Д. М. Златопольский

ЕГЭ ПО ИНФОРМАТИКЕ

РЕШЕНИЕ ЗАДАЧ ПО ПРОГРАММИРОВАНИЮ

Санкт-Петербург
«БХВ-Петербург»
2013

УДК 681.3.068(075.3)
ББК 32.973.26-018.1я721
З-67

Златопольский Д. М.

З-67 ЕГЭ по информатике. Решение задач по программированию. — СПб.: БХВ-Петербург, 2013. — 304 с.: ил. — (ИиИКТ)

ISBN 978-5-9775-0868-1

Книга предназначена для подготовки учащихся к Единому государственному экзамену по информатике в части решения задач по программированию. Рассмотрена методика решения основных типовых задач по программированию, а также заданий из демонстрационных вариантов ЕГЭ и из пособий, написанных разработчиками контрольно-измерительных материалов по информатике. Книга предназначена также студентам вузов и колледжей, преподавателям информатики и другим читателям при изучении программирования вне связи с ЕГЭ.

Для образовательных учреждений

УДК 681.3.068(075.3)
ББК 32.973.26-018.1я721

Группа подготовки издания:

Главный редактор	*Екатерина Кондукова*
Зам. главного редактора	*Людмила Еремеевская*
Зав. редакцией	*Екатерина Капалыгина*
Редактор	*Владимир Красовский*
Компьютерная верстка	*Ольги Сергиенко*
Корректор	*Зинаида Дмитриева*
Дизайн серии	*Инны Тачиной*
Оформление обложки	*Марины Дамбиевой*

Подписано в печать 31.08.12.
Формат 70×100 1/16. Печать офсетная. Усл. печ. л. 24,51.
Тираж 1200 экз. Заказ №
"БХВ-Петербург", 191036, Санкт-Петербург, Гончарная ул., 20.

Первая Академическая типография "Наука"
199034, Санкт-Петербург, 9 линия, 12/28

ISBN 978-5-9775-0868-1

Оглавление

Предисловие

На Едином государственном экзамене по информатике и ИКТ задания, связанные с программированием, занимают важное место. Так в [14] их 8[1] (А2, А14, В3, В6, В7, С1, С2, С4) при общем числе заданий — 32. При этом высока весомость заданий (максимальный балл за выполнение заданий группы части 3 равен 3–4).

Это говорит о том, что от умения решать задачи по программированию в значительной степени зависит успешность сдачи ЕГЭ в целом.

В то же время, как показывает опыт, такие задачи часто вызывают у школьников заметные трудности, особенно задачи части 3. В большой степени это связано с недостаточным числом часов, отводимых на изучение программирования в школе.

Данная книга должна восполнить этот недостаток — помочь учащимся подготовиться к экзамену самостоятельно. В ней системно, подробно и доступно описана методика решения задач по программированию, встречающихся в ЕГЭ.

Сначала в *главах 1* и *2* рассмотрены практически все частные, вспомогательные задачи, умение решать которые позволит успешно выполнить задания экзамена по программированию, после чего в *главах 3–7* описана методика решения задач С2 и С4 из ЕГЭ [2–7, 12, 16]. В *приложениях* приведены другие материалы, связанные с задачами программирования в Едином государственном экзамене.

При разработке программ (частных и из заданий ЕГЭ) использован школьный алгоритмический язык [11, 13]. Русский синтаксис этого языка и большое число комментариев сделают программы максимально понятными и легко переносимыми на любой другой язык программирования. Актуальность этого языка повышается в связи с тем, что он является одним из двух языков программирования, которые планируется использовать в компьютерном варианте ЕГЭ по информатике (его введение ожидается в 2013 году). В большинстве случаев приводятся также фрагменты программ на языке Паскаль. Полностью программы решения задач С4 на

[1] Без учета заданий, связанных с формальными исполнителями.

школьном алгоритмическом языке представлены в архиве, который выложен на FTP-сервер издательства по адресу: **ftp://ftp.bhv.ru/9785977508681.zip**. Ссылка доступна и со страницы книги на сайте **www.bhv.ru**.

Обратим внимание на широкое использование в книге задач от разработчиков контрольно-измерительных материалов для ЕГЭ [12, 16] — существует большая вероятность того, что подобные задачи будут включены в экзамен в будущем.

Кроме учащихся, готовящихся к сдаче экзамена самостоятельно, книгу могут использовать учителя и преподаватели информатики, а также студенты и учащиеся, изучающие программирование вне связи с ЕГЭ.

ГЛАВА 1

Задачи из Кодификатора для ЕГЭ

В Кодификаторе элементов содержания и требований к уровню подготовки выпускников общеобразовательных учреждений для проведения в 2012 году Единого государственного экзамена по информатике и ИКТ [10] приведен список возможных алгоритмических задач для подраздела 1.1 перечня требований к уровню подготовки выпускников, достижение которого проверяется на Едином государственном экзамене по информатике и ИКТ.

В перечисленных задачах представлены основные базовые программные конструкции, встречающиеся в задачах по программированию в ЕГЭ (в заданиях всех трех частей экзамена: 1, 2 и 3). Существует также большая вероятность того, что ряд задач будет включен в экзамен в будущем.

Далее рассмотрены возможные методы решения задач. Как отмечалось в *предисловии*, описание проводится с использованием школьного алгоритмического языка (в ряде случаев приводятся также аналогичные фрагменты программ на языке Паскаль).

1.1. Поиск минимума и максимума двух, трех, четырех данных чисел без использования массивов и циклов[1]

1.1.1. Поиск максимума/минимума среди двух чисел (*a* и *b*)

Для двух чисел задачи решаются так, как показано на рис. 1.1 и 1.2.

Обращаем внимание на тот факт, что случаи равенства переменных можно не рассматривать, т. к. нам безразлично, какое из двух равных значений считать максимумом или минимумом.

Соответствующие фрагменты программ на школьном алгоритмическом языке имеют вид[2]:

---

[1] Формулировки всех задач приведены согласно [10].

[2] Фрагменты программ, связанные с описанием переменных величин, вводом исходных данных, выводом результатов (как правило) и т. п., приводить не будем.

```
если a > b
   то
      макс := a
   иначе
      макс := b
все
```

и

```
если a < b
   то
      мин := a
   иначе
      мин := b
все
```

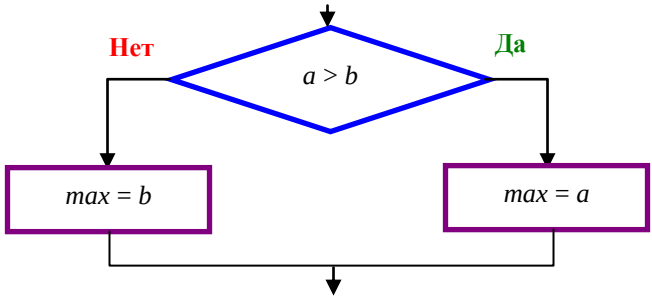


Рис. 1.1. Поиск максимума

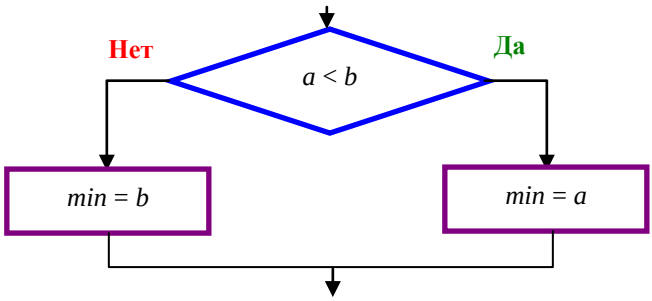


Рис. 1.2. Поиск минимума

Язык Паскаль

```
{Поиск максимума}
if a > b then max := a
  else max := b
```

```
{Поиск минимума}
if a < b then min := a
else min := b
```

Можно также использовать два неполных условных оператора:

1) при поиске максимума:

```
если a > b
  то
    макс := a
  все
если b > a
  то
    макс := b
все
```

2) при поиске минимума:

```
если a < b
  то
    мин := a
все
если b < a
  то
    мин := b
все
```

Язык Паскаль

```
{Поиск максимума}
if a > b then max := a;
if b < a then max := b;
{Поиск минимума}
if a < b then min := a;
if b < a then min := b;
```

1.1.2. Поиск максимума/минимума среди трех чисел (*a*, *b* и *c*)

Здесь также можно использовать три неполных условных оператора, каждый из которых соответствует отдельному варианту возможного ответа:

1) при поиске максимума:

```
если a > b и a > c
  то
    макс := a
все
если b > a и b > c
  то
    макс := b
```

```
все
если с > а и с > b
  то
    макс := с
все
```

2) при поиске минимума:

```
если а < b и а < с
  то
    мин := а
все
если b < а и b < с
  то
    мин := b
все
если с < а и с < b
  то
    мин := с
все
```

Язык Паскаль

```
{Поиск максимума}
if (a > b) and (a > c) then max := a;
if (b > a) and (b > c) then max := b;
if (c > a) and (c > b) then max := c;
{Поиск минимума}
if (a < b) and (a < c) then min := a;
if (b < a) and (b < c) then min := b;
if (c < a) and (c < b) then min := c;
```

Заметим, что попытка объединить любые два неполных условных оператора в один полный:

1) при поиске максимума:

```
если а > b и а > с
  то
    макс := а
все
если b > а и b > с
  то
    макс := b
  иначе
    макс := с
все
```

2) при поиске минимума:

```
если b < a и b < c
  то
    мин := b
все
если c < a и c < b
  то
    мин := c
  иначе
    мин := a
все
```

в ряде случаев дает неправильный результат. Например, результатом выполнения приведенных фрагментов в первом случае при `a = 5`, `b = 4`, `c = 3` будет значение величины `макс`, равное `3`, а во втором — при `a = 5`, `b = 4`, `c = 8` — значение величины `мин`, равное `5`.

Правильным вариантом решения является также такой — вместо трех неполных условных операторов применить один оператор для *поиска максимума*, построенный по следующему принципу:

1. Сначала сравниваются два любых числа, например *a* и *b*.
2. Если оказалось, что *a* больше, чем *b*, то именно число *a*, как возможный "кандидат" на статус максимума, мы сравниваем с третьим числом, *c*; если же, наоборот, число *b* больше, чем *a*, то именно *b* и сравнивается затем с числом *c*.
3. По результатам этого второго сравнения принимается окончательное решение о том, какое число из трех максимальное.

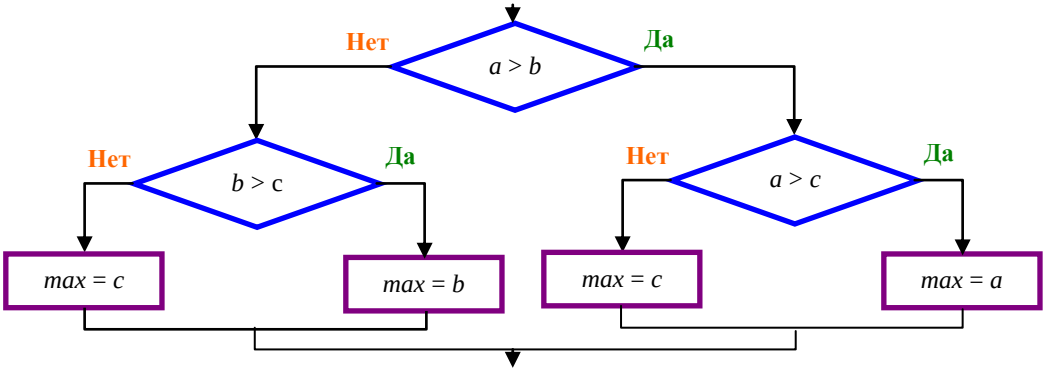


Рис. 1.3. Поиск максимума с помощью одного условного оператора

Эти рассуждения можно оформить в виде блок-схемы (рис. 1.3) и фрагмента программы:

```
если a > b
  то
    если a > c
```

```
        то
          макс := a
        иначе
          макс := c
      все
    иначе
      если b > c
        то
          макс := b
        иначе
          макс := c
      все
все
```

Заметим, что такую конструкцию называют *вложенный условный оператор*.

Язык Паскаль

```
if a > b then
  if a > c then max := a
  else max := c
else
  if b > c then max := b
  else max := c;
```

Аналогично, для *поиска минимума* также требуется конструкция из нескольких команд **если** (условных операторов), построенная по принципу:

1. Сначала сравниваются два любых числа, например *a* и *b*.
2. Если оказалось, что *a* меньше, чем *b*, то именно число *a*, как возможный "кандидат" на статус минимума, сравнивается с третьим числом, *c*; если же, наоборот, число *b* меньше, чем *a*, то именно *b* и сравнивается затем с числом *c*.
3. По результатам этого второго сравнения принимается окончательное решение о том, какое число из трех является минимальным.

Соответствующий фрагмент программы имеет вид:

```
если a < b
  то
    если a < c
      то
        мин := a
      иначе
        мин := c
    все
  иначе
    если b < c
      то
```

```
            мин := b
         иначе
            мин := c
      все
все
```

Язык Паскаль

```
if a < b then
   if a < c then
      min := a
   else
      min := c
else
   if b < c then
      min := b
   else
      min := c;
```

1.1.3. Поиск максимума/минимума среди четырех чисел (*a*, *b*, *c* и *d*)

Ситуация с четырьмя числами — наиболее сложная. Здесь при *поиске максимума* нужно вести сравнение (тоже с помощью цепочки многократно вложенных команд **если**) в следующем порядке:

- *a* сравнивается с *b*
- если *a* > *b*, то *a* сравнивается с *c*
  - если *a* > *c*, то *a* сравнивается с *d*
    - если *a* > *d*, то *макс* = *a*
      иначе *макс* = *d*
  - иначе *c* сравнивается с *d*
- если *c* > *d*, то *макс* = *c*

иначе *макс* = *d*

и т. д.

Приведем соответствующие фрагменты программ:

1) поиск максимума:

```
если a > b
  то
    если a > c
      то
        если a > d
          то
            макс := a
```

```
          иначе
            макс := d
        все
      иначе
        если c > d
          то
            макс := c
          иначе
            макс := d
        все
    все
  иначе |b > a
      если b > c
        то
          если b > d
            то
              макс := b
            иначе
              макс := d
          все
        иначе
          если c > d
            то
              макс := c
            иначе
              макс := d
          все
      все
все
```

Язык Паскаль

```
if a > b then
  if a > c then
    if a > d then
      max := a
    else
      max := d
  else
    if c > d then
      max := c
    else
      max := d
else {b > a}
  if b > c then
```

```
    if b > d then
      max := b
    else
      max := d
  else
    if c > d then
      max := c
    else
      max := d;
```

2) поиск минимума:

```
если a < b
  то
    если a < c
      то
        если a < d
          то
            мин := a
          иначе
            мин := d
        все
    иначе
      если c < d
        то
          мин := c
        иначе
          мин := d
      все
  иначе |b < a
    если b < c
      то
        если b < d
          то
            мин := b
          иначе
            мин := d
        все
      иначе
        если c < d
          то
            мин := c
          иначе
            мин := d
        все
    все
все
```

Язык Паскаль

```
if a < b then
  if a < c then
    if a < d then
      min := a
    else
      min := d
  else
    if c < d then
      min := c
    else
      min := d
else |b < a
  if b < c then
    if b < d then
      min := b
    else
      min := d
  else
    if c < d then
      min := c
    else
      min := d;
```

Полученные фрагменты программ — очень громоздкие и трудночитаемые. В них использованы по семь служебных слов **если** (**if**), **то** (**then**) и **иначе** (**else**)[1].

Можно упростить программу, если использовать *логические переменные*[2] [15].

Итак, пусть описаны переменные логического типа `x1`, `x2`, `x3`, `x4`, `x5`, `x6`.

Сначала мы присваиваем каждой такой переменной результат одного сравнения каждой возможной пары чисел:

```
x1 := a > b; x2 := a > c; x3 := a > d
x4 := b > c; x5 := b > d
x6 := c > d
```

[1] Чтение подобных "запутанных" фрагментов в определенной степени облегчает такое правило: "Каждое **else** (**иначе**) относится к тому ближайшему из предыдущих **if** (**если**), у которых нет «своего» **else** (**иначе**)". Проверьте это правило на приведенных фрагментах.

[2] Естественно, в средах программирования, в которых это возможно. В языках Бейсик (вариант QBasic) и Си можно использовать переменные числового типа.

Первые три логические переменные хранят результаты сравнения значения переменной a с каждой из трех остальных. Далее мы должны сравнить со всеми остальными значение переменной b, но вспомним, что с переменной a мы ее уже сравнивали и что требуемое сравнение b > a можно считать эквивалентным записи **не**(a > b). Поэтому достаточно записать сравнения переменной b только с переменными c и d. Аналогично переменную c нужно сравнивать только с переменной d.

Сформировав таким способом значения логических переменных, на их основе можно сформулировать логические условия, при которых та или иная из четырех имеющихся переменных является максимальной. Например, для переменной a:

```
если x1 и x2 и x3
  то
    макс := a
все
```

А когда максимальным является значение b? Это имеет место, когда b > a (т. е. значение переменной x1 — ложное) и при этом значение переменных x4 и x5 — истинное:

```
если не x1 и x4 и x5
  то
    макс := b
все
```

Рассуждая аналогично, можем записать:

```
если не x2 и не x4 и x6
  то
    макс := c
все
если не x3 и не x5 и не x6
  то
    макс := d
все
```

Язык Паскаль

```
x1 := a > b; x2 := a > c; x3 := a > d;
x4 := b > c; x5 := b > d; x6 := c > d;
if x1 and x2 and x3 then max := a;
if not x1 and x4 and x5 then max := b;
if not x2 and not x4 and x6 then max := c;
if not x3 and not x5 and not x6 then max := d;
```

В заключение заметим, что рассмотренные три задачи могут быть решены следующим образом.

Задача 1.1.1:

1) при поиске максимума:

```
макс := a
если b > макс
  то
    макс := b
все
```

2) при поиске минимума:

```
мин := a
если b < мин
  то
    мин := b
все
```

Задача 1.1.2:

1) при поиске максимума:

```
макс := a
если b > макс
  то
    макс := b
все
если c > макс
  то
    макс := c
все
```

2) при поиске минимума:

```
мин := a
если b < мин
  то
    ммн := b
все
если c < мин
  то
    мин := c
все
```

Задача 1.1.3:

1) при поиске максимума:

```
макс := a
если b > макс
  то
    макс := b
все
```

```
если c > макс
  то
    макс := c
все
если d > макс
  то
    макс := d
все
```

2) при поиске минимума:

```
мин := a
если b < мин
  то
    мин := b
все
если c < мин
  то
    мин := c
все
если d < мин
  то
    мин := d
все
```

Правда, здесь возникает вопрос: можно ли считать, что при их решении не использованы циклы? (Операторы цикла не применялись, но есть повторяющиеся, точнее — аналогичные, действия.)

1.2. Нахождение всех корней заданного квадратного уравнения

Если коэффициент a квадратного уравнения считать не равным нулю, то задача не должна представлять трудностей для решения.

Квадратное уравнение с коэффициентами a, b, c при $a \neq 0$ может иметь от 0 до 2 корней в зависимости от значения дискриминанта $d = b^2 - 4ac$:

- при $d > 0$ корней — два, и они вычисляются по формулам:

$$x1 = \frac{-b + \sqrt{d}}{2a};$$

$$x2 = \frac{-b - \sqrt{d}}{2a};$$

- при $d = 0$ корень — один:

$$x1 = \frac{-b}{2a};$$

- при $d < 0$ корней нет.

Так как возможны три варианта ответа, то, как и при решении *задачи 1.1.2*, в программе можно использовать три неполных условных оператора:

```
если d > 0
  то
    x1 := (-b + sqrt(d))/(2 * a)
    x2 := (-b - sqrt(d))/(2 * a)
    вывод нс, "Уравнение имеет два корня: "
    вывод "x1=", x1, " x2=", x2
все
если d = 0
  то
    x1 := (-b + sqrt(d))/(2 * a)
    вывод нс, "Уравнение имеет один корень: "
    вывод "x1=", x1
все
если d < 0
  то
    вывод нс, "Уравнение не имеет корней"
все
```

Язык Паскаль

```
if d > 0 then
  begin
    x1 := (-b + sqrt(d))/(2 * a);
    x2 := (-b - sqrt(d))/(2 * a);
    write('Уравнение имеет два корня: x1=', x1:7:2, ' x2=', x2:7:2)
  end;
if d = 0 then
  begin
    x1 := (-b + sqrt(d))/(2 * a);
    write('Уравнение имеет один корень: x1=', x1:7:2)
  end;
if d < 0 then
  write('Уравнение не имеет корней');
```

Здесь, как и при решении *задачи 1.1.2*, объединять любые два неполных условных оператора в один полный нельзя.

Можно вместо трех неполных условных операторов применить один вложенный условный оператор:

```
если d > 0
  то
    x1 := (-b + sqrt(d))/(2 * a)
    x2 := (-b - sqrt(d))/(2 * a)
    вывод нс, "Уравнение имеет два корня "
    вывод "x1=", x1, " x2=", x2
```

```
  иначе
    если d = 0
      то
        x1 := (-b + sqrt(d))/(2 * a)
        вывод нс, "Уравнение имеет один корень "
        вывод "x1=", x1
      иначе
        вывод нс, "Уравнение не имеет корней"
    все
все
```

Язык Паскаль

```
if d > 0 then
  begin
    x1 := (-b + sqrt(d))/(2 * a);
    x2 := (-b - sqrt(d))/(2 * a);
    write('Уравнение имеет два корня: 'x1=', x1:7:2, ' x2=', x2:7:2)
  end
else
  if d = 0 then
    begin
      x1 := (-b + sqrt(d))/(2 * a);
      write('Уравнение имеет один корень: x1=', x1:7:2)
    end
  else
    write('Уравнение не имеет корней');
```

Обратим внимание на то, что распространенной ошибкой при разработке программ решения обсуждаемой задачи является отсутствие скобок в знаменателе выражения для расчета корней.

1.3. Нахождение наибольшего общего делителя двух натуральных чисел (алгоритм Евклида)

Древнегреческий математик Евклид в своей знаменитой работе "Начала" (330–320 гг. до нашей эры) изложил алгоритм нахождения наибольшего общего делителя (НОД) двух натуральных чисел. Суть этого (считающегося самым древним) алгоритма состоит в следующем. Если число a больше числа b, то нужно от a отнять b, в противном случае, наоборот, от b отнять a и повторять эти действия до тех пор, когда a станет равно b. После этого искомый НОД будет равен одному из полученных чисел. Сказанное иллюстрирует следующая схема:

$$\text{НОД}(a, b) = \begin{cases} \text{НОД}(a - b, b), \text{ если } a > b \\ \text{НОД}(a, b - a), \text{ если } b < a \\ a \text{ (или } b\text{)}, \text{ если } a = b \end{cases}$$

Например, при $a = 26$, $b = 18$ имеем: НОД(26, 18) = НОД(8, 18) = НОД(8, 10) = НОД(8, 2) = НОД(6, 2) = НОД(4, 2) = НОД(2, 2) = 2.

Приведем фрагмент программы, работающий по этому алгоритму:

```
нц пока a <> b
  если a > b
    то
      a := a − b
    иначе
      b := b − a
  все
кц
НОД := a |Или НОД := b
```

Язык Паскаль

```
while a <> b do
  if a > b then a := a − b
    else b := b − a;
NOD := a {или NOD := b}
```

Как можно улучшить эту программу? Прежде всего, можно многократные вычитания заменить на определение остатка от деления одного целого числа на другое. Эти действия должны повторяться, пока ни одно из чисел не равно нулю:

```
нц пока a <> 0 и b <> 0
  если a > b
   то
     a := mod(a, b)
   иначе
     b := mod(b, a)
  все
кц
```

где `mod` — функция, возвращающая остаток от деления своего первого аргумента на второй. В других языках программирования для этого используется не функция, а специальная операция (как правило, знак этой операции также обозначается `mod`).

После окончания работы оператора цикла искомое значение НОД может быть определено так:

```
если a = 0
 то
  НОД := b
 иначе
  НОД := a
все
```

или, короче (и эффектно!), так:

```
НОД := a + b
```

Язык Паскаль

```
while (a <> 0) and (b <> 0) do
  if a > b
    then
      a := a mod b
    else
      b := b mod a;
NOD := a + b
```

Можно также применить оператор цикла с постусловием:

```
нц
  если a > b
   то
     a := mod(a, b)
   иначе
     b := mod(b, a)
  все
кц_при a = 0 или b = 0
НОД := a + b
```

Язык Паскаль

```
repeat
  if a > b
    then
      a := a mod b
    else
      b := b mod a
until (a = 0) or (b = 0);
NOD := a + b;
```

Дальнейшее усовершенствование: можно отказаться от многократного сравнения величин `a` и `b` — достаточно сделать это один раз, а потом учесть, что получаемый в цикле остаток всегда будет меньше, чем второй параметр функции `mod`. В приведенной далее программе, учитывающей это обстоятельство, использованы следующие новые величины:

- `макс` — максимальное из двух заданных чисел;
- `мин` — то же, минимальное;
- `остаток` — остаток от деления `макс` на `мин`.

```
|Определяем значения макс и мин
если a > b
 то
   макс := a
```

```
   мин := b
 иначе
   макс := b
   мин := a
все
остаток := mod(макс, мин)
нц пока остаток <> 0
  макс := мин
  мин := остаток
  остаток := mod(макс, мин)
кц
НОД := мин
```

Язык Паскаль

```
{Определяем значения max и min}
if a > b then
  begin
    max := a;
    min := b
  end
else
  begin
    max := b;
    min := a
  end;
ostatok := max mod min;
while ostatok <> 0 do
  begin
    max := min;
    min := ostatok;
    ostatok = max mod min
  end
NOD := min
```

И, наконец, самый короткий вариант решения задачи:

```
нц пока b <> 0
  остаток := mod(a, b)
  a := b
  b := остаток
кц
НОД := a
```

В его особенностях разберитесь самостоятельно. В частности, проанализируйте, правильно ли будет работать программа при $b > a$ и при $a = b$.

Язык Паскаль

```
while b <> 0 do
  begin
    ostatok := a mod b;
    a := b;
    b := ostatok
  end;
NOD := a;
```

1.4. Запись натурального числа в позиционной системе счисления с основанием меньшим или равным 10. Обработка и преобразование такой записи числа

Обозначим заданное натуральное десятичное число — *n*, а основание системы счисления, в которую нужно перевести заданное, — *основание*. Идея решения первой части задачи — получить все цифры записи числа *n* в системе с основанием *основание* и записать каждую из них в массив. Конечно, нужно получить и всю новую запись числа. После этого можно решать вторую часть задачи — обрабатывать полученную запись, точнее, массив с ее цифрами. Можно будет найти сумму цифр, максимальную цифру и т. п.

Напомним методику перевода целых чисел из десятичной системы счисления в систему с другим основанием. Необходимо определять остаток от деления заданного числа и всех промежуточных целочисленных частных на *основание* и делать это до тех пор, пока частное не станет равно нулю. Полученные остатки и представляют собой цифры новой записи числа.

В приведенном далее фрагменте программы кроме величин `n` и `основание` использованы также следующие основные переменные величины:

- `цифры` — массив с данными целого типа, в котором будут храниться цифры "нового" числа. Размер этого массива следует определить с учетом возможной длины новой записи числа;
- `кол_цифр` — фактическое количество цифр в новой записи числа.

Обратим внимание на то, что в массив цифры записываются начиная с последней цифры новой записи.

```
кол_цифр := 0
нц пока n > 0
  |Увеличиваем значение кол_цифр
  кол_цифр := кол_цифр + 1
  |Определяем очередную цифру и записываем ее в массив
  цифры[кол_цифр] := mod(n, основание)
  |Определяем целочисленное частное
  n := div(n, основание)
кц
```

```
|Выводим ответ (цифры в обратном порядке)
вывод нс, "Запись числа n в новой системе счисления: "
нц для i от кол_цифр до 1 шаг -1
  вывод цифры[i]
кц
```

где div — функция, определяющая целочисленное частное от деления своего первого аргумента на второй (в других языках программирования для этого используется не функция, а специальная операция).

Язык Паскаль

```
kol_zifr := 0;
while n > 0 do
  begin
    {Увеличиваем значение kol_zifr}
    kol_zifr := kol_zifr + 1;
    {Определяем очередную цифру}
    zifri[kol_zifr] := n mod osnovanie;
    {Определяем целочисленное частное}
    n := n div osnovanie
end;
{Выводим ответ (цифры в обратном порядке)}
write('Запись числа n в новой системе счисления: ');
for i := kol_zifr downto 1 do
  write(zifri[i]);
```

Если же считать, что в условии задачи требуется получить строковую (в терминах языка программирования Паскаль) величину новое, представляющую собой запись заданного числа в системе с основанием основание, то массив можно не использовать. Можно формировать значение величины новое, добавляя очередную найденную цифру в начало уже имеющегося значения:

```
новое := ""
нц пока n > 0
  |Определяем очередную цифру
  цифра := mod(n, основание)
  |Добавляем ее строковое представление
  |к значению новое
  новое := цел_в_лит(цифра) + новое
  n := div(n, основание)
кц
вывод нс, "Запись числа в новой системе счисления: ", новое
```

В последнем варианте применена величина цифра (смысл которой понятен), а также стандартная функция школьного алгоритмического языка цел_в_лит, осуществляющая преобразование числового значения в его символьное представление.

Полученное значение `новое` и можно использовать для решения второй части задачи, рассматриваемой в этом разделе (обработка и преобразование новой записи числа).

Язык Паскаль

```
novoe := '';
while n > 0 do
  begin
    {Определяем очередную цифру}
    zifra := n mod osnovanie;
    {Преобразуем ее в строку (символ)}
    Str(zifra, str_zifra);
    {и добавляем к значению novoe}
    novoe := str_zifra + novoe;
    n := n div osnovanie
  end;
{Выводим ответ}
write('Запись числа n в новой системе счисления: ', novoe);
```

Примечание

`Str` — процедура, преобразовывающая число — первый аргумент (в нашем случае — `zifra`) в строку символов — второй аргумент (`str_zifra`).

1.5. Нахождение сумм, произведений элементов данной конечной числовой последовательности (или массива)

В приведенных далее фрагментах программ использованы следующие основные величины:

- `n` — общее количество чисел в последовательности (оно может быть заранее задано в программе или вводиться в ходе ее выполнения);
- `a` — очередное обрабатываемое число последовательности.

Смысл величин `сумма` и `произведение` можно легко определить по их именам.

Так как количество обрабатываемых чисел *n* известно, в программах использован оператор цикла с параметром.

1.5.1. Суммирование всех чисел последовательности

```
сумма := 0
нц для i от 1 до n
  |Ввод очередного числа a последовательности
  ...
  сумма := сумма + a
кц
```

Примечание

В программах на языках программирования Бейсик, Паскаль, Си и ряде других оператор `сумма := 0` не является обязательным. Вместе с тем, "правилом хорошего тона" является начальное присваивание величине `сумма` нулевого значения (а в отдельных задачах это является обязательным).

Применительно к массиву задача рассмотрена в *разд. 1.8.4*.

Язык Паскаль

```
summa := 0;
for i := 1 to n do
  begin
    {Ввод очередного числа a последовательности}
    ...
    summa := summa + a
  end;
```

1.5.2. Нахождение произведения всех чисел последовательности

Здесь начальное присваивание `произведение := 1` является обязательным:

```
произведение := 1
нц для i от 1 до n
  |Ввод очередного числа a
  ...
  произведение := произведение * a
кц
```

Язык Паскаль

```
proizved := 1;
for i := 1 to n do
  begin
    {Ввод очередного числа a последовательности}
    ...
    proizved := proizved * a
  end;
```

Для массива (с именем `a`) решается аналогично:

```
произведение := 1
нц для i от 1 до n
  произведение := произведение * a[i]
кц
```

Язык Паскаль

```
proizved := 1;
for i := 1 to n do
  proizved := proizved * m[i]
```

В обоих случаях следует иметь в виду, что значение произведения не должно выходить за пределы, допускаемые для использованного типа данных.

1.6. Использование цикла для решения простых переборных задач (поиск наименьшего простого делителя данного натурального числа, проверка числа на простоту и т. д.)

В этом разделе мы рассмотрим методику решения трех задач.

1.6.1. Определить количество делителей натурального числа *n*

Прежде чем представлять методику, заметим, что в школьном алгоритмическом языке определить, является ли некоторое число *b* делителем числа *a*, можно с помощью функции `mod` *(см. разд. 1.3)*:

```
если mod(a, b) = 0
  то
   число b является делителем числа a
  иначе
   число b не является делителем числа a
все
```

Самый простой способ решения задачи — проверить по очереди делимость *n* на каждое из чисел 1, 2, 3 ... *n*. Соответствующий фрагмент программы имеет вид:

```
k := 0 |Искомое количество
нц для i от 1 до n
  |Если i — делитель числа n
  если mod(n, i) = 0
    то  |Увеличиваем значение k на 1
      k := k + 1
  все
кц
|Выводим ответ
вывод нс, k
```

Язык Паскаль

```
k := 0;
for i := 1 to n do
  {Если i — делитель числа n}
  if n mod i = 0 then
    {Увеличиваем значение k на 1}
    k := k + 1;
```

Однако легко убедиться, что в интервале $|n/2 + 1; n - 1|$ делителей числа *n* нет, т. е. количество проверок можно сократить почти вдвое:

```
k := 0
нц для i от 1 до div(n, 2)
  если mod(n, i) = 0
    то
      k := k + 1
  все
кц
|Учитываем в количестве делителей также число n
k := k + 1
|Выводим ответ
вывод нс, k
```

Язык Паскаль

```
k := 0;
for i := 1 to n div 2 do
  if n mod i = 0 then
    k := k + 1;
{Учитываем в количестве делителей также число n}
k := k + 1;
{Выводим ответ}
write(k);
```

Можно также учесть тот факт, что у нечетного *n* четных делителей быть не может:

```
k := 0
если mod(n, 2) = 0 |n — четное число
  то
    нц для i от 1 до div(n, 2)
      если mod(n, i) = 0
        то
          k := k + 1
      все
    кц
  иначе |n — нечетное число
    i := 1
    нц пока i <= div(n, 3)
      если mod(n, i) = 0
        то
          k := k + 1
      все
      i := i + 2 |Рассматриваем только нечетные возможные делители
    кц
все
|Учитываем в числе делителей также число n
k := k + 1
|Выводим ответ
вывод нс, k
```

Язык Паскаль

```
k := 0;
if n mod 2 = 0 then
  for i := 1 to n div 2 do
    if n mod i = 0 then k := k + 1
else {n — нечетное число}
  begin
    i := 1;
    while i <= n div 3 do
      begin
        if n mod i = 0 then k := k + 1;
        i := i + 2 {Рассматриваем только нечетные возможные делители}
      end
  end;
{Учитываем в числе делителей также число n}
k := k + 1;
{Выводим ответ}
write(k);
```

Количество проверок возможных делителей можно также существенно сократить, если учесть, что у каждого делителя *d* числа *n*, не превышающего $\sqrt{n}$, есть "симметричный" ему делитель, получающийся в результате деления *n* на *d*. Например, если *n* = 30, достаточно найти делители 1, 2, 3, 5 (целая часть квадратного корня из 30 равна 5), все прочие делители получаем следующим образом:

```
30 div 1 = 30;
30 div 2 = 15;
30 div 3 = 10;
30 div 5 = 6.
```

Правда, из приведенной методики есть одно исключение — если число *n* является точным квадратом, то у делителя *d*, равного $\sqrt{n}$, "симметричного" ему делителя нет.

