примере дифференциального уравнения *Y'*=-*Y*, а именно, =—B4. Легко заметить, что в ячейку формулы правой части в качестве значений *X* и *Y* вводятся адреса соответствующих ячеек.

	A	B	C	D	E	F	G	H
1								
2								
3	X	0	Решение уравнения (аргумент x, функция y=y(x))					
4	Y	1	Метод Рунге-Кутта					
5	F(X,Y)	-1	0	1				
6			0,1	0,904837				
7			0,2	0,818731				
8			0,3	0,740818				
9			0,4	0,67032				
10			0,5	0,606531				
11			0,6	0,548812				
12			0,7	0,496585				
13			0,8	0,449329				
14			0,9	0,40657				
15			1	0,367879				
16			1,1	0,332871				

Рис. 9.2. Пример ввода правой части дифференциального уравнения

4. Выберите или введите выходной интервал. Начиная с первой ячейки выходного интервала, будут выведены результаты вычислений.
5. Выберите или оставьте по умолчанию метод интегрирования дифференциального уравнения.
6. Выберите или оставьте по умолчанию параметры решения. Обратите внимания, что абсолютная погрешность действительна только для методов с автоматическим контролем точности.
7. Выберите или оставьте по умолчанию дополнительные параметры решения. Обратите внимание, что число итераций действительно только для методов, имеющих в схемах алгоритмов итерационные фрагменты, а число узлов сетки действительно только для многошаговых методов.
8. Нажмите кнопку **Выполнить расчет**.

После выполнения вычислений будет, начиная с первой ячейки выходного интервала, выведено название метода и результаты расчета.

За выбор адекватного исходным данным метода расчета несет ответственность пользователь. Модуль дифференциальных уравнений берет на себя верификацию исходных данных, выдавая подробную диагностику. При не-

верных действиях пользователя выдаются сообщения об ошибках. Обратим внимание, что не все сочетания параметров допустимы.

Время решения даже сравнительно простых по структуре дифференциальных уравнений различными методами может зависеть от производительности компьютерной системы, поэтому программой предусмотрено прерывание процесса, по желанию пользователя. О данной возможности пользователю сообщается в специальном информационном окне, показанном на рис.9.3, перед началом расчета.

Рис. 9.3. Информационное окно

Обратим внимание пользователя на очень интересный и полезный эффект при решении дифференциальных уравнений в среде Microsoft Excel. Решив уравнение, можно построить диаграмму стандартными средствами, благодаря прекрасным графическим возможностям Microsoft Excel. После этого можно удалить исходные данные, не удаляя диаграмму. Поле построенной диаграммы станет пустым. Затем повторим расчет с новыми параметрами. По мере заполнения области, определенной нами ранее в качестве исходных данных для диаграммы, результаты будут отражены в виде анимации.

Сообщения об ошибках

При ошибках ввода данных могут выдаваться диагностические сообщения следующих типов.

Рис. 9.4. Окно сообщения об ошибке

На рис.9.4 не выбран или неверно введен интервал ячеек, определяющих начальные значения или начало области вывода результатов расчета. Лучшим способом избежать ошибки является не ввод, а выделение интервала ячеек стандартным образом.

Рис. 9.5. Окно сообщения об ошибке

Проверьте на рис.9.5 содержимое ячейки рабочего листа, заданное как формула. Запись формулы должна соответствовать принятой в Microsoft Excel.

В процессе решения могут также выдаваться сообщения относительно недопустимого сочетания параметров.

Рис. 9.6. Окно сообщения об ошибке

На рис.9.6 задан минимальный шаг, превышающий шаг решения, или шаг вывода, меньший шага решения. Допустимые сочетания параметров установлены в полях формы, представленной на рис. 9.1, по умолчанию.

Рис. 9.7. Окно сообщения об ошибке

На рис.9.7 не задан один из требуемых параметров решения. Допустимые значения параметров установлены в полях формы, представленной на рис. 9.1. В случае сомнений, следует оставить параметры, установленные программой по умолчанию.

Рис. 9.8. Окно сообщения об ошибке в период выполнения

Более опасными, в смысле принципиальной невозможности получения решения задачи, чем перечисленные ошибки ввода данных, являются ошибки периода выполнения функций программного обеспечения DEQ, большинство из которых, надеемся, возникают достаточно редко. Данные ошибки являются отражением свойств решаемых уравнений. Функции модуля DEQ отслеживают все ошибки периода выполнения, выдавая сообщение, показанное на рис.9.8. В процессе решения дифференциального уравнения нужно быть готовым к тому, что иногда решение получить не удается.

Дальнейшие исследования

Дополнение вычислительного модуля DEQ возможностями решения систем дифференциальных уравнений.

Исследование возможности реализации быстрого комбинированного решения, состоящего из следующих этапов:

1. синтаксический разбор выражения правой части дифференциального уравнения, введенного в ячейку электронной таблицы;
2. генерация кода, составленного на основе полученных формул, на языке C и вывод его в текстовый файл;
3. вызов компилятора, работающего из командной строки, и создание модуля DLL, содержащего функцию правых частей;
4. регистрация DLL в модуле VBA;
5. решение уравнения с использованием функций, составленных на языке C, и вывод результатов в ячейки электронной таблицы.

Рекомендуемая литература

1. Numerical Recipes in C: The Art of Scientific Computing / W.H. Press, S.A. Teukolsky, W.T. Vetterling et al., New York: Cambridge University Press, 1992. Available from World Wide Web: < **http://www.nr.com** >.
2. Бабушка И., Витасек Э., Прагер М. Численные процессы решения дифференциальных уравнений. — М.: Мир, 1969.
3. Балантер Б.И., Ханин М.А., Чернавский Д.С. Введение в математическое моделирование патологических процессов. — М.: Медицина, 1980.

4. Васильков Ю.В., Василькова Н.Н. Компьютерные технологии вычислений в математическом моделировании. — М.: Финансы и статистика, 2002.

5. Вольтера В. Математическая теория борьбы за существование. — М.: Наука, 1976.

6. Выгодский М.Я. Справочник по высшей математике. — М.: Большая медведица, 2001.

7. Гайдышев И. Анализ и обработка данных: специальный справочник. — СПб: Питер, 2001.

8. Гантмахер Ф.Р. Теория матриц. — М.: Наука, 1988.

9. Губин В.Б. Об одном варианте принципа бритвы Оккама // Философские науки, № 2, 1998. С. 136-150. Available from World Wide Web: < **http://gubin.narod.ru/8.htm** >.

10. Кадыров Х.К., Антомонов Ю.Г. Синтез математических моделей биологических и медицинских систем. — Киев: Наукова думка, 1974.

11. Каханер Д., Моулер К., Нэш С. Численные методы и программное обеспечение. — М.: Мир, 2001. Source code available from World Wide Web: < **http://iris.gmu.edu/~snash/index.html** >

12. Кориков А.М. Математические методы планирования эксперимента. — Томск: ТГУ, 1973.

13. Краснощеков П.С., Петров А.А. Принципы построения моделей. — М.: ФАЗИС: ВЦ РАН, 2000.

14. Лампер Р.Е. Введение в теорию флаттера. — М.: Машиностроение, 1990.

15. Ланс Дж.Н. Численные методы для быстродействующих вычислительных машин. — М.: Издательство иностранной литературы, 1962.

16. Лоскутов А.Ю., Михайлов А.С. Введение в синергетику: Учебное руководство. — М.: Наука, 1990.

17. Мак-Кракен Д., Дорн У. Численные методы и программирование на ФОРТРАНе. — М.: Мир, 1977.

18. Марри Дж. Нелинейные дифференциальные уравнения в биологии. Лекции о моделях. — М.: Мир, 1983.

19. Математика в социологии: моделирование и обработка информации / Под ред. А.Г. Аганбегяна и Ф.М. Бородкина. — М.: Мир, 1977.

20. Математические моделирование / Под ред. Дж. Эндрюса и Р. Мак-Лоуна. — М.: Мир, 1979.

21. Математический энциклопедический словарь. — М.: Научное издательство «Большая Российская энциклопедия», 1995.

22. Орвис В. Дж. Excel для ученых, инженеров и студентов. — Киев: Юниор, 1999.
23. Рубан А.И. Идентификация нелинейных динамических объектов на основе алгоритма чувствительности. — Томск: ТГУ, 1975.
24. Современные численные методы решения обыкновенных дифференциальных уравнений / Под ред. Дж. Холла и Дж. Уатта. — М.: Мир, 1979.
25. Хайрер Э., Нерсетт С., Ваннер Г. Решение обыкновенных дифференциальных уравнений. Нежесткие задачи[2]. — М.: Мир, 1990.
26. Шустер Г. Детерминированный хаос: Введение. — М.: Мир, 1988.

[2] *В 1999 г. выпущен второй том, посвященный жестким задачам.*

ГЛАВА 10

Рекурсии на примере многочленов в виде XLL на XLM и C

«Итерация свойственна человеку, рекурсия — Богу».
Л. Питер Дойч[1]

Для работы с материалами главы понадобятся лицензионные средства разработки и тестирования программ:

- Microsoft Excel 97 или выше;
- Microsoft Excel Developer's Kit версии 8 или выше;
- Microsoft Visual C++ 5.0 или выше.

Потребуется знание XLM, C.

Решаемая проблема

Рекуррентным называется соотношение вида

$$a_{n+p} = F(n, a_n, a_{n+1}, ..., a_{n+p-1}),$$

позволяющее вычислять любой член последовательности, если заданы ее первые p членов. Рекуррентные соотношения вычисляются циклическими и рекурсивными алгоритмами. С циклическими алгоритмами мы встречались на протяжении всей книги. Рекурсивным называется алгоритм, в котором встречается вызов его самого.

---

[1] *Цит. по Страуструп Б. Язык программирования Си++. — М.: Радио и связь, 1991, с. 118.*

Рекурсивные функции являются распространенной программной реализацией рекуррентных соотношений. Так, например, в *главе 5*, при вычислении критерия рандомизации компонент Фишера для независимых выборок использована рекурсия для вычисления благоприятных сочетаний. В первоначальном варианте существовало несколько однотипных по структуре функций для различных параметров. Применение рекурсивной функции в несколько раз уменьшило объем программного кода. Примерами рекурсий являются функции вычисления многочленов, представленные в данной главе.

Особенности решения

Функции оформлены в виде модуля Polynom программного продукта ME.com — проект Mathematical and Engineering components (математические и инженерные компоненты). Программные особенности реализации в виде XLL подробно рассмотрены в *главе 2*. Особенностью данного решения является отсутствие интерфейсной части, кроме возможности вызова содержащихся в модуле функций непосредственно из рабочего листа электронных таблиц, и отсутствие кода VBA. Полные исходные тексты и дистрибутив находятся на прилагаемом компакт-диске.

Научные алгоритмы

Многочленом (полиномом) называют выражение общего вида

$$Ax^k y^l ... w^m + Bx^n y^p ... w^q + ... + Dx^r y^s ... w^t,$$

где $x, y, ..., w$ — переменные,

$A, B, ..., D$ — коэффициенты,

$k, l, ..., t$ — показатели степеней.

Многочлен от одной переменной можно записать в виде

$$P(x) = a_0 x^n + a_1 x^{n-1} + ... + a_{n-1} x + a_n = \sum_{k=1}^{n} a_k x^{n-k},$$

где n — степень многочлена, $a_0 \neq 0$.

Многочлены возникают в различных приложениях: при аппроксимации зависимостей, интерполяции, вычислении функций и т.д. Для асимптотических разложений статистических распределений применяются многочлены

Эрмита и Лагерра. Некоторые многочлены связаны с аналитическим решением специальных типов дифференциальных уравнений.

Мы рассмотрим вычисление многочленов из класса Аппеля: многочлены Бернулли, Лагерра, Эрмита, Чебышева и Лежандра. Многочлены Чебышева первого и второго рода и многочлены Лежандра являются частными случаями многочленов Якоби, являющихся, в свою очередь, специальным случаем гипергеометрической функции, представленной в виде ряда Гаусса:

$$F(\alpha,\beta,\gamma,z)=1+\sum_{n=1}^{\infty}\frac{\alpha(\alpha+1)...(\alpha+n-1)\beta(\beta+1)...(\beta+n-1)}{n!\gamma(\gamma+1)...(\gamma+n-1)}z^n,$$

где α,β,γ — параметры.

Связь многочленов Якоби и функции B-распределения (см. *приложение 2*) обсуждается в литературе.

Многие из рассмотренных многочленов обладают свойством ортогональности, которое заключается в том, что

$$\sum_{s=1}^{n}P_i\ (x_s)P_j\ (x_s)=0, i\neq j.$$

Многочлены Бернулли

Многочлены Бернулли определяются формулой

$$B_n(x)=\sum_{s=0}^{n}C_n^s B_s x^{n-s}, n=0,1,2,...,$$

где B_s — числа Бернулли.

Так как все нечетные числа Бернулли, кроме числа с номером 1, равны нулю, для обозначения чисел и многочленов Бернулли применяют также обозначение $B_{2m}(x), m=0,1,\ldots$. Рекуррентные соотношения для определения чисел Бернулли имеют вид:

$$B_0(x)=1,$$

$$B_n(x)=x^n-\frac{1}{n+1}\sum_{s=0}^{n-1}C_{n+1}^s B_s(x), n=1,2,...,$$

где C_{n+1}^s — число сочетаний из $(n+1)$ по s.

Можно было бы продолжить и составить рекурсию для числа сочетаний, чего мы делать не будем и оставим в качестве упражнения любознательному читателю.

Листинг 10.1

```
double __declspec(dllexport) Bernoulli (long int n,double x)
//
// Функция вычисления значения многочлена Бернулли.
// Обозначения:
//  n - степень многочлена,
//  x - значение переменной.
// Возвращаемое значение:
//  значение многочлена Бернулли.
//
{
 register long int s; // Счетчик

 double sum = 0; // Рабочая переменная

 if (!n)
   return 1;
 else
   {
    for (s = 0; s < n; s++)
      sum += Combination (n + 1,s) * Bernoulli (s,x);

    return pow (x,n) - sum / (n + 1);
   }
}

double Combination (long int m,long int n)
//
// Функция возвращает число сочетаний из m по n
//
{
 return exp (LogFactorial (m) - LogFactorial (n) - LogFactorial (m - n));
}

double LogFactorial (long int n)
//
```

```
// Функция возвращает значение логарифма факториала n.
// Примечание:
//  функция не отслеживает ошибку n <= 0.
//
{
 register int i;          // Счетчик

 double Ret = log (1.0); // Возвращаемое значение

 for (i = 1; i <= n; i++)
   Ret += log ((double) i);

 return Ret;
}
```

Многочлены Лагерра

Обобщенные (присоединенные) многочлены Лагерра являются решениями дифференциального уравнения

$xy'' + (\lambda + 1 - x)y' + ny = 0$

и определяются формулой

$$L_n^{(\lambda)}(x) = (-1)^n x^{-\lambda} e^x \frac{d^n}{dx^n}(x^{\lambda+n} e^{-x}), n = 0,1,2,...,$$

где $\lambda > -1$,

$0 \le x < \infty$.

Рекуррентные соотношения для вычисления обобщенных многочленов Лагерра выглядят как

$$L_0^{(\lambda)}(x) = 1,$$

$$L_1^{(\lambda)}(x) = x - \lambda - 1,$$

$$L_n^{(\lambda)}(x) = (x - \lambda - 2n + 1)L_{n-1}^{(\lambda)}(x) - (n-1)(\lambda + n - 1)L_{n-2}^{(\lambda)}(x),$$

$n = 2,3,\ldots.$

Связь многочленов Лагерра с интегралом вероятностей χ^2 и с Г-распределением обсуждается в литературе.

Листинг 10.2

```
double __declspec(dllexport) Laguerre (long int a,long int n,double x)
//
// Функция вычисления значения обобщенного многочлена Лагерра.
// Обозначения:
//  a - параметр,
//  n - степень многочлена,
//  x - значение переменной.
// Возвращаемое значение:
//  значение полинома Лагерра.
//
{
 if (!n)
   return 1;
 else if (n == 1)
   {
    return x - (a + 1) / pow (x,a);
   }
 else
   return (x - a - 2 * (n - 1) - 1) * Laguerre (a,n - 1,x) -
          (n - 1) * (a + n - 1) * Laguerre (a,n - 2,x);
}
```

Многочлены Эрмита

Многочлены Эрмита являются решениями дифференциального уравнения

$$y'' - 2xy' + 2ny = 0,$$

где $n = 0,1,2,...,$

и определяются формулой

$$H_n(x) = (-1)^n e^{x^2} \frac{d^n e^{-x^2}}{dx^n}.$$

Рекуррентные соотношения для вычисления многочленов Эрмита выглядят как

$$H_0(x) = 1,$$

$$H_1(x) = 2x,$$

$$H_n(x) = 2xH_{n-1}(x) - 2(n-1)H_{n-2}(x), n = 2,3,\ldots.$$

Листинг 10.3

```
double __declspec(dllexport) Hermite (long int n,double x)
//
// Функция вычисления значения многочлена Эрмита.
// Обозначения:
//  n - степень многочлена,
//  x - значение переменной.
// Возвращаемое значение:
//  значение полинома Эрмита.
//
{
 if (!n)
   return 1;
 else if (n == 1)
   return 2 * x;
 else
   return 2 * x * Hermite (n - 1,x) - 2 * (n - 1) * Hermite (n - 2,x);
}
```

Рекуррентные соотношения для вычисления многочленов Эрмита можно записать по-другому. Тогда их называют многочленами Чебышева—Эрмита и вычисляют как

$$H_0(x) = 1,$$

$$H_1(x) = -x,$$

$$H_n(x) = -xH_{n-1}(x) - (n-1)H_{n-2}(x), n = 2,3,\ldots.$$

В такой форме многочлены применяются для вычисления производных плотности нормального распределения (см. *приложение 2*) по формуле

$$\varphi^{(n)}(x) = \frac{d^n\varphi(x)}{dx^n} = H_n(x)\varphi_n(x).$$

Листинг 10.4

```
double __declspec(dllexport) ChebyshevHermite (long int n,double x)
//
// Функция вычисления значения многочлена Чебышева-Эрмита.
// Обозначения:
```

```
//  n - степень многочлена,
//  x - значение переменной.
// Возвращаемое значение:
//  значение многочлена Чебышева-Эрмита.
//
{
 if (!n)
   return 1;
 else if (n == 1)
   return - x;
 else
   return - x * ChebyshevHermite (n - 1,x) - (n - 1) * ChebyshevHermite
(n - 2,x);
}
```

Многочлены Чебышева

Многочлены Чебышева первого рода являются решениями дифференциального уравнения

$$(1-x^2)y''-xy'+n^2y=0$$

и определяются формулой

$$T_n(x)=\cos(n\arccos(x))=\frac{2^n n!}{(2n)!}\sqrt{1-x^2}\,\frac{d^n}{dx^n}\left[(1-x^2)^{n-1/2}\right], n=0,1,2,...,$$

где $-1\le x\le 1$.

Рекуррентные соотношения выглядят гораздо проще:

$$T_0(x)=1,$$

$$T_1(x)=x,$$

$$T_n(x)=2xT_{n-1}(x)-T_{n-2}(x), n=2,3,\ldots.$$

Многочлены Чебышева второго рода являются решениями дифференциального уравнения

$$(1-x^2)y''-3xy'+n(n+2)y=0$$

и определяются формулой

$$U_n(x)=\frac{\sin\left[(n+1)\arccos(x)\right]}{\sqrt{1-x^2}}=\frac{2^n(n+1)!}{(2n+1)!}\frac{1}{\sqrt{1-x^2}}\frac{d^n}{dx^n}\left[(1-x^2)^{n+1/2}\right],$$

$n = 0,1,2,...,$

где $-1 \le x \le 1$.

Рекуррентные соотношения выглядят как

$U_0(x) = 1,$

$U_1(x) = 2x,$

$U_n(x) = 2xU_{n-1}(x) - U_{n-2}(x), n = 2,3,\ldots.$

Многочлены Чебышева образуют систему, ортогональную на отрезке [-1,1]. В литературе дается формула связи многочленов Чебышева первого и второго рода. Функции, вычисляющие многочлены Чебышева первого и второго рода, могут быть легко построены в результате незначительной модификации функции, показанной в предыдущем примере, поэтому здесь не приводятся.

Многочлены Лежандра

Рассмотрим уравнение

$$(1-x^2)y'' - 2xy' + \left[n(n+1) - \frac{\mu^2}{1-x^2}\right]y = 0,$$

где n и μ — произвольные числа.

Если $n = 0,1,2,...,$ а $\mu = 0$, то ограниченные на [—1,1] решения уравнения называются многочленами Лежандра (сферическими многочленами)

$$P_n(x) = \frac{1}{n!2^n}\frac{d^n}{dx^n}(x^2-1)^n$$

и определяются по рекуррентным формулам

$P_0(x) = 1,$

$P_1(x) = x,$

$P_n(x) = \left[(1-2n)xP_{n-1}(x) + (n-1)P_{n-2}(x)\right]/n, n = 2,3,\ldots.$

Благодаря тому, что многочлены Лежандра ортогональны на отрезке [—1,1], они образуют полную систему функций и могут быть использованы для разложения в ряд произвольной функции, интегрируемой на отрезке [—1,1].

Листинг 10.5

```
double __declspec(dllexport) Legendre (long int n,double x)
//
```

```
// Функция вычисления значения полинома Лежандра.
// Обозначения:
//  n - степень полинома,
//  x - значение переменной.
// Возвращаемое значение:
//  значение полинома Лежандра.
//
{
 if (!n)
   return 1;
 else if (n == 1)
   return x;
 else
   return ((2 * n - 1) * x * Legendre (n - 1,x) -
          (n - 1) * Legendre (n - 2, x)) / n;
}
```

Руководство пользователя

Функции вычисления полиномов (многочленов), содержащиеся в модуле Polynom, доступны для вызова в категории **Многочлены**. Вызов этих функций производится выбором из меню Microsoft Excel пункта **Вставка ▸ Функция**. На экране появится окно как на рис. 10.1.

Рис. 10.1. Окно 1 из 2 Мастера функций при работе с пакетом Polynom в Microsoft Excel 2002

Следует выбрать категорию **Многочлены** и нужную функцию, после чего нажать кнопку **OK** для перехода к следующему шагу. На экране появится окно рис. 10.2.

Рис. 10.2. Окно 2 из 2 Мастера функций при работе с пакетом Polynom в Microsoft Excel 2002

Следует ввести или выбрать все необходимые параметры в соответствующих полях окна Мастера функций. После ввода или выбора нажать кнопку **OK**. Будет выполнен требуемый расчет, а результат помещен в ячейку, в которую была введена функция. Дополнительные сведения о порядке вызова функций Microsoft Excel можно получить из справочной системы Microsoft Excel.

Некоторым недостатком является то, что при экспорте рабочей книги (файла) Microsoft Excel, содержащей вызовы функций Polynom, адресат должен иметь установленный модуль Polynom. Впрочем, это проблема любого аналогичного программного обеспечения.

Рекомендуемая литература

1. Большев Л.Н., Смирнов Н.В. Таблицы математической статистики. — М.: Наука, 1983.
2. Вероятность и математическая статистика: Энциклопедия. — М.: Научное издательство «Большая Российская энциклопедия», 1999.
3. Корн Г., Корн Т. Справочник по математике для научных работников и инженеров. — М.: Наука, 1973.
4. Малоземов В.Н., Певный А.Б. Рекуррентные вычисления. — Л.: Издательство Ленинградского университета, 1976.

5. Математический энциклопедический словарь. — М.: Научное издательство «Большая Российская энциклопедия», 1995.
6. Орвис В.Дж. Excel для ученых, инженеров и студентов. — Киев: Юниор, 1999.
7. Страуструп Б. Язык программирования Си++. — М.: Радио и связь, 1991.
8. Хемминг Р.В. Численные методы для научных работников и инженеров. — М.: Наука, 1972.

ГЛАВА 11

Статистический контроль качества и графические средства VBA

«... великую силу и важность можно и скромным словам придать расстановкой и связью».
Гораций

Для работы с материалами главы понадобятся лицензионные средства разработки и тестирования программ:

- Microsoft Excel 97 или выше.

Потребуется знание VBA.

