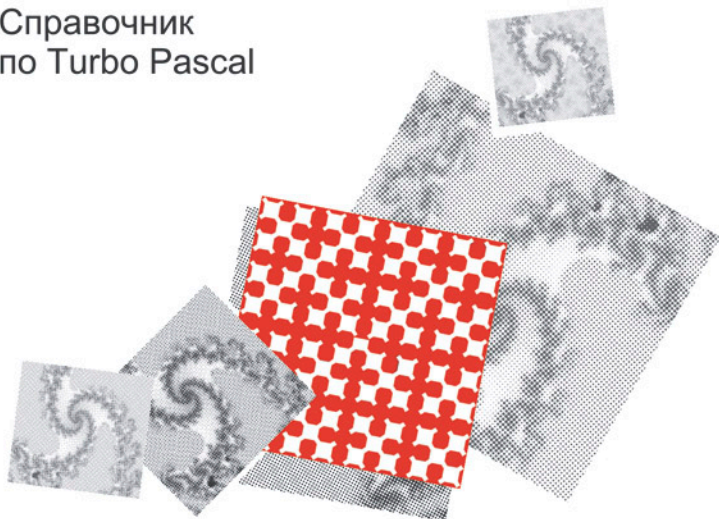


Turbo Pascal

в задачах и примерах

Более
200 задач

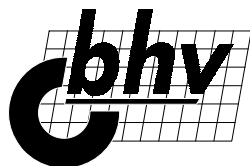
- Примеры и решения
в виде текстов программ
с подробными комментариями
- Справочник
по Turbo Pascal



Никита Культин

Turbo Pascal

в задачах и примерах



Санкт-Петербург

Дюссельдорф ♦ Киев ♦ Москва ♦ Санкт-Петербург

Книга представляет собой задачник по программированию на языке Turbo Pascal. Приведены как типовые задачи, т. е. ввод-вывод, использование циклов, работа с массивами, поиск и сортировка, так и задачи, которые, как правило, остаются за рамками традиционных курсов программирования: работа со строками и файлами, вывод на принтер, деловая графика, рекурсия. Для многих задач приведены решения — хорошо документированные тексты программ. Помимо задач и примеров книга содержит справочник по языку программирования Turbo Pascal (описание наиболее часто используемых процедур и функций) и может использоваться в качестве задачника для учеников 8—11 классов, изучающих программирование в школе.

Для начинающих программистов

Группа подготовки издания:

Главный редактор	<i>Екатерина Кондукова</i>
Зав. редакцией	<i>Наталья Таркова</i>
Редактор	<i>Татьяна Кручинина</i>
Компьютерная верстка	<i>Ольги Сергиенко</i>
Корректор	<i>Наталья Першакова</i>
Дизайн обложки	<i>Ангелины Лужиной</i>
Зав. производством	<i>Николай Тверских</i>

Культин Н. Б.

Turbo Pascal в задачах и примерах. — СПб.:
БХВ — Санкт-Петербург, 2000. — 256 с.: ил.

ISBN 5-8206-0061-4

© Н. Б. Культин, 2000

© Оформление, издательство "БХВ — Санкт-Петербург", 2000

Лицензия ЛР № 065953 от 15.06.98. Подписано в печать 06.05.2000.

Формат 60×90¹/₁₆. Печать офсетная. Усл. печ. л. 16.

Тираж 5000 экз. Заказ

"БХВ — Санкт-Петербург", 198005, Санкт-Петербург, Измайловский пр., 29.

Гигиеническое заключение на продукцию, товар, № 77.99.1.953.П.950.3.99
от 01.03.1999 г. выдано Департаментом ГСЭН Минздрава России.

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

СОДЕРЖАНИЕ

Предисловие	7
Как работать с книгой	8
Оформление решений	8
Часть I. Задачи	9
Объявление переменных	9
Инструкция присваивания	10
Вывод	13
Ввод	15
Программы с линейной структурой	16
Выбор	24
<i>If</i>	24
<i>Case</i>	30
Циклы	31
<i>For</i>	31
<i>Repeat</i>	41
<i>While</i>	43
Символы и строки	44
Массивы	47
Функции	53
Процедуры	55
Графика	56
Файлы	64
Рекурсия	69
Часть II. Решения	73
Часть III. Справочник	209
Структура программы	209
Основные типы данных	209
Целые числа	210

Действительные числа.....	210
Строки	210
Массивы	211
Записи	211
Выбор	212
Инструкция <i>IF</i>	212
Инструкция <i>CASE</i>	212
Циклы.....	213
Инструкция <i>FOR</i>	213
Инструкция <i>REPEAT</i>	214
Инструкция <i>WHILE</i>	215
Безусловный переход.....	215
Инструкция <i>Go To</i>	215
Объявление функции.....	215
Объявление процедуры.....	216
Встроенные функции и процедуры.....	216
Зарезервированные слова и директивы	218
Процедуры и функции	219
Математические функции	219
<i>Abs</i>	219
<i>Arc tan</i>	219
<i>Cos</i>	219
<i>Exp</i>	220
<i>Ln</i>	220
<i>Sin</i>	220
<i>Sqr</i>	220
<i>Sqrt</i>	220
<i>Random</i>	221
<i>Randomize</i>	221
Функции и процедуры преобразования.....	221
<i>Int</i>	221
<i>Round</i>	222
<i>Str</i>	222
<i>Trunc</i>	222
<i>Val</i>	222
Функции и процедуры для работы со строками и символами	223
<i>Chr</i>	223
<i>Concat</i>	223
<i>Copy</i>	223
<i>Delete</i>	223
<i>Length</i>	224
<i>Pos</i>	224
Процедуры и функции графического режима	224
<i>Arc</i>	224
<i>Bar</i>	225

<i>Bar3D</i>	225
<i>Circle</i>	225
<i>DetectGraph</i>	226
<i>Ellipse</i>	226
<i>GetX, GetY</i>	226
<i>GraphResult</i>	226
<i>InitGraph</i>	227
<i>Line</i>	227
<i>Line To</i>	227
<i>Move To</i>	228
<i>Move Rel</i>	228
<i>OutText</i>	228
<i>OutTextXY</i>	228
<i>PieSlice</i>	229
<i>PutPixel</i>	229
<i>Rectangle</i>	229
<i>Sector</i>	230
<i>SetColor</i>	230
<i>SetFillStyle</i>	231
<i>SetLineStyle</i>	232
<i>SetTexStyle</i>	233
Процедуры и функции для работы с файлами	234
<i>Append</i>	234
<i>Assign</i>	234
<i>Close</i>	234
<i>EOF</i>	234
<i>Erase</i>	235
<i>IOResult</i>	235
<i>Reset</i>	235
<i>Rewrite</i>	236
Прочие функции	236
<i>ClrEol</i>	236
<i>ClrScr</i>	236
<i>Delay</i>	236
<i>Dispose</i>	237
<i>Eoln</i>	237
<i>GotoXY</i>	237
<i>Halt</i>	237
<i>New</i>	238
<i>ParamCount</i>	238
<i>ParamStr</i>	238
<i>ReadKey</i>	238
<i>TextBackGround</i>	239
<i>TextColor</i>	240

<i>Where X</i>	240
<i>Where Y</i>	241
<i>Window</i>	241

Приложение 243

Вывод иллюстраций	243
Таблица кодировки символов	247
Представление информации в компьютере	248
Десятичные, двоичные и шестнадцатеричные числа	248
Память компьютера	250

Список дополнительной литературы 253

Предметный указатель 254

ПРЕДИСЛОВИЕ

Чтобы научиться программировать, не достаточно прочитать книгу, посвященную языку программирования, надо писать программы, решать конкретные задачи. Но где их можно найти? В учебниках, как правило, приводятся типовые, стандартные задачи, в основе которых лежит расчет по формулам. Такие задачи, несомненно, полезны, но они не всегда интересны.

В данной книге начинающему программисту предлагаются задачи, которые, с одной стороны ему по плечу, с другой — полезны и занимательны.

Книга состоит из трех частей и приложения.

В первой части собраны задачи. Они сгруппированы по темам и относятся практически ко всем разделам программирования: от объявления переменных и программ с линейной структурой до работы с графикой и файлами.

Во второй части приведены решения, представленные в виде хорошо документированных текстов программ, изучение которых поможет начинающему программисту даже в том случае, если задача решена им самостоятельно.

Третья часть представляет собой справочник по языку программирования Turbo Pascal, содержащий описание большинства процедур и функций.



Если вы имеете компьютер с процессором типа Pentium (Celeron), то используйте Turbo Pascal версии 7.1. Программы, созданные при помощи Turbo Pascal 7.0 и использующие библиотеку `Ctr`, на компьютерах с процессором типа Pentium (Celeron) не работают, при их запуске возникает ошибка "деление на ноль".

Еще раз хочется повторить, что научиться программировать можно только программируя, т. е. решая конкретные задачи.

Поэтому, чтобы получить максимальную пользу от книги, вы должны работать с ней активно. Решайте задачи. Изучайте приведенные решения. Вводите их в свой компьютер. Не бойтесь экспериментировать — вносите изменения в программы. Чем большую работу вы проделаете самостоятельно, тем большему научитесь.

Как работать с книгой

Группы задач следуют в книге в том порядке, в котором традиционно изучаются соответствующие разделы в курсе программирования. Поэтому перед тем как приступить к решению задач нужно изучить соответствующую тему — прочитать раздел учебника. Если сразу решить задачу не получается, то можно посмотреть ее решение и затем еще раз попытаться решить задачу самостоятельно. Писать программу лучше сначала на бумаге, а уже затем вводить программу в компьютер.

Задача считается решенной, если программа работает так, как сказано в условии задачи.

Оформление решений

Важно, чтобы решенная задача была правильно оформлена, что предполагает использование:

- осмысленных имен переменных, констант, функций и процедур;
- отступов при записи инструкций;
- комментариев.

Правильно оформленную программу легче отлаживать, кроме того, она производит хорошее впечатление.

Приведенные в книге примеры можно рассматривать как образцы оформления решений задач.



Часть I. Задачи

Объявление переменных

Приступая к решению задач этого раздела, следует помнить, что:

- каждая переменная программы должна быть объявлена;
- объявления переменных помещают в раздел, который начинается словом `var`;
- инструкция объявления переменной выглядит так:
`ИмяПеременной:Тип;`
- в имени переменной можно использовать буквы латинского алфавита и цифры (первым символом должна быть буква);
- основными числовыми типами языка Turbo Pascal являются: `integer` (целый), `real` (дробный);
- после инструкции объявления переменной рекомендуется указывать назначение переменной.

1. Объявите переменные, необходимые для вычисления значения функции $y = x^2$.

2. Объявите переменные, необходимые для пересчета веса из фунтов в килограммы.

3. Объявите переменные, необходимые для вычисления объема параллелепипеда.

4. Объявите переменные, необходимые для вычисления площади круга.

5. Объявите переменные, необходимые для вычисления площади кольца.

6. Объявите переменные, необходимые для вычисления объема и площади поверхности цилиндра.
7. Объявите переменные, необходимые для вычисления стоимости покупки, состоящей из нескольких тетрадей, карандашей и линейки.
8. Объявите переменные, необходимые для вычисления стоимости покупки, состоящей из нескольких тетрадей и такого же количества обложек.

Инструкция присваивания

Приступая к решению задач этого раздела, следует вспомнить, что:

- инструкция присваивания используется для изменения значений переменных, в том числе и для вычислений по формулам;
- тип выражения, находящегося в правой части инструкции присваивания, должен соответствовать типу переменной, имя которой стоит слева от символа инструкции присваивания (при нарушении соответствия типа переменной и выражения компилятор выводит сообщение об ошибке **Type miss match** — несоответствие типов).

9. Запишите инструкцию, которая присваивает переменной `x` значение $-1,5$.
10. Запишите инструкцию, которая присваивает переменной `summa` нулевое значение.
11. Запишите инструкцию, которая увеличивает на единицу значение переменной `n`.
12. Запишите инструкцию, которая уменьшает на два значение переменной `counter`.
13. Запишите инструкцию вычисления среднего арифметического переменных `x1` и `x2`.
14. Запишите в виде инструкции присваивания формулу вычисления значения функции $y = -2,7x^3 + 0,23x^2 - 1,4$.

15. Запишите в виде инструкции присваивания формулу вычисления значения функции $y = \frac{1}{x^2}$.

16. Запишите в виде инструкции присваивания формулу пересчета веса из фунтов в килограммы (один фунт равен 409,5 г).

17. Запишите в виде инструкции присваивания формулу пересчета расстояния из километров в версты (одна верста равна 1066,8 м).

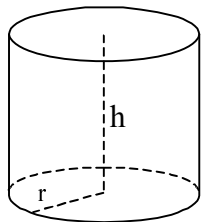
18. Запишите в виде инструкции присваивания формулу вычисления площади прямоугольника.

19. Запишите в виде инструкции присваивания формулу вычисления площади треугольника: $s = \frac{1}{2} \cdot a \cdot h$, где a — длина основания треугольника, h — его высота.

20. Запишите в виде инструкции присваивания формулу вычисления площади трапеции: $s = \frac{a+b}{2} \cdot h$, где a и b — длины оснований, h — высота трапеции.

21. Запишите в виде инструкции присваивания формулу вычисления площади круга: $s = \pi \cdot r^2$.

22. Запишите в виде инструкции присваивания формулы вычисления площади поверхности и объема цилиндра.

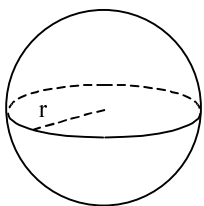


$$s = 2 \cdot \pi \cdot r(h + r)$$

$$v = \pi \cdot r^2 \cdot h$$

23. Запишите в виде инструкции присваивания формулу вычисления объема параллелепипеда.

24. Запишите в виде инструкции присваивания формулы вычисления объема и площади поверхности шара.

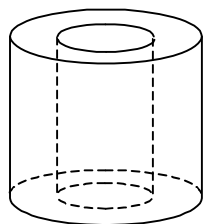


$$v = \frac{3}{4} \cdot \pi \cdot r^3$$

$$s = 4 \cdot \pi \cdot r^2$$

25. Запишите в виде инструкции присваивания формулу вычисления объема цилиндра.

26. Запишите в виде инструкции присваивания формулу вычисления объема полого цилиндра.



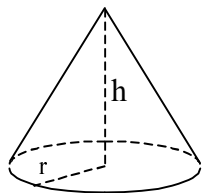
$$v = \pi \cdot h \cdot (r_1^2 - r_2^2)$$

r_1 — радиус цилиндра

r_2 — радиус отверстия

h — высота цилиндра

27. Запишите в виде инструкции присваивания формулу вычисления объема конуса.



$$s = \frac{1}{3} \cdot \pi \cdot r^2 \cdot h$$

28. Запишите в виде инструкции присваивания формулу вычисления объема цилиндра.

29. Запишите в виде инструкции присваивания формулу вычисления тока по известным значениям напряжения и сопротивления электрической цепи.

30. Запишите в виде инструкции присваивания формулу вычисления сопротивления электрической цепи по известным значениям напряжения и силы тока.

31. Запишите в виде инструкции присваивания формулу вычисления сопротивления электрической цепи, состоящей из трех последовательно соединенных резисторов.

32. Запишите в виде инструкции присваивания формулу вычисления сопротивления электрической цепи, состоящей из двух параллельно соединенных резисторов: $r = \frac{r_1 \cdot r_2}{r_1 + r_2}$.

33. Запишите в виде инструкции присваивания формулу пересчета сопротивления электрической цепи из омов в килоомы.

34. Запишите в виде инструкции присваивания формулу вычисления стоимости покупки нескольких тетрадей, карандашей и линейки.

35. Запишите в виде инструкции присваивания формулу вычисления стоимости покупки некоторого количества (по весу) помидоров, огурцов и яблок.

Вывод

Приступая к решению задач этого раздела, следует вспомнить, что:

- инструкции `write` и `writeln` предназначены для вывода на экран монитора сообщений и значений переменных;
- одна инструкция `write` (`writeln`) может вывести на экран значения нескольких переменных и (или) несколько сообщений;
- инструкция `writeln` без параметров переводит курсор в начало следующей строки экрана.

36. Написать программу, которая выводит на экран имя и фамилию.

37. Написать программу, которая выводит на экран сообщение Turbo Pascal 7.0.

38. Написать программу, которая выводит на экран четверостишие:

Унылая пора! Очей очарованье!
Приятна мне твоя прощальная краса —
Люблю я пышное природы увяданье,
В багрец и золото одетые леса.

А. С. Пушкин

39. Написать инструкцию вывода значения переменной *a* (тип *real*) с тремя цифрами в дробной части.

40. Определите, что делают инструкции:

```
write(a);  
write('a');
```

41. Написать инструкцию, которая выводит значения дробных переменных *a*, *b* и *c* в одной строке. Каждое число должно быть выведено с двумя знаками после десятичной точки:

```
writeln(a:6:2,' ',b:6:2,' ',c:6:2);
```

42. Написать инструкции вывода значений переменных *a*, *b* и *c*. Значение каждой переменной должно быть выведено на отдельной строке.

43. Написать инструкции вывода значений дробных переменных *x1* и *x2*. На экране перед значением переменной должен быть выведен поясняющий текст, представляющий собой имя переменной, за которым следует знак "равно".

Факультатив

Стандартная библиотека *Crt* содержит процедуры, используя которые, можно задать цвет фона и цвет символов, выводимых инструкциями *write* и *writeln*.

Приступая к решению задач этого раздела, следует вспомнить, что:

- для того, чтобы программе были доступны процедуры и функции библиотеки `Crt`, первой инструкцией программы должна быть инструкция `uses Crt`;
- цвет символов устанавливает процедура `TextBackGround`;
- цвет фона устанавливает процедура `TextColor`;
- при использовании процедур `TextBackGround` и `TextColor` цвет можно задать с помощью именованной или целой константы;
- процедура `ClrScr` очищает экран;
- в текст программы нужно включить директиву `uses Crt`, чтобы можно было использовать процедуры `ClrScr`, `TextBackGround` и `TextColor`.

44. Написать программу, которая выводит на синем фоне серыми буквами четверостишие:

Буря мглою небо кроет,
Вихри снежные крутя.
То как зверь она завоет,
То заплачет, как дитя.

А. С. Пушкин

45. Написать программу, которая выводит на экран фразу: **Каждый охотник желает знать, где сидят фазаны**, позволяющую запомнить порядок следования цветов радуги (первая буква слова кодирует цвет: каждый — красный, охотник — оранжевый, желает — желтый, знать — зеленый, где — голубой, сидят — синий, фазаны — фиолетовый). Каждое слово фразы должно быть выведено наиболее подходящим цветом.

Ввод

Приступая к решению задач этого раздела, следует вспомнить, что:

- для ввода с клавиатуры во время работы программы исходных данных (значений переменных) предназначена инструкция `readln`;
- используя одну инструкцию `readln`, можно ввести значения нескольких переменных;

- тип данных, вводимых во время работы программы, должен соответствовать типу переменной, указанной в инструкции `readln`;
- в случае несоответствия типа введенных данных типу переменной, значение которой вводится с клавиатуры, программа завершает работу и на экран выводится сообщение `Error 106: Invalid numeric format` (если программа запущена из среды разработки, т. е. из Turbo Pascal) или `Run time error 106` (если программа запущена из операционной системы).

46. Написать инструкцию, обеспечивающую во время работы программы ввод значения переменной `radius`.

47. Написать инструкции, которые обеспечивают ввод значений переменных `u` и `r`. Предполагается, что во время работы программы пользователь будет после набора каждого числа нажимать клавишу `<Enter>`.

48. Написать инструкцию, которая обеспечивает ввод значений переменных `u` и `r`. Предполагается, что во время работы программы пользователь будет набирать числа в одной строке.

49. Объявите необходимые переменные и напишите фрагмент программы вычисления объема цилиндра, обеспечивающий ввод исходных данных.

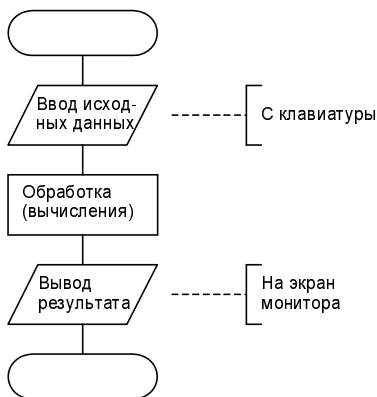
50. Объявите необходимые переменные и напишите инструкции ввода исходных данных для программы вычисления стоимости покупки нескольких тетрадей и карандашей. Предполагается, что во время работы программы пользователь будет вводить данные о каждой составляющей покупки в отдельной строке: сначала цену, затем количество.

Программы с линейной структурой

Приступая к решению задач этого раздела, следует вспомнить, что:

- программы с линейной структурой являются простейшими и используются, как правило, для реализации обычных вычислений по формулам;

- в программах с линейной структурой инструкции выполняются последовательно, одна за другой;
- алгоритм программы с линейной структурой может быть представлен следующим образом:



51. Написать программу вычисления площади параллелограмма. Ниже представлен рекомендуемый вид экрана во время работы программы (данные, введенные пользователем, выделены полужирным шрифтом).

Вычисление площади параллелограмма.

Введите исходные данные:

Длина (см) → **9**

Ширина (см) → **7.5**

Площадь параллелограмма: 67.50 кв.см.

52. Написать программу вычисления объема параллелепипеда. Ниже представлен рекомендуемый вид экрана во время работы программы (данные, введенные пользователем, выделены полужирным шрифтом).

Вычисление объема параллелепипеда.

Введите исходные данные:

Длинна (см) → **9**

Ширина (см) → **7.5**

Высота (см) → **5**

Объем: 337.50 куб.см.

53. Написать программу вычисления площади поверхности параллелепипеда. Ниже представлен рекомендуемый вид экрана во время работы программы (данные, введенные пользователем, выделены полужирным шрифтом).

Вычисление площади поверхности параллелепипеда.

Введите исходные данные:

Длина (см) → **9**

Ширина (см) → **7.5**

Высота (см) → **5**

Площадь поверхности: 90.00 кв.см.

54. Написать программу вычисления объема куба. Ниже представлен рекомендуемый вид экрана во время работы программы (данные, введенные пользователем, выделены полужирным шрифтом).

Вычисление объема куба.

Введите длину ребра (см) и нажмите <Enter>

→ **9.5**

Объем куба: 857.38 куб.см.

55. Написать программу вычисления объема цилиндра. Ниже представлен рекомендуемый вид экрана во время работы программы (данные, введенные пользователем, выделены полужирным шрифтом).

Вычисление объема цилиндра

Введите исходные данные:

Радиус основания (см) → **5**

Высота цилиндра (см) → **10**

Объем цилиндра 1570.80 куб. см.

Для завершения работы программы нажмите <Enter>.

56. Написать программу вычисления стоимости покупки, состоящей из нескольких тетрадей и карандашей. Ниже представлен рекомендуемый вид экрана во время работы программы (данные, введенные пользователем, выделены полужирным шрифтом).

Вычисление стоимости покупки.

Введите исходные данные:

Цена тетради (руб.) → **2.75**
Количество тетрадей → **5**
Цена карандаша (руб.) → **0.85**
Количество карандашей → **2**

Стоимость покупки: 15.45 руб.

57. Написать программу вычисления стоимости покупки, состоящей из нескольких тетрадей и такого же количества обложек к ним. Ниже представлен рекомендуемый вид экрана во время работы программы (данные, введенные пользователем, выделены полужирным шрифтом).

Вычисление стоимости покупки.
Введите исходные данные:
Цена тетради (руб.) → **2.75**
Цена обложки (руб.) → **0.5**
Количество комплектов (шт.) → **7**

Стоимость покупки: 22.75 руб.

58. Написать программу вычисления стоимости некоторого количества (по весу) яблок. Ниже представлен рекомендуемый вид экрана во время работы программы (данные, введенные пользователем, выделены полужирным шрифтом).

Вычисление стоимости покупки.
Введите исходные данные:
Цена одного килограмма яблок (руб.) → **8.5**
Вес яблок (кг) → **2.3**

Стоимость покупки: 19.55 руб.

59. Написать программу вычисления площади треугольника, если известна длина основания и высота. Ниже представлен рекомендуемый вид экрана во время работы программы (данные, введенные пользователем, выделены полужирным шрифтом).

Вычисление площади треугольника.
Введите исходные данные:
Основание (см) → **8.5**
Высота (см) → **10**

Площадь треугольника 42.50 кв.см.

60. Написать программу вычисления площади треугольника, если известны длины двух его сторон и величина угла между этими сторонами. Ниже представлен рекомендуемый вид экрана во время работы программы (данные, введенные пользователем, выделены полужирным шрифтом).

Вычисление площади треугольника.

Введите (через пробел) длины двух сторон (см) треугольника

→ **25 17**

Введите величину угла между сторонами треугольника

→ **30**

Площадь треугольника: 106.25 кв.см.

61. Написать программу вычисления сопротивления электрической цепи, состоящей из двух параллельно соединенных сопротивлений. Ниже представлен рекомендуемый вид экрана во время работы программы (данные, введенные пользователем, выделены полужирным шрифтом).

Вычисление сопротивления электрической цепи

при параллельном соединении элементов.

Введите исходные данные:

Величина первого сопротивления (Ом) → **15**

Величина второго сопротивления (Ом) → **20**

Сопротивление цепи: 8.57 Ом.

62. Написать программу вычисления сопротивления электрической цепи, состоящей из двух последовательно соединенных сопротивлений. Ниже представлен рекомендуемый вид экрана во время работы программы (данные, введенные пользователем, выделены полужирным шрифтом).

Вычисление сопротивления электрической цепи.

Введите исходные данные:

Величина первого сопротивления (Ом) → **15**

Величина второго сопротивления (Ом) → **27.3**

Сопротивление цепи (последовательное соединение): 42.30 Ом.

63. Написать программу вычисления силы тока в электрической цепи. Ниже представлен рекомендуемый вид экрана во время работы программы (данные, введенные пользователем, выделены полужирным шрифтом).

Вычисление силы тока в электрической цепи.

Введите исходные данные:

Напряжение (вольт) → **36**

Сопротивление (Ом) → **1500**

Сила тока: 0.024 Ампер.

64. Написать программу вычисления расстояния между населенными пунктами, изображенными на карте. Ниже представлен рекомендуемый вид экрана во время работы программы (данные, введенные пользователем, выделены полужирным шрифтом).

Вычисление расстояния между населенными пунктами.

Введите исходные данные:

Масштаб карты (количество километров в одном сантиметре) → **120**

Расстояние между точками, изображающими населенные пункты (см) → **3.5**

Расстояние между населенными пунктами 420 км.

65. Написать программу вычисления стоимости поездки на автомобиле на дачу (туда и обратно). Исходными данными являются: расстояние до дачи (в километрах); количество бензина, которое потребляет автомобиль на 100 км пробега; цена одного литра бензина. Ниже представлен рекомендуемый вид экрана во время работы программы (данные, введенные пользователем, выделены полужирным шрифтом).

Вычисление стоимости поездки на дачу и обратно.

Расстояние до дачи (км) → **67**

Расход бензина (литров на 100 км пробега) → **8.5**

Цена литра бензина (руб.) → **6.5**

Поездка на дачу и обратно обойдется в 74.04 руб.

66. Написать программу вычисления объема цилиндра. Ниже представлен рекомендуемый вид экрана во время работы программы (данные, введенные пользователем, выделены полужирным шрифтом).

Вычисление объема цилиндра.

Введите исходные данные:

Радиус основания (см) → 5.5

Высота цилиндра (см) → 7

Объем цилиндра: 665.23 куб. см.

67. Написать программу вычисления площади поверхности цилиндра. Ниже представлен рекомендуемый вид экрана во время работы программы (данные, введенные пользователем, выделены полужирным шрифтом).

Вычисление площади поверхности цилиндра.

Введите исходные данные:

Радиус основания (см) → 5.5

Высота цилиндра (см) → 7

Площадь поверхности цилиндра: 431.97 кв.см.

68. Написать программу вычисления объема параллелепипеда. Ниже представлен рекомендуемый вид экрана во время работы программы (данные, введенные пользователем, выделены полужирным шрифтом).

Вычисление объема параллелепипеда.

Введите в одной строке длину, ширину и высоту параллелепипеда (в сантиметрах).

Числа разделяйте пробелами. После ввода последнего числа нажмите <Enter>.

→ 7.5 2.5 3

Объем параллелепипеда 56.25 куб. см.

69. Написать программу пересчета расстояния из верст в километры (1 верста равняется 1066,8 м). Ниже представлен рекомендуемый вид экрана во время работы программы (данные, введенные пользователем, выделены полужирным шрифтом).

Пересчет расстояния из верст в километры.

Введите расстояние в верстах и нажмите <Enter>.

→ 100

100 верст(а/ы) — это 106.68 км.

70. Написать программу пересчета веса из фунтов в килограммы (1 российский фунт равен 409,5 г). Ниже представлен рекомен-

дуемый вид экрана во время работы программы (данные, введенные пользователем, выделены полужирным шрифтом).

Пересчет веса из фунтов в килограммы.

Введите вес в фунтах и нажмите <Enter>

→ **5**

5 фунт(а/ов) — это 2.05 кг.

71. Написать программу вычисления величины дохода по вкладу. Процентная ставка (в процентах годовых) и время хранения (в днях) задаются во время работы программы. Ниже представлен рекомендуемый вид экрана во время работы программы (данные, введенные пользователем, выделены полужирным шрифтом).

Вычисление дохода по вкладу.

Введите исходные данные:

Величина вклада (руб.) → **2500**

Срок вклада (дней) → **30**

Процентная ставка (годовых) → **20**

Доход: 41.10 руб.

Сумма по окончании срока вклада: 2541.10 руб.

72. Написать программу пересчета величины временного интервала, заданного в минутах, в величину, выраженную в часах и минутах. Ниже представлен рекомендуемый вид экрана во время работы программы (данные, введенные пользователем, выделены полужирным шрифтом).

Введите величину временного интервала (в минутах)

и нажмите <Enter>

→ **150**

150 минут — это 2 ч. 30 мин.

73. Написать программу, которая преобразует введенное с клавиатуры дробное число в денежный формат. Например, число 125 должно быть преобразовано к виду 12 руб. 50 коп.

Преобразование числа в денежный формат.

Введите дробное число → **23.6**

23.6 руб. — это 23 руб. 60 коп.

74. Написать программу пересчета веса из фунтов в килограммы (1 фунт равняется 409,5 г). Ниже представлен рекомендуемый вид экрана во время работы программы (данные, введенные пользователем, выделены полужирным шрифтом).

Пересчет веса из фунтов в килограммы.

Введите вес в фунтах и нажмите <Enter>.

→ **3.5**

3.5 фунт(а/ов) — это 1 кг 420 г.

75. Написать программу, которая вычисляет площадь треугольника, если известны координаты его углов. Ниже представлен рекомендуемый вид экрана во время работы программы (данные, введенные пользователем, выделены полужирным шрифтом).

Вычисление площади треугольника.

Введите координаты углов

(числа разделяйте пробелом):

x1, y1 → **-2 5**

x2, y2 → **1 7**

x3, y3 → **5 -3**

Площадь треугольника: 23.56 кв.см.

Выбор

If

Приступая к решению задач этого раздела, следует вспомнить, что:

- инструкция `if` используется для выбора одного из двух направлений дальнейшего хода программы (последовательности инструкций, которые должны быть выполнены);
- выбор последовательности инструкций осуществляется во время выполнения программы в зависимости от выполнения условия;
- условие — это выражение логического типа, которое может принимать одно из двух значений: `TRUE` (истина — условие выполняется) или `FALSE` (ложь — условие не выполняется);

- при помощи логических операций And (логическое "И") и Or (логическое "Или") из простых условий можно строить сложные.

76. Написать программу, которая вычисляет частное от деления двух чисел. Программа должна проверять правильность введенных пользователем данных и, если они неверные (делитель равен нулю), выдавать сообщение об ошибке. Ниже представлен рекомендуемый вид экрана во время работы программы (данные, введенные пользователем, выделены полужирным шрифтом).

Вычисление частного.

Введите в одной строке делимое и делитель,
затем нажмите <Enter>

→ **12 0**

Вы ошиблись. Делитель не должен быть равен нулю.

77. Написать программу вычисления площади кольца. Программа должна проверять правильность исходных данных. Ниже представлен рекомендуемый вид экрана во время работы программы (данные, введенные пользователем, выделены полужирным шрифтом).

Вычисление площади кольца.

Введите исходные данные:

Радиус кольца (см) → **3.5**

Радиус отверстия (см) → **7**

Ошибка! Радиус отверстия не может быть больше радиуса кольца.

78. Написать программу вычисления сопротивления электрической цепи, состоящей из двух сопротивлений. Сопротивления могут быть соединены последовательно или параллельно. Ниже представлен рекомендуемый вид экрана во время работы программы (данные, введенные пользователем, выделены полужирным шрифтом).

Вычисление сопротивления электрической цепи.

Введите исходные данные:

Величина первого сопротивления (Ом) → **15**

Величина второго сопротивления (Ом) → **27.3**

Тип соединения (1 - последовательное, 2 - параллельное) → **2**

Сопротивление цепи: 9.68 Ом.

79. Написать программу решения квадратного уравнения. Программа должна проверять правильность исходных данных и в случае, когда коэффициент при второй степени неизвестного равен нулю, выводить соответствующее сообщение. Ниже представлен рекомендуемый вид экрана во время работы программы (данные, введенные пользователем, выделены полужирным шрифтом).

* Решение квадратного уравнения *

Введите в одной строке значения коэффициентов и нажмите <Enter>

→ **12 27 -10**

Корни уравнения:

x1= -25.551

x2= -28.449

80. Написать программу вычисления стоимости покупки с учетом скидки. Скидка в 10% предоставляется, если сумма покупки больше 1000 руб. Ниже представлен рекомендуемый вид экрана во время работы программы (данные, введенные пользователем, выделены полужирным шрифтом).

Вычисление стоимости покупки с учетом скидки.

Введите сумму покупки и нажмите <Enter>

→ **1200**

Вам предоставляется скидка 10%

Сумма покупки с учетом скидки: 1080.00 руб.

81. Написать программу вычисления стоимости покупки с учетом скидки. Скидка в 3% предоставляется в том случае, если сумма покупки больше 500 руб., в 5% — если сумма больше 1000 руб. Ниже представлен рекомендуемый вид экрана во время работы программы (данные, введенные пользователем, выделены полужирным шрифтом).

Вычисление стоимости покупки с учетом скидки.

Введите сумму покупки и нажмите <Enter>

→ **640**

Вам предоставляется скидка 3%

Сумма покупки с учетом скидки: 620.80 руб.

82. Написать программу проверки знания даты основания Санкт-Петербурга. В случае неверного ответа пользователя программа должна выводить правильный ответ. Ниже представлен реко-

мендуемый вид экрана во время работы программы (данные, введенные пользователем, выделены полужирным шрифтом).

В каком году был основан Санкт-Петербург?

Введите число и нажмите <Enter>

→ **1705**

Вы ошиблись, Санкт-Петербург был основан в 1703 году.

83. Написать программу проверки знания даты начала второй мировой войны. В случае неверного ответа пользователя программа должна выводить правильный ответ. Ниже представлен рекомендуемый вид экрана во время работы программы (данные, введенные пользователем, выделены полужирным шрифтом).

В каком году началась вторая мировая война?

Введите число и нажмите <Enter>

→ **1939**

Правильно.

84. Написать программу проверки знания истории архитектуры. Программа должна вывести вопрос и три варианта ответа. Пользователь должен выбрать правильный ответ и ввести его номер. Ниже представлен рекомендуемый вид экрана во время работы программы (данные, введенные пользователем, выделены полужирным шрифтом).

Архитектор Исаакиевского собора:

1. Доменико Трезини
2. Огюст Монферран
3. Карл Росси

Введите номер правильного ответа и нажмите <Enter>

→ **2**

Вы ошиблись.

Архитектор Исаакиевского собора — Огюст Монферран.

85. Написать программу проверки знания истории архитектуры. Программа должна вывести вопрос и три варианта ответа, а пользователь — выбрать правильный ответ и ввести его номер. Ниже представлен рекомендуемый вид экрана во время работы программы (данные, введенные пользователем, выделены полужирным шрифтом).

Невский проспект получил свое название:

1. По имени реки, на берегах которой расположен Санкт-Петербург
2. По имени близко расположенного монастыря
Александро-Невской лавры
3. В память о знаменитом полководце Александре Невском

Введите номер правильного ответа и нажмите <Enter>

→ 2

Вы ошиблись.

Правильный ответ: 2.

86. Написать программу, которая сравнивает два числа, введенных с клавиатуры. Программа должна указать, какое число больше, или, если числа равны, вывести соответствующее сообщение. Ниже представлен рекомендуемый вид экрана во время работы программы.

Введите в одной строке два целых числа и нажмите <Enter>.

→ 34 67

34 меньше 67.

87. Написать программу, которая выводит пример на умножение двух однозначных чисел, запрашивает ответ пользователя, проверяет его и выводит сообщение "Правильно!" или "Вы ошиблись" и правильный результат. Ниже представлен рекомендуемый вид экрана во время работы программы (данные, введенные пользователем, выделены полужирным шрифтом).

Сколько будет 6х7?

Введите ответ и нажмите <Enter>

→ 56

Вы ошиблись. 6х7=42.

88. Написать программу, которая выводит пример на вычитание (в пределах 100), запрашивает ответ пользователя, проверяет его и выводит сообщение "Правильно!" или "Вы ошиблись" и правильный результат. Ниже представлен рекомендуемый вид экрана во время работы программы (данные, введенные пользователем, выделены полужирным шрифтом).

Сколько будет 83-17?

Введите ответ и нажмите <Enter>

→ 67

Вы ошиблись. 83-17=66.

89. Написать программу, которая проверяет, является ли четным введенное пользователем целое число. Ниже представлен рекомендуемый вид экрана программы во время ее работы (данные, введенные пользователем, выделены полужирным шрифтом).

Введите целое число и нажмите <Enter>

→ **23**

Число 23 — нечетное.

90. Написать программу, которая проверяет, делится ли на три целое число, введенное с клавиатуры. Ниже представлен рекомендуемый вид экрана программы во время ее работы (данные, введенные пользователем, выделены полужирным шрифтом).

Введите целое число и нажмите <Enter>

→ **451**

Число 451 нацело на три не делится.

91. Написать программу определения стоимости разговора по телефону с учетом скидки 20%, предоставляемой по субботам и воскресеньям. Ниже представлен рекомендуемый вид экрана программы во время ее работы (данные, введенные пользователем, выделены полужирным шрифтом).

Вычисление стоимости разговора по телефону.

Введите исходные данные:

Длительность разговора (целое количество минут) → **3**

День недели (1 — понедельник, ... 7 — воскресенье) → **6**

Предоставляется скидка 20%.

Стоимость разговора: 5.52 руб.