Учтем это в нашей программе. При ее составлении можно использовать оператор цикла с параметром или оператор цикла с условием. В первом случае фрагмент программы будет иметь вид:

```
k := 0
нц для i от 1 до int(sqrt(n))
  если mod(n, i) = 0
    то
      |Учитываем i и "симметричный" ему делитель
      k := k + 2
  все
кц
```

```
|Проверяем, является ли число n точным квадратом
если int(sqrt(n)) = sqrt(n)
  то |Является;
     |тогда не учитываем один делитель
    k := k — 1
все
|Выводим ответ
вывод нс, k
```

во втором:

```
k := 0
i := 1
нц пока i <= int(sqrt(n))
  если mod(n, i) = 0
    то
      k := k + 2
  все
  |Переходим к проверке следующего числа
  i := i + 1
кц
|Проверяем, является ли число n точным квадратом
если (i - 1) * (i - 1) = n
  то
    k := k — 1
все
|Выводим ответ
вывод нс, k
```

Предпочтительным является второй вариант, поскольку в первом происходит сравнение значений `int(sqrt(n))` и `sqrt(n)`, а вещественные числа (`sqrt(n)`) в компьютере не всегда представлены точно, что может привести к неправильному результату указанного сравнения.

Здесь же заметим, что в первом варианте проверка того факта, что число *n* является точным квадратом, может проводиться в виде условия, не использующего вещественные значения:

```
i * i = n
```

или

```
(i - 1) * (i -1) = n
```

Конкретный вид правильного условия зависит от среды программирования (от того, какое значение имеет величина `i` — параметр цикла после окончания работы оператора)[1].

[1] Во втором варианте программы такое условие всегда имеет вид: `(i - 1) * (i -1) = n`.

Язык Паскаль

```
k := 0;
i := 1;
while i <= int(sqrt(n)) do
  begin
    if n mod i = 0 then k := k + 2;
    {Переходим к проверке следующего числа}
    i := i + 1
    end;
{Проверяем, является ли число n точным квадратом}
if (i - 1) * (i - 1) = n then k := k - 1;
{Выводим ответ}
write(k);
```

1.6.2. Определить, является ли заданное натуральное число простым

Наиболее простой способ решения задачи такой:

1. Определить количество делителей заданного числа *n (см. разд. 1.6.1).*
2. Если это количество равно двум, то число *n* — простое.

Можно сократить число проверок, не проверяя заданные четные числа (они — не простые), а для нечетных *n* — рассматривать в качестве возможных делителей только нечетные значения. Однако такой вариант усложняет программу.

1.6.3. Найти наименьший простой делитель данного натурального числа

Можно рассуждать так. Если заданное число *n* — четное, то ответ — 2 (число 1 простым не считается). В противном случае надо рассматривать в качестве искомых числа 3, 5, 7 ... *n* до тех пор, пока не встретится первое простое число, являющееся делителем числа *n*. Итак, общая схема алгоритма решения такая:

```
если n — четное
  то
    Ответ равен 2
  иначе
    Рассматриваем в качестве искомых числа 3, 5, 7 ... n
    если среди них встретится простое число,
         являющееся делителем числа n
      то
         Прекращаем проверку
  все
    Ответ равен встреченному числу
все
```

В программе на школьном алгоритмическом языке используем следующие величины:

- n — заданное натуральное число;
- ответ — искомый делитель;
- возм_простое — возможное простое число;
- i — возможный делитель числа возм_простое;
- k — количество делителей числа возм_простое;
- найден — величина логического типа, фиксирующая факт нахождения искомого делителя (в программах на языках Бейсик и Си можно использовать величину целого типа).

Соответствующий фрагмент программы:

```
если mod(n, 2) = 0
  то
    ответ := 2
  иначе
    |Ищем простые числа
    возм_простое := 3; найден := нет
    нц пока возм_простое <= n и не найден
      |Проверяем число возм_простое на "простоту",
      |найдя количество его делителей
      |Учитываем как делитель число 1
      k := 1
      i := 3
      нц пока i <= возм_простое
        если mod(возм_простое, i) = 0
          то
            |i — делитель числа возм_простое
            |Учитываем его в количестве делителей
            k := k + 1
        все
        |Переходим к следующему возможному делителю числа возм_простое
        i := i + 2
      кц
      |Если встретилось простое число, являющееся также делителем n
      если k = 2 и mod(n, возм_простое) = 0
        то |
          найден := да
          ответ := возм_простое
        иначе
          |Переходим к следующему возможному простому числу
          возм_простое := возм_простое + 2
      все
    кц
все
вывод нс, "Минимальный простой делитель: ", ответ
```

Видно, что проверка числа `возм_простое` на "простоту" проводится так, как указано в начале обсуждения *задачи 1.6.1*.

Язык Паскаль

```
if n mod 2 = 0 then otvet := 2
else {Ищем простые числа}
  begin
    vozm_prostoe := 3; naiden := false;
    while (vozm_prostoe <= n) and not naiden do
       begin
        {Проверяем число vozm_prostoe на "простоту",
        найдя количество его делителей.
        Учитываем как делитель число 1}
        k := 1;
        i := 3;
        while i <= vozm_prostoe do
          begin
            if vozm_prostoe mod i = 0 then
              {i — делитель числа vozm_prostoe
              Учитываем его в количестве делителей}
              k := k + 1;
            {Переходим к следующему возможному делителю
            числа vozm_prostoe}
            i := i + 2
         end;
        {Если встретилось простое число, являющееся также делителем n}
        if (k = 2) and (n mod vozm_prostoe = 0) then
          begin
            {Ответ найден}
            naiden := true;
            otvet := vozm_prostoe
          end
        else {Переходим к следующему возможному простому числу}
          vozm_prostoe := vozm_prostoe + 2
       end
  end;
write('Минимальный простой делитель : ', otvet);
```

Но оказывается, что можно решить задачу гораздо проще, а именно: полностью избавиться от проверки делителей на простоту. Будем рассуждать так. Если взять несколько первых простых чисел (2, 3, 5...), то можно утверждать, что все бóльшие числа, кратные им (4, 6, 9, 10, 12, 15...), простыми не являются. Поэтому можно рассматривать все числа подряд (или после рассмотрения двойки проверять только нечетные числа, что ускорит работу программы) и найти первое число, на которое

делится n. Если такого числа до $\sqrt{n}$ не окажется, то число n — простое, и именно оно равно искомому в задаче значению.

Приведем пример программы на школьном алгоритмическом языке:

```
если mod(n, 2) = 0
  то
    ответ := 2
  иначе
    возм := 3
    нц пока возм <= sqrt(n) и mod(n, возм) <> 0
      |Переходим к следующему нечетному числу
      возм := возм + 2
    кц
    если mod(n, возм) = 0
      то
        ответ := возм
      иначе
        ответ := n
    все
все
вывод нс, "Минимальный простой делитель: ", ответ
```

Язык Паскаль

```
if n mod 2) = 0 then otvet := 2
  else begin
         vozm := 3;
         while (vozm <= sqrt(n)) and (n mod vozm <> 0) do
               {Переходим к следующему нечетному числу}
                vozm := vozm + 2;
         if mod(n, vozm) = 0
           then otvet := vozm
           else otvet := n
       end;
write('Минимальный простой делитель : ', otvet);
```

1.7. Заполнение элементов одномерного и двумерного массива по заданным правилам

Возможные способы заполнения числовых массивов описаны в *приложении 3*.

1.8. Операции с элементами массива

1.8.1. Линейный поиск элемента

1.8.1.1. Проверка факта наличия в массиве элемента с заданными свойствами

Пример такой задачи: "Определить, есть ли в целочисленном массиве четное число".

Если перебрать все элементы массива, для каждого проверять условие, соответствующее заданным свойствам, и, если оно выполняется, выводить "Да, имеется", в противном случае выводить "Нет, такого элемента нет":

```
нц для i от 1 до n[1]

  если <условие>  |<условие> – условие, соответствующее
                  |заданным свойствам
    то
      вывод нс, "Да, имеется"
    иначе
      вывод нс, "Нет, такого элемента нет"
  все
кц
```

то на экран будет выведено несколько ответов (причем противоречащих друг другу). Ясно, что ответ (правильный) должен выводиться только один раз, и приведенное решение неприемлемо.

Еще одна типичная ошибка, встречающая при решении обсуждаемой задачи, — использование переменной (логического типа, или принимающей значения 0 и 1), которая фиксирует факт соблюдения для каждого проверяемого элемента заданных свойств, и вывод ответа в зависимости от значения этой переменной:

```
нц для i от 1 до n
  если <условие>
    то
     имеется := 1
    иначе
     имеется := 0
  все
кц
если имеется = 1
  то
   вывод нс, "Да, имеется"
  иначе
   вывод нс, "Нет, такого элемента нет"
все
```

[1] Здесь и далее принимается, что массив описан с индексами от 1 до *n*.

Здесь ответ, выводимый на экран один раз, может быть ошибочным — он зависит от того, обладает ли заданными свойствами последний элемент массива!

Наиболее простой и понятный способ решения с правильным результатом такой:

1. Подсчитать количество элементов массива с заданными свойствами (такая задача рассмотрена в *разд. 1.11.2*).
2. В зависимости от найденного количества вывести соответствующий ответ.

Так как искомое значение может оказаться в начале массива, то после его нахождения продолжать проверку остальных элементов массива нерационально. Желательно прекратить обработку массива после нахождения искомого элемента. Для этого следует использовать оператор цикла с условием. Каким должно быть это условие? Можно рассуждать так. Необходимо рассматривать элементы до тех пор, пока не встретится заданное число или пока массив не будет исчерпан. Например, для задачи, приведенной в начале раздела (определить, имеется ли в массиве четное число), соответствующий фрагмент должен быть оформлен в виде:

```
i := 1
нц пока i <= n и не mod(a[i], 2) = 0
  i := i + 1
кц
|Вывод результата
если i <= n
  то
   вывод нс, "Да, имеется"
  иначе
   вывод нс, "Нет, такого элемента нет"
все
```

Обратите внимание — такое оформление допустимо для программ на школьном алгоритмическом языке и на языке Си. В программе на языке Бейсик (вариант QBasic) при выполнении программы может появиться сообщение об ошибке, связанное с выходом за пределы массива. Дело в том, что на этом языке вторая часть сложного условия будет проверяться даже тогда, когда первая часть оказывается ложной (имеет место так называемая *полная проверка сложных условий*). Поэтому, когда искомого числа в массиве нет, при $i = n + 1$ произойдет обращение к элементу с несуществующим индексом ($a[n + 1]$). Аналогичная ошибка может иметь место и в программе на языке Паскаль при определенных настройках компилятора. Поэтому при использовании языков Бейсик (обязательно) и Паскаль (при необходимости) следует в цикле проводить проверку не до последнего, а до предпоследнего элемента. Например, в программе на языке Паскаль:

```
i := 1;
while (i < n) and not <условие> do i := i + 1;
```

В результате величина `i` не будет превышать значение `n`. Если значение элемента, на котором обработка массива остановилась (на последнем элементе или "досроч-

но"), равно заданному числу, то это определяет один из вариантов ответа, в противном случае — второй вариант ответа:

```
{Вывод результата}
if a[i] = <условие>
  then write('Да, имеется')
  else write('Нет, такого элемента нет');
```

Ясно, что положительный ответ будет выведен и при наличии в массиве нескольких элементов с заданными свойствами (удовлетворяющих заданному условию).

1.8.1.2. Поиск индекса элемента массива, равного некоторому числу

В приведенных далее фрагментах программы величина значение — это число, индекс которого (искомый_индекс) ищется.

```
нц для i от 1 до n
  если a[i] = значение
    то
     искомый_индекс := i
  все
кц
```

Прежде чем обсуждать представленный вариант, предлагаем читателю ответить на вопрос: индекс какого элемента будет найден, если в массиве окажутся несколько элементов, значение которых равно искомому?

Нетрудно заметить, что при таком оформлении, в случае если числа значение в массиве нет, результат будет неправильным. Чтобы учесть в программе возможность такого случая, необходимо изменить фрагмент следующим образом:

```
искомый_индекс := 0
нц для i от 1 до n
  если a[i] = значение
    то
     искомый_индекс := i
  все
кц
|Вывод результата
если искомый_индекс > 0
  то
   вывод нс, "Индекс этого элемента: ", искомый_индекс
  иначе
   вывод нс, "Такого числа в массиве нет"
все
```

Язык Паскаль

```
iskomi_index := 0;
for i := 1 to n do
  if a[i] = <значение> then iskomi_index := i;
```

```
{Вывод результата}
if iskomi_index > 0
  then  write('Индекс этого элемента: ', iskomi_index)
  else write('Такого числа в массиве нет');
```

1.8.1.3. Поиск индекса *первого* элемента массива, равного некоторому числу

Для решения такого варианта задачи можно в приведенной в предыдущем разделе программе использовать величину логического типа первый, принимающую значение "истина" (**да**), если элемент, равный заданному числу, встретился в массиве впервые, и "ложь" (**нет**) — в противном случае:

```
|Встреченный элемент будет первым
первый := да
искомый_индекс := 0
нц для i от 1 до n
  если a[i] = значение
    то
     |Проверяем, впервые ли встретилось в массиве заданное число
     если первый = да
       |Если впервые
       то
         |Найден искомый элемент массива
         |Запоминаем его индекс
         искомый_индекс := i
         |Другие элементы, равные заданному числу,
         |уже будут не первыми
         первый := нет
     все
  все
кц
|Вывод результата
если искомый_индекс > 0
  то
   вывод нс, "Индекс этого элемента: ", искомый_индекс
  иначе
   вывод нс, "Такого числа в массиве нет"
все
```

Язык Паскаль

```
pervi := true; {Встреченный элемент будет первым}
iskomi_index := 0;
for i := 1 to n do
  if a[i] = <значение> then
     {Проверяем, впервые ли встретилось в массиве заданное число}
```

```
    if pervi {Если впервые} then
       begin
         {Найден искомый элемент массива.
         Запоминаем его индекс}
        iskomi_index := i;
        {Другие элементы, равные заданному числу, уже будут не первыми}
        первый := false
       end;
{Вывод результата}
if iskomi_index > 0
  then  write('Индекс этого элемента: ', iskomi_index)
  else write('Такого числа в массиве нет');
```

Как и при решении *задачи 1.8.1.1*, можно прекратить обработку массива после нахождения (возможного) искомого элемента:

```
искомый_индекс := 1
нц пока искомый_индекс <= n и a[искомый_индекс] <> значение
  искомый_индекс := искомый_индекс + 1
кц
|Вывод результата
если искомый_индекс <= n
  то
   вывод нс, "Индекс этого элемента: ", искомый_индекс
  иначе
   вывод нс, "Такого числа в массиве нет"
все
```

Для нахождения первого элемента, равного некоторому числу, можно также провести проверку, начиная с последнего элемента массива:

```
искомый_индекс := 0
нц для i от n до 1 шаг -1
  если a[i] = значение
    то
     искомый_индекс := i
  все
кц
|Вывод результата
если искомый_индекс > 0
  то
   вывод нс, "Индекс этого элемента: ", искомый_индекс
  иначе
   вывод нс, "Такого числа в массиве нет"
все
```

Язык Паскаль

```
iskomi_index := 0;
for i := n downto 1 do
  if a[i] = <значение> then iskomi_index := i;
{Вывод результата}
if iskomi_index > 0
  then  write('Индекс этого элемента: ', iskomi_index)
  else write('Такого числа в массиве нет');
```

1.8.2. Вставка и удаление элементов в массиве

1.8.2.1. Удаление из массива *k*-го элемента со сдвигом всех расположенных справа от него элементов на одну позицию влево[1]

Операторы, осуществляющие сдвиг необходимых элементов:

```
a[k] := a[k + 1]
a[k + 1] := a[k + 2]
...
a[n − 1] := a[n]
```

можно оформить с использованием оператора цикла с параметром в виде:

```
нц для i от k до n − 1
  a[i] := a[i + 1]
кц
```

или

```
нц для i от k + 1 до n
  a[i − 1] := a[i]
кц
```

> ***Внимание!***
>
> После сделанных изменений следует учесть, что количество элементов исходного массива уменьшилось на 1.

Обратите также внимание на прием, который был применен: если сразу оператор цикла с параметром оформить сложно, следует выписать действия, являющиеся телом этого оператора, принять изменяющуюся величину в качестве параметра оператора цикла и определить его начальное и конечное значения. Этот прием будет использован и при решении нескольких следующих задач и в будущем.

Язык Паскаль

```
for i := k to n - 1 do
  a[i] := a[i + 1];
кц
```

[1] Мы начинаем обсуждение с задачи удаления элемента из массива, т. к. эта задача немного проще.

или

```
for i := k + 1 to n do
  a[i − 1] := a[i];
```

1.8.2.2. Вставка в массив заданного числа на *k*-е место со сдвигом *k*-го, (*k* + 1)-го, (*k* + 2)-го ... *последнего* элемента на одну позицию вправо

Прежде всего, заметим, что для решения задачи массив должен быть описан с учетом дополнительного элемента. Еще одна особенность заключается в том, что в данном случае операторы, реализующие сдвиг элементов вправо, мы не можем записать следующим образом:

```
a[k + 1] := a[k]
a[k + 2] := a[k + 1]
...
a[n] := a[n − 1]
```

Почему не можем, понятно? Правильный вариант — сдвиг проводить начиная с конца изменяемой части массива:

```
a[n] := a[n − 1]
a[n − 1] := a[n − 2]
...
a[k + 2] := a[k + 1]
a[k + 1] := a[k]
```

или используя при этом оператор цикла:

```
нц для i от n до k + 1 шаг -1
  a[i] := a[i − 1]
кц
```

После сдвига можно в *k*-ю ячейку записать заданное число:

```
a[k] := ...
```

Язык Паскаль

```
for i := n downto k + 1 do
  a[i] := a[i − 1]
a[k] := ...;
```

1.8.3. Перестановка всех элементов массива в обратном порядке

Ясно, что при решении будут повторяться обмены, т. е. в программе можно будет применить оператор цикла с параметром. Но какими должны быть начальное и конечное значения параметра цикла? Для ответа на этот вопрос составим табл. 1.1.

Таблица 1.1

Индекс элемента	С каким элементом он меняется значениями
1	С последним (n-м)
2	С предпоследним, (n – 1)-м
...	...
n – 1	Со вторым
n	С первым

Правильно? Нет, конечно! При таком обмене каждый элемент будет менять значения дважды, и в результате массив не изменится. Менять значения (в левом столбце таблицы) нужно только до половины массива (табл. 1.2).

Таблица 1.2

Индекс элемента	С каким элементом он меняется значениями
1	С n-м
2	С (n – 1)-м
...	...
n `div` 2 – 1	С (n – n `div` 2 + 2)-м
n `div` 2	С (n – n `div` 2 + 1)-м

Здесь `div` — знак операции целочисленного деления.

> ***Внимание!***
>
> Индекс в последней строке левого столбца таблицы нельзя рассчитывать как n/2, т. к. значение индекса элемента массива может быть только целым. Здесь возникает также вопрос: а что будет, если n — нечетное число? Ответ — в этом случае значения в табл. 1.2 не изменятся, а средний элемент (его индекс — n `div` 2 + 1) меняться не будет (убедитесь в этом, рассмотрев конкретные значения n).

Для записи соответствующего фрагмента программы необходимо знать, с каким элементом будет меняться значениями i-й элемент. Анализ табл. 1.2 показывает, что сумма индексов обмениваемых элементов равна n + 1. Значит, i-й элемент будет меняться с (n + 1 – i)-м.

Итак, задача решается с помощью такого оператора цикла:

```
нц для i от 1 до div(n, 2)
  |Меняем местами i-й и (n + 1 — i)-й элементы
  вспомогательная := a[i]
  a[i] := a[n — i + 1]
  a[n — i + 1] := вспомогательная
кц
```

Язык Паскаль

```
for i := 1 to n div 2 do
  {Меняем местами i-й и (n + 1 — i)-й элементы}
  begin
    vspom := a[i]
    a[i] := a[n — i + 1]
    a[n — i + 1] := vspom
  end;
```

1.8.4. Суммирование элементов массива

Для решения задачи необходимо последовательно обратиться ко всем элементам массива и учесть их значения в уже рассчитанной ранее сумме. Удобно использовать оператор цикла с параметром:

```
сумма := 0
нц для i от 1 до n
  сумма := сумма + a[i]
кц
```

Язык Паскаль

```
summa := 0
for i := 1 to n do
  summa := summa + a[i];
```

1.8.5. Проверка соответствия элементов массива некоторому условию

1.8.5.1. Проверка того факта, что все элементы массива соответствуют некоторому условию

Пример такой задачи: "Определить, все ли элементы массива являются положительными числами".

Задачи такого типа проще всего (и понятнее) решить следующими образом:

1. Подсчитать количество элементов массива, удовлетворяющих заданному условию (такая задача рассмотрена в *разд. 1.11.2*)[1].
2. Сравнить найденное количество с общим числом элементов массива и вывести соответствующий ответ.

1.8.5.2. Проверка массива на упорядоченность

Пример такой задачи: "Определить, упорядочены ли элементы массива по неубыванию, т. е. каждый элемент массива начиная со второго не меньше предыдущего".

[1] Можно также определить количество элементов массива, не удовлетворяющих заданному условию.

И здесь проще всего определить число элементов, которые не меньше предыдущего. Если это число равно $n - 1$, то ответ положительный, в противном случае — отрицательный:

1) начиная со второго элемента массива, подсчитать количество элементов, которые не меньше предыдущих:

```
количество := 0
нц для i от 2 до n
  |Сравниваем i-й элемент с предыдущим
  если a[i] >= a[i — 1]
    то
     количество := количество + 1
  все
кц
```

2) зная значение величины количество, можно вывести ответ:

```
если количество = n — 1
  то
    вывод нс, "Массив упорядочен по неубыванию"
  иначе
    вывод нс, "Массив не упорядочен по неубыванию"
все
```

Язык Паскаль

```
kolichestvo := 0;
for i := 2 to n do
  {Сравниваем (i — 1)-й элемент с предыдущим}
  if a[i] >= a[i — 1] then kolichestvo := kolichestvo + 1
if kolichestvo = n — 1
  then write('Массив упорядочен по неубыванию')
  else write('Массив не упорядочен по неубыванию');
```

Можно, как и при решении ряда предыдущих задач, применить оператор цикла с условием, прекратив проверку, как только встретится элемент массива меньше предыдущего.

1.9. Нахождение минимального (максимального) значения в данном массиве и количества элементов, равных ему, за однократный просмотр массива

1.9.1. Определение максимального элемента массива

Алгоритм решения этой задачи аналогичен алгоритму действий человека, который определяет максимальное значение в некоторой одномерной таблице с числами.

Сначала он смотрит в первую ячейку таблицы и запоминает записанное там число. Затем смотрит во вторую ячейку и, в случае если имеющееся там число больше запомненного, в качестве максимального запоминает новое число. Для остальных ячеек действия аналогичны.

Соответствующий фрагмент программы:

```
|Начальное присваивание значения искомой величине
максимальное := a[1]
|Рассматриваем остальные элементы
нц для i от 2 до n
  |Сравниваем i-й элемент со значением максимальное
  если a[i] > максимальное
    то
     |Принимаем встреченный элемент в качестве максимальное
     максимальное := a[i]
  все
кц
|Выводим ответ
вывод нс, максимальное
```

Язык Паскаль

```
{Начальное присваивание значения искомой величине}
max := a[1];
{Рассматриваем остальные элементы}
for i := 2 to n do
  {Сравниваем i-й элемент со значением max}
  if a[i] > max
    then {Принимаем встреченный элемент в качестве значения max}
       max := a[i];
{Выводим ответ}
write(max);
```

Если диапазон возможных значений элементов в массиве известен, то фрагмент программы решения задачи может быть оформлен так:

```
максимальное := минимальное
|Рассматриваем все элементы
нц для i от 1 до n
  если a[i] > максимальное
    то
     максимальное := a[i]
  все
кц
|Выводим ответ
вывод нс, максимальное
```

где минимальное — нижняя граница диапазона возможных значений.

Так, например, если известно, что в массиве все элементы неотрицательные, то можем записать:

```
максимальное := 0
нц для i от 1 до n
  если a[i] > максимальное
    то
     максимальное := a[i]
  все
кц
```

Язык Паскаль

```
{Начальное присваивание значения искомой величине}
max := minim;
{Рассматриваем все элементы}
for i := 1 to n do
  if a[i] > max then max := a[i]
{Выводим ответ}
write(max);
```

1.9.2. Определение минимального элемента массива

Задача решается аналогично предыдущей (конечно, с необходимыми изменениями).

1.9.3. Определение индекса максимального элемента массива

Здесь также алгоритм решения задачи аналогичен алгоритму действий человека, определяющего номер ячейки с максимальным значением в некоторой одномерной таблице с числами, — сначала он запоминает первое число и номер 1, а затем рассматривает второе число. Если оно больше того числа, которое помнил, то запоминает новое число и номер 2 и переходит к следующему, в противном случае просто переходит к следующему, третьему, числу и делает то же самое и т. д.

Фрагмент программы, в котором решается задача:

```
|Начальное присваивание значений искомым величинам
максимальное := a[1]
номер_максимального := 1
|Рассматриваем остальные элементы
нц для i от 2 до n
  если a[i] > максимальное
    то
     |Принимаем встреченный элемент в качестве максимальное
     максимальное := a[i]
     |а его индекс — в качестве номер_максимального
     номер_максимального := i
  все
кц
```

```
|Выводим ответ
вывод нс, номер_максимального
```

Язык Паскаль

```
{Начальное присваивание значения искомой величине}
max := a[1];
nomer_max := 1;
{Рассматриваем остальные элементы}
for i := 2 to n do
  {Сравниваем i-й элемент со значением max}
  if a[i] > max then
    begin
      {Принимаем встреченный элемент в качестве значения max}
       max := a[i];
      {а его индекс — в качестве значения nomer_max)
      nomer_max := i
   end;
{Выводим ответ}
write(nomer_max);
```

Возникает вопрос: индекс какого элемента будет найден, если в массиве есть несколько элементов с максимальным значением? Что надо изменить в приведенном фрагменте, чтобы находился индекс последнего элемента с таким значением?

Если диапазон возможных значений элементов массива известен, то можно оформить фрагмент так:

```
максимальное := минимальное
|Рассматриваем все элементы
нц для i от 1 до n
  если a[i] > максимальное
    то
    |Принимаем встреченный элемент в качестве максимальное
    максимальное := a[i]
    |а его индекс — в качестве номер_максимального
    номер_максимального := i
  все
кц
|Выводим ответ
вывод нс, номер_максимального
```

где минимальное — нижняя граница диапазона возможных значений.

Видно, что в результате определяются два значения — номер_максимального и максимальное.

Если последнее значение находить не требуется, то без этой величины можно вообще обойтись. В самом деле, если нам известно значение индекса максимального

среди рассмотренных элементов, то мы знаем и значение соответствующего элемента (оно равно a[номер_максимального]):

```
номер_максимального := 1
нц для i от 2 до n
  если a[i] > a[номер_максимального]
    то
     номер_максимального := i
  все
кц
вывод нс, номер_максимального
```

Язык Паскаль

```
{Начальное присваивание значения искомой величине}
nomer_max := 1;
{Рассматриваем остальные элементы}
for i := 2 to n do
  if a[nomer_max] > max
    then nomer_max := i
{Выводим ответ}
write(nomer_max);
```

1.9.4. Нахождение индекса минимального элемента

Задача решается способами, аналогичными описанным применительно к предыдущей задаче.

1.9.5. Нахождение минимального (максимального) элемента массива и количества элементов, равных ему

Задача может быть решена двумя способами:

1. За два прохода по массиву.
2. За один проход по массиву.

В первом случае решение очевидно — на первом проходе следует найти минимальный (максимальный) элемент массива *(см. задачи 1.9.1 и 1.9.2)*, а на втором — подсчитать количество элементов, равных минимальному (максимальному) значению *(см. задачу 1.11.2)*.

Методика решения задачи за один проход по массиву (как это предусмотрено перечнем в Кодификаторе [10]) описана в *главе 2*, т. к. такая задача встречается не только применительно к массиву, но и к последовательности чисел, вводимых в программу во время ее выполнения. В этом случае "однопроходный" вариант позволяет не хранить все обрабатываемые значения, а значит — экономить память.

1.10. Нахождение второго по величине (второго максимального или второго минимального) значения в данном массиве за однократный просмотр массива

Такие задачи также подробно рассмотрены в *главе 2*.

1.11. Операции с элементами массива, отобранными по некоторому условию (например, нахождение минимального четного элемента в массиве, нахождение количества и суммы всех четных элементов в массиве)

1.11.1. Нахождение суммы элементов массива с заданными свойствами (удовлетворяющих некоторому условию)

Отличие задач данного типа от *задачи 1.8.4* в том, что добавлять значение элемента массива к уже рассчитанной ранее сумме следует только тогда, когда элемент обладает заданными свойствами:

```
сумма := 0
нц для i от 1 до n
  |Если элемент обладает заданными свойствами
  если <условие>
    то
      |Учитываем его значение в сумме
      сумма := сумма + a[i]
  все
кц
```

где `<условие>` — заданное условие для суммирования. Это условие может определяться значением элемента массива `a[i]` или/и его индексом `i`.

Однако так можно оформить программу только в случае, когда известно, что в массиве имеется хотя бы один элемент с заданными свойствами. Если допускается, что таких элементов может не быть, то фрагмент, связанный с выводом ответа, должен иметь вид:

```
если сумма = 0
  то
    вывод нс, "Чисел, удовлетворяющих условию, в массиве нет"
  иначе
    вывод нс, "Сумма чисел, удовлетворяющих условию, равна ", сумма
все
```

> ***Примечание***
>
> Случай, когда элементы с заданными свойствами в массиве имеются, но их сумма равна нулю, не учитывается.

Язык Паскаль

```
summa := 0;
for i := 1 to n do
  {Если элемент обладает заданными свойствами}
  if <условие>
    then {Учитываем его значение в сумме}
      summa := summa + a[i];
if summa = 0
  then write('Чисел, удовлетворяющих условию, в массиве нет'
  else write('Сумма чисел, удовлетворяющих условию, равна ', summa);
```

1.11.2. Нахождение количества элементов массива с заданными свойствами

Особенность данной задачи в том, что в случае, когда элемент обладает заданными свойствами (удовлетворяет некоторому условию), искомое количество увеличивается на 1:

```
количество := 0
нц для i от 1 до n
  |Если элемент обладает заданными свойствами
  если <условие>
    то
      |Учитываем его в искомом количестве
      количество := количество + 1
  все
кц
вывод нс, "Количество чисел, удовлетворяющих условию, равно ", количество
```

В данном случае условие в команде **если** (в условном операторе) определяется значением элемента массива `a[i]` или одновременно значениями `a[i]` и `i`. Количество элементов, зависящих только от значения индекса `i`, может быть найдено без использования оператора цикла (убедитесь в этом!).

Язык Паскаль

```
kolichestvo := 0;
for i := 1 to n do
  {Если элемент обладает заданными свойствами}
  if <условие>
    then {Учитываем его в искомом количестве}
      kolichestvo := kolichestvo + 1; {или inc(kolichestvo)}
write('Количество чисел, удовлетворяющих условию, равно ', kolichestvo);
```

1.11.3. Нахождение среднего арифметического значения элементов массива с заданными свойствами

Для нахождения искомого значения необходимо определить сумму элементов массива с заданными свойствами и их количество. Такие две задачи мы уже решили ранее. Здесь их можно объединить в одном операторе цикла:

```
сумма := 0
количество := 0
нц для i от 1 до n
  |Если элемент обладает заданными свойствами
  если <условие>
    то
      |Учитываем его значение в сумме
      сумма := сумма + a[i]
      |и учитываем этот элемент в количестве
      количество := количество + 1
  все
кц
|Подсчет результата
среднее_арифметическое := сумма/количество
```

Обратите внимание на то, что многократно определять значение *среднее_арифметическое* в "теле" условного оператора (команды если) необходимости нет. Это можно сделать один раз после окончания оператора цикла. Однако может оказаться, что чисел, удовлетворяющих заданному условию, в массиве не окажется. В этом случае при расчете будет иметь место деление на ноль, что недопустимо. Правильное оформление:

```
...
|Подсчет и вывод результата
если количество > 0
  то
   среднее_арифметическое := сумма/количество
   вывод нс, "Среднее арифметическое: ", среднее_арифметическое
  иначе
   вывод нс, "Чисел, удовлетворяющих условию, в массиве нет"
все
```

Язык Паскаль

```
summa := 0;
kolichestvo := 0;
for i := 1 to n do
  {Если элемент обладает заданными свойствами}
  if <условие> then
    begin
```

```
      {Учитываем его значение в сумме}
      summa := summa + a[i];
      {и учитываем этот элемент в количестве}
       kolichestvo := kolichestvo + 1
    end;
{Подсчет и вывод результата}
if kolichestvo > 0 then
  begin
    srednee_arifm := summa/kolichestvo;
    write('Среднее арифметическое: ', srednee_arifm:7:2)
  end
else
  write('Чисел, удовлетворяющих условию, в массиве нет');
```

Примечание

В ряде случаев в заданиях ЕГЭ указывается требование о разработке программы, эффективной с точки зрения используемой памяти. Для учета этого требования величину среднее_арифметическое можно не использовать, а включить в команду вывод выражение для ее расчета:

```
если количество > 0
  то
    вывод нс, "Среднее арифметическое: ", сумма/количество
  иначе
    вывод нс, "Чисел, удовлетворяющих условию, в массиве нет"
все
```

1.11.4. Изменение значений элементов массива с заданными свойствами

Приведем соответствующий фрагмент:

```
нц для i от 1 до n
  если <условие>
    |Если элемент обладает заданными свойствами
    то
     |Меняем его значение
     a[i] := ...
  все
кц
```

(Об особенностях значения <условие> здесь и далее — см. в *разд. 1.11.1*.)