Решаемая проблема

Контроль качества интересен как инструмент эффективного решения задач на этапе внедрения в производство передовых методов управления. Контроль качества выпускаемой продукции или оказываемых услуг необходим для своевременного выявления нарушений в организации производства и в технологических процессах.

Алгоритмы статистического контроля качества очень просты по сравнению с другими, описанными в книге. Результаты их работы наиболее видны в графиках, что обусловило использование графических возможностей Microsoft Excel.

Научные алгоритмы

Статистический контроль качества, по определению У. Деминга, — это применение статистических принципов и приемов на всех стадиях производства, направленное на экономичное производство изделия, максимально полезного и имеющего сбыт. Статистическое управление качеством, по определению Дж. Мердока, — это совокупность методов обнаружения неслучайных факторов, позволяющих диагностировать состояние процесса, провести его корректировку и, в конечном счете, способствующих улучшению качества продукции.

Рассмотренные в данной главе методы статистического контроля качества: гистограмма, диаграмма Парето, контрольная карта — принадлежат к совокупности семи элементарных методов контроля качества, описанных в классических источниках. К другим, не рассматриваемым здесь статистическим методам контроля качества, относятся точечный график и диаграмма разброса, которые могут быть легко построены стандартными средствами Microsoft Excel. Последние два классических метода: диаграмма Исикавы (диаграмма «причины — результат») и таблица контроля — не относятся к статистическим методам. Данные методы просты, наглядны, удобны. Применение совокупности этих методов решает 95% всех производственных проблем. Не стоит, однако, понимать статистическое управление качеством в том смысле, что его применение даст немедленный практический эффект. Статистическое управление качеством не предназначено для решения в принципе не разрешимых на современном этапе социального и технического развития проблем.

Отметим, что в статистическом управлении качеством используются и другие методы статистического анализа данных: описательная статистика пакета анализа Microsoft Excel; методы предварительной обработки данных (см. *главу* 3) и корреляционный анализ (см. *главу* 8).

Функция, составленная целиком на языке VBA, показана на листинге 11.1. Она обеспечивает верификацию исходных данных и управление процессом выполнения статистического контроля качества тремя избранными методами.

Гистограмма качества

Гистограмма качества позволяет наглядно отобразить выявленный характер разброса значений контролируемого параметра, среднее значение, стандартное отклонение и заданные границы допуска. На гистограмме качества, в отличие от гистограммы нормальности распределения, не наносится нормальная кривая.

Исходными данными для построения служат количества вариант (частоты), относящиеся к каждому интервалу отклонения исследуемого параметра. На графике частоты откладываются по оси ординат. По оси абсцисс откладываются кодовые обозначения интервалов отклонения параметра (классов). Частоты могут быть легко получены из исходных данных с помощью пакета анализа Microsoft Excel. Для этого нужно выбрать раздел **Гистограмма** и следовать указаниям Мастера (см. рис. 11.1).

Рис. 11.1. Пример гистограммы качества

Умение читать и анализировать гистограммы окажет неоценимую услугу специалисту не только в статистическом управлении качеством, но и в анализе данных.

Встречаются следующие основные типы гистограмм:

1. обычный тип. Гистограмма имеет симметричную колоколообразную форму. Среднее значение приходится примерно на середину размаха данных. Этот тип свидетельствует об однородности исходных данных, а в статистическом контроле качества — о нормальном протекании технологического процесса;
2. положительно (отрицательно) скошенное распределение. Форма асимметрична. Среднее значение локализуется справа (слева) от середины размаха. Такая форма встречается, когда нижняя (верхняя) граница регулируется либо теоретически, либо по значению допуска,или когда левое (правое) значение недостижимо;
3. распределение с обрывом слева. Форма асимметрична. Среднее арифметическое локализуется слева (справа) от середины размаха. Эта форма встречается при просеивании изделий из-за плохой воспроизводимости процесса;

4. плато (равномерное и прямоугольное распределения). Такая форма встречается в смеси нескольких распределений, имеющих различные средние;
5. бимодальное (двухпиковое) распределение. Такая форма встречается, когда смешиваются два распределения с далеко отстоящими друг от друга средними значениями;
6. распределение с изолированным пиком. Такая форма проявляется при наличии малых включений из другого распределения (из другого процесса), появления ошибки измерения или в случае нарушения нормальности процесса.

Рассмотрим влияние формы гистограммы качества на действия специалиста по контролю качества. В случае 1 процесс считается протекающим нормально. В случаях 2 и 3 требуется вмешательство специалистов для проверки и, если потребуется, отладки технологического процесса. Случаи 4, 5 и 6 свидетельствуют о неоднородности данных. Неоднородность может быть вызвана ошибками при сборе данных или свидетельствовать о нестабильности технологического процесса.

Как уже было отмечено, в некоторых источниках на гистограмму накладываются границы допуска (промежуток оси абсцисс между границами допуска называется полем допуска). Можно рассуждать о том, допустимо ли на одной диаграмме объединять график порядковой переменной (гистограмма) и количественной (границы допуска), а такие заблуждения распространены повсеместно. Однако так делают некоторые авторы, поэтому рассмотрим, как можно с некоторой пользой применить данную информацию. Процесс считается протекающим нормально, если почти вся гистограмма находится в границах допуска. В этом случае требуется лишь поддержание существующего состояния технологического процесса. Если гистограмма не удовлетворяет допуску, необходимо добиться смещения среднего значения ближе к центру поля допуска.

Функция построения гистограммы качества показана на листинге 11.1. Ее особенностью является то, что здесь мы обратились к графическим средствам Microsoft Excel. Вывод графиков в Microsoft Excel не является трудной задачей и подробно описан в литературе.

Примечание

Для анализа технологии построения графиков средствами VBA можно с большой пользой воспользоваться встроенным рекордером макросов Microsoft Excel.

Наша функция отличается универсальностью и содержит в себе несколько полезных элементов, которые могут пригодиться и в других проектах. Основными мы считаем возможности, с помощью данной функции, помещать график в указанное пользователем место рабочего листа.

Листинг 11.1

```
Private Sub CreateHistogram(Data As Range, Position As Range)
'
' Функция построения гистограммы
'
' Обозначения:
'  Data - интервал исходных данных
'  Pos - позиция вывода решения
'
  Dim myChartObject As ChartObject
  Dim myChart As Chart
  Set myChartObject = ActiveSheet.ChartObjects.Add(Position.Left, _
      Position.Top, 300, 250)
  Set myChart = myChartObject.Chart
  With myChartObject.Chart
    .ChartType = xlColumnStacked
    .SetSourceData Source:=Data
    .PlotBy = xlColumns
    .DisplayBlanksAs = xlZero
    .HasLegend = False
    .HasTitle = True
    .ChartTitle.Text = "Гистограмма"
  End With
  With myChart.Axes(xlValue)
    .HasMajorGridlines = False
    .HasTitle = True
    .AxisTitle.Text = "Частоты"
  End With
  With myChart.Axes(xlCategory)
    .HasTitle = True
    .AxisTitle.Text = "Интервалы отклонения параметра"
  End With
```

Следующий фрагмент кода делает из столбиковой диаграммы гистограмму.

Листинг 11.1 (продолжение)

```
  With ActiveChart.ChartGroups(1)
    .Overlap = 0
    .GapWidth = 0
  End With
End Sub
```

Диаграмма Парето

Диаграмма Парето (диаграмма распределения Парето) служит для наглядного выявления наиболее значимого фактора, влияющего на снижение качества продукции.

В разных источниках диаграмма Парето может изображаться по-разному, однако общей является комбинация в данной диаграмме графика типа гистограммы, изображающего процент брака по вине того или иного фактора, с ломаной линией (полигоном, кривой эффективности), отражающей накопленные проценты. Под браком здесь подразумевается несоответствие достигнутых показателей качества с показателями качества, определяемыми техническими заданиями.

Исходными данными для построения диаграммы Парето служат количества вариант, относящиеся к каждому фактору, ответственному за снижение качества продукции, либо частоты. На графике по оси ординат откладываются частоты, выраженные в процентах. По оси абсцисс откладываются кодовые обозначения факторов. Частоты могут быть найдены так же, как указано в предыдущем разделе (см. рис.11.2).

Рис. 11.2. Пример диаграммы Парето

Функция построения диаграммы Парето представлена на листинге 11.2. В отличие от описанной выше функции построения диаграммы качества, данная функция выводит на одном поле два графика различного типа.

Листинг 11.2

```
Private Sub CreatePareto(Percent As Range, _
Cumulate As Range, Position As Range)
'
' Функция построения диаграммы Парето
'
' Обозначения:
'  Percent - интервал ячеек процентов
'  Cumulate - интервал ячеек накопленных процентов
'  Pos - позиция вывода решения
'
  Dim myChartObject As ChartObject
  Dim myChart As Chart
  Dim mySeriesCollection As SeriesCollection
  Dim mySeries As Series
  Dim myChartObjectGroups As ChartGroups
  Dim myChartObjectGroup As ChartGroup

  Set myChartObject = ActiveSheet.ChartObjects.Add(Position.Left + _
                      ActiveCell.Width * 2, Position.Top + _
                      ActiveCell.Height, 300, 250)
  Set myChart = myChartObject.Chart

  Set mySeriesCollection = myChart.SeriesCollection
  mySeriesCollection.Add Source:=Percent, RowCol:=xlColumns
  mySeriesCollection.Add Source:=Cumulate, RowCol:=xlColumns

  Set mySeries = myChart.SeriesCollection(1)
  mySeries.ChartType = xlColumnClustered
  Set mySeries = myChart.SeriesCollection(2)
  mySeries.ChartType = xlLineMarkers

  With myChartObject.Chart
    .DisplayBlanksAs = xlZero
    .HasTitle = True
```

```
    .ChartTitle.Text = "Диаграмма Парето"
    .HasLegend = False
  End With
  With myChart.Axes(xlValue)
    .MinimumScale = 0
    .MaximumScale = 100
    .HasTitle = True
    .AxisTitle.Text = "Проценты"
  End With
  With myChart.Axes(xlCategory)
    .HasTitle = True
    .AxisTitle.Text = "Факторы"
  End With
  With ActiveChart.ChartGroups(1)
    .Overlap = 0
    .GapWidth = 0
  End With
End Sub
```

При построении диаграммы Парето применяется функция бухгалтерского округления действительных чисел. Данная функция, к сожалению, отсутствует в стандартных библиотеках программ, но бывает очень часто необходима.

Листинг 11.3

```
Private Function SmartRound(X As Double)
'
' Функция округления числа
'
 Dim Y As Long ' Целая часть числа

 Y = X
 If X - Y < 0.5 Then
   SmartRound = Y
 Else
   SmartRound = Y + 1
 End If
End Function
```

Контрольная карта

Контрольная карта предназначена для обнаружения отклонения характеристики качества выпускаемой продукции от заданных технологических норм и допусков.

Контрольная карта представляет собой график изменения исследуемой характеристики во времени. На график дополнительно наносятся предупреждающая граница и граница регулирования. Различные типы контрольной карты (обычно их выделяют шесть) описаны в литературе. В терминологии источников рассмотренная нами контрольная карта относится к типу *pn*-карты, когда показатель качества представлен числом дефектных изделий в последовательности выборок фиксированного объема, или к типу *c*-карты, когда управление процессом ведется по числу дефектов в изделиях одинакового размера.

Если характеристика качества производственного процесса находится ниже предупреждающей границы, то процесс протекает нормально. Если характеристика процесса находится между предупреждающей границей и границей регулирования, то технологический процесс функционирует, но не в соответствии с нормой. Попадание характеристики в зону выше границы регулирования означает, что должна быть произведена коррекция технологического процесса.

За границу регулирования часто принимается величина, равная утроенному стандартному отклонению, что, как известно, означает попадание в данные границы 95% вариант, в том случае, если распределение нормальное (см. *главу* 3).

Исходными данными для построения контрольной карты служат количества вариант, относящиеся к каждому фактору, ответственному за снижение качества продукции. На графике по оси ординат откладываются частоты, выраженные в абсолютных величинах (в штуках). По оси абсцисс откладываются кодовые обозначения номеров выборок, отобранных в процессе производства с целью его контроля. Дополнительно в тех же единицах измерения, что и исходные данные, должны быть заданы предупреждающая граница и граница регулирования (см. рис. 11.3).

Функция построения контрольной карты представлена на листинге 11.4. При построении графика стояла задача нанесения двух горизонтальных линий, соответствующих предупреждающей границе и границе регулирования. Чтобы упростить задачу, было принято решение вывести два массива значений, скрыв их под полем графика. Данные массивы как раз и отображают границы.

Горизонтальные линии, означающие предупреждающую границу и границу регулирования, нанести на график было бы не сложно, если бы не одно обстоятельство. Все точки графика Microsoft Excel являются прямым отобра-

жением содержимого ячеек рабочего листа. Взаимное расположение этих точек сохраняется при любом изменении содержимого ячеек или изменения размеров области диаграммы. Линии, однако, привязаны к области диаграммы, а не к области построения графика, и при изменении, например, шкалы значений графика не последуют за графиком. Их размещение придется корректировать вручную.

Рис. 11.3. Пример контрольной карты

Листинг 11.4

```
Private Sub CreateControlCard(Data As Range, Position As Range, _
                              AlarmLine As Range, ControlLine As Range)
'
' Функция построения контрольной карты
'
' Обозначения:
'  Data - интервал исходных данных
'  Position - позиция вывода решения
'  AlarmLine - предупреждающая граница
'  ControlLine - граница регулирования
'
  Dim myChartObject As ChartObject
  Dim myChart As Chart
  Dim mySeriesCollection As SeriesCollection
  Dim mySeries As Series
```

```
  Dim myChartObjectGroups As ChartGroups
  Dim myChartObjectGroup As ChartGroup

  Set myChartObject = ActiveSheet.ChartObjects.Add(Position.Left, _
                      Position.Top, 350, 250)
  Set myChart = myChartObject.Chart

  Set mySeriesCollection = myChart.SeriesCollection
  mySeriesCollection.Add Source:=Data, RowCol:=xlColumns
  mySeriesCollection.Add Source:=AlarmLine, RowCol:=xlColumns
  mySeriesCollection.Add Source:=ControlLine, RowCol:=xlColumns

  Set mySeries = myChart.SeriesCollection(1)
  mySeries.ChartType = xlLineMarkers
  Set mySeries = myChart.SeriesCollection(2)
  mySeries.ChartType = xlLine
  Set mySeries = myChart.SeriesCollection(3)
  mySeries.ChartType = xlLine

  With myChart
    .DisplayBlanksAs = xlZero
    .HasTitle = True
    .ChartTitle.Text = "Контрольная карта"
    .HasLegend = True
  End With
  With myChart.Axes(xlValue)
    .HasMajorGridlines = False
    .HasTitle = True
    .AxisTitle.Text = "Число дефектов"
  End With
  With myChart.Axes(xlCategory)
    .HasTitle = True
    .AxisTitle.Text = "Номер выборки"
  End With
  With myChart.Legend
    .Position = xlLegendPositionTop
  End With
End Sub
```

Руководство пользователя

Для работы с модулем статистического контроля качества (SQC) выберите из меню Microsoft Excel добавленный программным обеспечением AtteStat новый пункт **AtteStat ▸ Модуль SQC ▸ Статистический контроль качества**. На экране появится диалоговое окно, изображенное на рисунке 11.4.

Рис. 11.4. Окно модуля статистического контроля качества

1. Выберите или введите интервал переменной.
2. Выберите или введите выходной интервал. Начиная с первой ячейки выходного интервала, будут выведены результаты вычислений.
3. Выберите или оставьте по умолчанию тип диаграммы.
4. Введите границы для контрольной карты.
5. Нажмите кнопку **Выполнить расчет**.

Результаты расчета будут выведены, начиная с первой ячейки выходного интервала.

Модуль статистического контроля качества берет на себя верификацию исходных данных, выдавая подробную диагностику. При неверных действиях пользователя выдаются сообщения об ошибках.

Сообщения об ошибках

При ошибках ввода данных могут выдаваться диагностические сообщения следующих типов.

Рис. 11.5. Окно сообщения об ошибке

На рис. 11.5 не выбран или неверно введен интервал ячеек. Лучшим способом является не ввод, а выделение интервала ячеек стандартным образом.

Рис. 11.6. Окно сообщения об ошибке

На рис. 11.6 не выбран или неверно введен интервал ячеек вывода решения. Лучшим способом является не ввод, а выделение интервала ячеек стандартным образом.

Рис. 11.7. Окно сообщения об ошибке

Для построения контрольной карты на рис. 11.7 требуется задать предупреждающую границу.

Рис. 11.8. Окно сообщения об ошибке

Для построения контрольной карты на рис.11.8 требуется задать границу регулирования.

Рис. 11.9. Окно сообщения об ошибке

Сумма вариант выборки на рис. 11.9 равна нулю, поэтому расчет показателей качества для такой конфигурации исходных данных провести нельзя.

Ресурсы Интернет

Модули контроля качества в настоящее время входят в программы прикладного статистического анализа данных, упомянутые во введении. Доступны и удачные авторские разработки, такие, как программа «Контрольные карты Шухарта» (Г.Н. Лебский, **lebskiy.narod.ru**).

Рекомендуемая литература

1. Биллиг В.А. Мир объектов Excel 2000. – М.: Издательско-торговый дом «Русская редакция», 2001.
2. Боровиков В.П. STATISTICA: искусство анализа данных на компьютере. Для профессионалов. – СПб.: Питер, 2001.
3. Дюк В. Обработка данных на ПК в примерах. – СПб: Питер, 1997.
4. Контроль качества с помощью персональных компьютеров / Т. Макино, М. Охаси, Х. Докэ, К. Макино. – М.: Машиностроение, 1991.
5. Кулаичев А.П. Полное собрание сочинений в трех томах. Том. 1. Методы и средства анализа данных в среде Windows. STADIA. – М.: Информатика и компьютеры, 1999.
6. Никитин В.А. Управление качеством на базе стандартов ИСО 9000:2000. – СПб.: Питер, 2002.
7. Спицнадель В.Н. Системы качества (в соответствии с международными стандартами ISO семейства 9000). – СПб.: Издательский дом «Бизнес–пресса», 2000.
8. Статистические методы повышения качества / Пол. ред. Х. Кумэ. – М.: Финансы и статистика, 1990.
9. Уокенбах Дж. Подробное руководство по созданию формул в Excel 2002. – М.: Издательский дом «Вильямс», 2002.
10. Чекотовский Э.В. Графический анализ статистических данных в Microsoft Excel 2000. – М.: Издательский дом «Вильямс», 2002.

ГЛАВА 12

Разработка справочной системы и генерация дистрибутива

«Если вы создаете документ, а потом не используете, то может быть, это намек на то, что он вам все-таки был совсем не нужен».
Р. Джеффриз

Для работы с материалами главы понадобятся лицензионные средства разработки и тестирования программ:

- Microsoft Word версии 97 или выше;
- Microsoft Help Workshop версии 4.03 или выше;
- Crimson Help версии 2.3 или выше;
- Microsoft HTML Help Workshop версии 4.74 или выше;
- Microsoft HTML Help 1.4 SDK;
- Inno Setup Compiler версии 4.0.9 или выше;
- Microsoft Visual C++ версии 5.0 или выше.

Потребуется знание Microsoft Word, HTML, C++ и понимание общих принципов языков сценариев.

Решаемая проблема

В заключительной главе книги нам хотелось бы остановиться на более удобной организации работы с разработанными приложениями. Речь,

прежде всего, идет о:

- полноценной справочной системе, предлагающей существенную помощь по применяемым аналитическим алгоритмам;
- интеллектуальной программе установки приложения на компьютер пользователя, предлагающей верные решения независимо от программной конфигурации компьютерной системы и избавляющей пользователя от рутинной работы и поисков недостающих компонентов.

Отметим, что в данной главе не уделяется внимание дополнительным сервисным возможностям: генерации защищенного от копирования дистрибутива, привязке программы к конкретному компьютеру, манипуляциям с серийными номерами и активации продуктов, созданию демонстрационных версий и прочим вещам.

Разработка справочной системы

В представленные в книге учебные проекты файлы справочной системы не входят, кроме проекта справочной системы модуля SQC (см. *главу* 11) программы AtteStat. Файлы справочной системы входят в коммерческие версии представленных в книге модулей программного обеспечения AtteStat и ME.com. Поэтому предположим, что мы разрабатываем справочную систему для модуля SQC.

Существует два основных подхода к разработке справочных систем и, соответственно, к выбору современных инструментальных средств под управлением Microsoft Windows. Справочная система может быть создана в форматах HLP или HTML. Ниже рассказано подробно о каждом из подходов.

Для разработки возможно применение дорогостоящих инструментальных средств типа eHelp RoboHelp Office (eHelp Corp., **www.ehelp.com**), демонстрационная версия которого занимает более 90 Мбайт. В рекламных материалах по RoboHelp сказано, что он автоматизирует наиболее трудоемкие процедуры: построение оглавления и предметного указателя, добавляя функции полнотекстового поиска. Это полезные свойства программы. Но у нас несколько иная точка зрения на проблему построения справочной системы: наиболее трудоемким мы считаем ее смысловое наполнение. По утверждению ведущего специалиста компании Microsoft по системам безопасности Д. Лебланка (David LeBlanc), «никто и никогда не обращается к встроенной системе помощи». Нам хотелось бы добавить: к таким системам помощи, которыми оснащены большинство имеющихся на рынке программ.

Справочная система в формате HLP

Справочная система в формате HLP полностью поддерживается всеми версиями Microsoft Windows. Разработка ведется с помощью специального компилятора Microsoft Help Workshop, входящего в состав средств разработки приложений от Microsoft, например, в комплект Microsoft Visual C++, либо на сайте производителя.

Рассмотрим процесс создания справочной системы с помощью компактной и мощной программы Crimson Help (A. Zenchenko, **user.rol.ru/~crimson**). Оно функционирует как дополнительный модуль на базе текстового процессора Microsoft Word и компилятора Microsoft Help Workshop.

Перед началом разработки полезно продумать структуру будущей справочной системы, подробно проанализировав структуры известных систем и руководствуясь следующими рекомендациями.

1. Создать полный текст проектируемой справочной системы в одном файле. Желательно получить текстовый файл Windows без разметки и иллюстраций, с помощью, например, Microsoft WordPad или Microsoft Word.

2. Применить для формул удобный редактор Microsoft Equation, работа с которым подробно описана в справочной системе Microsoft Office, с той особенностью, что редактируемый файл следует сохранять после ввода каждой формулы.

3. Подготовить все иллюстрации в стандартном формате BMP, иллюстрации вставлять в документ не следует. Если справочная система будет содержать графические объекты, в том числе подготовленные в Microsoft Word, рекомендуется преобразовать их также в формат BMP. Отметим, что графики и различные составные иллюстрации удобно готовить в программе Microsoft PowerPoint (экспортируя в графические форматы). Если иллюстрации в ином стандартном графическом формате, то быстро конвертировать их можно с помощью любой графической программы, например, IrfanView (Irfan Skiljan, **www.irfanview.com**).

4. Разделы будущей справочной системы должны иметь заголовки, маркированные стандартным стилем **Заголовок 1**. При этом любые элементы данного стиля допустимо изменять, по усмотрению разработчика.

5. Избегать форматирования и оформления текста в Microsoft Word, таких как: маркированные списки, нумерованные списки, вставка символов. Данные возможности не всегда корректно обрабатываются ком-

пилятором справочной системы, и разработчик может получить в итоге работы не совсем то, что хотел. Можно использовать стандартное и пользовательское стилевое оформление документа. В нашем примере мы иногда используем стандартный стиль **Заголовок 2**.

Теперь, когда сделаны все необходимые приготовления и высказаны все предостережения, можно из меню кнопки **Пуск ▸ Программы ▸ Crimson Help** запустить программу Crimson Help. После установки на компьютер программного обеспечения Crimson Help файлы в формате RTF будут ассоциированы с данной программой. Этой возможностью можно будет пользоваться в дальнейшем, при отладке. А именно, при исполнении файла RTF будет загружаться Microsoft Word с надстройкой Crimson Help.

Для создания нового проекта следует выбрать из появившегося меню позицию **Создать новый проект**, как показано на рис. 12.1. Проект, в минимальном случае, включает в себя три или четыре файла: проекта с расширением HPJ, текста RTF, файл справочной системы HLP, при необходимости, файл содержания CNT. Кроме того, в проект могут быть включены иллюстрации в виде отдельных файлов в формате BMP. Данные файлы рекомендуется располагать в каталоге разрабатываемого проекта справочной системы, а файлы иллюстраций – в отдельном подкаталоге.