92. Написать программу, которая вычисляет оптимальный вес пользователя, сравнивает его с реальным и выдает рекомендацию о необходимости поправиться или похудеть. Оптимальный вес вычисляется по формуле: $\text{рост (в сантиметрах)} - 100$. Рекомендуемый вид экрана во время работы программы приведен ниже (данные, введенные пользователем, выделены полужирным шрифтом).

Введите в одной строке через пробел

рост (см) и вес (кг) затем нажмите <Enter>

→ **170 68**

Вам надо поправиться на 2.00 кг.

Case

Приступая к решению задач этого раздела, следует вспомнить, что:

- инструкция `Case` используется для выбора одного из нескольких направлений дальнейшего хода программы (последовательности инструкций, которые должны быть выполнены);
- выбор последовательности инструкций осуществляется во время выполнения программы в зависимости от равенства значения переменной-селектора константе, указанной перед группой инструкций;
- в качестве переменной-селектора можно использовать переменную целого (`integer`) или символьного (`char`) типа.

93. Написать программу, которая запрашивает у пользователя номер месяца и выводит соответствующее название времени года. В случае, если пользователь укажет недопустимое число, программа должна вывести сообщение "Ошибка ввода данных". Ниже представлен рекомендуемый вид экрана во время работы программы.

Введите номер месяца (число от 1 до 12) и нажмите <Enter>

→ 11

Зима.

94. Написать программу, которая запрашивает у пользователя номер дня недели и выводит одно из сообщений: "Рабочий день", "Суббота" или "Воскресенье".

95. Написать программу, вычисляющую стоимость междугороднего телефонного разговора (цена одной минуты определяется расстоянием до города, в котором находится абонент). Исходными данными для программы являются код города и продолжительность разговора. Ниже представлен рекомендуемый вид экрана во время работы программы (данные, введенные пользователем, и коды некоторых городов выделены полужирным шрифтом).

Вычисление стоимости разговора по телефону.

Введите исходные данные:

Код города → **812**

Длительность (целое количество минут) → 3

Город: Владивосток

Цена минуты: 2.20 руб.

Стоимость разговора: 6.60 руб.

Город	Код	Цена минуты, руб.
Владивосток	423	2, 2
Москва	095	1, 0
Мурманск	815	1, 2
Самара	846	1, 4

96. Написать программу, которая после введенного с клавиатуры числа (в диапазоне от 1 до 999), обозначающего денежную единицу, дописывает слово "рубль" в правильной форме. Например, 12 рублей, 21 рубль и т. д.

97. Написать программу, которая после введенного с клавиатуры числа (в диапазоне от 1 до 99), обозначающего денежную единицу, дописывает слово "копейка" в правильной форме. Например, 5 копеек, 41 копейка и т. д.

98. Написать программу, которая вычисляет дату следующего дня. Ниже представлен рекомендуемый вид экрана во время работы программы. (данные, введенные пользователем, выделены полужирным шрифтом).

Введите цифрами сегодняшнюю дату (число месяц год) → **31 12 1999**.

Последний день месяца!

С наступающим новым годом!

Завтра 1.1.2000.

Циклы

For

Приступая к решению задач этого раздела, следует вспомнить, что:

- инструкция `for` используется для организации циклов с фиксированным, определяемым во время разработки программы, числом повторений;

- количество повторений цикла определяется начальным и конечным значениями переменной-счетчика;
- переменная-счетчик должна быть целого типа (*integer*).

99. Написать программу, которая 10 раз выводит на экран имя и фамилию.

100. Написать программу, которая выводит таблицу квадратов первых десяти целых положительных чисел. Ниже представлен рекомендуемый вид экрана во время работы программы.

Таблица квадратов

Число Квадрат

1	1
2	4
3	9
4	16
5	25
6	36
7	49
8	64
9	81
10	100

101. Написать программу, которая выводит таблицу квадратов первых пяти целых положительных нечетных чисел. Ниже представлен рекомендуемый вид экрана во время работы программы.

Таблица квадратов нечетных чисел

Число Квадрат

1	1
3	9
5	25
7	49
9	81

102. Написать программу, которая вычисляет сумму первых *n* целых положительных целых чисел. Количество суммируемых

чисел должно вводиться во время работы программы. Ниже представлен рекомендуемый вид экрана во время работы программы (данные, введенные пользователем, выделены полужирным шрифтом).

Вычисление суммы положительных чисел.

Введите количество суммируемых чисел → **20**

Сумма первых 20 положительных чисел равна 210.

103. Написать программу, которая вычисляет сумму первых n целых положительных четных чисел. Количество суммируемых чисел должно вводиться во время работы программы. Ниже представлен рекомендуемый вид экрана во время работы программы (данные, введенные пользователем, выделены полужирным шрифтом).

Вычисление суммы четных положительных чисел.

Введите количество суммируемых чисел и нажмите <Enter>

→ **12**

Сумма первых 12 положительных четных чисел равна 156.

104. Написать программу, которая вычисляет сумму первых n членов ряда 1, 3, 5, 7, Количество суммируемых членов ряда задается во время работы программы. Ниже представлен рекомендуемый вид экрана во время работы программы (данные, введенные пользователем, выделены полужирным шрифтом).

Вычисление частичной суммы ряда: 1, 3, 5, 7, ...

Введите количество суммируемых членов ряда → **15**

Сумма первых 15 членов ряда равна 330.

105. Написать программу, которая вычисляет сумму первых n членов ряда $1 + \frac{1}{2} + \frac{1}{3} + \frac{1}{4} + \dots$. Количество суммируемых членов ряда задается во время работы программы. Ниже представлен рекомендуемый вид экрана во время работы программы (данные, введенные пользователем, выделены полужирным шрифтом).

Вычисление частичной суммы ряда: $1 + 1/2 + 1/3 + \dots$

Введите кол-во суммируемых членов ряда → **15**

Сумма первых 15 членов ряда равна 3.3182.

106. Написать программу, которая выводит таблицу степеней двойки (от нулевой до десятой). Ниже представлен рекомендуемый вид экрана во время работы программы.

Таблица степеней двойки

0	1
1	2
2	4
3	8
4	16
5	32
6	64
7	128
8	256
9	512
10	1024

107. Написать программу, которая вычисляет факториал числа, введенного с клавиатуры. (Факториалом числа n называется произведение целых чисел от 1 до n . Например, факториал числа 1 равен 1, а числа 8 — 40 320.)

Вычисление факториала.

Введите число, факториал которого надо вычислить

→ 7

Факториал 7 равен 5040.

108. Написать программу, которая выводит таблицу значений функции $y = -2,4x^2 + 5x - 3$ в диапазоне от -2 до 2 с шагом $0,5$. Ниже представлен рекомендуемый вид экрана во время работы программы.

x	y

-2	-22.60
-1.5	-15.90
-1	-10.40
-0.5	-6.10
0	-3.00
0.5	-1.10
1	-0.40
1.5	-0.90
2	-2.60

109. Написать программу, которая вводит с клавиатуры 5 дробных чисел и вычисляет их среднее арифметическое. Рекомендуемый вид экрана во время работы программы приведен ниже. Данные, введенные пользователем, выделены полужирным шрифтом.

Вычисление среднего арифметического последовательности дробных чисел. После ввода каждого числа нажимайте <Enter>

→ **5.4**

→ **7.8**

→ **3.0**

→ **1.5**

→ **2.3**

Среднее арифметическое введенной последовательности: 4.00
Для завершения работы программы нажмите <Enter>.

110. Написать программу, которая вычисляет среднее арифметическое вводимой с клавиатуры последовательности дробных чисел. Количество чисел должно задаваться во время работы программы. Рекомендуемый вид экрана во время работы программы приведен ниже.

Вычисление среднего арифметического последовательности дробных чисел.

Введите количество чисел последовательности → **5**

Вводите последовательность. После ввода каждого числа нажимайте <Enter>

→ **5.4**

→ **7.8**

→ **3.0**

→ **1.5**

→ **2.3**

Среднее арифметическое введенной последовательности: 4.00
Для завершения работы программы нажмите <Enter>.

111. Написать программу, которая вводит с клавиатуры последовательность из пяти дробных чисел и после ввода каждого числа выводит среднее арифметическое полученной части последовательности. Рекомендуемый вид экрана во время работы программы приведен ниже.

Обработка последовательности дробных чисел.

После ввода каждого числа нажимайте <Enter>

→ **12.3**

Введено чисел: 1 Сумма: 12.30 Сред.арифметическое: 12.30
→ **15**
Введено чисел: 2 Сумма: 27.30 Сред.арифметическое: 13.65
→ **10**
Введено чисел: 3 Сумма: 37.30 Сред.арифметическое: 12.43
→ **5.6**
Введено чисел: 4 Сумма: 42.90 Сред.арифметическое: 10.73
→ **11.5**
Введено чисел: 5 Сумма: 54.40 Сред.арифметическое: 10.88
Для завершения работы программы нажмите <Enter>.

112. Написать программу, которая вычисляет среднее арифметическое последовательности дробных чисел, вводимых с клавиатуры. После того, как будет введено последнее число, программа должна вывести минимальное и максимальное число последовательности. Количество чисел должно задаваться во время работы программы. Рекомендваемый вид экрана приведен ниже. Данные, введенные пользователем, выделены полужирным шрифтом.

Обработка последовательности дробных чисел.
Введите количество чисел последовательности → **5**
Вводите последовательность. После ввода каждого числа нажимайте <Enter>
→ **5.4**
→ **7.8**
→ **3.0**
→ **1.5**
→ **2.3**

Количество чисел: 5
Среднее арифметическое: 4.00
Минимальное число:
Максимальное число:
Для завершения работы программы нажмите <Enter>.

113. Написать программу, которая генерирует 10 случайных чисел в диапазоне от 1 до 10, выводит эти числа на экран и вычисляет их среднее арифметическое. Рекомендваемый вид экрана во время работы программы приведен ниже.

*** Случайные числа ***

1 3 4 2 7 4 9 6 2 1 сред.арифм. 3.9

114. Написать программу, которая генерирует три последовательности из десяти случайных чисел в диапазоне от 1 до 10, выводит каждую последовательность на экран и вычисляет среднее арифметическое каждой последовательности. Рекомендуемый вид экрана во время работы программы приведен ниже.

*** Случайные числа ***

6	10	4	2	5	8	1	7	7	3	сред. арифм.	5.30
10	3	6	1	10	1	3	8	7	6	сред. арифм.	5.50
5	2	2	5	4	2	2	1	6	10	сред. арифм.	3.90

Для завершения работы нажмите <Enter>.

115. Написать программу, которая выводит на экран таблицу стоимости, например, яблок в диапазоне от 100 г до 1 кг с шагом 100. Ниже представлен рекомендуемый вид экрана программы во время ее работы (данные, введенные пользователем, выделены полужирным шрифтом).

Введите цену одного килограмма и нажмите <Enter>

(копейки от рублей отделяйте точкой)

→ **16.50**

Вес, г	Стоимость, руб.
100	1.65
200	3.30
300	4.95
400	6.60
500	8.25
600	9.90
700	11.55
800	13.20
900	14.85
1000	16.50

116. Написать программу, которая выводит таблицу значений функции $y = |x|$. Диапазон изменения аргумента — от -4 до 4 , шаг приращения аргумента — $0,5$.

117. Написать программу, которая выводит таблицу значений функции $y = |x + 2|$. Диапазон изменения аргумента — от -4 до 4 , шаг приращения аргумента — $0,5$.

118. Написать программу, которая выводит таблицу значений функции $y = |x - 2| + |x + 1|$. Диапазон изменения аргумента — от -4 до 4 , шаг приращения аргумента — $0,5$.

119. Написать программу, которая выводит на экран таблицу умножения, например, на 7. Рекомендуемый вид экрана во время работы программы приведен ниже.

```
7 × 2 = 14
7 × 3 = 21
7 × 4 = 28
7 × 5 = 35
7 × 6 = 42
7 × 7 = 49
7 × 8 = 56
7 × 9 = 63
```

120. Написать программу, которая выводит двоичное представление введенного с клавиатуры десятичного целого числа в диапазоне от 0 до 255. Рекомендуемый вид экрана во время работы программы приведен ниже (данные, введенные пользователем, выделены полужирным шрифтом).

Преобразование десятичного числа в двоичное.

Введите целое число от 0 до 255 и нажмите <Enter>

→ **164**

Десятичному числу 64 соответствует двоичное 10100100

Для завершения работы программы нажмите <Enter>.

121. Написать программу, которая выводит на экран квадрат Пифагора — таблицу умножения. Рекомендуемый вид экрана во время работы программы приведен ниже.

	1	2	3	4	5	6	7	8	9	10
1	1	2	3	4	5	6	7	8	9	10
2	2	4	6	8	10	12	14	16	18	20
3	3	6	9	12	15	18	21	24	27	30
4	4	8	12	16	20	24	28	32	36	40
5	5	10	15	20	25	30	35	40	45	50
6	6	12	18	24	30	36	42	48	54	60
7	7	14	21	28	35	42	49	56	63	70
8	8	16	24	32	40	48	56	64	72	80
9	9	18	27	36	45	54	63	72	81	90

122. Напишите программу, которая вычисляет частичную сумму ряда $1 - 1/3 + 1/5 - 1/7 + 1/9 - \dots$ и сравнивает полученное значение с $\pi/4$ (при суммировании достаточно большого количества членов этого ряда, величина частичной суммы приближается к $\pi/4$).

123. Напишите программу приближенного вычисления интеграла функции $f(x) = 5x^2 - x + 2$ методом прямоугольников.

124. Напишите программу приближенного вычисления интеграла методом трапеций.

125. Напишите программу, которая выводит на экран изображение шахматной доски. Черные клетки отображать "звездочкой", белые — пробелом. Рекомендуемый вид экрана во время работы программы приведен ниже.

```
* * * *
 * * * *
* * * *
 * * * *
* * * *
 * * * *
* * * *
 * * * *
* * * *
```

126. Написать программу, которая преобразует введенное пользователем десятичное число в двоичное. Рекомендуемый вид экрана во время работы программы приведен ниже.

Преобразование десятичного числа в двоичное.

Введите целое число от 0 до 255 и нажмите <Enter>

→ 49

Десятичному числу 49 соответствует двоичное 00110001

Для завершения работы программы нажмите <Enter>.

Факультатив

127. Написать программу проверки знания таблицы умножения. Программа должна вывести 10 примеров и выставить оценку: за 10 правильных ответов — "отлично", за 9 и 8 — "хорошо", за 7 и 6 — "удовлетворительно", за 6 и менее — "плохо". Ниже представлен рекомендуемый вид экрана во время работы программы. Ответы пользователя выделены полужирным шрифтом.

*** Проверка знания таблицы умножения ***

После примера введите ответ и нажмите <Enter>

$$5 \times 3 = \mathbf{15}$$

$$7 \times 7 = \mathbf{49}$$

$$1 \times 4 = \mathbf{4}$$

$$4 \times 3 = \mathbf{12}$$

$$9 \times 4 = \mathbf{36}$$

$$8 \times 8 = \mathbf{64}$$

$$7 \times 8 = \mathbf{52}$$

Вы ошиблись! $7 \times 8 = 56$

$$4 \times 7 = \mathbf{28}$$

$$3 \times 5 = \mathbf{15}$$

$$2 \times 5 = \mathbf{10}$$

Правильных ответов: 9

Оценка: Хорошо.

128. Написать программу проверки умения складывать и вычитать числа в пределах 100. Программа должна вывести 10 примеров, причем в каждом примере уменьшаемое должно быть больше или равно вычитаемому, т. е. не допускается предлагать испытуемому примеры с отрицательным результатом. Оценка выставляется по следующему правилу: за 10 правильных ответов — "отлично", за 9 и 8 — "хорошо", за 7 и 6 — "удовлетворительно", за 6 и менее — "плохо". Ниже представлен рекомендуемый вид экрана во время работы программы. Ответы пользователя выделены полужирным шрифтом.

Проверка умения складывать и вычитать числа.

После примера введите ответ и нажмите <Enter>

$$75 - 4 = \mathbf{71}$$

$$35 - 9 = \mathbf{29}$$

Вы ошиблись! $35 - 9 = 26$

$$14 - 1 = \mathbf{13}$$

$$6 - 5 = \mathbf{1}$$

$$37 - 19 = \mathbf{28}$$

Вы ошиблись! $37 - 19 = 18$

$$53 - 14 = \mathbf{39}$$

$$94 - 87 = \mathbf{7}$$

$$90 - 16 = \mathbf{74}$$

$$4 - 2 = \mathbf{2}$$

$$89 - 41 = \mathbf{48}$$

Правильных ответов: 8

Оценка: Хорошо.

129. Написать программу, выводящую на экран "электронные часы", которые работают в течение, например, трех минут или до тех пор, пока пользователь не нажмет любую клавишу.

Repeat

Приступая к решению задач этого раздела, следует вспомнить, что:

- число повторений инструкций цикла `repeat` определяется ходом выполнения программы;
- инструкции цикла `repeat` выполняются до тех пор, пока условие, стоящее после слова `until`, ложно;
- после слова `until` надо записывать условие завершения цикла;
- для завершения цикла `repeat` в теле цикла обязательно должны быть инструкции, выполнение которых влияет на условие завершения цикла;
- цикл `repeat` — это цикл с постусловием, т. е. инструкции тела цикла будут выполнены хотя бы один раз;
- цикл `repeat`, как правило, используется для организации приближенных вычислений, задач поиска и обработки данных, вводимых с клавиатуры или из файла.

130. Написать программу, вычисляющую сумму и среднее арифметическое последовательности положительных чисел, которые вводятся с клавиатуры. Ниже представлен рекомендуемый вид экрана во время работы программы (данные, введенные пользователем, выделены полужирным шрифтом).

Вычисление среднего арифметического последовательности положительных чисел.

Вводите после стрелки числа. Для завершения ввода введите ноль.

-> **45**

-> **23**

-> **15**

-> **0**

Введено чисел: 3

Сумма чисел: 83

Среднее арифметическое: 27.67.

131. Написать программу, которая определяет максимальное число из введенной с клавиатуры последовательности положительных чисел (длина последовательности не ограничена). Ниже представлен рекомендуемый вид экрана во время работы программы (данные, введенные пользователем, выделены полужирным шрифтом).

Определение максимального числа последовательности положительных чисел.

Вводите после стрелки числа. Для завершения ввода введите ноль.

-> **56**

-> **75**

-> **43**

-> **0**

Максимальное число: 75.

132. Написать программу, которая проверяет, является ли целое число, введенное пользователем, простым. Рекомендуемый вид экрана во время работы программы приведен ниже. Данные, введенные пользователем, выделены полужирным шрифтом.

Введите целое число и нажмите <Enter>

-> **45**

45 – не простое число.

133. Написать программу приближенного вычисления интеграла методом трапеций. После каждого цикла программа должна выводить вычисленное значение, количество и величину интервалов.

134. Написать программу, которая "задумывает" число в диапазоне от 1 до 10 и предлагает пользователю угадать число за 5 попыток. Ниже представлен рекомендуемый вид экрана во время работы программы (данные, введенные пользователем, выделены полужирным шрифтом).

Игра "Угадай число".

Компьютер "задумал" от 1 до 10 число от 1 до 10.

Угадайте его за 5 попыток.

Введите число и нажмите <Enter>

→ 5

Нет.

→ 3

Вы выиграли! Поздравляю!

Факультатив

135. Написать программу-таймер, которая по истечении заданного промежутка времени выдает звуковой сигнал. Величина временного интервала вводится с клавиатуры во время работы программы.

While

Приступая к решению задач этого раздела, следует вспомнить, что:

- число повторений инструкций цикла `while` определяется ходом выполнения программы;
- инструкции цикла `while` выполняются до тех пор, пока условие, записанное после слова `while`, истинно;
- после слова `while` надо записывать условие выполнения инструкций цикла;
- для завершения цикла `while` в теле цикла обязательно должны присутствовать инструкции, влияющие на условие выполнения инструкций цикла;
- цикл `while` — это цикл с предусловием, т. е. инструкции тела цикла вообще могут быть не выполнены;
- цикл `while`, как правило, используется для организации приближенных вычислений, задач поиска и обработки данных, вводимых с клавиатуры или из файла.

136. Напишите программу, которая вычисляет π с заданной пользователем точностью. Для этого воспользуйтесь тем, что значение частичной суммы ряда $1 - 1/3 + 1/5 - 1/7 + 1/9 - \dots$ при суммировании достаточно большого количества членов приближается к $\pi/4$. Рекомендуемый вид экрана во время работы программы приведен ниже (данные, введенные пользователем, выделены полужирным шрифтом).

Задайте точность вычисления π $\rightarrow$ **0.001**.

Значение числа π с точностью 0.001000 равно 3.143589.

Просуммировано 502 членов ряда.

137. Написать программу, которая вычисляет наибольший общий делитель двух целых чисел.

Символы и строки

Приступая к решению задач этого раздела, следует вспомнить, что:

- каждому символу соответствует число — код символа;
- получить доступ к отдельному символу строки можно, указав его номер в квадратных скобках после имени переменной (строки).

138. Написать программу, которая запрашивает имя пользователя и здоровается с ним. Рекомендуемый вид экрана во время работы программы приведен ниже (данные, введенные пользователем, выделены полужирным шрифтом).

Как Вас зовут?

Введите свое имя и нажмите <Enter>

$\rightarrow$ **Вася**

Здравствуйте, Вася!

139. Написать программу, которая выводит на экран сообщение в "телеграфном" стиле, т. е. буквы сообщения должны появляться по одной, причем с некоторой задержкой.

140. Написать программу, которая выдает код символа, введенного пользователем. Программа должна завершать работу после ввода, например, точки. Рекомендуемый вид экрана во время работы программы приведен ниже (данные, введенные пользователем, выделены полужирным шрифтом).

Введите символ и нажмите <Enter>.

Для завершения работы программы введите точку.

$\rightarrow$ **1**

Символ: 1 Код: 49.

→ 2

Символ: 2 Код: 50

→ ы

Символ: ы Код: 235

→ .

141. Написать программу, которая выводит на экран первую часть таблицы кодировки символов (символы, имеющие коды от 0 до 127). Таблица должна состоять из восьми колонок и шестнадцати строк. В первой колонке должны быть символы, имеющие код от 0 до 15, во второй — от 16 до 31 и т. д.

142. Написать программу, которая в введенной с клавиатуры строке преобразует строчные буквы русского алфавита в прописные (учтите, что функция `UpCase` с символами русского алфавита не работает). Рекомендуемый вид экрана во время работы программы приведен ниже (данные, введенные пользователем, выделены полужирным шрифтом).

Введите строку текста и нажмите <Enter>

→ **изучив Паскаль**, можно приступить к изучению С

Строка, преобразованная к верхнему регистру:

ИЗУЧИВ ПАСКАЛЬ, МОЖНО ПРИСТУПИТЬ К ИЗУЧЕНИЮ С.

143. Написать программу, которая удаляет начальные пробелы из введенной с клавиатуры строки.

144. Написать программу, которая проверяет, является ли введенная с клавиатуры строка целым числом. Рекомендуемый вид экрана во время работы программы приведен ниже (данные, введенные пользователем, выделены полужирным шрифтом).

Введите число и нажмите <Enter>

→ **23.5**

Введенная строка не является целым числом.

145. Написать программу, которая проверяет, является ли введенная с клавиатуры строка двоичным числом.

146. Написать программу, которая проверяет, является ли введенная с клавиатуры строка шестнадцатеричным числом.

147. Написать программу, которая проверяет, является ли введенная с клавиатуры строка дробным числом.

148. Написать программу, которая преобразует введенное с клавиатуры восьмиразрядное двоичное число в десятичное. Рекомендуемый вид экрана во время работы программы представлен ниже (введенные пользователем данные выделены полужирным шрифтом).

Введите восьмиразрядное двоичное число

и нажмите <Enter>

→ **11101010**

Двоичному числу 11101010 соответствует десятичное 234

Для завершения работы программы нажмите <Enter>.

149. Написать программу, которая преобразует введенное с клавиатуры двухразрядное шестнадцатеричное число в десятичное.

150. Написать программу, которая преобразует введенное пользователем десятичное число в число в указанной системе счисления (от 2-х до 10-ти). Рекомендуемый вид экрана во время работы программы приведен ниже.

Введите целое число → **67**

Введите основание системы счисления → **2**

100011

151. Написать программу, которая преобразует введенное пользователем десятичное число в шестнадцатеричное.

152. Написать программу, которая вычисляет значение выражения $N_0O_1N_1O_2...O_kN_k$, где N_i — целое одноразрядное число, O_i — один из двух знаков простейших арифметических действий: сложения или вычитания. Ниже представлен рекомендуемый вид экрана во время работы программы (данные, введенные пользователем, выделены полужирным шрифтом).

Введите арифметическое выражение,
например, 4+5-3-5+2 и нажимает <Enter>

→ **9-5+4+2-6**

Значение введенного выражения: 4

Для завершения программы нажмите <Enter>.

Массивы

Приступая к решению задач этого раздела, следует вспомнить, что:

- массив — это структура данных, представляющая собой совокупность элементов одного типа;
- массив должен быть объявлен в разделе объявления переменных;
- при объявлении массива удобно использовать именованные константы;
- доступ к элементу массива осуществляется путем указания индекса (номера) элемента, в качестве которого можно использовать выражение целого типа, например, целую константу или переменную типа `integer`;
- для ввода, вывода и обработки массивов удобно применять инструкции циклов (`for`, `while`, `repeat`).

153. Написать программу, которая вводит с клавиатуры одномерный массив из 5 целых чисел и выводит количество ненулевых элементов. Перед вводом каждого элемента на экране должна появляться подсказка с его номером.

После ввода каждого числа нажмите `<Enter>`

`a[1] -> 12`

`a[2] -> 0`

`a[3] -> 3`

`a[4] -> -1`

`a[5] -> 0`

В массиве 3 ненулевых элемента.

154. Написать программу, которая выводит минимальный элемент введенного с клавиатуры массива целых чисел. Ниже представлен рекомендуемый вид экрана во время работы программы (данные, введенные пользователем, выделены полужирным шрифтом).

Поиск минимального элемента массива.

Введите в одной строке элементы массива (5 целых чисел) и нажмите `<Enter>`

`-> 23 0 45 -5 12`

Минимальный элемент массива: **-5**.

155. Написать программу, которая вычисляет среднее арифметическое ненулевых элементов введенного с клавиатуры массива целых чисел. Ниже представлен рекомендуемый вид экрана во время работы программы (данные, введенные пользователем, выделены полужирным шрифтом).

Введите элементы массива (10 целых чисел) в одной строке и нажмите <Enter>.

→ **23 0 45 -5 12 0 -2 30 0 64**

Сумма элементов массива: 184

Количество ненулевых элементов: 7

Среднее арифметическое ненулевых элементов: 23.86.

156. Написать программу, которая вычисляет среднюю (за неделю) температуру воздуха. Исходные данные должны вводиться во время работы программы. Рекомендуемый вид экрана во время работы программы приведен ниже (данные, введенные пользователем, выделены полужирным шрифтом).

Введите температуру воздуха за неделю.

Понедельник → **12**

Вторник → **10**

Среда → **16**

Четверг → **18**

Пятница → **17**

Суббота → **16**

Воскресенье → **14**

Средняя температура за неделю: 14.71 град.

157. Написать программу, которая проверяет, находится ли в массиве введенное с клавиатуры число. Массив должен вводиться во время работы программы.

158. Написать программу, которая проверяет, представляют ли элементы введенного с клавиатуры массива возрастающую последовательность.

159. Написать программу, которая проверяет, образуют ли элементы введенного с клавиатуры массива неубывающую последовательность.

160. Написать программу, которая вычисляет, сколько раз введенное с клавиатуры число встречается в массиве.

161. Написать программу, которая проверяет, есть ли во введенном с клавиатуры массиве элементы с одинаковым значением.

162. Написать программу, которая методом прямого выбора сортирует по убыванию введенный с клавиатуры одномерный массив.

163. Написать программу, которая методом обмена ("пузырька") сортирует по убыванию введенный с клавиатуры одномерный массив.

164. Написать программу, которая объединяет два упорядоченных по возрастанию массива в один, также упорядоченный по возрастанию массив. Рекомендуемый вид экрана во время работы программы приведен ниже (данные, введенные пользователем, выделены полужирным шрифтом).

Объединение двух упорядоченных по возрастанию массивов.

Введите в одной строке элементы первого массива,

(5 целых чисел) → **1 3 5 7 9**

Введите в одной строке элементы второго массива,

(5 целых чисел) → **2 4 6 8 10**

Массив — результат

1 2 3 4 5 6 7 8 9 10

Для завершения работы нажмите <Enter>.

165. Написать программу, которая, используя метод бинарного поиска, выполняет поиск в упорядоченном по возрастанию массиве.

166. Написать программу, которая определяет количество учеников в классе, чей рост превышает средний. Рекомендуемый вид экрана во время работы программы приведен ниже (введенные пользователем данные выделены полужирным шрифтом).

*** Анализ роста учеников ***

Введите рост (см) и нажмите <Enter>.

Для завершения введите 0 и нажмите <Enter>

→ **175**

→ **170**

→ **180**

→ **168**

→ **170**

→ **0**

Средний рост: 172.6 см

У 2-х человек рост превышает средний.

167. Написать программу, которая вводит по строкам с клавиатуры двумерный массив и вычисляет сумму его элементов по столбцам.

158. Написать программу, которая вводит по строкам с клавиатуры двумерный массив и вычисляет сумму его элементов по строкам.

168. Написать программу, которая вычисляет сумму диагональных элементов квадратной матрицы.

169. Написать программу, которая вводит с клавиатуры двумерный массив по строкам и вычисляет среднее арифметическое его элементов.

170. Написать программу, которая вычисляет определитель квадратной матрицы второго порядка. Рекомендуемый вид экрана во время работы программы приведен ниже (введенные пользователем данные выделены полужирным шрифтом).

Введите матрицу второго порядка.

После ввода элементов строки нажимайте <Enter>

→ **5** -7

→ **1** 3

Определитель матрицы

5.00 -7.00

1.00 3.00

Равен 22.00

171. Написать программу, которая проверяет, является ли введенная с клавиатуры квадратная матрица магическим квадратом.

Магическим квадратом называется матрица, сумма элементов которой в каждой строке, в каждом столбце и по каждой диагонали одинакова (см. приведенный ниже рисунок).

2	9	4
7	5	3
6	1	8

13	8	12	1
2	11	7	14
3	10	6	15
16	5	9	4

172. Написать программу, которая вычисляет определитель квадратной матрицы третьего порядка.

Факультатив

173. Написать программу подведения итогов Олимпийских игр. В программу пользователь должен ввести количество медалей разного достоинства, завоеванное каждой командой-участницей, а программа — подсчитать общее число медалей и соответствующее число очков, после чего упорядочить список в соответствии с набранным количеством очков, которое определяется по следующему правилу: за золотую медаль команда получает 7 очков, за серебряную — 6, за бронзовую — 5.

Рекомендуемый вид экрана во время работы программы приведен ниже (данные, введенные пользователем, выделены полужирным шрифтом).

Итоги Олимпийских игр.

Введите в одной строке количество золотых, серебряных и бронзовых медалей.

Австрия → **3 5 9**

Германия → **12 9 8**

Канада → **6 5 4**

Китай → **0 6 2**

Корея → **3 1 2**

Норвегия → **10 10 5**

Россия → **9 6 3**

США → **6 3 4**

Финляндия → **2 4 6**

Япония → **5 1 4**

Итоги зимней Олимпиады в Нагано, 1998 г.

	Страна	Золото	Серебро	Бронза	Всего	Очков
1	Германия	12	9	8	29	178
2	Норвегия	10	10	5	25	155
3	Россия	9	6	3	18	114
4	Австрия	3	5	9	17	96
5	Канада	6	5	4	15	92
6	США	6	3	4	13	80
7	Финляндия	2	4	6	12	68
8	Япония	5	1	4	10	61

9	Китай	0	6	2	8	46
10	Корея	3	1	2	6	37

174. Написать программу игры "Угадай число". Правила игры следующие. Играют двое. Один задумывает число, второй — угадывает. На каждом шаге угадывающий делает предположение, а задумавший число — говорит, сколько цифр числа угаданы и сколько из этих цифр занимают правильные позиции в числе. Например, если задумано число 725 и выдвинуто предположение, что задумано число 523, то считаются угаданными две цифры (5 и 2) и одна из них (2) занимает верную позицию.

Ниже представлен рекомендуемый вид экрана во время работы программы. Данные, введенные пользователем, выделены полужирным шрифтом.

Компьютер задумал трехзначное число. Вы должны его отгадать. После очередного числа вам будет сообщено, сколько цифр угадано и сколько из них находится на своих местах. После ввода числа нажимайте <Enter>.

Для завершения игры нажмите <Esc>.

Ваш вариант → **123** Угадано: 0. На своих местах: 0

Ваш вариант → **456** Угадано: 1. На своих местах: 0

Ваш вариант → **654** Угадано: 2. На своих местах: 2

Ваш вариант → **657** Угадано: 2. На своих местах: 2

Ваш вариант → **658** Угадано: 3. На своих местах: 3

*** ВЫ УГАДАЛИ ! ***

Нажмите <Enter> для завершения.

175. Напишите программу-телеграф, которая принимает от пользователя сообщение и выводит его на экран в виде последовательности точек и тире. Вывод точек и тире можно сопроводить звуковым сигналом соответствующей длительности. Азбука Морзе для букв русского алфавита приведена ниже.

А	.-	Б	-...	В	.---	Г	--.
Д	-..	Е		Ж	...-	З	---..
И	..	Й	.---	К	-.-	Л	.-..
М	--	Н	-.	О	---	П	.---.

Р	.-.	С	...	Т	-	У	..-
Ф	...-	Х		Ц	-..-	Ч	----.
Ш	----	Щ	--.-	Ъ	-..-	Ы	-.---
Ь	-.--	Э	...-	Ю	..--	Я	.-.-

Функции

Приступая к решению задач этого раздела, следует вспомнить, что:

- для передачи данных в функцию надо использовать только параметры (глобальные переменные, т. е. переменные, объявленные вне функции, применять не рекомендуется);
- среди выполняемых инструкций функции обязательно должна быть инструкция присваивания значения функции;
- тип каждого фактического параметра (константы или переменной) в инструкции вызова функции должен быть таким же, как тип соответствующего формального параметра, указанного при объявлении функции;
- если в инструкции объявления функции перед именем формального параметра не стоит слово `var`, то при вызове функции можно использовать в качестве формального параметра константу или переменную соответствующего типа. Если слово `var` указано, то формальным параметром может быть только переменная;
- если параметры функции используются для возврата результата в программу, вызвавшую функцию, то в объявлении функции перед именем соответствующего параметра должно присутствовать слово `var`.

176. Написать функцию, которая вычисляет объем цилиндра. Параметрами функции должны быть радиус и высота цилиндра.

177. Написать функцию, которая возвращает максимальное из двух целых чисел, полученных в качестве аргумента.

178. Написать функцию, которая сравнивает два целых числа и возвращает результат сравнения в виде одного из знаков: >, < или =.

179. Написать функцию, которая вычисляет сопротивление цепи, состоящей из двух резисторов. Параметрами функции являются величины сопротивлений и тип соединения (последовательное или параллельное). Функция должна проверять корректность параметров: если неверно указан тип соединения, то функция должна возвращать -1.

180. Написать функцию, которая вычисляет значение a^b . Числа a и b могут быть любыми дробными положительными числами.

181. Написать функцию `Procent`, которая возвращает процент от числа, полученного в качестве аргумента.

182. Написать функцию `Dohod`, которая вычисляет доход по вкладу. Исходными данными для функции являются: величина вклада, процентная ставка (годовых) и срок вклада (количество дней).

183. Написать функцию `Glasn`, которая возвращает значение `True`, если символ, полученный функцией в качестве аргумента, является гласной буквой русского алфавита.

184. Написать функцию `LTrim`, которая удаляет начальные пробелы из строки, полученной в качестве аргумента.

185. Написать функцию `Sogl`, которая возвращает `True`, если символ, полученный функцией в качестве аргумента, является согласной буквой русского алфавита.

186. Написать функцию, которая возвращает преобразованную к верхнему регистру строку, полученную в качестве аргумента. Обратите внимание, если эту функцию назвать `UpCase` (т. е. так, как называется функция языка Turbo Pascal, решающая подобную задачу для символов латинского алфавита), то функция программиста подменит функцию Turbo Pascal.

187. Написать функцию вычисления факториала (факториалом целого числа n называется число, равное произведению целых чисел от 1 до n).

Факультатив

188. Написать функцию для решения квадратного уравнения. Параметрами функции должны быть коэффициенты и корни уравнения. Значение функции должно использоваться для передачи в вызывающую ее программу информации о наличии корней уравнения: 2 — два разных корня, 1 — корни одинаковые, 0 — уравнение не имеет решения. Кроме того, функция должна проверять корректность исходных данных. Если исходные данные неверные, то функция должна возвращать -1.

189. Написать функцию, обеспечивающую ввод с клавиатуры целого положительного двухзначного числа. При нажатии пользователем какой-либо клавиши соответствующий ей символ должен появляться на экране только в том случае, если этот символ является цифрой. Функция должна позволять пользователю редактировать введенное число при помощи клавиши <BackSpace>. Если пользователь ввел хотя бы одну цифру, то при нажатии клавиши <Enter> функция должна завершать работу и возвращать введенное пользователем число вызвавшей ее процедуре.

Процедуры

Приступая к решению задач этого раздела, следует вспомнить, что:

- для передачи информации в процедуру следует использовать параметры, а не глобальные переменные, т. е. объявленные вне процедуры;
- тип каждого фактического параметра (константы или переменной) в инструкции вызова процедуры должен соответствовать типу соответствующего формального параметра, указанного при объявлении функции;
- если в инструкции объявления процедуры перед именем формального параметра нет слова `var`, то в качестве формального параметра в инструкции вызова процедуры можно использовать константу или переменную соответствующего типа. Если слово `var` присутствует в инструкции, то формальным параметром можно назначить только переменную;

- если аргумент процедуры применяется для возврата результата в программу, вызвавшую эту процедуру, то перед именем аргумента нужно поставить слово `var`.

190. Написать процедуру, которая выводит на экран строку, состоящую из звездочек. Длина строки (количество звездочек) является параметром функции.

191. Написать процедуру, которая выводит строку, состоящую из одинаковых символов. Длина строки и символ являются параметрами процедуры.

192. Написать процедуру, которая вычисляет объем и площадь поверхности параллелепипеда.

Факультатив

193. Написать процедуру `Frame`, которая выводит на экран рамку. В качестве параметров процедуре должны передаваться координаты левого верхнего угла рамки и ее размер.

Графика

Приступая к решению задач этого раздела, следует вспомнить, что:

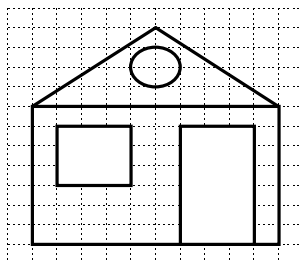
- в графическом режиме экран представляет собой совокупность точек, каждая из которых может быть окрашена в один из 16 цветов;
- координаты точек возрастают слева направо и сверху вниз; левая верхняя точка имеет координаты (0,0), а правая нижняя — (639,479);
- для того, чтобы программа могла выводить на экран графические примитивы (линии, окружности, прямоугольники), необходимо инициализировать графический режим.