Язык Паскаль

```
for i := 1 to n do
   {Если элемент обладает заданными свойствами}
   if <условие> then
     {Меняем его значение}
      a[i] := ...;
```

1.11.5. Вывод на экран элементов массива с заданными свойствами

Задача в общем виде: "Дан массив. Вывести на экран его элементы, удовлетворяющие некоторому условию. Известно, что элементы с заданными свойствами в массиве имеются".

Соответствующий фрагмент:

```
нц для i от 1 до n
  |Если очередной элемент обладает заданными свойствами
  если <условие>
    то
     |Выводим его
     вывод a[i], " "
  все
кц
```

Язык Паскаль

```
for i := 1 to n do
  {Если очередной элемент обладает заданными свойствами}
  if <условие> then
    {Выводим его}
    write(a[i], ' ');
```

Вариант 2 задачи: "Дан массив. Вывести на экран те его элементы, которые удовлетворяют некоторому условию, а затем на отдельной строке — количество таких элементов. Известно, что в массиве есть как минимум один элемент с заданными свойствами".

Вспомнив методику решения *задачи 1.11.2*, можем записать:

```
количество := 0
нц для i от 1 до n
  |Если очередной элемент обладает заданными свойствами
  если <условие>
    то
     |Выводим его значение
     вывод a[i], " "
     |и учитываем его в искомом количестве
     количество := количество + 1
  все
кц
|Выводим также искомое количество
вывод нс, количество
```

Язык Паскаль

```
kolichestvo := 0;
for i := 1 to n do
  {Если очередной элемент обладает заданными свойствами}
```

```
  if <условие> then
    begin
      {Выводим его значение}
       write(a[i], ' ');
      {и учитываем его в искомом количестве }
      kolichestvo := kolichestvo + 1;
   end;
{Выводим также искомое количество}
write(kolichestvo);
```

Вариант 3 задачи: "Дан массив. Вывести на экран те его элементы, которые удовлетворяют некоторому условию. Если таких элементов нет, вывести на экран сообщение «Таких значений нет»".

Задача решается аналогично варианту 2 — изменения должны коснуться вывода сообщения вместо количества значений:

```
количество := 0
нц для i от 1 до n
  если <условие>
    то
      вывод a[i], " "
      количество := количество + 1
  все
кц
если количество = 0
  то
    |Выводим сообщение
    вывод нс, "Таких значений нет"
все
```

Язык Паскаль

```
...
if kolichestvo = 0 then
  {Выводим сообщение}
  write('Таких значений нет');
```

Вариант 4 задачи: "Дан массив. Вывести на экран те его элементы, которые удовлетворяют некоторому условию. Если такой элемент единственный, то на следующей строке вывести его индекс. Известно, что указанные элементы в массиве имеются".

Здесь надо также запоминать индекс элемента, удовлетворяющего заданному условию. Если он окажется единственным, то именно его индекс и будет выведен, в противном случае — индекс не понадобится:

```
количество := 0
нц для i от 1 до n
```

```
  |Если очередной элемент обладает заданными свойствами
  если <условие>
    то
      |Выводим его значение
      вывод a[i], " "
      |Запоминаем его индекс
      индекс := i
       |и учитываем его в количестве
      количество := количество + 1
  все
кц
если количество = 1
  то
    |Выводим индекс единственного элемента
    вывод нс, индекс
все
```

Язык Паскаль

```
kolichestvo := 0;
for i := 1 to n do
  {Если очередной элемент обладает заданными свойствами}
  if <условие> then
    begin
      {Выводим его значение}
      write(a[i], ' ');
      {Запоминаем его индекс}
     index := i;
      {и учитываем его в количестве}
      kolichestvo := kolichestvo + 1
   end;
if kolichestvo = 1 then
  {Выводим индекс единственного элемента}
  write(index);
```

1.11.6. Нахождение номеров (индексов) элементов массива с заданными свойствами

Задача в общем виде: "Дан массив. Вывести на экран индексы тех его элементов, которые удовлетворяют некоторому условию. Известно, что элементы с заданными свойствами в массиве имеются".

Задача решается аналогично варианту 1 предыдущей задачи (выводятся не значения, а индексы соответствующих элементов).

Вариант 2 задачи: "Дан массив. Вывести на экран индексы тех его элементов, которые удовлетворяют некоторому условию, а затем на отдельной строке — количе-

ство таких элементов. Известно, что в массиве есть как минимум один элемент с заданными свойствами".

Задача решается аналогично варианту 2 *задачи 1.11.5*.

Вариант 3 задачи: "Дан массив. Вывести на экран индексы тех его элементов, которые удовлетворяют некоторому условию. Если таких элементов нет, вывести на экран сообщение «Таких значений нет»".

Задача решается аналогично варианту 3 *задачи 1.11.5*.

Вариант 4 задачи: "Дан массив. Вывести на экран индексы тех его элементов, которые удовлетворяют некоторому условию. Если такой элемент единственный, то на следующей строке вывести его значение. Известно, что указанные элементы в массиве имеются".

Задача решается аналогично варианту 4 *задачи 1.11.5*.

В заключение обратим внимание на одно важное обстоятельство. Если в условии, которое при обсуждении рассмотренных задач было обозначено как `<условие>`, должно использоваться равенство с данными вещественного типа, значения которых рассчитываются, то его необходимо оформлять как неравенство с учетом точности вычислений с такими данными. Например, если сравниваются два значения *a* и *b*, рассчитанные с точностью до 1 цифры после десятичной точки, то условие следует оформить так:

```
abs(a - b) < 0.0001,
```

где `abs` — функция, возвращающая абсолютную величину ее аргумента.

1.11.7. Определение минимального значения среди тех элементов массива, которые удовлетворяют некоторому условию

Пример: нахождение минимального четного элемента массива.

Сначала для простоты примем, что известен тот факт, что числа, удовлетворяющие заданному условию, в исследуемом массиве имеются.

Задача решается во многом аналогично *задаче 1.9.2* с учетом того обстоятельства, что в качестве начального значения искомого минимума нельзя принимать значение первого элемента массива, т. к. он может не входить во множество элементов, удовлетворяющих заданному условию:

```
минимальное := X
нц для i от 1 до n
  если <условие>
    то |Встретилось число, удовлетворяющее заданному условию
     |Сравниваем его с величиной минимальное
     если a[i] < минимальное
       то
        минимальное := a[i]
```

```
      все
  все
кц
|Выводим ответ
вывод нс, минимальное
```

где X — число, о котором заведомо известно, что оно не меньше максимального из чисел, для которых определяется минимальное значение. Например, если в массиве записаны только отрицательные значения, то можно принять X равным нулю.

Язык Паскаль

```
minim := X;
for i := 1 to n do
   {Если очередной элемент обладает заданными свойствами}
   if <условие> then
      {Сравниваем его с величиной minim}
      if a[i] < minim
        then minim := a[i];
{Выводим ответ}
write(minim);
```

Примечание

Можно также сразу записать сложное условие: <условие> **и** a[i] < минимальное, однако при этом в программах на некоторых языках будет проводиться полная проверка условия даже в случаях, когда очередной элемент не обладает заданными свойствами *(см. разд. 1.8.1.1)*.

Если же такое значение заранее неизвестно, задача несколько усложняется. Можно поступить так:

1. Найти первое значение в массиве, удовлетворяющее заданному условию.
2. Принять его в качестве минимального.
3. Рассмотреть оставшиеся элементы и те из них, которые удовлетворяют заданному условию, сравнить с минимальным среди уже рассмотренных ранее.

Соответствующий фрагмент программы:

```
|Этап 1
i := 1
нц пока <заданное условие не соблюдается>
  i := i + 1
кц
|Первый элемент массива, удовлетворяющий заданному условию, найден.
|Его индекс равен i
|Этап 2
минимальное := a[i]
```

```
|Этап 3
нц для j от i + 1 до n
  если <условие>
    то
     если a[j] < минимальное
       то
         минимальное := a[j]
     все
  все
кц
|Выводим ответ
вывод нс, минимальное
```

Язык Паскаль

```
{Этап 1}
i := 1;
while <заданное условие не соблюдается> do
  i := i + 1
{Первый элемент массива, удовлетворяющий заданному условию, найден.
Его индекс равен i}
{Этап 2}
minim := a[i];
{Этап 3}
for j := i + 1 to n do
  if <условие> then
     if a[j] < minim then minim:= a[j]
{Выводим ответ}
write(minim);
```

Если же допускается, что чисел, удовлетворяющих заданному условию, в исследуемом массиве может не быть, в программе необходимо учесть возможность выхода за пределы массива. Основные этапы решения задачи в таком случае:

1. Найти (попробовать найти) первое значение в массиве, удовлетворяющее заданному условию.

```
i := 1
нц пока i <= n и <заданное условие не соблюдается>
  i := i + 1
кц
```

2. Если такое значение в массиве есть:

```
если i <= n
  то
    |Его индекс равен i
    |Принимаем его в качестве минимального
    минимальное := a[i]
```

```
      |Рассматриваем оставшиеся элементы и те из них,
      |которые удовлетворяют заданному условию,
      |сравниваем с минимальным среди уже рассмотренных
      нц для j от i + 1 до n
        если <условие>
          то
            если a[j] < минимальное
              то
                минимальное := a[j]
            все
        все
      кц
      |Выводим ответ
      вывод нс, минимальное
    иначе |Элементов с заданными свойствами в массиве нет
      вывод нс, "Таких чисел в массиве нет"
  все
```

Язык Паскаль

```
{Ищем (пробуем найти) первое значение в массиве, удовлетворяющее заданному
условию}
i := 1;
while (i <= n) and <заданное условие не соблюдается> do
  i := i + 1;
{Если такое значение в массиве есть}
if i <= n then
  begin
    {Его индекс равен i.
    Принимаем его в качестве минимального}
    minim := a[i];
    {Рассматриваем оставшиеся элементы и те из них, которые удовлетворяют
    заданному условию, сравниваем с минимальным среди уже рассмотренных}
    for j := i + 1 to n do
      if <условие> then
        if a[j] < minim then minim:= a[j];
    {Выводим ответ}
    write(minim)
  end
else {Элементов с заданными свойствами в массиве нет}
  write('Таких чисел в массиве нет');
```

В программах на языках Бейсик и Паскаль при необходимости следует учесть возможность выхода за пределы массива *(см. разд. 1.8.1.1)*.

Возможен также более компактный вариант решения:

```
индекс_мин := 0 |Условно
|Рассматриваем все элементы
нц для i от 1 до n
```

```
  если <условие>
    то
     |Встретился элемент массива,
     |удовлетворяющий заданному условию.
     |Если это произошло впервые или он меньше "старого" минимума
     если индекс_мин := 0 или a[i] < a[индекс_мин]
       то
        |Запоминаем его индекс в качестве значения индекс_мин
        индекс_мин := i
     все
  все
кц
|Выводим ответ
если индекс_мин > 0
  то
    вывод нс, a[индекс_мин]
  иначе
    вывод нс, "Таких чисел в массиве нет"
все
```

В его особенностях разберитесь самостоятельно.

Язык Паскаль

```
index_min := 0; {Условно.
Рассматриваем все элементы}
for i := 1 to n do
  if <условие> then
    {Встретился элемент массива, удовлетворяющий заданному условию.
    Если это произошло впервые или он меньше "старого" минимума}
    if (index_min = 0) or (a[i] < a[index_min] then
      {Запоминаем его индекс в качестве значения index_min}
      index_min := i;
{Выводим ответ}
if index_min > 0 then
  write(a[index_min]:7:2)
    else write('Таких чисел в массиве нет');
```

1.11.8. Определение индекса минимального элемента среди элементов массива, которые удовлетворяют некоторому условию

Задача решается аналогично *задаче 1.9.4* с учетом имеющихся здесь особенностей (см. также предыдущую задачу).

Ясно, что можно также разделить задачи нахождения индекса *первого* или *последнего* элемента с минимальным значением среди удовлетворяющих заданному условию.

Задачи типа 1.11.7 и 1.11.8 применительно к максимальному элементу решаются аналогично.

1.11.9. Нахождение максимального количества подряд идущих элементов массива, обладающих заданными свойствами

Пример задачи: "Определить максимальное количество подряд идущих четных элементов в заданном целочисленном массиве".

В дальнейшем для краткости будем называть участок массива с указанными элементами "подмассивом".

Находить максимальное значение из нескольких чисел мы умеем (это *задача 1.9.1*). Значит, для решения обсуждаемой задачи осталось научиться определять длину каждого подмассива.

Используем в программе следующие основные величины:

- кол — число элементов в текущем подмассиве;
- макс_кол — искомое значение (максимальное количество подряд идущих элементов с заданными свойствами).

Можно рассуждать так. Если очередной элемент обладает заданными свойствами, то подмассив таких элементов продолжается или начался — увеличиваем его длину кол на 1, иначе — текущий подмассив кончился, и в этом случае:

1. Сравниваем его длину с максимальной длиной уже рассмотренных ранее подмассивов макс_кол. Если длина текущего подмассива больше, то принимаем ее в качестве нового значения величины макс_кол.
2. Имея в виду обработку следующего подмассива, обнуляем значение величины кол.

Фрагмент программы на основе сделанных рассуждений:

```
кол := 0
макс_кол := 0
нц для i от 1 до n
  если <условие>
   то
    |Подмассив продолжается или начался
    |Увеличиваем его длину на 1
    кол :=  кол + 1
   иначе |Встретился элемент, не обладающий заданными свойствами
    |Текущий подмассив кончился
    |Сравниваем его длину со значением макс_кол
    если кол >  макс_кол
     то
      макс_кол := кол
    все
```

```
    кол := 0 |Новое значение
  все
кц
```

Однако при таких рассуждениях в случае, когда последний, *n*-й, элемент массива также обладает заданными свойствами (например, четный), последний подмассив не будет учтен. Значит, нужно дополнительно учесть и его, сравнив длину этого подмассива с максимальной длиной:

```
|Проверяем длину последнего (возможного) подмассива
если кол > макс_кол
  то
    макс_кол := кол
все
|Выводим ответ
вывод нс, макс_кол
```

Обратим внимание на то, что после окончания текущего подмассива величине кол присваивается нулевое значение независимо от результата сравнения величин кол и макс_кол.

Язык Паскаль

```
kol := 0;
max_kol := 0;
for i := 1 to n do
  if <условие>
   then {Подмассив продолжается или начался.
         Увеличиваем его длину на 1}
     kol := kol + 1
   else {Встретился элемент, не обладающий
        заданными свойствами — текущий подмассив кончился.
        Сравниваем его длину со значением max_kol}
      begin
       if kol > max_kol then max_kol := kol;
       kol := 0 {Новое значение}
      end;
{Проверяем длину последнего (возможного) подмассива}
if kol > max_kol then max_kol := kol;
{Выводим ответ}
write(max_kol);
```

Возможны также задачи рассмотренного типа, в которых начальное значение величины кол и ее значение после окончания некоторого подмассива принимается равным 1. Это имеет место, когда первый элемент подмассива также учитывается в его длине.

Пример: "Определить максимальное количество подряд идущих совпадающих элементов целочисленного массива".

Ясно, что здесь первый из совпадающих элементов также должен быть учтен. Поэтому начальные значения переменных величин, хранящих длину текущего подмассива и максимальную длину подмассивов, равны 1 (если потом выяснится, что во всех парах находящихся рядом элементов массива нет одинаковых, то искомая величина будет равна 1):

```
кол_совп := 1
макс_совп := 1
нц для i от 2 до n
  если m[i] = m[i − 1]
    то
      кол_совп := кол_совп + 1
    иначе
      если кол_совп >  макс_совп
        то
          макс_совп := кол_совп
      все
      кол_совп := 1 |Новое значение
  все
кц
если кол_совп >  макс_совп
  то
    макс_совп := кол_совп
все
вывод макс_совп
```

где кол_совп — количество подряд идущих совпадающих элементов в текущем подмассиве; макс_совп — максимальное количество подряд идущих совпадающих элементов в подмассивах.

Язык Паскаль

```
kol_sovp := 1;
max_ sovp := 1;
for i := 2 to n do
  if m[i] = m[i − 1] then
        kol_sovp := kol_ sovp + 1
   else
      begin
        if kol_sovp >  max_sovp
            then max_sovp := kol_sovp;
        kol_sovp := 1 {Новое значение}
      end;
if kol_sovp >  max_sovp
   then max_sovp := kol_sovp;
write(max_sovp);
```

1.11.10. Нахождение максимальной суммы подряд идущих элементов массива, обладающих заданными свойствами

Пример задачи: "Определить максимальную сумму подряд идущих четных элементов в заданном целочисленном массиве".

Задача решается во многом аналогично предыдущей (при обработке подмассива рассчитывается сумма значений элементов, а по его окончанию — сумма сравнивается с максимальной суммой, найденной ранее). Конечно, и здесь нужно при необходимости дополнительно проверить возможный последний подмассив:

```
сум := 0
макс_сум := 0
нц для i от 1 до n
  если <условие>
   то
    |Подмассив продолжается или начался
    |Учитываем текущий элемент в сумме
    сум := сум + m[i]
   иначе |Встретился элемент, не обладающий заданными свойствами
    |Текущий подмассив кончился
    |Сравниваем сумму его элементов со значением макс_сум
    если сум >  макс_сум
     то
      макс_сум := сум
    все
    сум := 0 |Новое значение
  все
кц
|Проверяем сумму элементов последнего (возможного) подмассива
если сум >  макс_сум
  то
    макс_сум := сум
все
вывод нс, макс_сум
```

Язык Паскаль

```
summa := 0;
max_summa := 0;
for i := 1 to n do
  if <условие>
   then {Подмассив продолжается или начался.
         Увеличиваем его длину на 1}
     summa := summa + m[i]
   else {Встретился элемент, не обладающий
        заданными свойствами — текущий подмассив кончился}
```

```
      begin
        {Сравниваем сумму его элементов со значением max_summa}
        if summa > max_summa then max_summa := summa;
        summa := 0 {Новое значение}
      end;
{Проверяем сумму элементов последнего (возможного) подмассива}
if summa > max_summa then max_summa := summa;
{Выводим ответ}
write(max_summa);
```

Возможны также задачи обсуждаемого типа, в которых начальное значение величины сум и ее значение после окончания некоторого подмассива принимается равным первому элементу массива (и, соответственно, текущему). Это имеет место, когда первый элемент подмассива также учитывается в его сумме.

Пример: "Определить сумму элементов наибольшей возрастающей последовательности подряд идущих элементов массива".

Обратим внимание на то, что здесь необходимо определить не максимальную длину[1] (количество элементов) подмассива и не максимальную сумму его элементов, а сумму элементов в подмассиве максимальной длины. Значит, нужно в ходе поиска такого подмассива рассчитывать и запоминать также сумму его элементов (в том числе и первого элемента подмассива).

В программе решения приведенного примера используем следующие основные величины:

- кол_возр — количество элементов возрастающей последовательности в текущем подмассиве;
- сумма_возр — сумма значений элементов в таком подмассиве;
- макс_кол — максимальное количество элементов в возрастающих последовательностях элементов;
- макс_сумма — максимальная сумма значений элементов в соответствующей последовательности.

Соответствующий фрагмент программы:

```
кол_возр := 1       |Учитываем первый
сумма_возр := m[1]  |элемент
макс_кол := 1
нц для i от 2 до n
  если m[i] > m[i - 1]
    то
      кол_возр := кол_возр + 1
      сумма_возр := сумма_возр + m[i]
```

[1] Хотя эту величину также придется рассчитывать, но как вспомогательную.

```
    иначе
      если кол > макс_кол
        то
          макс_кол := кол_возр
          макс_сумма := сумма_возр
      все
      |Новые значения
      сумма_возр := m[i]
      кол_возр := 1
  все
кц
если кол_возр > макс_кол
  то
    макс_сумма := сумма_возр
    |Здесь уточняем только значение макс_сумма
все
вывод нс, макс_сумма
```

Обратим внимание на то, что начальные значения величины кол_возр принимаются равными 1. Заметим также, что в решении, приведенном в [12], начальное значение величины макс_кол принимается равным 0. Очевидно, авторы допускают возможность того, что в массиве не будет двух и более элементов, образующих возрастающую последовательность.

Язык Паскаль

```
kol_vozr := 1;       {Учитываем первый}
summa_vozr := m[1]; {элемент}
max_kol := 1;
for i := 2 to n do
  if m[i] m[i] > m[i — 1] then
      begin
        kol_vozr := kol_vozr + 1
        summa_vozr := summa_vozr + m[i]
      end
   else
      begin
        if kol_vozr > max_kol then
          begin
            max_kol := kol_vozr;
            max_summa := summa_vozr
          end;
        {Новые значения}
        summa_vozr := m[i];
        kol_vozr := 1
      end;
```

```
if kol_vozr > max_kol
  then max_summa := summa_vozr; {Здесь уточняем только значение max_summa}
write(max_summa);
```

1.12. Сортировка массива

Рассмотрению двух простейших методов сортировки массивов посвящено *приложение 4*.

1.13. Слияние двух упорядоченных массивов в один без использования сортировки

Другая формулировка задачи: "Из двух массивов, каждый из которых упорядочен, получить новый массив, состоящий из элементов двух заданных массивов, таким же образом упорядоченный".

Рассмотрим методику решения задачи на примере объединения (слияния) двух массивов, упорядоченных по неубыванию.

Основная идея решения заключается в следующем. Сравним первые элементы обоих заданных массивов и меньший из них запишем на 1-е место в третий массив. В массиве, из которого элемент уже "использован", после этого будем рассматривать следующий элемент. Затем опять сравним очередные элементы и меньший из них запишем на 2-е место в третий массив и т. д.

На какое место в третьем массиве `c` записываем очередной элемент в общем случае? Для ответа на этот вопрос обозначим индексы сравниваемых в двух заданных массивах элементов *ук*1 и *ук*2 и составим таблицу (табл. 1.3):

Таблица 1.3

ук1	ук2	На какое место записываем	Комментарий
1	1	1	
1	2	2	
2	1	2	
2	2	3	
5	3	7	Из 1-го массива уже записаны четыре числа, из 2-го — 2, всего в массиве *c* шесть значений

Из табл. 1.3 можно сделать вывод, что индекс элемента в третьем массиве, которому присваивается очередное значение, равен *ук*1 + *ук*2 – 1.

Действия по сравнению элементов из массивов `a` и `b` и записи меньшего из них в массив `c` можно смоделировать так:

```
если a[ук1] < b[ук2]
  то
    |Записываем элемент из 1-го массива
    c[ук1 + ук2 - 1] = a[ук1]
    |Смещаем "указатель" в этом массиве
    ук1 := ук1 + 1
  иначе
    |Записываем элемент из 2-го массива
    c[ук1 + ук2 - 1] = b[ук2]
    |Смещаем "указатель" в этом массиве
    ук2 := ук2 + 1
все
```

Указанные действия должны повторяться, пока не будет полностью "использован" один из массивов:

```
ук1 := 1 |Начальные
ук2 := 1 |значения
нц пока ук1 <= n1 и ук2 <= n2
  если a[ук1] < b[ук2]
    ... (см. выше)
кц
```

или (при использовании оператора цикла с постусловием):

```
ук1 := 1
ук2 := 1
нц
  если a[ук1] < b[ук2]
    ...(см. выше)
кц_при ук1 > n1 или ук2 > n2
```

После этого необходимо переписать все оставшиеся элементы в одном из массивов (в каком — не знаем!) в массив c:

```
если ук1 <= n1
  то
    |"Остались" элементы в 1-м массиве
    нц пока ук1 <= n1
      |Записываем элемент из 1-го массива
      c[ук1 + ук2 - 1] = a[ук1]
      |Смещаем "указатель" в этом массиве
      ук1 := ук1 + 1
    кц
  иначе
    |"Остались" элементы во 2-м массиве
    нц пока ук2 <= n2
      |Записываем элемент из 1-го массива
      c[ук1 + ук2 - 1] = b[ук1]
```

```
      |Смещаем указатель в этом массиве
      ук2 := ук2 + 1
    кц
```

Последний фрагмент можно упростить:

```
нц пока ук1 <= n1
  |Записываем элемент из 1-го массива
  c[ук1 + ук2 - 1] = a[ук1]
  |Смещаем "указатель" в этом массиве
  ук1 := ук1 + 1
кц
нц пока ук2 <= n2
  |Записываем элемент из 1-го массива
  c[ук1 + ук2 - 1] = b[ук1]
  |Смещаем "указатель" в этом массиве
  ук2 := ук2 + 1
кц
```

Можно также использовать операторы цикла с параметром[1]:

```
если ук1 <= n1
  то
   |"Остались" элементы в 1-м массиве
   |Записываем их в массив c
   нц для i от ук1 до n1
     c[ук2 + i] = a[ук1]
     |Или c[n2 + i] = a[ук1]
   кц
  иначе
    |"Остались" элементы во 2-м массиве
    |Записываем их в массив c
    нц для i от ук2 до n2
      c[ук1 + i] = a[ук2]
      |Или c[n1 + i] = a[ук2]
    кц
все
```

Итак, весь фрагмент программы, относящийся к слиянию двух упорядоченных массивов a и b, имеет вид:

```
ук1 := 1
ук2 := 1
нц пока ук1 <= n1 и ук2 <= n2
  если a[ук1] < b[ук2]
    то
      c[ук1 + ук2 - 1] = a[ук1]
      ук1 := ук1 + 1
```

[1] Обратите внимание на индекс элемента в массиве c, которому присваивается очередное значение.

```
    иначе
      с[ук1 + ук2 - 1] = b[ук2]
      ук2 := ук2 + 1
  все
кц
если ук1 <= n1
  то
    нц для i от ук1 до n1
      с[ук2 + i] = a[ук1]
      |Или с[n2 + i] = a[ук1]
    кц
  иначе
    нц для i от ук2 до n2
      с[ук1 + i] = a[ук2]
      |Или с[n1 + i] = a[ук2]
    кц
все
```

Язык Паскаль

```
yk1 := 1;
yk2 := 1;
while (k1 <= n1) and (yk2 <= n2) do
  if a[yk1] < b[yk2] then
    begin
      c[yk1 + yk2 - 1] = a[yk1];
      yk1 := yk1 + 1
    end
   else
    begin
      c[yk1 + yk2 - 1] = b[yk2];
      yk2 := yk2 + 1
    end;
if yk1 <= n1 then
  for i := yk1 to n1 do
      c[yk2 + i] = a[yk1]
      {Или c[n2 + i] = a[yk1]}
else
    for i := yk2 to n2 do
      c[yk1 + i] = a[yk2]
      {Или c[n1 + i] = a[yk2]}
```

В заключение заметим, что размер массива `c` должен быть равен сумме размеров двух заданных массивов.

1.14. Обработка отдельных символов данной строки. Подсчет частоты появления символа в строке

1.14.1. Определить, сколько раз в заданной строке встречается некоторый символ

В школьном алгоритмическом языке величина строкового (в терминах языка Паскаль) типа рассматривается как массив, элементами которого являются отдельные символы. Это позволяет обратиться к тому или иному символу строки как к элементу массива — по имени строковой величины и его (символа) номеру[1].

Рассмотреть же все символы строки можно, подсчитав их количество, что, в свою очередь, можно сделать с помощью функции длин:

```
количество := 0
нц для номер от 1 до длин(строка)
  если строка[номер] = симв
    то
      количество := количество + 1
  все
кц
вывод нс, количество
```

где строка — величина строкового типа — заданная строка; симв — символ, число вхождений которого в строку ищется.

Язык Паскаль

```
kolichestvo := 0;
for nomer := 1 to Length(stroka) do
  if Copy(stroka, nomer, 1) = simvol
    then kolichestvo := kolichestvo + 1;
write(kolichestvo);
```

1.14.2. Определить позицию (номер) первого вхождения некоторого символа в заданную строку (если символа в строке нет, то вывести 0)

Попытка по аналогии с предыдущей задачей решить рассматриваемую задачу так:

```
искомый_номер := 0
нц для номер от 1 до длин(строка)
```

[1] Так же можно обратиться к отдельному символу строковой величины и в программе на языке Паскаль. В языке Бейсик для этого используется функция MID$.

```
  если строка[номер] = симв
    то
      искомый_номер := номер
  все
кц
|Вывод ответа
вывод нс, искомый_номер
```

даст неправильный результат — при наличии в строке нескольких вхождений заданного символа ответом будет номер последнего (а не первого!) вхождения.

Можно решить задачу, использовав величину логического типа впервые, определяющую, впервые ли в строке встретился заданный символ:

```
искомый_номер := 0
впервые := да
нц для номер от 1 до длин(строка)
  если строка[номер] = симв и впервые
    то
      искомый_номер := номер
      |Следующие вхождения символа будут уже не первыми
      впервые := нет
  все
кц
```

Можно также прекратить проверки, как только встретится заданный символ:

```
номер := 1
найден := нет
нц пока номер <= длин(строка) и не найден
  если строка[номер] = симв
    то
      |Встретился искомый символ
      найден := да
      искомый_номер := номер
    иначе |Переходим к следующему символу
      номер := номер + 1
  все
кц
|Вывод ответа
вывод нс, искомый_номер
```

где найден — величина логического типа, фиксирующая факт нахождения первого вхождения заданного символа в строку.

И, конечно, можно решить задачу, исследовав заданную строку с ее конца.

В программе на языке Паскаль решение задачи значительно упрощается за счет того, что в этом языке имеется функция Pos, возвращающая позицию (индекс) первого вхождения некоторой подстроки (в том числе и отдельного символа) в заданную строку или 0, если подстрока в строке отсутствует:

```
write(Pos(simv, stroka));
```

1.14.3. Определить, есть ли в заданной строке некоторый символ

Прежде всего, заметим, что такой вариант решения:

```
нц для номер от 1 до длин(строка)
  если строка[номер] = симв
    то
      вывод "Заданный символ в строке есть"
    иначе
      вывод "В строке нет заданного символа"
  все
кц
```

ошибочный — на экран ответ будет выводиться многократно.

Неправильный результат дает также такой способ, использующий величину логического типа `есть`, которая фиксирует факт наличия заданного символа в заданной строке:

```
нц для номер от 1 до длин(строка)
  если строка[номер] = симв
    то
      есть := да
    иначе
      есть := нет
  все
кц
если есть
  то
    вывод "Заданный символ в строке есть"
  иначе
    вывод "В строке нет заданного символа"
все
```

Его особенности проанализируйте самостоятельно.

Наиболее простой правильный способ решения задачи такой:

1. Подсчитать число вхождений заданного символа в заданную строку *(см. задачу 1.14.1)*.
2. По найденному значению получить ответ.

Можно также не рассматривать весь массив, а прекратить проверки, как только встретится заданный символ (см. решение *задачи 1.14.2* с использованием величины `найден`):

```
номер := 1
найден := нет
нц пока номер <= длин(строка) и не найден
  если строка[номер] = симв
    то
      найден := да
```

```
    иначе |Переходим к следующему символу
      nomer := номер + 1
  все
кц
если найден
  то
    вывод "Заданный символ в строке есть"
  иначе
    вывод "В строке нет заданного символа"
все
```

В программе на языке Паскаль задачи также решаются просто, благодаря использованию функции Pos:

```
if Pos(simv, stroka) > 0 then write('Заданный символ в строке есть')
else write('В строке нет заданного символа');
```

1.15. Работа с подстроками данной строки с разбиением на слова по пробельным символам. Поиск подстроки внутри данной строки, замена найденной подстроки на другую строку

1.15.1. Определить, сколько раз в заданной строке встречается некоторая подстрока

Решение задачи во многом аналогично решению *задачи 1.14.1*. Единственное отличие в том, что проверять надо не по одному символу, а по несколько. При анализу будем называть несколько последовательных символов строки термином "вырезка". При этом возникает вопрос: чему равен номер последнего символа строки, начиная с которого следует формировать вырезку символов?

Обозначим количество символов в заданной строке — длина1, в подстроке — длина2. Рассмотрим несколько возможных значений (табл. 1.4):

Таблица 1.4

длина1	*длина2*	**Номер последнего символа строки для формирования вырезки**
10	2	9
10	3	8
15	5	11
15	6	10

Из табл. 1.4 можно установить, что номер последнего символа строки для формирования вырезки определяется по формуле: длина1 – длина2 + 1.

Кроме того, теперь надо сравнивать с заданной подстрокой сформированные вырезки. В школьном алгоритмическом языке вырезку из строки можно получить, указав имя переменной, из которой формируется вырезка, и в квадратных скобках — начальный и конечный номера символов вырезки, например: `строка[1:5]`, `строка[10:12]`, `строка[k1:k2]` и т. п. В языке Бейсик для этого используется функция `MID$`, в языке Паскаль — функция `Copy`. В этих функциях — три параметра: величина строкового типа, начальный номер символа для вырезки и длина вырезки.

Обратим внимание на то, что, хотя мы используем термин "вырезка", из самой строки ничего не вырезается — ее значение не изменяется.

Итак, фрагмент программы решения задачи:

```
длина1 := длин(строка)
длина2 := длин(подстрока)
количество := 0
нц для номер от 1 до длина1 – длина2 + 1
  если строка[номер:(номер + длина2 – 1)] = подстрока
    то
        количество := количество + 1
  все
кц
```

где `подстрока` — подстрока, число вхождений которой в строку ищется.

Язык Паскаль

```
dlina1 := Length(stroka);
dlina2 := Length(podstroka);
kolichestvo := 0;
for nomer := 1 to dlina1 – dlina2 + 1 do
  if Copy(stroka, nomer, dlina2) = podstroka
    then kolichestvo := kolichestvo + 1;
```

1.15.2. Определить позицию (номер) первого вхождения некоторой подстроки в заданную строку (если подстроки в строке нет, то вывести 0)

Сразу же заметим, что, как уже отмечалось *(см. задачу 1.14.3)*, в языке Паскаль имеется стандартная функция `Pos`, используя которую можно решить данную задачу. Тем не менее считаем полезным решить ее и на Паскале, условно допустив, что такой функции нет.

Решение данной задачи также во многом аналогично решению *задачи 1.14.2*. Неправильные варианты мы рассматривать не будем, а приведем два правильных варианта.