Рис. 12.1. Создание проекта справочной системы

После нажатия кнопки **OK** надстройка Crimson Help (рис. 12.2) поможет обеспечить корректную установку всей разметки файла RTF, необходимой для генерации справочной системы в формате HLP.

Рис. 12.2. Процесс создания справочной системы. Вставка страницы справки

Далее, необходимо скопировать в буфер текст справочной системы и вставить в открытое окно документа Microsoft Word.

Примечание

При работе с Microsoft Word 2002 под управлением Microsoft Windows XP возможно искажение текста, переданного через буфер. Для восстановления текста воспользуйтесь возможностью, доступной из меню **Сервис ▸ Исправить поврежденный текст...**.

Затем необходимо добавить или вставить страницы справки. Для этого в меню Crimson Help доступны следующие возможности:

- **Crimson Help ▸ Страница справки ▸ Добавить страницу справки**. При выборе данного пункта в конец документа будет предложено вставить новую страницу справки. Мы не будем пользоваться данной возможностью, т. к. уже имеем заранее подготовленный, для ускорения работы и настоящего изложения, полный текст справочной системы.

- **Crimson Help ▸ Страница справки ▸ Вставить страницу справки**. При выборе пункта курсор должен находиться в начале заголовка данной страницы справочной системы, как показано на рис. 12.2. Заголовок данной страницы имеет стиль **Заголовок 1**.

После выбора пункта вставки в окне свойств страницы справки на рис. 12.3 будет предложено определить свойства данной страницы.

Рис. 12.3. Процесс создания справочной системы. Определение свойств страницы справки

Можно оставить предложенные параметры по умолчанию и нажать кнопку **OK**. Любые дополнительные параметры, например, ключевые слова, можно ввести позднее, выбрав из меню **Crimson Help ▸ Страница справки ▸ Изменить страницу справки**. Аналогично следует поступить со всеми страницами проектируемой справочной системы.

Как мы отметили ранее, иллюстрации в справочную систему вставляются в формате BMP, однако помещать их непосредственно в документ RTF не следует. В отличие от формул, они не будут корректно обработаны компилятором. Иллюстрации, точнее, их теги, добавляются путем выбора из меню пункта **Crimson Help ▸ Картинка ▸ Добавить картинку**, как показано на рис. 12.4. При этом можно пользоваться такими стандартными возможностями форматирования, как размещение иллюстрации по левому краю, по центру или по правому краю. Обычно иллюстрации в размещаются по центру. Удаление иллюстрации, при необходимости, производится с помощью меню **Crimson Help ▸ Картинка ▸ Удалить картинку**. Все манипуляции по вставке и, особенно, удалению тегов и разметки следует производить только с помощью Crimson Help, во избежание ненужных проблем.

Следующей важной возможностью, которую предоставляет Crimson Help, является добавление гипертекстовых свойств: перекрестных ссылок и связей, как показано на рис. 12.5.

Рис. 12.4. Процесс создания справочной системы. Добавление иллюстрации

Рис. 12.5. Процесс создания справочной системы. Добавление связи

Связи могут быть следующих типов:

- Обычная связь. При ее выборе происходит переход к определенной странице справки. Для определения обычной связи следует выбрать из меню **Crimson Help ▸ Связь ▸ Обычная связь**. При этом должен быть помечен фрагмент текста, которому присваивается ссылка, как показано на рис. 12.5. После выбора данного пункта появится диалоговое окно, изображенное на рис. 12.6, в котором будет предложено выбрать страницу справки, определенной вводимой связью.
- Всплывающая связь. При ее выборе текст появляется в специальном небольшом окне. Связи данного типа применяются с целью вывода каких-либо коротких пояснений для пользователя. Особенностью данного типа связи является то, что странице справки, появляющейся в поясняющем окне, не назначается заголовок, а страница справки вставляется непосредственно перед текстом страницы.

Рис. 12.6. Процесс создания справочной системы. Определение обычной связи

- Команда. Данные связи применяются, например, для ввода электронного адреса или адреса Интернета. При их выборе данной ссылки происходит загрузка соответствующей программы, например, поч-

тового клиента или браузера. Для определения связи по ключевому слову, следует выбрать из меню **Crimson Help ▸ Связь ▸ Команда**. При этом должен быть помечен фрагмент текста, которому присваивается ссылка, аналогично тому, как это было сделано при добавлении обычной или всплывающей связи. После выбора данного пункта появится диалоговое окно, изображенное на рис.12.7, в котором будет предложено выбрать тип и назначить команду, определенную вводимой связью.

Рис. 12.7. Процесс создания справочной системы. Определение свойств команды

При разработке простых справочных систем может отсутствовать этап создания содержания. Содержание сохраняется в файле с расширением CNT. Для построения содержания следует выбрать из меню **Crimson Help ▸ Сервис ▸ Порядок страниц справки**, как показано на рис. 12.8.

При этом на экране появится окно для выбора порядка страниц справки, как показано на рис. 12.9. После появления окна список доступных страниц необходимо отобразить в левой половине окна нажатием кнопки с изображением вопросительного знака.

Рис. 12.8. Процесс создания справочной системы. Создание файла содержания

Затем, нажатием кнопки с изображением книги мы можем ввести необходимое количество новых разделов, а страницы справки, помещаемые в разделы, отмечаются в левой половине окна соответствующей установкой флажков нужных страниц справки.

На рис. 12.9 показаны основные разделы справочной системы. В нашем проекте мы остановились на трех, включающих в себя:

1. основную информацию о проекте, статус, описание возможностей и пояснения практической работы, подробное описание возможных ошибок;

2. теоретическое обоснование, как общее для рассматриваемого раздела науки, так и индивидуальное для каждого применяемого расчетного метода;

3. раздел «Прочее», включающий в себя полный текст лицензионного соглашения, информацию о поддержке и список рекомендуемых литературных источников для более глубокого изучения представленных расчетных методов.

Рис. 12.9. Процесс создания справочной системы. Этап определения порядка страниц в содержании справочной системы

После выполнения описанной выше процедуры создания содержания справочной системы можно приступить к выбору параметров создания и компиляции справочной системы.

Для задания параметров файла выбрать из меню **Crimson Help ▸ Создание справки ▸ Параметры создания справки**, после чего на экране возникнет окно, показанное на рис. 12.10. Основные параметры, которые следует выбрать или ввести, это имя файла, заголовок окна и начальная страница. Остальные параметры можно изменить или оставить по умолчанию, после чего нажать кнопку **OK**.

Компиляция выполняется путем выбора из меню **Crimson Help ▸ Создание справки ▸ Создать файл справки**, как показано на рис.12.11.

Рис. 12.10. Процесс создания справочной системы. Параметры создания файла справки

Рис. 12.11. Заключительный этап процесса создания справочной системы

Таким образом, мы завершили создание справочной системы. Файлами, которые должны будут предоставлены пользователю, являются файл содержания и собственно файл справочной системы. В нашем случае это файлы SQC.CNT и SQC.HLP. Запуск справочной системы производится файлом SQC.HLP, рис.12.12.

Рис. 12.12. Окно справочной системы, закладка Содержание

После стандартного выбора того или иного пункта меню справочной системы появляются окна с текстами статей справочной системы, рис. 12.3.

Рис. 12.13. Окно справочной системы

Поддерживаются стандартные возможности, предоставляемые справочной системой Windows: навигация, поиск по параметрам. Для получения подробных инструкций о работе со справочной системой Windows выберите **Пуск ▸ Справка**.

Разработка любого программного проекта, в том числе такого, как справочная система – процесс итерационный, поэтому после тестирования разработчик будет повторять описанные выше этапы, добиваясь заданного результата. Мы описали только основные возможности Crimson Help. Остальные возможности программы любознательному читателю предлагается исследовать самостоятельно.

Файлы справочной системы могут стандартно вызываться из модулей VBA. Для примера в *главе 11* вызов справки показан на листинге 12.1.

Листинг 12.1

```
Private Sub MakeHelp()
 Application.Help "Sqc.hlp"
End Sub
```

Для выполнения задачи применяется стандартный метод Help класса Application VBA Microsoft Excel, который одинаково хорошо обрабатывает и файлы справочной системы в формате HLP и в формате HTML.

Чтобы метод Help мог найти указанный файл справочной системы, можно:

- поместить данный файл (а также файл содержания CNT, при необходимости) в стандартный каталог, предназначенный для размещения файлов справочных систем,
- поместить указанные файлы в произвольный каталог, но указать при вызове метода Help полный путь файла справочной системы .

Как поместить файлы справочной системы при установке программы автоматически в нужные каталоги, рассказано в разделе «Генерация дистрибутива».

Справочная система в формате HTML

Новым стандартом для разработки справочных систем Windows стала система Microsoft HTML Help Workshop, последнюю версию которой можно установить с компакт-диска, приложенного к журналу PC Magazine/RE № 11 за 2003 г. (на этом же диске содержится справочная система Microsoft HTML Help 1.4 SDK) или загрузить с сайта **www.microsoft.com**. Порядок работы с данным инструментальным средством описан в материалах компакт-

диска упомянутого номера журнала, однако мы решили кратко рассмотреть данную тему, надеясь предоставить информацию читателю в компактном виде. Для работы с описываемым инструментарием знание языка HTML или хотя бы его основных тегов не повредит разработчику.

Основным достоинством справочных систем в формате HTML является мультиплатформенность. Файлы HTML, составляющие данную справочную систему, можно просматривать в любом браузере под управлением той операционной системы, для которой разрабатывалась данная справочная система. Для этого только следует использовать стандартные возможности HTML, избегая расширений, специфических для некоторых браузеров.

Перед началом разработки следует:

- создать полный текст проектируемой справочной системы в формате HTML, причем каждая страница справочной системы должна находиться в отдельном файле HTML. Рекомендуем применять для этого надежный текстовый редактор Microsoft Write. Можно использовать и любые специализированные HTML редакторы. Если необходимо ввести формулы в справочную систему, можно применить Microsoft Word или удобный редактор формул Microsoft Equation. Сохранять такой документ необходимо как файл HTML, а затем производить особую процедуру, удаляющую из документа HTML разметку. Это можно выполнить с помощью Microsoft Office HTML Filter (для Microsoft Word 2000), Absolute HTML Compressor (Alentum Software Inc., **www.alentum.com**) или StripFile (Nueway, **www.nuetools.co.uk**). HTML-фильтр Office 2000 – это инструмент для удаления специальных тегов разметки из документов Office 2000, сохраненных в формате HTML. Дело в том, что при создании HTML-документа в приложениях Office 2000 в текст документа встраиваются особые теги разметки. Эти теги обеспечивают «обратимость» документа, в целях его дальнейшего редактирования в Microsoft Word. В результате удаления этих тегов сокращается размер документа. Как следствие, документ занимает значительно меньше дискового пространства, а загрузка страницы происходит быстрее;

- подготовить все иллюстрации в стандартных форматах JPG или GIF. Может быть также использован новый формат PNG. Для максимального сжатия изображений можно воспользоваться программой типа IrfanView или StripFile. Если справочная система будет включать формулы, графики или иные графические объекты, их следует преобразовать также в указанные форматы. Для справочных систем многих научных программ формулы — обязательные элементы. Невозможность прямого ввода формул в HTML–файл отчасти компенсируется тем, что при сохранении документа, содержащего формулы, текстовый процессор Microsoft Word корректно переводит их в графические образы и аккуратно расставляет соответствующие теги. Все названные

иллюстрации вставляются в документ посредством специальных тегов HTML. Если иллюстрации в другом формате, то конвертировать их можно с помощью любой графической программы, например, IrfanView;

- для документов HTML использовать маркированные списки, нумерованные списки, таблицы, стилевое, цветовое и шрифтовое оформление, фрагменты на языке JavaScript и т.д. Необходимо, однако, чувство меры, а также знание и осознанное принятие основных рекомендаций разработчика Microsoft Excel. Кроме того, если мы разрабатываем справочную систему программного обеспечения научного или инженерного назначения, то следует придерживаться действующих стандартов оформления публикаций в печатных и электронных изданиях и традиций конкретной области знаний.

Разработка справочной системы начинается с выбора из меню **Пуск ▸ Программы ▸ HTML Help Workshop ▸ HTML Help Workshop**. Затем создается новый проект, как показано на рис.12.14.

Рис. 12.14. Окно HTML Help Workshop на этапе создания нового проекта

Необходимым на этапе создания нового проекта является работа мастера New project wizard, показанного на рис.12.15.

Рис. 12.15. Окно мастера New project wizard

Мастер может конвертировать из формата HLP в формат справочной системы HTML, как показано на следующем рис.12.16, однако мы данной возможностью пользоваться не будем.

Рис. 12.16. Окно мастера New project wizard

При нажатии кнопки **Далее** необходимо ввести имя нового проекта. В нашем случае это SQC.HHP (см. рис. 12.17).

Затем мастер предложит выбрать тип создаваемого файла проекта. Для нового проекта файлы содержания и файл индекса будут созданы в дальнейшем автоматически, а все файлы HTML проектируемой справочной системы в нашем распоряжении уже имеются, поэтому мы оставляем окно мастера без изменений и нажимаем кнопку **Далее** (см. рис. 12.18).

Рис. 12.17. Окно мастера New project wizard на этапе создания нового проекта

Рис. 12.18. Окно мастера New project wizard
на этапе создания нового проекта справочной системы

Проект содержит 3 секции (вкладки): Project (проект), Contents (содержание) и Index (индекс). Каждая секция в левой части содержит набор специфических инструментов, назначение которых определяется решаемыми задачами.

Секция Project позволяет вводить и редактировать различные разделы, основным из которых является Options (свойства). Секция Files (файлы) содержит список HTML файлов проекта. Секция Windows содержит необходимую служебную информацию. Пример секции показан на рис.12.19.

В секции Contents с помощью инструментов, находящихся в левой части окна, можно определять параметры содержания справочной системы, как показано на рис.12.20.

Рис. 12.19. Секция Project проекта HTML Help

Рис. 12.20. Секция Contents проекта HTML Help

Секция Index, показанная на рис.12.21, является основой поисковых возможностей проектируемой справочной системы. Хотя во многих программных продуктах данные возможности не используются, мы для примера ввели несколько ключевых слов, чтобы продемонстрировать работу поискового механизма.

Рис. 12.21. Секция Index проекта HTML Help

На рис.12.22 показано окно скомпилированной справочной системы. Разработанная справочная система поддерживает все стандартные возможности, включая навигацию и поиск по различным параметрам.

Рис. 12.22. Окно справочной системы

Вызов справочной системы из модуля VBA рассмотрен в предыдущем разделе.

Генерация дистрибутива

Дистрибутивом называется подборка программ, сформированная в соответствии с требованиями к комплекту готовых решений. Программы, входящие в дистрибутив, собраны в единой технологии и среды и оформлены в виде пакета, с учетом конфигурации компьютера. Современные дистрибутивы устанавливаются специальными программами, которые позволяют произвести установку, размещение и настройку системы. Дистрибутив является первым, с чем сталкивается пользователь программы, и от его качества может зависеть успех программы.

Часто дистрибутив оформляется в виде одного файла, включающего в себя и архив файлов устанавливаемой программы, и программу установки и настройки конфигурации.

Программы генерации дистрибутивов подразделяются на визуальные и скриптовые. Каждый из способов реализации имеет свои достоинства и недостатки. Главным доводом в защиту полезности скриптовой программы является тот факт, что скрипт (сценарий) пишется один раз и используется многократно. Перечислим основные программы создания дистрибутивов (в алфавитном порядке), ставшие промышленными стандартами для различных операционных систем:

- InstallAnywhere (Zero G Software Inc., **www.installanywhere.com**) для операционных систем Microsoft Windows, Mac OS X, Solaris на платформах SPARC Intel, Linux (включая генерацию пакетов RPM), HP–UX, AIX, FreeBSD, Compaq Tru64, NetWare, на других платформах с поддержкой Java.
- Installer VISE (MindVision Software, **www.mindvision.com**) для операционных систем Microsoft Windows, MacOS.
- InstallShield (InstallShield Software Corp., **www.installshield.com**) для операционных систем Microsoft Windows, MacOS, Linux, UNIX.
- SetupFactory (Indigo Rose Corp., **www.indigorose.com**) для операционных систем Microsoft Windows.
- Tarma Installer Pro (Tarma Software Research Pty Ltd., **www.tarma.com**) для операционных систем Microsoft Windows.
- Wise for Windows Installer (Wise Solutions Inc., **www.wisesolutions.com**) для операционных систем Microsoft Windows.

Для операционных систем Microsoft Windows растет популярность отечественной разработки CreateInstall (Gentee Inc., **www.gentee.com**, **www.gentee.ru**).

Достоинствами данных программ являются наличие локализации (в той или иной мере).

Привлекательными могут оказаться менее дорогостоящие пакеты авторитетных независимых разработчиков, как-то:

- AGInstaller (AGENTIX Software, **www.agentix.org**) для операционных систем Microsoft Windows.
- GP-Install (Quality Software Components Ltd., **www.qsc.co.uk**) для операционных систем Microsoft Windows.
- Inno Setup Compiler (Jordan Russell's Software, **www.jrsoftware.org**) для операционных систем Microsoft Windows.
- InstallAnywhere NOW! — бесплатная версия представленного выше профессионального программного обеспечения.
- Installer VISE — с выполнением особой процедуры регистрации.
- Installer2GO и Setup2GO (SDS Software, **www.dev4pc.com**), причем первый из пакетов является средством создания дистрибутивов типа Microsoft Windows Installer, т. к. непосредственное использование соответствующего SDK от компании Microsoft трудоемко.
- NSIS — Nullsoft Scriptable Install System (**nsis.sourceforge.ne**t) от всемирно известного производителя мультимедийных программ Nullsoft Inc. (**www.nullsoft.com**) для операционных систем Microsoft Windows.
- Tarma Installer — бесплатная версия представленного выше профессионального программного обеспечения.

Для Linux также существуют удобные программы генерации дистрибутивов, например, Linstall Wizard (New Republic Software, **www.newrepublicsoftware.com**).

Все дистрибутивы сгенерированы нами программой Inno Setup Compiler, заслужившей признание тысяч разработчиков. Начало 2004 г. принесло еще одно подтверждение правильности выбора. Специалисты из Jet Propulsion Laboratory (California Institute of Technology), разработавшие программное обеспечение Maestro для управления аппаратом исследовательской миссии Mars Exploration Rover Mission, успешно стартовавшей экспедиции на Марс, выбрали для публичной версии своего программного обеспечения генератор дистрибутивов Inno Setup Compiler. Отметим, что публичная версия Maestro, вместе с пакетами обновлений, содержащих последние результаты исследований, находится на **mars.telascience.org**.

Далее будет подробно описана работа с программой Inno Setup Compiler. Работа с другими программами строится аналогично.

Основные требования к программе генерации дистрибутива

Анализ программ генерации дистрибутива позволяет сформулировать основные требования пользователя к современной программе данного назначения:

- так называемый Point and Click (указать и щелкнуть)-пользовательский интерфейс, и, в то же время, наличие мощного языка сценариев, осуществляющего доступ к функциям программы;
- возможность для пользователя выбрать каталог назначения;
- сжатый архив для уменьшения объема при доставке дистрибутива пользователю, актуальный при распространении программ через Интернет;
- возможность показа пользователю файла типа README, содержащего важную актуальную информацию об устанавливаемом программном продукте;
- возможность потребовать от пользователя принятия условий лицензионного соглашения для продолжения установки;
- визуальное отображение процесса установки программного продукта;
- локализация на основные языки с возможностью добавления файлов пользовательской локализации;
- возможность добавления пользовательских модулей (plug-ins), позволяющих выполнять специальные функции, такие, как загрузка файлов через Интернет, создание спецэффектов, вызовы Win32 API и другие функции;
- отсутствие требований к наличию на пользовательских компьютерах специфических программных библиотек. Если такие библиотеки необходимы, они должны быть на законных основаниях включены в дистрибутив.

Программа установки должна также учитывать всевозможные конфигурации компьютеров пользователей и корректно осуществлять необходимую запись служебной информации, если это действительно необходимо.

Иногда программе установки поручают и более сложные функции, например, обновление дистрибутива через Интернет. Для некоторых операционных систем добавляется возможность автоматической компиляции программного продукта. Продукт, конечно, может поставляться в исходных кодах или в виде бинарных файлов, однако ничто не мешает программисту легко и быстро, как мы увидим далее, создать удобный дистрибутив. В том числе не только для пользователя, но и для себя самого.

Работа с программой Inno Setup Compiler

Далее, без потери общности, предполагается, что:

- операционная система Microsoft Windows установлена на диске C;
- каталог, в который установлена операционная система Microsoft Windows, имеет имя Windows или WINNT;
- каталог, содержащий программные файлы Microsoft Windows, называется Program Files.

Подобные значения параметров установки требуются многим современным дистрибутивам. Если пользователь изменил параметры установки Windows по умолчанию, мы могли бы испытать некоторые затруднения, если бы не предусмотрели этого заранее. Оказывается, что при использовании надежных программ генерации дистрибутива все параметры установки Windows, в том числе и измененные пользователем, можно легко учесть автоматически.

Программа Inno Setup Compiler имеет все необходимые средства для автоматической установки файлов программы именно в те каталоги, которые для них предназначены. Для этого программа предлагает основные глобальные константы:

- {app} — название каталога инсталлируемого приложения,
- {win} — название каталога Microsoft Windows, в зависимости от версии и пользовательских настроек — это могут быть, например, Windows или WinNT (здесь и далее — каталоги с полным путем, включая имя диска),
- {sys} — название системного каталога Microsoft Windows, в зависимости от версии, System или System32,
- {pf} — название каталога Program Files.

Файлы проектов Inno Setup Compiler с примерами использования констант находятся в соответствующих каталогах на прилагаемом компакт-диске.

Для генерации локализованного (русского) дистрибутива с сайта **www.jrsoftware.org** в каталог Inno Setup Compiler потребуется скопировать файл сообщений Russian.isl. Применение файла Russian.isl позволит избежать необходимости перевода сообщений. Файл является обычным текстовым файлом в кодировке Microsoft Windows, поэтому в него можно вносить изменения по усмотрению разработчика с помощью любого текстового редактора.

Примечание

О размещении файлов составленного программного обеспечения гласят также комментарии тестовой лаборатории «PC Magazine/Russian Edition» к программе MegaStat for Excel, содержащиеся в приложении на компакт-диске к №11 за 2002 г.

На листинге 12.2 представлен файл проекта дистрибутива для программного обеспечения, разработанного в *главе 11*.

Листинг 12.2

```
[Setup]
AppName=Statistical Quality Control for Excel
AppVerName=Statistical Quality Control for Excel, версия 1.0
AppCopyright=Copyright © Игорь Гайдышев, 2003
AppPublisher=Igor Gaidyshev
DefaultDirName={pf}\AtteStat\Sqc for Excel
DefaultGroupName=AtteStat\Sqc for Excel
InfoBeforeFile=Before.txt
LicenseFile=License.txt
OutputBaseFilename=Sqc
WizardStyle=modern
WizardImageFile=Image.bmp
WizardSmallImageFile=Logo.bmp

[Languages]
Name: "ru"; MessagesFile: "compiler:Russian.isl"

[Files]
; Копировать в стартовый каталог Microsoft Excel 2000
Source: "Sqc.xla"; DestDir: "{pf}\Microsoft Office\Office\XLStart";
Flags: ignoreversion

; Копировать в стартовый каталог Microsoft Excel 2002 (Office XP)
Source: "Sqc.xla"; DestDir: "{pf}\Microsoft Office\Office10\XLStart";
Flags: ignoreversion

; Копировать в стартовый каталог Microsoft Excel 2003
Source: "Sqc.xla"; DestDir: "{pf}\Microsoft Office\Office11\XLStart";
Flags: ignoreversion

; Копировать в системный каталог Help
Source: "Sqc.chm"; DestDir: "{win}\Help"; Flags: ignoreversion

[Icons]
Name: "{group}\Справочная система"; Filename: "{win}\Help\Sqc.chm"
Name: "{group}\Удаление программы"; Filename: "{uninstallexe}"
```

Необходимыми секциями проекта являются:

- секция [Setup], содержащая общую информацию об устанавливаемом программном обеспечении,
- секция [Languages], содержащая имена необходимых файлов локализации дистрибутива (не путать с локализацией самой устанавливаемой программы), содержащихся в каталоге Inno Setup Compiler,
- секция [Files], содержащая имена файлов Source, которые надлежит при установке скопировать в заданные каталоги DestDir, а также информацию Flags, сообщающую программе установки, как поступить, если в заданных каталогах будут обнаружены файлы с такими же именами,
- секция [Icons], содержащая информацию об именах файлов, которые будут помещены в стандартное меню кнопки **Пуск**.