Шаблон графической программы выглядит следующим образом:

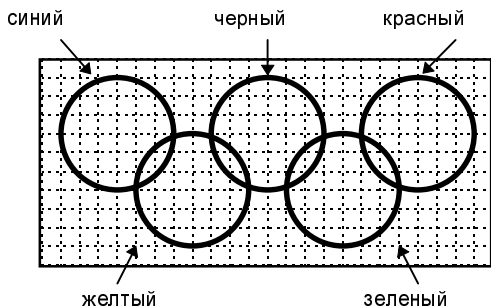
```
uses Graph;  
var  
    grDriver: Integer;
```

```
grMode: Integer;  
ErrCode: Integer;  
res: integer;  
begin  
  grDriver := detect;  
  InitGraph(grDriver, grMode, 'c:\tp\bgi');  
  ErrCode := GraphResult;  
  if ErrCode = grOk then  
    begin  
      { сюда следует поместить основные инструкции }  
    end;  
  CloseGraph;  
end.
```

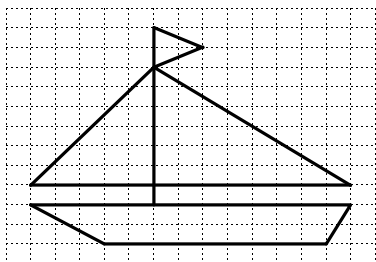
194. Написать программу, которая вычерчивает на экране домик.



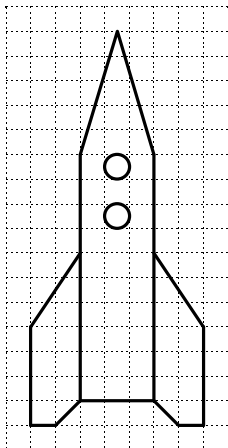
195. Написать программу, которая выводит на экран флаг Олимпийских игр. Изображение флага приведено ниже (одной клетке соответствует пять пикселей).



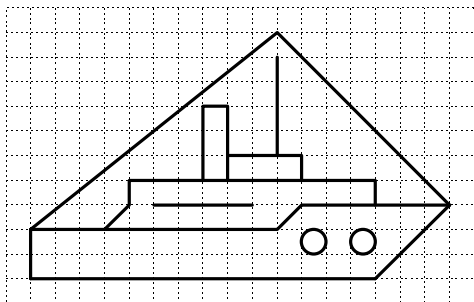
196. Написать программу, которая вычерчивает на экране кораблик.



197. Написать программу, которая вычерчивает на экране ракету.



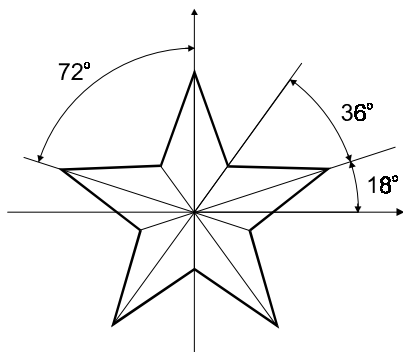
198. Написать программу, которая, используя метод базовой точки, выводит на экран изображение кораблика.



199. Написать программу, которая вычерчивает на экране узор из 100 окружностей случайного диаметра и цвета.

200. Написать программу, которая вычерчивает на экране ломанную линию, состоящую из 200 звеньев, окрашенных в разные цвета, выбираемые случайным образом, причем координаты звеньев тоже выбираются случайно.

201. Написать программу, которая выводит на экран контур пятиконечной звезды.



202. Написать программу, которая выводит на экран пятиконечную звезду.

203. Написать программу, которая вычерчивает на экране шестиугольник.

204. Написать программу, которая рисует на экране Государственный флаг России.

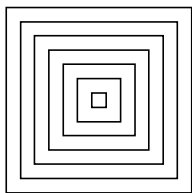
205. Написать программу, которая рисует на экране веселую рожицу желтого цвета.



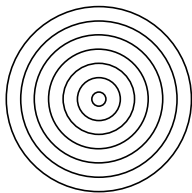
206. Написать программу, которая рисует на экране грустную рожицу.



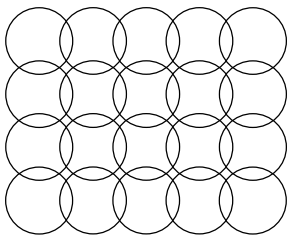
207. Написать программу, которая выводит на экран узор, изображенный ниже.



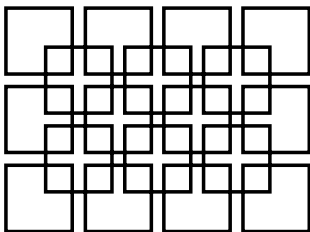
208. Написать программу, которая выводит на экран изображенный ниже узор. Окружности должны быть разного цвета (см. таблицу кодировки цветов).



209. Написать программу, которая выводит изображенный ниже узор.

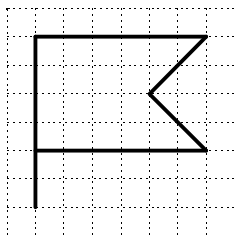


210. Написать программу, которая выводит изображенный ниже узор.

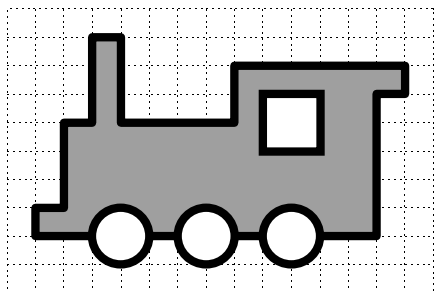


211. Написать программу, которая выводит на экран изображение шахматной доски.

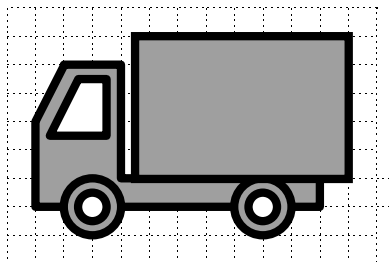
212. Написать программу, которая рисует на экране флажок красного цвета.



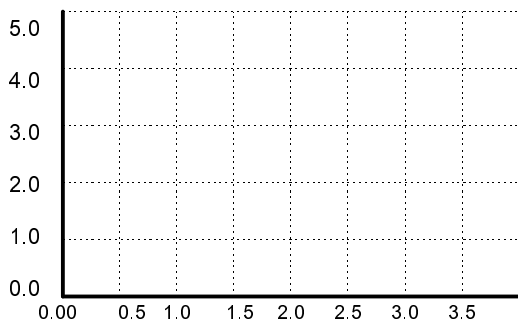
213. Написать программу, которая рисует на экране паровоз (используйте метод базовой точки).



214. Написать программу, которая рисует на экране автомобиль. Инструкции, обеспечивающие вычерчивание колеса автомобиля, оформите как процедуру.

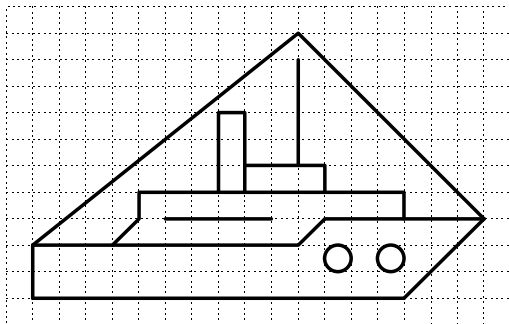


215. Написать программу, которая выводит на экран оцифрованную координатную сетку.



216. Написать программу, которая выводит на экран точечный график функции $y = 0,5x^2 + 4x - 3$. Диапазон изменения аргумента — от -15 до 5 , шаг аргумента — $0,1$. График вывести на фоне координатных осей, точка пересечения которых должна находиться в центре экрана.

217. Написать процедуру, которая рисует на экране кораблик. В качестве параметров процедура должна получать координаты базовой точки и цвет, которым следует рисовать. Используя эту процедуру, напишите программу, которая выводит на экран движущийся кораблик.



218. Написать программу, которая рисует окружность, движущуюся по экрану.

219. Написать программу, которая выводит на экран гистограмму успеваемости учеников класса, например, по итогам контрольной работы. Исходные данные следует ввести в алфавитно-цифровом режиме работы. Рекомендуемый вид экрана во время работы программы приведен ниже. На первом рисунке экран

показан во время ввода исходных данных, на втором приведен вид диаграммы на экране.

Обработка результатов контрольной работы.

Введите исходные данные:

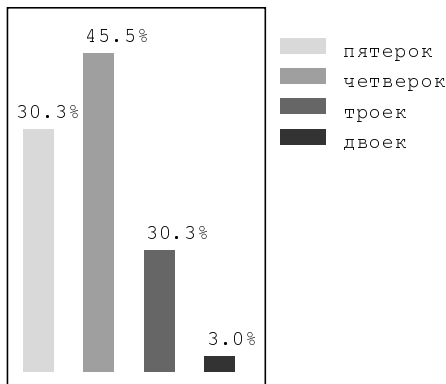
пятерок → 10

четверок → 15

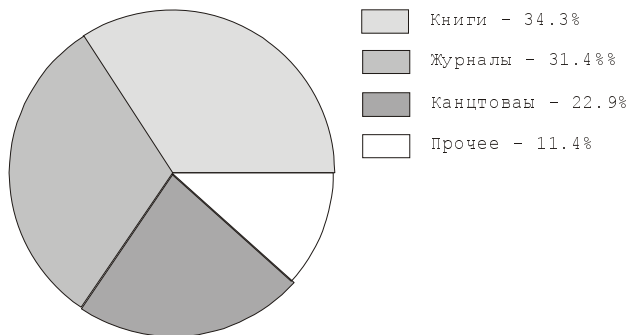
троек → 7

двоек → 1

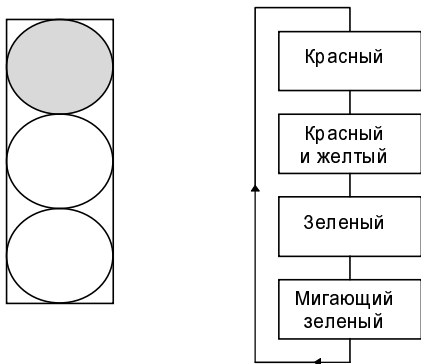
Результаты контрольной работы



220. Написать программу, которая выводит круговую диаграмму, отражающую товарооборот (в процентах) книжного магазина. Исходные данные (объем продаж в рублях по категориям: книги, журналы, открытки и канцтовары) вводятся во время работы программы. Пример диаграммы приведен ниже.



221. Написать программу, которая выводит на экран изображение работающего светофора. Рекомендуемый вид светофора и алгоритм его работы приведены ниже.



Факультатив

222. Написать программу, которая выводит на экран изображение идущих часов, имеющих секундную и минутную стрелки.

223. Написать программу, которая выводит на экран график функции $y = 2 \sin(x) e^{x/5}$.

224. Написать функцию, которая в графическом режиме вводит с клавиатуры дробное число. Набираемые пользователем символы (знак "минус", цифры и точка должны появляться на экране в зависимости от текущего положения точки вывода. Во время ожидания очередного символа на экране должен мигать курсор. Требуется, чтобы функция обеспечивала пользователю возможность простейшего редактирования — удаление последнего введенного символа (при помощи клавиши <BackSpace>).

Файлы

Приступая к решению задач этого раздела, следует вспомнить, что:

- в программе, которая выводит результаты в файл или читает исходные данные из файла, должна быть объявлена файловая переменная типа `text`;

- для доступа к конкретному файлу файловую переменную нужно связать с этим файлом (делается это при помощи инструкции `assign`);
- для того, чтобы файл был доступен, его надо открыть (для чтения с помощью инструкции `reset`, для записи — `rewrite`, для добавления — `append`);
- при работе с файлами возможны ошибки, например, из-за того, что программа пытается открыть файл, которого нет, поэтому после каждой инструкции, которая может привести к возникновению ошибки, желательно, используя функцию `IOResult`, проверять код завершения операции с файлом: чтобы программа могла контролировать результат выполнения операции с файлом, в ее текст надо поместить директиву `{SI-}`;
- запись в файл выполняют инструкции `write` и `writeln`, чтение — `read` и `readln`, причем в качестве первого параметра этих инструкций следует указывать файловую переменную;
- по завершении работы с файлом его нужно обязательно закрыть инструкцией `close`;
- файл, созданный программой, в которой тип файловой переменной объявлен как `text`, можно просмотреть при помощи редактора текста.

225. Написать программу, которая на сменном диске компьютера (A:) создает файл `numbers.txt` и записывает в него 5 введенных пользователем целых чисел. Просмотрите при помощи редактора текста, например, встроенного в Norton Commander, созданный файл. Убедитесь, что каждое число находится в отдельной строке.

226. Напишите программу, которая дописывает в файл `a:\numbers.txt` 5 введенных пользователем целых чисел. Убедитесь при помощи редактора текста, что в файле находятся 10 чисел.

227. Написать программу, которая выводит на экран содержимое файла `a:\numbers.txt`.

228. Написать программу, которая вычисляет среднее арифметическое чисел, находящихся в файле `a:\numbers.txt`.

229. Написать программу, которая позволяет просматривать текстовые файлы (выводит на экран содержимое файла), например, файлы исходных программ Turbo Pascal. Имя просматриваемого файла должно вводиться во время работы программы. Рекомендуемый вид экрана приведен ниже.

Просмотр текстового файла.

Введите полное имя файла и нажмите <Enter>

→ **c:\tp\pas\numb.pas**

230. Написать программу, которая дописывает в файл phone.txt, находящийся на диске A:, фамилию и номер телефона, например, вашего товарища. Если файла на диске нет, то программа должна создать его. В файле каждый элемент данных (имя, фамилия, телефон) должен находиться в отдельной строке. Рекомендуемый вид экрана во время работы программы приведен ниже.

Добавление в телефонный справочник.

Фамилия → **Сидоров**

Имя → **Вася**

Телефон → **234-84-37**

Информация добавлена.

Для завершения работы нажмите <Enter>.

231. Написать программу, используя которую можно за один сеанс работы добавить информацию о нескольких людях в файл a:\phone.txt.

232. Написать программу, которая позволяет найти нужные сведения в телефонном справочнике (a:\phone.txt). Программа должна запрашивать фамилию человека и выводить его телефон. Если в справочнике есть одинаковые фамилии, то программа должна вывести список всех людей, имеющих эти фамилии. Рекомендуемый вид экрана во время работы программы приведен ниже.

Поиск в телефонном справочнике.

Введите фамилию и нажмите <Enter>. Для завершения работы с программой сразу после приглашения нажмите <Enter>

→ **Петров**

В справочнике данных о Петров нет.

—> **Иванов**

Иванов Вася 578-12-45

Иванов Сергей 244-34-02

—>

233. Напишите программу, которая объединяет возможности программ Добавление в телефонный справочник и Поиск в телефонном справочнике. При запуске программы на экран должно выводиться меню:

*** Телефонный справочник ***

1. Добавление

2. Поиск

3. Завершение работы

Введите номер пункта меню и нажмите <Enter>

—>

Факультатив

234. Написать универсальную программу тестирования. Тест, последовательность вопросов и варианты ответов должны находиться в текстовом файле. Имя этого файла программа должна получать из командной строки ее запуска. Количество вопросов теста не ограничено. Однако предлагается ввести следующее ограничение: текст вопроса и альтернативных ответов не должен занимать более одной строки экрана.

Программа должна выставлять оценку по следующему правилу: **ОТЛИЧНО** — за правильные ответы на все вопросы, **ХОРОШО** — если испытуемый правильно ответил не менее, чем на 80% вопросов, **УДОВЛЕТВОРИТЕЛЬНО** — если правильных ответов более 60%, и **ПЛОХО** — если правильных ответов меньше 60%.

Ниже приведены рекомендуемая структура файла вопросов теста (N_i — количество альтернативных ответов к i -му вопросу, K_i — номер правильного ответа), пример файла теста и вид экрана во время работы программы (номера ответов, введенные пользователем, выделены полужирным шрифтом).

Вопрос₁

N_1 M_1

Ответ

...

Ответ

Вопрос₂

N₂ M₂

Ответ

...

Ответ

Вопрос_k

N_k M_k

Ответ

...

Ответ

Архитектор Исаакиевского собора

3 2

Доменико Трезини

Огюст Монферран

Карл Росси

Архитектор Зимнего дворца

2 2

Франческо Бартоломео

Огюст Монферран

Невский проспект получил свое название

3 2

по имени реки, на которой стоит Санкт-Петербург

по имени близкой расположенной Александро-Невской лавры

в память о знаменитом полководце — Александре Невском

Сейчас Вам будет предложен тест.

К каждому вопросу дается несколько вариантов ответа.

Вы должны ввести номер правильного ответа и нажать <Enter>.

Архитектор Исаакиевского собора:

1. Доменико Трезини

2. Огюст Монферран

3. Карл Росси

→ 2

Архитектор Зимнего дворца:

1. Франческо Бартоломео

2. Карл Росси

→ 2

Невский проспект получил свое название:

1. По имени реки, на которой стоит Санкт-Петербург.

2. По имени близко расположенной Александро-Невской лавры.
 3. В память о знаменитом полководце — Александре Невском.
- 2

Ваша оценка ОТЛИЧНО!

Для завершения работы программы нажмите <Enter>.

235. Напишите программу, которая по желанию пользователя выводит таблицу пересчета из дюймов в миллиметры на экран, принтер или в файл. Ниже представлен рекомендуемый вид экрана во время работы программы (данные, введенные пользователем, выделены полужирным).

*** Таблица пересчета из дюймов в миллиметры ***

Результат вывести:

- 1 — на экран;
- 2 — на принтер;
- 3 — в файл.

Введите число от 1 до 3 и нажмите <Enter>.

Ваш выбор → **2**

Дюймы	Миллиметры
-------	------------

0.5	12.7
1.0	25.4
1.5	38.1
2.0	50.8
2.5	63.5
3.0	76.2
3.5	88.9
4.0	101.6
4.5	114.3
5.0	127.0

Рекурсия

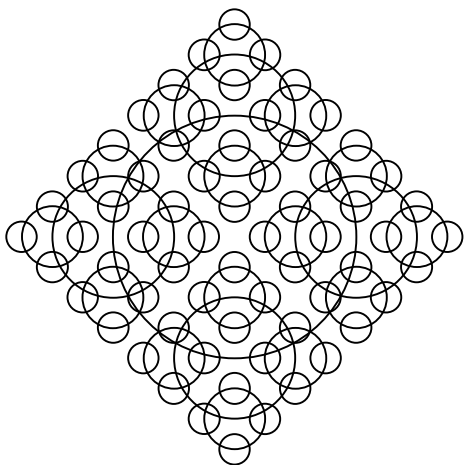
Приступая к решению задач этого раздела, следует вспомнить, что:

- рекурсивной называется такая процедура или функция, которая может вызывать сама себя;

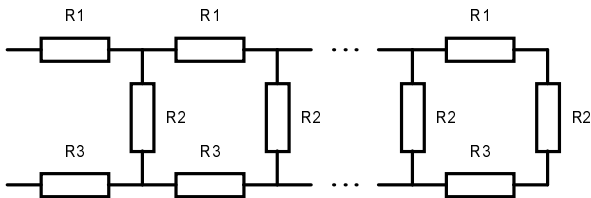
- для завершения процесса рекурсии в алгоритме рекурсивной функции (процедуры) обязательно должна быть веточка, обеспечивающая непосредственное завершение функции (процедуры).

236. Написать рекурсивную функцию вычисления факториала и программу, проверяющую ее работоспособность.

237. Написать программу, которая выводит на экран приведенный ниже узор.



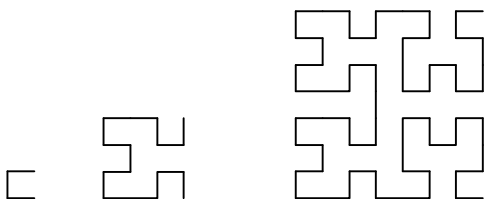
238. Написать программу, которая вычисляет сопротивление электрической цепи, схема которой приведена ниже. Величины сопротивлений и порядок цепи (количество сопротивлений R2) должны вводиться во время работы программы.



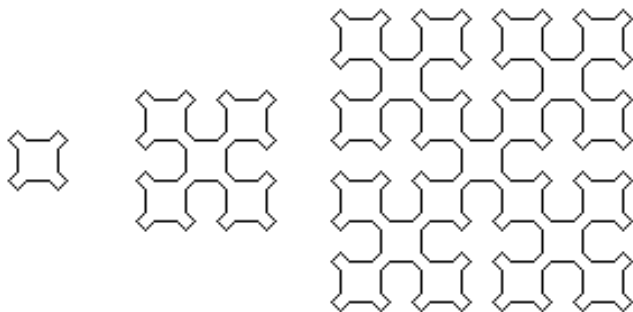
239. Написать программу, которая вычерчивает на экране приведенную выше схему электрической цепи. Порядок цепи должен вводиться во время работы программы.

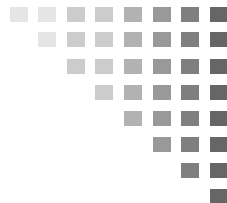
Факультатив

240. Написать программу, которая вычерчивает на экране кривую Гильберта, показанную ниже. Обратите внимание, что кривая второго порядка образуется путем соединения четырех кривых первого порядка, две из которых повернуты на 90° : одна по часовой стрелке, а другая — против. Аналогичным образом получается кривая третьего порядка, но при этом в качестве "кирпичиков" используются кривые второго порядка. Порядок вычерчивания кривой должен вводиться во время работы программы.



241. Написать программу, которая вычерчивает кривую Серпинского. Порядок кривой должен задаваться во время работы программы. Вид кривых Серпинского первого, второго и третьего порядков приведен ниже.





ЧАСТЬ II. РЕШЕНИЯ

Задача 1

```
x: real; { аргумент функции }  
y: real; { значение функции }
```

Задача 2

```
funt: real; { вес в фунтах }  
kg: real;   { вес в килограммах }
```

Задача 5

```
r1: real; { диаметр кольца }  
r2: real; { диаметр отверстия }  
s: real;  { площадь кольца }
```

Задача 7

```
CenaTetr: real;   { цена тетради }  
KolTetr: integer; { количество тетрадей }  
CenaKar: real;    { цена карандаша }  
KolKar: integer;  { количество карандашей }  
CenaLin: real;    { цена линейки }  
Summa: real;      { стоимость покупки }
```

Задача 11

```
n:=n+1;
```

Задача 14

```
y:=-2.7*x*x*x+0.23*x*x-1.4;
```

Задача 16

```
kg:=gunt*0.4095;
```

Задача 34

```
sum:=ct*nt+ck*nk+cl; {ct,ck —цена тетради, карандаша и линейки }  
      { nt, nk —количество тетрадей  
      и карандашей}
```

Задача 38

```
begin
```

```
  writeln('Унылая пора! Очей очарование!');  
  writeln('Приятна мне твоя прощальная пора.');
```

writeln('Люблю я пышное природы увяданье,');

```
  writeln('В багрец и золото одетые леса.');
```

writeln;

```
  writeln('                А.С.Пушкин');
```

readln; { чтобы стихотворение не исчезло с экрана }

```
end.
```

Задача 42

```
writeln(a:6:2);  
writeln(b:6:2);  
writeln(c:6:2);
```

Задача 43

```
writeln('x1=',x1:6:2,' x2=',x2:6:2);
```

Задача 44

```
uses Crt;
```

```
begin
```

```
  TextBackGround(Blue); { цвет фона }  
  TextColor(LightGray); { цвет символов }  
  ClrScr;                { очистить экран }  
  writeln('Буря мглою небо кроет');  
  writeln('Вихри снежные крутя.');
```

writeln('То как зверь она завоет,');

```
  writeln('То заплачет, как дитя.');
```

```
writeln;  
writeln('          А.С.Пушкин');  
readln; { чтобы стихотворение не исчезло с экрана }  
end.
```

Задача 45

```
Uses Crt;  
begin  
  TextBackGround(Black);  
  ClrScr;  
  TextColor(Red);  
  write('Каждый ');  
  TextColor(LightRed); {оранжевый заменим светло-красным}  
  write('охотник ');  
  TextColor(Yellow);  
  write('желает ');  
  TextColor(Green);  
  write('знать ');  
  TextColor(LightBlue);  
  write('где ');  
  TextColor(Blue);  
  write('сидит ');  
  TextColor(Magenta);  
  write('фазан. ');  
  readln; { чтобы фраза не исчезала с экрана }  
end.
```

Задача 47

```
readln(u);  
readln(r);
```

Задача 48

```
readln(u, r);
```

Задача 49

```
{ объявление переменных }  
r, h : real; {радиус и высота цилиндра}  
v: real; { объем цилиндра}
```

```
{ фрагмент программы }  
writeln('Введите исходные данные:');  
write('Радиус цилиндра ->');  
readln ( r );  
writeln('Высота цилиндра ->');  
readln(h);
```

Задача 51

```
{ Вычисление площади параллелограмма }  
var  
  l,w: real; { длина и ширина параллелограмма}  
  s: real;    { площадь параллелограмма}  
begin  
  writeln('Вычисление площади параллелограмма');  
  writeln('Введите исходные данные:');  
  write('Длина (см.)  -> ');  
  readln(l);  
  write('Ширина (см.) ->');  
  readln(w);  
  s:=l*w;  
  writeln('Площадь параллелограмма:',s:6:2,' кв.см.');
```

end.

Задача 53

```
{ Вычисление площади поверхности параллелепипеда }  
var  
  l,w,h : real; { длина, ширина и высота параллелепипеда}  
  s: real;      { площадь поверхности параллелепипеда}  
begin  
  writeln('Вычисление площади поверхности параллелепипеда');  
  writeln('Введите исходные данные:');  
  write('Длина (см)  -> ');  
  readln(l);  
  write('Ширина (см) ->');  
  readln(w);  
  write('Высота (см) ->');  
  readln(h);  
  s:=(l*w + l*h + w*h)*2;
```

```
writeln('Площадь поверхности параллелепипеда:',s:6:2,  
        ' кв.см.');
```

```
readln;
```

```
end.
```

Задача 55

```
{ Вычисление объема цилиндра }
```

```
var
```

```
  r,h,v: real; { радиус основания, высота и объем цилиндра }
```

```
begin
```

```
  writeln('Вычисление объема цилиндра');
```

```
  writeln('Введите исходные данные:');
```

```
  write('радиус основания (см) -> ');
```

```
  readln(r);
```

```
  write('высота цилиндра (см) -> ');
```

```
  readln(h);
```

```
  v := 2*3.1415926*r*r*h;
```

```
  writeln('Объем цилиндра ',v:6:2,' см.куб.');
```

```
  writeln('Для завершения работы программы нажмите <Enter>');
```

```
  readln;
```

```
end.
```

Задача 56

```
{ Вычисление стоимости покупки }
```

```
var
```

```
  kar,tetr: real; { цена карандаша и тетради }
```

```
  nk,nt: integer; { количество тетрадей и карандашей }
```

```
  summ: real;     { стоимость покупки }
```

```
begin
```

```
  writeln('Вычисление стоимости покупки.');
```

```
  writeln('Введите исходные данные:');
```

```
  write('Цена тетради (руб.) ->');
```

```
  readln(tetr);
```

```
  write('Количество тетрадей ->');
```

```
  readln(nt);
```

```
  write('Цена карандаша (руб.) ->');
```

```
  readln(kar);
```

```
  write('Количество карандашей ->');
```

```
  readln(nk);
```

```
summ:=tetr*nt + kar*nk;  
writeln;  
writeln('Стоимость покупки:',summ:6:2,' руб.');
```

readln;

end.

Задача 60

```
{ Вычисление площади треугольника по двум  
  сторонам и углу между ними }  
var  
  a,b: real; { длины сторон }  
  f: real;   { величина угла, выраженная в градусах }  
  s: real;   { площадь треугольника }  
begin  
  writeln('Вычисление площади треугольника.');
```

writeln('Введите в одной строке длины сторон треугольника');

write('->');

readln(a,b);

writeln('Введите величину угла между сторонами ',
 'треугольника');

write('->');

readln(f);

{ $s = a \cdot h / 2$

h (высота треугольника) может быть вычислена
по формуле: $h = b \cdot \sin(f)$.

Однако, в Turbo Pascal аргумент функции Sin должен быть
выражен в радианах (1 рад. = 180/3.1415925,
где 3.1415926 – число "ПИ").}

$s := a \cdot b \cdot \sin(f \cdot 3.1415926 / 180) / 2$;

writeln;

writeln('Площадь треугольника:',s:6:2,' кв.см.');

readln;

end.

Задача 61

```
{ Вычисление сопротивления электрической цепи,  
  состоящей из двух параллельно соединенных элементов. }  
var  
  r1,r2: real; { сопротивление элементов цепи }  
  r: real;     { суммарное сопротивление цепи }
```

begin

```
writeln('Вычисление сопротивления электрической цепи');
writeln('при параллельном соединении элементов.');
```

Введите исходные данные:

```
write('Величина первого сопротивления (Ом) ->');
readln(r1);
write('Величина второго сопротивления (Ом) ->');
readln(r2);
r:=r1*r2/(r1+r2);
writeln;
writeln('Сопротивление цепи:',r:6:2,' Ом');
```

end.

Задача 65

{ Вычисление стоимости поездки на дачу и обратно }

var

```
rast: real; { расстояние до дачи }
rash: real; { расход бензина на 100 км пути }
cena: real; { цена одного литра бензина }
summ: real; { стоимость поездки на дачу и обратно }
```

begin

```
writeln('Вычисление стоимости поездки на дачу и обратно.');
```

Расстояние до дачи (км) ->

```
readln(rast);
write('Расход бензина (литров на 100 км) ->');
readln(rash);
write('Цена литра бензина (руб.) ->');
readln(cena);
summ:=2*rast/100*rash*cena;
writeln;
writeln('Поездка на дачу и обратно обойдется в',
        summ:6:2,' руб.');
```

end.

Задача 67

{ Вычисление площади поверхности цилиндра }

var

```
r: real; { радиус основания цилиндра }
```

```
h: real; { высота цилиндра }
s: real; { площадь поверхности цилиндра }
begin
  writeln('Вычисление площади поверхности цилиндра');
  writeln('Введите исходные данные:');
  write('радиус основания цилиндра (см) ->');
  readln(r);
  write('высота цилиндра (см) ->');
  readln(h);
  {
    pi — именованная константа, ее значение равно числу "ПИ"
    Sqr — функция Turbo Pascal, ее значение равно квадрату
    аргумента, т. е.  $Sqr(x) = x * x$ 
  }
  s:=2*pi*sqr(r)+2*pi*r*h;
  writeln;
  writeln('Площадь поверхности цилиндра ',s:6:2,'кв.см. ');
  readln;
end.
```

Задача 69

```
{ Пересчет расстояния из верст в километры }
var
  v: real; { расстояние в верстах }
  k: real; { расстояние в километрах }
begin
  writeln('Пересчет расстояния из верст в километры');
  writeln('Введите расстояние в верстах и нажмите <Enter>');
  write('->');
  readln(v);
  k:=v*1.0668;
  writeln(v:6:2,' верст(а/ы) — это ',k:6:2,' км');
  writeln('Для завершения работы программы нажмите ',
    '<Enter>');
  readln;
end.
```

Задача 71

```
{ Вычисление дохода по вкладу }
var
  sum: real; { сумма вклада }
```

```
srok:real;    { срок вклада }
stavka:real; { процентная ставка }
dohod:real;  { доход по вкладу }

begin
  writeln('Вычисление дохода по вкладу. ');
  writeln('Введите исходные данные: ');
  write('Величина вклада (руб.) -> ');
  readln(sum);
  write('Срок вклада (дней) -> ');
  readln(srok);
  write('Процентная ставка -> ');
  readln(stavka);
  dohod:=(sum*stavka/100)/365*srok; { 365 — кол-во дней
                                   в году }
  sum:=sum+dohod;
  writeln;
  writeln('-----');
  writeln('Доход:',dohod:9:2,' руб. ');
  writeln('Сумма по окончании срока вклада:',sum:9:2,
          ' руб. ');
  readln;
end.
```

Задача 72

```
{ Преобразование величины, выраженной в минутах,
  в соответствующее ей значение, выраженное в часах и минутах }

var
  min: integer; { интервал в минутах }
  h: integer;   { количество часов }
  m: integer;   { количество минут }

begin
  writeln('Введите величину временного интервала (в минутах) ');
  write('-> ');
  readln(min);
  h:= min div 60;
  m:= min mod 60;
  writeln;
  writeln(min,' мин. — это ',h,' час.',m,' мин. ');
  readln;
end.
```

Задача 73

```
{ Преобразование числа в денежный формат }  
var  
  n: real;      { дробное число }  
  r: integer; { целая часть числа (рубли) }  
  k: integer; { дробная часть числа (копейки) }  
begin  
  writeln('Преобразование числа в денежный формат. ');  
  write('Введите дробное число ->');  
  readln(n);  
  r:= Round(n*100) div 100;  
  k:= Round(n*100) mod 100;  
  writeln;  
  writeln(n:6:2, ' руб. — это ', r, ' руб.', k, ' коп. ');  
  readln;  
end.
```

Задача 76

```
{ Вычисление частного }  
var  
  a,b,c: real; { делимое, делитель и частное }  
begin  
  writeln('Вычисление частного');  
  writeln('Введите в одной строке делимое и делитель, ');  
  writeln('затем нажмите <Enter>');  
  write('->');  
  readln(a,b);  
  if b <> 0 then  
    begin  
      c:=a/b;  
      writeln('частное от деления ', a:6:3, ' на ', b:6:3,  
        ' равно ', c:6:3);  
    end  
  else  
    writeln('Ошибка! Делитель не должен быть ',  
      ' равен нулю!');  
  readln;  
end.
```

Задача 77

```
{ Вычисление площади кольца }  
  
var  
    r1,r2 : real; { радиус кольца и отверстия }  
    s : real;      { площадь кольца }  
  
begin  
    writeln('Введите исходные данные:');  
    write('радиус кольца (см) -> ');  
    readln(r1);  
    write('радиус отверстия (см) -> ');  
    readln(r2);  
    if r1 > r2 then  
        begin  
            s:=2*3.14*(r1-r2);  
            writeln('Площадь кольца ',s:6:2,' кв.см');  
        end  
    else writeln('Ошибка! Радиус отверстия не может быть ',  
                'больше радиуса кольца.');    writeln('Для завершения работы программы нажмите <Enter>');  
    readln;  
end.
```

Задача 78

```
{ Вычисление сопротивления электрической цепи }  
  
var  
    r1,r2: real; { величины сопротивлений цепи }  
    r: real;     { суммарное сопротивление }  
    t: integer;  { тип соединения элементов:  
                  1 — последовательное;  
                  2 — параллельное }  
  
begin  
    writeln('Вычисление сопротивления электрической цепи');  
    writeln('Введите исходные данные:');  
    write('Величина первого сопротивления (Ом) ->');  
    readln(r1);  
    write('Величина второго сопротивления (Ом) ->');  
    readln(r2);  
    write('Тип соединения элементов (1-последовательное, ',  
          '2-параллельное) ->');  
    readln(t);
```

```
if t=1
then r:=r1+r2
else r:=r1*r2/(r1+r2);
writeln('Соппротивление цепи: ',r:5:2,' Ом');
readln;
end.
```

Задача 79

```
{ Решение квадратного уравнения }
program sqroot;
var
  a,b,c:real; { коэффициенты уравнения }
  x1,x2:real; { корни уравнения }
  d:real;      { дискриминант }
begin
  writeln('* Решение квадратного уравнения *');
  write('Введите в одной строке значения коэффициентов');
  writeln(' и нажмите <Enter>');
  write('->');
  readln(a,b,c);      { ввод коэффициентов }
  d:=b*b-4*a*c;        { вычисление дискриминанта }
  if d >= 0
  then
    begin
      x1:=-b+sqrt(d)/(2*a);
      x2:=-b-sqrt(d)/(2*a);
      writeln('Корни уравнения:');
      writeln('x1=',x1:9:3);
      writeln('x2=',x2:9:3);
    end
  else
    writeln('Корней нет');
end.
```

Задача 81

```
{ Вычисление стоимости покупки с учетом скидки }
var
  summ: real; { сумма покупки }
```

begin

```
writeln('Вычисление стоимости покупки с учетом скидки.');
```

```
writeln('Введите стоимость покупки и нажмите <Enter>.');
```

```
write('->');
```

```
readln(summ);
```

```
if summ > 500
```

```
    then { предоставляется скидка }
```

```
        begin
```

```
            if summ > 1000
```

```
                then
```

```
                    begin
```

```
                        writeln('Вам предоставляется скидка 5%');
```

```
                        summ:=0.95*summ;
```

```
                    end
```

```
                else
```

```
                    begin
```

```
                        writeln('Вам предоставляется скидка 3%');
```

```
                        summ:=0.97*summ;
```

```
                    end;
```

```
                writeln('Сумма покупки с учетом скидки: ',
```

```
                        summ:6:2, ' руб.');
```

```
            end
```

```
        else
```

```
            writeln('Скидка не предоставляется.');
```

```
        readln;
```

```
end.
```

Задача 82

```
{ Проверка знания истории }
```

var

```
year: integer;    { ответ испытуемого }
```

begin

```
writeln('В каком году был основан Санкт-Петербург?');
```

```
writeln('Введите число и нажмите <Enter>');
```

```
write('->');
```

```
readln(year);
```

```
if year = 1703
```

```
    then writeln('Правильно.')
```

```
    else
```

```
begin
  write('Вы ошиблись, ');
  writeln('Санкт-Петербург был основан в 1703 году.')
end;
readln;
end.
```

Задача 84

```
{ Проверка знания истории архитектуры }
var
  otv: integer; { номер ответа, выбранного испытуемым }
begin
  writeln('Архитектор Исаакиевского собора:');
  writeln('1. Доменико Трезини');
  writeln('2. Огюст Монферран');
  writeln('3. Карл Росси');
  writeln;
  writeln('Введите номер правильного ответа и нажмите <Enter>');
  write('->');
  readln(otv);
  if otv = 2
  then writeln('Правильно.')
  else
    begin
      writeln('Вы ошиблись. ');
      writeln('Архитектор Исаакиевского собора ',
        'Огюст Монферран. ');
    end;
  readln;
end.
```

Задача 86

```
{ Сравнение двух целых чисел }
var
  a,b: integer; { сравниваемые числа }
begin
  write('Введите в одной строке два целых ');
  writeln('числа и нажмите <Enter>');
  write('->');
```

```
readln(a,b);
if a = b
  then writeln('числа равны')
  else if a < b
    then writeln(a, ' меньше ',b)
    else writeln(a, ' больше ',b);
readln;
end.
```

Задача 87

```
{ Проверка умения умножать числа }
var
  m1,m2,p: integer; { сомножители и произведение }
  otv: integer;      { ответ испытуемого }
begin
  Randomize;         { инициализация генератора случайных
                      чисел }
  m1:=Random(9)+1;    { значение функции Random(a) — число }
                      { в диапазоне от 0 до a-1 }
  m2:=Random(9)+1;
  p:=m1*m2;
  writeln('Сколько будет ',m1,'x',m2,' ?');
  writeln('Введите ответ и нажмите <Enter>');
  write('-> ');
  readln(otv);
  if p = otv
    then writeln('Правильно.')
    else writeln('Вы ошиблись. ',m1,'x',m2,'=',p);
  readln;
end.
```

Задача 89

```
{ Проверяет на четность введенное с клавиатуры число }
var
  n: integer; { введенное пользователем число }
begin
  writeln('Введите целое число и нажмите <Enter>');
  write('->');
  readln(n);
  write('Число ',n,' - ');
```

```
if n mod 2 = 0
  then writeln('четное.')
  else writeln('нечетное.');
```

readln; { чтобы результат не исчез с экрана }

end.