1)
```
длина1 := длин(строка)
длина2 := длин(подстрока)
искомый_номер := 0
впервые := да
нц для номер от 1 до длина1 – длина2 + 1
  если строка[номер:(номер + длина2 – 1)] = подстрока и впервые
    то
      искомый_номер := номер
      |Следующие вхождения будут уже не первыми
      впервые := нет
  все
кц
вывод нс, искомый_номер
```

Язык Паскаль

```
dlina1 := Length(stroke);
dlina2 := Length(podstroka);
iskomiy_nomer := 0;
vpervie := true;
for nomer := 1 to dlina1 – dlina2 + 1 do
  if (Copy(stroka, nomer, dlina2) = podstroka) and vpervie
    then
      begin
        iskomiy_nomer := nomer;
        {Следующие вхождения будут уже не первыми}
        vpervie := false
      end;
{Вывод ответа}
write(iskomiy_nomer);
```

2)
```
длина1 := длин(строка)
длина2 := длин(подстрока)
искомый_номер := 0
номер := 1
найден := нет
нц пока номер <= длина1 – длина2 + 1 и не найден
  если строка[номер:(номер + длина2 – 1)] = подстрока
    то
      искомый_номер := номер
      найден := да
    иначе |Переходим к следующему символу
      номер := номер + 1
  все
кц
вывод нс, искомый_номер
```

Язык Паскаль

```
dlina1 := Length(stroka);
dlina2 := Length(podstroka);
iskomiy_nomer := 0;
naiden := false;
nomer := 1;
while (nomer <= dlina1 - dlina2 + 1) and not naiden do
  if Copy(stroka, nomer, dlina2) = podstroka then
    begin
      iskomiy_nomer := nomer;
      naiden := true
    end
  else {Переходим к следующему символу}
    nomer := nomer + 1;
{Вывод ответа}
write(iskomiy_nomer);
```

Конечно, и здесь можно решить задачу, исследовав заданную строку с ее конца, точнее — с (`dlina1 - dlina2 + 1`)-го символа.

1.15.3. Определить, есть ли в заданной строке некоторая подстрока

Наиболее простой правильный способ решения задачи такой:

1. Подсчитать число вхождений подстроки в заданную строку *(см. задачу 1.15.1)*.
2. По найденному значению получить ответ.

Можно также не рассматривать весь массив, а прекратить проверки, как только встретится заданная подстрока (см. решение *задачи 1.14.3* с использованием величины `найден`).

1.15.4. Удалить из заданной строки все вхождения некоторой подстроки

В программе на школьном алгоритмическом языке удалить из строки подстроку можно, присвоив вырезке с подстрокой *(см. разд. 1.15.1)* "пустое" значение, например для подстроки из четырех символов, начиная с 7-го:

```
строка[7:10] := ""
```

Еще две важные особенности программы решения задачи:

1. Поскольку в ходе обработки строки ее длина меняется, использовать оператора цикла с параметром, в конечном значении которого применяется величина `длина1` (количество символов в заданной строке), нельзя. Следует применить оператор цикла с условием следующего вида:

   ```
   номер <= длин(строка) – длина2 + 1
   ```

   где `длина2` — количество символов в удаляемой подстроке.

2. При рассмотрении начальных номеров подстрок после нахождения искомой подстроки и ее удаления величина номер не меняется (убедитесь в этом!), а в случае, когда искомая подстрока не встретилась, — увеличивается на 1.

С учетом сказанного фрагмент программы, решающей задачу, имеет вид:

```
номер := 1
нц пока номер <= длин(строка) – длина2 + 1
  если строка[номер:(номер + длина2 – 1)] = подстрока
    то |Встретилась искомая подстрока
      |Удаляем ее
      строка[номер:(номер + длина2 – 1)] := ""
      |Величина номер – не меняется
    иначе
      |Переходим к следующему символу для проверки
      номер := номер + 1
  все
кц
вывод нс, строка
```

Язык Паскаль

```
nomer := 1;
while nomer <= Length(stroka) - dlina2 + 1 do
  if Copy(stroka, nomer, dlina2) = podstroka then
    {Встретилась искомая подстрока.
    Удаляем ее}
    Delete(stroka, nomer, dlina2)
    {Величина nomer – не меняется}
    else
      {Переходим к следующему символу для проверки}
      nomer := nomer + 1;
write(stroka);
```

1.15.5. Заменить в заданной строке все вхождения некоторой подстроки на другую подстроку

Используем в программе величины:

- строка — заданная строка;
- подстрока1 — заменяемая подстрока;
- подстрока2 — новая подстрока;
- длина1 — количество символов в величине подстрока1;
- длина2 — то же, в величине подстрока2.

Приводя фрагмент программы, обращаем внимание на условие в операторе цикла (см. предыдущую задачу), а также на особенности изменения величины номер:

```
длина1 := длин(подстрока1)
длина2 := длин(подстрока2)
номер := 1
нц пока номер <= длин(строка) - длина1 + 1
  если строка[номер:(номер + длина1 — 1)] = подстрока1
    то |Встретилась искомая подстрока
      |Заменяем ее
      строка[номер:(номер + длина1 — 1)] := подстрока2
      |Изменяем значение величины номер
      номер := номер + длина2
    иначе
      |Переходим к следующему символу для проверки
      номер := номер + 1
  все
кц
вывод нс, строка
```

В языке Паскаль для замены одной подстроки на другую следует использовать процедуры `Delete` и `Insert`.

Язык Паскаль

```
dlina1 := Length(podstroka1);
dlina2 := Length(podstroka2);
nomer := 1;
while nomer <= Length(stroka) - dlina1 + 1 do
  if Copy(stroka, nomer, dlina1) = podstroka1 then
    {Встретилась искомая подстрока}
    begin
      {Удаляем ее}
      Delete(stroka, nomer, dlina1);
      {Вставляем новую подстроку}
      Insert(podstroka2, stroka, nomer);
      {Изменяем значение величины nomer}
       nomer := nomer + dlina2
    end
  else
    {Переходим к следующему символу для проверки}
    nomer := nomer + 1;
write(stroka);
```

В языке Бейсик для замены предназначена процедура `MID$`, однако (!) она работает, только когда число символов в обеих подстроках одинаковое.

1.15.6. Дана фраза, слова которой отделены друг от друга одним пробелом (начальных и конечных пробелов нет). Получить массив слов этой строки

Основные величины, используемые в программе:

- фраза — заданная фраза;
- слово — очередное слово фразы;
- число_слов — счетчик количества слов во фразе;
- массив_слов — массив для хранения слов (его размер должен быть описан с запасом);
- номер — номер очередного рассматриваемого символа фразы.

Прежде чем приводить фрагмент программы, заметим, что:

1) структуру заданной фразы можно представить в виде (рис. 1.4):

слово	пробел	слово	пробел	...	слово

Рис. 1.4. Структура фразы

2) признак окончания очередного слова — встретившийся пробел.

Итак, фрагмент программы:

```
число_слов := 0
номер := 1
слово := ""
нц пока номер <= длин(фраза)
  если фраза[номер] <> " "
  |Если символ - не пробел
    то
      |Добавляем его к величине слово
      слово := слово + фраза[номер]
    иначе|Пробел
      |Очередное слово кончилось
      |Его номер
      число_слов := число_слов + 1
      |Записываем его в массив
      массив_слов[число_слов] := слово
      |Готовимся формировать новое слово
      слово := ""
  все
|Переходим к следующему символу для проверки
номер := номер + 1
кц
```

```
|Записываем в массив последнее слово, которое было сформировано,
|но не записано (см. рис. 1.4)
число_слов := число_слов + 1
массив_слов[число_слов] := слово
```

Язык Паскаль

```
chislo_slov := 0;
nomer := 1;
slovo := '';
while nomer <= Length(fraza) do
  begin
    if fraza[nomer] <> ' ' {Если символ – не пробел}
      then
        {Добавляем его к величине slovo}
        slovo := slovo + fraza[nomer]
      else {Встретился пробел — очередное слово кончилось}
        begin
          {Его номер}
          chislo_slov := chislo_slov + 1;
          {Записываем его в массив}
          massiv_slov[chislo_slov] := slovo;
          {Готовимся формировать новое слово}
          slovo := ''
      end;
    {Переходим к следующему символу для проверки}
    nomer := nomer + 1
  end;
{Записываем в массив последнее слово, которое было сформировано, но не записано
(см. рис. 1.4)}
chislo_slov := chislo_slov + 1;
massiv_slov[chislo_slov] := slovo;
```

Сформированный массив слов может быть использован для решения различных задач, таких как: нахождение слова максимальной (минимальной) длины или его номера, подсчет количества тех или иных букв в каждом слове, поиск заданных слов и т. п.

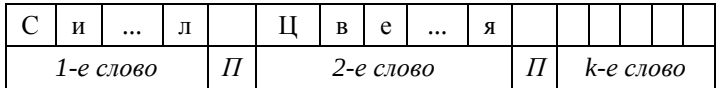
ГЛАВА 2

Другие типовые задачи программирования

Задачи, рассматриваемые в данной главе, наряду с задачами, методика решения которых обсуждена в *главе 1*, используются как вспомогательные при решении задач С4 ЕГЭ, а также как задачи С2.

2.1. Группа задач на выделение частей строки

В *задачах 2.1.1–2.1.4* обрабатывается задаваемая во время выполнения программы строка символов с именем `строка`, имеющая в общем случае следующую структуру:



где *П* — пробел.

2.1.1. Выделение первого слова

Соответствующий фрагмент имеет вид:

```
слово1 := ""     |Формируемое слово (начальное значение)
номер_симв := 1 |Номер очередного символа строки
нц пока строка[номер_симв] <> " " |Пока не встретится 1-й пробел
  |Добавляем текущий символ к "старому" значению величины слово1
  слово1 := слово1 + строка[номер_симв]
  |Переходим к следующему символу
  номер_симв := номер_симв + 1
кц
```

Аналогичный фрагмент на языке Паскаль:

```
slovo1 := ''; {Формируемое слово (начальное значение)}
nomer_simv := 1; {Номер очередного символа строки}
while stroka[nomer_simv] <> ' ' {Пока не встретится 1-й пробел}
  do begin
      {Добавляем текущий символ к "старому" значению величины слово1}
       slovo1 := slovo1 + stroka[nomer_simv];
```

```
      {Переходим к следующему символу}
       nomer_simv := nomer_simv + 1
    end;
```

Можно также применить оператор цикла с постусловием:

```
слово1 := ""
номер_симв := 1
нц
  слово1 := слово1 + строка[номер_симв]
  номер_симв := номер_симв + 1
кц_при строка[номер_симв] = " " |Встретился 1-й пробел
```

Язык Паскаль

```
slovo1 := '';
nomer_simv := 1;
repeat
  slovo1 := slovo1 + stroka[nomer_simv];
  nomer_simv := nomer_simv + 1
until stroka[nomer_simv] = ' '{Встретился 1-й пробел}
```

Заметим здесь же, что при оформлении фрагмента в виде:

```
слово1 := ""
номер_симв := 0 |Начальный номер равен нулю
нц
  номер_симв := номер_симв + 1
  слово1 := слово1 + строка[номер_симв]
кц_при строка[номер_симв] = " "
```

пробел будет включен в состав первого слова.

Если в языке программирования имеется функция, возвращающая позицию первого вхождения в строку той или иной подстроки (а в языке Паскаль, как указывалось в *главе 1*, такая функция есть), то для решения задачи можно не использовать оператор цикла:

```
slovo1 := Copy(stroka, 1, Pos(stroka, ' ') - 1);
```

Так же можно обойтись без использования оператора цикла, если длина первого слова известна (и равна `длина1/dlina1`):

- в программе на школьном алгоритмическом языке:

```
слово1 := строка[1:длина1]
```

- в программе на Паскале:

```
slovo1 := Copy(stroka, 1, dlina1);
```

Программа на языке Паскаль может быть также улучшена за счет того, что вводимые символы строки можно считывать сразу во время их ввода, используя оператор `read` с параметром символьного типа:

```
repeat
  {Считываем очередной введенный символ}
  read(c);
  {и добавляем его к "старому" значению величины slovo1}
  slovo1 := slovo1 + c
until c = ' ';
```

Это позволяет отказаться от использования переменной nomer_simv и от переменной stroka строкового типа. Однако в этом случае пробел будет включен в состав первого слова.

Обратим внимание на то, что в решениях на языке Паскаль, приведенных в [2–5, 7, 12, 16], используется именно этот способ — он более эффективен с точки зрения использования памяти по сравнению с предварительным запоминанием всей строки в целом.

2.1.2. Выделение второго слова

Для этого следует сначала пропустить первое слово:

```
номер_симв := 1 |Номер очередного символа строки
нц
  |Переходим к следующему символу
  номер_симв := номер_симв + 1
  |пока не встретится 1-й пробел
кц_при строка[номер_симв] = " "
|Величина номер_симв указывает на 1-й пробел
```

После этого можно сформировать второе слово:

```
слово2 := ""  |Формируемое слово (начальное значение)
|Переходим к 1-му символу 2-го слова
номер_симв := номер_симв + 1
нц пока строка[номер_симв] <> " "   |Пока не встретится 2-й пробел
  |Добавляем текущий символ к "старому" значению величины слово2
  слово2 := слово2 + строка[номер_симв]
  |Переходим к следующему символу
  номер_симв := номер_симв + 1
кц
```

В программе на языке Паскаль можно также первое слово пропустить, использовав функцию Pos (см. выше) — установить значение величины nomer_simv, равное номеру символа после первого пробела.

Кроме того, в программе на этом языке, как и применительно к предыдущей задаче, вводимые символы строки можно считывать сразу во время их ввода, используя оператор read с параметром символьного типа:

```
{Пропускаем первое слово}
repeat
 read(c);
until c = ' ';
```

```
{Формируем второе слово}
repeat
  {Считываем очередной введенный символ}
  read(c);
  {и добавляем его к "старому" значению величины slovo1}
  slovo1 := slovo1 + c
until c = ' ';
```

Обращаем внимание на тот факт, что и здесь второй пробел будет включен в состав второго слова.

2.1.3. Выделение двух первых слов как единой величины

В программах на Паскале, приведенных в [2–5, 7, 12, 16] для вариантов, в которых такая задача возникает, пробел между словами и пробел после второго слова включается в выделяемую величину. Поступая аналогично, можем использовать отмеченный при решении *задачи 2.1.1* вариант:

```
слово12 := ""
номер_симв := 0 |Начальный номер равен нулю
нц
  |Переходим к следующему символу
  номер_симв := номер_симв + 1
  |Переходим к следующему символу
  слово12 := слово12 + строка[номер_симв]
кц_при строка[номер_симв] = " "
|Первое слово и пробел после него
|включены в состав величины слово12
|Обрабатываем второе слово
нц
  номер_симв := номер_симв + 1
  слово12 := слово12 + строка[номер_симв]
кц_при строка[номер_симв] = " "
|Второй пробел также учтен
```

где `слово12` — формируемая величина.

Можно также несколько сократить фрагмент, использовав вложенный оператор цикла:

```
слово12 := ""
номер_симв := 0
нц для i от 1 до 2
  нц
    номер_симв := номер_симв + 1
    слово12 := слово12 + строка[номер_симв]
  кц_при строка[номер_симв] = " "
кц
```

Конечно, когда выделяемых слов три и более, применение вложенного оператора цикла дает больший эффект.

Язык Паскаль

```
{Формируем первое слово}
repeat
  {Считываем очередной введенный символ}
  read(c);
  {и добавляем его к "старому" значению величины slovo12}
  slovo12 := slovo12 + c
until c = ' ';
{Добавляем второе слово}
repeat
  read(c);
  slovo12 := slovo12 + c
until c = ' ';
{Второй пробел также учтен}
```

2.1.4. Выделение последнего слова

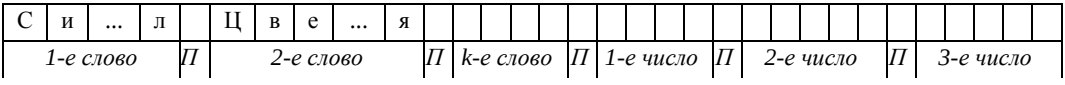
В программе на школьном алгоритмическом языке задача решается аналогично *задаче 2.1.2*:

1. Пропускаются все слова, кроме последнего.
2. Формируется последнее слово.

В программе на языке Паскаль после пропуска всех слов, кроме последнего, его можно считать, применив оператор `readln` с параметром, соответствующим искомому слову:

```
readln(slovo);
```

В *задачах 2.1.5–2.1.8* обрабатываемая строка символов с именем `строка` имеет в общем случае следующую структуру:



где *П* — пробел; *1-е число*, *2-е число* и *3-е число* — символы строки, представляющие собой запись чисел.

2.1.5. Выделение числа после первого слова

Будем считать, что искомым числом строка заканчивается.

Для решения задачи необходимо сначала пропустить первое слово и пробел после него:

```
|Пропускаем 1-е слово
номер_симв := 1 |Номер очередного символа строки
```

```
нц
  |Переходим к следующему символу
  номер_симв := номер_симв + 1
  |пока не встретится 1-й пробел
кц_при строка[номер_симв] = " "
|Величина номер_симв указывает на 1-й пробел
|Пропускаем его
номер_симв := номер_симв + 1
```

После этого надо получить последовательность символов-цифр и преобразовать ее в число:

```
|Получаем символы строки, представляющие собой запись числа,
|путем копирования соответствующей подстроки
|и преобразуем ее в число
число := лит_в_цел(строка[номер_симв : длин(строка)], успех)
```

где `длин` — функция, возвращающая общее количество символов в строке; `лит_в_цел` — функция, возвращающая числовое представление числа — ее аргумента, представленного символами-цифрами *(подробнее об этой функции см. в разд. 2.5.3)*.

Программа на языке Паскаль существенно упрощается, если после пропуска первого слова и пробела считать оставшиеся символы-цифры как единое число, применив оператор `readln`:

```
readln(chislo);
```

Считанное значение присваивается величине числового типа `chislo`, т. е. преобразование строки в число не требуется.

Если длина числа известна (и равна `длина1/dlina1`), то искомое значение можно получить без использования оператора цикла, учитывая, что "местоположение" числа в строке известно:

```
число := лит_в_цел(строка[длин(строка) - длина1 + 1 : длин(строка)], успех)
```

Соответствующий оператор в программе на языке Паскаль:

```
Val(Copy(stroka, Length(stroka) - dlina1 + 1, dlina1), chislo, code);
```

2.1.6. Выделение числа после второго слова

Будем считать, что искомым числом строка заканчивается.

Задача решается аналогично предыдущей с той разницей, что пропустить необходимо два первых слова и пробел после второго слова:

```
|Пропускаем 1-е слово
номер_симв := 1 |Номер очередного символа строки
нц
  |Переходим к следующему символу
  номер_симв := номер_симв + 1
  |пока не встретится 1-й пробел
кц_при строка[номер_симв] = " "
```

```
|Величина номер_симв указывает на 1-й пробел
|Пропускаем 2-е слово
нц
  |Переходим к следующему символу
  номер_симв := номер_симв + 1
  |пока не встретится 1-й пробел
кц_при строка[номер_симв] = " "
|Пропускаем 2-й пробел
номер_симв := номер_симв + 1
|Получаем символы строки, представляющие собой запись числа,
|путем копирования соответствующей подстроки
|и преобразуем ее в число
число := лит_в_цел(строка[номер_симв : длин(строка)], успех)
```

В программе на языке Паскаль можно также после пропуска слов и пробелов считать оставшиеся символы-цифры как единое число:

```
readln(chislo);
```

Если длина числа известна, то искомое числовое значение можно получить без использования оператора цикла *(см. предыдущую задачу и разд. 1.15.1)*.

2.1.7. Выделение двух чисел после второго слова

Будем считать, что количество цифр в каждом числе известно и равно соответственно `длина1(dlina1)` и `длина2(dlina2)`.

Соответствующий фрагмент:

```
|Пропускаем два первых слова
|и второй пробел
... (см. предыдущую задачу)
|Получаем первое число
число1 := лит_в_цел(строка[номер_симв : номер_симв + длина1 — 1], успех)
|Пропускаем первое число и пробел после него
номер_симв := номер_симв + длина1 + 1
|Получаем второе число
число2 := лит_в_цел(строка[номер_симв : номер_симв + длина2 — 1], успех)
```

В программе на языке Паскаль можно также после пропуска слов и пробелов считать оставшиеся символы-цифры как два отдельных числа:

```
readln(chislo1, chislo2);
```

Если длина чисел известна, то необходимые значения можно получить, зная их "местонахождение" в конце строки (см. выше).

2.1.8. Выделение трех чисел после второго слова

Такая задача, используемая, например, в [3], решается аналогично.

Задания для самостоятельной работы

1. Задана строка с информацией об ученике в формате:

`<Фамилия> <Номер школы>`

где `<Фамилия>` — значение, состоящее не более чем из 30 символов без пробелов; `<Номер школы>` — целое число в диапазоне от 1 до 99. Эти данные записаны через один пробел (т. е. всего в строке один пробел).

Получить фамилию ученика:

1) включая пробел после нее;

2) без указанного пробела.

Задачу 2) решить также для случая, когда длина величины `<Фамилия>` известна.

2. Задана строка с информацией о стоимости бензина на АЗС в формате:

`<Компания> <Улица> <Цена>`

где `<Компания>` — строка, состоящая не более чем из 20 символов без пробелов; `<Улица>` — строка, состоящая не более чем из 20 символов без пробелов; `<Цена>` — целое число в диапазоне от 1000 до 3000, обозначающее стоимость одного литра бензина в копейках. `<Компания>` и `<Улица>`, `<Улица>` и `<Цена>` разделены ровно одним пробелом (т. е. всего два пробела).

Получить название улицы, на которой находится данная АЗС.

Задачу решить двумя способами:

1) с использованием оператора цикла;

2) без его использования.

3. Задана строка с информацией об ученике в формате:

`<Фамилия> <Имя> <Оценка>`

где `<Фамилия>` — значение, состоящее не более чем из 30 символов без пробелов; `<Имя>` — строка, состоящая не более чем из 20 символов без пробелов; `<Оценка>` — целое число в диапазоне от 2 до 5. Эти данные записаны через один пробел, причем ровно один между каждой парой (т. е. всего два пробела в строке).

Получить фамилию и имя ученика как одну величину:

1) включая пробелы до и после имени;

2) без пробела после имени.

4. Задана строка с информацией о футбольной команде в формате:

`<Название> <Количество>`

где `<Название>` — значение, состоящее не более чем из 20 символов без пробелов; `<Количество>` — число раз, которое команда участвовала в чемпионате. Эти данные записаны через один пробел (т. е. всего в строке один пробел).

Определить, сколько раз данная команда участвовала в чемпионате.

5. Задана строка с информацией о городе в формате:

<Название города> <Год основания>

где <Название города> — значение, состоящее не более чем из 20 символов без пробелов; <Год основания> — четырехзначное число. Эти значения записаны через один пробел (т. е. всего в строке один пробел).

Определить год основания данного города. Задачу решить без использования оператора цикла.

6. Задана строка с информацией об ученике в формате:

<Фамилия> <Имя> <Номер школы>

где <Фамилия> — значение, состоящее не более чем из 30 символов без пробелов; <Имя> — строка, состоящая не более чем из 20 символов без пробелов; <Номер школы> — целое число в диапазоне от 1 до 1599. Эти данные записаны через один пробел, причем ровно один между каждой парой (т. е. всего три пробела в строке).

Определить номер школы.

7. Задана строка с информацией об ученике в формате:

<Фамилия> <Имя> <Год рождения>

где <Фамилия> — значение, состоящее не более чем из 20 символов без пробелов; <Имя> — строка, состоящая не более чем из 15 символов без пробелов; <Год рождения> — четырехзначное число. Эти значения записаны через один пробел (т. е. всего в строке два пробела).

Определить год рождения данного ученика. Задачу решить без использования оператора цикла.

8. Задана строка с информацией о марке и стоимости бензина на АЗС в формате:

<Компания> <Улица> <Марка> <Цена>

где <Компания> — строка, состоящая не более чем из 20 символов без пробелов; <Улица> — строка, состоящая не более чем из 20 символов без пробелов; <Марка> — одно из чисел 92, 95 или 98; <Цена> — целое число в диапазоне от 1000 до 3000, обозначающее стоимость одного литра бензина в копейках. <Компания> и <Улица>, <Улица> и <Марка>, а также <Марка> и <Цена> разделены ровно одним пробелом (т. е. всего в строке три пробела).

Получить марку и стоимость бензина.

9. Задана строка с информацией об оценках ученика по 5-балльной системе по трем предметам в формате:

<Фамилия> <Имя> <Оценка1> <Оценка2> <Оценка3>

где каждые значения отделены друг от друга одним пробелом (т. е. всего в строке четыре пробела).

Определить каждую из оценок.

Задачу решить двумя способами:

1) с использованием оператора цикла;

2) без его использования.

2.2. Группа задач на подсчет количества каждого из значений

Прежде чем обсуждать задачи данной группы, решим задачу подсчета количества чисел в заданной последовательности, равных некоторому числу. Решение последней задачи во многом аналогично решению такой же задачи применительно к массиву *(см. разд. 1.11.2)*.

```
|Ввод общего количества чисел последовательности n
ввод n
количество := 0
нц для i от 1 до n
  |Ввод очередного числа a последовательности
  ввод a
  |Если оно равно заданному числу X
  если a = X
    то
     |Учитываем его в искомом количестве
     количество := количество + 1
  все
кц
|Вывод ответа
вывод нс, "Количество чисел, равных X: ", количество
```

Язык Паскаль

```
{Ввод общего количества чисел последовательности n}
readln(n);
kolichestvo := 0;
for i := 1 to n do
  begin
    {Ввод очередного числа a последовательности}
    readln(n);
    {Если оно равно заданному числу X}
     if a = X then {Учитываем его в искомом количестве}
       kolichestvo := kolichestvo + 1; {или Inc(kolichestvo)}
{Вывод ответа}
write('Количество чисел, равных X: ', kolichestvo);
```

2.2.1. Подсчет количества каждой из цифр в заданной последовательности

Задача: "Даны *n* цифр. Подсчитать количество каждой из них. Массив для хранения цифр заданной последовательности не использовать".

Анализ решения

Прежде всего, учитывая требование, связанное с массивом цифр, каждую цифру последовательности следует "обрабатывать" сразу после ее ввода. Это же надо делать и с другими аналогичными данными в задачах, рассматриваемых ниже.

Далее, ясно, что нужно получить 10 значений. Но хранить искомое количество каждой из цифр с помощью десяти переменных величин нерационально — лучше использовать массив кол_цифр из десяти элементов с индексами от 0 до 9.

Обсудим, как учесть очередную цифру в этом массиве.

Проверка каждой из цифр с использованием оператора выбора (варианта) следующим образом:

```
выбор
  при цифра = 0:
    |Увеличиваем количество цифр 0
    кол_цифр[0] := кол_цифр[0] + 1
  при массив1[i] = 1:
    |Увеличиваем количество цифр 1
    кол_цифр[1] := кол_цифр[1] + 1
  ...
  при массив1[i] = 9:
    |Увеличиваем количество цифр 9
    кол_цифр[9] := кол_цифр[9] + 1
 все
кц
```

— также является нерациональной.

Подумаем, значение какого элемента массива кол_цифр должно быть увеличено на 1, если очередная цифра равна цифра? Ответ — элемента с индексом цифра. Это позволяет записать оператор для подсчета количества каждой из цифр в последовательности очень компактно:

```
ввод n
нц для i от 1 до n
  |Вводим очередную цифру
  ввод цифра
  |Увеличиваем ее "счетчик" на 1
  кол_цифр[цифра] := кол_цифр[цифра] + 1
кц
```

Язык Паскаль

```
readln(n);
{Рассматриваем все заданные цифры}
for i := 1 to n do
 begin
  {Вводим очередную цифру}
  readln(zifra);
```

```
  {Увеличиваем ее "счетчик" на 1}
  Inc(kol_zifr[zifra])
end;
```

Примечание

Как уже отмечалось в *главе 1*, в школьном алгоритмическом языке начальное присваивание переменным, в том числе и элементам массива, нулевых значений не происходит, поэтому следует предварительно обнулить все элементы массива кол_цифр.

2.2.2. Подсчет количества каждой из цифр в заданной строке. Вариант 1

Задача: "Дана строка, состоящая из цифр. Подсчитать количество каждой из цифр. Массив для хранения отдельных цифр заданной строки не использовать".

Анализ решения

Здесь, как и в предыдущей задаче, искомое количество каждой из цифр запишем в массив кол_цифр из 10 элементов с индексами от 0 до 9.

Идея решения задачи достаточно понятная:

1) выделить отдельную цифру строки в виде числа (пусть ее имя — цифра);

2) учесть ее в массиве кол_цифр.

Соответствующий фрагмент:

```
|Ввод строки
ввод строка
|Обнуляем элементы массива кол_цифр
нц для i от 0 до 9
  кол_цифр[i] := 0
кц
|Рассматриваем все символы заданной строки
нц для i от 1 до длин(строка)
  |Преобразуем i-й символ в число
  цифра := лит_в_цел(строка[i], успех)
  |Увеличиваем счетчик этой цифры на 1
  кол_цифр[цифра] := кол_цифр[цифра] + 1
кц
```

Язык Паскаль

```
{Ввод строки}
readln(stroka);
{Обнуляем элементы массива kol_zifr}
for i := 0 to 9 do kol_zifr[i] := 0;
{Рассматриваем все символы заданной строки}
for i := 1 to Length(stroka) do
```

```
  begin
    {Преобразуем символ-цифру stroka[i] в число}
    Val(stroka[i], zifra, code);
    {Учитываем его в массиве kol_zifr}
    Inc(kol_zifr[zifra])
  end;
```

Программа на языке Паскаль может быть упрощена за счет того, что значениями индексов элементов массива `kol_zifr` в этом языке могут быть данные символьного типа. Это позволит отказаться от преобразования символов в число:

```
var
kol_zifr: array ['0'..'9'] of byte;
...
BEGIN
{Ввод строки}
readln(stroka);
{Рассматриваем все символы заданной строки}
for i := 1 to Length(stroka) do
  {Для символа-цифры stroka[i] увеличиваем на 1
  его значение в массиве kol_zifr}
  Inc(kol_zifr[stroka[i]]);
```

Как отмечалось в *разд. 2.1.1*, в программе на языке Паскаль можно отдельно не вводить всю заданную строку и потом обрабатывать ее, а выделять символы-цифры сразу во время ввода исходных данных. Это можно сделать благодаря использованию оператора `read`, аргументом которого является величина символьного типа (`char`):

```
var
  dlina: integer; simv: char; ...
BEGIN
write('Задайте длину строки ');
read(dlina);
{Считываем по одному символу заданной строки}
i := 1;
writeln('Введите строку ');
repeat
  read(simv);
  {Для очередного символа
  увеличиваем на 1 его значение в массиве kol_zifr}
  Inc(kol_zifr[simv]);
  {Переходим к следующему символу}
  i := i + 1
until i > dlina;
```

Аналогично на языке Паскаль можно решать задачи, в которых известна не длина строки, а ее последний символ. В этом случае величина `dlina` не используется:

```
var
  simv: char; ...
BEGIN
writeln('Введите строку ');
{Считываем первый символ}
read(simv);
while simv <> posled do
  begin
    {Для очередного символа
    увеличиваем на 1 его значение в массиве kol_zifr}
    Inc(kol_zifr[simv]);
    {Считываем следующий символ}
    read(simv);
  end;
```

где `posled` — последний символ строки.

Заметим, что применение оператора цикла с постусловием:

```
writeln('Введите строку ');
repeat
  {Считываем очередной символ}
  read(simv);
  {Для очередного символа
  увеличиваем на 1 его значение в массиве kol_zifr}
  Inc(kol_zifr[simv]);
until simv = posled;
```

в данном случае невозможно, т. к. для последнего символа — "нецифры" нельзя выполнить процедуру `Inc`.

2.2.2. Подсчет количества каждой из цифр в заданной строке. Вариант 2

Задача: "Дана строка, в которой есть цифры. Подсчитать количество каждой из цифр. Массив для хранения отдельных цифр заданной строки не использовать".

Анализ решения

Здесь новым является то, что в заданной строке, кроме цифр, имеются и другие символы, и наша задача — выделить и обработать только символы-цифры. Выделение цифр можно провести так:

```
|Ввод строки
ввод строка
|Рассматриваем все символы заданной строки
нц для i от 1 до длин(строка)
  |Если i-й символ — цифра
  если строка[i] >= "0" и строка[i] <= "9"
    то
```

```
      |Преобразуем его в число
      цифра := лит_в_цел(строка[i], успех)
      |Учитываем цифру в массиве кол_цифр
      кол_цифр[цифра] := кол_цифр[цифра] + 1
  все
кц
```

Язык Паскаль

```
{Ввод строки}
readln(stroka);
for i := 1 to Length(stroka) do
  if (simv >= '0') and (simv <= '9') {Если i-й символ — цифра}
    then Inc(kol_zifr[stroka[i]]);
```

Задачи такого типа, в которых известна не длина строки, а ее последний символ, на языке Паскаль можно решать так же, как и предыдущую задачу:

```
writeln('Введите строку ');
{Считываем первый символ}
read(simv);
while simv <> posled do
  begin
    if (simv >= '0') and (simv <= '9')
      then Inc(kol_zifr[simv]);
    {Считываем следующий символ}
    read(simv);
  end;
```

где `posled` — последний символ строки.

Интересно, что в этой задаче применение оператора цикла с постусловием возможно:

```
writeln('Введите строку ');
repeat
  {Считываем очередной символ}
  read(simv);
  if (simv >= '0') and (simv <= '9')
    then Inc(kol_zifr[simv]);
until simv = posled;
```

2.2.3. Подсчет количества каждой из букв в заданной строке. Вариант 1

Задача: "Дана строка, состоящая из прописных букв латинского алфавита. Подсчитать количество каждой из букв алфавита. Массив для хранения отдельных букв заданной строки не использовать".