Рассмотрим подробнее секцию [Setup]. Параметры означают:

- AppName определяет имя приложения.
- AppVerName — имя приложения, включая номер версии.
- AppCopyright несет информацию об авторских правах.
- AppPublisher определяет имя издателя программного обеспечения.
- DefaultDirName определяет наименование каталога приложения. При этом можно пользоваться перечисленными выше глобальными константами.
- DefaultGroupName специфицирует значение глобальной константы {group}, определяющей имя раздела в стандартном меню кнопки **Пуск**.
- InfoBeforeFile определяет имя файла, который будет показан пользователю перед началом установки.
- LicenseFile содержит имя файла с текстом лицензионного соглашения.
- OutputBaseFilename не является обязательным и содержит имя файла генерируемого дистрибутива. Введение данного параметра избавляет от необходимости переименовывать файл дистрибутива после каждой генерации.
- WizardStyle определяет внешний вид мастера установки и в современных программах обычно полагается равным modern. При этом мастер установки отображается в современном стиле Microsoft Windows 2000/XP.
- WizardImageFile определяет имя BMP-файла изображения (битовая карта с максимальным отображаемым размером 164 x 314 пикселей), пока-

зываемого при старте и финише программы установки. Данный файл содержит информацию по усмотрению разработчика. Обычно в него помещают привлекательное стилизованное изображение, имеющее отношение к задаче, выполняемой программой.

- WizardSmallImageFile определяет имя BMP-файла изображения (битовая карта с максимальным отображаемым размером 55 x 55 пикселей), в качестве которого обычно используют логотип разработчика либо значок проекта. Данный логотип будет появляться в правом верхнем углу каждого диалогового окна программы установки, кроме первого.

Назначение остальных параметров секций понятно из контекста. Многие из параметров введены опционно. Дистрибутив может генерироваться и без них. При отсутствии значений тех или иных параметров программа Inno Setup Compiler автоматически подставляет некоторые стандартные значения.

Рис. 12.23. Окно программы Inno Setup Compiler

Для генерации дистрибутива осталось выбрать из меню **File ▸ Compile** или нажать кнопку **Run**, как показано на рис. 12.23.

Программа Inno Setup Compiler предлагает много возможностей, из которых мы рассмотрели только наиболее необходимые.

Автозапуск

Автозапуск компакт-диска реализован специальной утилитой. В сети Интернет можно найти немало проектов утилит автозапуска компакт-дисков в Microsoft Windows. Например, в Центре компьютерных технологий Башкирского государственного университета (**www.magistr.magarif.ru**) разработано программное обеспечение DiskFile, позволяющее использовать специальный файл конфигурации, в котором перечисляются необходимые параметры автозапуска. Дополнительно можно применить различные визуальные эффекты. По простоте использования утилита похожа на описанный выше пакет Inno Setup Compiler.

Разработчик Microsoft Windows в своем API уже учел многие возможности, и в столь объемных проектах для решения сравнительно простой задачи нет необходимости. В минимальном варианте в корневой каталог компакт-диска помещается исполняемый файл дистрибутива, например, SQC.EXE, и файл AUTORUN.INF (все показанные здесь и далее имена файлов в системе Microsoft Windows набираются в любом регистре), содержимое которого представлено на листинге 12.3.

Листинг 12.3

```
[autorun]
open = sqc.exe
```

В файле AUTORUN.INF обязательной является секция [autorun], где параметру open присваивается значение, состоящее из имени файла, который надлежит исполнить.

Иногда требуется решить другую задачу — открыть в окне браузера HTML-файл, с помощью которого производится вывод на экран компьютера определенной информации и, возможно, загрузка того или иного приложения. На листинге показан возможный текст утилиты автозапуска. Компиляцию проекта утилиты автозапуска следует производить с использованием системы Microsoft Visual C++ версии 5.0 или выше. При этом следует выбрать для проекта тип «консольное приложение Win32».

Листинг 12.4

```
#include <windows.h>
#include <tchar.h>
#include <stdio.h>

int WINAPI WinMain(HINSTANCE, HINSTANCE, LPSTR lpCmdLine, int nCmdShow)
{
```

```
  ShellExecute(NULL, _T("open"), lpCmdLine, NULL, NULL, nCmdShow);
  return 0;
}
```

Построенная утилита, назовем ее OPEN.EXE, размещается в корневом каталоге компакт-диска. В корневом каталоге помещается также файл HTML, например, INDEX.HTM, который необходимо запустить, когда пользователь вставит компакт-диск. Третий необходимый файл, помещаемый в корневой каталог компакт-диска, как и в первом случае, носит имя AUTORUN.INF. Листинг данного файла для нашего проекта показан далее.

Листинг 12.5

```
[autorun]
open = open index.htm
```

В файле AUTORUN.INF обязательной является секция [autorun], в которой параметру open присваивается значение, состоящее из имени файла утилиты автозапуска (можно без расширения) и имени файла (с расширением), который надлежит исполнить.

Рекомендуемая литература

1. Lammers D. Connecting HTML Help to C++/MFC Programs // EC Software, 2001. Available from World Wide Web: < **http://www.helpandmanual.com** >.
2. Lammers D. Connecting HTML Help to Visual Basic Programs // EC Software, 2000. Available from World Wide Web: < **http://www.helpandmanual.com** >.
3. Lammers D. Connecting WinHelp to C++/MFC Programs // EC Software, 2001. Available from World Wide Web: < **http://www.helpandmanual.com** >.
4. Lammers D. Connecting WinHelp to Visual Basic Programs // EC Software, 2000. Available from World Wide Web: < **http://www.helpandmanual.com** >.
5. Биллиг В.А. Средства разработки VBA-программиста. Офисное программирование. Том 1. — М.: Издательско-торговый дом «Русская редакция», 2001.
6. Матросов А.В., Сергеев А.О., Чаунин М.П. HTML 4.0 — СПб.: БХВ–Петербург, 2001.

ПРИЛОЖЕНИЕ 1

Разработка дополнительных модулей для StarOffice и OpenOffice.org

«Декларируя переход к открытой системе, вы предлагаете тем самым принять участие в проекте и всем остальным».
Д. Брэдли

Для работы с материалами приложения понадобятся лицензионные средства разработки и тестирования программ:

- Sun StarOffice версии 5.2 или выше;
- OpenOffice.org версии 1.0 или выше;
- Bloodshed Dev-C++ 4.01.

Потребуется знание C.

Решаемая проблема

Применение стандартных электронных таблиц в качестве основы для быстрой разработки приложений, выполняющих различные прикладные задачи, помогает значительно сократить время разработки. Благодаря документированному прикладному интерфейсу программирования (API), наличию встроенного языка Visual Basic for Application (VBA), для выполнения задачи обычно используются популярные электронные таблицы Microsoft Excel.

В настоящее время на рынке программного обеспечения только для Microsoft Windows предлагается множество офисных пакетов программ, которые

теоретически могут явиться основой для пользовательских приложений. Все входящие в данные пакеты электронные таблицы объявляются совместимыми с форматом Microsoft Excel. Под этим понимается возможность работы с файлами, но отнюдь не возможность исполнения приложений для Microsoft Excel, составленных на языках программирования, поддерживаемых Microsoft Excel.

Очень немногие электронные таблицы действительно могут служить для разработки и исполнения пользовательских приложений. По всей видимости, здесь можно отметить лишь электронные таблицы Quattro Pro, входящие в офисный пакет WordPerfect Office, производимый фирмой Corel Corporation (**www.corel.com**), и электронные таблицы StarCalc, входящие в офисный пакет StarOffice, производимый компанией Sun Microsystems (**www.sun.com**). Бесплатный аналог StarOffice – офисный пакет с открытыми исходными кодами OpenOffice.org из одноименного проекта на **www.openoffice.org**.

Итак, в качестве менее дорогостоящей, но и менее удобной альтернативы Microsoft Excel могут применяться электронные таблицы StarCalc, входящие в комплект офисного пакета Sun StarOffice или электронные таблицы Calc из офисного пакета OpenOffice.org.

История StarOffice

Коммерческий пакет офисных приложений StarOffice впервые был представлен общественности германской компанией StarDivision Gmbh в 1995 г. и включал в числе других компонентов электронные таблицы StarCalc. Начиная с версии 5.2, StarOffice поставляется с документированным API, что важно для разработки пользовательских компонентов. С 1999 г. StarOffice принадлежит фирме Sun и до 2002 г. распространялся бесплатно. Современная коммерческая версия StarOffice7 работает под управлением ОС Linux, Solaris и Windows. В настоящее время из одного исходного кода, разрабатываемого сообществом OpenOffice.org, выпускаются два продукта: StarOffice, в который добавляются компоненты под проприетарной лицензией, и свободный OpenOffice.org. В отличие от StarOffice продукт, созданный в рамках финансируемого Sun проекта OpenOffice.org Foundation, является доступным как в бинарном виде, так и в исходных текстах. Он стандартно входит почти во все дистрибутивы Linux, а для Microsoft Windows доступен свободно[1].

---

[1] Англоязычная версия доступна в приложении на компакт-диске к «Компьютер-пресс», №9 за 2002 г., русская версия 1.0.1 доступна в приложении на компакт-диске к «PC Magazine / Russian Edition», №11 за 2002 г., русская версия 1.1 доступна для загрузки с сайта ALT Linux.

Мы рассмотрим программирование дополнительных модулей (подключаемых модулей, add-ins) для StarCalc/Calc на примере разработки пакета непараметрической статистики, который может найти применение при анализе данных в различных областях науки. Данная разработка является функциональным аналогом решения, представленного в *главе 5* настоящей книги для Microsoft Excel.

Важно, что программирование и вызов пользовательских функций из дополнительных модулей для StarCalc и для Calc совершенно одинаковы. В комплект поставки StarCalc входят содержательные и работающие, в отличие от некоторых стандартных коммерческих разработок, примеры, посвященные работе со скалярными и текстовыми данными, а также многоязыковой поддержке. В поставке Calc примеры, к сожалению, отсутствуют, о чем неплохо было бы сообщить пользователям в справочной системе, которая в данном вопросе слишком оптимистична.

Работа с массивами в примерах не рассмотрена совершенно, как не даны и рекомендации по выбору средств разработки. В справочной системе StarCalc/Calc имеется подробная информация, достаточная для начала самостоятельной разработки тривиальных проектов. Выбрать правильно средства разработки дополнительных модулей для StarCalc/Calc не слишком просто: стандартные средства разработки для Windows типа Microsoft Visual C++ применить не удалось. Это было к лучшему: применение Visual C++ в качестве инструментального средства разработки для свободного OpenOffice.org было бы отходом от стиля. Представленный пример будет полезен еще и потому, что в нем подробно разобрана работа с массивами, и в итоге получен готовый программный продукт.

Средства разработки для StarCalc/Calc

Разработка дополнительных модулей для StarCalc/Calc выполняется на языке программирования C. Разработка может успешно осуществляться с помощью системы Bloodshed Dev–C++ 4.01. Подробно создание проекта DLL в системе программирования Bloodshed Dev–C++ описано в *главе 2*, поэтому в следующем разделе мы остановимся только на отличиях подходов к разработке дополнительных модулей для Microsoft Excel и StarCalc/Calc.

За исключением исходных текстов разработки, которые для StarCalc/Calc могут быть выполнены только на языке C, порядок действий в основном тот же, что и при создании DLL для Microsoft Excel. На рис. П1.1 показано окно интегрированной системы Dev-C++ с только что успешно скомпилированным проектом DLL для StarCalc/Calc.

Рис. П1.1. Окно интегрированной системы Bloodshed Dev-C++ 4.01 с успешно скомпилированным проектом DLL для StarCalc/Calc

Примечание

На момент данной публикации была доступна только предварительная версия Bloodshed Dev-C++ 5.0. Все проблемы ее автором решены не были, поэтому мы рекомендуем, вплоть до выхода релиза, пользоваться версий 4.01.

Особенности разработки дополнительных модулей для StarCalc/Calc

Подходы, принятые в Microsoft Excel и StarCalc/Calc для программирования дополнительных модулей, различны. API как Microsoft Excel, так и StarCalc/Calc хорошо документированы, а их основные особенности сведены в таблицу П.1. Каждый подход имеет свои достоинства и недостатки.

***Таблица П.1**. Основные особенности API Excel и StarCalc/Calc*

	Microsoft Excel, надстройка в стиле Excel 4	**Microsoft Excel, надстройка**		**StarCalc/Calc**
Место регистрации функций DLL	Лист макросов	Модуль VBA	Stand-alone DLL (XLL)	DLL
Способ регистрации	Функция РЕГИСТРАТОР	Оператор Declare	Специальные функции на C	Специальные функции на C
Реализация пользовательского интерфейса	Функция РЕГИСТРАТОР или функции модуля VBA	Функции VBA	Специальные функции на C/C++	Специальные функции на C
Структура для обмена с функциями C	Структура, документированная API	Стандартные массивы или структуры	Структура, документированная API	Структура, документированная API
Использование расширений C++	Возможно	Возможно	Да	Нет*

* Речь только о программировании дополнительных модулей, а не о программировании компонентов, составляющих программные продукты StarOffice/OpenOffice.org.

Как видно из таблицы П.1, в StarCalc/Calc, как и в одной из реализаций для Microsoft Excel, все операции по реализации пользовательского интерфейса выполняются особыми сервисными функциями на стандартном языке C. Можно даже сказать, что в отличие от электронных таблиц Microsoft Excel, основным встроенным языком программирования которых является VBA (надстройки могут выполняться также на C/C++), в StarCalc/Calc встроенным языком программирования является C. Преимуществом использования C является более высокая скорость вычислений по сравнению с интерпретатором VBA. К числу недостатков StarCalc/Calc мы бы отнесли тот факт, что эти пакеты поставляются без средств разработки. Дополнительных проблем с включением какого-нибудь бесплатного компилятора в комплект поставки не возникло бы.

Сервисные функции StarCalc/Calc

Минимально необходимый набор сервисных функций StarCalc/Calc, которые программист должен составить или аккуратно заполнить по образцу, включает функции:

- генерации текстовых описаний функций и параметров `getText`;
- передачи в StarCalc/Calc числа импортируемых функций `GetFunctionCount`;
- передачи в StarCalc/Calc описаний функций `GetFunctionData`;
- передачи в StarCalc/Calc описаний параметров функций `GetParameterDescription`.

Перечисленные функции подробно описаны в примерах, входящих в комплект поставки StarCalc/Calc. Более простые и наглядные решения можно посмотреть в разработанном нами проекте, тем более что в примерах StarCalc/Calc (как, впрочем, и во всех источниках по программированию на VBA в среде Microsoft Excel) совершенно не рассматривается работа с массивами.

Системно-зависимые типы данных и макроопределения описаны в составленном разработчиками StarCalc/Calc файле addin.h. Его необходимо включить в разрабатываемый проект директивой `#include`. Основная особенность, которую следует учесть при программировании пользовательских функций: все экспортируемые из DLL в StarCalc/Calc пользовательские функции должны иметь тип void и системно-зависимый модификатор CALLTYPE, описанный в упомянутом файле addin.h. При этом возвращаемое функцией значение обязательно следует сделать первым формальным параметром. На функции C, используемые в проекте, ограничения не накладываются.

Структура данных StarCalc/Calc

Структура данных StarCalc/Calc довольно замысловата и начинается с общей информации об области данных электронной таблицы. Затем следуют элементы массива данных, причем каждый элемент массива данных представляет собой структуру, несущую полную информацию о местонахождении и содержимом передаваемой ячейки листа электронной таблицы. Структура одного такого элемента данных представлена на листинге П.1.

Листинг П.1

```
typedef struct
  {
   USHORT Col;    // Номер столбца
   USHORT Row;    // Номер строки
   USHORT Tab;
```

```
   USHORT Error;
   double Value; // Величина варианты
  } DATA;
```

Массив данных, переданный из электронной таблицы, в пользовательской функции нужно принять как массив системно-зависимого типа `USHORT` (см. файл addin.h), хотя на самом деле он представляет собой структуру, содержание которой подробно описано во встроенной документации пакета StarOffice/OpenOffice.org. Данная встроенная документация, несмотря на незначительный объем, заслуживает всяческих похвал за конкретность и разумную структурированность. Сейчас нас интересуют только некоторые элементы принятой структуры. Шестой элемент (при нумерации с нуля) – это число переданных ячеек электронной таблицы. Начиная с седьмого элемента, следуют представленные на листинге П.1 структуры типа DATA по одной на каждый переданный элемент данных. Чтобы извлечь данные из принятой структуры, требуется только вычислить начальный адрес первой из структур элементов данных, как показано в прокомментированном фрагменте программы, показанном на листинге П.2. Сделано это просто, но вычисления совсем не очевидны.

Листинг П.2

```
void CALLTYPE FisherIrwin (double* Ret,USHORT *ix,USHORT *iy)
//
// Функция вычисления критерия значимости различий
// точным методом Фишера-Ирвина.
// Обозначения:
//  *Ret - p-значение для односторонней гипотезы,
//    ix - информационный массив 1 выборки,
//    iy - информационный массив 2 выборки.
// Примечание:
//  это специальная версия функции для вызова из электронных
//  таблиц StarCalc.
// Возвращаемое значение:
//  нет.
//
{
 USHORT i,          // Счетчик
        nx = ix[6], // Численность 1 выборки
        ny = iy[6]; // Численность 2 выборки
```

```
int a = 0,          // Число наблюдений с эффектом А в первой выборке
    b = 0,          // Число наблюдений без эффекта А в первой выборке
    c = 0,          // Число наблюдений с эффектом А во второй выборке
    d = 0;          // Число наблюдений без эффекта А во второй выборке

double Buf,         // Вероятность получения текущей таблицы
       Const,       // Постоянная часть формулы
       Save;        // Вероятность получения исходной таблицы

DATA *x,*y;         // Массивы выборок

//////////////////////////////////////////////
// Вычисление начальных адресов вариант выборок
x = (DATA*)(&ix[7]);
y = (DATA*)(&iy[7]);

for (i = 0; i < nx; i++)
  if (x[i].Value == 1)
    a++;
  else
    b++;
for (i = 0; i < ny; i++)
  if (y[i].Value == 1)
    c++;
  else
    d++;

//////////////////////////
// Постоянная часть формулы
Const = LogFactorial (a + b) + LogFactorial (c + d) +
        LogFactorial (a + c) + LogFactorial (b + d) -
        LogFactorial (a + b + c + d);

////////////////////////////////////////
// Вероятность получения исходной таблицы
*Ret = exp (Const - LogFactorial (a) - LogFactorial (b) -
       LogFactorial (c) - LogFactorial (d));
```

```
 Save = *Ret;

 ////////////////////////////////////////////////////
 // Перебор всевозможных вариантов заполнения таблицы
 if (a != 0 && d != 0)
   {
    if (a <= d)
      while (a)
        {
         if ((Buf = exp (Const - LogFactorial (--a) - LogFactorial (++b)
-
              LogFactorial (++c) - LogFactorial (--d))) <= Save)
           *Ret += Buf;
        }
    else
      while (d)
        {
         if ((Buf = exp (Const - LogFactorial (--a) - LogFactorial (++b)
-
              LogFactorial (++c) - LogFactorial (--d))) <= Save)
           *Ret += Buf;
        }
   }
 else if (b != 0 && c != 0)
   {
    if (b <= c)
      while (b)
        {
         if ((Buf = exp (Const - LogFactorial (++a) - LogFactorial (--b)
-
              LogFactorial (--c) - LogFactorial (++d))) <= Save)
           *Ret += Buf;
        }
    else
      while (c)
        {
         if ((Buf = exp (Const - LogFactorial (++a) - LogFactorial (--b)
-
```

```
                LogFactorial (--c) - LogFactorial (++d))) <= Save)
            *Ret += Buf;
        }
    }
}
```

Использование add-in для StarCalc/Calc

Полученный файл DLL следует скопировать в подкаталог /program/addin того каталога, в который установлен пакет StarOffice/OpenOffice.org. Функции, экспортируемые DLL, без всяких дополнительных манипуляций будут доступны для употребления при следующем запуске StarCalc/Calc. Как и для стандартных функций, будут доступны два способа их вызова: при вводе в ячейки электронных таблиц или через меню **AutoPilot: Functions** (в русской версии **Мастер: Функции**) в категории **Add-in** (**Подключаемый модуль**), как показано на рис. П1.2.

Рис. П1.2. Окно AutoPilot: Functions при работе в программе StarCalc

Порядок использования функций через AutoPilot почти ничем не отличается от порядка вызова функций, принятого в других типах электронных таблиц. В частности, порядок вызова функций StarCalc/Calc с точностью до деталей похож на описанный в *главах 3* и *5* (сравните также рис. П1.2 с рис. 5.8) в применении к электронным таблицам Microsoft Excel.

Рекомендуемая литература

1. Рихтер Дж. Windows для профессионалов: создание эффективных Win32-приложений с учетом специфики 64-разрядной версии Windows. – СПб: Питер; М.: Издательско-торговый дом «Русская Редакция», 2001.

ПРИЛОЖЕНИЕ 2

Статистические распределения

Для работы с материалами приложения понадобятся лицензионные средства разработки и тестирования программ:

- Borland C++ 5.5 или выше,
- Bloodshed Dev–C++ 4.01 или выше,
- Microsoft Visual C++ 5.0 или выше.

Потребуется знание C.

Вычисление распределений

В данном приложении рассмотрены стандартные статистические распределения, применяемые в книге.

При вычислении статистических распределений могут применяться методы:

- точное вычисление критических значений;
- прямое интегрирование формул распределения с помощью методов численного интегрирования или методов численного решения обыкновенных дифференциальных уравнений (см. *главу 9*);
- разложение в бесконечный ряд или бесконечную (цепную) дробь и взятие конечного числа членов ряда.

Менее распространены другие методы:

- аппроксимация с помощью кривых из семейства распределений Пирсона;
- аппроксимация многочленами (см. *главу 10*);

- аппроксимация элементарными функциями и аппроксимация с помощью нейронной сети.

Точное вычисление обычно применяется только при вычислении дискретных распределений. При этом используется тот или иной рекуррентный алгоритм, что часто ведет к большому времени счета. Эта проблема затрудняет программную реализацию диалоговой статистической программной системы и часто без потери точности выводов решается с помощью аппроксимации статистик критериев другими стандартными распределениями.

Прямое интегрирование при вычислении непрерывных распределений производится по формуле трапеций либо с помощью формулы Симпсона, хотя может быть применен и любой другой метод, например метод Монте–Карло или метод Гаусса. Для вычисления распределений могут применяться методы решения обыкновенных дифференциальных уравнений. Метод дает компактный код, но требует большого времени вычисления, хотя затраты времени для некоторых типов распределений могут оказаться даже менее значительными, чем при использовании метода разложения в ряд. Прямым интегрированием можно добиться точности до 5–6 знака после десятичной запятой, но для некоторых «трудных» интервалов (для малых или, наоборот, больших значений параметров) статистических распределений приходится предпринимать дополнительные меры с целью получения достаточной точности решения.

Метод разложения в ряд или в бесконечную дробь является одним из самых эффективных. Данный метод обеспечивает контроль точности вычислений в ходе самого процесса вычислений. Он легко реализуется и обеспечивает заданную точность.

Многими авторами отмечалось, что связь различных типов распределений может быть использована для унификации их вычисления. Это позволяет иметь несколько хороших реализаций только нескольких базовых компонент и на их основе построить надежную и стройную систему вычислений. При реализации очень важно предпринять необходимые меры для получения заданной точности вычислений при любых допустимых значениях параметров.

В основе вычисления применяемых непрерывных распределений: нормального, хи-квадрат, t-распределения – лежат всего три компонента. Это:

- гамма-функция Эйлера,
- неполная гамма-функция,
- неполная бета-функция.