Задача 91

```
{ Вычисление стоимости телефонного разговора с учетом
скидки, предоставляемой по субботам и воскресеньям }
```

var

Time:integer; { длительность разговора }

Day:integer; { день недели }

Summa:real; { стоимость разговора }

begin

writeln('Вычисление стоимости разговора по телефону.');

writeln('Введите исходные данные:');

write('Длительность разговора');

write(' (целое кол-во минут) ->');

readln(Time);

write('День недели');

write(' (1-понедельник, ..., 7-воскресенье) -> ');

readln(Day);

Summa:= 2.3 * Time; { цена минуты 2.3 руб. }

if (Day = 6) or (Day = 7)

then begin

writeln('Предоставляется скидка 20%');

Summa:=Summa * 0.8;

end;

writeln('Стоимость разговора:', Summa:8:2, ' руб.');

readln;

end.

Задача 92

```
{ Контроль веса }
```

var

w:real; { вес }

h:real; { рост }

```
    opt:real; { оптимальный вес }
    d:real;   { отклонение от оптимального веса }

begin
    writeln('Введите в одной строке через пробел');
    writeln('рост (см) и вес (кг), затем нажмите <Enter>');
    write('->');
    readln(h,w);
    opt:=h-100;
    if w=opt
    then
        writeln('Ваш вес оптимален!')
    else
        if w<opt
        then begin
            d:=opt-w;
            writeln('Вам надо поправиться на ',
                    d:5:2,' кг.');
        end
        else begin
            d:=w-opt;
            writeln('Вам надо похудеть на ',
                    d:5:2,' кг');
        end;

    readln;
end.
```

Задача 93

```
{ Определение времени года по номеру месяца }

var
    month: integer; { номер месяца }

begin
    writeln('Введите номер месяца (число от 1 до 12)
            и нажмите <Enter>');
    write('->');
    readln(month);
    case month of
        1,2,12: writeln('Зима');
        3..5:   writeln('Весна');
        6..8:   writeln('Лето');
        9..11:  writeln('Осень');
```

```
    else      writeln('Число должно быть от 1 до 12');  
end;  
readln;  
end.
```

Задача 95

```
{ Определение стоимости междугороднего  
  телефонного разговора }  
var  
  kod: integer; { код города }  
  cena: real;    { цена минуты }  
  dlit: integer; { длительность разговора }  
  summ: real;    { стоимость разговора }  
begin  
  writeln('Вычисление стоимости разговора по телефону.');  writeln('Введите исходные данные:');  
  write('Код города ->');  
  readln(kod);  
  write('Длительность разговора (целое кол-во минут) ->');  
  readln(dlit);  
  write('Город: ');  
  case kod of  
    432: begin  
      writeln('Владивосток');  
      cena:=2.2;  
    end;  
    095: begin  
      writeln('Москва');  
      cena:=1;  
    end;  
    815: begin  
      writeln('Мурманск');  
      cena:=1.2;  
    end;  
    846: begin  
      writeln('Сапара');  
      cena:=1.4;  
    end;  
  end;  
  summ := cena * dlit;  
  writeln('Цена минуты:',cena:6:2,' руб.');
```

```
writeln('Стоимость разговора:',summ:6:2,' руб. ');
readln;
end.
```

Задача 96

```
{ Дописывает после числа слово "рубль" в правильной форме }
var
  n: integer; { число }
  m: integer; { остаток от деления n на 100 (последние две
               цифры) }
  r: integer; { остаток от деления n на 10 (последняя цифра) }
begin
  writeln('Введите целое число (от 1 до 999)',
          ' и нажмите <Enter>');
  write('-> ');
  readln(n);

  write(n, ' ');
  { окончание определяется двумя последними цифрами }
  if n>100
    then m := n mod 100;

  if (m >= 11) and (m <= 14)
    then
      writeln('рублей')
    else begin
      r:=n mod 10;
      case r of
        0, 5 .. 9: writeln('рублей');
        1:      writeln ('рубль');
        2..4:   writeln ('рубля');
      end;
    end;
  readln;
end.
```

Задача 98

```
{ Вычисление даты следующего дня }
var
  day:integer;
```

```

month:integer;
year:integer;
last:boolean;{ TRUE, если текущий день последний день
               месяца }
r:integer;   { если год високосный, то остаток от }
               { деления year на 4 равен нулю }

```

begin

```

write('Введите цифрами сегодняшнюю дату');
write('(число месяц год) -> ');
readln(day,month,year);
last:=FALSE;
case month of
    1,3,5,7,8,10,12:if day=31
                        then last:=TRUE;

    4,6,9,11:          if day=30
                        then last:=TRUE;

    2:                 if day = 28 then
                        begin
                            r:=year mod 4;
                            if r <> 0
                                then last:=TRUE;
                        end;
end; { case }

```

if last

then begin

```

    writeln('Последний день месяца!');
    day:=1;

```

if month=12

then begin

```

    month:=1;

```

```

    year:=year + 1;

```

```

    writeln('С наступающим Новым годом!');

```

end

```

    else month:=month + 1;

```

end

else day:=day + 1;

```

writeln('Завтра ',day,'.',month,'.',year );

```

```
        readln;
end.
```

Задача 100

```
{ Выводит таблицу квадратов первых десяти
  целых положительных чисел }
var
  x: integer; { число }
  y: integer; { квадрат числа }
  i: integer; { счетчик циклов }
```

```
begin
```

```
  writeln('Таблица квадратов');
  writeln('-----');
  writeln('Число      Квадрат');
  writeln('-----');
```

```
  for i:=1 to 10 do
```

```
    begin
```

```
      y:=x*x;
      writeln(x:3,y:10);
      x:=x+1;
```

```
    end;
```

```
  writeln('-----');
  readln;
```

```
end.
```

Задача 102

```
{ Вычисляет сумму первых n целых положительных чисел }
var
  n: integer;      { кол-во суммируемых чисел }
  summ: integer;   { сумма }
  i: integer;      { счетчик циклов }
```

```
begin
```

```
  writeln('Вычисление суммы положительных чисел');
  write('Введите количество суммируемых чисел ->');
  readln(n);
  summ:=0;
```

```
  for i:=1 to n do
```

```
    summ:=summ+i;
```

```
  write('Сумма первых ',n,' целых положительных чисел ');
```

```
writeln('равна ',summ);  
readln;  
end.
```

Задача 104

```
{ Вычисляет частичную сумму ряда: 1,3,5,7 ... }  
var  
  e: integer;      { член ряда }  
  n: integer;      { кол-во суммируемых членов }  
  summ: integer;   { сумма членов ряда }  
  i: integer;      { счетчик циклов }  
begin  
  writeln('Вычисление частичной суммы ряда: 1,3,5,7, ...');  
  writeln('Введите количество суммируемых членов ->');  
  readln(n);  
  e:=1; { первый член ряда }  
  summ:=0;  
  for i:=1 to n do  
    begin  
      summ:=summ+e; { прибавить к сумме очередной член ряда }  
      e:=e+2;       { вычислить значение следующего члена  
ряда }  
    end;  
  writeln('Сумма первых ',n,' членов ряда равна ',summ);  
  readln;  
end.
```

Задача 105

```
{ Вычисление суммы ряда 1+1/2+1/3+ ... }  
var  
  n: integer;      { кол-во суммируемых членов ряда }  
  i: integer;      { номер элемента ряда }  
  elem: real;      { значение элемента ряда }  
  summ: real;      { сумма элементов ряда }  
begin  
  writeln('Вычисление частичной суммы ряда  
1+1/2+1/3+...');  
  writeln('Введите кол-во суммируемых членов ряда');  
  write('->');
```

```
readln(n);
summ:=0;
for i:=1 to n do
begin
    elem:=1/i;
    summ:=summ+elem;
end;
write('Сумма первых ',n);
writeln(' членов ряда равна ',summ:6:4);
readln;
```

end.

Задача 106

```
{ Таблица степеней двойки }
var
    n: integer; { показатель степени }
    x: integer; { значение 2 в степени n }
begin
    writeln('Таблица степеней двойки');
    x:=1;
    for n:=0 to 10 do
        begin
            writeln(n:2,x:6);
            x:=x*2;
        end;
    readln;
end.
```

Задача 108

```
{ Таблица функции }
const
    LB=-2; { нижняя граница диапазона изменения аргумента }
    HB=2; { верхняя граница диапазона изменения аргумента }
    DX=0.5; { приращение аргумента }
var
    x,y: real; { аргумент и значение функции }
    n: integer; { кол-во точек }
    i: integer; { счетчик циклов }
```

```

begin
  n:=Round((HB - LB)/DX)+1;
  x:=LB;
  writeln('-----');
  writeln('      x      |      y');
  writeln('-----');
  for i:=1 to n do
    begin
      y:=-2.4*x*x+5*x-3;
      writeln(x:8:2,' | ',y:8:2);
      x:=x+DX;
    end;
  writeln('-----');
  readln;
end.

```

Задача 111

{ Вычисляет среднее арифметическое последовательности дробных чисел, вводимой с клавиатуры }

const

L=5; { длина последовательности }

var

a: real; { число }

n: integer; { кол-во введенных чисел }

sum: real; { сумма введенных чисел }

sred: real; { среднее арифметическое введенных чисел }

begin

writeln('Обработка последовательности дробных чисел');

writeln('После ввода каждого числа нажимайте <Enter>');

sum:=0;

for n:=1 **to** L **do**

begin

write('->');

readln(a);

sum:=sum+a;

sred:=sum/n;

writeln('Введено чисел:',n:3,

' Сумма: ', sum:6:2,

' Сред.арифметическое:',sred:6:2);

end;

```
writeln('Для завершения работы программы нажмите <Enter>');  
readln;  
end.
```

Задача 112

```
{ Вычисляет среднее арифметическое и определяет  
минимальное и максимальное число последовательности  
дробных чисел, вводимых с клавиатуры }  
  
var  
  a: real;      { очередное число }  
  n: integer;   { количество чисел }  
  sum: real;    { сумма введенных чисел }  
  sred: real;   { среднее арифметическое }  
  min: real;    { минимальное число последовательности }  
  max: real;    { максимальное число последовательности }  
  i: integer;   { счетчик циклов }  
  
begin  
  writeln('Обработка последовательности дробных чисел.');  write ('Введите количество чисел последовательности ->');  
  readln(n);  
  writeln('Вводите последовательность.');  writeln('После ввода каждого числа нажимайте <Enter>');  write('->');  
  readln(a);      { вводим первое число последовательности }  
  { предположим, что: }  
  min:=a; { первое число является минимальным }  
  max:=a; { первое число является максимальным }  
  sum:=a;  
  { введем остальные числа }  
  for i:=1 to n-1 do  
    begin  
      write('->');  
      readln(a);  
      sum:=sum+a;  
      if a < min then min:=a;  
      if a > max then max:=a;  
    end;  
  sred:=sum/n;
```

```
writeln('Количество чисел:',n);
writeln('Среднее арифметическое:',sred:6:2);
writeln('Минимальное число:',min:6:2);
writeln('Максимальное число:',max:6:2);
writeln('Для завершения работы программы нажмите <Enter>');
readln;
end.
```

Задача 114

```
{ Генерирует три последовательности
случайных чисел и вычисляет среднее
арифметическое каждой последовательности }
const
  N=3; { кол-во последовательностей }
  L=10; { длина последовательности }
var
  r: integer; { случайное число }
  sum: integer; { сумма чисел последовательности }
  sred: real; { среднее арифметическое }
  i,j: integer; { счетчики циклов }
begin
  writeln('Случайные числа');
  Randomize; { инициализация генератора случ. чисел }
  for i:=1 to N do
    begin
      { генерируем последовательность }
      sum:=0; { не забыть обнулить ! }
      for j:=1 to L do
        begin
          r:=Random(10)+1;
          write(r:3);
          sum:=sum+r;
        end;
      sred:=sum/L;
      writeln(' Сред.арифм.: ',sred:6:2);
    end;
  writeln('Для завершения работы программы нажмите <Enter>');
  readln;
end.
```

Задача 116

```
{ Таблица функции y=|x| }  
const  
  LB=-4; { нижняя граница диапазона изменения аргумента }  
  HB=4; { верхняя граница диапазона изменения аргумента }  
  DX=0.5; { приращение аргумента }  
  
var  
  x,y: real; { аргумент и значение функции }  
  n: integer; { кол-во точек }  
  i: integer; { счетчик циклов }  
  
begin  
  writeln('Таблица значений функции y=|x| ');  
  n:=Round((HB - LB)/DX)+1;  
  x:=LB;  
  for i:=1 to n do  
    begin  
      y:=Abs(x);  
      writeln(x:6:2,y:6:2);  
      x:=x+DX;  
    end;  
  readln;  
end.
```

Задача 119

```
{ Выводит таблицу умножения на 7 }  
  
var  
  m: integer; { число, для которого надо вывести  
               таблицу умножения (множимое) }  
  n: integer; { множитель }  
  p: integer; { произведение }  
  
begin  
  m:=7;  
  for n:=1 to 10 do  
    begin  
      p:=m*n;  
      writeln(m,'x',n,'=',p);  
    end;  
  readln; { чтобы результат не исчезал с экрана }  
end.
```

Задача 120

```
{ Преобразование десятичного числа в двоичное }  
var  
  dec: integer; { десятичное число }  
  v: integer;   { вес формируемого разряда }  
  i: integer;   { номер формируемого разряда }  
begin  
  writeln('Преобразование десятичного числа в двоичное');  
  writeln('Введите целое число от 0 до 255 и нажмите ',  
    '<Enter>');  
  write('->');  
  readln(dec);  
  write('Десятичному числу ',dec,' соответствует двоичное ');  
  v:=128; { вес старшего разряда }  
  for i:=1 to 8 do  
    begin  
      if dec >= v then  
        begin  
          write('1');  
          dec:=dec-v;  
        end  
      else write('0');  
      v:=Round(v/2); { вес следующего разряда в два раза }  
                   { меньше веса текущего разряда }  
    end;  
  readln;  
end.
```

Задача 121

```
{ Выводит на экран квадрат Пифагора - таблицу умножения. }  
var  
  i,j: integer; { номер строки и столбца таблицы }  
  p: integer;   { произведение i на j }  
begin  
  write('':4);           { левая верхняя клетка таблицы }  
  for j:=1 to 10 do    { первая строка - номера столбцов }  
    write(j:4);  
  writeln;  
  for i:=1 to 10 do
```

```
begin
    write(i:4);          { номер строки }
    for j:=1 to 10 do { строка таблицы }
        write(i*j:4);
    writeln;
end;
readln; { чтобы результат не исчезал с экрана }
end.
```

Задача 122

```
{ Вычисление суммы ряда  $1 - 1/3 + 1/5 - 1/7 + \dots$  }
var
    x: real;      { член ряда }
    n: integer;   { количество суммируемых членов }
    summ: real;   { частичная сумма }
    i: integer;   { счетчик циклов }
begin
    writeln('Вычисление суммы ряда  $1 - 1/3 + 1/5 - 1/7 + \dots$ ');
    write('Введите кол-во суммируемых членов ряда ->');
    readln(n);
    summ:=0;
    for i:= 1 to n do
        begin
            x:=1/(2*i - 1);
            if (i mod 2) = 0
                then x:=-1*x;
            summ:=summ+x;
        end;
    writeln('Сумма ряда:      ',summ:8:6);
    writeln('Значение pi/4 ',pi/4:8:6);      { pi — именованная
                                              константа }
    readln;
end.
```

Задача 123

```
{ Приближенное вычисление интеграла методом прямоугольников }
var
    a,b: real;    { границы отрезка }
    dx: real;     { приращение аргумента (величина интервала) }
```

```
s: real;      { приближенное значение интеграла }
n: integer;   { количество интервалов }
x: real;      { аргумент }
y: real;      { значение функции в начале интервала }
i: integer;

begin
  writeln('Приближенное вычисление интеграла');
  write('Нижняя граница отрезка -> ');
  readln(a);
  write('Верхняя граница отрезка -> ');
  readln(b);
  write('Приращение аргумента -> ');
  readln(dx);
  n:=Round((b-a)/dx);
  x:=a;
  s:=0;
  for i:=1 to n do
    begin
      y:=5*x*x-x+2; { значение функции в начале интервала }
      s:=s+y*dx;
      x:=x+dx;
    end;
  writeln('Значение интеграла: ',s:9:3);
  writeln('Для завершения работы программы нажмите <Enter>');
  readln;
end.
```

Задача 124

```
{ Приближенное вычисление интеграла методом трапеций }
var
  a,b: real;    { границы отрезка }
  dx: real;     { приращение аргумента (величина интервала) }
  s: real;      { приближенное значение интеграла }
  n: integer;   { количество интервалов }
  x: real;      { аргумент }
  y1,y2: real; { значение функции в начале и в конце интервала }
  i: integer;
```

begin

```
writeln('Приближенное вычисление интеграла');
write('Нижняя граница отрезка -> ');
readln(a);
write('Верхняя граница отрезка -> ');
readln(b);
write('Приращение аргумента -> ');
readln(dx);
n:=Round((b-a)/dx);
x:=a;
s:=0;
for i:=1 to n do
  begin
    y1:=5*x*x-x+2; { значение ф-и в начале интервала }
    x:=x+dx;
    y2:=5*x*x-x+2; { значение ф-и в конце интервала }
    s:=s+(y1+y2)*dx/2;
  end;
writeln('Значение интеграла: ',s:9:3);
writeln('Для завершения работы программы нажмите <Enter>');
readln;
end.
```

Задача 126

```
{ Преобразование десятичного числа в двоичное }
var
  dec: integer; { десятичное число }
  v: integer;   { вес формируемого разряда }
  i: integer;   { номер формируемого разряда }
begin
  writeln('Преобразование десятичного числа в двоичное');
  writeln('Введите целое число от 0 до 255 и нажмите ',
    '<Enter>');
  write('->');
  readln(dec);
  write('Десятичному числу ',dec,' соответствует двоичное ');
  v:=128; { вес старшего (восьмого) разряда }
  for i:=1 to 8 do
```

```

begin
  if dec >= v then
    begin
      write('1');
      dec:=dec-v;
    end
    else write('0');
  v:=Round(v/2); { вес следующего разряда в два раза
                  меньше }
end;
readln;
end.

```

Задача 127

```

{ Программа проверяет знание таблицы умножения }
uses
  Crt;
var
  numb1,numb2:integer; { Первое и второе число }
  res:integer; { Результат умножения первого числа на второе }
  otv:integer; { Ответ испытуемого }
  kol:integer; { Количество правильных ответов }
  i:integer;

begin
  ClrScr; { очистить экран }
  writeln(' *** Проверка знания таблицы умножения ***');
  writeln(' После примера введите ответ и нажмите <Enter>.');
  writeln;
  kol:=0; { правильных ответов }
  Randomize; { инициализация генератора случайных чисел }
  for i:=1 to 10 do { 10 примеров }
  begin
    numb1:=Random(9)+1;
    numb2:=Random(9)+1;
    res:=numb1 * numb2;
    write(' ',numb1,'x',numb2,'=');
    readln(otv);
    if otv=res

```

```
    then
        kol:=kol+1
    else begin
        writeln(' Вы ошиблись!
',numb1,'*',numb2,'=',res);
    end;
end;
writeln(' Правильных ответов:',kol);
write(' Ваша оценка: ');
case kol of
    10:  writeln('5');
    9,8: writeln('4');
    7,6: writeln('3');
    0..5:writeln('2');
end;
writeln(' Для завершения работы с программой нажмите
<Enter>');
readln;
end.
```

Задача 128

```
{ Проверка умения складывать и вычитать числа. }
uses
    Crt;

var
    numb1,numb2:integer; { числа }
    op:integer; { Действие над числами 0 - сложение,
                1 - вычитание }
    sop: char; { знак операции - "плюс" или "минус" }
    res:integer; { Результат }
    otv:integer; { Ответ испытуемого }
    kol:integer; { Количество правильных ответов }
    buf:integer; { буфер для обмена numb1 и numb2, в случае,
                если numb1<numb2 }
    i: integer; { счетчик циклов }

begin
    ClrScr;
    writeln('Проверка умения складывать и вычитать числа. ');
    writeln('После примера введите ответ и нажмите <Enter>.' );
```

```
kol:=0;
Randomize;
for i:=1 to 10 do
begin
    { сгенерируем пример }
    numb1:=Random(9)+1;    { первое число }
    numb2:=Random(9)+1;    { второе число }
    op:=Random(2);          { действие над числами }
    if op=0 then
        begin { сложение }
            res:=numb1+numb2;
            sop:='+';
        end
    else
        begin { Вычитание }
            sop:='-';
            if numb1 < numb2 then
                begin { обменяем numb1 и numb2 }
                    buf:=numb2;
                    numb2:=numb1;
                    numb1:=buf;
                end;
            res:=numb1-numb2;
        end;
    write(' ',numb1,sop,numb2,'='); { вывести пример }
    readln(otv);                    { получить ответ
                                    испытуемого }

    if otv = res
        then
            kol:=kol+1
        else
            begin
                writeln(' Вы ошиблись!
',numb1,sop,numb2,'=',res);
            end;
end;

writeln(' Правильных ответов:',kol);
write(' Ваша оценка:');

case kol of
    10:  writeln('5');
    9,8: writeln('4');
```

```
7,6: writeln('3');
0..5:writeln('2');
end;
writeln(' Для завершения работы с программой нажмите
      <Enter>');
readln;
end.
```

Задача 129

```
{ Простые электронные часы }
uses Crt; { используем процедуры и функции библиотеки Crt }
var
    min,sec: integer; { минуты, секунды }
begin
    ClrScr; { очистить экран }
    for min:=1 to 3 do
    begin
        for sec:=1 to 60 do
        begin
            Delay(1000);      { задержка 1000 ms }
            GotoXY(1,1);      { установить курсор в 1-ю колонку
                               1-й строки }
            write(min,':',sec,' ');
            if KeyPressed      { если нажата клавиша }
                then Halt; { завершить программу }
        end;
    end;
end.
```

Задача 130

```
{ Вычисление среднего арифметического
последовательности положительных чисел }
var
    a : integer; { число, введенное с клавиатуры }
    n : integer; { количество чисел }
    s : integer; { сумма чисел }
    m : real;     { среднее арифметическое }
begin
    s:=0;
```

```
n:=0;
writeln('Вычисление среднего арифметического
        последовательности положительных чисел. ');
writeln('Вводите после стрелки числа. ',
        'Для завершения ввода введите ноль. ');
repeat
    write('> ');
    readln(a);
    s:=s+a;
    n:=n+1;
until a <= 0;
n:=n-1;
writeln('Введено чисел:',n);
writeln('Сумма чисел: ',s);
m := s/n;
writeln('Среднее арифметическое: ',m:6:2);

writeln('Для завершения работы программы нажмите
        <Enter>');
readln;
end.
```

Задача 131

{ Определение максимального числа последовательности
положительных чисел }

var

```
a : integer; { очередное число }
m : integer; { максимальное число }
```

begin

```
write('Определение максимального числа ');
writeln('последовательности положительных чисел. ');
writeln('Вводите после стрелки числа. ',
        'Для завершения ввода введите ноль. ');
m:=0;
repeat
    write('> ');
    readln(a);
    if a > m then m := a;
until a <= 0;
writeln('Максимальное число: ',m);
```

```
writeln('Для завершения работы программы нажмите  
      <Enter>');  
readln;  
end.
```

Задача 132

```
{ Проверка, является ли введенное с клавиатуры  
  целое число простым }  
program prost;  
var  
    n: integer;  
    d: integer;  
    r: integer;  
begin  
    write('Введите целое число-> ');  
    readln(n);  
    d:=2;          { сначала будем делить на два }  
    repeat  
        r:=n mod d;  
        if r<>0 { n не разделилось нацело на d }  
            then d:=d + 1;  
    until r=0; { пока не нашли число, на которое делится n }  
    if d=n  
        then writeln( n, ' – простое число. ' )  
        else writeln(n, ' – не простое число. ');  
end.
```

Задача 134

```
{ Игра "Угадай число" }  
const  
    NPOP=5; { количество попыток, предоставляемое игроку }  
var  
    comp: integer; { число, "задуманное" компьютером }  
    igrok: integer; { вариант игрока }  
    n: integer;     { кол-во попыток, сделанное игроком }  
begin  
    Randomize; { инициализация генератора случайных чисел }  
    comp:=Random(9)+1; { компьютер задумал число }
```

```
writeln('Игра "Угадай число.");
writeln('Компьютер "задумал" число от 1 до 10. ');
writeln('Угадайте его за ',NPOP,' попыток. ');
writeln('Введите число и нажмите <Enter> ');
repeat
    n:=n+1;
    write('-> ');
    readln(igrok);
until (n = NPOP) or (comp = igrok);
if comp = igrok
    then writeln('Вы выиграли!')
    else writeln('Вы проиграли! Компьютер задумал число ',
        comp);
readln;
end.
```

Задача 135

```
{ Таймер }
uses Crt;
var
    min,sec: integer; { задержка: минут и секунд }
begin
    writeln('Введите величину задержки в минутах и секундах, ');
    writeln('например 2 30 и нажмите <Enter> ');
    write('-> ');
    readln(min,sec);
    ClrScr;
    GotoXY(2,2);
    write(min,':',sec,' ');

    repeat
        if sec = 0 then
            begin
                min:=min-1;
                sec:=60;
            end;
        repeat
            Delay(1000);    { задержка 1 сек }
            sec:=sec-1;
            GotoXY(2,2);
```

```
write(min,':',sec,' ');
if KeyPressed { если нажата клавиша }
    then Halt; { завершить программу }
until sec = 0;
until (min = 0) and (sec = 0);

{ звуковой сигнал }
Sound(1000); { включить звук - сигнал частотой 1000 герц}
Delay(500);
Nosound; { выключить звук}
end.
```

Задача 136

```
{ Вычисление числа "ПИ"}
var
    p:real; { вычисляемое значение ПИ }
    t:real; { точность вычисления }
    n:integer; { номер члена ряда }
    elem:real; { значение члена ряда }

begin
    p:=0;
    n:=1;
    elem:=1; { начальное значение}
    write('Задайте точность вычисления ПИ-> ');
    readln(t);
    writeln('Вычисление ПИ с точностью',t:9:6);
    while elem >= t do
        begin
            elem:=1/(2*n-1);
            if (n MOD 2) = 0
                then p:=p-elem
                else p:=p+elem;
            n:=n+1;
        end;
    p:=p*4;
    writeln('Значение ПИ с точностью',t:9:6,' равно',p:9:6);
    writeln('Просуммировано ',n,' членов ряда. ');
    readln;
end.
```

Задача 137

```
{ Вычисление наибольшего общего делителя
двух целых чисел }

var
  n1,n2:integer; { числа, НОД которых надо найти }
  nod:integer;    { наибольший общий делитель }
  r:integer;      { остаток от деления n1 на n2 }

begin
  writeln('Вычисление наибольшего общего делителя');
  writeln('для двух целых чисел. ');
  write('Введите в одной строке два числа ');
  writeln('и нажмите <Enter>');
  write('->');
  readln(n1,n2);
  while (n1 mod n2) <> 0 do
    begin
      r:=n1 mod n2; { остаток от деления}
      n1:=n2;
      n2:=r;
    end;
  nod:=n2;
  writeln('НОД чисел ',n1,' и ',n2,' это ',nod);
  readln;
end.
```

Задача 138

```
{ Приветствие }

var
  name: string[40]; { имя пользователя }

begin
  writeln('Как Вас зовут?');
  writeln('(введите свое имя и нажмите <Enter>');
  write('->');
  readln(name);
  writeln('Здравствуйте, ',name,'!');
  readln;
end.
```

Задача 139

```
{ Посимвольный вывод сообщения }  
uses  
    Crt; { для доступа к процедуре Delay }  
var  
    msg: string[80]; { сообщение }  
    n: integer;      { номер выводимого символа }  
begin  
    msg:='Приветствую великого программиста! '  
    for n:=1 to Length(msg) do  
        begin  
            write(msg[n]);  
            Delay(100); { задержка 0.1 сек }  
        end;  
    readln;  
end.
```

Задача 140

```
{ Выводит код введенного символа }  
var  
    sim: char; { СИМВОЛ }  
    code: integer; { код символа }  
begin  
    writeln('Введите символ и нажмите <Enter>');  
    writeln('Для завершения работы программы введите точку');  
    repeat  
        write('->');  
        readln(sim);  
        code:=Ord(sim);  
        writeln('Символ: ',sim,' Код: ',code);  
    until sim = '.';  
end.
```

Задача 141

```
{ Вывод таблицы кодировки символов }  
var  
    ch:char; { СИМВОЛ }  
    dec:integer; { десятичный код символа }  
    i,j:integer;
```

```

begin
  dec:=0;
  for i:=0 to 15 do { шестнадцать строк }
    begin
      dec:=i; { чтобы получить таблицу кодировки }
              { для символов с кодами 128-255, }
              { эту инструкцию надо }
              { заменить на dec:=i+128; }
      for j:=1 to 8 do { восемь колонок }
        begin
          if (dec<7) or (dec>=14)
            then
              write(dec:4, '-', ' ',
                    chr(dec):1, chr(179))
            else { СИМВОЛЫ CR, LF, TAB не отображаются }
              write(dec:4, '- ', chr(179));
          dec:=dec+16;
        end;
      writeln;      { переход к новой строке экрана }
    end;
  readln;
end.

```

Задача 142

```

var
  st: string[80]; { строка текста }
  len: integer;   { длина строки }
  i: integer;     { номер обрабатываемого символа }
begin
  writeln('Введите строку текста и нажмите <Enter>');
  write('->');
  readln(st);
  len:=Length(st);
  for i:=1 to len do
    case st[i] of
      'a'..'n': st[i]:=chr(ord(st[i])-32);
      'p'..'я': st[i]:=chr(ord(st[i])-80);
    end;
  writeln('Строка, преобразованная к верхнему регистру:');
  writeln(st);

```

```
    readln;  
end.
```

Задача 143

```
{ Удаление начальных пробелов строки }  
  
var  
    st:string[80]; { строка }  
  
begin  
    writeln('Удаление начальных пробелов строки.');    write('Введите строку:');  
    readln(st);  
    while (pos(' ',st) = 1) and (length(st)>0) do  
        delete(st,1,1);  
    write('Строка без начальных пробелов:',st);  
    readln;  
end.
```

Задача 144

```
{ Проверка, является ли введенная  
с клавиатуры строка целым числом }  
  
var  
    st: string[40]; { строка }  
    n: integer;     { номер проверяемого символа }  
  
begin  
    writeln('Введите число и нажмите <Enter>');  
    write('->');  
    readln(st);  
    n:=1;  
    while (n <= Length(st)) and  
        ((st[n] >= '0') and (st[n] <='9'))  
        do n:=n+1;  
  
    write('Введенная строка ');  
    if n < Length(st)  
        then write('не ');  
    writeln('является целым числом.');    readln;  
end.
```

Задача 146

```
{ проверяет, является ли введенная строка
  шестнадцатеричным числом }

var
  st: string[20]; { строка }
  i: integer;      { номер проверяемого символа }
  error: boolean;  { в строке есть неверный символ }

begin
  writeln('Введите шестнадцатеричное число и нажмите
    <Enter>');
  write('->');
  readln(st);

  { преобразуем введенную строку к верхнему регистру }
  for i:=1 to Length(st)
    do st[i]:=UpCase(st[i]);

  i:=1;
  error:=FALSE;
  while (i <= Length(st)) and (not error)
    do if ((st[i] >= '0') and (st[i] <= '9')) or
        ((st[i] >= 'A') and (st[i] <= 'F'))
        then i:=i+1
        else error:=TRUE;

  write('Строка ');
  if error
    then write('не ');
  writeln('является шестнадцатеричным числом. ');
  readln;
end.
```

Задача 147

```
{ проверяет, является ли введенная строка
  дробным числом без знака }

var
  st: string[20]; { строка }
  i: integer;      { номер проверяемого символа }
  err: boolean;    { TRUE - строка не дробное число }
```

begin

```
writeln('Введите дробное число и нажмите <Enter>');
write('->');
readln(st);
i:=1;
err:=TRUE; { пусть строка - не дробное число }
if (st[i] >='1') and (st[i] <='9') then { первый символ
                                         цифра }
```

begin

```
{ за цифрой возможно следуют еще цифры }
```

```
while (st[i] >='1') and (st[i] <='9') and
(i<Length(st))
```

```
do i:=i+1;
```

```
{ за цифрами следует точка, но она не последний символ }
```

```
if ((st[i] = '.') and (i < Length(st))) then { точка }
```

begin

```
i:=i+1;
```

```
{ за точкой должна быть хотя бы одна цифра }
```

```
if ((st[i] >='1') and (st[i] <='9')) then
```

begin

```
while ((st[i] >='1') and (st[i] <='9') and
(i<Length(st)))
```

```
do i:=i+1;
```

```
if i = Length(st) { последний символ - цифра }
```

```
then err:=FALSE; { предположение
                   об ошибке ложно }
```

end;

end;

end;

```
write('Строка ');
```

```
if err
```

```
then write('не ');
```

```
writeln('является дробным числом.');
```

```
readln;
```

end.