Анализ решения

В программе на Паскале задача решается во многом аналогично первому варианту *задачи 2.2.2*, если использовать массива `kol_bykv` с индексами от "`A`" до "`Z`":

```
var
  kol_bykv: array ['A'..'Z'] of byte;
...
BEGIN
{Ввод строки}
readln(stroka);
{Рассматриваем все буквы заданной строки}
for i := 1 to Length(stroka) do
  {Для буквы stroka[i] увеличиваем на 1 ее значение в массиве kol_bykv}
  Inc(kol_bykv[stroka[i]]);
```

В программе на школьном алгоритмическом языке заполняемый массив `кол_букв` должен иметь индексы, соответствующие кодам прописных букв латинского алфавита:

```
цел таб кол_букв[65:90]
```

а в программе должны использоваться коды букв:

```
|Ввод строки
ввод строка
|Обнуляем элементы массива кол_букв
нц для i от 65 до 90
  кол_букв[i] := 0
кц
|Рассматриваем все буквы заданной строки
нц для i от 1 до длин(строка)
  |Выделяем i-ю букву
  буква := строка[i]
  |Увеличиваем счетчик этой буквы на 1
  кол_букв[код(буква)] := кол_букв[код(буква)] + 1
кц
```

2.2.3. Подсчет количества каждой из букв в заданной строке. Вариант 2

Задача: "Дана строка, в которой есть прописные буквы латинского алфавита. Подсчитать количество каждой из букв алфавита. Массив для хранения отдельных букв не использовать".

Анализ решения

Здесь, как и в аналогичной *задаче 2.2.2*, особенность в том, что в заданной строке кроме прописных букв латинского алфавита имеются и другие символы, и наша задача — выделить и обработать только нужные буквы.

Это можно сделать так:

```
|Ввод строки
ввод строка
|Рассматриваем все символы заданной строки
нц для i от 1 до длин(строка)
  |Если i-й символ – прописная буква латинского алфавита
  если строка[i] >= "A" и строка[i] <= "Z"
    то
      ...
```

Соответствующий фрагмент на языке Паскаль:

```
{Ввод строки}
readln(stroka);
{Рассматриваем все символы заданной строки}
for i + 1 to Length(stroka) do
  {Если i-й символ – прописная буква латинского алфавита}
   if (stroka[i] >= 'A') and (stroka[i] <= 'Z') then
      ...
```

Примечание

Если в двух последних задачах предполагается, что в строке есть строчные буквы, а искомое количество каждой из букв необходимо найти без учета регистра, то в программах на Паскале можно использовать функцию `UpCase` или `LowerCase`.

2.2.4. Подсчет количества каждого из числовых значений в заданной последовательности чисел

Задача: "Даны *n* чисел со значениями от 0 до 50. Подсчитать количество каждой из них. Массив для хранения чисел заданной последовательности не использовать".

Задача решается аналогично *задаче 2.2.1* (используется массив `кол_чисел/kol_chisel` из 51 элемента).

2.2.5. Подсчет количества каждого из числовых значений в заданном наборе строк

Задача: "Даны *n* строк, в каждой из которых находится информация в формате:

`<Подстрока> <Число>`

где `<Подстрока>` — последовательность не более чем из 30 букв без пробелов; `<Число>` — целое число в диапазоне от 1 до 99. Эти данные записаны через пробел (т. е. всего один пробел в каждой строке). Подсчитать количество каждого из чисел".

Анализ решения

Общая схема решения задачи заполнения массива с количествами каждого из значений от 1 до 99:

```
1. Ввод значения n
2. Цикл для i от 1 до n
   2.1. Ввод i-й строки
   2.2. Выделение числа
   2.3. Учет этого числа в массиве "счетчиков"
конец цикла
```

Возможные методы выполнения этапа 2.2 рассмотрены в *разд. 2.1*. Учет числа в массиве "счетчиков" проводится так же, как и в *задачах 2.2.1–2.2.4* (естественно, с учетом диапазона значений).

Здесь же заметим, что в программе на Паскале использование массива с индексами строкового типа от "1" до "99" недопустимо.

Задания для самостоятельной работы

1. Известны оценки каждого из 25 учеников класса на контрольной работе по физике. Подсчитать количество двоек, троек, четверок и пятерок. Массив для хранения оценок и четыре переменные для хранения искомых значений не использовать.

2. Дана строка, состоящая из цифр 3, 4, 5, 6, 7. Подсчитать количество каждой из таких цифр. Массив для хранения отдельных цифр заданной строки и пять переменных для хранения искомых значений не использовать.

3. Дана строка, в которой есть числовые значения. Подсчитать количество каждой из цифр. Массив для хранения отдельных цифр заданной строки не использовать.

4. Дан текст, состоящий из не более чем 250 прописных букв латинского алфавита. Подсчитать количество каждой из букв алфавита. Массив для хранения отдельных букв заданной строки не использовать.

5. Дан текст, состоящий из не более чем 250 букв латинского алфавита. Подсчитать количество каждой из прописных букв алфавита. Массив для хранения отдельных букв заданной строки не использовать.

6. Даны *n* строк, в каждой из которых находится информация в формате:

```
<Фамилия> <Номер школы>
```

где `<Фамилия>` — строка, состоящая не более чем из 30 символов без пробелов; `<Номер школы>` — целое число в диапазоне от 1 до 99. Эти данные записаны через пробел (т. е. всего один пробел в каждой строке).

Подсчитать количество учеников в каждой из школ. Массив для хранения номеров школ не использовать.

2.3. Группа задач на подсчет количества и вывод значений, удовлетворяющих некоторому условию

2.3.1. Подсчет количества тех чисел последовательности, которые удовлетворяют некоторому условию

Задача в общем виде: "Даны *n* чисел. Определить количество чисел, удовлетворяющих некоторому условию".

Анализ решения

В приведенном далее фрагменте решения задачи <условие> — заданное условие для подсчета чисел (это может быть равенство, неравенство, сложное условие и т. п.), количество — искомое количество:

```
|Ввод общего количества чисел последовательности n
ввод n
количество := 0
нц для i от 1 до n
  |Ввод очередного числа a
  ввод a...
  |Если заданное условие соблюдается
  если <условие>
    то
      |Учитываем число a в искомом количестве
      количество := количество + 1
  все
кц
вывод нс, "Количество чисел, удовлетворяющих условию, равно ", количество
```

Заметим, что <условие> зависит от:

1) значения числа a;

2) значения числа a и от порядкового номера i.

Если оно зависит только от порядкового номера i, то задача может быть решена без использования оператора цикла (убедитесь в этом!).

В случае если обрабатываемое число является частью строки, необходимо предварительно нужное значение из строки (см. типовые задачи группы 2.1).

Применительно к массиву чисел задача решалась в *главе 1 (см. разд. 1.11.2)*.

Язык Паскаль

```
{Ввод общего количества чисел последовательности n}
readln(n);
kolichestvo := 0;
for i := 1 to n do
 begin
```

```
  {Ввод очередного числа a}
  readln(a);
  {Если заданное условие соблюдается}
  if <условие>
    then
      {Учитываем число a в искомом количестве}
      kolichestvo := kolichestvo + 1 {или inc(kolichestvo)}
end;
write('Количество чисел, удовлетворяющих условию, равно ', kolichestvo);
```

2.3.2. Вывод на экран элементов массива, соответствующих элементам другого массива с заданными свойствами

Задача в общем виде: "Даны два массива. Вывести на экран те элементы первого массива, которые соответствуют элементам второго массива с заданными свойствами".

Пример задачи: "Даны два массива с информацией о 25 людях:

1) с их фамилиями;

2) со значениями их роста в сантиметрах.

Вывести на экран фамилии людей, рост которых превышает 170 см. Известно, что такие люди среди представленных имеются".

Решение в общем виде

Соответствующий фрагмент:

```
нц для i от 1 до n[1]
  |Если очередной элемент второго массива обладает заданными свойствами
  если <условие, связанное со вторым массивом>
    то
      |Выводим соответствующий элемент 1-го массива
      вывод массив1[i], " "
  все
кц
```

Решение для приведенной задачи

```
нц для i от 1 до n
   если рост[i] > 170
    то
     вывод фамилия[i], " "
  все
кц
```

где `рост` и `фамилия` — заданные массивы.

[1] Здесь и далее принимается, что массив описан с индексами от 1 до *n*.

Язык Паскаль

```
for i := 1 to n do
  {Если очередной элемент второго массива обладает заданными свойствами}
  if <условие, связанное со вторым массивом>
    then {Выводим соответствующий элемент первого массива}
      write(massiv1[i], ' ');
```

Примечание

При решении задачи такого типа с использованием языка Паскаль можно вместо двух массивов использовать один массив с данными типа `запись`.

В заключение заметим, что методика решения задач на подсчет количества максимальных/минимальных значений рассмотрена в *разд. 2.4*.

Задания для самостоятельной работы

1. Известны оценки 22 учеников класса по алгебре. Определить количество четверок.

2. Дан массив целых чисел из 12 элементов. Определить количество чисел, оканчивающих нулем (известно, что такие числа в массиве есть).

3. Дан массив натуральных чисел из 10 элементов. Вывести на экран те из них, которые кратны трем, а на следующей строке — количество таких чисел. Известно, что числа, о которых идет речь, в массиве есть.

4. В массиве записана информация о росте каждого из 20 человек (в см). Вывести значения роста, бóльшие 190 см. Если таких значений в массиве нет, вывести сообщение "Нет людей с таким ростом".

5. Дан массив из 15 элементов с вещественными значениями. Вывести отрицательные элементы. Если такой элемент единственный, то на следующей строке вывести его индекс. Известно, что в массиве есть как минимум один отрицательный элемент.

6. В массиве записано количество участников олимпиады из каждой из школ с номерами от 10 до 20. Вывести номера школ, число участников из которых больше двух, а затем на отдельной строке — количество таких школ.

7. В массиве записана информация о количество баллов, набранных каждым из 20 учащихся на ЕГЭ по информатике. Вывести условные номера учащихся, набравших более 90 баллов (условный номер соответствует порядковому номеру баллов учащихся в массиве). Если таких учащихся нет, вывести сообщение "Нет таких учащихся".

8. Дан массив с общим количеством учащихся в каждой из параллелей школы (от 1-й до 11-й). Вывести номера параллелей, общее число учащихся в которых превышает 60. Если такая параллель — единственная, то на следующей строке вывести число учащихся в ней. Известно, что в массиве есть как минимум одна параллель с указанным числом учащихся.

9. В массиве записаны значения роста 15 человек в формате `???.??`, где `?` — цифра. Определить число людей, рост которых равен среднему значению роста.

10. Даны два массива с информацией о 30 участниках олимпиады по информатике:

1) с их фамилиями;

2) с суммой набранных ими баллов.

Вывести на экран фамилии участников, набравших не более 30 баллов. Известно, что такие учащиеся имеются.

2.4. Группа задач на нахождение максимальных (минимальных) элементов массива, их индексов номеров, количеств и т. п.

Хотя методика решения всех задач этой группы описана применительно к массивам, подобные задачи для последовательности чисел в большинстве случаев решаются аналогичными методами (вместо *i*-го элемента массива нужно рассматривать очередное число последовательности).

Напомним, что задачи определения максимального/минимального элемента массива, а также их индексов обсуждены в *главе 1*.

2.4.1. Нахождения второго по величине максимального элемента

Данная задача допускает два толкования. Если рассматривать, например, массив 5 10 22 6 22 20 6 12, то каким должен быть ответ?

Под "вторым по величине максимальным элементом", или, короче, "вторым максимумом", можно понимать:

1) значение элемента массива, который стоял бы на предпоследнем месте, если бы массив был отсортирован по неубыванию. При таком толковании — 22;

2) значение элемента массива, *больше* которого только максимальный. В этом случае ответ — 20.

Если в массиве только один максимальный элемент (все остальные меньше), то оба толкования совпадают, и искомые значения будут одними и теми же, в противном случае — нет.

Обсудим оба варианта задачи. В каждом из них будем использовать две переменные:

1) `максимум1` — максимальный элемент массива (первый максимум);

2) `максимум2` — второй максимум (искомое значение).

2.4.1.1. Поиск элемента массива, который стоял бы на предпоследнем месте, если бы массив был отсортирован по неубыванию

Сначала рассмотрим вариант, в котором диапазон значений элементов массива известен.

В качестве начальных значений величин максимум1 и максимум2 (см. выше) принимаем число, которое заведомо меньше нижней границы диапазона значений элементов массива (например, при диапазоне от –1000 до 1000 — число –1001):

```
максимум1 := -1001
максимум2 := -1001
```

Строго говоря, значение максимум2 можно не задавать, что будет показано далее.

Затем рассматриваем все элементы, сравнивая их сначала со значением максимум1, а затем (при необходимости) — и со значением максимум2:

```
нц для i от 1 до n
  если a[i] > максимум1
    |Встретился элемент, больший максимум1
    то
      |Бывший первый максимум станет вторым
      максимум2 := максимум1
      |Первым максимумом станет встреченный элемент
      максимум1 := a[i]
      |Внимание — именно в таком порядке!
    иначе
      |Очередной элемент не больше максимум1
      |Сравниваем его со значением максимум2
      если a[i] > максимум2
        |Встретился элемент, больший максимум2
        то
          |Принимаем его в качестве нового значения максимум2
          максимум2 := a[i]
          |Значение максимум1 не меняется
      все
  все
кц
```

Нетрудно убедиться, что уже после рассмотрения первого элемента произойдет "переприсваивание"

```
максимум2 := максимум1
```

т. е. начальное значение величины максимум2, равное –1001, можно не задавать.

Язык Паскаль

```
maximum1 := -1001;
for i := 1 to n do
  if a[i] > maximum1
    {Встретился элемент, больший maximum1}
     then begin
            {Бывший первый максимум станет вторым}
            maximum2 := maximum1;
            {Первым максимумом станет встреченный элемент}
            maximum1 := a[i];
```

```
            {Внимание — именно в таком порядке!}
          end
    else
      {Очередной элемент не больше maximum1
      Сравниваем его со значением maximum2}
      if a[i] > maximum2
        then          {Встретился элемент, больший maximum2}
          {Принимаем его в качестве нового значения maximum2}
          maximum2 := a[i];
          {Значение максимум1 не меняется}
```

Если же диапазон значений элементов массива неизвестен, то предварительно нужно определить начальные значения величин максимум1 и максимум2, сравнив первый и второй элементы массива:

```
если a[1] > a[2]
  то
    максимум1 := a[1]
    максимум2 := a[2]
  иначе
    максимум1 := a[2]
    максимум2 := a[1]
все
```

или короче:

```
максимум1 := a[1]
максимум2 := a[2]
если a[2] > a[1]
  то
    максимум1 := a[2]
    максимум2 := a[1]
все
```

Затем рассматриваем остальные элементы, как и ранее, сравнивая их сначала со значением максимум1, а затем (при необходимости) — и со значением максимум2:

```
нц для i от 3 до n
  если a[i] > максимум1
    ...
```

Язык Паскаль

```
maximum1 := a[1];
maximum2 := a[2];
if a[2] > a[1] then
  begin
    maximum1 := a[2];
    maximum2 := a[1]
   end;
for i := 3 to n do ...
```

2.4.1.2. Нахождения элемента массива, больше которого только максимальный

Примем, что диапазон значений элементов массива известен.

В качестве начального значения величины максимум1[1] принимаем число, которое заведомо меньше нижней границы диапазона значений элементов массива (например, при диапазоне от –1000 до 1000 — число –1001):

```
максимум1 := -1001
```

Рассматриваем все элементы массива и проверяем каждое из них сначала на первый, а затем на второй максимум:

```
нц для i от 1 до n
  если a[i] > максимум1
    |Встретился элемент, больший максимум1
    то
      |Бывший первый максимум станет вторым
      максимум2 := максимум1
      |Первым максимумом станет встреченный элемент
      максимум1 := a[i]
    иначе
      |Очередной элемент не больше максимум1
      |В этом случае сравниваем его со значением максимум2
      если a[i] < максимум1 и a[i] > максимум2
        |Только в этом случае!
        то
          |Встретился элемент, меньший максимум1 и больший максимум2
          |Принимаем его в качестве нового значения максимум2
          максимум2 := a[i]
          |Значение максимум1 не меняется
      все
  все
кц
```

Фрагмент программы, относящийся к выводу ответа, оформляется так:

```
если максимум2 = -1001
  |Второй максимум не встретился
  то
   вывод нс, "Нет такого значения в массиве "
  иначе
   вывод нс, "Второй максимум равен ", максимум2
все
```

Язык Паскаль

```
maximum1 := -1001;
for i := 1 to n do
```

[1] Для величины максимум2 начальное значение, как и для первого варианта, можно не задавать.

```
  if a[i] > maximum1
    {Встретился элемент, больший maximum1}
     then begin
           {Бывший первый максимум станет вторым}
           maximum2 := maximum1;
           {Первым максимумом станет встреченный элемент}
           maximum1 := a[i]
          end
    else
      {Очередной элемент не больше maximum1.
      Сравниваем его со значением maximum2}
      if (a[i] < maximum1) and (a[i] > maximum2)
        {Только в этом случае!}
        then {Встретился элемент, меньший maximum1 и больший maximum2.
          Принимаем его в качестве нового значения maximum2}
          maximum2 := a[i];
          {Значение maximum1 не меняется}
writeln;
{Вывод ответа}
if maximum2 = -1001 then
  {Второй максимум не встретился}
  writeln('Нет такого значения в массиве')
else
   writeln('Второй максимум равен ', maximum2)
```

Ясно, что второго максимума в принятом толковании может не быть только тогда, когда все элементы массива равны.

2.4.2. Нахождение второго минимума

Все варианты этой задачи решаются аналогично предыдущей (конечно, с необходимыми изменениями).

2.4.3. Нахождение количества максимальных элементов

Как отмечалось в *главе 1*, задача может быть решена двумя способами:

1) за два прохода по массиву;

2) за один проход по массиву.

Первый способ описан в *разд. 1.9.5*.

Во втором случае идея решения такая: проходя по массиву, кроме значения максимума контролировать также количество элементов, равных максимальному (кол_макс). Если очередной элемент оказывается больше текущего максимума — он принимается в качестве максимального значения, а величина кол_макс — равной 1. Если же очередной элемент не больше максимального, то сравниваем его с максимумом. Если они равны, то встретился еще один максимум, и значение кол_макс увеличиваем на 1.

Соответствующий фрагмент на школьном алгоритмическом языке:

```
|Начальное присваивание значений искомым величинам
максимальное := a[1]
кол_макс := 1
|Рассматриваем остальные элементы
нц для i от 2 до n
  если a[i] > максимальное
    то      |Встретился новый максимум
      |Принимаем его в качестве значения максимальное
      максимальное := a[i]
      |Пока он — единственный
      кол_макс := 1
    иначе
      |Проверяем, не равно ли очередное значение "старому" максимуму
      если a[i] = максимальное
        то  |Встретился еще один максимум
            |Учитываем это
            кол_макс := кол_макс + 1
      все
  все
кц
вывод нс, кол_макс
```

Язык Паскаль

```
{Начальное присваивание значений искомым величинам}
maximal := a[1];
kol_max := 1;
{Рассматриваем остальные элементы}
for i := 2 to n do
  if a[i] > maximal then
    begin
      {Встретился новый максимум.
      Принимаем его в качестве значения maximal}
      maximal := a[i];
      {Пока он — единственный}
      kol_max := 1
    end
  else
    {Проверяем, не равно ли очередное значение "старому" максимуму}
     if a[i] = maximal then
        {Встретился еще один максимум.
        Учитываем это}
        kol_max := kol_max + 1;
writeln;
writeln(kol_max);
```

Если диапазон возможных значений элементов массива известен, то и здесь можно отдельно не рассматривать первый элемент:

```
|Начальное присваивание значений искомым величинам
максимальное := минимальное - 1
кол_макс := 0
|Рассматриваем все элементы
нц для i от 1 до n
  если a[i] > максимальное
      ...
```

где минимальное — нижняя граница диапазона возможных значений.

Язык Паскаль

```
{Начальное присваивание значений искомым величинам}
maximal := minimal - 1;
kol_max := 0;
{Рассматриваем все элементы}
for i := 1 to n do
  if a[i] > maximal then ...
```

В задачах, в которых количество максимальных элементов выводится на экран после вывода их индексов, целесообразно подсчет этого количества проводить во время второго прохода (при отборе нужных элементов и выводе их индексов):

```
{1-й проход — определение максимального значения (см. задачу 1.9.1)}
...
{2-й проход — отбор элементов с максимальным значением и вывод их индексов}
кол_макс := 0
|Рассматриваем все элементы
нц для i от 1 до n
  если a[i] = максимальное
    то
    |Выводим  индекс
    вывод i, " "
    |Изменяем значение кол_макс
    кол_макс := кол_макс + 1
  все
|Выводим значение кол_макс
вывод нс, кол_макс
```

Язык Паскаль

```
{1-й проход — определение максимального значения (см. задачу 1.9.1)}
...
{2-й проход — отбор элементов с максимальным значением и вывод их индексов}
kol_max := 0;
```

```
for i := 1 to n do
  if a[i] = maximal then
    begin
      {Выводим индекс}
       writeln(i, ' ');
      {Изменяем значение kol_max}
      kol_max := kol_max + 1
    end;
writeln;
{Выводим значение kol_max}
writeln(kol_max);
```

2.4.4. Нахождение количества минимальных элементов

Задача решается аналогично предыдущей (конечно, с необходимыми изменениями).

2.4.5. Нахождение количества вторых максимумов

Начнем решать задачу для варианта с использованием второго толкования понятия "второй максимум" *(см. задачу 2.4.1)*.

2.4.5.1. Нахождение количества значений в массиве, равных элементу, больше которого только максимальный

Примем, что диапазон значений элементов массива известен.

В качестве начального значения величины `максимум1` (см. выше) принимаем число, которое заведомо меньше нижней границы диапазона значений элементов массива (например, при диапазоне от –1000 до 1000 — число –1001):

```
максимум1 := -1001
```

Начальное значение `максимум2`, как и при решении *задачи 2.4.1.2*, можно не задавать.

Обсудим вопрос о начальных значениях искомых величин `кол_максимум1` и `кол_максимум2` (их смысл ясен из имен).

Если вспомнить *(см. задачу 2.4.1.2)*, как происходит "переприсваивание" значений, то можно сказать, что уже после рассмотрения первого элемента произойдет "переприсваивание" не только значения, но и количества:

```
максимум2 := максимум1
кол_максимум2 := кол_максимум1
```

т. е. начальное значение `кол_максимум2` также можно не задавать.

Значение же `кол_максимум1` также можно задать любым "неположительным" (например, 0). После рассмотрения первого элемента оно "перейдет" ко второму максимуму, а после встречи первого элемента, меньшего максимального, значение `кол_максимум2` станет равным 1 (убедитесь в этом, рассмотрев возможные варианты

согласно приведенному далее фрагменту программы). Если же все элементы массива равны, то это значение кол_максимум2 останется неизменным.

Итак, после задания начальных значений максимум1 и кол_максимум1 рассматриваем все элементы массива, сравнивая их сначала со значением максимум1, а затем (при необходимости) — и со значением максимум2:

```
нц для i от 1 до n
  если a[i] > максимум1   |Встретился элемент, больший максимум1
    то
     |Бывший первый максимум станет вторым
     максимум2 := максимум1
     |а значение кол_максимум2 станет равно
     |"старому" значению величины кол_максимум1
     кол_максимум2 := кол_максимум1
     |Первым максимумом станет встреченный элемент
     максимум1 := a[i]
     |Внимание — именно в таком порядке!
     |Новый первый максимум встретился впервые
     кол_максимум1 := 1
    иначе
      |Проверяем, не равно ли очередное значение
      |"старому" первому максимуму
      если a[i] = максимум1
        то  |Встретился еще один первый максимум
          |Учитываем это
          кол_максимум1 := кол_максимум1 + 1
        иначе
          |Сравниваем очередное значение со вторым максимумом
          если a[i] > максимум2
            то |Встретился элемент, больший максимум2
               |Вторым максимумом станет встреченный элемент
               максимум2 := a[i]
              |Пока он - единственный
              кол_максимум2 := 1
              |Значения максимум1 и кол_максимум1 не меняются
            иначе
              если a[i] = максимум2
                то |Встретился еще один второй максимум
                  кол_максимум2 := кол_максимум2 + 1
              все
          все
      все
  все
кц
```

Язык Паскаль

```
maximum1 := -1001;
kol_maximum1 := 0;
for i := 1 to n do
  if a[i] > maximum1 {Встретился элемент, больший maximum1} then
    begin
      {Бывший первый максимум станет вторым}
      maximum2 := maximum1;
      {а значение kol_maximum2 станет равно
      "старому" значению величины kol_maximum1}
        kol_maximum2 := kol_maximum1;
        {Первым максимумом станет встреченный элемент}
        maximum1 := a[i]; {Внимание — именно в таком порядке!
        Новый первый максимум встретился впервые}
        kol_maximum1 := 1
    end
  else
    {Проверяем, не равно ли очередное значение
    "старому" первому максимуму}
    if a[i] = maximum1 then
      {Встретился еще один первый максимум — учитываем это}
      kol_maximum1 := kol_maximum1 + 1
    else
      {Сравниваем очередное значение со вторым максимумом}
      if a[i] > maximum2 then
        begin
          {Встретился элемент, больший maximum2.
          Вторым максимумом станет встреченный элемент}
          maximum2 := a[i];
          {Пока он - единственный}
          kol_maximum2 := 1
          {Значения maximum1 и kol_maximum1 не меняются}
        end
      else
         if a[i] = maximum2 then
            {Встретился еще один второй максимум}
            kol_maximum2 := kol_maximum2 + 1;
```

2.4.5.2. Нахождение количества значений в массиве, равных элементу, который стоял бы на предпоследнем месте, если бы массив был отсортирован по неубыванию

При таком толковании понятия "второй максимум" решение задачи, основанное на сравнении каждого очередного элемента со "старыми" максимумами, — будет громоздким, учитывая, что вторым максимумом может быть как элемент, равный максимальному, так и меньший его. Кроме того, неясно, как считать количество первых максимумов — с учетом вторых максимумов, равных первому, или нет.

Понятнее всего следующая идея решения задачи (а программа, ее реализующая — достаточно простая). Можно определить количество вторых максимумов при толковании этого понятия, принятого в предыдущей задаче. Зная это значение кол_максимум2 (и количество первых максимумов кол_максимум1, которое также определяется в ходе расчетов), ответ можно получить следующим образом:

```
если кол_максимум1 = 1
  то
    |Все, что было определено для второго максимума,
    |является искомым результатом
    вывод нс, максимум2
    вывод нс, кол_максимум2
  иначе
    |Второй максимум равен первому
    вывод нс, максимум1
    вывод нс, кол_максимум1 - 1     |Один первый максимум не учитываем
```

Эффектно, не правда ли?

2.4.6. Нахождение количества вторых минимумов

Все варианты этой задачи решаются аналогично предыдущей (конечно, с необходимыми изменениями).

2.4.7. Нахождение третьего максимума

"Третьим максимумом" будем называть элемент, который стоял бы на третьем справа месте, если бы весь массив был отсортирован по неубыванию.

Задача решается аналогично *задаче 2.4.1* (в первом толковании термина — *см. разд. 2.4.1.1*):

```
максимум1 := ...
максимум2 := ...
|Значение максимум3 можно не задавать
нц для i от 1 до n
  если a[i] > максимум1
    |Встретился элемент, больший максимум1
    то
      |Бывший второй максимум станет третьим
      максимум3 := максимум2
      |Бывший первый максимум станет вторым
      максимум2 := максимум1
      |Первым максимумом станет встреченный элемент
      максимум1 := a[i]
      |Внимание — именно в таком порядке!
    иначе
      |Очередной элемент не больше максимум1
      |Сравниваем его со значением максимум2
```

```
      если a[i] > максимум2
        |Встретился элемент, больший максимум2
        то
          |Бывший второй максимум станет третьим,
          максимум3 := максимум2
          |а встреченный элемент принимаем в качестве
          |нового значения максимум2
          максимум2 := a[i]
          |Значение максимум1 не меняется
        иначе
          |Очередной элемент не больше максимум2
          |Сравниваем его со значением максимум3
          если a[i] > максимум3
            то
              |Меняем только значение максимум3
              максимум3 := a[i]
          все
      все
  все
кц
```

Язык Паскаль

```
maximum1 := ...;
maximum2 := ...;
{Значение maximum3 можно не задавать}
for i := 1 to n do
  if a[i] > maximum1 then
    {Встретился элемент, больший maximum1}
    begin
      {Бывший второй максимум станет третьим}
      maximum3 := maximum2;
      {Бывший первый максимум станет вторым}
      maximum2 := maximum1;
      {Первым максимумом станет встреченный элемент}
      maximum1 := a[i]
      {Внимание — именно в таком порядке!}
    end
  else {Очередной элемент не больше maximum1
       Сравниваем его со значением maximum2}
    if a[i] > maximum2 {Встретился элемент, больший maximum2}
      then
        begin
          {Бывший второй максимум станет третьим}
          maximum3 := maximum2;
          {а встреченный элемент принимаем в качестве
          нового значения maximum2}
          maximum2 := a[i]
```

```
      {Значение maximum1 не меняется}
    end
  else
    {Очередной элемент не больше maximum2.
    Сравниваем его со значением максимум3}
    if a[i] > maximum3 then
      {Меняем только значение maximum3}
      maximum3 := a[i];
```

2.4.8. Нахождение третьего минимума

Задача решается аналогично предыдущей (конечно, с необходимыми изменениями).

Задания для самостоятельной работы

1. В массиве хранится информация о стоимости 1 килограмма 20 видов конфет. Определить, сколько стоят самые дешевые конфеты.

2. Известны расстояния от Москвы до нескольких городов. Найти расстояние от Москвы до самого удаленного от нее города из представленных в списке городов.

3. Известны результаты каждого из участников соревнований по лыжным гонкам (время, затраченное на прохождение дистанции гонки). Спортсмены стартовали по одному. Результаты даны в том порядке, в каком спортсмены стартовали. Определить, каким по порядку стартовал лыжник, показавший лучший результат? Если таких спортсменов несколько, то должен быть найден первый из них.

4. В массиве хранится информация о количестве осадков, выпавших за каждый день июля. Определить дату самого дождливого дня. Если таких дней было несколько, то должна быть найдена дата:

1) первого из них;

2) последнего из них.

5. Известна информация о максимальной скорости каждой из 12 марок легковых автомобилей. Определить скорость автомобиля, больше которой только максимальное значение в массиве.

6. В массиве хранится информация о результатах 22 спортсменов, участвовавших в соревнованиях по бегу на 100 м. Определить результаты спортсменов, занявших первое и второе места. Задачу решить, не используя два прохода по массиву.

7. В одном массиве записаны названия 20 команд — участниц чемпионата по футболу, в другом — соответствующее им количество набранных очков. Определить команды, занявшие первое и второе место. Задачу решить, не используя два прохода по массиву.

8. Известна информация о среднедневной температуре за каждый день июля. Определить даты двух самых холодных дней.

9. Известна информация о баллах, набранных каждым из 25 учеников на ЕГЭ по информатике. Определить, сколько учеников набрали максимальную сумму баллов.

10. В массиве записана информация о весе в килограммах каждого из 30 человек. Вывести на экран:

1) в первой строке — порядковые номера людей, которые имеют максимальный вес среди представленных;
2) во второй строке — их количество.

2.5. Разные задачи

2.5.1. Суммирование значений для различных категорий

Анализ решения

Задача этого типа в общем виде формулируется так: "Дана последовательность чисел, каждое из которых относится к некоторой категории (номеру школы, номеру параллели класса и т. п.). Необходимо определить сумму всех чисел для каждой категории".

Можно также сформулировать задачу по-другому: "Даны *N* пар чисел, второе из которых в каждой паре определяет некоторую категорию (номер школы, номер параллели класса и т. п.). Необходимо для каждой категории определить сумму чисел, задаваемых первыми в паре".

Как и при решении задач группы 2.2, целесообразно использовать массив с индексами, соответствующими значениями категорий (возможно, и неиспользуемыми). В этот массив и будем записывать искомые значения. Если его имя — `всего`, то фрагмент, в котором решается задача, имеет вид:

```
|Обнуляем элементы массива всего
нц для i от ... до ... |Используются границы диапазона категорий
  всего[i] := 0
кц
|Вводим и учитываем числа
нц для i от 1 до N
  ввод число, категория
  |Учитываем значение число в массиве всего
  |для соответствующей категории
  всего[категория] := всего[категория] + число
кц
```

Язык Паскаль

```
{Обнуляем элементы массива всего}
for i := ... to ... {Используются границы диапазона категорий}
  do vsego[i] := 0;
for i := 1 to N do
  begin
    readln(chislo, kategoria);
    vsego[kategoria] := vsego[kategoria] + chislo
   end;
```

Если суммируемые числа являются частью строки, то их необходимо предварительно выделить *(см. разд. 2.1)*.

2.5.2. Расчет среднего значения с точностью до целых

Для расчета среднего арифметического нескольких чисел с точностью до целых необходимо определить сумму всех чисел (`сум`) и их количество (`кол`), а затем рассчитать искомое значение `сред`:

```
сред := int(сум/кол)
```

Если обрабатываемые числа — целые, то задача может быть решена также следующим образом:

```
сред := div(сум, кол)
```

где `div` — функция, возвращающая целую часть от деления первого параметра на второй.

Язык Паскаль

```
sred := trunc(sum/kol);
```

или

```
sred := sum div kol;
```

2.5.3. Преобразование строкового представления числа в число

В школьном алгоритмическом языке преобразование строкового представления целого или вещественного числа в число выполняется соответственно с помощью стандартных функций `лит_в_цел` и `лит_в_вещ`, общий вид которых:

```
лит_в_цел(строка, успех)
```

и

```
лит_в_вещ(строка, успех)
```

где `строка` — преобразуемая строка; `успех` — величина логического типа. Если `строка` содержат только целое/вещественное число, то величине `успех` присваивается значение **`да`**, и функция возвращает соответствующее число, в противном случае величине `успех` присваивается значение **`нет`**, и функция возвращает 0.

В языке Паскаль для решения указанной задачи используется процедура `Val`. Кроме того, в случае преобразования символов-цифр может быть применена функция `Ord`.

Задания для самостоятельной работы

1. После единых выпускных экзаменов по информатике в район пришла информация о том, какой ученик какой школы сколько набрал баллов. Необходимо опреде-

лить сумму баллов, набранных всеми учениками каждой школы. Экзамен сдавали ученики школ с номерами от 1-го до 20-го.

2. Для условий предыдущей задачи для каждой школы определить средний балл ее учеников с точностью до целых.

3. Дана строка формата:

```
<двузначное число> <пробел> <цифра> <пробел> <трехзначное число>
```

Определить сумму трех числовых значений.

ГЛАВА 3

Задачи С2

В задачах С2, как указывается в [14], проверяются "умения написать короткую (10–15 строк) простую программу (например, обработки массива) на языке программирования или записать алгоритм на естественном языке".