Набор базовых компонент позволяет всего в нескольких строчках кода реализовать любое распределение из упоминаемых в данной книге. Точно так же можно легко реализовать и другие распределения, применяемые в прикладном статистическом анализе.

Программные реализации функций распределения выполнены на C/C++. Это вызвано повышенными требованиями к скорости вычислений.

В проектах, приведенных в главах данной книги, для вычисления распределений использована комбинация методов прямого вычисления распределений с методами аппроксимации. Чтобы использовать методы разложения в ряд, достаточно заменить код ранее использованных функций на представленный ниже код, и перекомпилировать программное обеспечение.

Гамма-функция Эйлера

Гамма-функцией Эйлера называется функция вида

$$\Gamma(x) = \int_0^\infty t^{x-1} e^{-t}\, dt,$$

где x — положительное число.

Существует несколько методов вычисления гамма-функции. Прямое вычисление гамма-функции дает очень хорошие результаты для целых и полуцелых значений аргумента благодаря наличию удобных рекуррентных формул. Для других значений прямым интегрированием высокой скорости вычисления добиться не удается. Поэтому показанная ниже функция основана на аппроксимации Ланксоса. Приводятся также функция вычисления логарифма гамма-функции и функция вычисления степени числа, отсутствующая в стандартной библиотеке C.

Листинг П2.1

```
double Gamma (double x)
//
// Функция вычисления Г-функции действительного аргумента.
// Обозначения:
//  x - аргумент.
// Возвращаемое значение:
//  значение Г-функции.
//
{
short i;                                    // Счетчик

 double s = 1,                               // Сумма ряда
        c[] = {76.18009173,-86.50532033,     // Коэффициенты ряда
```

```
               24.01409822,-1.231739516,
               0.120858003E-2,-0.536382E-5};

 for (i = 0; i <= 5; i++)
   s += c[i] / (x + i);

 return 2.50662827463100050241 * s / exp (x + 4.5) *
        PowerOf(x + 4.5,x - 0.5);
}

double LogGamma (double x)
//
// Функция вычисления логарифма Г-функции действительного аргумента.
// Обозначения:
//  x - аргумент.
// Возвращаемое значение:
//  значение логарифма Г-функции.
//
{
 short i;                                    // Счетчик

 double s = 1,                               // Сумма ряда
        c[] = {76.18009173,-86.50532033,     // Коэффициенты ряда
               24.01409822,-1.231739516,
               0.120858003E-2,-0.536382E-5};

 for (i = 0; i <= 5; i++)
   s += c[i] / (x + i);

 return log (2.50662827463100050241 * s) - x - 4.5 + (x - 0.5) * log(x +
4.5);
}

double PowerOf (double x, double y)
//
// Функция вычисления степени числа.
// Обозначения:
//  x - число,
//  y - степень числа.
```

```
// Возвращаемое значение:
//  степень числа.
//
{
 if (y == 0)
   return 1;
 else if (x == 0)
   return 0;
 else
   return exp (y * log (x));
}
```

Находит применение также бета-функция

$$B(z,w)=\int_0^1 t^{z-1}(1-t)^{w-1}dt\ ,$$

которая связана с гамма-функцией как

$$B(z,w)=\frac{\Gamma(z)\Gamma(w)}{\Gamma(z+w)}.$$

При составлении показанной ниже функции вычисления значения бета-функции мы постарались избежать возможной вычислительной проблемы деления одного большого числа на другое большое число с помощью стандартного приема – логарифмирования с последующим потенцированием.

Листинг П2.2

```
double Beta (double a,double b)
//
// Функция вычисления B-функции.
// Обозначения:
//  a - аргумент,
//  b - аргумент.
// Возвращаемое значение:
//  значение B-функции.
//
{
 return exp (LogGamma(a) + LogGamma(b) - LogGamma(a + b));
}
```

Неполная гамма-функция

Неполной гамма-функцией называется функция вида

$$P(a,x)=\frac{1}{\Gamma(a)}\int_0^x t^{a-1}e^{-t}dt,$$

где a – положительное число,

x — параметр.

В показанной ниже функции используется 2 метода вычисления, в зависимости от значений параметров – разложение в ряд и цепная дробь. Методы гарантируют заданную точность и обеспечивают высокую скорость вычислений.

Листинг П2.3

```
double Gamma (double a, double x)
//
// Функция вычисления неполной Г-функции.
// Обозначения:
//  a - параметр,
//  x - параметр.
// Возвращаемое значение:
//  значение неполной Г-функции.
// Примечание:
//  функция основана на разработке GINCGM из Брандт З. Анализ данных.
//  Статистические и вычислительные методы для научных работников и
//  инженеров. - М.: Мир, ООО "Издательство АСТ", 2003, с. 567.
//
{
 double i,              // Счетчик бесконечного цикла
        cf,             // Значение функции
        cfn,            // Значение функции на предыдущем шаге
        f,s,an,h,       // Рабочие параметры разложения
        a0,a1,b0,b1,    // Параметры расчета цепной дроби
        a2,a21,b2,b21,  // Коэффициенты  аппроксимации
        fn;             // Нормирующий множитель

 ////////////////////////////
 // Выбор формулы аппроксимации
```

```
if (x <= a + 1)
  {
   ////////////////////
   // Разложение в ряд
   f = 1 / a;
   s = f;
   an = a;
   for (i = 1;; i++)
     {
      f *= x / ++an;
      s += f;
      if (f < 1E-6)
        break;
     }
   if (x < 1E-6)
     return 0;
   else
     {
      h = a * log (x) - x - LogGamma(a);
      if (fabs (h) >= 500)
        return 0;
      else
        return s * exp(h);
     }
  }
else
  {
   ////////////////
   // Цепная дробь
   a0 = 0;
   b0 = 1;
   a1 = 1;
   b1 = x;
   cf = 1;
   fn = 1;
   for(i = 1;;i++)
     {
      a2 = i - a;
```

```
        a21 = i;
        b2 = 1;
        b21 = x;
        a0 = (b2 * a1 + a2 * a0) * fn;
        b0 = (b2 * b1 + a2 * b0) * fn;
        a1 = b21 * a0 + a21 * a1 * fn;

        if ((b1 = b21 * b0 + a21 * b1 * fn) != 0)
          {
           /////////////////
           // Перенормировка
           fn = 1 / b1;
           cfn = a1 * fn;

           ////////////////////////////////////////
           // Проверка достижения заданной точности
           if (fabs(cf - cfn) / cf < 1E-6)
             break;
           cf = cfn;
          }
      }
    h = a * log(x) - x - LogGamma(a);
    if (fabs(h) >= 500)
      return 1;
    else
      return 1 - exp(h) * cfn;
   }
}
```

Неполная бета-функция

Неполной бета-функцией называется функция вида

$$I_x(a,b) = \frac{1}{\mathrm{B}(a,b)} \int_0^x t^{a-1}(1-t)^{b-1}\,dt,$$

где a и b — положительные числа,

x — параметр.

В показанной ниже функции используется разложение в цепную дробь. Метод гарантирует заданную точность и обеспечивает высокую скорость вычислений.

Листинг П2.4

```
double BDistribution (double aa,double bb,double xx)
//
// Функция вычисления неполной функции B-распределения.
// Обозначения:
//  aa - параметр,
//  bb - параметр,
//  xx - параметр.
// Возвращаемое значение:
//  значение неполной функции B-распределения.
// Примечание:
//  функция основана на разработке GINCBT из книги Брандт З.
//  Анализ данных. Статистические и вычислительные методы для научных
//  работников и инженеров. - М.: Мир, ООО "Издательство АСТ", 2003,
//  с. 569.
//
{
 short flag;          // Флаг выбора параметров расчета

 double m,            // Счетчик бесконечного цикла
        a,b,x,        // Параметры распределения
        d2m,d2m1,     // Коэффициенты  аппроксимации
        a1,a2,b1,b2,  // Параметры расчета цепной дроби
        ap,           // Рабочий параметр
        cf,           // Значение функции распределения
        cfn,          // Значение функции распределения на предыдущем шаге
        fn;           // Нормирующий множитель

 ////////////////////////////////////////////
 // Выбор параметров для обеспечения сходимости
 if (xx < (aa + 1) / (aa + bb + 1))
   {
    flag = 0;
    a = aa;
```

```
    b = bb;
    x = xx;
   }
else
   {
    flag = 1;
    a = bb;
    b = aa;
    x = 1 - xx;
   }
if (x < 1E-8)
   cf = 0;
else
   {
    ///////////////
    // Цепная дробь
    a1 = 1;
    b1 = 1;
    a2 = 1;
    b2 = 1 - (a + b) * x / (a + 1);
    fn = 1 / b2;
    cf = a2 * fn;
    for(m = 1;; m++)
      {
       ap = a + 2 * m;
       d2m = m * (b - m) * x / (ap - 1) / ap;
       d2m1 = -(a + m) * (a + b + m) * x / ap / (ap + 1);
       a1 = (a2 + d2m * a1) * fn;
       b1 = (b2 + d2m * b1) * fn;
       a2 = a1 + d2m1 * a2 * fn;

       if ((b2 = b1 + d2m1 * b2 * fn) != 0)
         {
          ////////////////////
          // Перенормировка
          fn = 1 / b2;
          cfn = a2 * fn;
```

```
          ////////////////////////////////////////
          // Проверка достижения заданной точности
          if (fabs(cf - cfn) / cf < 1E-8)
            break;
          cf = cfn;
         }
      }
    cf *= PowerOf(x,a) * PowerOf(1 - x,b) / a / Beta(a,b);
   }
 if (flag != 0)
   return 1 - cf;
 else
   return cf;
}
```

Нормальное распределение

Функция плотности нормального распределения имеет вид:

$$f(x)=\frac{1}{\sigma\sqrt{2\pi}}e^{-\frac{(x-a)^2}{2\sigma^2}},$$

где a — обычно среднее арифметическое, $-\infty<a<\infty$,

x — параметр, $-\infty<x<\infty$,

σ^2 — дисперсия, $\sigma>0$.

Путем введения нормированного отклонения

$$t=\frac{x-a}{\sigma},$$

где t — стандартизованная (стандартная) случайной величины,

формуле плотности нормального распределения придан иной вид

$$f(t)=\frac{1}{\sqrt{2\pi}}e^{-\frac{t^2}{2}}.$$

Функция плотности нормального распределения имеет краткое обозначение $N(a,\sigma^2)$. Стандартное нормальное распределение обозначается как $N(0,1)$. Функция стандартного нормального распределения равна

$$\Phi_0(x) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{x} e^{-y^2/2} dy$$

и называется функцией Лапласа (вторым законом распределения Лапласа) либо интегралом вероятности Гаусса (законом Гаусса, гауссовым распределением) в честь применения данного закона распределения для изучения ошибок наблюдений.

Быстрое вычисление функции стандартного нормального распределения с заданной точностью обеспечивает как прямое интегрирование, так и метод разложения в ряд. На листинге показано вычисление через неполную гамма-функцию по формуле

$$\Phi_0(x) = \frac{1}{2}\left[1 + sign(x) P\left(\frac{1}{2}, \frac{x^2}{2}\right)\right].$$

Листинг П2.5

```
double NormalDistribution (double x)
//
// Функция вычисления стандартного нормального распределения.
// Обозначения:
//  x - стандартизированная случайная величина.
// Возвращаемое значение:
//  значение функции стандартного нормального распределения.
//
{
 if (x >= 0)
   return 0.5 + 0.5 * Gamma(0.5,0.5 * x * x);
 else
   return 0.5 - 0.5 * Gamma(0.5,0.5 * x * x);
}
```

В случае n-мерного нормального распределения плотность распределения совокупности определяется формулой

$$P(X) = \frac{1}{(2\pi)^{n/2} |S|^{1/2}} e^{-\frac{1}{2}(X-\overline{X})' S^{-1} (X-\overline{X})},$$

где S —дисперсионно-ковариационная матрица,

$\overline{X}$ — вектор математического ожидания,

штрих означает операцию транспонирования.

Хи-квадрат - распределение

Сначала рассмотрим Г-распределение (гамма-распределение). Его плотность определяется формулой

$$f(x)=\begin{cases}\dfrac{b^a}{\Gamma(a)}x^{a-1}e^{-bx}, x\geq 0,\\ 0, x<0.\end{cases}$$

где a и b — положительные числа,

x — параметр, $-\infty<x<\infty$.

χ^2-распределение (хи-квадрат распределение) является частным случаем Г-распределения с параметрами $a=n/2, b=1/2$,

поэтому его плотность равна

$$f(x)=\begin{cases}\dfrac{1}{2^{n/2}\Gamma(n/2)}x^{n/2-1}e^{-x/2}, x\geq 0,\\ 0, x<0,\end{cases}$$

где n — параметр.

Вычисление функции хи-квадрат-распределения часто производится по формуле

$$F_n(x)=1-P_n(x),$$

где $P_n(x)$ — интеграл вероятностей χ^2, вычисляемый по формуле

$$P_n(x)=\frac{1}{2^{n/2}\Gamma(n/2)}\int_x^\infty y^{n/2-1}e^{-y/2}dy.$$

На листинге показано вычисление функции хи-квадрат-распределения через неполную гамма-функцию по формуле

$$F_n(x)=P\left(\frac{n}{2},\frac{x}{2}\right).$$

Листинг П2.6

```
double ChiSquareDistribution (double n,double f)
//
// Функция вычисления функции хи-квадрат-распределения.
// Обозначения:
//  n - параметр,
//  f - параметр.
// Возвращаемое значение:
//  значение функции хи-квадрат-распределения.
//
{
 return Gamma (0.5 * n,0.5 * f);
}
```

Распределение Стьюдента

Плотность *t*-распределения Стьюдента вычисляется по формуле

$$f(x)=\frac{\Gamma\left((a+1)/2\right)}{\sqrt{a\pi}\,\Gamma\left(a/2\right)}\left(1+\frac{x^2}{a}\right)^{-\frac{a+1}{2}},$$

где x — параметр, $-\infty<x<\infty$,

a — положительное число.

Функция *t*-распределения определяется по формуле

$$S_n(t)=\frac{1}{\sqrt{n\pi}}\frac{\Gamma((n+1)/2)}{\Gamma(n/2)}\int_{-\infty}^{t}\left(1+\frac{u^2}{n}\right)^{-\frac{n+1}{2}}du,$$

где n — параметр.

В представленной функции вычисления распределения Стьюдента учитывается его взаимосвязь с неполной B-функцией.

$$S_n(t)=\frac{1}{2}\cdot\left\{1+sign(t)\left[1-I_{n(n+t^2)}\left(\frac{n}{2},\frac{1}{2}\right)\right]\right\}.$$

Листинг П2.7

```
double TDistribution (double n,double t)
//
// Функция вычисления функции t-распределения.
// Обозначения:
//  n - параметр,
//  t - параметр.
// Возвращаемое значение:
//  значение функции t-распределения.
//
{
 if (t >= 0)
   return 1 - 0.5 * BDistribution(0.5 * n,0.5,n / (n + t * t));
 else
   return 0.5 * BDistribution(0.5 * n,0.5,n / (n + t * t));
}
```

В некоторых задачах требуется вычисление обратной функции распределения. Продемонстрируем методику ее вычисления на примере обратной функции *t*-распределения.

Интервалы локализации корня заранее известны, поэтому сразу применяется простейший метод деления отрезка пополам, хотя может использоваться любой метод локальной оптимизации. Уточнение корня производится до получения заданной точности.

Листинг П2.8

```
double InverseTDistribution (double n,double p)
//
// Функция вычисления обратной функции t-распределения.
// Обозначения:
//  n - нецелый параметр,
//  p - параметр.
// Возвращаемое значение:
//  значение обратной функции t-распределения.
//
```

```
{
 double c = 0, // Левая граница локализации корня
        d = 7, // Правая граница локализации корня
        y,     // Середина интервала локализации
        cx;    // Значения функции на конце интервала

 ////////////////////////////////////////////////
 // Уточнение корня методом деления отрезка пополам
 do
   {
    cx = p - TDistribution (n,c);
    y = (c + d) / 2;
    cx *= p - TDistribution (n,y);
    if (!cx)
      return y;
    else if (cx > 0)
      c = y;
    else
      d = y;
   } while (fabs (c - d) > 0.000001);

 return y;
}
```

Рекомендуемая литература

1. Большев Л.Н., Смирнов Н.В. Таблицы математической статистики. — М.: Наука, 1983.
2. Брандт З. Анализ данных. Статистические и вычислительные методы для научных работников и инженеров. — М.: Мир, ООО «Издательство АСТ», 2003.
3. Вероятность и математическая статистика: Энциклопедия. — М.: Научное издательство «Большая Российская энциклопедия», 1999.
4. Гайдышев И. Анализ и обработка данных: специальный справочник. — СПб: Питер, 2001.

5. Де Гроот М. Оптимальные статистические решения. — М.: Мир, 1974.
6. Мюллер П., Нойман П., Шторм Р. Таблицы по математической статистике. — М.: Финансы и статистика, 1982.
7. Петрович М.Л., Давидович М.И. Статистическое оценивание и проверка гипотез на ЭВМ. — М.: Финансы и статистика, 1989.
8. Поллард Дж. Справочник по вычислительным методам статистики. — М.: Финансы и статистика, 1982.
9. Попов Б.А., Теслер Г.С. Вычисление функций на ЭВМ. Справочник. — Киев: Наукова думка, 1984.
10. Тюрин Ю.Н., Макаров А.А. Статистический анализ данных на компьютере. — М.: ИНФРА-М, 1999.
11. Уилкс С. Математическая статистика. — М.: Наука, 1967.

ПРИЛОЖЕНИЕ 3

Разработка консольных приложений на C/C++

«То, что можно объяснить посредством меньшего, не следует выражать посредством большего».
У. Оккам

Для работы с материалами приложения понадобятся лицензионные средства разработки и тестирования программ:

- Borland C++ 5.5 или выше,
- Bloodshed Dev-C++ 4.01 или выше,
- Microsoft Visual C++ 5.0 или выше.

Потребуется знание C.

Решаемая проблема

Консольные приложения хорошо подходят для тестирования алгоритмов и использования в исследовательских целях.

Здесь показаны решения, выполненные на языке C++ в виде самостоятельных приложений. Функции C++ отличаются от эквивалентных им VBA-функций высоким быстродействием. Они работают в сотни раз быстрее. Эти функции предназначены для создания быстродействующих программ в сложных проектах, поэтому приводятся нами наравне с VBA-вариантами.

Все программы построены как консольные приложения. Для разработки консольных приложений можно применить любой компилятор для Micro-

soft Windows. Задачи можно также построить как приложения операционной системы MS-DOS, с помощью любого компилятора C++, с учетом специфических ограничений MS-DOS на размер массива.

Консольное приложение на Visual C++

Для создания нового проекта консольного приложения Windows в Microsoft Visual C++ выберите из меню интегрированной среды разработки Visual C++ пункт **File ▸ New ▸ Projects ▸ Win32 ▸ Console Application**, укажите имя проекта, адрес на диске и нажмите кнопку **OK**, как показано на рис. П3.1.

Рис. П3.1. Окно Microsoft Visual C++ 5 с проектом Adams и консольным Win32-приложением

Visual C++ при запуске консольного приложения из интегрированной среды разработки не закрывает окно приложения автоматически после окончания. Он самостоятельно добавляет стандартное сообщение Press any key to continue (нажмите любую клавишу для продолжения), позволяя наблюдать на экране результаты вычислений.

Консольное приложение на Bloodshed Dev-C++

Для аналогичных целей в Bloodshed Dev-C++ выберите из меню интегрированной среды разработки **File ▸ New Project ▸ Console Application**, нажмите кнопку **OK** и укажите имя проекта, адрес на диске, как показано на рис. П3.2.

Рис. П3.2. Окно Bloodshed Dev-C++ 4
с успешно скомпилированным проектом Merson

Dev-C++ сразу после окончания решения закрывает окно приложения, поэтому программисту приходится позаботиться о том, чтобы это произошло только после того, как он ознакомится с результатами расчета. Чтобы можно было посмотреть результаты расчета, применяются разнообразные решения: выдача в файл из самой программы, перенаправление выдачи в файл средствами операционной системы и другие решения.

Рекомендуемая литература

1. Рихтер Дж. Windows для профессионалов: создание эффективных Win32-приложений с учетом специфики 64-разрядной версии Windows. – СПб: Питер; М.: Издательско-торговый дом «Русская Редакция», 2001.

ПРИЛОЖЕНИЕ 4

Средства программирования (обзор)

«Software is like sex: it's better when it's free».[1]
Л. Торвальдс

Известно более 2000 языков программирования. Некоторые алгоритмические языки создавались для решения узких специальных задач, другие являются универсальными.

Научное программирование предназначено для решения различных задач научного, технического, производственного и практического характера и имеет свои специфические особенности, подобно тому, как отличаются друг от друга создание программ обработки изображений, игр, программирование обучающих программ. Для различных отраслей использование библиотек прикладных программ имеет специфику.

Компиляторы преобразуют исходный текст программы (код) на тех или иных алгоритмических языках, в объектный (машинный) код. Непосредственно пошаговое выполнение кодов производит интерпретатор, быстродействие такой программы сравнительно ниже. У компилируемой и интерпретируемой программ имеются свои достоинства и недостатки. Современные технологии программирования позволяют комбинировать решения, выявляя лучшие свойства каждого решения. Например, известны и компиляторы, и интерпретаторы языка C или Basic. Существуют и доступны специальные конверторы программ Basic в эквивалентный код C, например, BCX (Kevin Diggins, **www.download.com**). Доступна программа для конвертирования кода

---

[1] *В переводе теряется прелесть оригинальной цитаты. Тем не менее, «программное обеспечение похоже на секс: оно лучше, когда бесплатно».*

Visual Basic в код Delphi, а именно VB2Delphi Converter (Levent Baykan Bayar, **www.geocities.com/leventbbayar/vb2delphi.htm**).

Постоянно находятся желающие положить свой камень в основание Вавилонской башни мирового программирования. Но есть универсальные программные средства, пригодные для использования в любых отраслях. Можно указать следующие особенности научного программирования, в сравнении с другими видами программирования:

- применение изощренных математических алгоритмов;
- повышенная трудоемкость вычислений, заставляющая искать новые алгоритмы и программные решения, чтобы получить приемлемую скорость вычислений на данном этапе развития аппаратного и системного обеспечения;
- обеспечение и доказательство гарантированной точности вычислений;
- повышенная надежность программ, особенно в автоматизированных системах сбора и анализа данных;
- научные программы часто не предназначаются для широкого распространения, их коммерческий успех не связан с непосредственными продажами;
- научные программы иногда имеют стратегическое назначение, что создает определенные трудности в их распространении и совершенствовании, поэтому остаются в единственном экземпляре;
- стойкость алгоритмов научных программ, обоснованная математическим аппаратом.

Нет особой важности в выборе языка программирования для решения прикладной задачи, некоторые авторы создают псевдокод, как выражение основных логических конструкций языка программирования высокого уровня, пригодный для быстрого переложения на любой алгоритмический язык.

Операционные системы

Говоря о персональных компьютерах (Personal Computer, PC), обычно имеют в виду компьютеры, разработанные на основе 32-разрядного процессора, впервые созданного компанией Intel Corp. (**www.intel.com**), основанной в 1968 г. Г. Муром (Gordon Moore), Р. Нойсом (Robert Noyce) и Э. Гроувом (Andrew Grove). В спецификации при этом указывают, что «система IBM PC-совместима» в память о персональном компьютере, разработанном под руководством Д. Эстриджа в подразделении Entry Systems Division, возглавляемым Б. Лоу, и выпущенным International Business Machines Corp. (IBM, **www.ibm.com**) в 1981 г. Более современное название данной платформы —

Intel (x86), просто x86 или IA–32. 16-разрядный процессор Intel 8086, давший наименование семейству, разработан Intel в 1978 г., 32-разрядный Intel 80386 – в 1985 г., а процессор Intel Pentium – в 1993 г. Различными производителями поставляются также клоны Intel-совместимых процессоров и основанные на них решения.