Задача 148

{ Программа преобразует введенное с клавиатуры
двоичное восьмиразрядное число в десятичное }

var

```
bin: string[8]; { изображение двоичного числа }
```

```
dec: integer;    { десятичное число }
r: string[1];    { i-й разряд двоичного числа }
v: integer;      { вес i-го разряда двоичного числа }
i: integer;      { номер разряда двоичного числа }

begin
  writeln('Введите восьмиразрядное двоичное число');
  writeln('и нажмите <Enter>.');
  write('->');
  readln(bin);

  if Length(bin) <> 8 then
    writeln('Число должно быть восьмиразрядным.')
  else begin
    dec:=0;
    v:=128; {вес старшего (8-го) разряда двоичного числа }
    for i:=1 to 8 do
      begin
        r:=bin[i]; { выделить i-й разряд }
        if r = '1' then
          dec:=dec+v;
          v:=Round(v/2); { вычислить вес следующего разряда }
        end;
      write('Двоичному числу ',bin,');
      writeln(' соответствует десятичное ',dec);
    end;
    writeln('Для завершения работы программы нажмите  
      <Enter>.');
    readln;
  end.
```

Задача 149

```
var
  st: string[2]; { шестнадцатеричное число }
  d: integer;    { десятичное число, соответствующее  
                введенному шестнадцатеричному }
  v: integer;    { вес разряда шестнадцатеричного числа }
  i: integer;

begin
  writeln('Введите двухразрядное шестнадцатеричное число ',
    'и нажмите <Enter>');
```

```
write('->');
readln(st);

{ преобразуем введенную строку к верхнему регистру }
for i:=1 to Length(st)
    do st[i]:=UpCase(st[i]);

i:=Length(st); { обрабатываем с младшего разряда }
v:=1;          { и его вес равен единице }

while (i>0) and
    (((st[i] >= '0') and (st[i] <= '9')) or
     ((st[i] >= 'A') and (st[i] <= 'F')))) do
begin
    { здесь СИМВОЛ - цифра или
      латинская буква от A до F }
    if (st[i] >= '0') and (st[i] <= '9')
        then d:=d + v * ( Ord(st[i])-48 )
            { Ord('0') = 48, Ord('1') = 49, и т.д. }
        else d:=d + v * (Ord(st[i]) - 55);
            { Ord('A') = 65, Ord('B') = 66, и т.д. }

    i:=i-1; { к предыдущему разряду }
    v:=v*16;
end;

if i=0 { обработаны все разряды }
    then writeln('Шестнадцатеричному числу ',st,
                 ' соответстvue десятичное ',d)
    else writeln('Введенная строка не является ',
                 'шеснадцатеричной цифрой.');
```

```
readln;
end.
```

Задача 150

```
{ Программа преобразует десятичное число
  в число в указанной пользователем системе счисления
  (от 2-х до 10-ти) }

uses Crt;
```

```
var
  o:integer;      { Основание системы счисления }
  n:integer;      { Исходное число }
  r:integer;      { Остаток от деления числа на основание
                  сист. счисл. }
  s:string[16];   { Представление числа в заданной
                  сист. счисл. }
  buf:string[1];

begin
  ClrScr;
  write('Введите целое число -> ');
  readln(n);
  write('Введите основание системы счисления -> ');
  readln(o);
  s:='';
  { делим исходное число на основание системы
    счисления до тех пор, пока остаток от деления
    больше основания системы счисления.
    Остаток от деления на каждом шаге - очередная
    цифра. }
  repeat
    r:=n mod o; { очередная цифра }
    n:=n div o; { целая часть деления }
    Str(r,buf); { преобразование цифры в строку}
    s:=buf+s;
  until (n<o);
  Str(n,buf);
  s:=buf+s;
  writeln(s);
  readln;
end.
```

Задача 151

```
{ Программа преобразует десятичное число
  в шестнадцатеричное }
uses Crt;
var
  n:integer;      { Исходное число }
  r:integer;      { Остаток от деления числа на основание
                  сист. счисл. }
```

```
s:string[16];    { Представление числа в заданной
                  сист. счисл. }

buf:string[1];

begin
  ClrScr;
  write('Введите целое число -> ');
  readln(n);
  s:='';
  { делим исходное число на основание системы
    счисления (16) до тех пор, пока остаток от деления
    больше основания системы счисления.
    Остаток от деления на каждом шаге - очередная
    цифра. }
  write('Десятичному числу ',n);
  write(' соответствует шестнадцатеричное ');
  repeat
    r:=n mod 16;  { очередная цифра }
    n:=n div 16;  { целая часть деления }
    if r<10
      then buf:=chr(r+48) { chr(48) = '0', chr(49)='1'
                          и т.д. }
      else buf:=chr(r+55); { chr(65) = 'A', chr(66)='B'
                          и т.д. }

    s:=buf+s;
  until(n<16);
  if n <> 0 then begin
    if n<10
      then buf:=chr(n+48)
      else buf:=chr(n+55);
    s:=buf+s;
  end;
  writeln(s);
  readln;
end.
```

Задача 152

```
{ Программа вычисляет значение арифметического
  выражения, введенного с клавиатуры }

uses Crt;
```

```
var
  s: string[80]; { строка }
  l: integer;    { длина строки }
  z: integer;    { значение выражения }
  n: integer;    { очередное число }
  i: integer;    { номер числа в строке }
  err: integer; { код ошибки, при преобразовании символа
                  в число }

begin
  ClrScr;
  writeln('Введите арифметическое выражение,');
  writeln('например, 4+5-3-5+2 и нажмите клавишу <Enter>');
  write('->');
  readln(s);
  val(s[1],z,err);
  i:=3;
  repeat
    val(s[i],n,err); { получить очередное однозначное
                     число }

    if s[i-1] = '+'
      then z:=z+n
      else z:=z-n;
    i:=i+2;
  until i > Length(s);
  writeln('Значение введенного выражения: ',z);
  writeln('Для завершения программы нажмите <Enter>');
  readln;
end.
```

Задача 153

```
{ Подсчет кол-ва ненулевых эл-тов массива }

const
  SIZE=5; { размер массива }

var
  a: array[1..SIZE] of integer;
  n: integer; { кол-во ненулевых эл-тов }
  i: integer; { индекс }

begin
  writeln('Ввод массива целых чисел.');
```

end.

```
{ Поиск минимального элемента массива }

const
    NB=5; { размер массива }

var
    a : array [1..NB] of integer;
    min: integer; {минимальный элемент массива }
    i : integer;

begin
    writeln('Поиск минимального элемента массива. ');
    writeln('Введите в одной строке элементы массива
        (' ,NB, ' целых чисел) ');
    writeln('и нажмите <Enter>. ');
    write('-> ');
    for i:=1 to NB-1 do
        read(a[i]);
    readln(a[NB]); { чтобы правильно работала
                    находящаяся в конце программы
                    инструкция readln }
    min:=a[1]; { Предположим, что первый эл-т минимальный }
    { сравним оставшиеся эл-ты массива с минимальным }
    for i:=2 to NB do
        if a[i] < min then min := a[i];
    writeln('Минимальный элемент массива: ',min);
    readln;
end.
```

Задача 156

```
{ Вычисление средней (за неделю) температуры воздуха }
const
    { названия дней недели - массив строковых констант }
    day: array[1..7] of string[11]=('Понедельник','Вторник',
                                     'Среда','Четверг','Пятница',
                                     'Суббота','Воскресенье');

var
    temper:array[1..7] of real; { температура }
    sum:real;                   { сумма температур за неделю }
    sredn: real;                { средняя температура за неделю }
    i:integer;

begin
    writeln('Задайте температуру воздуха за неделю.');
```

```
    for i:=1 to 7 do
        begin
            write(day[i],'->');
            readln(temper[i]);
        end;

    { вычисление средней температуры за неделю }
    sum:=0;
    for i:=1 to 7 do
        sum:=sum + temper[i];
    sredn:=sum/7;
    writeln('Средняя температура за неделю:',sredn:6:2);
    readln;

end.
```

Задача 157

```
{ Поиск в массиве методом перебора элементов }

var
    massiv:array[1..10] of integer; { массив целых }
    obrazec:integer; { образец для поиска }
    naiden:boolean; { признак совпадения с образцом }
    i:integer;

begin
    { ввод 10 целых чисел }
    writeln('Поиск в массиве.');
```

```
    write('Введите 10 целых в одной строке через пробел ');
```

```
writeln ('и нажмите <Enter>');
write('->');
for i:=1 to 10 do read(massiv[i]);
{ числа введены в массив }
write('Введите образец для поиска (целое число)-> ');
readln(образец);
{ поиск простым перебором }
naiden:=FALSE; { совпадений нет }
i:=1; { проверяем с первого элемента массива }
repeat
    if massiv[i] = образец
        then naiden:=TRUE      { совпадение с образцом }
        else i:=i+1;          { переход к следующему элементу }
until (naiden) or (i>10); { завершим, если совпадение }
                        { с образцом или проверен }
                        { последний элемент массива }

if naiden
    then writeln('Совпадение с элементом номер', i:3, '. ')
    else writeln('Совпадений с образцом нет. ');
readln;
end.
```

Задача 158

```
{ Проверка, образуют ли элементы массива
  возрастающую последовательность }
const
    SIZE=5;
var
    a:array[1..SIZE] of integer; { массив }
    k: integer;    { индекс }
    ok: boolean;   { ok = True - последовательность неубывающая }
begin
    writeln('Проверка, образуют ли элементы массива');
    writeln('возрастающую последовательность');
    write('Введите в одной строке ',SIZE,' целых чисел ');
    writeln ('и нажмите <Enter>');
    for k:=1 to SIZE-1 do
        read(a[k]);
    readln(a[SIZE]);
```

```
k:=1;
ok:=TRUE;
repeat
    if a[k] < a[k+1]
        then k:=k+1
        else ok:=FALSE;
until (ok =FALSE) or (k = SIZE);

write('Введенная последовательность ');
if not ok
    then write('не ');
writeln('является возрастающей. ');
readln;
end.
```

Задача 160

```
{ Проверка, сколько раз введенное с клавиатуры число
  встречается в массиве }
const
    HB=5;
var
    a: array[1..HB] of integer; { массив }
    obr: integer;                { образец }
    n: integer; { кол-во элементов массива,
                  значение которых равно образцу }
    i: integer; { индекс }
begin
    write('Введите массив - ');
    writeln(HB, ' целых чисел в одной строке. ');
    write('->');
    for i:=1 to HB-1 do
        read(a[i]);
    readln(a[HB]);
    write('Введите образец для сравнения -> ');
    readln(obr);
    n:=0;
    { обработка }
    for i:=1 to HB do
        if a[i] = obr then n:=n+1;
```

```
if n <> 0
    then writeln('Число ',obr,' встречается в массиве ',
                n,' раз.')
    else writeln('Ни один элемент массива не равен
                образцу');
readln;
end.
```

Задача 162

```
{ Сортировка массива по возрастанию методом прямого выбора }
const
    SIZE=5; { размер массива }
var
    a:array[1..SIZE] of integer;
    i:integer; { номер элемента, от которого ведется поиск }
                { минимального эл-та }
    min:integer; { номер минимального элемента в части }
                { массива от i до верхней границы массива }
    j:integer; { номер эл-та, сравниваемого с минимальным }
    buf:integer; { буфер, используемый при обмене эл-тов }
                { массива }
    k:integer; { индекс для ввода и вывода }
begin
    writeln('Сортировка массива. ');
    write('Введите в одной строке ',SIZE);
    writeln('целых чисел и нажмите <Enter>');
    write('-> ');

    for k:=1 to SIZE-1
        do read(a[k]);
    readln(a[SIZE]);

    writeln('Сортировка');
    for i:=1 to SIZE-1 do
        begin
            { поиск минимального эл-та
              в части массива от a[i] до a[SIZE] }
            min:=i;
            for j:=i+1 to SIZE do
```

```

        if a[j]<a[min] then min:=j;
    { поменяем местами a[min] и a[i] }
    buf:=a[i];
    a[i]:=a[min];
    a[min]:=buf;
    { цикл сортировки закончен }
    { отладочная печать }
    { выведем промежуточное состояние массива }
    for k:=1 to SIZE do write(a[k], ' ');
    writeln;

end;
writeln;
{ выведем отсортированный массив }
writeln('Массив отсортирован.');
```

for k:=1 to SIZE **do**
 write(a[k], ' ');
 readln;

end.

Задача 163

```

{ Сортировка массива целых по возрастанию методом "пузырька" }
const
    SIZE=5;
var
    a:array[1..SIZE] of integer;
    i:integer; { счетчик циклов }
    k:integer; { текущий индекс элемента массива }
    buf:integer;
begin
    writeln('Сортировка массива методом "пузырька".');
    write('Введите в одной строке ',SIZE,' целых чисел ');
    writeln ('и нажмите <Enter>');
    for k:=1 to SIZE-1 do
        read(a[k]);
    readln(a[SIZE]);

    writeln('Сортировка.');
```

for i:=1 to SIZE-1 **do**
 begin
 for k:=1 to SIZE-1 **do**

```
begin
  if a[k]>a[k+1] then
    begin
      { обменяем k-й и (k+1)-й элементы }
      buf:=a[k];
      a[k]:=a[k+1];
      a[k+1]:=buf;
    end;
  end;
  { отладочная печать - состояние
    массива после очередного цикла сортировки }
  for k:=1 to SIZE do write(a[k], ' ');
  writeln;
end;
writeln;
writeln('Массив отсортирован.');
```

for k:=1 to SIZE do write(a[k], ' ');

readln;

end.

Задача 164

```
{ Объединение (слияние) двух упорядоченных массивов в один }
const
  NB = 5; { размер исходных массивов }
var
  a,b: array[1..NB] of integer; { исходные массивы }
  c: array[1..2*NB] of integer; { массив - результат }
  k,l,m : integer; { индексы массивов a,b и c }

begin
  writeln('Объединение двух упорядоченных по возрастанию ',
    'массивов.');
```

writeln('Введите в одной строке элементы первого ',

'массива,');

write('(',NB,' целых чисел) -> ');

for k:=1 to NB-1 **do**

 read(a[k]);

 readln(a[NB]);

writeln('Введите в одной строке элементы второго массива');

write('(',NB,' целых чисел) -> ');

```
for l:=1 to HB-1 do
    read(b[l]);
readln(b[HB]);

k:=1; l:=1; m:=1;
repeat
    if a[k] < b[l]
        then
            begin
                c[m]:=a[k];
                m:=m+1;
                k:=k+1;
            end
        else
            if a[k] > b[l]
                then
                    begin
                        c[m]:=b[l];
                        m:=m+1;
                        l:=l+1;
                    end
                else
                    begin
                        c[m]:=a[k];
                        c[m+1]:=b[l];
                        k:=k+1;
                        l:=l+1;
                        m:=m+2;
                    end;
            end;
until (k > HB) or (l > HB); { один из двух исходных
массивов полностью переписан в массив C }

while k <= HB do { есть эл-ты A, не переписанные в C }
begin
    c[m]:=a[k];
    k:=k+1;
    m:=m+1;
end;

while l <= HB do { есть эл-ты B, не переписанные в C }
begin
    c[m]:=b[l];
```

```
    l:=l+1;
    m:=m+1;
end;

writeln('Массив - результат');
for i:=1 to 2*HB do
    write(c[i], ' ');
writeln;
writeln('Для завершения работы нажмите <Enter>.');
readln;
end.
```

Задача 165

```
{ Бинарный поиск в упорядоченном массиве }
label
    bye;
const
    HB=10;
var
    a:array[1..10] of integer; { массив целых }
    obr:integer;               { образец для поиска }
    ok: boolean;               { TRUE - массив упорядочен }

    sred,verh,niz:integer; { номера среднего, верхнего
                           и нижнего эл-тов массива }
    found:boolean;{ признак совпадения с образцом }
    n:integer;    { счетчик сравнений с образцом }
    i:integer;

begin
    { ввод массива }
    writeln('*** Бинарный поиск в упорядоченном массиве ***');
    write('Введите массив (в одной строке ',HB);
        writeln(' целых чисел) и нажмите <Enter>');
    write('->');
    for i:=1 to HB-1 do
        read(a[i]);
    readln(a[HB]);

    { проверим, является ли массив упорядоченным }
    i:=1;
```

```
ok:=TRUE; { пусть массив упорядочен }
while (i < HB) and ok
  do if a[i] <= a[i+1]
    then i:=i+1
    else ok:=FALSE;

if not ok then
begin
  writeln('Введенный массив не является',
    'упорядоченным по возрастанию');
  goto bye;
end;

write('Введите образец для поиска (целое число) -> ');
readln(obr);

{ бинарный поиск }
verh:=1;
niz:=HB;
found:=FALSE;
n:=0;
repeat
  sred:=(niz-verh) div 2 + verh; { делим массив пополам }
  n:=n+1;
  if a[sred]=obr
    then found:=TRUE
    else
      { в какой части, в верхней или в нижней,
        может находиться искомый элемент? }
      if obr<a[sred]
        then niz:=sred-1
        else verh:=sred+1;
until (verh > niz) or found;
if found
  then write('Совпадение с элементом номер ',
    sred, '. Выполнено ', n, ' сравнений.')
  else writeln('Образец в массиве не найден.');
```

bye:

```
  readln;
end.
```

Задача 166

```
{ Анализ роста учеников }  
const  
    NSB=30; { максимально возможное  
              кол-во учеников в классе }  
  
var  
    rost: real; { рост ученика }  
    class: array[1..NSB] of real; { рост учеников }  
    n: integer; { кол-во учеников, о которых  
                  введены сведения }  
    sred: real; { средний рост }  
    m: integer; { кол-во учеников, чей рост  
                  больше среднего }  
    sum: real; { суммарный рост }  
    i: integer;  
  
begin  
    writeln('*** Анализ роста учеников ***');  
    writeln('Введите рост (см) и нажмите <Enter>');  
    writeln('Для завершения введите 0 и нажмите <Enter>');  
  
    n:=0;  
    repeat  
        write('->');  
        readln(rost);  
        if rost <> 0 then  
            begin  
                n:=n+1;  
                class[n]:=rost;  
                sum:=sum+rost;  
            end;  
    until (n > NSB) or (rost = 0);  
  
    if n > 0 then  
        begin  
            sred:=sum/n;  
  
            m:=0;  
            { сравним рост каждого со средним }  
            for i:=1 to n do  
                if class[i] > sred then m:=m+1;
```

```

        writeln('Средний рост: ',sred:6:1,' см');
        writeln('У ',m,'-х учеников рост превышает ',
                'средний.');
```

end

```

        else writeln('Нет данных для обработки.');
```

readln;

end.

Задача 167

```

{ Вычисление суммы элементов массива (по столбцам) }

const
    ROW=3; { кол-во строк }
    COL=5; { кол-во столбцов }

var
    a: array[1..ROW,1..COL] of integer; { массив }
    s: array[1..COL] of integer; { сумма элементов }
    i,j: integer;

begin
    writeln('Введите массив.');
```

writeln('После ввода элементов каждой строки,',

COL,' целых чисел, нажимайте <Enter>');

```

for i:=1 to ROW do { ROW строк }
    begin
        write('->');
        for j:=1 to COL-1 do
            read(a[i,j]);
            readln(a[i,COL]);
        end;

    writeln('Введенный массив');
```

```

for i:=1 to ROW do
    begin
        for j:=1 to COL-1 do
            write(a[i,j]:4);
            writeln(a[i,COL]:4);
        end;

    { обработка }
    for j:=1 to COL do      { для каждого столбца }
        for i:=1 to ROW do { суммируем эл-ты одного столбца }
            s[j]:=s[j]+a[i,j];
```

```
writeln('-----');  
for i:=1 to COL do  
    write(s[i]:4);  
writeln;  
readln;  
end.
```

Задача 170

```
{ вычисление определителя матрицы второго порядка }  
var  
    a: array[1..2,1..2] of real;  
    det: real;      { определитель (детерминант) }  
    i,j: integer; { индексы массива }  
begin  
    writeln('Введите матрицу второго порядка. ');  
    writeln('После ввода элементов строки нажимайте <Enter>');  
    for i:=1 to 2 do  
        begin  
            write('->');  
            read(a[i,1]);  
            readln(a[i,2]);  
        end;  
    det:=a[1,1]*a[2,2] - a[1,2]*a[2,1];  
  
    writeln('Определитель матрицы ');  
    for i:=1 to 2 do  
        begin  
            for j:=1 to 2 do  
                write(a[i,j]:6:2);  
            writeln;  
        end;  
    writeln('равен ',det:6:2);  
    readln;  
end.
```

Задача 171

```
{ Проверяет, является ли матрица магическим квадратом }  
const  
    MAX=5; { максимальный размер матрицы }
```

var

```
a: array[1..MAX,1..MAX] of integer; { матрица }
n: integer;    { размер проверяемой матрицы }
ok:boolean;    { TRUE - матрица является маг. квадратом }
i,j: integer;  { индексы массива }
sum: integer;  { сумма эл-тов главной диагонали матрицы }
temp: integer; { сумма элементов текущей строки, столбца
                или второй диагонали матрицы }
```

begin

```
write('Введите размер матрицы (3..4) ',
      'и нажмите <Enter> ->');
readln(n);
Writeln('Введите строки матрицы');
Writeln('После ввода строки',n,' целых чисел,',
      'нажимайте <Enter>');
```

for i:=1 **to** n **do****begin**

```
write('->');
for j:=1 to n-1 do read(a[i,j]);
readln(a[i,n]);
```

end;

```
ok:=TRUE; { пусть матрица - магический квадрат }
sum:=0;
```

```
{ вычислим сумму элементов главной диагонали }
```

for i:=1 **to** n **do**

```
sum:=sum+a[i,i];
```

```
{ вычисляем суммы по строкам }
```

```
i:=1;
```

repeat

```
temp:=0; { сумма эл-тов текущей строки }
for j:=1 to n do temp:=temp+a[i,j];
i:=i+1;
if temp <> sum then ok:=FALSE;
```

until (not ok) **or** (i > n);**if** ok **then**

```
{ здесь сумма элементов каждой строки
  равна сумме эл-тов главной диагонали }
```

```
begin
  { вычисляем суммы по столбцам }
  j:=1;
  repeat
    temp:=0; { сумма эл-тов текущего столбца }
    for i:=1 to n do temp:=temp+a[i,j];
    j:=j+1;
    if temp <> sum then ok:=FALSE;
  until (not ok) or (j > n);
  if ok then
    { здесь сумма эл-тов каждой строки
      равна сумме эл-тов каждого столбца и
      равна сумме эл-тов главной диагонали}
    begin
      { вычислим сумму эл-тов второй
        главной диагонали }
      temp:=0;
      j:=n;
      for i:=1 to n do
        begin
          temp:=temp+a[i,j];
          j:=j-1
        end;
      if temp <> sum then ok:=FALSE;
    end;
  end;
  write('Введенная матрица ');
  if not ok
    then write('не ');
  writeln('является магическим квадратом. ');
  readln;
end.
```

Задача 173

```
{ Подводит итоги Олимпийских игр }
const
  N=10; {количество стран-участниц }
  страна: array[1..N] of string[9]=('Австрия','Германия',
    'Канада','Китай','Корея','Норвегия','Россия',
    'США','Финляндия','Япония');
```

```

var
  { таблица результатов }
  result: array[1..N+1, 1..5] of integer;
  { N+1-я строка используется как буфер
    при сортировке таблицы }

  i,j: integer;
  max: integer; { номер строки таблицы, в которой
                  количество очков максимально }
  buf: string[9]; { используется при сортировке }
begin
  writeln('Итоги Олимпийских игр');
  writeln('Введите в одной строке количество золотых, ',
    'серебряных и бронзовых медалей.');
```

{ ввод исходных данных }

```

for i:=1 to N do
  begin
    write(strana[i], ' ->');
    read(result[i,1],result[i,2]); { кол-во золотых
                                    и серебряных }
    readln(result[i,3]);           { кол-во бронзовых }
  end;
{ вычислим общее кол-во медалей и очков }
for i:=1 to N do
  begin
    result[i,4]:=result[i,1]+result[i,2]+result[i,3];
    re-
sult[i,5]:=result[i,1]*7+result[i,2]*6+result[i,3]*5;
  end;
{ сортировка массива в соответствии с количеством очков }
{ методом простого выбора }
for i:=1 to N-1 do
  begin
    { в части таблицы начиная со строки i
      найти j-ю строку, в которой элемент
      result[j,5] максимальный }
```

max:=i; { пусть это строка с номером i }

```

  for j:=i+1 to N do
    if result[j,5] > result[max,5] then max:=j;
```

```
{ Обменяем i-ю строку со строкой с номером max.
  В качестве буфера используем последнюю,
  не используемую строку таблицы. }
buf:=strana[i];
strana[i]:=strana[max];
strana[max]:=buf;
for j:=1 to 5 do
  begin
    result[N+1,j]:=result[i,j];
  end;
for j:=1 to 5 do
  begin
    result[i,j]:=result[max,j];
  end;
for j:=1 to 5 do
  begin
    result[max,j]:=result[N+1,j];
  end;

end;
{ здесь таблица упорядочена }
writeln;
writeln('Итоги зимней Олимпиады в Нагано, 1998 г. ');
writeln('Страна':12 , 'Золото':8, 'Серебро':8, 'Бронза':8,
        'Всего':8, 'Очков':8);
for i:=1 to N do
  begin
    write(i:2, strana[i]:10);
    for j:=1 to 5 do
      write(result[i,j]:8);
    writeln;
  end;
readln;
end.
```

Задача 174

```
{ Игра "Угадай число" }
```

```
const
```

```
  N=3; { уровень сложности - количество цифр в числе }
```

var

```
igrok: array[1..N] of char; { комбинация игрока }
comp:  array[1..N] of char; { комбинация компьютера }

a: array[1..N] of boolean; { a[i]= TRUE, если i-я цифра
                           компьютера совпала с одной из
                           цифр игрока }

ugad:integer;{ угадано чисел }
mesto:integer;{ из них на своих местах }

i,j: integer; { индексы массива }
```

begin

```
writeln('Компьютер задумал трехзначное число. ',
        'Вы должны его отгадать. ');
writeln('После ввода очередного числа, вам будет ',
        'сообщено, сколько цифр');
writeln('угадано и сколько из них находятся ',
        'на своих местах. ');
writeln('После ввода числа нажимайте <Enter>.');

{ компьютер "задумывает" свое число }
randomize;
for i:=1 to N do
    comp[i]:=chr(random(10)+48); { 48 - код символа '0' }

write('Компьютер задумал: '); for i:=1 to N do
write(comp[i]); writeln;

repeat
    write('Ваш вариант-> ');
    {получить вариант игрока }
    for i:=1 to N-1 do
        read(igrok[i]);
    readln(igrok[N]);

    for i:=1 to N do a[i]:=FALSE;
    { a[i] = TRUE, если i-я цифра числа компьютера
      совпала с одной из цифр числа игрока }

    { проверим, сколько цифр угадано }
    ugad:=0;
```

```

for i:=1 to N do      { проверим каждую цифру игрока }
  for j:=1 to N do
    begin
      if (igrok[i] = comp[j]) and not a[j] then
        begin
          ugad:=ugad+1;
          a[j]:=TRUE;    { запретим сравнивать
                           эту цифру компьютера с оставшимися,
                           еще не проверенными цифрами игрока }
        end;
      end;
    end;

  { проверим, сколько на своих местах }
  mesto:=0;
  for i:=1 to N do
    if igrok[i] = comp[i] then mesto:=mesto+1;

    writeln('Угадано:',ugad,'. На своих местах:',mesto);
  until (ugad = N) and (mesto = N);
  writeln('***ВЫ УГАДАЛИ ЧИСЛО!***');
  write('Нажмите <Enter> для завершения. ');
  readln;
end.

```

Задача 175

```

{ Телеграф - передача сообщений при помощи азбуки Морзе. }
{ Замечание: возможно надо увеличить величины задержек. }

uses Crt;

const
  morse: array[128..159] of string[4] = (
    '. - ', '. - . . . ', '. - - - ', '. - - . ', {А,Б,В,Г}
    '- . . ', '. . . ', '. . . - ', '- . - . . ', {Д,Е,Ж,З}
    '. . . ', '. . - - - ', '. - . - ', '. . - . . ', {И,Й,К,Л}
    '- - . ', '. - . ', '- - - - ', '. - - . . ', {М,Н,О,П}
    '. - . . ', '. . . . ', '- . . . . ', '. . - . . ', {Р,С,Т,У}
    '. . . - . ', '. . . . . ', '- . - . . . ', '- - - - . ', {Ф,Х,Ц,Ч}
    '- - - - - ', '- - - . - ', '- . . . . ', '- . - - . ', {Ш,Щ,Ъ,Ы}
    '- . . . - ', '. . . . . ', '. . . - - ', '. - . - - ' {Ь,Э,Ю,Я}
  );

```

```

{ параметры передачи }
TONE=100; { частота сигнала (Гц) }
L1=50; { длительность (мс) "точки" }
L2=100; { длительность (мс) "тире" }
L3=50; { пауза (мс) между точками и тире одной буквы }
L4=100; { пауза (мс) между буквами }
L5=150; { пауза (мс) между словами }

```

var

```

mes: string; { сообщение }
sim: string[4]; { символ в кодировке Морзе -
                последовательность точек и тире }
znak: string[1]; { "передаваемый" знак - тире или точка }
i,j: integer; { номер символа и знака }

```

begin

```

ClrScr;
writeln('*** Телеграф ***');
writeln('Введите сообщение, которое надо передать');
writeln('(используйте только большие русские буквы)');
write('->');
readln(mes);
for i:=1 to Length(mes) do

```

begin

```

    if (mes[i] >= 'А') and (mes[i] <='Я') then

```

begin

```

    { определим код очередной буквы (ф-я Ord) сообщения
      и получим из таблицы кодировки соответствующий
      элемент массива - последовательность точек и тире }
    sim:=morse[ord(mes[i])];

```

```

    j:=1;

```

repeat

```

    if (sim[j] = '-') or (sim[j] = '.') then

```

begin

```

        write(sim[j]);

```

```

        sound(1000);

```

```

        case sim[j] of

```

```

            '.': Delay(50);

```

```

            '-': Delay(100);

```

```

        end;

```

```

        NoSound;

```

```

        Delay(50);

```

end;

```
j:=j+1;
until ((sim[j] = ' ') or (j>4));
Delay(100); { пауза между буквами }
end
else
  if mes[i] = ' ' then { пробел между словами }
  begin
    write(' '); { пробел между словами сообщения }
    Delay(150);
  end;
end;
writeln;
writeln('Сообщение передано!');
writeln('Для завершения работы с программой нажмите ',
  '<Enter>');
readln;
end.
```

Задача 177

```
{ Функция max возвращает максимальное из двух чисел }
function max(a,b: integer): integer;
begin
  if a > b
  then max:=a
  else max:=b;
end;
```

Задача 178

```
{ Возвращает результат сравнения чисел
в виде символа отношения }
function Compare(a,b: real): char;
begin
  if a > b then Compare:='>'
  else
    if a < b then Compare:='<'
    else Compare:='=';
  end;
```

```

var
  x1,x2: real; { сравниваемые числа }
  res: char;   { результат сравнения }

begin
  writeln('Введите два числа и нажмите <Enter>');
  write('->');
  readln(x1,x2);
  res:=Compare(x1,x2); { вызов функции программиста }
  writeln(x1:6:2,res,x2:6:2);
  readln;
end.

```

Задача 179

```

{ Вычисляет сопротивление электрической цепи }

function Sopr(r1,r2: real; t: integer): real;
{ r1,r2 - величины сопротивлений
  t - тип соединения:
      1 - последовательное;
      2 - параллельное.
  Если тип соединения указан неверно,
  то возвращает -1 }

begin
  if t=1 then Sopr:= r1+r2;
  if t=2 then Sopr:= r1*r2/(r1+r2)
  else Sopr:=-1; { неверно указан тип соединения }

end;

```

Задача 180

```

{ Вычисление степени числа с использованием
  свойств логарифмов }

function InStep(a,b:real):real;

begin
  { А в степени В равно С
  Логарифмируем обе части равенства и получаем:
  B*ln(A) = ln(C)
  Нас интересует значение С, поэтому вычисляем
  Е в степени B*ln(A). Значение этого выражения
  равно С, что и требовалось вычислить. }

```

```
InStep:=exp(b*ln(a));
end;

var
  a: real; { число }
  b: real; { степень }
  c: real; { число в степени }
begin
  writeln('Введите число и показатель степени');
  readln(a,b);
  c:=InStep(a,b);
  writeln(a:6:3,' в степени ',b:6:3,' = ',c:6:3);
  readln;
end.
```

Задача 182

```
{ вычисляет доход по вкладу }

function Dohod(sum: real;      { сумма вклада }
              ставка: real; { процентная ставка (годовых) }
              srok: integer { срок вклада (дней) }
              ): real;

begin
  Dohod:=sum*(ставка/100/365)*srok; { 365 кол-во дней
                                     в году }
end;
```

Задача 183

```
{ Проверяет, является ли символ гласной буквой }
Function Glasn(sim:char): boolean;

const
  { гласные буквы }
  ListOfGlasn:string='АаЕеИиОоУуЫыЭэЮюЯя';

var
  p: byte; { позиция проверяемого символа
            в списке гласных }

begin
  p:=Pos(sim,ListOfGlasn); { !!!! }
  if p <> 0 { символ найден в списке }
```

```
    then Glasn:=True
    else Glasn:=False;
end;
```

Задача 184

```
{ Удаляет из строки начальные пробелы }
function LTrim(st: string): string;
begin
    while (pos(' ',st) = 1) and (length(st) > 0) do
        delete(st,1,1);
    LTrim:=st;
end;

{ проверка работы функции LTrim}
var
    s:string[80]; { строка }
begin
    writeln('Удаление из строки начальных пробелов. ');
    write('Введите строку -> ');
    readln(s);
    write('Строка без начальных пробелов:', LTrim(s));
    readln;
end.
```

Задача 186

```
{ Преобразование строчных букв в прописные }
{ Замещает стандартную процедуру UpCase}
function UpCase(st:string): string;
var
    i:integer;
begin
    for i:=0 to Length(st) do { символы нумеруются с нуля ! }
        case st[i] of
            { латинские буквы}
            'a'..'z':UpCase[i]:=chr(ord(st[i])-32);
            { русские буквы}
            'а'..'я':UpCase[i]:=chr(ord(st[i])-80);
```

```
        else { остальные символы не преобразуем }
            UpCase[i]:=st[i];
    end;
end;

{ пример использования функции UpCase }
var
    st: string;
begin
    writeln(' Введите текст и нажмите <Enter>');
    write('->');
    readln(st);
    writeln(UpCase(st));
    readln;
end.
```

Задача 188

```
{ Решение квадратного уравнения }
function KvadrUr(a,b,c: real; var x1,x2: real): integer;
    { a,b,c - коэффициенты уравнения }
    { x1,x2 - корни уравнения }
    { значение функции - количество корней
      или -1, если неверные исходные данные }
var
    d: real; { дискриминант }
begin
    if a = 0 then KvadrUr := -1
    else begin
        d:=b*b-4*a*c;
        if d < 0 then
            KvadrUr:=0 { уравнение не имеет решения }
        else begin
            if d > 0
            then KvadrUr:=2 { два разных корня }
            else KvadrUr:=1; { корни одинаковые }
            x1:=(-b+Sqrt(d))/(2*a);
            x2:=(-b-Sqrt(d))/(2*a);
        end;
    end;
end;
```

```
{ основная программа }
var
  a,b,c: real; { коэффициенты уравнения }
  x1,x2: real; { корни уравнения }
begin
  writeln('Решение квадратного уравнения');
  writeln('Введите в одной строке коэффициенты и нажмите ',
    '<Enter>');
  write('->');
  readln(a,b,c);
  case KvadrUr(a,b,c,x1,x2) of
    -1: writeln('Ошибка исходных данных. ');
    0: writeln('Уравнение не имеет решения. ');
    1: writeln('x=',x1:6:2,' Корни одинаковые. ');
    2: writeln('x1=',x1:6:2,' x2=',x2:6:2);
  end;
  readln;
end.
```

Задача 189

```
uses Crt;
var
  a:integer; { число, введенное пользователем }

{ Функция GetInt предназначена для ввода целого
  положительного числа, состоящего из одной или двух цифр.
  Во время ввода для редактирования может использоваться
  <Backspace>.
  При нажатии <Enter> функция возвращает введенное число. }

function GetInt:integer;
const
  K_BACK=8; { код клавиши <Backspace> }
  K_ENTER=13; { код клавиши <Enter> }
var
  ch:char; { символ }
  dec:byte; { код символа }
  buf:array[1..2] of char; { введенные цифры }
  n:integer; { кол-во введенных цифр }
  x,y:integer; { положение курсора }
begin
  buf[1]:=' '; buf[2]:=' ';
```

```
n:=0;
repeat
  ch:=Readkey;
  dec:=ord(ch);
  if (ch>='0') and (ch<='9') and (n<2) then
    begin
      write(ch);
      n:=n+1;
      buf[n]:=ch;
    end
  else
    if (dec=K_BACK) and (n>0) then
      begin
        n:=n-1;
        x:=WhereX;
        y:=WhereY;
        GotoXY(x-1,y);
        write(' ');
        GotoXY(x-1,y);
      end;
    until (n>0) and (dec=K_ENTER);
  { преобразуем введенную строку в число }
  if n=2
    then GetInt:=(ord(buf[1])-48)*10+ord(buf[2])-48
    else GetInt:=ord(buf[1])-48;
end;

begin
  ClrScr;
  writeln('*** Демонстрация работы функции GetInt. ***');
  writeln;
  writeln('Функция GetInt предназначена для ввода целого ',
    'положительного числа,');
  writeln('состоящего из одной или двух цифр. ');
  writeln('Во время ввода для редактирования может ',
    'использоваться <Backspace>.' );
  writeln('При нажатии <Enter> функция возвращает ',
    'введенное число. ');
  writeln;
  writeln('Введите число и нажмите <Enter>');
  write('->');
```

```
a:=GetInt;
writeln;
write('Вы ввели число ',a);
readln;
end.
```

Задача 190

```
{ Выводит строку из звездочек }
procedure StarLine(len: integer);
var
    i: integer;
begin
    for i:=1 to len do
        write('*');
end;
```

Задача 193

```
{ Вычерчивает рамку }
procedure Frame(l:integer; t:integer; w:integer; h:integer);
    { l,t - координаты верхнего левого угла,
      w - ширина рамки,
      h - высота рамки }

var
    x,y:integer; { координаты выводимого символа }
    i:integer;
    c1,c2,c3,c4,c5,c6:char; { символы, из которых рисуется
                              рамка }

begin
    c1:=chr(218); { символ верхнего левого угла }
    { функция chr возвращает символ, номер которого }
    { указан при вызове функции }
    c2:=chr(196); { символ горизонтальной линейки }
    c3:=chr(191); { символ правого верхнего угла }
    c4:=chr(179); { символ вертикальной линейки }
    c5:=chr(192); { символ левого нижнего угла }
    c6:=chr(217); { символ правого нижнего угла }
    GoToXY(l,t);
    write(c1);
```

```

for i:=1 to w-2 do { символы верхней границы рамки }
    write(c2);
write(c3);
y:=t+1;
x:=l+w-1;
for i:=1 to h-2 do{символы левой и правой границ }
    begin
        GoToXY(l,y);
        write(c4);
        GoToXY(x,y);
        write(c4);
        y:=y+1;
    end;
GoToXY(l,y);
write(c5);
for i:=1 to w-2 do { символы нижней границы }
    write(c2);
write(c6);
end;

```

Задача 195

```

{ Рисует Олимпийский флаг }
uses Graph;
var
    grDriver:integer; { драйвер }
    grMode:integer;   { графический режим }
    grPath:string;    { место расположения драйвера }
    ErrCode:integer;  { результат инициализации граф. режима }

begin
    grDriver := VGA;      { режим VGA }
    grMode:=VGAHi;        { разрешение 640x480 }
    grPath:='e:\tp\bgi'; { драйвер, файл EGAVGA.BGI, находится
                          в каталоге e:\tp\bgi }

    InitGraph(grDriver, grMode,grPath);
    ErrCode := GraphResult;
    if ErrCode <> grOk then
        begin
            writeln('Ошибка инициализации графического режима.');
```

```
writeln('Для завершения работы программы нажмите ',
        '<Enter>');
readln;
Halt(1);
end;

{ полотнище флага }
SetFillStyle(SolidFill,LightGray); { сплошная заливка
                                     серым цветом }

Bar(80,80,200,135);

{ кольца }
SetColor(Green);      { зеленое }
Circle(100,100,15);
SetColor(Black);      { черное }
Circle(140,100,15);
SetColor(Red);        { красное }
Circle(180,100,15);
SetColor(Yellow);     { желтое }
Circle(120,115,15);
SetColor(Blue);       { синее }
Circle(160,115,15);

readln;
CloseGraph;
end.
```

Задача 198

```
{ Рисует кораблик с использованием метода базовой точки }
uses Graph;
const
    { шаг сетки }
    dx=5; { по X}
    dy=5; { по Y}
var
    grDriver:integer; { драйвер }
    grMode:integer;   { графический режим }
    grPath:string;    { место расположения драйвера }
    ErrCode:integer;  { результат инициализации граф. режима }
```

```
x,y:integer; { координаты базовой точки кораблика }
```

```
begin
```

```
grDriver := VGA;      { режим VGA}  
grMode:=VGAHi;        { разрешение 640x480}  
grPath:='e:\tp\bgi'; { драйвер, файл EGAVGA.BGI, находится  
                     в каталоге d:\tp\bgi }
```

```
InitGraph(grDriver, grMode,grPath);
```

```
ErrCode := GraphResult;
```

```
if ErrCode <> grOk then Halt(1);
```

```
x:=10;
```

```
y:=200;
```

```
{ корпус }
```

```
MoveTo(x,y);
```

```
LineTo(x,y-2*dy);
```

```
LineTo(x+10*dx,y-2*dy);
```

```
LineTo(x+11*dx,y-3*dy);
```

```
LineTo(x+17*dx,y-3*dy);
```

```
LineTo(x+14*dx,y);
```

```
LineTo(x,y);
```

```
{ надстройка }
```

```
MoveTo(x+3*dx,y-2*dy);
```

```
LineTo(x+4*dx,y-3*dy);
```

```
LineTo(x+4*dx,y-4*dy);
```

```
LineTo(x+13*dx,y-4*dy);
```

```
LineTo(x+13*dx,y-3*dy);
```

```
Line(x+5*dx,y-3*dy,x+9*dx,y-3*dy);
```

```
{ капитанский мостик }
```

```
Rectangle(x+8*dx,y-4*dy,x+11*dx,y-5*dy);
```

```
{ труба }
```

```
Rectangle(x+7*dx,y-4*dy,x+8*dx,y-7*dy);
```

```
{ иллюминаторы }
```

```
Circle(x+12*dx,y-2*dy,Trunc(dx/2));
```

```
Circle(x+14*dx,y-2*dy,Trunc(dx/2));
```

```
{ мачта }
```

```
Line(x+10*dx,y-5*dy,x+10*dx,y-10*dy);
```

```
{ оснастка }
```

```
MoveTo(x+17*dx,y-3*dy);
```

```
LineTo(x+10*dx,y-10*dy);
```

```
LineTo(x,y-2*dy);
```

```
readln;  
CloseGraph;  
end.
```