Обратим внимание на то, что начиная с демонстрационного варианта ЕГЭ 2010 года в условии задачи С2 приводится перечень переменных величин, используемых в программе ("Исходные данные объявлены так, как показано ниже"), и требование, касающееся их использования ("Запрещается использовать переменные, не описанные ниже, но разрешается не использовать часть из них"). Пример:

Естественный язык

Объявляем массив *A* из 30 элементов.

Объявляем целочисленные переменные *I, X, Y*.

Объявляем вещественную переменную *S*.

В цикле от 1 до *N* вводим элементы массива *A* с 1-го по 30-й.

...

В качестве ответа вам необходимо привести фрагмент программы (или описание алгоритма на естественном языке), который должен находиться на месте многоточия. Вы можете записать решение также на другом языке программирования (укажите название и используемую версию языка программирования, например Borland Pascal 7.0) или в виде блок-схемы. В этом случае вы должны использовать переменные, аналогичные переменным, используемым в алгоритме, записанном на естественном языке, с учетом синтаксиса и особенностей используемого вами языка программирования.

Учитывая это, мы не будем при решении приводить этапы, связанные с описанием величин, заполнением и выводом массива на экран (вывод отсутствует и в [5–7, 12, 16]).

3.1. Задача из [4]

Условие

Опишите на русском языке или на одном из языков программирования алгоритм получения из заданного целочисленного массива размером 30 элементов другого массива, который будет содержать модули значений элементов первого массива (не используя специальной функции, вычисляющей модуль числа).

Анализ решения

Ясно, что необходимо использовать два массива размером 30 элементов: заданный и тот, который требуется получить. Пусть их имена соответственно `m1` и `m2`.

Вспомним *задачу 1.11.4* ("Изменение значений элементов массива с заданными свойствами"). В ней изменения происходили в заданном массиве (для некоторых элементов). В данном случае необходимо значения элементов массива `m1`, удовлетворяющего условию (они должны быть отрицательными), менять на положительные, но новые значения записывать на соответствующее место в массив `m2`. Другие значения (неотрицательные) остаются без изменения и также записываются на соответствующее место во втором массиве.

Описание фрагмента алгоритма на русском языке

В цикле с параметром `i` от первого элемента до последнего сравниваем элементы заданного массива с нулем и в зависимости от результата сравнения получаем значения элементов второго массива:

- для отрицательных элементов меняем знак (умножая их значения на –1 или записывая значения со знаком "–");
- для неотрицательных элементов — значение не меняем.

Каждое `i`-е значение (измененное или исходное) записываем в элемент второго массива с тем же индексом.

Фрагменты программы

Школьный алгоритмический язык

```
нц для i от 1 до n[1]
  если m1[i] < 0
    то
      m2[i] := –m1[i]
    иначе
      m2[i] := m1[i];
  все
кц
```

[1] Здесь и далее принимается, что массив описан с индексами от 1 до *n*.

Язык Паскаль

```
for i := 1 to n do
  if m1[i] < 0
    then m2[i] := -m1[i]
    else m2[i] := m1[i];
```

где n — число элементов массивов m1 и m2.

3.2. Задача варианта 8 из [12]

Условие

Дан целочисленный массив из 40 элементов. Элементы массива могут принимать произвольные значения. Опишите на русском языке или на одном из языков программирования алгоритм, который находит и выводит номер элемента массива, наименее отличающегося от среднего арифметического всех его элементов.

Анализ решения

Задача представляет собой "сумму" двух частных задач:

1. Расчета среднего арифметического всех элементов массива.
2. Поиска индекса элемента, значение которого наименее отличается от найденного среднего арифметического.

Для решения первой частной задачи необходимо найти сумму всех элементов массива *(см. разд. 1.8.4)*.

Вторая частная задача аналогична *задаче 1.9.4*, с той разницей, что рассматриваться должны не значения элементов массива, а их "отклонения" от среднего арифметического. При этом, поскольку эти "отклонения" могут быть как положительными, так и отрицательными, сравнения должны проводиться с использованием функции, возвращающей модуль числа.

Перечень величин

n — размер массива (константа, равная 40);

m — заданный массив (из *n* элементов целого типа);

сумма — сумма всех значений элементов массива;

сред — их среднее арифметическое (величина вещественного типа);

i — индексы элементов;

номер — искомый номер элемента.

Описание фрагмента алгоритма на естественном языке[1]

1. Записываем[2] в переменную сумма нулевое значение.

[1] Так записано в [12] и в других источниках.

[2] Мы используем стиль изложения алгоритма, аналогичный применяемому в [2–7].

2. В цикле с параметром i перебираем все элементы от первого до последнего и значение переменной сумма увеличиваем на значение текущего (i-го) элемента.
3. Рассчитываем значение сред, деля значение сумма на количество элементов n.
4. Переменной номер присваиваем начальное значение, равное 1.
5. В цикле с параметром i перебираем все элементы массива от второго до n-го. Если модуль разности текущего, i-го, элемента и значения сред оказывается меньше модуля разности значения элемента с индексом номер и значения сред, то в переменную номер записываем значение i.
6. Выводим значение переменной номер.

Фрагменты программы

Школьный алгоритмический язык

```
|Расчет среднего арифметического
сумма := 0
нц для i от 1 до n
сумма := сумма + m[i]
кц
сред := сумма/n
|Поиск номера элемента с минимальным
|отклонением от среднего арифметического
номер := 1
нц для i от 2 до n
  если abs(m[i] - сред) < abs(m[номер] - сред)
   то
    номер := i
   все
кц
|Вывод результата
вывод нс, номер
```

Язык Паскаль

```
{Расчет среднего арифметического}
summa := 0;
for i := 1 to n do
  summa := summa + m[i];
sred := summa/n;
{Поиск номера элемента с минимальным отклонением от среднего арифметического}
nomer := 1;
for i := 2 to n do
  if abs(m[i] - sred) < abs(m[nomer] - sred)
    then nomer := i;
{Вывод результата}
write(nomer);
```

Примечания

1. В решении, приведенном в [12], величина `сумма` используется также для хранения значения среднего арифметического (`сумма := сумма/n`).
2. Аналогично можно найти не номер, а значение соответствующего элемента.

3.3. Задача из [2]

Условие

Опишите на русском языке или на одном из языков программирования алгоритм поиска номера первого из двух последовательных элементов в целочисленном массиве из 30 элементов, сумма которых максимальна (если таких пар несколько, то можно выбрать любую из них).

Анализ решения

Здесь для каждой пары последовательных элементов необходимо рассчитать сумму значений элементов и найти индекс такого "левого" элемента пары, для которого эта сумма максимальна (последняя задача аналогична *задаче 1.9.3*). В ходе расчетов переменную величину, хранящую значение суммы для каждой пары, можно не применять, а использовать выражение суммы значений двух последовательных элементов ($m[i] + m[i + 1]$).

Перечень величин

n — размер массива (константа, равная 30);

m — заданный массив (из n элементов целого типа);

макс_сумма — максимальное значение суммы двух последовательных элементов среди рассмотренных пар;

i — индексы "левых" элементов в каждой паре;

номер_макс — искомый индекс.

Описание фрагмента алгоритма на естественном языке

1. Записываем в переменную `номер_макс` значение, равное 1, а в переменную `макс_сумма` — сумму значений первого и второго элементов (первая пара).
2. В цикле с параметром `i` перебираем все элементы от второго до предпоследнего и сравниваем сумму `i`-го и `(i + 1)`-го элементов со значением `макс_сумма`. Если сумма двух элементов оказывается больше, то в переменную `номер_макс` записываем значение `i`, а в переменную `макс_сумма` — сумму значений `i`-го и `(i + 1)`-го элементов.
3. Выводим значение переменной `номер_макс`.

Фрагменты программы

Школьный алгоритмический язык

```
номер_макс := 1           |Начальные
макс_сумма := m[1] + m[2] |значения
```

```
нц для i от 2 до n - 1
  если m[i] + m[i + 1] > макс_сумма
    то
      номер_макс := i
      макс_сумма := m[i] + m[i + 1]
   все
кц
вывод нс, номер_макс
```

Язык Паскаль

```
{Начальные значения}
nomer_max := 1;
max_summa := m[1] + m[2];
for i := 2 to n — 1 do
  if m[i] + m[i + 1] > max_summa then
    begin
      nomer_max := i;
      max_summa := m[i] + m[i + 1]
    end;
write(nomer_max);
```

Примечание

Аналогично можно найти не индекс, а значение соответствующего элемента.

3.4. Задача варианта 10 из [12]

Условие

Дан целочисленный массив из 40 элементов. Элементы массива могут принимать произвольные значения. Опишите на русском языке или на одном из языков программирования алгоритм, который находит и выводит номера двух элементов массива, наименее отличающихся друг от друга.

Анализ решения

В данной задаче надо найти не одну, а две величины (два индекса). Рассмотреть все возможные пары элементов с индексами `i` и `j` можно с помощью такого вложенного цикла:

```
нц для i от 1 до n - 1
  нц для j от i + 1 до n


  кц
кц
```

Для каждой пары индексов `i` и `j` необходимо рассчитать разность значений их элементов и найти пару индексов, для которых эта разность минимальна (последняя

задача аналогична *задаче 1.9.4*). Ясно, что здесь, как и при решении *задачи 3.2*, сравнения должны проводиться с использованием функции, возвращающей модуль числа. В качестве начальных значений (рассчитываемых до вложенного цикла) можно принять значения, соответствующие индексам 1 и 2.

Перечень величин (кроме индексов i и j)

n — размер массива (константа, равная 40);

m — заданный массив (из *n* элементов целого типа);

*инд*1 и *инд*2 — искомые номера элементов;

мин_разн — модуль минимальной разности значений пар элементов.

Описание фрагмента алгоритма на естественном языке

1. Записываем в переменную `инд1` значение, равное 1, в переменную `инд2` — равное 2, а в переменную `мин_разн` — равное абсолютному значению разности между `m[инд1]` и `m[инд2]`.
2. В цикле с параметром `i` перебираем все элементы от первого до предпоследнего, а внутри него в цикле с параметром `j` перебираем все элементы от `(i + 1)`-го до последнего. Внутри второго цикла сравниваем:
   - абсолютное значение разности между элементами с индексами `инд1` и `инд2`;
   - значение `мин_разн`.

   Если первое значение оказывается меньше второго, то в переменную `мин_разн` записываем первое значение, в переменную `инд1` — значение, равное `i`, в переменную `инд2` — равное `j`.
3. Выводим значения переменных `инд1` и `инд2`.

Фрагменты программы

Школьный алгоритмический язык

```
инд1 := 1
инд2 := 2
|Начальное значение величины мин_разн
мин_разн := abs(m[инд1] - m[инд2])
нц для i от 1 до n - 1
  нц для j от i + 1 до n
    если abs(m[i] - m[j]) < мин_разн
       то
         мин_разн := abs(m[i] - m[j])
         инд1 := i
         инд2 := j
    все
  кц
кц
вывод нс, инд1, " ", инд2
```

Язык Паскаль

```
ind1 := 1;
ind2 := 2;
{Начальное значение величины min_razn}
min_razn := abs(m[ind1] - m[ind2]);
for i := 1 to n - 1 do
  for j := i + 1 to n do
    if abs(m[i] - m[j]) < min_razn then
      begin
        min_razn := abs(m[i] - m[j]);
        ind1 := i;
        ind2 := j
      end;
write(ind1, ' ', ind2);
```

Примечание

Аналогично можно найти не индексы, а значения соответствующих элементов.

3.5. Задача варианта 9 из [12]

Условие

Дан целочисленный массив из 40 элементов. Элементы массива могут принимать произвольные значения. Опишите на русском языке или на одном из языков программирования алгоритм, который находит и выводит номера двух элементов массива, сумма которых минимальна.

Анализ решения

Данную задачу можно решить следующим образом: рассмотреть все возможные пары элементов массива так, как это делалось в *задаче 3.4*, и из всех пар найти такую пару, для которых сумма значений элементов минимальна (последняя задача также аналогична решавшейся в предыдущей задаче). Конечно, функцию `abs` здесь использовать необходимости нет.

Фрагменты соответствующей программы

Школьный алгоритмический язык

```
|Начальные значения величин
инд1 := 1
инд2 := 2
мин_сумма := m[1] + m[2]
нц для i от 1 до n - 1
  нц для j от i + 1 до n
    если m[i] + m[j] < мин_сумма
      то
        мин_сумма := m[i] + m[j]
```

```
          инд1 := i
          инд2 := j
      все
    кц
кц
вывод нс, инд1, " ", инд2
```

Язык Паскаль

```
{Начальные значения величин}
ind1 := 1;
ind2 := 2;
min_summa := m[1] + m[2];
for i := 1 to n - 1 do
  for j := i + 1 to n do
    if m[i] + m[j] < min_summa then
      begin
        min_summa := m[i] + m[j];
        ind1 := i;
        ind2 := j
      end;
write(ind1, ' ', ind2);
```

При использовании описанного метода решения общее число сравнений в программе примерно равно n2/2. Это число можно значительно сократить, если применить метод, основанный на том, что искомая минимальная сумма будет у двух элементов, которые стояли бы на двух первых местах, если массив был упорядочен по неубыванию. Иными словами, мы пришли к задаче нахождения индексов первого и второго минимумов (при первом толковании понятия "второй минимум" — *см. разд. 2.4.1*). Именно такой вариант решения задачи представлен в [12]. Поэтому приведем его (см. также *задачу 3.21*).

Описание фрагмента алгоритма на естественном языке

1. Записываем в переменную инд1 значение, равное 1, в переменную инд2 — равное 2.
2. Сравниваем значения первого и второго элементов массива. Если значение второго элемента массива меньше, чем первого, то записываем в переменную инд1 значение, равное 2, в переменную инд2 — равное 1.
3. В цикле с параметром i от 3-го элемента до последнего сравниваем текущие элементы с элементом массива с индексом инд1:
   - если текущий (i-й) элемент массива окажется меньше, то в переменную инд2 записываем значение переменной инд1, а в переменную инд1 — значение индекса текущего элемента i;

- в противном случае сравниваем значение текущего элемента с элементом массива с индексом инд2; если i-й элемент окажется меньше, то в переменную инд2 записываем значение индекса текущего элемента i.

4. Выводим результат (значения инд1 и инд2).

Фрагменты программы

Школьный алгоритмический язык

```
|Начальные значения величин
инд1 := 1 |Индекс первого минимума
инд2 := 2 |Индекс второго минимума
если m[2] > m[1]
  то
    инд1 := 2
    инд2 := 1
все
нц для i от 3 до n
  если m[i] < m[инд1]
    то
      инд2 := инд1;
      инд1 := i
    иначе
      если m[i] < m[инд2]
        то
          инд2 := i
      все
  все
кц
вывод нс, инд1, " ", инд2
```

Язык Паскаль

```
{Начальные значения величин}
ind1 := 1; {Индекс первого минимума}
ind2 := 2; {Индекс второго минимума}
if m[ind2] < m[ind1] then
  begin
    ind1 := 2;
    ind2 := 1
  end;
for i := 3 to n do
  if m[i] < m[ind1] then
    begin
      ind2 := ind1;
      ind1 := i
    end
```

```
  else
    if m[i] < m[ind2]
        then ind2 := i;
write(ind1, ' ', ind2);
```

Обратим внимание на то, что в *главе 2* определялись значения двух первых минимумов, а не их индексов, как в обсуждаемой задаче.

3.6. Задача варианта 4 из [12]

Условие

Дан целочисленный массив из 30 элементов. Элементы массива могут принимать произвольные значения. С клавиатуры вводится целое число *X*. Опишите на русском языке или на одном из языков программирования алгоритм, который находит и выводит наименьший номер элемента массива, равного *X*, или сообщение, что такого элемента нет.

Анализ решения

Проще всего решить задачу аналогично тому, как решался вариант *задачи 1.8.1.2* ("Поиск индекса первого элемента массива, равного некоторому числу") при реализации поиска элемента, начиная с его конца:

```
|Поиск искомого номера
номер := 0
нц для i от n до 1 шаг -1
  если m[i] = X
    то
     номер := i
  все
кц
|Вывод результата
если номер > 0
  то
   вывод нс, номер
  иначе
   вывод нс, "Такого числа в массиве нет"
все
```

где:

`n` — размер массива (константа, равная 30);

`m` — заданный массив (из *n* элементов целого типа);

`X` — число, вводимое с клавиатуры;

`i` — индексы элементов;

`номер` — искомый номер элемента.

Соответствующий фрагмент на языке Паскаль:

```
nomer := 0;
for i := n downto 1 do
  if m[i] = X
    then nomer := i;
{Вывод результата}
if nomer > 0
   then write(nomer)
   else write('Такого числа в массиве нет');
```

Однако целесообразно прекратить обработку массива после нахождения (возможного) искомого элемента *(см. задачу 1.8.1.1)*.

Описание фрагмента соответствующего алгоритма на естественном языке

1. Записываем в переменную номер значение, равное 0.
2. В цикле с предусловием, пока значение номер меньше или равно n и пока значение m[номер] не равно X, увеличиваем значение переменной номер на 1.
3. Если значения номер оказалось равно нулю, то выводим сообщение об отсутствии в массиве чисел, равных X, в противном случае выводим значение переменной номер.

Фрагменты соответствующей программы

Школьный алгоритмический язык

```
номер := 0
нц пока номер <= n и m[номер] <> X
  номер := номер + 1
кц
|Вывод результата
если номер >0
  то
   вывод нс, номер
  иначе
   вывод нс, "Такого числа в массиве нет"
все
```

Во фрагменте на языке Паскаль применим вариант, учитывающий возможный выход за пределы массива *(см. разд. 1.8.1.1)*:

```
nomer := 1;
while (nomer < n) and (m[nomer] <> X do
  nomer := nomer + 1
{Вывод результата}
if m[nomer] <> X
  then write('Такого числа в массиве нет')
    else write(nomer);
```

3.7. Задача варианта 2 из [12]

Условие

Дан вещественный массив из 50 элементов. Элементы массива могут принимать произвольные значения. Опишите на русском языке или на одном из языков программирования алгоритм, который находит и выводит наименьший номер отрицательного элемента массива или сообщение, что такого элемента нет.

Задача решается аналогично предыдущей. Предлагаем читателям оформить решение самостоятельно.

3.8. Задача варианта 1 из [16]

Условие

Опишите на русском языке или на одном из языков программирования алгоритм подсчета суммы всех отрицательных элементов заданного целочисленного массива размером из 30 элементов. Если отрицательных элементов нет, сообщите об этом.

Анализ решения

Задача такого типа рассматривалась в *разд. 1.11.1*. Обратим внимание на возможность отсутствия в массиве отрицательных элементов.

Перечень величин

n — размер массива (константа, равная 30);

m — заданный массив (из *n* элементов целого типа);

сумма — сумма всех отрицательных элементов массива;

кол — количество таких элементов;

i — индексы элементов.

Описание фрагмента алгоритма на естественном языке

1. Записываем в переменные `сумма` и `кол` нулевые значения.
2. В цикле с параметром `i` от первого элемента до последнего сравниваем элементы заданного массива с нулем и для отрицательных элементов:
   - прибавляем к текущему значению переменной `сумма` значение рассматриваемого (`i`-го) элемента;
   - увеличиваем значение переменной `кол` на 1.
3. Сравниваем значение `кол` с нулем (проверка наличия в массиве отрицательных элементов). Если `кол` больше нуля, то выводим значение `сумма`, в противном случае выводим сообщение "Отрицательных элементов в массиве нет".

Фрагменты программы

Школьный алгоритмический язык

```
|Расчет значений сумма и кол
сумма := 0
кол := 0
нц для i от 1 до n
  если m[i] < 0
    то
      сумма := сумма + m[i]
      кол := кол + 1
  все
кц
|Вывод результата
если кол > 0
  то
   вывод нс, сумма
  иначе
   вывод нс, "Отрицательных элементов в массиве нет"
все
```

Язык Паскаль

```
{Расчет значений summa и kol}
summa := 0;
for i := 1 to n do
 begin
   summa := summa + m[i];
   kol := kol + 1
 end;
{Вывод результата}
if kol > 0
  then write(summa)
    else write('Отрицательных элементов в массиве нет');
```

Примечание

Величину кол/kol можно не использовать. В этом случае сообщение об отсутствии в массиве отрицательных элементов выводится в случае, когда значение величины сумма/summa равно нулю.

3.9. Задача варианта 2 из [16]

Условие

Опишите на русском языке или на одном из языков программирования алгоритм подсчета произведения всех отрицательных элементов заданного целочисленного

массива размером 30 элементов в предположении, что в массиве есть хотя бы один отрицательный элемент.

Анализ решения

Задача такого типа ранее не рассматривалась. Однако видно, что она отличается от предыдущей тем, что рассчитывается не сумма, а произведение, а также тем, что случай, когда в массиве нет отрицательных элементов, рассматриваться не должен. Напомним также, что начальное значение произведения должно быть равно 1.

Оформите решение самостоятельно.

3.10. Задача варианта 3 из [16]

Условие

В целочисленном массиве размером 30 элементов задан рост учащихся выпускного класса (в сантиметрах). Опишите на русском языке или на одном из языков программирования алгоритм подсчета количества учащихся, чей рост превосходит 175 см. Если таких учащихся нет, сообщите об этом.

Анализ решения

Это задача типа *задачи 1.11.2* ("Нахождение количества элементов массива с заданными свойствами") с учетом возможности отсутствия в массиве таких элементов.

Перечень величин

n — размер массива (константа, равная 30);

m — заданный массив (из *n* элементов целого типа);

кол — количество элементов, больших 175;

i — индексы элементов.

Описание фрагмента алгоритма на естественном языке

1. Записываем в переменную `кол` нулевое значение.
2. В цикле с параметром `i` от первого элемента до последнего сравниваем элементы заданного массива с числом 175 и для элементов, которые больше 175, увеличиваем значение переменной `кол` на 1.
3. Сравниваем значение `кол` с нулем. Если `кол` больше нуля, то выводим соответствующее значение, в противном случае выводим сообщение "Таких учащихся нет".

Фрагменты программы

Школьный алгоритмический язык

```
|Расчет значения кол
кол := 0
```

```
нц для i от 1 до n
  если m[i] > 175
    то
      кол := кол + 1
  все
кц
|Вывод результата
если кол > 0
  то
   вывод нс, кол
  иначе
   вывод нс, "Таких учащихся нет"
все
```

Язык Паскаль

```
{Расчет значения kol}
kol := 0;
for i := 1 to n do
  if m[i] > 175
    then kol := kol + 1;
{Вывод результата}
if kol > 0
  then write(kol)
    else write('Таких учащихся нет');
```

3.11. Задача варианта 4 из [16]

Условие

В вещественном массиве размером 30 элементов задан вес спортсменок одной команды (в килограммах с округлением до десятых). Опишите на русском языке или на одном из языков программирования алгоритм подсчета количества спортсменок, чей вес превышает 50 кг, но не более 57 кг. Если таких спортсменок нет, сообщите об этом.

Анализ решения

Решение аналогично решению предыдущей задачи, с той разницей, что условие для подсчета — сложное (составное):

```
вес[i] > 50 и вес[i] <= 57
```

Оформите его самостоятельно.

3.12. Задача варианта 6 из [12]

Условие

Дан целочисленный массив из 40 элементов. Элементы массива могут принимать произвольные значения. Опишите на русском языке или на одном из языков про-

граммирования алгоритм, который находит и выводит номер третьего положительного элемента массива (если из массива вычеркнуть все неположительные элементы, этот элемент стоял бы в получившемся массиве на третьем месте). Если в массиве меньше, чем три положительных элемента, вывести об этом сообщение.

Анализ решения

Решать задачу за два прохода по массиву:

1) сначала подсчитать количество положительных элементов;
2) потом в зависимости от этого количества либо вывести сообщение об отсутствии трех положительных элементов, либо на втором проходе найти соответствующий элемент и вывести его;

— нерационально.

Лучше за один проход подсчитывать количество положительных элементов, а когда это количество станет равно 3 (этого может и не быть) — запомнить номер соответствующего элемента в качестве искомого значения.

Перечень величин

n — размер массива (константа, равная 40);

m — заданный массив (из *n* элементов вещественного типа);

кол — количество положительных элементов;

i — индексы элементов;

номер — искомый номер элемента.

Описание фрагмента алгоритма на естественном языке

1. Записываем в переменную `кол` нулевое значение.
2. В цикле с параметром `i` от первого элемента до последнего сравниваем элементы заданного массива с нулем. Если текущий элемент больше нуля, то:
   - увеличиваем значение переменной `кол` на 1;
   - сравниваем значение переменной `кол` с числом 3; если оно равно трем, то записываем в переменную `номер` индекс `i` текущего элемента.
3. Сравниваем значение `кол` с числом 3. Если это значение больше либо равно трем, то выводим значение `номер`, в противном случае выводим сообщение "В массиве нет такого элемента".

Фрагменты программы

Школьный алгоритмический язык

```
кол := 0
нц для i от 1 до n
  если m[i] > 0
    то
      кол := кол + 1
```

```
      если кол = 3
        то
          номер := i
      все
  все
кц
|Вывод результата
если кол >= 3
  то
   вывод нс, номер
  иначе
   вывод нс, "В массиве нет такого элемента"
все
```

Язык Паскаль

```
kol := 0;
for i := 1 to n do
  if m[i] > 0 then
    begin
      kol := kol + 1;
      if kol = 3 then nomer := i
    end;
{Вывод результата}
if kol >= 3
  then write(nomer)
    else write('В массиве нет такого элемента');
```

3.13. Задача варианта 5 из [16]

Условие

Опишите на русском языке или на одном из языков программирования алгоритм подсчета среднего значения отрицательных элементов в целочисленном массиве из 30 элементов в предположении, что в массиве есть хотя бы один отрицательный элемент.

Анализ решения

Такая задача рассматривалась в *разд. 1.11.3* ("Нахождение среднего арифметического значения элементов массива с заданными свойствами"). Случай, когда в массиве нет отрицательных элементов, рассматриваться не должен.

Перечень величин

n — размер массива (константа, равная 30);

m — заданный массив (из *n* элементов целого типа);

сумма — сумма всех отрицательных элементов массива;

кол — количество таких элементов;

сред — среднее арифметическое значение отрицательных элементов массива (величина вещественного типа);

i — индексы элементов.

Описание фрагмента алгоритма на естественном языке

1. Записываем в переменные сумма и кол нулевые значения.
2. В цикле с параметром i от первого элемента до последнего сравниваем элементы заданного массива с нулем и для отрицательных элементов:
   - прибавляем к текущему значению переменной сумма значение рассматриваемого (i-го) элемента;
   - увеличиваем значение переменной кол на 1.
3. Рассчитываем значение сред как частное от деления сумма на кол.
4. Выводим результат (значение сред).

Фрагменты программы

Школьный алгоритмический язык

```
|Расчет значений сумма и кол
сумма := 0
кол := 0
нц для i от 1 до n
  если m[i] < 0
    то
      сумма := сумма + m[i]
      кол := кол + 1
  все
кц
|Расчет значения сред
сред := сумма/кол
|Вывод результата
вывод нс, сред
```

Язык Паскаль

```
{Расчет значений summa и kol}
summa := 0;
kol := 0;
for i := 1 to n do
 begin
   summa := summa + m[i];
   kol := kol + 1
 end;
```

```
{Расчет значения sred}
sred := summa/kol;
{Вывод результата}
write(sred:7:2);
```

Примечание

Величину `сред/sred` можно не использовать. В этом случае в качестве результата выводится выражение `сумма/кол` (`summa/kol`).

3.14. Задача из [6]

Условие

Дан целочисленный массив из 30 элементов. Элементы массива могут принимать значения от 0 до 1000. Опишите на русском языке или на одном из языков программирования алгоритм, который позволяет подсчитать и вывести среднее арифметическое элементов массива, имеющих нечетное значение.

Гарантируется, что в исходном массиве хотя бы один элемент имеет нечетное значение.

Анализ решения

Задача решается аналогично предыдущей. Предлагаем читателям оформить решение самостоятельно.

Обратим также внимание на важное обстоятельство, касающееся программ на языке Паскаль, — если бы в массиве могли присутствовать и отрицательные значения, то проверку элемента массива на "нечетность" следующим образом:

```
m[i] mod 2 = 1
```

проводить было бы нельзя, т. к. в этом языке остаток от деления нечетного отрицательного числа на 2 равен –1 (правильное условие: `m[i] mod 2 <> 0`).

3.15. Задача варианта 10 из [16]

Условие

Дан целочисленный массив из 30 элементов. Элементы массива могут принимать значения от –20 до 20 — сведения о температуре за каждый день ноября. Опишите на русском языке или на одном из языков программирования алгоритм, который находит и выводит максимальную температуру среди дней, когда были заморозки (т. е. температура опускалась ниже нуля). Гарантируется, что хотя бы в один день ноября была отрицательная температура.

Анализ решения

Данная задача относится к типу "Определение максимального значения среди тех элементов массива, которые удовлетворяют некоторому условию" (аналог *задачи 1.11.7*). При этом диапазон значений элементов массива известен.

Перечень величин

n — размер массива (константа, равная 30);

m — заданный массив (из *n* элементов целого типа);

макс — максимальное значение среди отрицательных элементов массива;

i — индексы элементов.

Описание фрагмента алгоритма на естественном языке

1. Записываем в переменную макс значение, равное –20 (оно не больше минимального из отрицательных элементов массива).
2. В цикле с параметром i от первого элемента до последнего сравниваем элементы заданного массива с нулем и для отрицательных элементов дополнительно сравниваем их значение со значением переменной макс[1]. Если текущий элемент массива больше макс, то в качестве нового значения переменной макс принимаем значение этого элемента.
3. Выводим результат (значение макс).

Фрагменты программы

Школьный алгоритмический язык

```
макс := -20
нц для i от 1 до n
  если m[i] < 0
    то
      если m[i] > макс
        то
          макс := m[i]
      все
  все
кц
вывод нс, макс
```

Язык Паскаль

```
max := -20;
for i := 1 to n do
  if m[i] < 0 then
    if m[i] > max then
      then max := m[i];
write(max);
```

[1] Можно также сразу записать сложное условие: m[i] > 0 **и** m[i] > макс, однако при этом в программах на некоторых языках будет проводиться полная проверка условия даже в случаях, когда очередной элемент — неотрицательный *(см. разд. 1.8.1.1)*. Сложное условие можно записывать и в нескольких следующих задачах. *См. также разд. 1.11.7*.

3.16. Задача варианта 9 из [16]

Условие

Опишите на русском языке или на одном из языков программирования алгоритм вычисления разности максимального среди элементов, имеющих четные значения, и максимального среди элементов, имеющих нечетные значения, в заданном целочисленном массиве из 30 положительных элементов (в предположении, что в массиве есть и четные, и нечетные элементы).

Анализ решения

В данной задаче по сравнению с предыдущей добавляется также вторая частная задача того же типа, но с другим условием для учета элементов. Причем отдельно второе условие можно не проверять, а рассматривать его как "ветвь" **`иначе`** (**`if`**) полного условного оператора. Конечно, после вычисления двух максимальных значений в качестве ответа нужно вывести их разность.

Перечень величин

n — размер массива (константа, равная 30);

m — заданный массив (из *n* элементов целого типа);

макс_чет — максимальное значение среди четных элементов массива;

макс_нечет — то же, среди нечетных элементов;

i — индексы элементов.

Описание фрагмента алгоритма на естественном языке

1. Записываем в переменные `макс_чет` и `макс_нечет` значения, равные нулю (согласно условию, все значения в массиве — положительные, а 0 не больше минимального из четных и не больше минимального из нечетных элементов массива).
2. В цикле с параметром `i` от первого элемента до последнего проверяем, является ли очередной, `i`-й, элемент заданного массива четным:
   - если является, то дополнительно сравниваем значение `i`-го элемента со значением переменной `макс_чет`. Если он больше (`m[i] > макс_чет`), то в качестве нового значения переменной `макс_чет` принимаем значение этого элемента `m[i]`;
   - если не является, то дополнительно сравниваем значение `i`-го элемента со значением переменной `макс_нечет`. Если он больше (`m[i] > макс_нечет`), то в качестве нового значения переменной `макс_нечет` принимаем значение этого элемента.
3. Выводим результат (значение `макс_чет - макс_нечет`).

Фрагменты программы

Школьный алгоритмический язык

```
макс_чет := 0
макс_нечет := 0
нц для i от 1 до n
  если mod(m[i], 2) = 0
    то
      если m[i] > макс_чет
        то
          макс_чет := m[i]
      все
    иначе
      если m[i] > макс_нечет
        то
          макс_нечет := m[i]
      все
  все
кц
|Вывод результата
вывод нс, макс_чет - макс_нечет
```

Язык Паскаль

```
max_chet := 0;
max_nechet := 0;

for i := 1 to n do
 if m[i] mod 2 = 0 then
    if m[i] > max_chet then
      max_chet := m[i]
 else
    if m[i] > max_nechet then
      max_nechet := m[i]
{Вывод результата}
write(max_chet - max_nechet);
```

3.17. Задача из [5]

Условие

Дан целочисленный массив из 30 элементов. Элементы массива могут принимать целые значения от 0 до 100 — баллы учащихся выпускного класса за итоговый тест по информатике. Для получения положительной оценки за тест требовалось набрать не менее 20 баллов. Опишите на русском языке или на одном из языков программирования алгоритм, который находит и выводит минимальный балл среди

учащихся, получивших за тест положительную оценку. Известно, что в классе хотя бы один учащийся получил за тест положительную оценку.

Анализ решения

Данная задача относится к типу "Определение минимального значения среди тех элементов массива, которые удовлетворяют некоторому условию" *(см. задачу 1.11.7)*. При этом известен диапазон значений элементов массива, а также тот факт, что соответствующие элементы в массиве имеются.

Перечень величин

n — размер массива (константа, равная 30);

m — заданный массив (из *n* элементов целого типа);

мин — минимальное значение среди элементов массива, не меньших 20 (искомая величина);

i — индексы элементов.

Описание фрагмента алгоритма на естественном языке

1. Записываем в переменную мин значение, равное 100 (оно не меньше максимального из элементов массива).
2. В цикле с параметром i от первого элемента до последнего сравниваем элементы заданного массива с 20. Если текущий (i-й) элемент массива больше или равен 20, то дополнительно сравниваем его значение со значением переменной мин. Если он меньше мин, то в качестве нового значения переменной мин принимаем значение этого элемента.
3. Выводим результат (значение мин).