Другие персональные аппаратные платформы распространены меньше PC. Компанией Apple Computer Inc. (**www.apple.com**), основанной в 1975 г. под наименованием Apple Computer Company и окончательно сформированной в 1977 г. в том же году был выпущен персональный компьютер Apple, созданный С. Возняком (Steve Wozniak) и С. Джобсом (Steve Jobs). В 1983 г. Apple был представлен персональный компьютер Lisa с графической операционной системой, не получивший распространения из-за очень высокой цены, но положивший начало системе Macintosh. Основа Macintosh – 16-разрядный процессор Motorola 68000 был выпущен в 1979 г., а 32-разрядный Motorola 68020 в 1984 г. Современные модели Apple базируются на процессорах PowerPC производства IBM и Motorola (**www.motorola.com**). Первый процессор Motorola PowerPC 601 был выпущен в 1993 г.

Основной тенденцией развития операционных систем, интересной пользователю, является сближение их функциональных возможностей и пользовательского интерфейса. Уже сейчас, когда мы предлагали нашим пользователям, не являющимся программистами, поработать с распространенными приложениями (браузер, текстовый процессор, электронные таблицы), многие после достаточно продолжительного времени работы с удивлением узнавали, что работали они не с Microsoft Windows и Microsoft Office, а со специально настроенным интерфейсом операционной системы ALT Linux и входящими в комплект ее поставки офисными пакетами. Еще более благоприятно складывается ситуация для англоязычных пользователей, для которых возможности выбора операционной системы и прикладных программ очень широки. Сближению различных операционных систем в первую очередь способствовали Интернет и Интранет-технологии , связанные с ними стандарты (форматы файлов, протоколы), стандарты приложений и внедрение цифровых технологий в бытовые устройства.

Кратко перечислим основные ОС, применяемые на персональных компьютерных системах, а затем рассмотрим языки программирования и средства разработки приложений.

UNIX, GNU и Linux

Начать исторический обзор операционных систем предлагаем с системы UNIX, разработанной К. Томпсоном (Ken Thompson) в 1969 г. в AT&T Bell Laboratories (сегодня Bell Laboratories – научно-исследовательское подразделение компании Lucent Technologies). В настоящее время UNIX (Unix) –

это и название семейства операционных систем, и общеупотребительный термин, и торговая марка.

Компания SCO (**www.sco.com**, **www.thescogroup.com**), созданная в 1979 г., произвела первый пакет UNIX System (названный SCO XENIX System V) для PC на базе процессоров Intel 8086 and 8088 в 1983 г. В 1985 г. SCO выпустила операционную систему SCO XENIX 286 для компьютеров на базе Intel 80286. С 1987 г. поставлялась SCO XENIX 386, первая 32-битная операционная система UNIX для компьютеров на базе процессоров Intel 386. Э. Танненбаум выпустил Minix, версию UNIX для PC, Mac, Amiga, and Atari ST, в 1987 г. С 1989 г. поставлялась SCO UNIX System V/386. В том же году SCO представляет первый 32-битный графический пользовательский интерфейс SCO Open Desktop для UNIX на Intel-совместимых компьютерах. В 1995 г. компания SCO выкупила всю интеллектуальную собственность и обязательства в области UNIX у AT&T. В 2003 г. технологии компании SCO лицензировала корпорация Microsoft Corp. (**www.microsoft.com**), с 1980 г. по лицензии AT&T уже поставлявшая операционную систему XENIX для компьютеров Intel 8086, Zilog Z8000, Motorola M68000, DEC PDP–11.

Из современных UNIX для Intel (x86) отметим операционную систему FreeBSD (BSD – Berkeley Standard Distribution), разработанную в рамках FreeBSD Project (**www.freebsd.org**), впервые выпущенную в 1993 г. FreeBSD ведет начало, также как NetBSD и OpenBSD, от операционной системы Berkeley UNIX, соданной Б. Джоем (Bill Joy).

Широкое распространение имеет операционная система Solaris Intel Platform Edition, разработанная компанией Sun Microsystems Inc. (**www.sun.com**), которую основали в 1982 г. Б. Джой, В. Косла (Vinod Khosla), С. МакНили (Scott McNealy) и Э. Бехтольсхайм (Andy Bechtolsheim). Основные UNIX-технологии компания Sun лицензировала в 1994 г. В настоящее время компания Sun активно продвигает на рынке программный продукт Java Desktop System, представляющий собой ядро Linux, среду GNOME и множество программных инструментов: Java 2 Standard Edition, Star Office (см. *приложение 1*), браузер Mozilla.

Компанией IBM выпускается собственная версия UNIX под названием AIX. К 2003 г. система AIX занимает 15% рынка серверов UNIX, однако в 2003 г. лицензия на UNIX была отозвана у IBM компанией SCO. Собственные версии UNIX выпускает еще ряд крупных компаний.

В 2001 г. выпущена операционная система Mac OS X, развитие приобретенной компанией Apple 1997 г. операционной системы FreeBSD и базирующейся на UNIX операционной системы OpenStep от компании NeXT, основанной в 1986 г. С. Джобсом. Напомним, что первая версия «классической» Mac OS, названная System, выпущена в 1984 г. Это была первая коммерческая операционная система с графическим пользовательским интерфейсом.

Операционная система Mac OS System 7.0 существовала и в версии для платформы x86, но в 1993 г. проект был закрыт.

UNIX-подобной ОС является Linux (**www.linux.org**), разработка которой началась в 1991 г. по инициативе Л. Торвальдса (Linus Torvalds), и в 1992 г. была поддержана проектом GNU (GNU's Not Unix – ГНУ Не Юникс, произносится как «гу-нью») под эгидой сообщества Free Software Foundation Inc. (**www.gnu.org**), созданного в 1984 г. Р. Столлменом (Richard M. Stallman) для разработки свободного клона UNIX, поэтому новая ОС стала называться GNU/Linux. Чтобы оградить себя и мировое сообщество свободных разработчиков от мошенников и спекулянтов, торговая марка Linux была зарегистрирована Л. Торвальдсом в 1995 г. в Германии.

До рассмотрения конкретных продуктов сначала необходимо прояснить ситуацию взаимоотношений UNIX, GNU и Linux. Позволим себе достаточно объемную оригинальную цитату, снимающую все вопросы. Как совершенно ясно из комментария Р. Столлмена, «Unix – это целая операционная система, а Linux – это лишь часть операционной системы GNU/Linux называется операционная система GNU, в которой Linux работает в качестве ядра. Ядро – это часть системы, занимающаяся выделением ресурсов машины под другие программы. Эта часть и есть сам Linux». И далее, «В 1992 году Линус Торвальдс сделал свое ядро, Linux, свободным программным обеспечением. Другие люди объединили GNU и Linux, создав первую полностью свободную операционную систему Сегодня GNU может работать с разными ядрами. Кроме Linux, может использоваться наше ядро GNU Hurd и ядро NetBSD. В основном, это одна и та же система, какое бы ядро мы не использовали». И, наконец, «сам Linux здесь уже не важен: система GNU стала популярна в сочетании с ядром Linux, но сегодня она работает еще с двумя ядрами BSD и ядром GNU».

Дистрибутивы Linux (их более 200) от различных поставщиков, хотя и состоят из сходных наборов системных и прикладных программ, различаются по качеству. Рекомендуем (в алфавитном порядке):

- Debian GNU/Linux (**www.debian.org**), впервые выпущенный в 1993 г. И. Мэрдоком (Ian Murdock), причем слово Debian произносится как «дэбиэн» и образовано из имени создателя Debian И. Мэрдока (Ian Murdock) и его жены Дебры (Debra). На дистрибутиве Debian Linux основан программный продукт под названием Xandros Desktop (Xandros Inc., **www.xandros.com**);

- Linux-Mandrake (MandrakeSoft S.A., **www.mandrakesoft.com**), впервые выпущенный в 1998 г.;

- Linux Slackware (Slackware Linux Inc., **www.slackware.com**), впервые выпущенный П. Волкердингом (Patrick Volkerding) в 1993 г.;

- Red Hat Linux (Red Hat Software Inc., **www.redhat.com**), впервые выпущенный М. Эвингом (Marc Ewing) в 1994 г. и в 1995 г. приобретенная компанией ACC Corporation, созданной в 1993 г. Б. Янгом (Bob Young), которая получила имя Red Hat Software. Дистрибутив Red Hat с 2003 г. продается (без возможности бесплатной загрузки) только в качестве сервера Red Hat Enterprise Linux. Для рабочих станций компания Red Hat Software Inc. предлагает дистрибутив Fedora Core, разрабатываемый в рамках спонсируемого ею Fedora Project (fedora.redhat.com);
- дистрибутив SUSE LINUX, выпущенный компанией SUSE LINUX AG (SUSE Inc., **www.suse.com**), в 2003 г. приобретенной компанией Novell Corp. (**www.novell.com**). Для открытых систем вопрос, какой национальной компанией контролируется тот или иной программный продукт, не важен или не так важен, как для закрытых систем.

Для отечественного пользователя выбор дистрибутива не так широк. На данный выбор влияет наличие локализованных приложений. Отметим качественные отечественные продукты московских компаний ALT Linux (ООО «АЛЬТ Линукс», **www.altlinux.com**), на основе Mandrake, и ASPLinux (ASPLinux, **www.asplinux.ru**), на основе Red Hat. Авторитет данных дистрибутивов очень высок: ALT Linux прекрасно подходит и для домашнего компьютера, и для рабочей станции, и для развертывания сервера Интернет. Операционная система ASPLinux с успехом применяется и как настольная система, и как сервер, на совместимость, прежде всего с операционной системой ASPLinux, тестируются серверы крупных производителей.

В 2001 г. компания ASPLinux объявила о присоединении Донбасского коллектива Black Cat Linux Team, разработчиков популярного одноименного дистрибутива Linux. Объединенную команду разработчиков возглавил Л. Кантер, лидер Black Cat Linux Team.

В 2003 г. компания «АЛЬТ Линукс» объявила о начале продаж защищенной интегрированной системы (ЗИС) «Утес-К», созданной совместно с НПФ «Промтехн» и предназначенной для работы с конфиденциальной информацией. Система сертифицирована Гостехкомиссией при Президенте Российской Федерации по 5 классу средств вычислительной техники (СВТ) защиты конфиденциальной информации, 4 уровню контроля отсутствия недекларированных возможностей.

Системы UNIX применяются для работы с приложениями, требующими особой надежности (системы управления технологическими процессами, медицинские лечебно-диагностические комплексы, серверы локальных и глобальных компьютерных сетей, коммуникационные серверы). Европейской комиссией финансируется проект «Промышленные встраиваемые системы» (Industrial Embedded Systems, INES), который ставит своей целью внедрение открытого программного обеспечения в управление станками, роботами, производственными комплексами и другим промышленным обо-

рудованием. В проекте принимают компании и научные организации из Бельгии, Великобритании, Италии, Словении, Румынии и Швеции.

Диапазон применений Linux необычайно широк. Некоторые характерные применения:

1. сетевой маршрутизатор (включая http-сервер) FREESCO (FREE ciSCO, **www.freesco.org**), умещающийся на одну дискету (начиная с версии 0.3.1, легко устанавливается и на жесткий диск);
2. встроенная операционная система наладонных компьютеров и других устройств, таких как сетевые камеры;
3. суперкомпьютер МВС–1000М, собранный на процессорах Alpha и работающий под управлением Linux, обеспечивает вычислительные потребности Российской Академии Наук;
4. компанией Silicon Graphics Inc. построен 128-процессорный Linux-кластер[2] Altix на процессорах Intel Itanium 2;
5. компанией Cray Inc. (**www.cray.com**) в 2003 г. принято решение использовать для управления хранением данных строящегося суперкомпьютера Red Storm систему SUSE LINUX. Суперкомпьютер Red Storm оснащен 16 тысячами процессоров AMD Opteron. Использоваться суперкомпьютер будет для разработок в областях создания ядерного оружия и атомной энергетики;
6. в подразделении IBM Research ведутся работы по адаптации Linux к сверхмощным вычислительным системам: первый компьютер серии Blue Gene/L будет иметь 65 тысяч процессоров и 16 терабайт оперативной памяти. Суперкомпьютер сооружается для Ливерморской национальной лаборатории имени Лоуренса (США), которая занимается разработкой ядерного оружия;
7. суперкомпьютер OctigaBay 12K (OctigaBay Systems Corp., **www.octigabay.com**). Основу компьютера составляют двенадцать 64-разрядных процессоров AMD Opteron. Допускается объединение в один кластер до 1000 серверных стоек, позволяя использовать одновременно 12000 процессоров с суммарной мощностью в 60 терафлопс.

Linux не является операционной системой реального времени (ОСРВ), хотя известны попытки создать на базе Linux ОСРВ. Для сетевых приложений (особенно в глобальных сетях) на сегодняшний день решения на базе FreeBSD и Linux – оптимальные. Linux используют практически все ведущие киностудии, занимающиеся анимацией и спецэффектами: DreamWorks, Digital Domain, Lucas Digital, Imageworks и другие. Linux работает на систе-

---

[2] *По определению Ш. Дейга (Sean Dague) из Технологического Linux центра при IBM, кластер – это «более, чем одна машина, работающая совместно над одной либо несколькими задачами».*

мах рендеринга, насчитывающих тысячи процессоров. Известно о попытке применения Debian GNU/Linux для управления любительской ракетой для суборбитальных полетов.

Операционная система Linux и разработки на ее базе широко используются в системе министерства обороны США. Национальной криптологической организацией National Security Agency (NSA, **www.nsa.gov**) разработана версия Linux повышенной безопасности под наименованием SELinux (Security-Enhanced Linux). В числе других применений SELinux была внедрена в специализированное военное программное обеспечение, управляющее системами оружия американской армии. На основе Linux Российскими разработчиками создана операционная система МСВС (мобильная система вооруженных сил) для министерства обороны России. Особенность МСВС 3.0 – встроенные средства защиты от несанкционированного доступа, удовлетворяющие требованиям Руководящего документа Гостехкомиссии по 2 классу СВТ.

Операционная система Linux и приложения для нее с успехом используются также и теми, кто намерен прекратить нелегальное использование коммерческого программного обеспечения, но не имеет достаточно средств для решения проблемы. Задача имеет два решения: приобрести лицензии либо перейти на бесплатные альтернативы. Правительства некоторых государств и муниципальные власти отдельных городов, например, Мюнхена (Бавария, Германия), заявили о намерении перевести на Linux 14000 служебных компьютеров. Более 500 правительственных организаций Германии перешли на SUSE LINUX. Все государственные учреждения и компании Вьетнама к 2005 г. должны перейти на использование свободного программного обеспечения: на каждый компьютер, продаваемый в стране, в обязательном порядке будет ставиться ОС Linux. Правительство Бразилии уже заключило с компанией IBM соглашение о продвижении Linux в общественном секторе. Аналогичные соглашения были заключены с российским Министерством связи и информатизации, которое согласилось на создание Центра образования Linux в Москве.

В 2003 г. объявлено о совместных планах Японии, Южной Кореи и Китая по созданию (вероятно, на базе GNU/Linux) операционной системы, альтернативной Microsoft Windows. Основной целью проекта является снижение зависимости от иностранного монополиста и возвращение национальной информационной безопасности на должный уровень.

Объявили о продвижении Linux на рынке настольных систем компания IBM (оказывающая большую поддержку продвижению Linux на рынке серверов) и организация Open Source Development Lab (OPDL), в числе других членов включающая компании Hewlett-Packard, Dell и Intel. Планируется создать технологию, облегчающую обмен данными и документами между приложениями Linux и Windows. Данные технологии уже существуют и на-

зываются Интранет и Samba (**www.samba.org**). Мы не встретили каких-либо особых трудностей в установке, настройке и эксплуатации Samba в нашей небольшой локальной корпоративной компьютерной сети, насчитывающей около трех сотен машин. Серверы и рабочие станции работали под управлением Sun Solaris, FreeBSD, ALT Linux, Microsoft Windows 2000 Server и Professional, Microsoft Windows NT 4.0, Microsoft Windows XP Home и Professional Edition, Microsoft Windows 95/98/98SE/Me, FREESCO, выполняя разнообразные задачи, активно обмениваясь данными в различных форматах и совместно используя базы данных.

Microsoft MS-DOS и Windows

Операционная система Microsoft MS-DOS была разработана в 1981 г. на основе выпущенной компанией Seattle Computer Products операционной системы 86-DOS (известной также под наименованием QDOS – Quick and Dirty Operating System), созданной Т. Патерсоном (Tim Paterson) в 1979 г., перешедшим впоследствии на работу в Microsoft. Окончательно все права на QDOS были куплены Microsoft в 1985 г. Напомним, что корпорация Microsoft была основана в 1975 г. П. Алленом (Paul Allen) и У. Гейтсом (William H. Gates III, Bill Gates, Билл Гейтс). В 1976 г. зарегистрирована торговая марка Microsoft, а одноименная компания официально создана в 1977 г.

Аналогом MS-DOS является операционная система PC DOS, разработанная Microsoft по контракту и поставляемая в настоящее время компанией IBM. Помимо операционной системы, в Microsoft разработали для компьютера IBM PC языки Бейсик, Паскаль и Ассемблер 8088, а затем компиляторы с Фортрана и Кобола. Русскоязычная версия MS-DOS 4.01 была выпущена в 1990 г. Более удачной, по нашему мнению, следует считать частичную локализацию MS-DOS, выполненную в одной из последних версий 6.22.

История Microsoft Windows началась в 1985 г., когда отдел операционных систем Microsoft возглавил С. Балмер (Steve Ballmer). Для MS-DOS была разработана графическая оболочка Microsoft Windows. По официальной информации, полученной из оригинальной сопроводительной документации (мы просто сравнили надписи на коробках с Microsoft Windows 3.0 и 3.1), начиная с версии 3.1, выпущенной в 1992 г., разработчик называет Microsoft Windows графической операционной системой.

Известно большое число вариантов версий Microsoft Windows 95 и 98, но замечено, что любая программа достигает совершенства к концу своего развития. Лучшей версий является операционная система Microsoft Windows 98SE (включая MS-DOS 7.1). Последней во всех отношениях версией стала Microsoft Windows Me (Millennium edition, включая MS-DOS 8.0), на которой развитие Microsoft Windows завершилось.

На том основании, что скверно составленное приложение (в том числе, к сожалению, и составленное программистами компании Microsoft) может привести систему к краху, считается, что надежность Microsoft Windows невысока, однако при качественной и исправной аппаратной части PC и использовании хороших прикладных программ (например, отечественного производства) операционная система Microsoft Windows вполне надежна. Свойственные всем версиям Microsoft Windows недоработки, в том числе не исправленные на протяжении десятилетий, известны и могут легко обходиться опытными пользователями.

Хотя установка и настройка Microsoft Windows просты и теоретически ориентированы на обычного пользователя, для надежной и продолжительной работы (в течение не 3–6 месяцев, как часто бывает у неопытных пользователей, а 5 и более лет) без сбоев — установка и настройка Microsoft Windows должны производиться квалифицированными специалистами. В этом смысле Microsoft Windows не имеет никаких преимуществ перед операционными системами, названными в предыдущем разделе. Ее успех — это успех талантливого агрессивного маркетинга. Microsoft Windows — очень удобная операционная система для пользователей и программистов, если не заставлять ее делать то, чего она не умеет, и не пользоваться ее недокументированными функциями (размышления о недокументированных функциях см. в *Заключении*).

Microsoft Windows NT

В 1987 г. Microsoft опубликовала Operating System 2 (OS/2). Операционная система Microsoft OS/2 версии 3.0 после завершения в 1990 г. сотрудничества Microsoft и IBM в области создания операционной системы в 1991 г. была переименована в Microsoft Windows NT. Разработка велась под руководством Д. Катлера, и формально ОС Microsoft Windows NT увидела свет в 1993 г.

Под управлением Microsoft Windows NT 4.0, существующей в вариантах рабочей станции и сервера, могут выполняться многие приложения, созданные для Microsoft Windows, но по утверждению одного из ведущих разработчиков Windows NT З. Вудолла, «уходит много времени, чтобы заставить все WIN32-приложения, разработанные под Windows 95, нормально работать под управлением Windows NT». Данные усилия не всегда увенчивались успехом, и многие программы имели описанные проблемы, что вызвано, видимо, использованием недокументированных возможностей, неэффективной системой управления проектами и недостаточно квалифицированным тестированием.

Дальнейшим развитием Microsoft Windows NT 4.0 стала версия Microsoft Windows 2000 (Windows NT 5.0), существующая в профессиональной версии и в нескольких версиях серверов. В 2001 г. выпущена операционная система

Microsoft Windows XP (Windows NT 5.1) в версиях для домашнего и профессионального применения. В 2003 г. создана операционная система Microsoft Windows Server 2003.

Операционные системы Microsoft Windows 2000 и XP показались нам очень удобными и надежными. Они оказались намного более дружественными к пользователям, чем системы Linux на современном этапе развития. Требования к качеству всех компонентов аппаратной части компьютера и ресурсов (дисковое пространство, память, быстродействие процессора), в настоящее время, вряд ли можно признать чрезмерными.

Операционные системы семейства Microsoft Windows 4.0 на момент составления данной информации уже не поддерживаются производителем, однако данные ОС продолжают оставаться популярными среди разработчиков. На базе Microsoft Windows 4.0 продолжают создаваться заказные разработки, например, в области научных исследований и здравоохранения.

Установка Microsoft Windows XP с лицензированного носителя обычно не вызывает затруднений у пользователей, однако для улучшения стабильности данная процедура, по рекомендации разработчика, должна производиться квалифицированными специалистами. Некачественно выполненная установка может подорвать доверие даже к такому превосходному программному продукту, как Microsoft Windows XP. Лучшим вариантом является покупка компьютера с предустановленной операционной системой.

Операционные системы Microsoft Windows 2000 и XP, а также до сих пор эксплуатирующиеся версии 95 и 98, доминируют в качестве настольных офисных и домашних систем, графических рабочих станций, насчитывая более 300 миллионов установленных систем по всему миру. Специальные версии Microsoft Windows 2000 и 2003 широко используются в качестве файловых серверов, серверов приложений, серверов Интернет. Они успешно конкурируют с операционными системами Linux и FreeBSD по уровню надежности, поддержки, не говоря уже об удобстве работы администраторов и программистов.

2003 г. был богатым на события вокруг распространения продуктов корпорации Microsoft. Было заключено несколько крупных контрактов на поставку программного обеспечения для армии и министерства внутренней безопасности США. Механизированной дивизией «Железный конь» произведены испытания «боевого Интернет» на участке суши размером 30 на 50 км. Все танки, вертолеты и командные пункты были оснащены компьютерами с IP-адресами и получали информацию о расположении и боевом состоянии подразделений и солдат. Компьютеры работали под управлением операционной системы Microsoft Windows[3].

---

[3] *«Вокруг света», № 12, 2003 г., с.119.*

В том же году было заключено соглашение, согласно которому во всех правительственных ведомствах Бахрейна будет использоваться лицензионное программное обеспечение Microsoft. В 2003 г. первой страной, подписавшей с Microsoft соглашение в рамках программы Government Security Program, предусматривающей предоставление компетентным правительственным организациям доступа к исходному коду операционных систем Microsoft Windows, стала Россия.

Эмуляторы и клоны

По определению, эмулятором в мире компьютерной техники называют программу, которая обеспечивает виртуальную машину (виртуальную среду выполнения). Эмулятор позволяет в окнах существующей операционной системы запускать приложение, составленное для другой операционной системы, и возможно, исполняемой на базе других аппаратных средств. Данное приложение на виртуальной машине работает в существенно более медленном темпе, чем это было бы на реальных аппаратных средствах и под управлением операционной системы, для которой было составлено данное приложение.

Клоном, в отличие от биологического определения, в компьютерных науках называют аппаратную часть компьютера или программную систему, обеспечивающие реальную среду выполнения для данного приложения ту же самую, для которой было составлено приложение, но, соответственно, имеющие аппаратную конструкцию или системное программное обеспечение, отличные от тех, для которых изначально было составлено приложение. Иначе, клон можно назвать функциональным аналогом, но последнее определение менее конкретное.

Существенная черта клона — он создается другой компанией, часто на других технологических принципах и без использования патентов, защищающих оригинал, обеспечивая только интерфейс оригинала. Современное патентное право — наиболее мощное искусственно созданное препятствие, стоящее на пути научно-технического прогресса — выступает в качестве основного обстоятельства, составляющего главную причину написания клона. Другим обстоятельством является создание среды выполнения, свободной от недокументированных функций, в целях обеспечения информационной безопасности. Третьим обстоятельством можно считать энтузиазм разработчиков.