Задача 199

{ Выводит узор из 100 произвольно размещенных
окружностей произвольного радиуса и цвета }

```
Uses Graph;  
  
var  
    grDriver:integer;  
    grMode:integer;  
    grPath:string;  
    ErrCode:integer;  
    x,y,r: integer; { координаты центра и радиус окружности }  
    i: integer;  
  
begin  
    grDriver:=VGA;  
    grMode:=VGAHi;  
    grPath:='e:\tp\bgi';  
    InitGraph (grDriver,grMode,grPath);  
    ErrCode:=GraphResult;  
    if ErrCode <> grOK then  
        begin  
            writeln ('Ошибка инициализации графического режима.');            writeln ('Для завершения работы нажмите <Enter>');            readln;  
            Halt (1);  
        end;  
  
    Randomize;  
    for i:=1 to 100 do  
        begin  
            x:=Random(640);  
            y:=Random(480);  
            r:=Random(240);  
            Setcolor(Random(16));  
            Circle(x,y,r);  
        end;  
    readln;  
end.
```

Задача 200

```
{ Выводит узор из 200 случайно размещенных  
линий разного цвета }
```

```
Uses Graph;
```

```
var
```

```
  grDriver:integer;
```

```
  grMode:integer;
```

```
  grPath:string;
```

```
  ErrCode:integer;
```

```
  x,y: integer; { координаты конца линии }
```

```
  i: integer;
```

```
begin
```

```
  grDriver:=VGA;
```

```
  grMode:=VGAHi;
```

```
  grPath:='e:\tp\bgi';
```

```
  InitGraph (grDriver,grMode,grPath);
```

```
  ErrCode:=GraphResult;
```

```
  if ErrCode <> grOK then
```

```
    begin
```

```
      writeln ('Ошибка инициализации графического режима.');
```

```
      writeln ('Для завершения работы нажмите <Enter>');
```

```
      readln;
```

```
      Halt (1);
```

```
    end;
```

```
  Randomize;
```

```
  for i:=1 to 200 do
```

```
    begin
```

```
      x:=Random(640);
```

```
      y:=Random(480);
```

```
      Setcolor(Random(16));
```

```
      LineTo(x,y);
```

```
    end;
```

```
  readln;
```

```
end.
```

Задача 201

```
{ Рисует контур пятиконечной звезды }
```

```
uses Graph;
```

label

bye;

var

r: integer; { радиус звезды }

x0,y0: integer; { координаты центра звезды }

x,y: integer; { координаты конца луча }

a: integer; { угол между осью OX и прямой, соединяющей
центр звезды и конец луча }

i: integer;

grDriver: Integer;

grMode: Integer;

ErrCode: Integer;

res: integer;

begin

grDriver := detect;

InitGraph(grDriver, grMode, 'e:\tp\bgi');

ErrCode := GraphResult;

if ErrCode <> grOk **then****begin**

writeln('Ошибка инициализации графического режима.');

goto bye;**end;**

x0:=100;

y0:=100;

r:=20;

a:=18; { строим от правого гор. луча }

x:=x0+Round(r*cos(a*2*pi/360));

y:=y0-Round(r*sin(a*2*pi/360));

MoveTo(x,y);

for i:=1 **to** 5 **do****begin**

a:=a+36;

x:=x0+Round(r/2*cos(a*2*pi/360));

y:=y0-Round(r/2*sin(a*2*pi/360));

LineTo(x,y);

a:=a+36;

if a > 360 **then** a:=18;

x:=x0+Round(r*cos(a*2*pi/360));

```
        y:=y0-Round(r*sin(a*2*pi/360));
        LineTo(x,y);
    end;
    readln;
bye:
end.
```

Задача 202

```
{ Рисует пятиконечную звезду }
uses Graph;
label
    bye;
const
    k=0.01745; { коэф. пересчета величины угла из градусов
                в радианы k=2*p/360, где p - число "ПИ" }
var
    r: integer;      { радиус звезды }
    x0,y0: integer; { координаты центра звезды }

    p: array[1..10] of PointType; { координаты концов лучей
                                    и впадин звезды }

    a: integer;      { угол между осью OX и прямой, соединяющей
                        центр звезды и конец луча или впадину }

    i: integer;

    grDriver: Integer;
    grMode: Integer;
    ErrCode: Integer;
    res: integer;

begin
    grDriver := detect;
    InitGraph(grDriver, grMode,'e:\tp\bgi');
    ErrCode := GraphResult;
    if ErrCode <> grOk then
        begin
            writeln('Ошибка инициализации графического режима. ');
            goto bye;
        end;
    end;
```

```
x0:=100;
y0:=100;
r:=20;

a:=18; { строим от правого гор. луча }
for i:=1 to 10 do
begin
    if (i mod 2) <> 0 then
        begin { луч }
            p[i].x:=x0+Round(r*cos(a*k));
            p[i].y:=y0-Round(r*sin(a*k));
        end
    else
        begin { впадина }
            p[i].x:=x0+Round(r/2*cos(a*k));
            p[i].y:=y0-Round(r/2*sin(a*k));
        end;
    a:=a+36;
    if a > 360 then a:=18;
end;
SetFillStyle(SolidFill,Red);
FillPoly(10,p);
readln;
bye:
end.
```

Задача 204

```
{ Рисует российский флаг }
uses Graph;
var
    x,y: integer; { координаты левого верхнего угла флага }
    l,h: integer; { длина и высота флага }
    w: integer;   { ширина полосы флага }

    grDriver: Integer;
    grMode: Integer;
    ErrCode: Integer;
    res: integer;
begin
    grDriver := detect;
```

```
InitGraph(grDriver, grMode, 'e:\tp\bgi');
ErrCode := GraphResult;
if ErrCode = grOk then
begin
    x:=100;
    y:=100;
    l:=50;
    h:=25;
    w:=Round(h/3);

    { рисуем флаг }
    SetFillStyle(SolidFill,White);
    Bar(x, y, x+l, y+w);
    SetFillStyle(SolidFill,Blue);
    Bar(x, y+w, x+l, y+2*w);
    SetFillStyle(SolidFill,Red);
    Bar(x, y+2*w, x+l, y+3*w);
    OutTextXY(x, y+h+5, 'Россия');
end;
readln;
CloseGraph;
end.
```

Задача 205

```
{ Рисует веселую рожицу желтого цвета }
uses Graph;
var
    grDriver:integer; { драйвер }
    grMode:integer;   { графический режим }
    grPath:string;    { место расположения драйвера }
    ErrCode:integer;  { результат инициализации граф. режима }

begin
    grDriver := VGA;      { режим VGA }
    grMode:=VGAHi;        { разрешение 640x480 }
    grPath:='e:\tp\bgi'; { драйвер, файл EGAVGA.BGI, находится
                          в каталоге e:\tp\bgi }

    InitGraph(grDriver, grMode,grPath);
    ErrCode := GraphResult;
```

```

if ErrCode <> grOk then
  begin
    writeln('Ошибка инициализации графического режима. ');
    writeln('Для завершения работы программы ',
            'нажмите <Enter>');
    readln;
    Halt(1);
  end;

  SetFillStyle(SolidFill, Yellow);
  SetColor(Yellow);           { чтобы на круге не было линии }
  PieSlice(100, 100, 0, 360, 20);
  SetColor(Black);
  Arc(100, 102, 180, 360, 10); { рот }
  { глаза }
  Circle(93, 93, 2);
  Circle(107, 93, 2);
  readln;
  CloseGraph;
end.

```

Задача 208

```

{ Выводит узор из концентрических окружностей разного цвета }
Uses Graph;

var
  grDriver: integer;
  grMode: integer;
  grPath: string;
  ErrCode: integer;
  x, y, r: integer; { координаты центра и радиус окружности }
  dr: integer;      { приращение радиуса окружности }
  i: integer;       { счетчик циклов }

begin
  grDriver := VGA;
  grMode := VGAHi;
  grPath := 'e:\tp\bgi';
  InitGraph (grDriver, grMode, grPath);
  ErrCode := GraphResult;
  if ErrCode <> grOK then

```

```
begin
    writeln ('Ошибка инициализации графического режима. ');
    writeln ('Для завершения работы нажмите <Enter>');
    readln;
    Halt (1);
end;
x:=100; y:=100;
r:=5; dr:=5;
for i:=1 to 15 do
    begin
        SetColor(i);
        Circle(x,y,r);
        r:=r+dr;
    end;
readln;
end.
```

Задача 209

```
{ Вычерчивает узор из окружностей }
Uses Graph;
var
    grDriver:integer;
    grMode:integer;
    grPath:string;
    ErrCode:integer;
    x,y: integer;      { координаты центра окружности }
    r: integer;        { радиус окружности }
    l: integer;        { расстояние между центрами окружностей }
}
i,j: integer;         { счетчики циклов }
begin
    grDriver:=VGA;
    grMode:=VGAHi;
    grPath:='e:\tp\bgi';
    InitGraph (grDriver,grMode,grPath);
    ErrCode:=GraphResult;
    if ErrCode <> grOK then
        begin
            writeln ('Ошибка инициализации графического режима. ');
            writeln ('Для завершения работы нажмите <Enter>');
```

```

        readln;
        Halt (1);

    end;
y:=100;
r:=20;
l:=30;
for i:=1 to 4 do
begin
    x:=100;
    for j:=1 to 5 do
    begin
        Circle(x,y,r);
        x:=x+l;
    end;

    y:=y+l;
end;
readln;
end.

```

Задача 210

{ Вычерчивает узор из квадратов }

Uses Graph;

var

grDriver:integer;

grMode:integer;

grPath:string;

ErrCode:integer;

x,y: integer; { коорд. левого верхнего угла квадрата }

d: integer; { длина стороны квадрата }

n: integer; { кол-во квадратов в ряду }

l: integer; { расстояние между квадратами }

i,j: integer; { счетчики циклов }

begin

grDriver:=VGA;

grMode:=VGAHi;

grPath:='e:\tp\bgi';

InitGraph (grDriver,grMode,grPath);

ErrCode:=GraphResult;

```
if ErrCode <> grOK then
begin
    writeln ('Ошибка инициализации графического режима. ');
    writeln ('Для завершения работы нажмите <Enter>');
    readln;
    Halt (1);

end;
y:=100;
d:=30;
l:=10;
for i:=1 to 5 do
begin
    if ((i mod 2) = 1)
    then begin { нечетный ряд }
        n:=5; { пять квадратов в ряду }
        x:=100;
        end
    else begin { четный ряд }
        n:=4;
        x:=100+Round(d/2+l/2);
        end;
    for j:=1 to n do
    begin
        Rectangle(x,y,x+d,y+d);
        x:=x+d+l;
    end;
    y:=y+Round(d/2+l/2);
end;
readln;
end.
```

Задача 211

{ рисует на экране шахматную доску }

uses Graph;

var

```
x0,y0: integer; { координаты левого верхнего угла доски }
x,y: integer;    { координаты левого верхнего угла клетки }
w: integer;      { размер клетки }
i,j: integer;    { номер строки и колонки }
```

```
grDriver: Integer;
grMode: Integer;
ErrCode: Integer;
res: integer;

begin
  grDriver := detect;
  InitGraph(grDriver, grMode, 'e:\tp\bgi');
  ErrCode := GraphResult;
  if ErrCode = grOk then
  begin
    x0:=100;
    y0:=100;
    w:=25;
    x:=x0;
    y:=y0;
    for i:=1 to 8 do { восемь строк }
    begin
      for j:=1 to 8 do { восемь клеток в строке }
      begin
        { если сумма номера строки и номера колонки,
          на пересечении которых находится клетка, четная,
          то клетка - коричневая, иначе - желтая }
        if ((i+j) mod 2) = 0
        then SetFillStyle(SolidFill, Brown)
        else SetFillStyle(SolidFill, Yellow);
        Bar(x, y, x+w, y+w);
        x:=x+w;
      end;
      x:=x0;
      y:=y+w;
    end;
    readln;
  end;
  CloseGraph;
end.
```

Задача 212

```
{ Рисует флажок }
Uses Graph;
var
  grDriver: integer;
```

```
grMode:integer;
grPath:string;
ErrCode:integer;

flag: array[1..6] of PointType; { координаты точек флага }
begin
  grDriver:=VGA;
  grMode:=VGAHi;
  grPath:='e:\tp\bgi';
  InitGraph (grDriver,grMode,grPath);
  ErrCode:=GraphResult;
  if ErrCode <> grOK then
    begin
      writeln ('Ошибка инициализации графического режима. ');
      writeln ('Для завершения работы нажмите <Enter>');
      readln;
      Halt (1);
    end;
  { задать координаты контура - флага }
  flag[1].x:=100;flag[1].y:=100;
  flag[2].x:=160;flag[2].y:=100;
  flag[3].x:=140;flag[3].y:=120;
  flag[4].x:=160;flag[4].y:=140;
  flag[5].x:=100;flag[5].y:=140;
  flag[6].x:=100;flag[6].y:=100;
  SetFillStyle(SolidFill,Red);
  FillPoly(6,flag);
  Line(100,140,100,170);
  readln;
end.
```

Задача 213

```
{ Выводит на экран паровоз }
uses Graph;

var
  grDriver: Integer;
  grMode: Integer;
  ErrCode: Integer;
  res: integer;
```

```

x0,y0: integer; { координаты базовой точки паровоза }
dx,dy: integer; { шаг координатной сетки }
tr: array[1..15] of PointType; { координаты точек контура
                                паровоза }

```

begin

```

grDriver := detect;
InitGraph(grDriver, grMode, 'e:\tp\bgi');
ErrCode := GraphResult;
if ErrCode = grOk then
begin
    x0:=100; y0:=100;
    dx:=5; dy:=5;
    { корпус }
    tr[1].x:=x0+0*dx;   tr[1].y:=y0+7*dy;
    tr[2].x:=x0+0*dx;   tr[2].y:=y0+6*dy;
    tr[3].x:=x0+1*dx;   tr[3].y:=y0+6*dy;
    tr[4].x:=x0+1*dx;   tr[4].y:=y0+3*dy;
    tr[5].x:=x0+2*dx;   tr[5].y:=y0+3*dy;
    tr[6].x:=x0+2*dx;   tr[6].y:=y0+0*dy;
    tr[7].x:=x0+3*dx;   tr[7].y:=y0+0*dy;
    tr[8].x:=x0+3*dx;   tr[8].y:=y0+3*dy;
    tr[9].x:=x0+7*dx;   tr[9].y:=y0+3*dy;
    tr[10].x:=x0+7*dx;  tr[10].y:=y0+1*dy;
    tr[11].x:=x0+13*dx; tr[11].y:=y0+1*dy;
    tr[12].x:=x0+13*dx; tr[12].y:=y0+2*dy;
    tr[13].x:=x0+12*dx; tr[13].y:=y0+2*dy;
    tr[14].x:=x0+12*dx; tr[14].y:=y0+7*dy;
    tr[15].x:=x0+0*dx;  tr[15].y:=y0+7*dy;
    DrawPoly(15,tr);

    { окно }
    Rectangle(x0+8*dx,y0+2*dy,x0+10*dx,y0+4*dy);
    { колеса }
    SetFillStyle(SolidFill,Red);
    SetColor(Red);
    PieSlice(x0+3*dx,y0+7*dy,0,360,1*dx);
    PieSlice(x0+6*dx,y0+7*dy,0,360,1*dx);
    PieSlice(x0+9*dx,y0+7*dy,0,360,1*dx);
    { окантовка колес }
    SetColor(White);

```

```
Circle(x0+3*dx,y0+7*dy,1*dx);  
Circle(x0+6*dx,y0+7*dy,1*dx);  
Circle(x0+9*dx,y0+7*dy,1*dx);
```

```
readln;
```

```
end;
```

```
CloseGraph;
```

```
end.
```

Задача 215

```
{ Координатные оси и оцифрованная сетка }
```

```
program grid;
```

```
uses Graph;
```

```
var
```

```
  x0,y0:integer; { координаты начала координатных осей }
```

```
  dx,dy:integer; { шаг координатной сетки (в пикселах) }
```

```
  h,w:integer;   { высота и ширина области вывода координатной  
                  сетки }
```

```
  x,y:integer;
```

```
  lx,ly:real;     { метки (оцифровка) линий сетки по X Y }
```

```
  dlx,dly:real;   { шаг меток (оцифровки) линий сетки по X и Y }
```

```
  st:string;      { изображение метки линии сетки }
```

```
grDriver: Integer;
```

```
grMode: Integer;
```

```
ErrCode: Integer;
```

```
begin
```

```
  grDriver := VGA;
```

```
  grMode:=VGAHi;
```

```
  InitGraph(grDriver, grMode,'e:\tp\bgi');
```

```
  ErrCode := GraphResult;
```

```
  if ErrCode <> grOk then Halt(1);
```

```
  x0:=50; y0:=450; { оси начинаются в точке (40,450) }
```

```
  dx:=40; dy:=40;  { шаг координатной сетки 40 пикселей }
```

```
  dlx:=0.5;        { шаг меток оси X, метками будут: 0.5, 1.0,  
                    1.5 и т.д. }
```

```

dly:=1;           { шаг меток оси Y, метками будут: 1, 2, 3
                  и т.д. }

h:=300;
w:=400;

lx:=0;           { начало координат помечается метками 0 }
ly:=0;

Line(x0,y0,x0,y0-h); { ось X }
Line(x0,y0,x0+w,y0); { ось Y }

{ засечки, сетка и оцифровка по оси X }
x:=x0;
repeat
  { засечка }
  SetLineStyle(SolidLn, 0, NormWidth);
  Line(x,y0-3,x,y0+3);
  { оцифровка }
  Str(lx:0:1,st);
  OutTextXY(x-8,y0+5,st);
  lx:=lx+dly;
  { линия сетки }
  SetLineStyle(DottedLn, 0, NormWidth);
  Line(x,y0-3,x,y0-h);
  x:=x+dx;
until (x>x0+w);

{ засечки, сетка и оцифровка по оси Y }
y:=y0;
repeat
  { засечка }
  SetLineStyle(SolidLn, 0, NormWidth);
  Line(x0-3,y,x0+3,y);
  { оцифровка }
  Str(ly:0:1,st);
  OutTextXY(x0-40,y,st);
  ly:=ly+dly;
  { линия сетки }
  SetLineStyle(DottedLn, 0, NormWidth);
  Line(x0+3,y,x0+w,y);

```

```
SetLineStyle(SolidLn, 0, NormWidth);
y:=y-dy;
until (y<y0-h);

Readln;
CloseGraph;

end.
```

Задача 216

Uses Graph;

var

```
x,dx: real;      { аргумент и его приращение }
x1,x2: real;     { диапазон изменения аргумента }
y: real;         { значение функции }
mx,my: integer;  { масштаб по X и Y - кол-во точек
                  экрана, соответствующее единице
                  по осям координат }
x0,y0: integer;  { начало осей координат }
px,py: integer;  { координаты точек на экране }
```

```
grDriver:integer;
grMode:integer;
grPath:string;
ErrCode:integer;
i: integer;
```

begin

```
grDriver:=VGA;
grMode:=VGAHi;
grPath:='e:\tp\bgi';
InitGraph (grDriver,grMode,grPath);
ErrCode:=GraphResult;
if ErrCode <> grOK then
  begin
    writeln ('Ошибка инициализации графического режима.');
```

writeln ('Для завершения работы нажмите <Enter>');

readln;

Halt (1);

end;

```
x0:=320; y0:=240;
mx:=20; my:=20;
{ оси координат }
Line(10,y0,630,y0);
Line(x0,10,x0,470);
{ график }
x1:=-15;
x2:=5;
dx:=0.1;
x:=x1;
while (x<x2) do
begin
    y:= 0.5*x*x+x*4-3;
    px:=x0+Round(x*mx);
    py:=y0-Round(y*my);
    PutPixel(px,py,White);
    x:=x+dx;
end;
readln;
end.
```

Задача 217

```
{ Движущееся сложное изображение }
uses Graph,
      Crt;

var
  grDriver:integer; { драйвер }
  grMode:integer;   { графический режим }
  grPath:string;    { место расположения драйвера }
  ErrCode:integer;  { результат инициализации граф. режима }

  x,y:integer; { координаты кораблика }
  color:word;  { цвет кораблика }
  bkcolor:word; { цвет фона экрана }

{ Кораблик }
Procedure Titanik(x,y:integer; { координаты базовой точки }
                  color:word); { цвет корабля }

const
  dx=5;
  dy=5;
```

```
var
    OldColor:word;

begin
    OldColor:=GetColor; { сохранить текущий цвет }
    SetColor(color);     { установить новый цвет }

    { корпус }
    MoveTo(x,y);
    LineTo(x,y-2*dy);
    LineTo(x+10*dx,y-2*dy);
    LineTo(x+11*dx,y-3*dy);
    LineTo(x+17*dx,y-3*dy);
    LineTo(x+14*dx,y);
    LineTo(x,y);
    { надстройка }
    MoveTo(x+3*dx,y-2*dy);
    LineTo(x+4*dx,y-3*dy);
    LineTo(x+4*dx,y-4*dy);
    LineTo(x+13*dx,y-4*dy);
    LineTo(x+13*dx,y-3*dy);
    Line(x+5*dx,y-3*dy,x+9*dx,y-3*dy);
    { капитанский мостик }
    Rectangle(x+8*dx,y-4*dy,x+11*dx,y-5*dy);
    { труба }
    Rectangle(x+7*dx,y-4*dy,x+8*dx,y-7*dy);
    { иллюминаторы }
    Circle(x+12*dx,y-2*dy,Trunc(dx/2));
    Circle(x+14*dx,y-2*dy,Trunc(dx/2));
    { мачта }
    Line(x+10*dx,y-5*dy,x+10*dx,y-10*dy);
    { оснастка }
    MoveTo(x+17*dx,y-3*dy);
    LineTo(x+10*dx,y-10*dy);
    LineTo(x,y-2*dy);
    SetColor(OldColor); { восстановить текущий цвет }

end;
```

```
begin
    grDriver := VGA;      { режим VGA }
    grMode:=VGAHi;        { разрешение 640x480 }
    grPath:='e:\tp\bgi'; { драйвер, файл EGAVGA.BGI, находится
                          в каталоге d:\tp\bgi }
```

```
InitGraph(grDriver, grMode, grPath);
ErrCode := GraphResult;
if ErrCode <> grOk then Halt(1);

x:=10;
y:=200;
color:=LightGray;
SetBkColor(Blue);
bkcolor:=GetBkColor;
repeat
  Titanik(x,y,color);      { нарисовать корабль }
  Delay(100);
  Titanik(x,y,bkcolor);    { стереть корабль }
  PutPixel(x,y,color);    { след от корабля }
  x:=x+2;
until (x>500);
OutTextXY(10,10,'Рейс завершен!');
readln;
CloseGraph;
end.
```

Задача 219

```
{ Обрабатывает результаты контрольной работы
  и отражает их в виде диаграммы }
uses Crt, Graph;

{ возвращает изображение дробного числа }
function RealToStr(r: real; n,m: integer): string;
var
  st: string;
begin
  Str(r:n:m,st);
  RealToStr:=st;
end;

const
  { подсказка при вводе исходных данных и
    подпись рядом с прямоугольником легенды }
  mes: array[2..5] of string[10] =
    ('двоек', 'троек', 'четверок', 'пятерок');
```

var

```
n: array[2..5] of integer; { количество пятерок, четверок,  
                             троек и двоек }  
s: integer;                { всего оценок }  
p: array[2..5] of real;    { процент каждой оценки }  
h: array[2..5] of integer; { высоты столбиков диаграмм }
```

```
m: integer; { номер максимального эл-та массива n }  
i: integer; { индекс массива }
```

```
x,y: integer; { координаты левого нижнего угла столбика  
                диаграммы }
```

```
grDriver:integer; { драйвер }  
grMode:integer;   { графический режим }  
grPath:string;    { место расположения драйвера }  
ErrCode:integer;  { результат инициализации граф. режима }
```

begin

```
{ ввод исходных данных }  
TextBackground(Blue);  
TextColor(LightGray);  
ClrScr;  
writeln('Обработка результатов контрольной работы');  
writeln('Введите исходные данные:');
```

for i:=5 **downto** 2 **do****begin**

```
  write(mes[i], ' -> ');  
  readln(n[i]);
```

end;**for** i:=2 **to** 5 **do** s:=s+n[i]; { всего оценок }

{ вычислим процент каждой оценки }

for i:=2 **to** 5 **do** p[i]:=(n[i]/s)*100;

{ вычислим высоту каждого столбика диаграммы, }

{ но сначала определим, каких оценок больше }

m:=5; { пусть больше всего пятерок }

for i:=4 **downto** 2 **do if** n[i]>n[m] **then** m:=i;

```

{ Пусть количеству оценок, которых больше,
  соответствует столбик высотой 200 пикселей.
  Вычислим высоты остальных столбиков. }
for i:=5 downto 2 do
    h[i]:=Round( (200/n[m])*n[i] );

{ обработка выполнена, строим диаграмму }
grDriver := VGA;      { режим VGA }
grMode:=VGAHi;        { разрешение 640x480 }
grPath:='e:\tp\bgi'; { драйвер, файл EGAVGA.BGI, находится
                      в каталоге e:\tp\bgi }
InitGraph(grDriver, grMode,grPath);
ErrCode := GraphResult;
if ErrCode <> grOk then
    begin
        writeln('Ошибка инициализации графического режима. ');
        writeln('Для завершения работы программы ',
                'нажмите <Enter>');
        readln;
        Halt(1);
    end;

{ строим диаграмму }
OuttextXY(40,50,'Результаты контрольной работы');
Rectangle(40,80,170,310);
x:=50; y:=300; { левый нижний угол первого столбика }
{ столбики диаграммы }
for i:=5 downto 2 do
    begin
        SetFillStyle(SolidFill,i);
        Bar(x,y,x+10,y-h[i]); { столбик }
        { OutTextXY(x,y-h[i]-10,RealToStr(p[i],5,2)+'%'); }
        x:=x+20;
    end;

{ численные значения }
x:=50;
for i:=5 downto 2 do
    begin
        SetFillStyle(SolidFill,i);
        {Bar(x,y,x+10,y-h[i]); { столбик }
        OutTextXY(x,y-h[i]-10,RealToStr(p[i],5,1)+'%');
        x:=x+20;
    end;

```

```

{ легенда }
x:=200;y:=100;
for i:=5 downto 2 do
  begin
    SetFillStyle(SolidFill,i);
    Bar(x,y,x+20,y+10); { столбик }
    OutTextXY(x+25,y,mes[i]);
    y:=y+20;
  end;
readln;
CloseGraph;
end.

```

Задача 220

```

{ Выводит круговую диаграмму }
uses Graph;
const
  N=4; { количество категорий }
  name: array[1..N] of
string[10]=('Книги','Журналы','Канцтовары','Прочее');
var
  kol: array[1..N] of real; { количество для категории }
  dol: array[1..N] of real; { доля категории в общем
                               количестве }
  sum: real; { общее кол-во по всем категориям }
  a1,a2: integer; { угол начала и конца сектора }
  x,y: integer; { координаты вывода легенды }
  st: string; { изображение числа }
  i: integer;

  grDriver:integer; { драйвер }
  grMode:integer; { графический режим }
  grPath:string; { место расположения драйвера }
  ErrCode:integer; { результат инициализации граф. режима }

begin
  grDriver := VGA; { режим VGA }
  grMode:=VGAHi; { разрешение 640x480 }
  grPath:='e:\tp\bgi'; { драйвер, файл EGAVGA.BGI, находится
                        в каталоге e:\tp\bgi }

```

```

InitGraph(grDriver, grMode,grPath);
ErrCode := GraphResult;
if ErrCode <> grOk then
    begin
        writeln('Ошибка инициализации графического режима. ');
        writeln('Для завершения работы программы ',
            'нажмите <Enter>');
        readln;
        Halt(1);
    end;

{ ввод исходных данных }
writeln('Введите количество по каждой категории');
sum:=0;
for i:=1 to N do
    begin
        write(name[i], ' ->');
        readln(kol[i]);
        sum:=sum+kol[i];
    end;

{ вычислим долю каждой категории в общей сумме }
for i:=1 to N do
    dol[i]:=kol[i]/sum*100;

{ строим диаграмму }
al:=0; { от оси OX }
x:=350; y:=100; { левый верхний угол области легенды }
for i:=1 to N do
    begin
        { сектор }
        a2:=al+Round(3.6*Dol[i]); { 1% - 3.6 градуса }
        if a2 > 360 then a2:=360;
        SetFillstyle(SolidFill,i);
        PieSlice(200,200,a1,a2,100);
        al:=a2; { следующий сектор - от конца текущего }
        { легенда }
        Bar(x,y,x+30,y+10);
        Rectangle(x,y,x+30,y+10);
        Str(dol[i]:6:1,st);
        OutTextXY(x+50,y,name[i]+' -'+st+'%');
        y:=y+20;
    end;

```

```
readln;  
CloseGraph;  
end.
```

Задача 221

```
{ Светофор }  
uses Graph, Crt;  
  
var  
    grDriver: Integer;  
    grMode: Integer;  
    ErrCode: Integer;  
    res: integer;  
  
    i,j: integer; { счетчики циклов }  
  
{ Рисует круг заданного цвета }  
{ x,y,r - координаты центра и радиус круга }  
{ fc,bc - цвет круга и окантовки }  
Procedure Krug(x,y,r: integer; fc,bc: integer);  
begin  
    SetFillStyle(SolidFill,fc);  
    SetColor(fc);  
    PieSlice(x,y,0,360,r);  
    SetColor(bc);  
    Circle(x,y,r);  
end;  
  
{ Основная программа }  
begin  
    grDriver := detect;  
    InitGraph(grDriver, grMode,'e:\tp\bgi');  
    ErrCode := GraphResult;  
    if ErrCode = grOk then  
    begin  
        OutTextXY(10,10,'Соблюдайте правила уличного движения!');  
        Rectangle(88,88,112,152);  
        { Горит красный свет }  
        Krug(100,100,10,Red,White);  
        Krug(100,120,10,LightGray,White);  
        Krug(100,140,10,LightGray,White);
```

```

for i:=1 to 3 do
  begin
    { Здесь горит красный }
    Delay(3000); { задержка 3 сек }
    Krug(100,120,10,Yellow,White); { Включить
                                     желтый }

    Delay(1000);
    Krug(100,100,10,LightGray,White); { Выключить
                                         красный }

    Krug(100,120,10,LightGray,White); { Выключить
                                         желтый }

    Krug(100,140,10,Green,White);      { Включить
                                         зеленый }

    Delay(2000);
    { Мигающий зеленый сигнал }
    for j:=1 to 3 do { мигает 5 раз }
      begin
        Delay(500);
        Krug(100,140,10,Green,White);      { Включить
                                              зеленый }

        Delay(500);
        Krug(100,140,10,LightGray,White); { Выключить
                                              зеленый }

      end;

    Krug(100,120,10,Yellow,White);      { Включить
                                         желтый }

    Delay(1500);
    Krug(100,120,10,LightGray,White);    { Выключить
                                         желтый }

    Krug(100,100,10,Red,White);          { Включить
                                         красный }

  end;
end;
CloseGraph;
end.

```

Задача 222

```

{ Часы с минутной и секундной стрелками }
uses Graph, Crt;

{ Выводит вектор заданной длины из текущей точки }
{ Используется для вывода изображения стрелки }

```

```
procedure Vector(a: integer; { угол между вектором  
                             и осью OX }  
                l: integer); { длина вектора }  
  
const  
    GRAD=0.0174532; { коэф. пересчета величины угла  
                     из градусов в радианы }  
  
var  
    x0,y0: integer; { координаты начала вектора }  
    x1,y1: integer; { координаты конца вектора }  
  
begin  
    x0:=GetX;  
    y0:=GetY;  
    x1:=Round(x0+l*cos(a*GRAD));  
    y1:=Round(y0-l*sin(a*GRAD));  
    LineTo(x1,y1);  
  
end;  
  
{ главная процедура }  
  
var  
    grDriver: Integer;  
    grMode: Integer;  
    ErrCode: Integer;  
    res: integer;  
    x0,y0: integer; { координаты центра часов }  
    d: integer;      { диаметр циферблата }  
    s: integer;      { время, кол-во секунд }  
    m: integer;      { время, кол-во минут }  
    as: integer;     { угол наклона секундной стрелки }  
    am: integer;     { угол наклона минутной стрелки }  
    sts: string;     { кол-во секунд }  
    stm: string;     { кол-во минут }  
  
begin  
    grDriver := detect;  
    InitGraph(grDriver, grMode, 'e:\tp\bgi');  
    ErrCode := GraphResult;  
    if ErrCode = grOk then  
    begin  
        x0:=80; y0:=80;  
        d:=50;  
        s:=0;  
        m:=0;
```

```
{ начальное положение стрелок: угол
  между стрелкой и осью OX равен 90 град. }
as:=90;
am:=90;

Circle(x0,y0,d+5);
SetFillStyle(SolidFill,0);
Str(m,stm);
repeat
  { вывести секундную стрелку }
  MoveTo(x0,y0);
  SetColor(Yellow);
  Vector(as,d);

  { вывести минутную стрелку }
  MoveTo(x0,y0);
  SetColor(Green);
  Vector(am,d-10);

  { вывести "цифровые" часы }
  Bar(10,10,50,20);
  Str(s,sts);
  OutTextXy(10,10,stm+':'+sts);

  Delay(100); { задержка }

  { стереть стрелки }
  SetColor(0);
  { секундную }
  MoveTo(x0,y0);
  Vector(as,d);

  { минутную }
  MoveTo(x0,y0);
  Vector(am,d-10);

  s:=s+1;
if s = 60 then
begin
  m:=m+1;
  Str(m,stm);
  s:=0;
```

```
Sound(1000);  
Delay(10);  
NoSound;
```

```
am:=am-6; { шаг движения минутной  
          стрелки 6 градусов }  
if am < 0 then am:=354;  
end;
```

```
as:=as-6;  
if as < 0 then as:=354;
```

```
until KeyPressed;  
end;  
CloseGraph;  
end.
```

Задача 223

```
{ Построение графика функции }  
program groffunc;  
uses Graph;  
  
var  
  x1,x2:real;      { границы изменения аргумента функции }  
  y1,y2:real;      { границы изменения значения функции }  
  x:real;           { аргумент функции }  
  y:real;           { значение функции в точке x }  
  dx:real;          { приращение аргумента }  
  l,b:integer;      { левый нижний угол области вывода графика }  
  w,h:integer;      { ширина и высота области вывода графика }  
  mx,my:real;       { масштаб по осям X и Y }  
  x0,y0:integer;    { точка - начало координат }  
  st:string;       { изображение числа }  
  
  grDriver: Integer;  
  grMode: Integer;  
  ErrCode: Integer;  
  
{ Функция, график которой надо построить }  
Function f(x:real):real;
```

begin

f:=2*Sin(x)*exp(x/5);

end;

Function f2(x:real):real;

begin

f2:=Ln(x);

end;**begin**

grDriver := VGA;

grMode:=VGAHi;

InitGraph(grDriver, grMode,'e:\tp\bgi');

ErrCode := GraphResult;

if ErrCode <> grOk **then** Halt(1);

l:=40;

b:=400;

h:=200;

w:=200;

x1:=0;

x2:=25;

dx:=0.01;

{ найдем максимальное и минимальное значения
функции на отрезке [x1,x2] }

y1:=f(x1); { минимум }

y2:=f(x1); { максимум }

x:=x1;

repeat

y:=f(x);

if y<y1 **then** y1:=y;**if** y>y2 **then** y2:=y;

x:=x+dx;

until (x>=x2);

my:=h/abs(y2-y1);

mx:=w/abs(x2-x1);

{ оси }

x0:=l;

y0:=b-Abs(Round(y1*my));

```
Line(l,b,l,b-h);
Line(x0,y0,x0+w,y0);
Str(y2:5:1,st);
OutTextXY(l+5,b-h,st);
Str(y1:5:1,st);
OutTextXY(l+5,b,st);

{ построение графика }
x:=x1;
repeat
  y:=f(x);
  PutPixel(x0+Round(x*mx),y0-Round(y*my),15);
  x:=x+dx;
until (x>=x2);

Readln;
CloseGraph;

end.
```

Задача 224

```
uses Graph,Crt;

{ в графическом режиме вводит с клавиатуры дробное число }
Function GetReal: real;

var
  ch: char;           { символ нажатой клавиши }
  buf: string[80];    { введенная строка }
  numb: real;         { введенное число }
  code: integer;      { код ошибки преобразования строки
                      в число }

begin
  buf:='';
  repeat
    { ждем нажатия клавиши, курсор мигает }
  repeat
    if not KeyPressed then
      begin
        Setcolor(White);
        Line(GetX,GetY,GetX,GetY+8);
        Delay(250);
```

```

        Setcolor (Black);
        Line (GetX, GetY, GetX, GetY+8);
    end;
    if not KeyPressed then Delay(250);
until KeyPressed;
{ здесь нажата клавиша }
ch:=ReadKey;
SetColor(White);
case ch of
    '0'..'9': begin Outtext(ch); buf:=buf+ch; end;
    '.': if Pos('.',Buf) = 0 then begin Outtext(ch);
        buf:=buf+ch; end;
    '-': if Length(buf) = 0 then begin Outtext(ch);
        buf:=ch; end;
    Chr(8): if Length(buf) <> 0 then { нажата <BackSpace> }
        begin
            SetFillStyle (SolidFill, Black);
            Bar (GetX, GetY, GetX-8, GetY+8);
            MoveTo (GetX-8, GetY);
            Delete (Buf, Length (buf), 1);
        end;
end;
until ch=Chr(13);
Val (buf, numb, code);
GetReal:=numb;
end;
```

var

```

grDriver:integer; { драйвер }
grMode:integer;   { графический режим }
grPath:string;    { место расположения драйвера }
ErrCode:integer;  { результат инициализации граф. режима }
```

```

n: real; { дробное число }
```

begin

```

grDriver := VGA;      { режим VGA}
grMode:=VGAHi;        { разрешение 640x480}
grPath:='e:\tp\bgi'; { драйвер, файл EGAVGA.BGI, находится
                     в каталоге e:\tp\bgi }
```

```
InitGraph(grDriver, grMode,grPath);
ErrCode := GraphResult;
if ErrCode <> grOk then
    begin
        writeln('Ошибка инициализации графического режима. ');
        writeln('Для завершения работы программы нажмите  
      <Enter>');
        readln;
        Halt(1);
    end;
outtext('Введите целое число -> ');
n:=GetReal;
readln;
CloseGraph;
end.
```

Задача 225

```
{ Создает на диске А: файл и записывает в него
  5 целых чисел, введенных пользователем }
var
    f: text;      { текстовый файл }
    n: integer; { число }
    i: integer; { счетчик чисел }
begin
    writeln('Создание файла');
    writeln('Введите пять целых чисел. ');
    writeln('После ввода каждого числа нажимайте <Enter>');
    Assign(f,'a:\numbers.txt');
    Rewrite(f); { открыть в режиме перезаписи }
    for i:=1 to 5 do
        begin
            write('->');
            readln(n);
            writeln(f,n);
        end;
    close(f);      { закрыть файл }
    writeln('Введенные числа записаны в файл ',
            'a:\numbers.txt');
    readln;
end.
```

Задача 226

```
{ Дописывает в файл a:\numbers.txt
   пять введенных пользователем целых чисел }

var
    f: text;      { текстовый файл }
    n: integer;   { число }
    i: integer;   { счетчик чисел }

begin
    writeln('Добавление в файл a:\numbers.txt');
    writeln('Введите пять целых чисел. ');
    writeln('После ввода каждого числа нажимайте <Enter>');
    Assign(f, 'a:\numbers.txt');
    Append(f);    { открыть файл в режиме добавления }
    for i:=1 to 5 do
        begin
            write('->');
            readln(n);
            writeln(f, n);
        end;
    Close(f);     { закрыть файл }
    writeln('Введенные числа добавлены в файл ',
            'a:\numbers.txt');
    readln;
end.
```

Задача 227

```
{ Выводит на экран содержимое файла a:\numbers.txt }

var
    f: text;      { текстовый файл }
    n: integer;   { число }

begin
    writeln('Содержимое файла a:\numbers.txt');
    writeln('-----');
    Assign(f, 'a:\numbers.txt');
    Reset(f);     { открыть файл для чтения }
    While not EOF(f) do { пока не достигнут конец файла }
        begin
            readln(f, n); { прочитать число из файла }
            writeln(n);   { вывести прочитанное число на экран }
        end;
end;
```

```
Close(f);      { закрыть файл }
writeln('-----');
readln;
```

end.