Фрагменты программы

Школьный алгоритмический язык

```
мин := 100
нц для i от 1 до n
  если m[i] >= 20
    то
      если m[i] < мин
        то
          мин := m[i]
      все
  все
кц
вывод нс, мин
```

Язык Паскаль

```
min := 100;
for i := 1 to n do
  if m[i] >= 20 then
```

```
    if m[i] < min then
      then min := m[i];
write(min);
```

3.18. Задача варианта 1 из [12]

Условие

Дан целочисленный массив из 28 элементов. Элементы массива могут принимать значения от 0 до 100 — процент выполнения учащимися домашних заданий по информатике. Для получения положительной оценки за год требовалось набрать не менее 40 баллов. Опишите на русском языке или на одном из языков программирования алгоритм, который находит и выводит минимальный балл среди учащихся, получивших за год положительную оценку. Гарантируется, что в классе хотя бы один учащийся получил за год положительную оценку.

Задача решается аналогично предыдущей задаче.

3.19. Задача из [7]

Условие

Дан целочисленный массив из 20 элементов. Элементы массива могут принимать целые значения от –1000 до 1000. Опишите на русском языке или на одном из языков программирования алгоритм, позволяющий найти и вывести минимальное значение среди элементов массива, которые имеют четное значение и не делятся на три. Гарантируется, что в исходном массиве есть хотя бы один элемент, значение которого четно и не кратно трем.

Анализ решения

Здесь также требуется определить минимальное значение среди тех элементов массива, которые удовлетворяют некоторому условию. Особенность в том, что это условие — сложное:

```
mod(m[i], 2) = 0 и mod(m[i], 3) <> 0
```

3.20. Задача варианта 3 из [12]

Условие

Дан вещественный массив из 40 элементов. Элементы массива могут принимать произвольные значения. Опишите на русском языке или на одном из языков программирования алгоритм, который находит и выводит минимальный положительный элемент массива или сообщение, что такого элемента нет.

Анализ решения

Особенность (и сложность!) данной задачи в том, что диапазон значений элементов массива неизвестен и что положительных элементов в массиве может не быть. Такая задача рассмотрена в *разд. 1.11.7*.

Напомним два основных этапа ее решения:

1. Ищем (пробуем найти) индекс первого положительного элемента в массиве.
2. Если он найден:

   то

   - принимаем значение этого элемента в качестве искомого минимального значения;
   - рассматриваем все следующие элементы и положительных из них сравниваем со "старым" минимальным значением. Если текущий положительный элемент меньше минимального, то принимаем его в качестве нового значения искомой величины;
   - выводим найденное минимальное значение;

   иначе

   - выводим сообщение о том, что положительных элементов в массиве нет.

Перечень величин

n — размер массива (константа, равная 40);

m — заданный массив (из *n* элементов вещественного типа);

мин — минимальное значение среди положительных элементов массива (искомая величина);

i — индекс (возможный) первого (при просмотре массива от его начала) положительного элемента;

j — индекс элементов, которые больше, чем *i*.

Описание фрагмента алгоритма на естественном языке

1. Записываем в переменную `i` значение, равное 1 (начинаем поиск с первого элемента массива).
2. В цикле с предусловием, пока `i` не больше `n` и пока `i`-й элемент — неположительный, увеличиваем значение `i` на 1.
3. Если после окончания цикла окажется, что `i` меньше либо равно `n`, то:

   - записываем в переменную мин значение `i`-го элемента;
   - в цикле с параметром `j` рассматриваем элементы от (`i + 1`)-го до последнего и сравниваем их с нулем. Если текущий элемент больше нуля, то сравниваем его со значением мин. Если он меньше минимального, то принимаем текущий положительный элемент (`m[j]`) его в качестве нового значения величины мин;
   - выводим результат (значение мин)

   иначе выводим сообщение "Таких чисел в массиве нет".

Фрагменты программы

Школьный алгоритмический язык

```
|Ищем (пробуем найти) первый положительный элемент
i := 1
нц пока i <= n и не m[i] > 0
  i := i + 1
кц
|Если такой элемент найден
если i <= n
  то
    |Его индекс равен i
    |Принимаем этот элемент в качестве минимального
    мин := m[i]
    |Рассматриваем оставшиеся элементы
    нц для j от i + 1 до n
      |и положительные из них сравниваем
      |с минимальным среди уже рассмотренных
      если m[j] > 0
        то
          если m[j] < мин
            то
              мин := m[j]
          все
      все
    кц
    |Выводим ответ
    вывод нс, мин
  иначе |Положительных элементов в массиве нет
    |Выводим соответствующее сообщение
    вывод нс, "Положительных чисел в массиве нет"
все
```

Язык Паскаль

```
{Ищем (пробуем найти) первый положительный элемент}
i := 1;
while (i <= n) and not (m[i] > 0) do
  i := i + 1;
{Если такой элемент найден}
if i <= n then
  begin
    {Его индекс равен i.
    Принимаем этот элемент в качестве минимального}
    min := m[i];
```

```
    {Рассматриваем оставшиеся элементы}
    for j := i + 1 to n do
      {и положительные из них сравниваем
      с минимальным среди уже рассмотренных}
      if m[j] > 0 then
        if m[j] < min then min := m[j];
    {Выводим ответ}
    write(min:7:2)
  end
else {Положительных элементов в массиве нет.
    Выводим соответствующее сообщение}
    write('Положительных чисел в массиве нет');
```

Приведем также компактный вариант решения, представленный в *разд. 1.11.7*.

Школьный алгоритмический язык

```
индекс_мин := 0 |Условно
|Рассматриваем все элементы
нц для i от 1 до n
  если m[i] > 0
    то |Встретился положительный элемент
     |Если это произошло впервые или он меньше "старого" минимума
     если индекс_мин = 0 или m[i] < m[индекс_мин]
       то
        |Запоминаем его индекс в качестве значения индекс_мин
        индекс_мин := i
     все
  все
кц
|Выводим ответ
если индекс_мин > 0
  то
    вывод нс, m[индекс_мин]
  иначе
    вывод нс, "Таких чисел в массиве нет"
все
```

Язык Паскаль

```
index_min := 0; {Условно}
for i := 1 to n do {Рассматриваем все элементы}
  if m[i] > 0 then
   {Встретился положительный элемент.
   Если это произошло впервые или он меньше "старого" минимума}
    if (index_min = 0) or (m[i] < m[index_min]) then
```

```
      {Запоминаем его индекс в качестве значения index_min}
      index_min := i;
{Выводим ответ}
if index_min > 0 then
  write(m[index_min]:7:2)
    else write('Таких чисел в массиве нет');
```

3.21. Задача варианта 5 из [12]

Условие

Дан целочисленный массив из 40 элементов. Элементы массива могут принимать произвольные значения. Опишите на русском языке или на одном из языков программирования алгоритм, который находит и выводит значение второго максимума (элемента, который в отсортированном по неубыванию массиве стоял бы вторым).

Анализ решения

Задача нахождения второго максимума анализировалась в *разд. 2.4.1*. Здесь принимается первое из двух возможных толкований этого понятия. Еще одна особенность — диапазон значений элементов массива неизвестен (такой вариант рассмотрен в *разд. 2.4.1.1*).

Перечень величин

n — размер массива (константа, равная 40);

m — заданный массив (из *n* элементов целого типа);

*макс*1 — максимальный элемент массива (первый максимум);

*макс*2 — второй максимум (искомое значение);

i — индексы элементов.

Описание фрагмента алгоритма на естественном языке

1. Записываем в переменную `макс1` значение 1-го элемента массива, в переменную `макс2` — 2-го элемента.
2. Сравниваем значения 1-го и 2-го элементов массива. Если 2-й элемент больше, то в переменную `макс1` записываем значение 2-го элемента массива, в переменную *макс*2 — 1-го элемента.
3. В цикле с параметром `i` от 3-го элемента до последнего сравниваем элементы массива со значением `макс1`:
   - если текущий (`i`-й) элемент массива окажется больше `макс1`, то в переменную `макс2` записываем значение переменной `макс1`, а в переменную `макс1` — значение текущего элемента массива;
   - в противном случае сравниваем значение текущего элемента массива со значением `макс2` — если `i`-й элемент окажется больше `макс2`, то в переменную `макс2` записываем значение текущего элемента массива.
4. Выводим результат (значение `макс2`).

Фрагменты программы

Школьный алгоритмический язык

```
макс1 := m[1]
макс2 := m[2]
если m[2] > m[1]
  то
   макс1 := m[2]
   макс2 := m[1]
все
нц для i от 3 до n
  если m[i] > макс1
    то
      макс2 := макс1 |Именно в таком
      макс1 := m[i]  |порядке
    иначе
      если m[i] > макс2
        то
          макс2 := m[i]
      все
  все
кц
вывод нс, макс2
```

Язык Паскаль

```
max1 := m[1];
max2 := m[2];
if m[2] > m[1] then
  begin
   max1 := m[2];
   max2 := m[1]
 end;
for i := 3 to n do
  if m[i] > max1 then
    begin
        max2 := max1; {Именно в таком}
        max1 := m[i]  {порядке}
    end
  else
    if m[i] > max2
        then max2 := m[i];
write(max2);
```

3.22. Задача варианта 6 из [16]

Условие

Опишите на русском языке или на одном из языков программирования алгоритм подсчета максимального количества подряд идущих отрицательных элементов в целочисленном массиве длины 30.

Анализ решения

Задача такого типа рассматривалась в *разд. 1.11.9*. Здесь также для краткости будем называть участок массива с подряд идущими отрицательными элементами "подмассивом".

Перечень величин

n — размер массива (константа, равная 30);

m — заданный массив (из *n* элементов целого типа);

кол_отр — количество отрицательных элементов в текущем подмассиве;

макс_отр — максимальное количество подряд идущих отрицательных элементов в подмассивах (искомая величина);

i — индексы элементов.

Описание фрагмента алгоритма на естественном языке

1. Записываем в переменные кол_отр и макс_отр нулевые значения.
2. В цикле с параметром i рассматриваем элементы от первого до последнего и сравниваем их с нулем. Если текущий элемент — меньше нуля, то увеличиваем значение кол_отр на 1, в противном случае:
   - сравниваем значение величин кол_отр и макс_отр. Если первое значение больше второго, в переменную макс_отр записываем значение кол_отр;
   - в переменную кол_отр записываем нулевое значение.
3. По окончанию цикла дополнительно сравниваем значение величин кол_отр и макс_отр. Если кол_отр > макс_отр, то в переменную макс_отр записываем значение кол_отр.
4. Выводим значение величины макс_отр.

Фрагменты программы

Школьный алгоритмический язык

```
кол_отр := 0
макс_отр := 0
нц для i от 1 до n
  если m[i] < 0
    то
      |Подмассив отрицательных элементов продолжается или начался.
      |Увеличиваем его длину на 1
      кол_отр := кол_отр + 1
```

```
    иначе |Встретился неотрицательный элемент.
      |Подмассив отрицательных элементов кончился.
      Сравниваем его длину со значением макс_отр
      если кол_отр > макс_отр
        то
          макс_отр := кол_отр
      все
      кол_отр := 0 |Новое значение
  все
кц
|Проверяем длину последнего (возможного) подмассива
если кол_отр > макс_отр
  то
    макс_отр := кол_отр
все
|Выводим ответ
вывод нс, макс_отр
```

Язык Паскаль

```
kol_otr := 0;
max_otr := 0;
for i := 1 to n do
  if m[i] < 0
    then {Подмассив отрицательных элементов продолжается или начался.
      Увеличиваем его длину на 1}
      kol_otr := kol_otr + 1
    else {Встретился неотрицательный элемент.
      Подмассив отрицательных элементов кончился.
      Сравниваем его длину со значением max_otr}
      begin
        if kol_otr > max_otr
           then max_otr := kol_otr;
        kol_otr := 0 {Новое значение}
      end;
{Проверяем длину последнего (возможного) подмассива}
if kol_otr > max_otr
  then max_otr := kol_otr;
{Выводим ответ}
write(max_otr);
```

Обратим внимание на то, что после окончания текущего подмассива величине кол_отр присваивается нулевое значение независимо от результата сравнения величин кол_отр и макс_отр.

3.23. Задача варианта 8 из [16]

Условие

Опишите на русском языке или на одном из языков программирования алгоритм подсчета максимального количества подряд идущих четных элементов в целочисленном массиве длины 30.

Задача решается аналогично предыдущей.

3.24. Задача варианта 7 из [16]

Условие

Опишите на русском языке или на одном из языков программирования алгоритм подсчета максимального количества подряд идущих элементов, каждый из которых больше предыдущего, в целочисленном массиве длины 30.

Анализ решения

Решение данной задачи также во многом аналогично двум предыдущим. Отличия:

1. Параметр цикла *i* должен меняться от 2 до *n* (или от 1 до *n* – 1).
2. Условие для подсчета длины подмассива: `m[i] > m[i - 1]` (или `m[i] < m[i + 1]`).

3.25. Задача из [3]

Условие

Опишите на русском языке или на одном из языков программирования алгоритм подсчета максимального количества подряд идущих совпадающих элементов в целочисленном массиве длины 30.

Анализ решения

Особенность данной задачи по сравнению с тремя предыдущими в том, что первый из совпадающих элементов также должен быть учтен. Поэтому начальные значения переменных величин, хранящих длину текущего подмассива и максимальную длину подмассивов, равны 1 (если потом выяснится, что во всех парах находящихся рядом элементов массива нет одинаковых, то искомая величина будет равна 1).

Перечень величин

n — размер массива (константа, равная 30);

m — заданный массив (из *n* элементов целого типа);

кол_совп — количество подряд идущих совпадающих элементов в текущем подмассиве;

макс_совп — максимальное количество подряд идущих совпадающих элементов в подмассивах, рассмотренных ранее (искомая величина);

i — индексы элементов.

Описание фрагмента алгоритма на естественном языке

1. Записываем в переменные кол_совп и макс_совп значения, равные 1.
2. В цикле с параметром i рассматриваем элементы от второго до последнего и каждый из них сравниваем с предыдущим. Если текущий (i-й) элемент равен предыдущему, то увеличиваем значение кол_совп на 1, в противном случае сравниваем значение величин кол_совп и макс_совп. Если первое значение больше второго, то:
   - в переменную макс_совп записываем значение кол_совп;
   - в переменную кол_совп записываем значение, равное 1.
3. По окончанию цикла дополнительно сравниваем значение величин кол_совп и макс_совп. Если кол_совп > макс_совп, то в переменную макс_совп записываем значение кол_совп.
4. Выводим значение величины макс_совп.

Фрагменты программы

Школьный алгоритмический язык

```
кол_совп := 1
макс_совп := 1
нц для i от 2 до n
  если m[i] = m[i − 1]
    то
      кол_совп := кол_совп + 1
    иначе
      если кол_совп > макс_совп
        то
          макс_совп := кол_совп
      все
      кол_совп := 1 |Новое значение
  все
кц
если кол_совп > макс_совп
  то
    макс_совп := кол_совп
все
вывод нс, макс_совп
```

Язык Паскаль

```
kol_sovp := 1;
max_ sovp := 1;
for i := 2 to n do
  if m[i] = m[i - 1] then
        kol_sovp := kol_ sovp + 1
```

```
   else
      begin
        if kol_sovp > max_sovp
            then max_sovp := kol_sovp;
        kol_sovp := 1 {Новое значение}
       end;
if kol_sovp > max_sovp
   then max_sovp := kol_sovp;
write(max_sovp);
```

3.26. Задача варианта 7 из [12]

Условие

Дан целочисленный массив из 40 элементов. Элементы этого массива могут принимать производные значения. Опишите на русском языке или на одном из языков программирования алгоритм, который находит и выводит сумму элементов наибольшей возрастающей последовательности подряд идущих элементов массива.

Анализ решения

Здесь необходимо определить не максимальную длину[1] (количество элементов) подмассива и не максимальную сумму его элементов, а сумму элементов в подмассиве максимальной длины. Значит, нужно в ходе поиска такого подмассива рассчитывать и запоминать также сумму его элементов. При этом в данном случае необходимо в длине подмассива учитывать и его первый элемент (при решении *задачи 3.24* этот элемент не учитывался).

Перечень величин

n — размер массива (константа, равная 40);

m — заданный массив (из *n* элементов целого типа);

кол_возр — количество элементов возрастающей последовательности в текущем подмассиве;

сумма_возр — сумма значений элементов в таком подмассиве;

макс_кол — максимальное количество элементов в возрастающих последовательностях элементов;

макс_сумма — максимальная сумма значений элементов в соответствующей последовательности;

i — индексы элементов.

[1] Хотя эту величину также придется рассчитывать, но как вспомогательную. — *См. далее*.

Фрагменты программы

Школьный алгоритмический язык

```
кол_возр := 1        |Учитываем первый
сумма_возр := m[1]   |элемент
макс_кол := 1
нц для i от 2 до n
  если m[i] > m[i – 1]
    то
      кол_возр := кол_возр + 1
      сумма_возр := сумма_возр + m[i]
    иначе
      если кол_возр > макс_кол
        то
          макс_кол := кол_возр
          макс_сумма := сумма_возр
      все
      |Новые значения
      сумма_возр := m[i]
      кол_возр := 1
  все
кц
если кол_возр > макс_кол
  то
    макс_сумма := сумма_возр
    |Здесь уточняем только значение макс_сумма
все
вывод нс, макс_сумма
```

Язык Паскаль

```
kol_vozr := 1;       {Учитываем первый}
summa_vozr := m[1]; {элемент}
max_kol := 1;
for i := 2 to n do
  if m[i] m[i] > m[i – 1] then
      begin
        kol_vozr := kol_vozr + 1
        summa_vozr := summa_vozr + m[i]
      end
   else
      begin
        if kol_vozr > max_kol then
          begin
            max_kol := kol_vozr;
            max_summa := summa_vozr
          end;
```

```
        {Новые значения}
        summa_vozr := m[i];
        kol_vozr := 1
      end;
if kol_vozr > max_kol then
  max_summa := summa_vozr; {Здесь уточняем только значение max_summa};
write(max_summa);
```

Обратим внимание на то, что начальные значения величины кол_возр принимаются равными 1. Заметим также, что в решении, приведенном в [12], начальное значение величины макс_кол принимается равным нулю. Очевидно, авторы допускают возможность того, что в массиве не будет двух и более элементов, образующих возрастающую последовательность.

В заключение приведем перечень типовых задач программирования, встречающихся в задачах С2 ЕГЭ по информатике (табл. 3.1).

Таблица 3.1

№	Типовая задача	Где встречается	Примечание
1	Изменение значений элементов массива с заданными свойствами	В [4]	Новые значения записываются в другой массив
2	Расчет среднего арифметического всех элементов массива	В [12] (вариант 8)	
3	Поиск индекса элемента массива, значение которого наименее отличается от некоторого значения	Там же	Аналог задачи поиска индекса минимального элемента. Используется функция, возвращающая абсолютное значение аргумента
4	Поиск индекса первого из двух последовательных элементов массива, сумма которых максимальна	В [2]	Аналог задачи поиска индекса максимального элемента (сравниваются суммы значений пар соседних элементов)
5	Поиск индексов двух элементов массива, наименее отличающихся друг от друга	В [12] (вариант 10)	Аналог задачи поиска индекса минимального элемента. Искомых значений — 2. Используется функция, возвращающая абсолютное значение аргумента
6	Поиск индексов двух элементов массива, сумма которых минимальна	В [12] (вариант 9)	Аналог задачи поиска индекса минимального элемента (сравниваются суммы значений пар соседних элементов). Искомых значений — 2.

Таблица 3.1 *(продолжение)*

№	Типовая задача	Где встречается	Примечание
			Задача может быть также решена как задача поиска индексов двух первых минимумов (в первом толковании термина "второй минимум")
7	Поиск индекса первого элемента массива, равного некоторому числу	В [12] (варианты 2, 4)	Искомого элемента в массиве может не быть
8	Нахождение суммы элементов массива с заданными свойствами (удовлетворяющих некоторому условию)	В [16] (вариант 1)	Элементов с заданными свойствами в массиве может не быть
9	Нахождение суммы элементов массива с заданными свойствами (удовлетворяющих некоторому условию)	В [16] (вариант 2)	
10	Нахождение количества элементов массива с заданными свойствами	В [16] (варианты 3–4)	Элементов с заданными свойствами в массиве может не быть. В задаче варианта 4 условие для подсчета — сложное (составное)
11	Поиск индекса третьего положительного элемента массива (если из массива вычеркнуть все неположительные элементы, этот элемент стоял бы в получившемся массиве на третьем месте)	В [12] (вариант 6)	Используется подсчет количества элементов массива с заданными свойствами *(см. предыдущий пункт таблицы)*
12	Нахождение среднего арифметического значения элементов массива с заданными свойствами	В [16] (вариант 5), в [6]	Случай, когда в массиве нет отрицательных элементов, рассматриваться не должен
13	Определение максимального значения среди тех элементов массива, которые удовлетворяют некоторому условию	В [16] (варианты 9–10)	Диапазон значений элементов массива известен
14	Определение минимального значения среди тех элементов массива, которые удовлетворяют некоторому условию	В [5], в [12] (вариант 1), в [7]	Известен диапазон значений элементов массива, а также тот факт, что соответствующие элементы в массиве имеются

Таблица 3.1 *(окончание)*

№	Типовая задача	Где встречается	Примечание
15	Определение минимального значения среди тех элементов массива, которые удовлетворяют некоторому условию	В [12] (вариант 3)	Диапазон значений элементов массива неизвестен. Соответствующие элементы в массиве могут не быть
16	Нахождения второго максимума (в первом толковании термина)	В [12] (вариант 5)	
17	Нахождение максимального количества подряд идущих элементов массива, обладающих заданными свойствами	В [3], в [16] (варианты 6–8), в [12] (вариант 7)	В задаче варианта 7 из [12] находится также сумма элементов такого подмассива

ГЛАВА 4

Задачи С4 из демонстрационных вариантов ЕГЭ по информатике

4.1. Задача из демонстрационного варианта экзамена 2012 года

Условие задачи

В командных соревнованиях по программированию для решения предлагается не больше 11 задач. Команда может решать предложенные задачи в любом порядке. Подготовленные решения команда посылает в единую проверяющую систему соревнований. Вам предлагается написать эффективную, в том числе и по используемой памяти, программу, которая будет статистически обрабатывать пришедшие запросы, чтобы определить наиболее популярные задачи. Следует учитывать, что количество запросов в списке может быть очень велико, т. к. многие соревнования проходят с использованием Интернета.

Перед текстом программы кратко опишите используемый вами алгоритм решения задачи.

На вход программе в первой строке подается количество пришедших запросов N. В каждой из последующих N строк записано название задачи в виде текстовой строки. Длина строки не превосходит 100 символов, название может содержать буквы, цифры, пробелы и знаки препинания.

Пример входных данных:

```
6
A+B
Крестики-Нолики
Прямоугольник
Простой делитель
A+B
Простой делитель
```

Программа должна вывести список из трех наиболее популярных задач с указанием количества запросов по ним. Если в запросах упоминаются менее трех задач, то

выведите информацию об имеющихся задачах. Если несколько задач имеют ту же встречаемость, что и третья по частоте встречаемости задача, их тоже нужно вывести.

Пример выходных данных для приведенного примера входных данных:

```
A+B 2
Простой делитель 2
Крестики-Нолики 1
Прямоугольник 1
```

Анализ решения

Прежде всего, обратим внимание на то, что в задании С4 демоварианта экзамена 2012 года впервые представлено требование кратко описать используемый алгоритм решения. Кроме того, заметим, что, по нашему мнению, использование термина "запрос" в условии является не совсем правильным (запрос — это обращение, например, к базе данных, в то время как, согласно условию, в программу поступают как бы ответы на запрос об очередной решенной задаче).

В программе на школьном алгоритмическом языке применим два массива из 11 элементов в каждом:

1. С именем `задачи` — для хранения названий задач.
2. С именем `кол_задач` — для хранения общего числа упоминания каждой из задач в результатах запросов.

Используем также величину `всего_разных_задач` — количество различных задач в результатах запросов.

Общая схема решения обсуждаемой задачи такая:

```
1. Ввести количество запросов N
2. Цикл для i от 1 до N |Ввод и обработка входных строк (каждого
                        |результата запроса)
  2.1. Прочитать название задачи
  2.2. Определить, есть ли такая задача в списке
       ранее введенных задач (в массиве задачи)
       если есть
         то
           увеличить счетчик количества соответствующей задачи
           в массиве кол_задач
         иначе
           – увеличить на 1 количество различных
             задач всего_разных_задач;
           – записать новую задачу в массив задачи;
           – соответствующему ей значению в массиве кол_задач
             присвоить 1
       все
конец цикла
```

```
3. Сравнить значение всего_разных_задач с 3:
   если всего_разных_задач < 3
     то
       вывести из массива с названиями
       задачи с индексами от 1 до всего_разных_задач
       и для каждой такой задачи – число ее вхождений
       из массива кол_задач
     иначе
       вывести из массива с названиями задачи,
       количество которых в массиве кол_задач не меньше,
       чем у задачи, занимающей третье место
       по частоте встречаемости, и для каждой такой задачи – ее
       встречаемость из массива кол_задач
   все
```

Для выполнения, так сказать, "ветви" **иначе** этапа 3 целесообразно отсортировать массив кол_задач в порядке невозрастания значений его элементов (и соответственно изменить структуру массива задачи)[1].

Из проведенного анализа вытекает, что для решения обсуждаемой задачи необходимо уметь решать ряд частных задач. Обсудим их.

4.1.1. Определение того факта, что некоторая решенная задача уже имеется в списке ранее введенных задач (в массиве *задачи*)

Более подробное условие задачи: "В массиве задачи записаны названия некоторого известного числа задач. Определить, имеется ли в массиве заданная задача. Если имеется, то определить также ее индекс. Известно, что общее число различных названий — не больше 11".

Решение

В дополнение к массивам задачи и кол_задач и величине всего_разных_задач (см. выше) используем в программе также величину задача — название очередной задачи.

С целью упрощения программы и ее отладки целесообразно ввести условные названия решенных задач в виде "1", "2"... и задать некоторое значение величины всего_разных_задач.

Наиболее рациональный вариант решения обсуждаемой задачи — применение оператора цикла с предусловием (условие продолжения работы оператора — сложное):

```
алг Вспомогательная_задача_1
нач лит таб задачи[1:11], лит задача, цел всего_разных_задач, i
  |Заполняем массив задачи
```

[1] Это не помешает (☺) выполнению и "ветви" **то**.

```
  всего_разных_задач := 5 |Условно
  задачи[1] := "1"
  задачи[2] := "2"
  задачи[3] := "3"
  задачи[4] := "4"
  задачи[5] := "5"
  |Вводим название очередной задачи
  ввод задача
  |Проверяем названия имеющихся задач
  i := 1
  нц пока i <= всего_разных_задач и задачи[i] <> задача
    i := i + 1
  кц
  если i <= всего_разных_задач
    то |Такая задача уже в массиве есть
      вывод нс, "Такая задач есть. Ее индекс: ", i
    иначе
      вывод нс, "Такой задачи нет"
  все
кон
```

Примечание

При отладке целесообразно проверить работу программы для случаев, когда значение величины `всего_разных_задач` равно 0 и 11.

4.1.2. Заполнение массива *задачи* неповторяющимися значениями

Более подробное условие задачи: "Известны названия *N* задач, среди которых есть повторяющиеся названия. Заполнить массив `задачи` неповторяющимися названиями. Известно, что общее число различных задач — не больше 11".

Решение

Здесь для решения сначала для каждого очередного названия задачи необходимо определить (как в предыдущей задаче), имеется ли оно в массиве. Если нет — то:

1. Увеличить общее число различных названий `всего_разных_задач` на 1.
2. Записать это название в массив (в "новый" элемент).

Соответствующая программа:

```
алг Вспомогательная_задача_2
нач цел N, всего_разных_задач, i, j, лит таб задачи[1:11], лит задача
  |Ввод значения N
  ввод N
  всего_разных_задач := 0
  |Вводим N названий
  |и заполняем массив задачи различными названиями
  нц для i от 1 до N
```

```
    ввод задача
    |Определяем, имеется ли такая задача в массиве
    j := 1
    нц пока j <= всего_разных_задач и задачи[j] <> задача
      j := j + 1
    кц
    если j > всего_разных_задач
      то |Встретилась новая задача
        |Увеличиваем значение всего_разных_задач
        всего_разных_задач := всего_разных_задач + 1
        |Записываем новую задачу в массив
        задачи[всего_разных_задач] := задача
    все
  кц
  |Выводим названия задач из массива
  нц для i от 1 до всего_разных_задач
    вывод задачи[i], " "
  кц
кон
```

Новое название можно также записать в массив следующим образом:

```
задачи[j] := задача
```

4.1.3. Заполнение массива *задачи* неповторяющимися значениями и определение "встречаемости" (количества вхождений) каждой задачи

Более подробное условие задачи: "Известны названия *N* задач, среди которых есть повторяющиеся названия. Заполнить массив задачи неповторяющимися названиями, а также определить частоту встречаемости каждого названия. Известно, что общее число различных задач — не больше 11".

Решение

В дополнение к *задаче 4.1.2* необходимо использовать массив кол_задач и для "новой" задачи записать в соответствующий элемент этого массива значение 1, а для имеющейся — увеличить ее счетчик на 1:

```
алг Вспомогательная_задача_3
нач цел N, всего_разных_задач, i, j, лит таб задачи[1:11],
    цел таб кол_задач[1:11], лит задача
  |Обнуление элементов массива кол_задач
  нц для i от 1 до 11
    кол_задач[i] := 0
  кц
  |Ввод значения N
  ввод N
  |Вводим N названий и заполняем массив задачи,
  |подсчитывая также число вхождений каждой задачи
```

```
  нц для i от 1 до N
    ввод задача
    |Определяем, имеется ли такая задача в массиве задачи
    ... (см. выше)
    если j <= всего_разных_задач
      то  |Такая задача уже в массиве есть
        |Увеличиваем ее счетчик
        кол_задач[j] := кол_задач[j] + 1
      иначе |Встретилась новая задача
        |Увеличиваем значение всего_разных_задач
        всего_разных_задач := всего_разных_задач + 1
        |Записываем новую задачу в массив задачи
        задачи[всего_разных_задач] := задача
        |Число вхождений этой задачи пока равно 1
        кол_задач[всего_разных_задач] := 1
    все
  кц
  |Выводим названия задач из массива задачи
  |и соответствующее число вхождений из массива кол_задач
  нц для i от 1 до всего_разных_задач
    вывод задачи[i], " ", кол_задач[i]
  кц
кон
```

4.1.4. Сортировка массива *кол_задач* в порядке невозрастания (и соответственно ей — изменение массива *задачи*)

Применим для решения этой задачи один из методов сортировки — сортировку обменом ("пузырьковую" сортировку). Различные варианты этого метода описаны в *приложении 4*.

Используем при сортировке массивов кол_задач и задачи вариант, который в *приложении 4* назван "более запоминающимся":

```
нц для i от 1 до всего_разных_задач - 1
  нц для лев от 1 до всего_разных_задач - i
    если кол_задач[лев] < кол_задач[лев + 1]
      то |Проводим обмен элементов массива кол_задач
        всп := кол_задач[лев]
        кол_задач[лев] := кол_задач[лев + 1]
        кол_задач[лев + 1] := всп
        |и соответствующих элементов массива задачи
        всп_задача := задача[лев]
        задача[лев] := задача[лев + 1]
        задача[лев + 1] := всп_задача
    все
  кц
кц
```

где лев — индекс левого элемента в паре сравниваемых; всп и всп_задача — вспомогательные переменные соответственно числового и строкового типов.

После решения частных задач обсудим также вопрос о выводе требуемых в условии "основной" задачи результатов.

Итак, мы отсортировали массив кол_задач в порядке невозрастания и изменили соответственно структуру массива задачи. После этого вывод результатов согласно требованиям условия может быть оформлен в виде:

```
если всего_разных_задач < 3
  то
    |Выводим все задачи и их количество
    нц для i от 1 до всего_разных_задач
      вывод нс, задачи[i], " ", кол_задач[i]
    кц
  иначе
    |Выводим две задачи с наибольшим числом вхождений
    нц для i от 1 до 2
      вывод нс, задачи[i], " ", кол_задач[i]
    кц
    |и задачи с той же частотой встречаемости,
    |что и третья по частоте задача
    i := 3
    нц пока i <= всего_разных_задач и кол_задач[i] = кол_задач[3]
      вывод нс, задачи[i], " ", кол_задач[i]
      i := i + 1
    кц
все
```

Обратим внимание на сложное условие в последнем операторе цикла.

Можно упростить фрагмент, связанный с выводом результатов, следующим образом:

```
если всего_разных_задач < 3
  то
    j := всего_разных_задач
  иначе
    j := 3
все
i := 1
нц пока i <= всего_разных_задач и кол_задач[i] >= кол_задач[j]
  вывод нс, задачи[i], " ", кол_задач[i]
  i := i + 1
кц
```

Именно так оформлен этот фрагмент в решении задачи в демонстрационном варианте.

Полностью "собрать" программу предлагаем читателям самостоятельно (как и программы решения всех других задач С4, обсуждаемых в книге далее)[1].

Здесь же перечислим ее основные части:

1. Обнуление элементов массива `кол_задач` (в школьном алгоритмическом языке это является обязательным).
2. Ввод значения *N*.
3. Ввод *N* названий задач и заполнение ими массива `задачи` с подсчетом числа вхождений каждой задачи.
4. Сортировка массива `кол_задач` в порядке невозрастания и, соответственно ей, — изменение массива `задачи`.
5. Вывод требуемых результатов.

Заметим также, что в демонстрационном варианте ЕГЭ в программе на языке Паскаль величина `s` (название очередной задачи) и массив `Names` (с названием задач) описаны как:

```
s: string;
Names: array [1..11] of string;
```

в то время как согласно условию длина строки с названием задачи не превышает 100 символов.

Кроме того, при упорядочивании массивов в качестве величины, вспомогательной для обмена элементов массива `Names`, используется все та же величина `s`.

4.2. Задача из демонстрационного варианта экзамена 2010 года

Условие

На автозаправочных станциях (АЗС) продается бензин с маркировкой 92, 95 и 98. В городе N был проведен мониторинг цены бензина на различных АЗС.

Напишите эффективную по времени работы и по используемой памяти программу (укажите используемую версию языка программирования, например Borland Pascal 7.0), которая будет определять для каждого вида бензина, сколько АЗС продают его дешевле всего. На вход программе в первой строке подается число данных о стоимости бензина. В каждой из последующих *N* строк находится информация в следующем формате:

`<Компания> <Улица> <Марка> <Цена>`

где `<Компания>` — строка, состоящая не более чем из 20 символов без пробелов; `<Улица>` — строка, состоящая не более чем из 20 символов без пробелов; `<Марка>` — одно из чисел 92, 95 или 98; `<Цена>` — целое число в диапазоне от 1000 до 3000,

[1] Программы решения задач на школьном алгоритмическом языке приведены на сайте издательства.