Если мы говорим о клоне операционной системы, то обязательно имеем в виду совместимость не только с приложениями, но и с драйверами аппаратных устройств, составленными для оригинала.

Эмуляторы

Роль эмуляторов трудно переоценить хотя бы по одной причине. Например, в различных организациях накопилось достаточно большое количество ус-

таревшего программного обеспечения, которое, тем не менее, продолжает выполнять свои функции на адекватном оборудовании. Сказанное относится и к информационным системам, и к автоматизированным системам сбора и анализа экспериментальной и клинической информации, и к управляющим системам. Аппаратное обеспечение, в отличие от программ, подвержено износу, а устаревшие системные программы просто не работают на новом оборудовании (и не поддерживаются производителями). Эмуляторы позволяют запускать эти приложения и, таким образом, значительно экономят финансовые средства и время пользователей. Кроме того, современные операционные системы, например, Microsoft Windows XP, не вполне совместимы с устаревшими приложениями MS-DOS. Это создает дополнительные препятствия для эксплуатации программ, в которые были вложены крупные финансовые средства, и которые могли бы приносить выгоду и дальше. Можно назвать и еще немало фактов полезности эмуляторов.

Для выполнения приложений Microsoft Windows под управлением FreeBSD, Solaris и Linux с 1993 г. компанией Wine Development HQ (**www.winehq.com**) развивается эмулятор WINE. На базе данного эмулятора предложены коммерческие решения:

- операционная система LindowsOS (**www.lindows.com**), созданная в 2002 г. под руководством М. Робертсона — дистрибутив Linux, основанный на Xandros Linux, на базе на Debian GNU/Linux. LindowsOS позволяет запускать некоторые приложения, составленные для Windows;
- программа CrossOver Office, представленная компанией CodeWeavers (codeweavers.com). Данная программа дает пользователю возможность устанавливать пакет прикладных программ Microsoft Office для Windows и работать в нем на компьютере под управлением операционной системы Linux;
- коммерческий эмулятор WineX компании Transgaming Technologies.

В стандартную поставку многих версий Linux входит еще один DOS/Windows 3.1 эмулятор DOSEMU (sourceforge.net/projects/dosemu), преобразующий системные вызовы DOS-приложений в команды для Linux. Под управлением данного эмулятора на компьютере с Linux может исполняться Microsoft Windows 3.1.

Группой The Bochs Project (bochs.sourceforge.net) развивается кросс—платформенный эмулятор Bochs. Планируется, что на рабочей станции UNIX сможет программно эмулироваться аппаратная часть Intel (x86), с целью запуска Windows и ее приложений.

Под управлением эмулятора VirtualPC 5.0, разработанного Connectix (**www.connectix.com**), в Mac OS X версии 10.1 и выше могут исполняться приложения MS-DOS и Windows. Технология данного наиболее продвинутого эмулятора в 2003 г. была приобретена Microsoft, неоднократно осуще-

ствлявшей подобные проекты самостоятельно. Еще в 1980 г. Microsoft анонсировала свой продукт под названием Z-80 SoftCard для Apple II (8-битный микропроцессор Zilog Z-80 имел набор команд, включающий набор команд процессора Intel 8080). Карта позволяла Apple II обеспечить совместимость с операционной системой CP/M, лицензированной Microsoft у компании Digital Research. Эта комбинация обеспечила компьютеру Apple II успех, позволяя исполнять популярные прикладные программы, существовавшие тогда для операционной системы CP/M. В середине 2003 г. было объявлено о включении пакета VirtualPC в профессиональную версию пакета Microsoft Office X для компьютеров Apple.

Корпорация IBM с 1994 г. поставляет операционную систему OS/2 Warp, которая может исполнять некоторые приложения Microsoft MS-DOS и Microsoft Windows 3.X. Операционная система IBM OS/2 Warp находит своих пользователей. Причиной малого распространения OS/2 Warp аналитики считают недостаточно эффективный маркетинг.

Объявлено о поддержке приложений Microsoft MS-DOS и Microsoft Windows операционной системой SkyOS, разработанной одноименной фирмой (**www.skyos.org**), и приложений MS-DOS операционной системой PETROS, разработанной компанией Trumpet Software International Pty Ltd. (**www.petros—project.com**).

Программные эмуляторы многих устаревших аппаратных компьютерных платформ на современных компьютерах представлены в обзоре №4 журнала «Мир ПК» за 2003 г. В начале 2003 г. появилась информация о создании компанией Intel программного продукта под названием IA-32 Execution Layer. Продукт выполнен в виде специального блока, встраиваемого в ядра операционных систем для новых 64-разрядных процессоров Itanium 2 и последующих модификаций, отвечающего за трансляцию на лету программ в инструкциях x86 в машинный код Itanium 2.

Клоны

Наибольшее число клонов имеет операционная система Microsoft MS-DOS. Данные операционные системы в настоящее время находят применение для разработки встроенных систем и систем реального времени. Благодаря простоте и надежности, они незаменимы там, где требуются недорогие и надежные системы автоматического сбора информации и управления экспериментальными, контролирующими или иными полезными в народном хозяйстве и обороне устройствами. Хотя поведение систем типа MS-DOS является более предсказуемым, чем, например, Microsoft Windows, они также не являются операционными системами реального времени. Обычно язык написания клонов MS-DOS — ассемблер, чем и определяются их достоинства (компактность, скорость) и недостатки (несовместимость аппаратных платформ, трудное совершенствование и сопровождение).

Операционная система DR–DOS создана в 1987 г. компанией Digital Research Inc. и в свое время называлась «лучшей DOS, чем MS-DOS». Она пришла на смену операционной системе CP/M (Control Program for Microcomputer), созданной в 1973 г. Г. Килдаллом (Gary Kildall) и в 1975 г. ставшей собственностью основанной им компании Digital Research. После CP/M компания Digital Research выпустила 16-разрядную версию CP/M-86, которая поставлялась в комплекте с персональным компьютером IBM PC, конкурируя с MS-DOS. Графическая оболочка для DR-DOS не получила распространения из-за маркетинговых просчетов.

Компания Digital Research была куплена вместе с продуктами DR-DOS и CP/M в 1990 г. корпорацией Novell Corp., известным производителем сетевых операционных систем Novell Netware и клонов UNIX. Операционная система DR–DOS была переименована сначала в Novell-DOS, затем в Novell DOS. Компания Novell добавила в операционную систему сетевые возможности и улучшила управление памяти, однако новые владельцы не учли, что система устарела для настольных применений. В 1996 г. операционная система была приобретена Caldera Systems Inc. (**www.caldera.com**) и переименована в Caldera DOS, а в 1997 г. распространялась под наименованием OpenDOS (**www.drdos.org**) с открытыми исходными кодами. Напомним, что компания Caldera Inc. образована в 1994 г. и с 1998 г. создает решения на базе Linux. В 2001 Caldera приобретает подразделения по разработке серверного программного обеспечения и профессиональной поддержки компании Santa Cruz Operation Inc. и формирует новую компанию Caldera International Inc. В 2002 г. компания меняет имя на SCO Group и в основном занимается не проектированием и разработкой, а поиском путей извлечения прибыли из принадлежащих ей патентов, унаследованных от Santa Cruz Operation, и шантажом мирового сообщества UNIX и Linux. В 2003 г. компания SCO объявила себя единственным владельцем Linux, ищет якобы содержащуюся в Linux принадлежащую ей «интеллектуальную собственность» и пытается торговать лицензиями на Linux, большая часть которой создана независимыми разработчиками, иногда даже не подозревающими о существовании SCO. Уважение к имени фирмы – это такое качество, которое заслуживается длительным напряженным трудом поколений талантливых разработчиков, а теряется в один момент из-за предательства менеджеров и всеядности адвокатов.

В 1997 г. система OpenDOS перестала быть открытой и поменяла название на DR-DOS, DR-OpenDOS и вновь и окончательно — DR-DOS. С 1998 г. данная операционная система принадлежала отколовшейся в этом же году от Caldera компании Lineo Inc. (подразделение Embedix Inc.), пытавшейся применять данную операционную систему (а также Linux) для разработки встроенных систем и операционных систем реального времени. С 2002 г. система поставляется компанией DeviceLogics Inc. (**www.drdos.com**), хотя на некоторых неофициальных сайтах еще можно встретить все перечисленные

выше версии. По сообщениям прессы, DR-DOS планируется использовать при создании встроенных систем.

Из отечественных разработок необходимо назвать операционную систему PTS-DOS, поставляемую Российской компанией ООО «Физтех-софт» (PhysTechSoft Ltd, **www.phystechsoft.com**), которой и принадлежит торговая марка PTS-DOS. Из особенностей оригинальной PTS-DOS, отметим поддержку файловой системы FAT32 и отсутствие недокументированных функций. Благодаря этому, а также добротности и надежности, свойственным отечественным программным разработкам, оригинальная операционная система PTS-DOS сертифицирована Министерством обороны России для применения в приложениях, требующих особых условий надежности и информационной безопасности.

Официальные отечественные системные разработки в области программных технологий никогда не были ориентированы на удовлетворение нужд массового пользователя и не велись в условиях жесткой конкуренции. Это, в конце концов, привело к тому, что в настоящее время нет отечественной альтернативы современным зарубежным системным разработкам, кроме сборок Linux производства двух упомянутых частных московских компаний.

Операционную систему PTS DOS 2000 Pro поставляет германская компания Paragon Technologie GmbH (**www.paragon.ru**). Под публичной лицензией GNU разрабатывается клон MS-DOS, а именно, FreeDOS (**freedos.org**). По утверждению разработчика, данная операционная система отличается надежностью и применяется для разработки встроенных систем. Следующий разрабатываемый под публичной лицензией GNU клон MS-DOS — операционная система RxDOS (координатор Mike Podanoffsky, **sourceforge.net/projects/rxdos**). Система поддерживает FAT32 и длинные имена файлов Windows 95/98.

С 1998 г. под лицензией GNU разрабатывается операционная система ReactOS (**www.reactos.com**), совместимая с приложениями и драйверами Microsoft Windows версий NT 4.0, 2000, XP. В 2003 г. была продемонстрирована функционирующая альфа-версия системы.

Языки научного программирования

В жанре научного программирования обычно применяются различные диалекты универсальных алгоритмических языков (в алфавитном порядке, до 1991 г. прописными буквами, с 1991 г. по решению ISO — международной организации по стандартизации — прописной пишется только первая буква названия языка; остальные буквы, если они есть, пишутся строчными):

- Алгол (ALGOL, Algol — аббревиатура от ALGOrithmic Language),
- Бейсик (BASIC, Basic — аббревиатура от Beginner's All-purpose Symbolic Instruction Code),

- Паскаль (PASCAL, Pascal),
- Си (C),
- Фортран (FORTRAN, Fortran — аббревиатура от FORmula TRANslating language).

Конечно, никто не запретит разработчику использовать любой другой язык. Он даже может разработать специальный язык самостоятельно, если найдется заказчик, готовый оплачивать его научные изыскания. Алгоритмический язык — средство общения программиста не только с компьютером. Алгоритмические языки служат и для общения программистов между собой, и для передачи опыта одних программистов другим.

Выбор языка и системы программирования для научной программы должен быть обусловлен получением максимальной скорости вычислений. На втором месте — удобство в использовании. Научная программа должна обладать графическим интерфейсом и обеспечивать импорт данных в стандартных форматах.

Можно выделить следующие формальные критерии выбора языка для научного программирования:

1. стандартизация либо повсеместное распространение языка, наличие его реализаций на большинстве или на всех программно-аппаратных платформах, возможность создания кросс-платформенных приложений;
2. поддержка применяемой для разработки научной программы системы программирования, подкрепленная опытом и устойчивостью ее производителя, и в то же время наличие бесплатно распространяемых версий систем программирования на данном языке;
3. компактность и эстетика языковых конструкций и проектных решений;
4. быстрое переложение на другие языки программирования, в том числе перспективные;
5. программа или ее вычислительные модули, при использовании трудоемких алгоритмов, должны быть компилируемыми, а не интерпретируемыми, для обеспечения приемлемой скорости вычислений.

Алгол

Алгоритмический язык Алгол разработан в 1958 г. Первый компилятор Algol 60 написал профессор Э.В. Дейкстра (Нидерланды), который активно участвовал в разработке этого языка программирования.

Алгол успешно применялся для программирования научных задач, в частности, в качестве инструмента решения задач линейной алгебры. Трансляторами с Алгол было оснащено большинство ЭВМ, а Алгол считался единым универсальным языком.

Бейсик

Алгоритмический язык Бейсик разработали сотрудники Дартмутского университета (Нью–Гемпшир, США) Дж. Кемени (John Kemeny) и Т. Курц (Thomas Kurtz) в 1964 г. Язык стандартизован ANSI — американским национальным институтом по стандартизации — в 1978 г. и 1987 г. и ISO в 1984 г. и 1991 г.

Бейсик был вторым языком программирования, портированным на персональный компьютер (первым был PL/I для CP/M, разработанный Г. Килдаллом).

Корпорация Microsoft впервые анонсировала Бейсик для Intel 8086 в 1979 г. Microsoft BASIC для Macintosh был анонсирован в 1984 г. Бейсик, по мнению У. Гейтса, является наиболее простым языком высокого уровня. Основные концепции визуального программирования, положенные в основу Microsoft Visual Basic (VB), разработаны А. Купером. В 1991 г. разработана система разработки приложений VB для Windows. Язык VB берет начало (в ретроспективе) от Microsoft QuickBasic (его ограниченная версия называлась QBasic и поставлялась в комплекте последних версий MS-DOS), GWBasic, Basic и разработанного в 1975 г. П. Алленом и У. Гейтсом интерпретатора Бейсик для первого миникомпьютера MITS Altair 8800, на базе на процессоре Intel 8080. По информации официального сайта Microsoft, У. Гейтс разработал язык программирования Basic для Altair во время своего трехлетнего пребывания на юридическом факультете Гарвардского университета, закончившегося в 1975 г. Разработка велась на эмуляторе Altair на базе PDP-10. Фирма Microsoft предоставила компании MITS лицензию на распространение языка Бейсик, а в 1977 г. ей пришлось оспаривать у MITS свои права на Бейсик в судебном порядке.

В 1979 г. Б. Зейл (Bob Zale) разработал первый интерактивный компилятор BASIC/Z для CP/M и MDOS. В 1987 г. компания Borland International Inc. (**www.borland.com**) выпустила его под именем Turbo Basic, который стал главным конкурирующим продуктом для Basic от Microsoft. В свое время эти программные продукты были равноценны. В 1989 г., в результате соглашения, Borland отказалась от поддержки Basic, а Microsoft — от поддержки Pascal. В 1990 г. Б. Зейл возродил Turbo Basic под торговой маркой PowerBASIC. В настоящее время компания PowerBASIC Inc. (**www.powerbasic.com**) предлагает свою систему программирования, включающую быстрый компилятор, для DOS и Windows. В ближайших планах — поддержка Linux.

Бейсик популярен среди разработчиков различного программного обеспечения в качестве встроенного языка, предназначенного для решения пользовательских задач. Подмножеством VB, а именно Visual Basic for Application (VBA), начиная с 1993 г., оснащается офисный пакет Microsoft Office. Язык

VBA является языком программирования не только для программ, входящих в Microsoft Office. Он лицензирован рядом производителей и встроен в популярные программные продукты.

Языки VB и VBA представляют особый интерес в качестве основных интерфейсных компонентов для программирования на разных алгоритмических языках в пределах решения одной задачи. По рекомендациям источников по программированию для Microsoft Windows, пользовательский интерфейс приложения следует выполнять на VB или VBA для удобства и скорости разработки, прикладную логику реализовать на C или C++ для скорости вычислений.

Если судить о популярности по числу инсталляций приложений, составленных на том или ином алгоритмическом языке, VBA не имеет и, возможно, не будет иметь равных себе в мире. Такую популярность обеспечили VBA компьютерные вирусы, написанные на нем, и неоднократно побывавшие или, по крайней мере, сделавшие такую попытку, на каждом персональном компьютере.

Паскаль и объектный Паскаль

Профессор политехнической школы Цюриха Н. Вирт (Niklaus K. Wirth) разработал алгоритмический язык Паскаль в 1970 г. для обучения программированию. Как отмечал сам автор, «по сегодняшним меркам Паскаль обладал явными недостатками при программировании больших систем». По словам Кернигана (Brian W. Kernighan), «Паскаль может быть превосходным языком для обучения новичков тому, как писать программы... Но в своем стандартном виде (как нынешнем, так и предлагаемом) Паскаль не подходит для написания реальных программ».

Первый компилятор Паскаля ETH Pascal, созданный в Швейцарском федеральном технологическом институте (Eidgenoessische Technische Hochschule) был написан на самом Паскале в 1970 г. для семейства компьютеров CDC-6000 производства Control Data Corporation. В 1978–1981 гг. на основе Паскаля Н. Вирт разработал язык программирования Modula-2 (MODUlar LAnguage). В 1986 г. фирмой Apple совместно с Н. Виртом создан объектный Паскаль (Object Pascal).

Вторым компилятором Паскаля был созданный 1975 г. в Университете Калифорнии (University of California at San Diego — UCSD) на основе инструментария, полученного из Швейцарии, UCSD Pascal, оснащенный интегрированной средой разработки (IDE — Integrated Development Environment). Система программирования UCSD Pascal стала прародительницей системы Turbo Pascal, наиболее успешной реализации для PC, созданной в 1983 г. А. Хейлсбергом для CP/M и MS-DOS. С 1983 г. все права на Turbo Pascal принадлежат основанной в том же году компании Borland International Inc.

Начиная с версии 5, система программирования Turbo Pascal основана на Object Pascal. В 1995 г. А. Хейлсбергом разработана система программирования Delphi. С 2002 г. разработчик Delphi предпочитает не упоминать о Паскале, а сообщает, что Delphi — это язык программирования. Это, в общем, справедливо. Напомним, что в 1998 г. компания приняла непродуманное решение о смене названия на Inprise (от Integration the Enterprise — интеграция предприятий). В 2000 г. компания вернула себе название, созвучное изначальному — Borland Software, но не пользователей, перешедших к тому времени на конкурирующие продукты. В настоящее время компания поставляет также программный продукт Kylix — средство разработки программ для Linux в стиле Delphi.

Из бесплатных программных продуктов для разработки приложений на языке Паскаль отметим систему программирования Bloodshed Dev-Pascal (Bloodshed Software, **www.bloodshed.net**), и отечественную авторскую разработку Странник «Модула-Си-Паскаль», созданную А. Андреевым из города Пермь, **home.perm.ru/~strannik**.

На Паскале были написаны первые версии Microsoft Windows. Компания Microsoft формально не поддерживает язык Паскаль, отказавшись от него, по соглашению с Borland, однако исследовательские лаборатории Microsoft Research сотрудничают с Университетом Квинсленда (Queensland University) в реализации компилятора языка Component Pascal, созданного на базе Pascal, Oberon, Oberon-2, для среды Microsoft .NET и для Java VM.

Object Pascal лежал в основе графической оболочки Apple Macintosh. На Delphi составлены известная программа статистического анализа данных STADIA и система разработки приложений Bloodshed Dev-C++.

C и C++

Алгоритмический язык C представлен в 1972 г. для ОС UNIX, работающей на компьютере PDP-11 фирмы DEC. Разработка велась с 1966 г. по 1970 г. Д. Ритчи (Dennis M. Ritchie), К. Томпсоном (Ken Thompson) и Б. Керниганом в AT&T Bell Laboratories. Язык стандартизован ANSI в 1988 г. Соответствующая стандарту реализация называется ANSI C. Лаконичный и чрезвычайно простой язык применяется для программирования любых системных и прикладных задач на всех типах современных компьютеров.

Язык C++ разработан Б. Страуструпом (Bjarn Stroustrup) в AT&T Bell Laboratories в 1983 г. на основе созданного в 1980 г. языка C with classes.

Первым компилятором C для PC был разработанный в 1982 г. фирмой Lattice Inc. (**www.lattice.com**) компилятор Lattice C. Версии компилятора были выпущены для MS-DOS, OS/2 на PC, а также для MVS, VMS, UNIX, Ami-

gaDOS. По соглашению с Microsoft, компилятор Lattice C одно время повторно упаковывался и продавался как изделие Microsoft. В 1987 г. фирма Lattice была приобретена институтом SAS, а Lattice C стал основной технологией SAS. В 1990 г. развитие компилятора было прекращено. После того, как фирма Lattice вновь стала независимой в 1993 г., к развитию компилятора она больше не возвращалась, сосредоточившись на разработке прикладных программ, в том числе для карманных компьютеров, хотя компилятор Lattice C поставляется до сих пор.

Ведущее место среди систем программирования для MS-DOS и Windows занимают компиляторы Microsoft (Microsoft C, Microsoft Quick C, Microsoft Visual C++), начавшие свою историю с Lattice C. Первым разработанным самой Microsoft был компилятор Microsoft C версии 4.0. В 2001 г. корпорация Microsoft предложила версию C++, разработанную перешедшим в 1996 г. Microsoft А. Хейлсбергом и названную C# (произносится «си-шарп»). В 2003 г. компилятор C# представлен также компанией Borland.

В 1986 г. компанией Wigard Software Systems Inc. был разработан лучший для того времени компилятор Wizard C. Компания Borland International Inc. приобрела компанию Wigard Software Systems Inc. и в 1987 г., на основе Wizard C, разработала свой компилятор Turbo C, с которого началась линия Turbo C. Сегодня продолжения Turbo C представлены в виде Borland C++ и C++ Builder. Несомненным плюсом является наличие доступных версий компиляторов Borland: нескольких, имеющих лишь историческое значение, и современной версии 5.5 без IDE, что не является большим препятствием для полноценной разработки приложений.

В 1987 г. один из создателей Borland Н. Йенсен (Niels Jensen) вместе с группой разработчиков покинул компанию и создал компанию Jensen and Partners International (JPI), положив начало серии компиляторов TopSpeed для процессоров семейства x86: Assembler, Modula-2, Pascal, Ada, C и C++. В 1990 г. технология TopSpeed была лицензирована у JPI компанией Clarion Software Corp. В 1992 году слиянием JPI и Clarion Software Corp. образовалась компания TopSpeed Corp. (**www.topspeed.ru**), а линия компиляторов TopSpeed прекратила существование. Дальнейшая деятельность TopSpeed Corp., позднее неоднократно переименованной, не была связана с компиляторами C.

Компилятор Datalight Optimum-C, позднее переименованный в Zortech C/C++, для операционных систем MS-DOS, Windows, OS/2, Macintosh и UNIX разработан У. Брайтом (Walter Bright). Фирма Zortech Inc. в 1991 г. была приобретена корпорацией Symantec Corp. (**www.symantec.com**). Примерно в то же время, корпорация Symantec приобрела компанию THINK Technologies, выпускавшую компиляторы THINK C и THINK Pascal для Macintosh. Компилятор Zortech C/C++ был переименован в Symantec C++. В настоящее время компилятор Symantec C++ прекратил существование.

Единоличное обладание авторскими правами помогло сохранить продукт, который трансформировался в проект Digital Mars C++ (**www.digitalmars.com**), имеющий, как и Borland C++, бесплатную версию, поставляемую без IDE. В комплекте поставляется высокоэффективный компоновщик OPTLINK, в свое время приобретенный Symantec у SLR Systems. У. Брайт на базе C и C++ создал и продвигает новый язык D.

В 1988 г. компания Microway (**www.microway.com**) разработала серию компиляторов для защищенного режима MS-DOS, UNIX и XENIX. Серия включала компиляторы FORTRAN, Pascal и C. В 1991 г. Microway выпустила серию превосходных компиляторов NDP для Windows, в 1992 г. для OS/2. В 1993–1994 г. линия компиляторов включает NDP Fortran-90 и ряд математических пакетов для разработчика. В 1998 г. Microway выпускает 64-битную версию NDP Fortran для Linux. Дальнейшая деятельность компании, судя по имеющейся информации, с поставкой программных решений, по крайней мере, для широкого рынка средств разработки программного обеспечения, не связана, а столь мощные наработки вряд ли исчезли без всякого следа.