Задача 228

```
{ Вычисляет среднее арифметическое чисел,
  находящихся в файле a:\numbers.txt }

var
  f: text;      { текстовый файл }
  n: integer;   { число, прочитанное из файла }
  kol: integer; { кол-во прочитанных чисел }
  sum: integer; { сумма прочитанных чисел }
  sa: real;     { среднее арифметическое }

begin
  writeln('Вычисление среднего арифметического чисел,');
  writeln('находящихся в файле a:\numbers.txt');
  writeln('Чтение из файла. Подождите.');
```

sum:=0;

kol:=0;

Assign(f,'a:\numbers.txt');

Reset(f); { открыть файл для чтения }

While **not** EOF(f) **do** { пока не достигнут конец файла }

begin

 readln(f,n); { прочитать число из файла }

 sum:=sum+n;

 kol:=kol+1;

end;

Close(f); { закрыть файл }

sa:=sum/kol;

writeln('Прочитано чисел: ',kol);

writeln('Сумма чисел: ',sum);

writeln('Среднее арифметическое: ',sa:9:2);

readln;

end.

Задача 229

```
{ Выводит на экран содержимое файла, имя которого
  указано пользователем }

uses Crt;
```

```
var
    f: text;           { текстовый файл }
    fname: string[80]; { имя файла }
    st: string;        { строка, прочитанная из файла }
    nst: integer;      { кол-во выведенных на экран строк }
    key: char;         { клавиша, нажатая пользователем }

begin
    ClrScr; { очистить экран }
    writeln('Просмотр текстового файла');
    writeln('Введите полное имя файла и нажмите <Enter>');
    write('-> ');
    readln(fname);
    Assign(f, fname);
    Reset(f); { открыть файл для чтения }
    ClrScr;
    nst:=0;
    While not EOF(f) do { пока не достигнут конец файла }
        begin
            readln(f, st); { прочитать число из файла }
            writeln(st);
            nst:=nst+1;
            if nst = 23 then { выведены очередные 23 строки }
                begin
                    writeln;
                    write('Для продолжения вывода ',
                        'нажмите любую клавишу...');
                    key:=Readkey;
                    GotoXY(1, WhereY); { курсор в начало
                                         текущей строки }
                    DelLine; { удалить сообщение
                               "Для продолжения ..." }
                    nst:=0;
                end;
            end;
        end;
    Close(f); { закрыть файл }
    writeln;
    write('Для завершения просмотра нажмите любую ',
        'клавишу...');
    key:=Readkey;
end.
```

Задача 230

```
{ Дописывает в файл a:\phone.txt
  фамилию, имя и номер телефона.
  Если файла на диске нет, то создает его. }

label
  bye;

var
  f: text;      { текстовый файл }
  fam: string[15]; { фамилия }
  name: string[15]; { имя }
  tel: string[9];  { номер телефона }

begin
  writeln('Добавление в телефонный справочник');
  Assign(f, 'a:\phone.txt');
  {$I-}
  Append(f);      { сначала откроем в режиме добавления }
  if IOResult <> 0 then
    { вероятно файла phone.txt нет на диске A: }
    { создадим его }
    begin
      Rewrite(f);  { открыть в режиме перезаписи }
      if IOResult <> 0 then
        begin
          writeln('Ошибка обращения к диску A:');
          goto bye;
        end;
      end;
    { получим данные от пользователя }
    write('Фамилия ->');
    readln(fam);
    write('Имя ->');
    readln(name);
    write('Телефон ->');
    readln(tel);
    { и запишем их в файл }
    writeln(f, fam);
    writeln(f, name);
    writeln(f, tel);
    close(f);
    writeln('Информация добавлена.');
```

```
bye: writeln('Для завершения работы ',
            'нажмите <Enter>.');
      readln;
end.
```

Задача 232

```
{ Поиск в телефонном справочнике }
label
  bye;
var
  f: text;           { текстовый файл }
  obr: string[15];   { фамилия для поиска }
  n: integer;        { кол-во записей, удовлетворяющих
                     { запросу }

  fam: string[15];   { фамилия }
  name: string[15];  { имя }
  tel: string[9];    { номер телефона }
begin
  writeln('Поиск в телефонном справочнике');
  Assign(f, 'a:\phone.txt');
  {$I-}
  reset(f);          { откроем файл для чтения }
  if IOResult <> 0 then
    { вероятно файла phone.txt на диске A: нет }
    begin
      writeln('Файл a:\phone.txt не доступен');
      goto bye;
    end;
  repeat
    write('ФАМИЛИЯ ->');
    readln(obr);
    if Length(obr) <> 0 then { пользователь ввел строку }
      begin
        reset(f); { просматриваем файл от начала }
        n:=0;
        while not EOF(f) do { просматриваем весь файл }
          begin
            { читаем из файла }
            readln(f, fam);
```

```
    readln(f,name);
    readln(f,tel);
    if fam = obr then { найдена нужная фамилия }
        begin
            writeln(fam,' ',name,' ',tel);
            n:=n+1;
        end;
    end;
    if n = 0 then writeln('Сведений о ' ,
                        obr,' нет.');
```

end;

```
until Length(obr) = 0;
bye:
    writeln('Для завершения работы нажмите <Enter>.');
    readln;
end.
```

Задача 234

```
{ Универсальная программа тестирования }
uses Crt;

label
    bye;

var
    fname: string[40]; { имя файла теста }
    f: text; { файл теста }

    VsegoVopr: integer; { количество вопросов теста }
    PravOtv: integer; { количество правильных ответов }

    { для текущего вопроса }
    nOtv: integer; { количество альтернативных ответов }
    prav: integer; { номер правильного ответа }
    Otv: integer; { номер ответа, выбранного пользователем }

    st: string; { строка, читаемая из файла теста }
    p: integer; { процент правильных ответов }

    i: integer; { счетчик циклов }
```

```
begin
```

```
  if ParamCount = 0 then
```

```
    begin
```

```
      writeln('Не задан файл вопросов теста!');
```

```
      writeln('Командная строка: test ИмяФайлаТеста');
```

```
      goto bye;
```

```
    end;
```

```
  fname:=ParamStr(1); { имя файла из командной строки }
```

```
  Assign(f, fname);
```

```
  {$I-}           { это директива компилятору, а не комментариев! }
```

```
  Reset(f);      { открыть файл для чтения }
```

```
  {$I+}
```

```
  if IOResult <> 0 then
```

```
    begin
```

```
      writeln('Не найден файл теста ', fname);
```

```
      goto bye;
```

```
    end;
```

```
writeln('Сейчас Вам будет предложен тест.');
```

```
writeln('К каждому вопросу дается несколько вариантов ',  
        'ответа.');
```

```
writeln('Вы должны ввести номер правильного ответа',  
        'и нажать клавишу <Enter>');
```

```
writeln;
```

```
writeln('Удачи! Для начала тестирования нажмите <Enter>');
```

```
TextBackGround(Blue);
```

```
ClrScr;
```

```
VsegoVopr:=0;
```

```
Prav:=0;
```

```
while not EOF(f) do
```

```
  begin
```

```
    {ClrScr;}
```

```
    VsegoVopr:=VsegoVopr+1;
```

```
    readln(f,st);      { читаем из файла вопрос }
```

```
    TextColor(White);
```

```
    writeln(st);      { выводим вопрос на экран }
```

```
    readln(f,nOtv,Prav); { читаем кол-во альтернативных  
                          ответов и номер правильного  
                          ответа }
```

```
    TextColor(LightGray);
```

```
    for i:=1 to nOtv do { читаем и выводим альтернативные  
                        ответы }
```

```
begin
    readln(f,st);
    writeln(i, '. ',st);
end;
writeln;
write('Ваш выбор ->');
readln(Otv);
if Otv = Prav then PravOtv:=PravOtv+1;
writeln;
end;
{ обработка результата тестирования }
{ вычислим процент правильных ответов }
p:=Round((PravOtv/VsegoVopr)*100);
write('Ваша оценка ');
case p of
    100:writeln('ОТЛИЧНО!');
    80..99:writeln('ХОРОШО. ');
    60..79:writeln('УДОВЛЕТВОРИТЕЛЬНО. ');
    else writeln('ПЛОХО!');
end;
bye:
write('Для завершения работы программы нажмите <Enter>');
readln;
end.
```

Задача 235

```
{
    Выводит таблицу пересчета из дюймов в миллиметры
    на экран, принтер или в файл
}
var
    f:text; { файл вывода }
    fname:string; { имя файла вывода }
    dest:integer; { 1 - на экран, 2 - на принтер,
                    3 - в файл }

    d: real; { величина в дюймах }
    m: real; { величина в миллиметрах }

begin
    writeln('*** Таблица пересчета из дюймов ',
            'в миллиметры ***');
```

```
writeln('Результат выводить:');
writeln('1 - на экран;');
writeln('2 - на принтер;');
writeln('3 - в файл.');
```

Введите число от 1 до 3 и нажмите <Enter> ');

```
write('Ваш выбор -> ');
readln(dest);
case dest of
    1: fname:='';      { на экран }
    2: begin           { на принтер }
        fname:='prn';
        write('Включите принтер и нажмите <Enter>');
        readln;
    end;
    3: begin           { в файл }
        write('Задайте имя файла для вывода -> ');
        readln(fname);
    end;
end;
assign(f,fname);
rewrite(f);
writeln(f, ' -----');
writeln(f, '   Дюймы   Миллиметры');
writeln(f, ' -----');
d:=0.5;
while d < 10 do
    begin
        m:=25.4*d; { 1 дюйм - 25,4 мм }
        writeln(f,d:6:1,m:10:1);
        d:=d+0.5;
    end;
writeln(f, ' -----');
close(f);
if dest = 3
    then writeln('Таблица записана в файл ',fname);

writeln('Для завершения работы программы',
        'нажмите <Enter>');
readln;
end.
```

Задача 236

```
{ Рекурсивная функция "Факториал" }  
function factorial(k:integer):integer;  
  begin  
    if k = 1  
      then factorial:=1  
      else factorial:= k*factorial(k-1);  
  end;  
  
var  
  n: integer; { число, факториал которого надо вычислить }  
  f: integer; { факториал числа n }  
begin  
  writeln('Вычисление факториала. ');  
  writeln('Введите число, факториал которого надо ',  
    'вычислить');  
  write('->');  
  readln(n);  
  f:=factorial(n);  
  writeln('Факториал числа ',n,' равен ',f);  
  readln;  
end.
```

Задача 237

```
{ Выводит на экран узор }  
Uses Graph,Crt;  
  
{ Рисует элемент узора }  
procedure Elem(x,y,r,p: integer);  
  { x,y,r - координаты и радиус центра основного  
    элемента узора  
    p - порядок узора }  
begin  
  if p>=0 then  
    begin  
      Circle(x,y,r);  
      Delay(100);  
      Elem(x+r,y,Round(r/2),p-1);  
      Elem(x,y-r,Round(r/2),p-1);  
      Elem(x-r,y,Round(r/2),p-1);
```

```

        Elem(x,y+r,Round(r/2),p-1);
    end;
end;

var
    grDriver:integer; { драйвер }
    grMode:integer;   { графический режим }
    grPath:string;    { путь к файлу драйвера }
    ErrCode:integer;  { код ошибки графического режима }

begin
    grDriver:=VGA;
    grMode:=VGAHi;
    grPath:='e:\tp\bgi';
    InitGraph (grDriver,grMode,grPath);
    ErrCode:=GraphResult;
    if ErrCode <> grOK then
        begin
            writeln ('Ошибка инициализации графического режима. ');
            writeln ('Для завершения работы нажмите <Enter> ');
            readln;
            Halt (1);
        end;

    { основная программа }
    Elem(320,240,60,3);      { рисуем узор 3-го порядка }
    OutText('Для завершения работы программы ',
            'нажмите <Enter> ');
    readln;
end.

```

Задача 238

{ Используя механизм рекурсии, вычисляет сопротивление n-звенной электрической цепи }

```

var
    r1,r2,r3: real; { величины сопротивлений,
                    из которых состоит цепь }
    n: integer;     { количество звеньев (порядок) цепи }
    rc: real;       { сопротивление цепи }

```

```
{ величина сопротивления цепи n-го порядка }
function Cep(n: integer): real;
begin
    if n=1
    then Cep:=R1+R2+R3
    else begin
        rc:=Cep(n-1);
        Cep:=R2*rc/(R2+rc);
    end;
end;

{ основная процедура }
begin
    writeln('Вычисление сопротивления электрической цепи. ');
    writeln('Введите величины сопротивлений (Ом): ');
    write('r1 -> ');
    readln(r1);
    write('r2 -> ');
    readln(r2);
    write('r3 -> ');
    readln(r3);
    write('Порядок цепи -> ');
    readln(n);

    writeln('Сопротивление цепи: ', Cep(n):6:2, ' Ом');
    { величины сопротивлений передаются
      в процедуру Cep через глобальные
      переменные r1, r2 и r3 }
    readln;
end.
```

Задача 239

```
{ Вычерчивает схему сложной электрической цепи. }
{ При вычерчивании схемы цепи используется рекурсия. }
Uses Graph;
const
    { шаг сетки }
    dx=7;
    dy=7;
```

```
var
  grDriver:integer;
  grMode:integer;
  grPath:string;
  ErrCode:integer;
  x,y: integer;

  k: integer; { порядок цепи }

{ выводит схему цепи k-го порядка }
Procedure Cep(k: integer; x,y: integer);
begin
  SetColor(Green);
  Line(x,y,x+2*dx,y);
  Rectangle(x+2*dx,y-dy,x+6*dx,y+dy);
  Line(x+6*dx,y,x+8*dx,y);
  OuttextXY(x+3*dx,y-3*dy,'R1');
  SetColor(Yellow);
  Line(x+8*dx,y,x+8*dx,y+2*dy);
  Rectangle(x+7*dx,y+2*dy,x+9*dx,y+6*dy);
  Line(x+8*dx,y+6*dy,x+8*dx,y+8*dy);
  OuttextXY(x+10*dx,y+2*dy,'R2');
  SetColor(LightGray);
  Line(x,y+8*dy,x+2*dx,y+8*dy);
  Rectangle(x+2*dx,y+7*dy,x+6*dx,y+9*dy);
  Line(x+6*dx,y+8*dy,x+8*dx,y+8*dy);
  OuttextXY(x+3*dx,y+5*dy,'R3');

  if k>1 then Cep(k-1,x+8*dx,y);
end;

begin
  grDriver:=VGA;
  grMode:=VGAHi;
  grPath:='e:\tp\bgi';
  InitGraph (grDriver,grMode,grPath);
  ErrCode:=GraphResult;
  if ErrCode <> grOK then
    begin
      writeln ('Ошибка инициализации графического режима.');
```

Для завершения работы нажмите <Enter>');

```
        readln;
        Halt (1);
    end;
```

```
OutTextXY(10,10,'Введите порядок цепи и нажмите <Enter>');
readln(k);
cep(k,10,50);
readln;
CloseGraph;
end.
```

Задача 240

```
{ Демонстрация понятия "рекурсия".
  Программа строит кривую Гильберта. }

uses Graph, Crt;

var
    u: integer; { Длина штриха кривой Гильберта }

    { Заменяет стандартную процедуру для вычерчивания
      по точкам горизонтальных и вертикальных линий. }

procedure LineTo(x2,y2: integer);
const
    DT = 3; { задержка между выводом точек линии }
var
    x1,y1: integer; { координаты начала прямой,
                     x2,y2 - координаты конца }
    x,y : integer; { координаты текущей точки }
    dx: integer; { приращение аргумента }
    dy: integer; { приращение у при рисовании
                  вертикальной линии }
    color: integer; { цвет линии }
    a,b: real; { коэф-ты уравнения прямой }
    n: integer; { кол-во точек }
    i: integer;

begin
    x1:=GetX;
    y1:=GetY;
```

```
if x1 <> x2
then begin
    { не вертикальная линия }
    a:=(y2-y1)/(x2-x1);
    b:=y1-a*x1;
    n:=abs(x2-x1)+1;
    if x2 > x1
    then dx:=1
    else dx:=-1;
    x:=x1;
    color:=GetColor;
    for i:=1 to n do
    begin
        y:=Round(a*x+b);
        PutPixel(x,y,color);
        delay(DT);
        x:=x+dx;
    end;
end
else begin { вертикальная линия }
    n:=abs(y2-y1);
    if y2 > y1
    then dy:=1
    else dy:=-1;
    x:=x1;
    y:=y1;
    color:=GetColor;
    for i:=1 to n do
    begin
        PutPixel(x,y,color);
        delay(DT);
        y:=y+dy;
    end;
end;
PutPixel(x2,y2,color);
MoveTo(x2,y2);
end;

{ Кривая состоит из четырех элементов: a,b,c и d.
  Каждый элемент строит соответствующая процедура. }
procedure a(i:integer); external;
```

```
procedure b(i:integer); external;  
procedure c(i:integer); external;  
procedure d(i:integer); external;
```

```
{ Элементы кривой. }
```

```
procedure a(i: integer);  
begin  
  if i > 0 then  
  begin  
    d(i-1); LineTo(GetX-u,GetY);  
    a(i-1); LineTo(GetX,GetY+u);  
    a(i-1); LineTo(GetX+u,GetY);  
    b(i-1);  
  end;  
end;
```

```
procedure b(i: integer);  
begin  
  if i > 0 then  
  begin  
    c(i-1); LineTo(GetX,GetY-u);  
    b(i-1); LineTo(GetX+u,GetY);  
    b(i-1); LineTo(GetX,GetY+u);  
    a(i-1);  
  end;  
end;
```

```
procedure c(i: integer);  
begin  
  if i > 0 then  
  begin  
    b(i-1); LineTo(GetX+u,GetY);  
    c(i-1); LineTo(GetX,GetY-u);  
    c(i-1); LineTo(GetX-u,GetY);  
    d(i-1);  
  end;  
end;
```

```
procedure d(i: integer);  
begin  
  if i > 0 then
```

```
begin
    a(i-1); LineTo(GetX,GetY+u);
    d(i-1); LineTo(GetX-u,GetY);
    d(i-1); LineTo(GetX,GetY-u);
    c(i-1);
end;
end;

{ главная процедура }
var
    grDriver: Integer;
    grMode: Integer;
    ErrCode: Integer;
    res: integer;

    p : integer; { Порядок кривой Гильберта }
    st: string;
begin
    grDriver := detect;
    InitGraph(grDriver, grMode, 'e:\tp\bgi');
    ErrCode := GraphResult;
    if ErrCode = grOk then
    begin
        p:=5;
        Str(p,2,st);
        OuttextXY(0,0,'Кривая Гильберта'+st+'-го порядка. ');
        MoveTo(450,50);
        u:=10;
        a(p);
        OuttextXY(0,16,'Для завершения работы программы ',
                  'нажмите <Enter>.' );
        readln;
    end;
    CloseGraph;
end.
```

Задача 241.

{ Программа строит кривую Серпинского.
см. Вирт Н. Алгоритмы и структуры данных:
Пер. с англ. – М.: Мир, 1989. – 360 с.: ил. }

```
uses Graph, Crt;

var
  u: integer; { Длина штриха }

{ LineTo - вычерчивает по точкам линию из текущей
  точки в заданную.
  Заменяет стандартную процедуру
  LineTo для того, чтобы можно было видеть
  процесс вычерчивания. Возможно надо увеличить
  величину задержки между выводом точек. }
procedure LineTo(x2,y2: integer);
    { x2,y2 - координаты конца линии}
const
  DT = 3; { задержка между выводом точек линии }
var
  x1,y1: integer; { координаты начала прямой }
  x,y : integer; { координаты текущей точки }
  dx: integer; { приращение аргумента }
  dy: integer; { приращение y при рисовании
                вертикальной линии }
  color: integer; { цвет линии}
  a,b: real; { коэф-ты уравнения прямой }
  n: integer; { кол-во точек }
  i: integer;
begin
  x1:=GetX;
  y1:=GetY;
  if x1 <> x2
  then begin
    { не вертикальная линия }
    a:=(y2-y1)/(x2-x1);
    b:=y1-a*x1;
    n:=abs(x2-x1)+1;
    if x2 > x1
    then dx:=1
    else dx:=-1;
    x:=x1;
    color:=GetColor;
    for i:=1 to n do
```

```

begin
    y:=Round(a*x+b);
    PutPixel(x,y,color);
    delay(DT);
    x:=x+dx;
end;
end
else begin { вертикальная линия }
    n:=abs(y2-y1);
    if y2 > y1
        then dy:=1
        else dy:=-1;
    x:=x1;
    y:=y1;
    color:=GetColor;
    for i:=1 to n do
        begin
            PutPixel(x,y,color);
            delay(DT);
            y:=y+dy;
        end;
    end;
    PutPixel(x2,y2,color);
    MoveTo(x2,y2);
end;

```

```

procedure Vector(a: integer;      { а - угол между вектором
                                   и осью OX }
                 l: integer); { длина вектора }
{ Угол задается целым числом от 0 до 7.
  0 соответствует нулю градусов, 1 - 45, 2 - 90 и т. д. }
var
    x0,y0: integer; { координаты начала вектора }
    x1,y1: integer; { координаты конца вектора }
begin
    x0:=GetX;
    y0:=GetY;
    x1:=Round(x0+l*cos(a*Pi/4));
    y1:=Round(y0-l*sin(a*Pi/4));
    LineTo(x1,y1);
end;

```

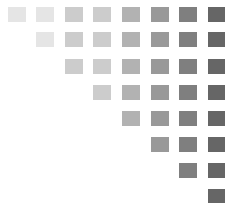
```
{ Кривая состоит из четырех элементов: a,b,c и d.  
  Каждый элемент строит соответствующая процедура. }  
procedure a(i:integer); external;  
procedure b(i:integer); external;  
procedure c(i:integer); external;  
procedure d(i:integer); external;  
  
{ Элементы кривой. }  
procedure a(i: integer);  
begin  
  if i > 0 then  
    begin  
      a(i-1);Vector(7,u);  
      b(i-1);Vector(0,2*u);  
      d(i-1);Vector(1,u);  
      a(i-1);  
    end;  
end;  
  
procedure b(i: integer);  
begin  
  if i > 0 then  
    begin  
      b(i-1);Vector(5,u);  
      c(i-1);Vector(6,2*u);  
      a(i-1);Vector(7,u);  
      b(i-1);  
    end;  
end;  
  
procedure c(i: integer);  
begin  
  if i > 0 then  
    begin  
      c(i-1);Vector(3,u);  
      d(i-1);Vector(4,2*u);  
      b(i-1);Vector(5,u);  
      c(i-1);  
    end;  
end;
```

```
procedure d(i: integer);
begin
  if i > 0 then
  begin
    d(i-1); Vector(1,u);
    a(i-1); Vector(2,2*u);
    c(i-1); Vector(3,u);
    d(i-1);
  end;
end;

{ главная процедура }
var
  grDriver: Integer;
  grMode: Integer;
  ErrCode: Integer;
  res: integer;

  p : integer; { Порядок кривой Гильберта }
  st: string;
  i: integer;
begin
  writeln('Демонстрация понятия "рекурсия".');
  writeln('Программа строит кривую Серпинского. ');
  writeln('Введите порядок кривой (1-4) ',
    'и нажмите <Enter>');
  write('->');
  readln(p);
  grDriver := detect;
  InitGraph(grDriver, grMode, 'e:\tp\bgi');
  ErrCode := GraphResult;
  if ErrCode = grOk then
  begin
    Str(p:2,st);
    OuttextXY(0,0,'Кривая Серпинского'+st+'-го порядка. ');
    MoveTo(10,30);
    u:=5;
    a(p); Vector(7,u);
    b(p); Vector(5,u);
```

```
c(p) ; Vector(3,u) ;  
d(p) ; Vector(1,u) ;  
OuttextXY(0,16,  
    'Для завершения работы программы нажмите <Enter>.' ) ;  
readln ;  
end ;  
CloseGraph ;  
end .
```

Часть III. Справочник

Структура программы

Программа на языке Pascal состоит из разделов:

- раздел объявления меток;
- раздел объявления констант;
- раздел объявления типов;
- раздел объявления переменных;
- раздел объявления процедур и функций;
- раздел инструкций программы.

Структура программы в общем виде выглядит следующим образом:

```
label
    { объявления меток }
const
    { объявления констант }
type
    { объявления типов }
var
    { объявления переменных }

{ объявления процедур и функций программиста }

begin
    { инструкции основной программы }
end.
```

Основные типы данных

К основным типам данных языка Pascal относятся:

- целые числа (INTEGER и др.);
- действительные числа (REAL и др.);

- символы (CHAR);
- строки (STRING);
- логический (BOOLEAN).

Целые числа и числа с плавающей точкой могут быть представлены в различных форматах.

Целые числа

Формат	Диапазон значений
SHORTINT	−128—127
INTEGER	−32 768—32 767
LONGINT	−2 147 483 648—2 147 483 647
BYTE	0—255
WORD	0—65 535

Действительные числа

Формат	Диапазон значений	Количество значащих цифр
REAL	2.9E-39—1.7e38	11—12
SINGLE	1.5E-45—3.4e38	7—8
DOUBLE	5.0E-324—1.7e308	15—16
EXTENDED	3.4E-4932—1.1e4932	19—20

Строки

Строки могут быть объявлены двумя способами.

Вариант 1:

Имя: **string**;

Объявлена переменная-строка длиной 255 символов.

Вариант 2:

Имя: **string**[ДлинаСтроки];

Объявлена переменная-строка указанной длины.

Массивы

Объявление одномерного массива:

ИмяМассива: **array**[НижнийИндекс..ВерхнийИндекс]
of ТипЭлементов;

Объявление двумерного массива:

ИмяМассива: **array**[НижнийИндекс1..ВерхнийИндекс1,
НижнийИндекс2..ВерхнийИндекс2]
of ТипЭлементов;

Записи

Записи можно объявить двумя способами.

Вариант 1:

ПеременнаяЗапись: **record**
 Поле1: Тип1;
 Поле2: Тип2;
 ...
 ПолеJ: ТипJ;
end;

Вариант 2 (сначала объявляется тип записи, затем — переменная-запись):

type
 ИмяТипаЗаписи: **record**
 Поле1: Тип1;
 Поле2: Тип2;
 ...
 ПолеJ: ТипJ;
 end;
var
 ПеременнаяЗапись: ИмяТипаЗаписи;

Выбор

Инструкция *IF*

Вариант 1:

```
if УСЛОВИЕ
then
    begin
        { Эти инструкции выполняются, }
        { если УСЛОВИЕ истинно }
    end
else
    begin
        { Эти инструкции выполняются, }
        { если УСЛОВИЕ ложно }
    end;

```

Вариант 2:

```
if УСЛОВИЕ
then
    begin
        { Эти инструкции выполняются, }
        { если УСЛОВИЕ истинно }
    end;

```

Примечание

Если между **begin** и **end** находится только одна инструкция, то слова **begin** и **end** можно не писать.

Инструкция *CASE*

Вариант 1:

```
case Выражение of
    Список_констант1: begin
        { Инструкции 1 }
    end;
    Список_констант2: begin

```

```
        { Инструкции 2 }  
    end;  
    ...  
    Список_константJ: begin  
        { Инструкции 3 }  
    end;  
end;
```

Вариант 2:

```
case Выражение of  
    Список_констант1: begin  
        { Инструкции 1 }  
    end;  
    Список_констант2: begin  
        { Инструкции 2 }  
    end;  
    ...  
    Список_константJ: begin  
        { Инструкции J }  
    end  
    else begin  
        { Инструкции }  
    end;  
end;
```

Инструкции между `begin` и `end` выполняются в том случае, если значение выражения после слова `case` совпадает с константой из соответствующего списка. Если это не так, то выполняются инструкции, следующие после `else`, расположенные между `begin` и `end..`

Циклы

Инструкция **FOR**

Вариант 1 (с увеличением счетчика):

```
for Счетчик := НачальноеЗначение to КонечноеЗначение do  
    begin
```

```

    { Инструкции }
end;

```

Инструкции между `begin` и `end` выполняются $[(\text{КонечноеЗначение} - \text{НачальноеЗначение}) + 1]$ раз.

Если *НачальноеЗначение* больше, чем *КонечноеЗначение*, то инструкции между `begin` и `end` не выполняются.

Примечание

Если между `begin` и `end` находится только одна инструкция, то слова `begin` и `end` можно не писать.

Вариант 2 (с уменьшением счетчика):

```

for Счетчик := НачальноеЗначение downto КонечноеЗначение do
begin
    { Инструкции }
end;

```

Инструкции между `begin` и `end` выполняются $[(\text{НачальноеЗначение} - \text{КонечноеЗначение}) + 1]$ раз.

Если *НачальноеЗначение* меньше, чем *КонечноеЗначение*, то инструкции между `begin` и `end` не выполняются.

Инструкция **REPEAT**

```

repeat
begin
    { Инструкции }
end
until Условие;

```

Выполняются инструкции, находящиеся между `begin` и `end` (инструкции цикла), после этого проверяется значение выражения *Условие*. Если оно равно `False` (т. е. условие не выполняется), то инструкции цикла выполняются еще раз. И так продолжается до тех пор, пока *Условие* не станет истинным. Таким образом, после слова `until` записывается условие завершения цикла.

Примечание

Если между `begin` и `end` находится только одна инструкция, то слова `begin` и `end` можно не писать.

Инструкция *WHILE*

```
while Условие do  
  begin  
    { Инструкции }  
  end;
```

Проверяется значение выражения *Условие*, если оно равно `True` (условие выполняется), то выполняются инструкции, находящиеся между `begin` и `end` (инструкции цикла). Затем снова проверяется значение выражения *Условие*, и так продолжается до тех пор, пока значение выражения *Условие* не станет равным `False`. Таким образом, после слова `while` записывается условие выполнения инструкций цикла.

Примечание

Если между `begin` и `end` находится только одна инструкция, то слова `begin` и `end` можно не писать.

Безусловный переход

Инструкция *GoTo*

```
GoTo Метка;
```

Осуществляется переход к инструкции, перед которой указана *Метка*, объявленная в разделе `label`.

Объявление функции

```
Function ИмяФункции(var Параметр1:Тип1;  
  var Параметр2:Тип2;  
  . . .  
  var ПараметрJ:ТипJ) :Тип;
```

```
const
    { Описание констант }
var
    { Описание переменных }
begin
    { Инструкции функции }

    ИмяФункции:=Значение;
end;
```

Объявление процедуры

```
Procedure ИмяПроцедуры(var Параметр1:Тип1;
    var Параметр2:Тип2;
    . . .
    var ПараметрJ:ТипJ) ;
const
    { Описание констант }
var
    { Описание переменных }
begin
    { Инструкции процедуры }
end;
```

Встроенные функции и процедуры

При описании функций и процедур приняты следующие обозначения:

- имена функций и процедур выделены полужирным шрифтом;
- параметры выделены курсивом (В качестве параметра могут использоваться константы, переменные или выражения соответствующих типов. Если параметром обязательно должна быть переменная основной программы, то перед ним поставлено слово *var*. После параметра указывается его тип.);
- после списка параметров функций через двоеточие указан тип результата, возвращаемого функцией.

Математические	Комментарий
Abs (<i>Выражение</i>)	Абсолютное значение аргумента (целый или вещественный тип)
Sqr (<i>Выражение</i>)	Квадрат аргумента (целый или вещественный тип)
Sqrt (<i>Выражение</i> :real):real	Квадратный корень из аргумента
Sin (<i>Выражение</i> :real):real	Синус аргумента, выраженного в радианах
Cos (<i>Выражение</i> :real):real	Косинус аргумента, выраженного в радианах
Arctan (<i>Выражение</i> :real):real	Арктангенс аргумента, выраженного в радианах
Exp (<i>Выражение</i> :real):real	Экспонента аргумента
Ln (<i>Выражение</i> :real):real	Натуральный логарифм аргумента
Int (<i>Выражение</i> :real):real	Целая часть аргумента
Trunc (<i>Выражение</i> :real):longint	Целая часть аргумента
Round (<i>Выражение</i> :real):longint	Округление значения аргумента до ближайшего целого
Str (<i>Выражение</i> ; var <i>Строка</i> :string)	Преобразование числового выражения в строку
Val (<i>Строка</i> :string; var <i>Переменная</i> , var <i>КодОшибки</i> :integer)	Преобразование строки, изображающей целое или вещественное число, в число

Для работы со строками и символами	Комментарий
Concat (<i>Строка1</i> :string; ...; <i>СтрокаN</i> :string):string	Объединение нескольких строк в одну

(окончание)

Для работы со строками и символами	Комментарий
Copy (<i>Строка</i> :string; <i>НомерСимвола</i> :integer; <i>Длина</i> :integer):string	Выделение подстроки
Delete (var <i>Строка</i> :string; <i>НомерСимвола</i> :integer; <i>Сколько</i> :integer)	Удаление части строки
Length (<i>Строка</i> :string):integer	Длина строки
Pos (<i>Строка</i> :string; <i>Подстрока</i> :string):byte	Позиция подстроки в строке
Chr (<i>КодСимвола</i> :byte):Char	Символ с указанным кодом

Зарезервированные слова и директивы

Зарезервированные слова языка Turbo Pascal:

and	file	not	then
array	for	object	to
asm	function	of	type
begin	goto	or	unit
case	if	packed	until
const	implementation	procedure	uses
constructor	in	program	var
destructor	inherited	record	while
div	inline	repeat	with
do	interface	set	xor
downto	label	shl	
else	mod	shr	
end	nil	string	

Директивы языка Turbo Pascal:

absolute	external	forward	near	public
assembler	far	interrupt	private	virtual

Процедуры и функции

В описании функций и процедур приняты следующие обозначения:

- имена функций и процедур выделены полужирным шрифтом;
- параметры выделены курсивом (В качестве параметра можно использовать константы, переменные или выражения соответствующих типов. Если параметром обязательно должна быть переменная основной программы, то перед ним поставлено слово **var**, а после параметра — его тип);
- необязательные параметры заключены в квадратные скобки;
- после списка параметров функций (через двоеточие) указан тип результата, возвращаемого функцией.

Математические функции

Abs

Синтаксис:

```
function Abs (X) ;
```

Действие:

Возвращает абсолютное значение аргумента, в качестве которого можно использовать выражение целого или вещественного типа.

Arctan

Синтаксис:

```
function Arctan (X: real): real;
```

Действие:

Возвращает арктангенс аргумента — угла, величина которого выражена в радианах.

Cos

Синтаксис:

```
function Cos (X: real): real;
```

Действие:

Возвращает косинус аргумента — угла, величина которого выражена в радианах.

Exp

Синтаксис:

```
function Exp(X: real): real;
```

Действие:

Возвращает значение, равное экспоненте аргумента.

Ln

Синтаксис:

```
function Ln(X: real): real;
```

Действие:

Возвращает значение, равное натуральному логарифму аргумента.

Sin

Синтаксис:

```
function Sin(X: real): real;
```

Действие:

Возвращает синус аргумента — угла, величина которого выражена в радианах.

Sqr

Синтаксис:

```
function Sqr(X);
```

Действие:

Возвращает квадрат аргумента, в качестве которого можно использовать выражение целого или вещественного типа.

Sqrt

Синтаксис:

```
function Sqrt(X: real): real;
```

Действие:

Возвращает значение, равное квадратному корню из аргумента.

Random

Синтаксис:

```
function Random [ (Диапазон: word) ] ;
```

Действие:

Если параметр *Диапазон* не указан, то возвращает случайное число X , которое удовлетворяет условию $0 \leq X < 1$. Если параметр *Диапазон* указан, то функция возвращает случайное число типа *word*, удовлетворяющее условию $0 \leq X < \text{Диапазон}$.

Примечание

Перед первым обращением к функции `Random` необходимо с помощью вызова процедуры `Randomize` инициализировать программный генератор случайных чисел.

Randomize

Синтаксис:

```
procedure Randomize;
```

Действие:

Инициализирует программный генератор случайных чисел.

Функции и процедуры преобразования

Int

Синтаксис:

```
function Int(X: real): real;
```

Действие:

Возвращает целую часть аргумента как значение вещественного типа. Дробная часть аргумента при преобразовании не учитывается, т. е. функция не производит округление.

Round

Синтаксис:

```
function Round(X: real): longint;
```

Действие:

Возвращает округленное к ближайшему целому значение аргумента.

Str

Синтаксис:

```
procedure Str(X [:ВсегоСимволов [:ДробнаяЧасть ]]);  
var Строка: string;
```

Действие:

Выполняет преобразование числового выражения в его строковое представление. *ВсегоСимволов* и *ДробнаяЧасть* — необязательные выражения целого типа, которые задают общее количество символов и количество символов дробной части в изображении числа.

Trunc

Синтаксис:

```
function Trunc(X: real): longint;
```

Действие:

Возвращает целую часть аргумента как значение целого типа. Дробная часть аргумента при преобразовании не учитывается, т. е. функция не производит округление.

Val

Синтаксис:

```
procedure Val(Строка: string; var Переменная;  
          var Ошибка : integer);
```

Действие:

Выполняет преобразование строки, изображающей целое или вещественное число, в число. Полученное значение присваива-

ется переменной, указанной при вызове процедуры. Если преобразование не может быть выполнено, то в переменную *Ошибка* записывается номер символа строки, который явился причиной неудачи преобразования. Если преобразование выполнено успешно, то значение *Ошибка* равно нулю.

Функции и процедуры для работы со строками и символами

Chr

Синтаксис:

```
function Chr(КодСимвола: byte): char;
```

Действие:

Возвращает символ с указанным кодом.

Concat

Синтаксис:

```
function Concat(s1 [, s2, ..., sN]: string): string;
```

Действие:

Возвращает строку, являющуюся объединением строк, указанных при вызове функции.