Коммерческие системы разработки приложений на C/C++ для различных ОС на PC поставляют также компании Metrowerks (**www.metrowerks.com**), Salford Software Ltd (**www.salford.co.uk**), SAS Institute Inc. (**www.sas.com**), Silicon Graphics (**www.sgi.com**), Sun, Intel.

Таким образом, в настоящее время на рынке коммерческих компиляторов C/C++ для PC представлены два крупных производителя.

Системы разработки приложений на C/C++ всегда были интересны большому числу независимых создателей средств разработки для различных операционных систем. Бесплатная система программирования Bloodshed Dev-C++, разработанная Bloodshed Software на основе компилятора MinGW (Minimalist GNU For Windows, **www.mingw.org**) — портированного под Win32 компилятора GNU GCC, с любезного разрешения ее автора, помещена на прилагаемый к настоящей книге компакт-диск. Отметим также созданную Р. Шниткером (Rainer Schnitker, **www.mathematik.uni–bielefeld.de/~rainer**) интегрированную среду RSXIDE для компилятора GNU GCC.

В 2003 г. появилась долгожданная версия 1.0 бесплатной системы разработки приложений Open Watcom C/C++, ранее известной как Watcom C/C++. Напомним, что компания Watcom стала частью Powersoft в 1994 г., а затем перешла к Sybase Inc. (**www.sybase.com**). В настоящее время система программирования Open Watcom C/C++ поддерживается сообществом программистов (**www.openwatcom.org**). Несмотря на то, что действия разработчиков координируются компаниями Sybase и SciTech Software Inc. (**www.scitechsoft.com**), Open Watcom в его сегодняшнем виде

представляет собой продукт, немного преждевременно покинувший лаборатории разработчиков.

Бесплатно для некоммерческого применения поставляются системы разработки приложений LCC-Win32 для Windows и QIDE для UNIX/Linux (Q Software Solutions GmbH, **www.q-software-solutions.com**).

Считается, что локализация профессиональных средств разработки является необязательной, т.к. все программисты будто бы обязаны владеть английским языком для получения доступа к технической информации. На самом деле, просто приятно иметь качественно локализованную среду разработки. В предыдущем разделе была упомянута отечественная авторская разработка Странник «Модула-Си-Паскаль», изначально поддерживающая русский и английский язык. Система программирования Bloodshed Dev-C++ версии 5 также поставляется, как это принято в мире открытых систем, с поддержкой десятков языков, причем, даже для предварительной версии, качество локализации можно считать эталоном.

По сформулированным выше критериям, выбор C/C++ в качестве средства научного программирования представляется оправданным. Расчетная программа, написанная для одной ОС и не использующая специфических для нее средств, скорее всего, будет работать и под управлением другой операционной системы.

Благодаря поддержке крупнейших фирм-разработчиков, с одной стороны, и наличию качественных бесплатно распространяемых программных продуктов, с другой стороны, удовлетворяется практически весь диапазон требований и финансовых возможностей потребителей систем программирования на C/C++.

На языке программирования C написаны большинство современных ОС (UNIX, Linux, QNX, Microsoft Windows и т.д.), множество компьютерных игр и серьезных прикладных программ всех назначений.

По программированию на языках C и C++ имеется большое число источников, однако оригинальных работ, которые необходимо иметь в библиотечке программиста, всего две (кроме, естественно, справочной документации для конкретных платформ). Это — выдержавшие несколько переизданий книги создателей данных языков: по языку C — известнейшая и непревзойденная по качеству монография Кернигана и Ритчи, по языку C++ — книга Страуструпа. Имеется также несколько хороших источников по программированию на языках C и C++ для различных операционных систем.

Фортран

Алгоритмический язык Фортран (компилятор FORTRAN I) был разработан в корпорации IBM группой под руководством Дж. Бэкуса (John W. Backus)

в 1954–1957 г.г. Язык стандартизован ANSI в 1966 и 1978 гг. Фортран до сих пор вне конкуренции среди средств научного программирования на больших ЭВМ. Такая ситуация вызвана не выдающимися характеристиками Фортрана, как языка программирования, и особыми качествами трансляторов, а большим количеством наработок и наличием консервативных школ программирования.

Корпорация Microsoft создала свой вариант Microsoft FORTRAN в 1977 г., но в настоящее время продукт не развивается. По информации, доступной на момент написания данной книги, компиляторы Fortran для PC поставляли (в алфавитном порядке): Absoft Corp. (**www.absoft.com**), Intel, Lahey Computer Systems Inc. (**www.lahey.com**), Numerical Algorithms Group Ltd (**www.nag.co.uk**), Open Watcom, Salford, Silicon Graphics, Sun, Sybase.

Ресурсы Интернет

При подготовке обзора использованы материалы с официальных сайтов компаний, в том числе:

- **www.maxframe.com** корпорации MaxFrame,
- **www.osp.ru** издательства «Открытые системы»,
- **www.bytemag.ru** издательства «Byte Россия»,
- **www.computerra.ru** издательства «Компьютерра»,
- проекта **www.knoppix.ru** (**www.linux.su**),
- **www.softline.ru** компании ООО «СофтЛайн Груп»,
- **www.pcmag.ru** ЗАО «СК Пресс», издателя журнала «PC Magazine/Russian Edition».

Рекомендуемая литература

1. Биллиг В.А. VBA в Office 2000. Офисное программирование. —М.: Издательско-торговый дом «Русская редакция», 1999.
2. Каталог программного обеспечения LINUX. — М.: SoftLine, 2003.
3. Каталог программного обеспечения SoftLine-direct. — М.: SoftLine, 2003.
4. Керниган Б., Ритчи Д. Язык программирования Си. — М.: Финансы и статистика, 1992. Available from World Wide Web: < **http://lib.ru/ctotor/kernigan.txt** >.

5. Круглински Д., Уингоу С., Шеферд Дж. Программирование на Microsoft Visual C++ для профессионалов. — СПб: Питер, 2001.

6. Мак–Кракен Д., Дорн У. Численные методы и программирование на ФОРТРАНе. — М.: Мир, 1977.

7. Орвис В.Дж. Excel для ученых, инженеров и студентов. — Киев: Юниор, 1999.

8. Рихтер Дж. Windows для профессионалов: создание эффективных Win32-приложений с учетом специфики 64-разрядной версии Windows. — СПб: Питер; М.: Издательский дом «Русская редакция», 2001.

9. Страуструп Б. Язык программирования Си++. — М.: Радио и связь, 1991. Available from World Wide Web: < **http://lib.ru/cpphb/cpptut.txt** >. См. также «Введение в язык C++». Available from World Wide Web: < **http://www.citforum.ru/programming/cpp/aglav.shtml** >.

10. Уэйт М., Прата С., Мартин Д. Язык Си. Руководство для начинающих. — М.: Мир, 1988.

11. Шилдт Г. Теория и практика C++. — СПб.: BHV–Санкт-Петербург, 1996.

ПРИЛОЖЕНИЕ 5

Работа с прилагаемым компакт-диском

Технические требования

Программное обеспечение на прилагаемом к книге компакт-диске работает с различными конфигурациями оборудования и программного обеспечения. Для просмотра и установки содержимого компакт-диска необходим компьютер с операционной системой Microsoft Windows 95/98 или Windows NT/2000/XP, локализованной или международной, с русскими региональными установками.

Установка программного обеспечения

Поместите компакт-диск в соответствующий CD-дисковод. После автозапуска на экране появится окно браузера, как на рис. П5.1.

Рис. П5.1. Примерный вид окна загрузки компакт-диска в браузере

Если автоматический запуск не произошел, откройте в браузере файл index.htm, находящийся в корневом каталоге компакт-диска. Появившееся на экране меню, в виде столбца ссылок, предоставляет возможности:

- загрузить исходные тексты и проекты прилагаемых к книге программ;
- загрузить дистрибутивы модулей программ AtteStat и ME.com;
- загрузить исходные тексты и проекты применяемых в книге утилит.

На экране содержится также информация о владельце авторских прав.

Дальнейший ход работы зависит от типа используемого браузера. Щелчок левой кнопкой мыши на выбранной ссылке (пункте меню) управляет запуском приложений. Данное программное обеспечение не содержит компьютерных вирусов. Установка содержимого прилагаемого к книге компакт-диска не представляет опасности для компьютерной системы.

Процедура установки дистрибутивов модулей AtteStat и ME.com подробно описана далее. Дистрибутивы на компакт-диске содержат программы установки, которые помогут установить и использовать модули программного обеспечения пользователям, которые хотели бы освоить и использовать продвинутые технологии анализа данных и математического моделирования в своей работе. На рисунках и в комментариях показаны этапы установки программного обеспечения на примере одного из модулей.

Установка каждого модуля начинается с запуска исполняемого файла дистрибутива (рис. П5.1). Если будут заданы вопросы, разрешить ли исполнение программы, следует ответить утвердительно. Начальное окно программы установки[1] для одного из модулей показано на рис. П5.2.

Рис. П5.2. Начальное окно программы установки

Отметим, что отменить установку можно почти на любом шаге работы программы, нажав кнопку **Нет** или **Отменить**. В продолжение установки на экране появится приветственное окно программы установки, как на рис. П5.3.

---

[1] *Первым окном может быть приглашение выбора языка установки, в зависимости от настройки генератора дистрибутива.*

Рис. П5.3. Приветственное окно программы установки

При нажатии кнопки **Далее** установка программы продолжится. Существенную информацию содержит окно лицензионного соглашения, показанное на рис. П5.4. Установка может быть продолжена только, если пользователь выразит свое согласие с условиями Лицензионного соглашения. Полный текст Лицензионного соглашения дан в следующем разделе данного приложения.

Рис. П5.4. Примерный вид окна Лицензионного соглашения (вид может отличаться в зависимости от версии генератора дистрибутива, рассмотренного в *главе 12*)

Далее появится информационное окно с дополнительной технической информацией об инсталлируемом программном продукте (рис. П5.5).

Рис. П5.5. Информационное окно

При нажатии кнопки **Далее** пользователю будет предложено выбрать каталог установки (рис.П5.6). Рекомендуется выбрать каталог по умолчанию и нажать кнопку **Далее**.

Рис. П5.6. Окно выбора каталога для установки

Последнее окно перед установкой программы — это окно подтверждения выбора установочных параметров (рис. П5.7). Здесь еще есть возможность вернуться назад и изменить какие-либо параметры установки.

Рис. П5.7. Окно подтверждения выбора установочных параметров

О ходе установки будет сообщать соответствующее окно, после чего появится окно с файлом Readme.txt (рис. П5.8), содержащим важную оперативную информацию. В данный файл разработчик программного обеспечения обычно помещает информацию, не вошедшую в печатное «Руководство пользователя», либо какие-либо изменения и дополнения.

Рис. П5.8. Окно с файлом Readme.txt, содержащим важную оперативную информацию

Последний этап на утомительном пути установки – конечное окно программы установки (рис.П5.9). Некоторые типы программ требуют после установки перезагрузки компьютера, однако в нашем случае это не требуется.

Удаление программного обеспечения может быть выполнено стандартными средствами Windows либо входящей в комплект модуля специальной утилитой. Утилита удаления запускается (для рассматриваемого модуля) выбором **Пуск ▸ Программы ▸ AtteStat ▸ Cla for Excel ▸ Удаление Cluster Analysis for Excel**. Для других модулей порядок удаления модуля такой же.

Рис. П5.9. Конечное окно программы установки

Запуск программ и работа с ними

Запуск программ, содержащихся на компакт-диске, и работа с ними подробно описаны в каждой из глав книги, в которых данное программное обеспечение рассматривается.

Техническая поддержка

На прилагаемом к книге компакт-диске содержатся учебные версии программного обеспечения. На учебные версии программ не распространяется какая-либо поддержка, однако автор постарается ответить на возникшие вопросы относительно ошибок, обнаруженных в программном обеспечении. По поводу возможных проблем с качеством носителей следует обращаться в торговую организацию, в которой приобретена книга.

Правовые вопросы использования программного обеспечения

Прилагаемые к данной книге учебные версии программ AtteStat и ME.com распространяются вместе с книгой только для учебных целей. Для использования с целью получения коммерческой выгоды должны быть приобретены соответствующие коммерческие версии программ.

Программа анализа данных AtteStat зарегистрирована за № 2002611109 Российским агентством по патентам и товарным знакам 28 июня 2002 г. и защищена законодательством Российской Федерации. На рис. П5.10 показано свидетельство об официальной регистрации программы AtteStat.

РОССИЙСКАЯ ФЕДЕРАЦИЯ

РОССИЙСКОЕ АГЕНТСТВО ПО ПАТЕНТАМ
И ТОВАРНЫМ ЗНАКАМ
(РОСПАТЕНТ)

СВИДЕТЕЛЬСТВО

Об официальной регистрации программы для ЭВМ

№ 2002611109

На основании Закона Российской Федерации "О правовой охране программ для электронных вычислительных машин и баз данных", введенного в действие 20 октября 1992 года, Российским агентством по патентам и товарным знакам выдано настоящее свидетельство об официальной регистрации программы для ЭВМ

Программа анализа данных AtteStat

Правообладатель(ли):

Гайдышев Игорь Павлович (RU)

Автор(ы):

Гайдышев Игорь Павлович (RU)

Страна: Российская Федерация

по заявке № 2002610891, дата поступления: 23 мая 2002 г.

Зарегистрировано в
Реестре программ для ЭВМ

г. Москва, ***28 июня 2002 г.***

Генеральный директор

А.Д. Корчагин

Рис. П5.10. Свидетельство об официальной регистрации программы анализа данных AtteStat

Программа **Математические и инженерные компоненты** ME.com зарегистрирована за № 2003611195 Российским агентством по патентам и товарным знакам 22 мая 2003 г. и защищена законодательством Российской Федерации. На рис. П5.11 показано свидетельство об официальной регистрации программы ME.com.

РОССИЙСКАЯ ФЕДЕРАЦИЯ

РОССИЙСКОЕ АГЕНТСТВО ПО ПАТЕНТАМ
И ТОВАРНЫМ ЗНАКАМ
(РОСПАТЕНТ)

СВИДЕТЕЛЬСТВО

Об официальной регистрации программы для ЭВМ

№ 2003611195

На основании Закона Российской Федерации "О правовой охране программ для электронных вычислительных машин и баз данных", введенного в действие 20 октября 1992 года, Российским агентством по патентам и товарным знакам выдано настоящее свидетельство об официальной регистрации программы для ЭВМ

Математические и инженерные компоненты ME.com

Правообладатель(ли):

Гайдышев Игорь Павлович (RU)

Автор(ы):

Гайдышев Игорь Павлович (RU)

Страна: Российская Федерация

по заявке № 2003610676, дата поступления: 31 марта 2003 г.

Зарегистрировано в
Реестре программ для ЭВМ

г. Москва, 22 мая 2003 г.

Генеральный директор

Л.Д. Корчагин

Рис. П5.11. Свидетельство об официальной регистрации программы Математические и инженерные компоненты ME.com

Вашему вниманию предлагается текст составленного автором лицензионного соглашения, которое действует на момент публикации данного издания и может быть без предупреждения изменено.

Правовые вопросы в области программного обеспечения не являются темой данной книги. Однако автор хотел бы отметить, что он не может полностью принять ни стандартные лицензии какого-либо производителя программ,

ни различного рода свободные лицензии. Первые, по нашему мнению, ограничивают права пользователей, вторые — права разработчиков. У пользователя есть неотъемлемое право получить качественное программное обеспечение за свои деньги, и он может всегда защитить свое право в суде. Производитель же вправе ставить такие условия распространения своих программ, какие сочтет нужными, и пересматривать эти условия тогда, когда захочет. Единственным ограничением служит только требование отсутствия дискриминации по признакам, определенным правами человека.

Лицензионное соглашение

Данное программное обеспечение может распространяться в трех версиях: полной, учебной, демонстрационной. Полная версия является коммерческой. Учебная версия приобретается только в комплекте с книгой и отдельно от книги не распространяется. Демонстрационная версия распространяется бесплатно. Приобретая или получая (на дискете, компакт-диске, при загрузке с сайта) любую версию данного программного обеспечения, следует соблюдать следующие условия.

Общие положения

1. Данное программное обеспечение защищено Законом РФ от 23 сентября 1992 г. № 3523–1 «О правовой охране программ для электронных вычислительных машин и баз данных».
2. Вы вправе устанавливать и эксплуатировать приобретенный экземпляр полной или учебной версии программного обеспечения только на одном компьютере. Если модули программного обеспечения приобретаются отдельно, Вы вправе устанавливать и эксплуатировать каждый модуль индивидуально, как если бы это был отдельный экземпляр программного обеспечения. Если программное обеспечение приобретено в комплекте пакета из нескольких модулей, эти модули лицензируются как единое целое. Демонстрационную версию данного программного обеспечения Вы вправе свободно устанавливать на любое число компьютеров.
3. В случае приобретения полной версии данного программного обеспечения Вы вправе использовать ее для любых коммерческих и/или некоммерческих целей, не противоречащих действующему законодательству и указанным ограничениям применения. В случае приобретения учебной версии данного программного обеспечения Вы вправе использовать ее только для учебных целей (в том числе в учебном процессе, построенном на коммерческой основе). В случае получения демонстрационной версии данного программного обеспечения Вы вправе использовать ее только для целей ознакомления с данным программным обеспечением.

4. Вы можете использовать данное программное обеспечение или его фрагменты как составную часть своих коммерческих программных проектов только при условии письменного разрешения автора в согласованных объемах и на согласованных условиях.
5. Вы вправе делать резервные копии носителей полной или учебной версии данного программного обеспечения только для архивных целей для замены утраченного или пришедшего в негодность правомерно приобретенного экземпляра. Вы не вправе изготавливать в любой форме копию документации, полученной в комплекте с приобретенным программным обеспечением. На территориях других государств Вы обязаны поступать в соответствии с аналогичным законодательством этих государств. Если данное положение противоречит национальному законодательству, данное положение изымается из лицензионного соглашения.
6. В случае продажи или иной законной передачи данного программного обеспечения другому владельцу Вы обязаны передать ему все материалы (при их наличии в комплекте поставки), включая оригинальную упаковку, лицензионное соглашение и Руководство пользователя, приобретенные или полученные в комплекте с данным программным обеспечением. В случае приобретения учебной версии данного программного обеспечения Вы вправе передавать ее другому владельцу только в комплекте с книгой, в комплект которой входит данное программное обеспечение. Демонстрационную версию данного программного обеспечения Вы вправе свободно распространять на некоммерческой основе в любом числе копий.

Гарантии и ответственность

1. Гарантия на качество носителей программы устанавливается организацией-распространителем, в которой Вы приобрели экземпляр данного программного обеспечения. Данная гарантия не может быть передана любому последующему пользователю. Возможность возврата программного обеспечения организации-распространителю оговаривается при приобретении программного обеспечения.
2. Автор гарантирует, что данное программное обеспечение содержится на носителях в объеме, соответствующем сопроводительной документации.
3. Автор гарантирует, что выполняемые данным программным обеспечением функции соответствуют функциям, указанным в сопроводительной документации.
4. Автор не гарантирует, что данное программное обеспечение полностью свободно от программных и/или алгоритмических ошибок, и что оно обязательно удовлетворит Ваши требования. Однако если проблемы будут обнаружены в процессе эксплуатации программного обеспечения, автор предпримет все разумные меры, чтобы исправить проблемы.

5. Ни при каких условиях автор не несет ответственности за возможный прямой и/или косвенный ущерб, включая упущенную выгоду, связанный с применением или неправильным применением данного программного обеспечения. Вы применяете данное программное обеспечение на свой страх и риск.

Ограничения применения

1. Данное программное обеспечение не предполагалось сертифицировать для применения в медицинских учреждениях Российской Федерации с целями, связанными с диагностикой и выбором тактики лечения и самим лечением пациентов. Согласно Указу Президента Российской Федерации №161 от 18 февраля 1995 г. и Приказу Министра здравоохранения №85 от 4 апреля 1995 г. «О сертификации программных средств и баз данных в системе Минздравмедпрома России» оно не может быть применено для указанных целей. Аналогичные правила действуют и на территории других государств.

© Игорь Гайдышев, 2003.

Все права зарезервированы.

Содержимое прилагаемого компакт-диска

Папки	Описание	Главы
\ \	Файлы для автоматического запуска диска, индексный файл	12
\Code\Pictures	Изображения для оформления компакт-диска	
\Code\Program	Дистрибутивы программного обеспечения анализа данных и математического моделирования	2-12
\Code\Source	Полные исходные тексты проектов программного обеспечения анализа данных и математического моделирования	2-12
\Code\Docs	Лицензионные документы	3-12

Заключение

«Найти ошибку легче, чем истину».
А. Боровик

В заключение нам хотелось бы остановиться на информационной безопасности и ошибках программирования.

Об информационной безопасности

Необходимыми условиями информационно безопасной программы являются отсутствие недокументированных функций и доступные для изучения исходные тексты. Эти требования предъявляются, в первую очередь, к операционным системам, но справедливы и в отношении прикладных программ.

Некоторые требования к информационной безопасности прикладной программы:

1. Использование только стандартных документированных функций операционной системы.
2. Предоставление всех исходных текстов и проектов прикладных программ пользователям по их требованию на тех или иных условиях.
3. Применение программных средств и библиотек сторонних разработчиков только с открытыми исходными текстами.
4. При установке программа не должна модифицировать системные файлы и другие ресурсы операционной системы. Запись в загрузочный сектор, системный каталог или реестр отсутствует или максимально ограничена. Полное удаление всех компонентов программы при деинсталляции.
5. Отсутствие в прикладной программе недокументированных функций.

При реальной разработке эти требования удовлетворить трудно, поэтому неудивительно, что безопасных программ, как и программ, свободных от ошибок, не существует.

Недокументированными (недекларированными) функциями операционной системы называют такие функции API, о которых не имеется полной и точ-

ной технической информации. Существование их обусловлено естественным желанием разработчика иметь некоторые служебные функции для отладки системы. Кроме того, сокрытие технической информации дает производителю возможность разработать более быстрые и надежные приложения, чем конкуренты.

Наличие недокументированных функций опасно не только потому, что компания-разработчик или иное учреждение теоретически могут воспользоваться ими, например, для несанкционированного сбора персональной информации, снятии информации с компьютера пользователя, уничтожения файлов пользователя, нарушения работы информационной системы. Этими функциями может воспользоваться злоумышленник или террорист для деструктивных действий, особенно опасных для компьютера в глобальной сети.

Вспомним, что недокументированным возможностям операционных систем в свое время посвящались монографии, и знание таких возможностей считалось одним из признаков профессионализма программиста. Однако в современных условиях достоинства обернулись недостатками. Понимая эту ситуацию, в своей программистской практике мы никогда не применяли недокументированных функций.

Об ошибках программирования

Основным алгоритмическим языком для разработки научных и инженерных прикладных программ был и пока еще остается Фортран. Однако все рекомендации по программированию численных алгоритмов, введенные авторами для разработчиков на Фортране, справедливы и для разработчиков на других языках программирования.

Наиболее тяжелыми являются ошибки при работе с памятью. Мы не будем рассматривать ошибки типа нехватки памяти или освобождения незанятой памяти, которым уделяли внимание многие авторы. Решим только один вопрос: «Всегда ли нужно освобождать память, выделенную функцией на языке C или C++, в самой функции»? Некоторые источники не требовали делать этого. При завершении приложения выделенная динамически память, по идее, должна освобождаться автоматически. Если повторно не вызывать функцию, выделившую динамическую память, приложение должно завершиться без ошибок.

Однако при работе с DLL имеются свои сложности. Ответ на поставленный вопрос здесь отличается от общепринятого: «Если мы хотим создать универсальный модуль DLL и не устранять внезапно возникающие ошибки в своей дальнейшей работе, то — всегда». Если в модуле DLL есть функция, выделяющая память под динамические массивы, в нем обязательно должна

быть функция, которая освобождает всю выделенную память. Это относится и к стандартным С-функциям `malloc` и `free`, и к эквивалентным С++-операторам `new` и `delete`.

Рекомендуемая литература

1. Боровин Г.К., Комаров М.М., Ярошевский В.С. Ошибки-ловушки при программировании на фортране. — М.: Наука, 1987.
2. Гайдышев И. Анализ и обработка данных: специальный справочник. — СПб: Питер, 2001.
3. Мак-Кракен Д., Дорн У. Численные методы и программирование на ФОРТРАНе. — М.: Мир, 1977.