Copy

Синтаксис:

```
function Copy(s: string; n:integer; l: integer): string;
```

Действие:

Возвращает подстроку — часть строки *s*. Подстрока начинается с символа с номером *n* и состоит из *l* символов.

Delete

Синтаксис:

```
procedure Delete(var s: string; НомерСимвола:  
integer; Сколько: integer);
```

Действие:

Удаляет из строки *s* ее часть, которая начинается с символа с номером *n* и состоит из *l* символов.

Length

Синтаксис:

```
function Length(Строка: string): integer;
```

Действие:

Возвращает значение, равное количеству символов строки-аргумента.

Pos

Синтаксис:

```
function Pos(Строка: string; Подстрока: string): byte;
```

Действие:

Возвращает позицию (номер символа) подстроки в строке.

Процедуры и функции графического режима

Arc

Синтаксис:

```
procedure Arc(x,y: integer; УголНачала, УголКонца:  
integer; Радиус: integer);
```

Действие:

Вычерчивает дугу с центром в точке с координатами (x,y) . Параметры *УголНачала* и *УголКонца* задают угловые координаты начальной и конечной точек линии дуги, которая вычерчивается против часовой стрелки от начальной точки к конечной. Угловые координаты задаются в градусах. Значение угловой координаты возрастает против часовой стрелки. Параметр *Радиус* задает радиус дуги.

Дуга вычерчивается цветом, заданным процедурой *SetColor*.

Bar

Синтаксис:

```
procedure Bar(x1, y1, x2, y2: integer);
```

Действие:

Вычерчивает закрашенный прямоугольник. Параметры *x1* и *y1* задают положение левого верхнего угла прямоугольника, *x2* и *y2* — правого нижнего. Используемый стиль и цвет заливки задается процедурой `SetFillStyle`.

Bar3D

Синтаксис:

```
procedure Bar3D(x1, y1, x2, y2: integer; Глубина: word;  
  Граница: boolean);
```

Действие:

Вычерчивает параллелепипед. Параметры *x1* и *y1* задают положение левого верхнего, а *x2* и *y2* — правого нижнего угла ближней грани параллелепипеда. Параметр *Глубина* задает расстояние между передней и задней гранями. Параметр *Граница* определяет, нужно ли вычерчивать верхнюю границу задней грани параллелепипеда.

Цвет и стиль закраски ближней грани параллелепипеда можно задать при помощи процедуры `SetFillStyle`, цвет линий границы — процедурой `SetColor`.

Circle

Синтаксис:

```
procedure Circle(x, y: integer; r: word);
```

Действие:

Вычерчивает окружность радиуса *r* с центром в точке с координатами (*x*, *y*).

Цвет окружности можно задать при помощи процедуры `SetColor`.

DetectGraph

Синтаксис:

```
DetectGraph(var Драйвер, Режим: integer);
```

Действие:

Проверяет графический адаптер и определяет, какой графический драйвер и режим используются.

Ellipse

Синтаксис:

```
procedure Ellipse(x, y: integer; УголНачала, УголКонца: word;  
                  РадиусX, РадиусY: word);
```

Действие:

Вычерчивает эллипс или дугу эллипса с центром в точке с координатами (x, y) . Параметры *УголНачала* и *УголКонца* задают угловые координаты начальной и конечной точек линии эллипса, которая вычерчивается против часовой стрелки от начальной точки к конечной. Угловые координаты задаются в градусах, их значения возрастают против часовой стрелки. Параметры *РадиусX* и *РадиусY* определяют горизонтальный и вертикальный радиусы эллипса. Линия эллипса или дуги вычерчивается в соответствии с установками процедуры *SetColor*.

GetX, GetY

Синтаксис:

```
function GetX: integer;  
function GetY: integer;
```

Действие:

Возвращают координаты x и y указателя вывода.

GraphResult

Синтаксис:

```
function GraphResult: integer;
```

Действие:

Возвращает результат (код ошибки) последней выполненной графической операции. Если операция выполнена успешно, функция возвращает ноль. Код ошибки выполнения графической операции устанавливают процедуры `Bar`, `Bar3D`, `InitGraph`, `PieSlice`, `SetFillPattern`, `SetFillStyle`, `SetLineStyle`, `SetTextStyle` и др.

InitGraph

Синтаксис:

```
InitGraph(var Driver: integer; var Mode: integer; Path: string)
```

Действие:

Инициализирует графический режим. Параметр *Driver* определяет драйвер видеосистемы, параметр *Mode* — режим работы видеосистемы, параметр *Path* — каталог, где находится драйвер.

Line

Синтаксис:

```
procedure Line(x1, y1, x2, y2: integer);
```

Действие:

Вычерчивает линию между двумя точками экрана, координаты которых указаны при вызове процедуры. Линия вычерчивается стилем, установленным процедурой `SetLineStyle`. Цвет линии можно задать при помощи процедуры `SetColor`.

LineTo

Синтаксис:

```
procedure Line(x1, y1: integer);
```

Действие:

Вычерчивает линию от текущего положения указателя вывода до точки, координаты которой указаны при вызове процедуры. Линия вычерчивается стилем, установленным процедурой `SetLineStyle`. Цвет линии можно задать при помощи процедуры `SetColor`.

MoveTo

Синтаксис:

```
procedure MoveTo(x, y: integer);
```

Действие:

Перемещает указатель вывода в точку с координатами *x* и *y*.

MoveRel

Синтаксис:

```
procedure MoveRel(dx, dy: integer);
```

Действие:

Перемещает указатель вывода на *dx* и *dy* пикселей. Если значение параметра *dx* (*dy*) положительное, то указатель перемещается вниз (влево), если отрицательное, то — вверх (вправо).

OutText

Синтаксис:

```
procedure OutText(Текст: string);
```

Действие:

Выводит строку символов *Текст* от текущего положения указателя вывода и перемещает указатель вывода в точку, расположенную за последним выведенным символом.

Цвет выводимых символов можно задать при помощи процедуры *SetColor*, а шрифт — *SetTextStyle*.

OutTextXY

Синтаксис:

```
procedure OutTextXY(x, y: integer; Текст: string);
```

Действие:

Устанавливает указатель вывода в точку с координатами (*x, y*) и выводит строку символов *Текст*, при этом указатель вывода своего положения не меняет, остается в точке с координатами (*x, y*).

Цвет выводимых символов можно задать при помощи процедуры `SetColor`, а шрифт — процедурой `SetTextStyle`.

PieSlice

Синтаксис:

```
procedure PieSlice(x, y: integer; УголНачала, УголКонца,  
                  Радиус: word);
```

Действие:

Вычерчивает сектор радиуса *Радиус* с центром в точке с координатами (*x, y*). Параметры *УголНачала* и *УголКонца* задают угловые координаты начальной и конечной точек линии окружности (в градусах), которая вычерчивается против часовой стрелки от начальной к конечной точке. Значение угловой координаты возрастает против часовой стрелки. Нулевому углу соответствует горизонтальный отрезок, проведенный из точки (*x, y*) в сторону возрастания координаты *x*. Если *УголНачала* равен 0°, а *УголКонца* равен 360°, то процедура `PieSlice` вычерчивает круг.

Сектор закрашивается в соответствии со стилем, заданным процедурой `SetFillStyle`, линия и границы вычерчиваются цветом, установленным процедурой `SetColor`.

PutPixel

Синтаксис:

```
procedure PutPixel(x, y: integer; Цвет: word);
```

Действие:

Окрашивает пиксел, точку с координатами (*x, y*), цветом *Цвет*. В качестве параметра *Цвет* обычно используют именованную константу (см. `SetColor`).

Rectangle

Синтаксис:

```
procedure Rectangle(x1, y1, x2, y2: integer);
```

Действие:

Вычерчивает прямоугольник. Параметры *x1* и *y1* задают положение левого верхнего угла прямоугольника, *x2* и *y2* — правого нижнего угла прямоугольника.

Вид контура прямоугольника можно задать при помощи процедуры `SetLineStyle`, а цвет — при помощи процедуры `SetColor`.

Sector

Синтаксис:

```
procedure Sector(x, y: integer; Угол1, Угол2, РадиусX,  
                 РадиусY: word);
```

Действие:

Вычерчивает эллиптический ($\text{Радиус}X \neq \text{Радиус}Y$) или круговой ($\text{Радиус}X = \text{Радиус}Y$) сектор. Параметры x и y задают положение центра сектора, Угол1 и Угол2 — углы прямых, ограничивающих сектор, а $\text{Радиус}X$ и $\text{Радиус}Y$ — радиусы эллипса по осям X и Y , из которого "вырезается" отображаемый сектор. Нулевому углу соответствует горизонтальный отрезок, проведенный из точки (x, y) в сторону возрастания координаты x . Если Угол1 равен 0° , а Угол2 равен 360° , то процедура `Sector` вычерчивает полный круг (эллипс).

Цвет и стиль заливки можно задать при помощи процедуры `SetFillStyle`, цвет границы сектора — процедурой `SetColor`.

SetColor

Синтаксис:

```
procedure SetColor(Цвет: word);
```

Действие:

Задаёт цвет для вывода текста (процедуры `OutTextXY` и `OutText`), вычерчивания линий и фигур (процедуры `Line`, `Circle`, `Rectangle` и др.). В качестве параметра *Цвет* обычно используют именованные константы.

Цвет	Константа	Значение константы
Черный	Black	0
Синий	Blue	1
Зеленый	Green	2
Бирюзовый	Cyan	3

(окончание)

Цвет	Константа	Значение константы
Красный	Red	4
Сиреневый	Magenta	5
Коричневый	Brown	6
Белый	LightGray	7
Серый	DarkGray	8
Голубой	LightBlue	9
Светло-зеленый	LightGreen	10
Светло-бирюзовый	LightCyan	11
Светло-красный (алый)	LightRed	12
Светло-сиреневый	LightMagenta	13
Желтый	Yellow	14
Белый, яркий	White	15

SetFillStyle

Синтаксис:

```
procedure SetFillStyle(Стиль, Цвет: word);
```

Действие:

Устанавливает стиль и цвет заливки (закрашивания), используемый процедурами вывода областей (Bar, Bar3D, Sector и др.). В качестве параметра *Стиль* обычно используют одну из именованных констант, список которых приведен ниже. Параметр *Цвет* также задается именованной константой (см. SetColor).

Константа	Стиль заполнения области
EmptyFill	Без заливки (сплошная заливка цветом фона)
SolidFill	Сплошная заливка текущим цветом
LineFill	Горизонтальная штриховка

(окончание)

Константа	Стиль заполнения области
LtSlashFill	Штриховка под углом 45° влево тонкими линиями
SlashFill	Штриховка под углом 45° влево
BkSlashFill	Штриховка под углом 45° вправо тонкими линиями
LtBkSlashFill	Штриховка под углом 45° вправо
HatchFill	Штриховка клеткой
XhatchFill	Штриховка под углом 45° редкой косой клеткой
InterleaveFill	Штриховка под углом 45° частой косой клеткой
WideDotFill	Заполнение редкими точками
CloseDotFill	Заполнение частыми точками
UserFill	Тип заполнения определяется программистом

SetLineStyle

Синтаксис:

```
procedure SetLineStyle(ТипЛинии: word; Образец: word;  
    Толщина: word);
```

Действие:

Устанавливает стиль вычерчиваемых контуров и линий (см. процедуры Line, Circle и др.).

Параметр *ТипЛинии*, в качестве которого обычно используется одна из именованных констант, определяет вид линии.

Константа	Тип линии
SolidLn	Сплошная, непрерывная
DottedLn	Пунктирная, с постоянной длиной штрихов
CenterLn	Штрих-пунктирная линия
DashedLn	Пунктирная, длина штрихов чуть больше, чем у линии типа DottedLn
UserBitLn	Определенный программистом тип линии

Параметр *Толщина* определяет толщину линии. Линия может быть обычной толщины (константа `NormWidth`) или утолщенная (константа `ThickWidth`).

Параметр *Образец* используется в том случае, если процедура `SetLineStyle` устанавливает тип линии, определяемый программистом. Значением параметра *Образец* должна быть четырехрядная шестнадцатеричная константа, кодирующая отрезок линии длиной в 16 пикселей.

SetTextStyle

Синтаксис:

```
procedure SetTextStyle(Шрифт, Ориентация, Размер: word);
```

Действие:

Устанавливает шрифт, размер и ориентацию текста, выводимого процедурами `OutTextXY` и `OutText`. В качестве параметра *Шрифт* можно использовать одну из перечисленных ниже констант.

Константа	Значение	Шрифт
<code>DefaultFont</code>	0	Стандартный. Каждый выводимый символ формируется в квадрате размером 8 на 8 пикселей
<code>TriplexFont</code>	1	Triplex шрифт
<code>SmallFont</code>	2	Мелкий
<code>SansSerifFont</code>	3	SansSerif шрифт
<code>GothicFont</code>	4	Готический

Примечание

В шрифтах, отличных от стандартного (`DefaultFont`), букв русского алфавита нет.

Параметр *Ориентация* задает ориентацию выводимого процедурами `OutText` и `OutTextXY` текста. Текст может быть ориентирован обычным образом (значение параметра *Ориентация* в этом случае равно именованной константе `Normal`) или вертикально, т. е.

снизу вверх (в этом случае значение параметра *Ориентация* равно Vertical).

Процедуры и функции для работы с файлами

Append

Синтаксис:

```
procedure Append(var F: text);
```

Действие:

Открывает существующий файл, связанный с файловой переменной *F* в режиме добавления в конец файла. При попытке открыть несуществующий файл появляется ошибка времени выполнения программы.

Assign

Синтаксис:

```
procedure Assign(var F; ИмяФайла: string);
```

Действие:

Связывает файловую переменную *F* с конкретным файлом.

Close

Синтаксис:

```
procedure Close(var F);
```

Действие:

Закрывает файл, связанный с файловой переменной *F*.

EOF

Синтаксис:

```
function EOF(var F): boolean;
```

Действие:

Проверяет, не достигнут ли конец файла (end of file) при чтении из файла, связанного с файловой переменной *F*. Если указатель

чтения достиг конца файла, то функция `EOF` возвращает значение `TRUE`, в противном случае — `FALSE`.

Erase

Синтаксис:

```
procedure Erase (var F);
```

Действие:

Уничтожает файл, имя которого связано с файловой переменной *F*.

IOResult

Синтаксис:

```
function IOResult: integer;
```

Действие:

Возвращает код результата последней выполненной операции файлового ввода-вывода (в том числе открытия и закрытия файла). Если операция ввода-вывода выполнена успешно, функция возвращает ноль.

Примечание

Чтобы программа могла использовать функцию `IOResult`, надо перед инструкциями, в результате которых может возникнуть ошибка ввода-вывода, поместить директиву `{SI-}`, а после этих инструкций — директиву `{SI+}`.

Reset

Синтаксис:

```
procedure Reset (var F [:file; RecSize: word]);
```

Действие:

Открывает существующий файл. Тип файла может быть любым. Если элементы файла не относятся к одному из стандартных типов, то параметр *RecSize* задает длину записи (в байтах). При попытке открыть несуществующий файл возникает ошибка времени выполнения.

Rewrite

Синтаксис:

```
procedure Rewrite(var F [:file; RecSize: word]);
```

Действие:

Создает и открывает файл, имя которого связано с файловой переменной *F*. Если файл с таким именем уже существует, то процедура Rewrite его уничтожает и создает новый.

Прочие функции

ClrEol

Синтаксис:

```
procedure ClrEol;
```

Действие:

Очищает текущую строку (строка, в которой находится курсор) от позиции, в которой находится курсор, до конца строки, закрашивая ее текущим цветом, заданным процедурой TextBackGround.

ClrScr

Синтаксис:

```
procedure ClrScr;
```

Действие:

Очищает текущее (заданное процедурой Window) окно экрана, закрашивая его цветом, заданным процедурой TextBackGround.

Delay

Синтаксис:

```
procedure Delay(Задержка: word);
```

Действие:

Обеспечивает задержку выполнения программы в миллисекундах.

Dispose

Синтаксис:

```
procedure Dispose (var p);
```

Действие:

Освобождает память, занимаемую динамической переменной, на которую указывает *p*.

Eoln

Синтаксис:

```
function Eoln [ (var F: text)]: boolean;
```

Действие:

Если при обращении к функции *Eoln* параметр не указан, то функция проверяет, не является ли очередной символ, читаемый из буфера клавиатуры, символом "новая строка". Если является, то функция возвращает значение *TRUE*, в противном случае — *FALSE*. Если параметр указан, то функция аналогичным образом проверяет очередной символ текстового файла, связанного с файловой переменной, указанной при вызове функции.

GotoXY

Синтаксис:

```
procedure GoToXY (x, y: byte);
```

Действие:

Перемещает курсор в точку экрана, имеющую координаты (*x*, *y*).

Halt

Синтаксис:

```
procedure Halt [ (КодЗавершения: word) ];
```

Действие:

Завершает выполнение программы и передает управление операционной системе.

New

Синтаксис:

```
procedure New(var p) ;
```

Действие:

Выделяет память для динамической переменной и присваивает указателю *p* адрес выделенной области.

ParamCount

Синтаксис:

```
function ParamCount: word;
```

Действие:

Возвращает количество параметров командной строки.

ParamStr

Синтаксис:

```
function ParamStr(N: word): string;
```

Действие:

Возвращает параметр командной строки, номер которого указан при обращении к функции. Значением `ParamStr(0)` является путь к файлу выполняемой программы и его имя (например, `c:\tp\exe&tpu\myprog.exe`).

ReadKey

Синтаксис:

```
function ReadKey : char;
```

Действие:

Возвращает символ, соответствующий нажатой клавише. Используя функцию `Ord`, можно получить код нажатой клавиши.

Если нажата служебная клавиша (например, клавиша перемещения курсора), то функция `ReadKey` возвращает символ с кодом 0. В этом случае для идентификации служебной клавиши следует еще раз вызвать функцию `ReadKey`, которая теперь вернет код служебной клавиши.

Ниже приведены коды некоторых часто используемых клавиш.

Клавиша	Код
<Esc>	27
<Backspace>	8
<Enter>	13
<Space>	32
<Курсор вверх>	0;72
<Курсор вниз>	0;80
<Курсор влево>	0;75
<Курсор вправо>	0;77

TextBackGround

Синтаксис:

```
procedure TextBackGround (Цвет: byte);
```

Действие:

Задаёт цвет фона сообщений, выводимых инструкциями `write` и `writeln`. В качестве параметра *Цвет* может использоваться одна из перечисленных ниже именованных констант.

Константа	Цвет	Номер цвета
Black	Черный	0
Blue	Синий	1
Green	Зеленый	2
Cyan	Бирюзовый	3
Red	Красный	4
Magenta	Сиреневый	5
Brown	Коричневый	6
LightGray	Белый	7

TextColor

Синтаксис:

```
procedure TextColor (Цвет: byte) ;
```

Действие:

Устанавливает цвет символов сообщений, выводимых инструкциями `write` и `writeln`. В качестве параметра *Цвет* может использоваться одна из перечисленных ниже именованных констант.

Константа	Цвет	Номер цвета
Black	Черный	0
Blue	Синий	1
Green	Зеленый	2
Cyan	Бирюзовый	3
Red	Красный	4
Magenta	Сиреневый	5
Brown	Коричневый	6
LightGray	Белый	7
DarkGray	Серый	8
LightBlue	Голубой	9
LightGreen	Светло-зеленый	10
LightCyan	Светло-бирюзовый	11
LightRed	Светло-красный (алый)	12
LightMagenta	Светло-сиреневый	13
Yellow	Желтый	14
White	Белый, яркий	15

WhereX

Синтаксис:

```
function WhereX: byte;
```

Действие:

Возвращает координату X курсора в текущем окне, заданном процедурой `Window`.

WhereY

Синтаксис:

```
function WhereY: byte;
```

Действие:

Возвращает координату Y курсора в текущем окне, заданном процедурой `Window`.

Window

Синтаксис:

```
procedure Window(x1, y1, x2, y2: byte);
```

Действие:

Определяет окно — область экрана. Параметры *x1*, *y1* задают координаты левого верхнего угла окна, а параметры *x2*, *y2* — правого нижнего угла.

ПРИЛОЖЕНИЕ

Вывод иллюстраций

В языке Turbo Pascal нет стандартной процедуры, которая обеспечивала бы вывод на экран картинки, находящейся в файле. Программист сам может разработать такую процедуру. Однако эта задача является довольно сложной. Поэтому ниже приведен текст разработанной автором функции Draw, которая выводит на экран 16-цветную картинку, созданную в среде Windows, например, при помощи графического редактора Paint.

```
{ (с) Культин Н. Б., 2000 }
```

```
{ Выводит 16-цветную картинку,  
  содержимое bmp-файла }
```

```
function Draw(x0,y0: integer; fname: string;  
              transparent: boolean ):integer;  
{ x0,y0 — координаты левого верхнего угла  
  области вывода;  
  fname — имя файла картинки;  
  transparent = True — не выводить точки,  
                  цвет которых совпадает  
                  с цветом левой нижней точки  
                  картинки.
```

Значения функции:

успех — высота картинки в пикселах;

-1 — не найден файл;

-2 — картинка не является 16-цветной.

```
}
```

```
label
```

```
  bye;
```

const

```
{ таблица преобразования кодировки цвета из Windows в DOS }
color: array[0..15] of byte = (0,4,2,6,1,5,3,7,8,12,10,
                               14,9,13,11,15);
```

type

```
{ Прочитав из bmp-файла эту запись, можно получить
информацию о находящейся в этом файле картинке:
размере и кол-ве цветов. }
```

```
bpmpinfo = record
```

```
  hl,h2:      char;      { файл должен начинаться буквами BM }
  Size:       longInt;   { размер файла, байт }
  Reserved:   longint;   { резерв, не используется }
  Offset:     longint;   { смещение данных относительно
                           начала файла }

  b:          longint;   { не используется }
  Width:      longint;   { ширина картинки }
  Height:     longint;   { высота картинки }
  Plans:      word;      { кол-во планов, должно содержать 1 }
  bpp:        word;      { кол-во битов на пиксел (1, 4, 8
                           или 24 ) }
```

```
end;
```

var

```
f: file of bpmpinfo; { для получения информации о картинке }
bmpf: file of byte;  { для чтения картинки }
res: integer;         { код ошибки открытия файла }
```

```
info: bpmpinfo;
```

```
x,y:integer;         { координаты пиксела }
```

```
b: byte;             { байт, прочитанный из файла }
```

```
bh: byte;            { сдвинутые на четыре разряда четыре
старшие бита прочитанного байта }
```

```
bl: byte;            { четыре младшие бита прочитанного байта }
```

```
nb: integer;         { кол-во байт (кратное четырем)
соответствующее строке }
```

```
np: integer;         { кол-во выведенных пикселов строки }
```

```
tpcolor: byte;       { "прозрачный" цвет. Закрашенные этим
цветом точки картинки на экран
не выводятся }
```

```
i,j : integer;
```

begin

assign(f,fname);

{ \$I- }

reset(f);

{ \$I+ }

res:=IOResult;

if res <> 0 **then****begin**

Draw:=-1; { не найден файл картинки }

goto bye;**end;**

read(f,info); { читаем информацию о картинке }

close(f); { и закрываем файл }

if info.bpp <> 4 **then****begin**

Draw:=-2; { картинка не 16-цветная }

goto bye;**end;**

x:=x0;

y:=y0+info.height;

nb:= (info.width **div** 8)*4;**if** (info.width **mod** 8) <> 0 **then** nb:=nb+4;

assign(bmpf,fname);

reset(bmpf);

seek(bmpf,info.offset);

if transparent **then****begin**

read(bmpf,b);

tpcolor:=b **shr** 4;

seek(bmpf,info.offset);

end**else** tpcolor := 17; { в качестве прозрачного зададим
несуществующий цвет }**for** i:=1 **to** info.height **do****begin**

np:=0; { кол-во выведенных пикселей }

```
for j:=1 to nb do { вывод строки }
begin
  { if not EOF(bmpf) then }
  read(bmpf,b);
  if np < info.width then
  begin
    bh:= b shr 4;
    if bh <> tpcolor
      then putpixel(x,y,color[bh]);
    inc(x);
    inc(np);
  end;
  if np < info.width then
  begin
    bl:= b and 15;
    if bl <> tpcolor
      then putpixel(x,y,color[bl]);
    inc(x);
    inc(np);
  end;
end;
end;
x:=x0;
dec(y);
end;
close(bmpf);
Draw := info.height;
bye:
end;
```

Таблица кодировки символов

Символы, имеющие коды от 0 до 127:

0-	16-	32-	48-	64-	80-	96-	112-
1- ☉	17- ◀	33- !	49- 1	65- A	81- Q	97- a	113- q
2- ☿	18- ▶	34- "	50- 2	66- B	82- R	98- b	114- r
3- ♥	19- ⚡	35- #	51- 3	67- C	83- S	99- c	115- s
4- ♦	20- ⚡	36- \$	52- 4	68- D	84- T	100- d	116- t
5- ♣	21- ⚡	37- %	53- 5	69- E	85- U	101- e	117- u
6- ♠	22- -	38- &	54- 6	70- F	86- V	102- f	118- v
7-	23- ↑	39- '	55- 7	71- G	87- W	103- g	119- w
8-	24- ↓	40- (	56- 8	72- H	88- X	104- h	120- x
9- ○	25- ↓	41-)	57- 9	73- I	89- Y	105- i	121- y
10-	26- →	42- *	58- :	74- J	90- Z	106- j	122- z
11- ♂	27- ←	43- +	59- ;	75- K	91- [	107- k	123- {
12- ♀	28- L	44- ,	60- <	76- L	92- \	108- l	124-
13-	29- →	45- -	61- =	77- M	93-]	109- m	125- }
14- ♪	30- ▲	46- .	62- >	78- N	94- ^	110- n	126- ~
15- ♫	31- ▼	47- /	63- ?	79- O	95- _	111- o	127- 0

Символы, имеющие коды от 128 до 255:

128- A	144- P	160- a	176- ☐	192- L	208- ▬	224- p	240- Ё
129- Б	145- C	161- б	177- ☐	193- ⊥	209- ⊥	225- с	241- ё
130- В	146- T	162- в	178- ☐	194- ⊥	210- ⊥	226- т	242- €
131- Г	147- У	163- г	179- ☐	195- ⊥	211- ⊥	227- у	243- €
132- Д	148- Ф	164- д	180- ☐	196- ⊥	212- ⊥	228- ф	244- €
133- Е	149- X	165- е	181- ☐	197- ⊥	213- ⊥	229- х	245- €
134- Ж	150- Ц	166- ж	182- ☐	198- ⊥	214- ⊥	230- ц	246- €
135- З	151- Ч	167- з	183- ☐	199- ⊥	215- ⊥	231- ч	247- €
136- И	152- Ш	168- и	184- ☐	200- ⊥	216- ⊥	232- ш	248- €
137- Й	153- Щ	169- й	185- ☐	201- ⊥	217- ⊥	233- щ	249- •
138- К	154- Ъ	170- к	186- ☐	202- ⊥	218- ⊥	234- ъ	250- •
139- Л	155- Ы	171- л	187- ☐	203- ⊥	219- ⊥	235- ы	251- ✓
140- М	156- Ь	172- м	188- ☐	204- ⊥	220- ⊥	236- ь	252- №
141- Н	157- Э	173- н	189- ☐	205- =	221- ⊥	237- э	253- α
142- О	158- Ю	174- о	190- ☐	206- ⊥	222- ⊥	238- ю	254- ■
143- П	159- Я	175- п	191- ☐	207- ⊥	223- ⊥	239- я	255-

Символы с кодами 7, 8, 10 и 13 являются служебными и на экране не отображаются. Каждому служебному символу соответствует определенное действие.

Код символа Действие

- | | |
|----|---|
| 07 | Короткий звуковой сигнал |
| 08 | Перемещение курсора влево на одну позицию |
| 10 | Перемещение курсора на следующую строку без изменения его положения в колонке |
| 13 | Перемещение курсора в начало текущей строки |

Представление информации в компьютере

Десятичные, двоичные и шестнадцатеричные числа

В повседневной жизни человек имеет дело с десятичными числами. В десятичной системе счисления для представления чисел используются цифры от 0 до 9. Значение числа определяется как сумма произведений цифр числа на их весовые коэффициенты, которые зависят от места цифр в числе. Весовой коэффициент самой правой цифры равен единице, цифры перед ней — десяти, затем ста и т. д. Например, число 2703 можно представить как $2 \times 1000 + 7 \times 100 + 0 \times 10 + 3 \times 1$.

Если места расположения цифр (разряды) в числе пронумеровать справа налево и самой правой позиции присвоить номер ноль, то можно заметить, что вес i -го разряда равен i -ой степени десятки (рис. 1).

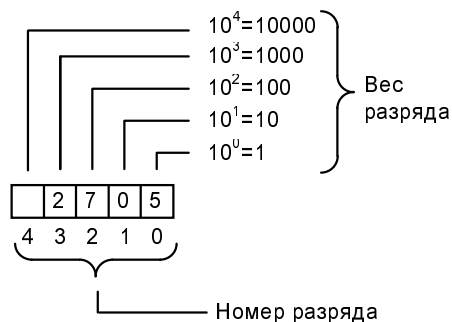


Рис. 1. Вес разрядов десятичного числа

Для внутреннего представления чисел в компьютере используется двоичная система счисления. Двоичные числа записываются при помощи двух цифр — нуля и единицы. Как и десятичная, двоичная система — позиционная. Весовой коэффициент разряда i -го равен двум в i -ой степени (рис. 2).

Двоичные числа наиболее точно отражают состояние памяти, регистров процессора и внешних устройств. Вместе с тем, рабо-

тать с двоичными числами не совсем удобно, т. к. приходится записывать слишком много цифр. Поэтому была разработана шестнадцатеричная система счисления, позволяющая компактно записывать двоичные числа и обеспечивающая простой способ перевода двоичного числа в шестнадцатеричное и обратно.

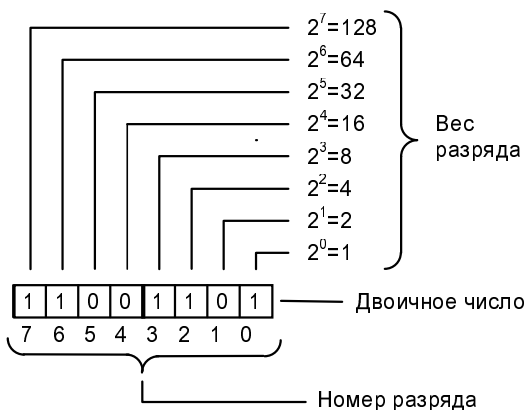


Рис. 2. Вес разрядов двоичного числа

Примечание

Следует отметить, что десятичные числа можно также рассматривать как компактное представление двоичных чисел, однако процесс перевода двоичного числа в десятичное и обратно нельзя назвать простым.

В основе шестнадцатеричной системы счисления лежит тот факт, что, используя четыре двоичные цифры, можно записать шестнадцать чисел (максимальное значение четырехразрядного двоичного числа равно пятнадцати). Схема получения шестнадцатеричного числа из двоичного представлена на рис. 3.

Цифры двоичного числа делятся на группы по четыре. Каждой группе ставится в соответствие сначала десятичное число, являющееся эквивалентом четырехзначного двоичного, затем полученное десятичное число записывается шестнадцатеричной цифрой. Ниже приведены десятичные числа (от нуля до 15) и соответствующие им шестнадцатеричные цифры.

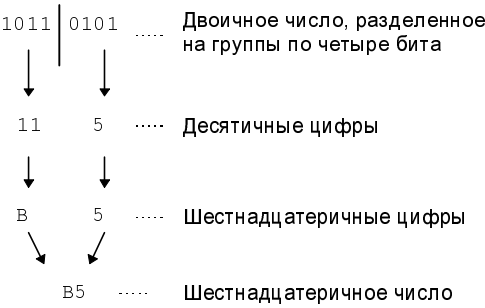


Рис. 3. Схема получения шестнадцатеричного числа из двоичного

Десятичное число	Шестнадцатеричная цифра	Десятичное число	Шестнадцатеричная цифра
0	0	8	8
1	1	9	9
2	2	10	A
3	3	11	B
4	4	12	C
5	5	13	D
6	6	14	E
7	7	15	F
8	8		

В тексте программы перед первой цифрой шестнадцатеричного числа ставится значок \$, например: \$2A, \$FF, \$01.

Память компьютера

В памяти компьютера числа хранятся в виде битов. Каждый бит может принимать значение одной двоичной цифры. Следовательно, значением бита может быть ноль или единица. Восемь

битов объединены в байт. Максимальное число, которое можно записать при помощи восьми двоичных цифр, — это 11111111, что соответствует десятичному числу 255, минимальное — ноль. Поэтому, значением байта может быть число от нуля до 255.

Переменные хранятся в памяти. Так как переменные различных типов могут принимать различные значения, то для их хранения нужен разный объем памяти. Память под переменные выделяется в байтах (целое число). Например, значением переменной типа `char` может быть любой из 256 символов. Поэтому для хранения переменной такого типа достаточно одного байта. Значением переменной типа `integer` может быть число от $-32\,768$ до $32\,767$ (65 535 значений). Для хранения переменной этого типа требуется два байта. Очевидно, что чем больше диапазон значений типа, тем больше байтов нужно для хранения переменной этого типа:

Тип переменной	Занимаемая память, байт	Диапазон значений
CHAR	1	Любой символ
STRING	256	Строка до 256 символов
STRING [N]	1 × n	Строка до n символов
BYTE	1	0–255
WORD	2	0–65 535
INTEGER	2	–32 768–32 767
LONGINT	4	–2 147 483 648–2 147 483 647
REAL	6	2.9E-39–1.7e38
SINGLE	4	1.5E-45–3.4e38
DOUBLE	8	5.0E-324–1.7e308
EXTENDED	8	3.4E-4932–1.1e4932

В программе для одного и того же значения можно использовать переменные разного типа (при этом они будут занимать разный объем памяти). Например, если в программе используется пере-

менная Day (число месяца), то для нее можно задать типы `byte`, `integer` или `longint`. В первом случае будет занят один байт памяти, во втором — два, в третьем — четыре. Однако реально будет использоваться только один байт, а остальные — просто заняты. Поэтому следует подбирать наиболее подходящий тип для каждой переменной. Особое внимание нужно обращать на описание строковых переменных и массивов.

Выделяя память для строковых переменных, помните, что если не указана предельная длина строки, то переменной выделяется 256 байтов. При объявлении переменной, предназначенной, например, для хранения имени человека, следует писать `name:string[30]`, а не `name:string`.

Каждому массиву программы выделяется память, объем которой определяется как типом элементов массива, так и их количеством. Например, для хранения двумерного массива вещественных чисел (например, 20×20) нужно более трех килобайт памяти ($20 \times 20 \times 8 = 3200$).

Хотя объем памяти компьютера кажется неограниченным, но если ее нерационально использовать, то в один прекрасный момент памяти может не хватить.

Список дополнительной литературы

1. Вирт Н. Алгоритмы и структуры данных: Пер. с англ. — М.: Мир, 1989. — 360 с.: ил.
2. Зелковиц М., Шоу А., Гэннон Дж. Принципы разработки программного обеспечения: Пер. с англ. — М.: Мир, 1982. — 386 с.: ил.
3. Культин Н. Б. Программирование в Turbo Pascal 7.0 и Delphi / Второе издание, переработанное и дополненное. — СПб.: БХВ — Санкт-Петербург, 1999. — 416 с.: ил.
4. Культин Н. Б. Программирование на Object Pascal в Delphi 5. СПб.: БХВ — Санкт-Петербург, 1999. — 464 с.: ил.
5. Практическое руководство по программированию: Пер. с англ. Б. Мик, П. Хит, Н. Рашби и др.; под ред. Б. Мик, П. Хит, Н. Рашби. — М.: Радио и связь, 1986. — 168 с.: ил.
6. Фокс Дж. Программное обеспечение и его разработка: Пер. с англ. — М.: Мир, 1985. — 368 с.: ил.
7. Энциклопедический словарь юного математика. Сост. А. П. Савин. — М.: Педагогика, 1989.—352 с.: ил.
8. Язык компьютера: Пер. с англ. / Под ред. и с предисл. В. М. Курочкина. — М.: Мир, 1989. — 240 с.: ил.

ПРЕДМЕТНЫЙ УКАЗАТЕЛЬ

А

Алгоритм Евклида 44

Б

Байт 251

Бит 250

В

Ввод из файла 67

Вывод

в файл 65, 69

иллюстрации 243

на принтер 69

Г

График 61, 62, 64

Д

Деловая графика 62, 63

Диаграмма 62

И

Игра

"Азбука морзе" 52

"Угадай число" 42, 52

Интеграл 39

К

Код символа 44, 45

М

Магический квадрат 50

Массив

поиск минимального

(максимального) элемента

47

поиск элемента 48, 49

слияние 49

сортировка 49

Мультипликация 62, 64

Н

Наибольший общий делитель 44

П**Поиск**

- в массиве 48
- в последовательности 42
- в упорядоченном массиве 49

Преобразование

- в строчные 45
- двоичное в десятичное 46
- десятичное в двоичное 38, 39
- десятичное в другое 46
- десятичное
 - в шестнадцатеричное 46
- шестнадцатеричное
 - в десятичное 46

Программа тестирования 39, 67**Простое число 42****Процедуры и функции**

- Abs 219
- Append 234
- Arc 224
- Arctan 219
- Assign 234
- Bar 225
- Bar3D 225
- Chr 223
- Circle 225
- Close 234
- ClrEol 236
- ClrScr 236
- Concat 223
- Copy 223
- Cos 219
- Delay 236
- Delete 223
- DetectGraph 226
- Dispose 237
- Ellipse 226
- EOF 234
- Eoln 237
- Erase 235
- Exp 220
- GetX 226
- GotoXY 237
- GraphResult 226
- Halt 237
- InitGraph 227
- Int 221
- IOResult 235
- Length 224
- Line 227
- LineTo 227
- Ln 220
- MoveRel 228
- MoveTo 228
- New 238
- OutText 228
- OutTextXY 228
- ParamCount 238
- ParamStr 238
- PieSlice 229
- Pos 224
- PutPixel 229
- Random 221
- Randomize 221
- ReadKey 238
- Rectangle 229
- Reset 235
- Rewrite 236
- Round 222
- Sector 230
- SetColor 230
- SetFillStyle 231
- SetLineStyle 232
- SetTextStyle 233
- Sin 220
- Sqr 220
- Sqrt 220
- Str 222
- TextBackGround 239
- TextColor 240
- Trunc 222
- Val 222
- WhereX 240
- WhereY 241
- Window 241

С

Система счисления
 двоичная 248
 десятичная 248
 шестнадцатеричная 249

Сортировка
 массива 51, 49
 методом "пузырька" 49
 методом обмена 49

Т

Таймер 43

Ф

Файл
 добавление 65
 просмотр 66
 создание 65

Ч

Число "ПИ" 43
Чтение из файла 65

Ш

Шестнадцатеричные числа 45, 250