обозначающее стоимость одного литра бензина в копейках. <Компания> и <Улица>, <Улица> и <Марка>, а также <Марка> и <Цена> разделены ровно одним пробелом. Пример входной строки:

```
Синойл Цветочная 95 2250
```

Программа должна выводить через пробел 3 числа — количество АЗС, продающих дешевле всего 92-й, 95-й и 98-й бензин соответственно. Если бензин какой-то марки нигде не продавался, то следует вывести 0.

Пример выходных данных:

```
12 1 0
```

Анализ решения

В целом можно сказать, что обсуждаемая задача представляет собой задачу нахождения количества минимальных значений цены для каждой из трех марок бензина.

Общая структура программы решения задачи такая:

```
1. Ввод значения N
2. Цикл для i от 1 до N |Ввод и обработка входных строк
   2.1. Ввод информации об i-й АЗС
   2.2. Выделение марки бензина и ее цены на i-й АЗС (такая задача
        рассмотрена в главе 2 — см. разд. 2.1.7[1])
   2.3. Подсчет количества минимальных значений (ТЗ 2.4.3, ТЗ 2.4.4)
конец цикла
3. Вывод ответа
```

Сначала рассмотрим вариант решения задачи без применения массивов.

В программе на школьном алгоритмическом языке используем следующие основные величины:

- *N* — число данных о стоимости бензина (см. условие задачи);
- *строка* — величина строкового типа, о которой шла речь в условии;
- *марка* — марка бензина (величина целого типа, равная 92, 95 или 98);
- *цена* — его цена (величина целого типа со значениями от 1000 до 3000);
- *мин92*, *мин95*, *мин98* — минимальная цена соответственно на 92-й, 95-й и 98-й бензин;
- *кол92*, *кол95*, *кол98* — искомые значения (количество АЗС, продающих дешевле всего 92-й, 95-й и 98-й бензин соответственно).

С их использованием фрагмент, относящийся к этапу 2.3, будет выглядеть так:

```
|Сравниваем полученную цену с минимальной ценой
|на соответствующую марку бензина
выбор
```

[1] В дальнейшем ссылки на типовые задачи программирования в *главах 1* и *2* будем обозначать в виде, аналогичном следующему: ТЗ 2.1.7.

```
при марка = 92:
  если цена < мин92
    то
      мин92 := цена
      кол92 := 1
    иначе
      если цена = мин92
        то
          кол92 := кол92 + 1
    все
  все
при марка = 95:
  ...
при марка = 98:
  ...
```

а фрагмент, связанный с выводом ответа:

```
вывод нс, кол92, " " кол95, " ", кол98
|Если бензина какой-то марки не было,
|соответствующее значение равно нулю
```

Можно также вместо шести величин мин92, мин95, мин98, кол92, кол95 и кол98 использовать два массива с индексами от 92 до 98:

1. С именем мин — для хранения минимальных цен на бензин разной марки.
2. С именем кол — для хранения искомых значений количества АЗС, продающих дешевле всего 92-й, 95-й и 98-й бензин соответственно.

При их использовании изменения в программе коснутся:

1. Описания переменных величин:

```
цел таб кол[92:98], мин[92:98]
```

2. Задания начальных значений элементов массивов:

```
нц для i от 92 до 98
 кол[i] := 0
 мин[i] := 3001
кц
```

3. Сравнения выделенной в каждой строке цены с минимальной ценой на соответствующую марку бензина для рассмотренных ранее АЗС:

```
выбор
  при марка = 92:
    если цена < мин[92]
      то
        мин[92] := цена
        кол[92] := 1
      иначе
        если цена = мин[92]
```

```
            то
              кол[92] := кол[92] + 1
          все
      все
    при марка = 95:
      если цена < мин[95]
        то
          мин[95] := цена
          кол[95] := 1
        иначе
          если цена = мин[95]
            то
              кол[95] := кол[95] + 1
          все
      все
    при марка = 98:
    ...
```

4. Вывода ответа:

```
вывод нс, кол[92], " ", кол[95], " ", кол[98]
```

4.3. Задача из демонстрационного варианта экзамена 2009 года

Условие

На вход программе подаются сведения о номерах школ учащихся, участвовавших в олимпиаде. В первой строке сообщается количество учащихся *N*, каждая из следующих *N* строк имеет формат:

```
<Фамилия> <Инициалы> <номер школы>
```

где <Фамилия> — строка, состоящая не более чем из 20 символов; <Инициалы> — строка, состоящая из четырех символов (буква, точка, буква, точка); <номер школы> — не более чем двузначный номер. <Фамилия> и <Инициалы>, а также <Инициалы> и <номер школы> разделены одним пробелом.

Пример входной строки:

```
Иванов П.С. 57
```

Требуется написать как можно более эффективную программу (укажите используемую версию языка программирования, например Borland Pascal 7.0), которая будет выводить на экран информацию, из каких школ было меньше всего участников (но из этих школ был хотя бы один участник).

Анализ решения

Ясно, что необходимо узнать количество участников олимпиады от каждой школы. Для хранения информации об этом применим массив с именем всего с индексами от 1 до 99. Заполнение этого массива следует провести во время ввода исходных

данных после выделения номера школы (имя величины — школа), в которой учится очередной участник.

Методика выделения числа после двух слов рассмотрена при обсуждении *ТЗ 2.1.6*, а задача заполнения массива всего — это аналог *ТЗ 2.2.5*.

Можно также сразу выделить номер школы (по его "местонахождению" в строке), не пропуская фамилию и инициалы ученика:

```
если строка[длин(строка) – 1] = " "
  то |Номер школы – однозначный
    школа := лит_в_цел(строка[длин(строка)], успех)
  иначе |Двузначный
    школа := лит_в_цел(строка[длин(строка) – 1: длин(строка)], успех)
все
```

Узнав номер школы, увеличим ее "счетчик" в массиве всего:

```
всего[школа] := всего[школа] + 1
```

Поскольку нужна информация о том, из каких школ было меньше всего участников (а в ходе одного прохода значение минимального элемента массива меняется), то эта информация определяется за два прохода по массиву всего.

После заполнения массива всего минимальное количество участников из одной школы (с учетом того, что должен быть хотя бы один участник) находится так *(ТЗ 1.9.1, ТЗ 1.9.2)*:

```
мин := N
нц для i от 1 до 99
 |Учитываем только школы, из которых был хотя бы один участник
 если всего[i] > 0
  то
    если всего[i] < мин
     то
       мин := всего[i]
    все
 все
кц
```

а фрагмент, связанный с выводом ответа, имеет вид *(ТЗ 1.11.6)*:

```
нц для i от 1 до 99
 если всего[i] = мин
  то
   вывод нс, i
 все
кц
```

Общая структура программы решения задачи:

```
1. Заполнение массива всего нулевыми значениями
2. Ввод значения N
```

```
3. Цикл для i от 1 до N |Ввод и обработка входных строк
 3.1. Ввод информации об i-м ученике
 3.2. Выделение номера школы (ТЗ 2.1.6)
 3.3. Учет этой школы в массиве всего (ТЗ 2.2.5)
конец цикла
4. Определение минимального элемента массива всего (см. выше)
5. Поиск и вывод необходимых номеров школ (см. выше)
```

4.4. Задача из демонстрационного варианта экзамена 2008 года

Условие

На вход программе подаются сведения о сдаче экзаменов учениками 9-х классов некоторой средней школы. В первой строке подается число учеников *N*, которое не меньше 10 и не превосходит 100. В каждой из последующих *N* строк находится информация в следующем формате:

```
<Фамилия> <Имя> <оценки>
```

где <Фамилия> — строка, состоящая не более чем из 20 символов; <Имя> — строка, состоящая не более чем из 10 символов; <оценки> — через пробел три целых числа, соответствующие оценкам по пятибалльной системе. <Фамилия> и <Имя>, а также <Имя> и <оценки> разделены ровно одним пробелом.

Пример входной строки:

```
Иванов Петр 4 5 3
```

Напишите как можно более эффективную программу (укажите используемую версию языка программирования, например Borland Pascal 7.0), которая будет выводить фамилии и имена трех худших по среднему баллу учеников. Если среди остальных есть ученики, набравшие тот же средний балл, что и один из трех худших, то следует вывести и их фамилии и имена.

Анализ решения

Прежде всего обратим внимание на то, что так как все ученики сдавали одинаковое число экзаменов, то их сравнение можно проводить не по среднему баллу, а по сумме баллов, набранных каждым учеником.

Значит, для вывода искомых фамилий и имен нужно иметь два массива[1] из 100 элементов каждый, в одном из которых (с именем фам_имя) будут записаны фамилии и имена, в другом (сумма) — суммы баллов, набранных каждым учеником.

Обсудим методы заполнения этих двух массивов.

Массив фам_имя

Выделение из входной строки фамилии ученика и его имени как единого целого — *ТЗ 2.1.3*.

[1] В программе на языке Паскаль можно использовать один массив с данными типа запись. Вариант с использованием массива записей приведен в [3].

После выделения соответствующей величины (пусть ее имя — фиу) ее сразу же следует записать в массив фам_имя:

```
нц для i от 1 до N |Ввод и обработка входных строк
  |Формируем величину фиу
  ...
  |и записываем ее в массив
  фам_имя[i] := фиу
  |Выделяем оценки и записываем их сумму в массив сумма
  ...
кц
```

Обратим внимание, что в данном случае впервые для всех рассмотренных задач входная информация (точнее, ее часть — фамилия и имя ученика) записывается в массив.

В программе на языке Паскаль можно обойтись без использования аналога величины фио — добавлять в массив каждый считанный с помощью оператора read один символ *(см. ТЗ 2.1.1)*:

```
for i := 1 to N do
  begin
    fam_ima := '';
    repeat
      read(c);
      fam_ima := fam_ima + c
    until c = ' '; {Считана и записана фамилия}
    repeat
      read(c);
      p[i].name := fam_ima + c
    until c = ' '; {Добавлено имя}
  ...
```

Обратите внимание — присваивание каждому элементу массива (поле name) начального ("пустого") значения отдельно не проводится — это делается в цикле ввода данных.

Массив сумма

Для расчета суммы баллов необходимо выделить каждую из трех оценок. Такая задача — типовая *(ТЗ 2.1.8)*. После получения трех оценок i-го ученика их сумма записывается в соответствующий массив:

```
сумма[i] := оценка1 + оценка2 + оценка3
```

Так как получаемые оценки должны быть просуммированы, то вместо трех переменных величин можно применить оператор цикла с параметром, в котором будет использоваться только одна величина — очередная оценка:

```
нц для i от 1 до N |Ввод и обработка входных строк
  |Формируем величину фиу
  ... см. выше
  |Пропускаем 2-й пробел
  номер_симв := номер_симв + 1
```

```
  |Получаем сумму баллов и записываем ее в массив всего
  сумма[i] := 0
  |Получаем две первых оценки и учитываем каждую в сумме баллов
  нц для i от 1 до 2
    оценка := лит_в_цел(строка[номер_симв], успех)
    суммавсего[i] := сумма[i] + оценка
    |Пропускаем пробел после оценки
    номер_симв := номер_симв + 1
  кц
  |Получаем третью оценку
  оценка := лит_в_цел(строка[номер_симв], успех)
  |и также учитываем ее в сумме баллов
  сумма[i] := сумма[i] + оценка
кц
```

Видно, что в приведенном фрагменте обнуление массива сумма проводится в цикле обработки входных строк (в случае применения трех величин — оценок обнуление можно не проводить).

Можно также вообще величину оценка не использовать, а учитывать в сумме значение функции лит_в_цел(строка[номер_симв], успех).

В программе на языке Паскаль все три оценки можно просуммировать в одном цикле:

```
{Получаем три оценки и учитываем каждую в сумме баллов}
for j := 1 to 3 do
  begin
    read(ozenka);  {см. ТЗ 2.1.6}
    summa[i] := summa[i] + ozenka
  end;
readln;
```

Итак, массивы фам_имя и сумма будут заполнены. Как вывести требуемую информацию?

Анализ условия показывает, что отбор учащихся для вывода можно проводить по следующему условию: сумма баллов должна быть не больше (!) суммы баллов, набранных учеником, который является "третьим из худших". Значит, возникла задача нахождения значения "третьего минимума" в массиве сумма. Это — *ТЗ 2.4.8*.

После нахождения третьего минимума фрагмент, связанный с выводом ответа, должен быть оформлен так *(см. ТЗ 2.3.2)*:

```
нц для i от 1 до N
  если сумма[i] <= минимум3
    то |Выводим фамилию и имя
     вывод нс, фам_имя[i] |Каждые – на отдельной строке[1]
```

[1] В условии это не оговорено.

```
  все
кц
```

где минимум3 — третий минимум в массиве сумма.

Общая структура программы решения задачи такая:

```
1. Ввод значения N
2. Цикл для i от 1 до N |Ввод и обработка входных строк
   2.1. Ввод информации об очередном ученике
   2.2. Выделение фамилии и имени как единого целого (ТЗ 2.1.3)
   2.3. Запись фамилии и имени в массив фам_имя
   2.4. Выделение трех оценок (ТЗ 2.1.8)
   2.5. Запись их суммы в массив сумма
конец цикла
3. Определение "третьего минимума" в массиве сумма (ТЗ 2.4.7, ТЗ 2.4.8)
4. Вывод ответа (фамилий и имен из массива фам_имя — см. выше)
```

4.5. Задача из демонстрационного варианта экзамена 2007 года

Условие

На вход программе подаются сведения о сдаче экзаменов учениками 9-х классов некоторой средней школы. В первой строке сообщается количество учеников *N*, которое не меньше 10, но не превосходит 100, каждая из следующих *N* строк имеет следующий формат:

```
<Фамилия> <Имя> <оценки>
```

где <Фамилия> — строка, состоящая не более чем из 20 символов; <Имя> — строка, состоящая не более чем из 15 символов; <оценки> — через пробел три целых числа, соответствующие оценкам по пятибалльной системе. <Фамилия> и <Имя>, а также <Имя> и <оценки> разделены одним пробелом.

Пример входной строки:

```
Иванов Петр 4 5 4
```

Требуется написать программу, которая будет выводить на экран фамилии и имена трех лучших по среднему баллу учеников. Если среди остальных есть ученики, набравшие тот же средний балл, что и один из трех лучших, то следует вывести и их фамилии и имена. Требуемые имена и фамилии можно выводить в произвольном порядке.

Задача решается аналогично предыдущей (с учетом того, что речь идет не о трех худших, а о трех лучших по среднему баллу учениках).

> ***Примечание***
>
> Задача С4 демонстрационного варианта Единого государственного экзамена по информатике и ИКТ 2011 года имеет ряд особенностей, существенно отличающих ее от задач, рассматриваемых в данной главе. Анализ этой задачи представлен на сайте издательства.

ГЛАВА 5

Задачи С4 из книги [16]

5.1. Вариант 1

Условие

На вход программе подаются сведения о номерах школ учащихся, участвовавших в олимпиаде. В первой строке сообщается количество учащихся *N*, каждая из следующих *N* строк имеет формат:

```
<Фамилия> <Инициалы> <номер школы>
```

где <Фамилия> — строка, состоящая не более чем из 20 символов; <Инициалы> — строка, состоящая из четырех символов (буква, точка, буква, точка); <номер школы> — не более чем двузначный номер. <Фамилия> и <Инициалы>, а также <Инициалы> и <номер школы> разделены одним пробелом.

Пример входной строки:

```
Иванов П.С. 57
```

Требуется написать как можно более эффективную программу (укажите используемую версию языка программирования, например Borland Pascal 7.0), которая будет выводить на экран информацию, из какой школы было больше всего участников (таких школ может быть несколько). Также программа должна подсчитать общее количество школ, приславших больше всего участников.

Следует учитывать, что *N* >= 1000.

Анализ решения

Данная задача аналогична задаче из демоварианта ЕГЭ 2009 года *(см. разд. 4.3)*.

Здесь также необходимо узнать количество участников олимпиады от каждой школы. Для хранения информации об этом применим массив с именем `всего` с индексами от 1 до 99. Заполнение этого массива следует провести во время ввода исходных данных после выделения номера школы, в которой учится очередной участник.

Методика выделения числа после двух слов рассмотрена при обсуждении *ТЗ 2.1.6*, а задача заполнения массива `всего` — это аналог *ТЗ 2.2.5*.

Можно также сразу выделить номер школы (по его "местонахождению" в строке), не пропуская фамилию и инициалы ученика:

```
если строка[длин(строка) — 1] = " "
  то |Номер школы – однозначный
    школа := лит_в_цел(строка[длин(строка)], успех)
  иначе |Двузначный
    школа := лит_в_цел(строка[длин(строка) – 1: длин(строка)], успех)
все
```

Узнав номер школы, увеличим ее "счетчик" в массиве `всего`:

```
всего[школа] := всего[школа] + 1
```

Далее, если бы требовалось подсчитать общее количество школ, приславших больше всего участников, то это можно было сделать за один проход по массиву `всего` *(см. ТЗ 2.4.3)*. Но поскольку нужна также информация о том, из какой школы было больше всего участников (а в ходе одного прохода соответствующие данные изменяются), то значения искомых величин надо определять за два прохода:

```
...
|1. Ищем максимальный элемент массива всего (см. ТЗ 1.9.1)
макс := всего[1]
нц для i от 2 до 99
  если всего[i] > макс
    то
      макс := всего[i]
    все
кц
|2. Выводим номера школ с максимальным числом участников,
|одновременно подсчитывая их количество k (см. типовую задачу 2.4.3),
k := 0
нц для i от 1 до 99
  если всего[i] = макс
    то
      вывод i, " "
        k := k + 1
    все
кц
|которое затем выводим на экран
вывод нс, "Количество школ, приславших наибольшее число участников ", k
```

Итак, общая структура программы решения обсуждаемой задачи такая:

```
1. Заполнение массива всего нулевыми значениями
2. Ввод значения N
3. Цикл для i от 1 до N |Ввод и обработка входных строк
   3.1. Ввод информации об i-м ученике
   3.2. Выделение номера школы (ТЗ 2.1.6)
   3.3. Учет этой школы в массиве всего (ТЗ 2.2.5)
конец цикла
```

```
4. Определение максимального элемента массива всего (ТЗ 1.9.1)
5. Поиск и вывод необходимых номеров школ (с подсчетом значений k)
6. Вывод значения k
```

5.2. Вариант 2

Условие

На вход программе подаются сведения о номерах школ учащихся, участвовавших в олимпиаде. В первой строке сообщается количество учащихся *N*, каждая из следующих *N* строк имеет формат:

```
<Фамилия> <Инициалы> <номер школы>
```

где <Фамилия> — строка, состоящая не более чем из 20 символов; <Инициалы> — строка, состоящая из четырех символов (буква, точка, буква, точка); <номер школы> — не более чем двузначный номер. <Фамилия> и <Инициалы>, а также <Инициалы> и <номер школы> разделены одним пробелом.

Пример входной строки:

```
Иванов П.С. 57
```

Требуется написать как можно более эффективную программу (укажите используемую версию языка программирования, например Borland Pascal 7.0), которая определяет среднее количество участников олимпиады из одной школы.

Следует учитывать, что *N* >= 1000.

Анализ решения

Кроме того, обращаем внимание на то, что найти требуется среднее количество участников от одной, а не от каждой школы. Это значение должно рассчитываться как N/кол_школ, где N — общее количество участников олимпиады, кол_школ — число школ, ученики которых принимали участие в олимпиаде (имеется в виду, что от некоторых школ участников не было[1]).

В свою очередь, для расчета значения величины кол_школ необходимо знать количество участников от каждой школы. Для хранения информации об этом применим массив с именем всего с индексами от 1 до 99. Заполнение этого массива проводится так, как описано применительно к *ТЗ 2.2.5* (до этого предварительно надо выделить номер школы очередного участника — *см. ТЗ 2.1.6*).

Общая структура программы решения задачи такая:

```
1. Заполнение массива всего нулевыми значениями
2. Ввод значения N
3. Цикл для i от 1 до N |Ввод и обработка входных строк
   3.1. Ввод информации об i-м ученике
```

[1] Это следует из анализа решения, представленного в [16].

```
   3.2. Выделение номера школы (ТЗ 2.1.6)
   3.3. Учет данного ученика в массиве всего (ТЗ 2.2.5)
конец цикла
4. Определение числа школ, ученики которых принимали участие в олимпиаде
(ТЗ 2.3.1)
5. Вывод значения ответа
```

В заключение обратим внимание на то, что ни в приведенной структуре программы, ни в программах, представленных на сайте издательства, искомое значение среднего количества участников от одной школы предварительно не рассчитывалось (выражение для его расчета включено в вывод ответа). Так сделано по аналогии с решением в [16], хотя мы считаем это методически неправильным.

Кроме того, искомое значение будет нецелым (а согласно программе, приведенной в [16], — будет выведено в экспоненциальной форме).

5.3. Вариант 3

Условие

Во входном файле meteo.dat 365 строк, которые содержат информацию о среднесуточной температуре всех дней 2003 года. Формат каждой из строк следующий: сначала записана дата в виде `dd.mm` (на запись номера дня и номера месяца в числовом формате отводится строго по два символа, день от месяца отделен точкой), затем через пробел записано значение температуры — число со знаком плюс или минус, с точностью до одной цифры после десятичной точки. Данная информация отсортирована по значению температуры, т. е. хронологический порядок нарушен. Требуется написать программу на языке Паскаль или Бейсик, которая будет выводить на экран информацию о месяце (месяцах) с максимальной среднемесячной температурой. В первой строке вывести количество месяцев с максимальной среднемесячной температурой, во второй строке — номера месяцев через запятую, в третьей строке — значение максимальной среднемесячной температуры.

Анализ решения

Прежде всего, обратим внимание на то, что, хотя в условии задачи фигурируют фраза "входной файл" и имя файла, ввод исходных данных осуществляется путем задания 365 строк с клавиатуры.

Ключевым для решения задачи является массив из 12 элементов с суммой среднедневных температур для дней каждого месяца. Пусть имя такого массива — `всего_за_месяц`. Для заполнения этого массива необходимо из каждой строки (информации о каждом дне года) выделить номер месяца (`месяц`) и соответствующее значение среднедневной температуры (`темп`). Так как "местоположение" этих значений в строке известно, то их можно получить, используя "вырезки"[1]:

[1] В программе на языке Паскаль для этого можно применить функцию `Copy`.

```
месяц := лит_в_цел(строка[4:5], успех)
темп := лит_в_вещ(строка[7:длин(строка)], успех)
```

где строка — заданная строка.

После этого значение следует учесть в соответствующем элементе массива всего_за_месяц *(см. ТЗ 2.5.1)*:

```
всего_за_месяц[месяц] := всего_за_месяц[месяц] + темп
```

Далее нужно оперировать средними значениями температуры для каждого месяца. Но дополнительный массив для таких значений можно не применять — для определения максимальной среднемесячной температуры и для отбора и вывода на экран необходимых по условию значений можно использовать данные массива всего_за_месяц.

Так как найти требуется не только значение максимальной среднемесячной температуры и количество месяцев с такой температурой (что можно было сделать за один проход — *см. ТЗ 2.4.3*), но и номера месяцев, то придется выполнить два прохода по массиву:

1. Для поиска максимальной среднемесячной температуры макс_сред и количества таких значений кол_макс *(см. ТЗ 2.4.3)*.
2. Для вывода ответа:

```
|Количество месяцев с максимальной среднемесячной температурой
вывод нс, кол_макс
|Номера месяцев через запятую
нц для i от 1 до 12
  если abs(всего_за_месяц[i]/дней_в_месяце[i] − макс_сред) < 0.0001
    то
      вывод нс, i, ","   |Последняя запятая будет лишней
  все
кц
|Максимальная среднемесячная температура
вывод нс, макс_сред
```

Видно (и ясно), что для расчета среднемесячной температуры для каждого месяца надо иметь данные о количестве дней в том или ином месяце. Такие данные хранятся в массиве дней_в_месяце, который в программе на школьном алгоритмическом языке может быть заполнен так:

```
|Январь, март, май, июль
нц для i от 1 до 7 шаг 2
  дней_в_месяце[i] := 31
кц
|Август, октябрь, декабрь
нц для i от 8 до 12 шаг 2
  дней_в_месяце[i] := 31
кц
дней_в_месяце[2] := 28 |Февраль
```

```
нц для i от 4 до 6 шаг 2
  дней_в_месяце[i] := 30 |апрель и июнь
  дней_в_месяце[i + 5] := 30 |сентябрь и ноябрь
кц
```

В программе на языке Паскаль можно применить аналогичный массив как константу.

Кроме того, обращаем внимание на то, что при отборе для вывода на экран месяцев с максимальной среднемесячной температурой, являющейся величиной вещественного типа, должна быть учтена погрешность вычислений с такими величинами. Это же должно учитываться при подсчете величины кол_макс.

Общая структура программы решения задачи такая:

```
1. Заполнение массива дней_в_месяце
2. Обнуление массива всего_за_месяц
3. Цикл для i от 1 до 365 |Ввод и обработка входных строк
   3.1. Ввод информации об i-м дне
   3.2. Выделение номера месяца и температуры (см. выше)
   3.3. Учет температуры в массиве всего_за_месяц (ТЗ 2.5.1)
конец цикла
4. Поиск максимальной среднемесячной температуры
   и подсчет значения кол_макс (ТЗ 2.4.3)
5. Вывод ответа (см. выше).
```

5.4. Вариант 4

Условие

На вход программе подаются сведения о сдаче экзаменов учениками 9-х классов некоторой средней школы. В первой строке сообщается количество учеников *N*, которое не меньше 10, но не превосходит 100, каждая из следующих *N* строк имеет следующий формат:

```
<Фамилия> <Имя> <оценки>
```

где <Фамилия> — строка, состоящая не более чем из 20 символов; <Имя> — строка, состоящая не более чем из 15 символов; <оценки> — через пробел три целых числа, соответствующие оценкам по пятибалльной системе. <Фамилия> и <Имя>, а также <Имя> и <оценки> разделены ровно одним пробелом.

Пример входной строки:

```
Иванов Петр 4 5 4
```

Требуется написать программу, которая будет выводить на экран фамилии и имена учащихся, сдавших экзамены только на 4 и 5. Требуемые имена и фамилии можно выводить в произвольном порядке. В случае, если таких учащихся нет, сообщите об этом.

Анализ решения — в *разд. 5.4.1* и *5.4.2*.

5.4.1. Первый способ

Можно завести два массива из 100 элементов каждый, один — с фамилиями и именами учащихся, второй — с данным логического типа (или с величинами, принимающими значения 0 и 1), определяющими, сдал ли тот или иной ученик экзамены только на 4 и 5.

Заполнение первого массива (фам_имя) проводится после выделения фамилии и имени i-го ученика как единого целого *(см. ТЗ 2.1.3)*, второго (четыре_и_пять) — так:

```
кол45 := 0
нц для i от 1 до N
  |Выделяем фамилию и имя, а также три оценки
  ...
  если оценка1 >= 4 и оценка2 >= 4 и оценка3 >= 4
    то
      четыре_и_пять[i] := да
      кол45 := кол45 + 1
  все
кц
```

Видно, что одновременно подсчитывается количество учеников, сдавших экзамены только на 4 и 5 (кол45).

Оценки можно выделить так, как это делалось применительно к *ТЗ 2.1.8* или непосредственно обращаясь к нужным символам строки:

```
оценка3 := лит_в цел(строка[длин(строка)])
оценка2 := лит_в цел(строка[длин(строка) - 2])
оценка1 := лит_в цел(строка[длин(строка) - 4])
```

После заполнения массивов ответ можно вывести так:

```
если кол45 = 0
  то
    вывод нс, "Нет таких учеников"
  иначе
    вывод нс
    нц для i от 1 до 100
      если четыре_и_пять[i]
        то
          вывод фам_имя[i], " "
      все
    кц
все
```

Обратим внимание на то, что если общее число учеников будет меньше 100, то для "отсутствующих" учащихся соответствующие элементы массива фам_имя будут заполнены "пустыми" значениями, а массива четыре_и_пять — значениями ложь (нет, false и т. п.).

5.4.2. Второй способ

Этот способ отличается тем, что используется один массив фам_имя, который заполняется только "нужными" учениками:

```
кол45 := 0
нц для i от 1 до N
  |Выделяем фамилию и имя ученика, а также три его оценки
  ...
  если оценка1 >= 4 и оценка2 >= 4 и оценка3 >= 4
    то
      |Встретился очередной ученик с оценками не ниже 4
      кол45 := кол45 + 1
      |Записываем его фамилию и имя в массив в очередной элемент
      фам_имя[кол45] := фиу
  все
кц
```

где фиу — фамилия и имя очередного ученика как единая величина.

Далее выводим все "непустые" элементы массива фам_имя либо сообщение:

```
если кол45 = 0
  то
    вывод нс, "Нет таких учеников"
  иначе
    нц для i от 1 до кол45
      вывод нс, фам_имя[i]
    кц
все
```

Общая структура программы решения задачи по второму способу:

```
1. Заполнение массива фам_имя "пустыми" значениями
2. Цикл для i от 1 до N |Ввод и обработка входных строк
   2.1. Ввод информации об i-м ученике
   2.2. Выделение фамилии и имени как единой целого (ТЗ 2.1.3)
   2.3. Выделение трех оценок (см. ТЗ 2.1.8 или выше)
   2.4. Запись (при необходимости) i-го ученика в массив фам_имя
        с одновременным подсчетом значения кол45 (см. выше)
конец цикла
3. Вывод ответа (см. выше)
```

Можно также выделение имени и фамилии проводить не для всех учащихся, а только для имеющих оценки не ниже 4:

```
1. Заполнение массива фам_имя "пустыми" значениями
2. Ввод значения N
3. Цикл для i от 1 до N |Ввод и обработка входных строк
   3.1. Ввод информации об i-м ученике
   3.2. Определение оценок
```

```
   3.3.  Если все оценки не ниже 4, то
     3.3.1. Выделение фамилии и имени ученика как единого целого
     3.3.2. Запись i-го ученика в массив фам_имя
            с одновременным подсчетом значения кол45
конец цикла
4. Вывод ответа
```

5.5. Вариант 5

Условие

На вход программе подаются 365 строк, которые содержат информацию о среднесуточной температуре всех дней 2007 года. Формат каждой из строк следующий: сначала записана дата в виде `dd.mm` (на запись номера дня и номера месяца в числовом формате отводится строго по два символа, день от месяца отделен точкой), затем через пробел (для Бейсика — через запятую) записано значение температуры — число со знаком плюс или минус, с точностью до одной цифры после десятичной точки. Данная информация отсортирована по значению температуры, т. е. хронологический порядок нарушен. Требуется написать эффективную программу на языке Паскаль или Бейсик, которая будет выводить на экран информацию о месяце (месяцах) с максимальной среднемесячной температурой. Найденные максимальные значения следует выводить в отдельной строке для каждого месяца в виде: номер месяца, значение среднемесячной температуры, округленное до одной цифры после десятичной точки.

Анализ решения

Задача во многом аналогична задаче варианта 3 *(см. разд. 5.3)*. Отличие — в выводе ответа после заполнения массива с суммами дневных температур для каждого месяца. Кроме того, не потребуется определение числа месяцев с максимальной среднемесячной температурой.

Как и в задаче варианта 3, придется выполнить два прохода по массиву:

1. Для поиска максимальной среднемесячной температуры `макс_сред` *(см. ТЗ 1.9.1)*.
2. Для вывода ответа:

```
нц для i от 1 до 12
  если abs(всего_за_месяц[i]/дней_в_месяце[i] – макс_сред) < 0.0001
    то
      вывод нс, i, " ", округл_макс_сред
  все
кц
```

где `округл_макс_сред` — максимальное значение среднемесячной температуры, округленное до одной цифры после десятичной точки. Поскольку в школьном алгоритмическом языке форматированный вывод с заданной точностью пока невозможен, то это значение может быть рассчитано (до вывода на экран) следующим образом:

```
если 10 * макс_сред — int(10 * макс_сред) < 0.5
  то
    округл_макс_сред := int(10 * макс_сред)/10
  иначе
    округл_макс_сред := (int(10 * макс_сред) + 1)/10
все
```

Обратим внимание на то, что приведенное в условии указание о вводе в программе на языке Бейсик значения даты и температуры через запятую говорит о том, что при этом вводятся значения двух отдельных величин.

Сделаем также два замечания по программе решения задачи на языке Паскаль в [16]:

1. В операторе ввода данных:

   ```
   readln(c1,c1,c1,c1,c2,t);
   ```

   три первых значения переменной `c1` не используются (для формирования номера месяца и получения значения температуры применяются остальные переменные[1]).

2. При выводе на экран максимальной среднемесячной температуры в виде:

   ```
   writeln(j, ' ',m[j]/d[j]:0:1);
   ```

   количество позиций, отводимых для вывода числа, задано равным нулю!

5.6. Вариант 7

Условие

На вход программе подаются сведения об учениках некоторой средней школы. В первой строке сообщается количество учеников *N*, каждая из следующих *N* строк имеет формат:

`<Фамилия> <Инициалы> <класс>`

где `<Фамилия>` — строка, состоящая не более чем из 20 символов; `<Имя>` — строка, состоящая не более чем из 15 символов; `<класс>` — год обучения (от 1 до 12) и заглавная буква (от "А" до "Я") без пробела. `<Фамилия>` и `<Имя>`, а также `<Имя>` и `<класс>` разделены одним пробелом. Пример входной строки:

```
Иванов Петр 10Б
```

Требуется написать программу, которая будет выводить на экран информацию о параллелях (годе обучения) с наибольшим числом учеников. Программа должна выводить на экран в первой строке количество учеников в искомых параллелях, а во второй — в порядке возрастания номера параллелей через пробел.

---

[1] Напомним, что для получения этих значений можно также использовать "вырезку" из строки (см. выше разбор решения задачи варианта 3). Правда, при этом необходимо применить переменную строкового типа.

Например:

```
100
1 7 11
```

Анализ решения

Данная задача похожа на задачу варианта 1 *(см. разд. 5.1)*. Отличие в том, что нужно выделять из строки не номер школы, а номер параллели. Это также можно сделать, зная его возможное "местонахождение" в строке, не пропуская фамилию и имя ученика:

```
если строка[длин(строка) — 2] = " "
  то |Номер параллели – однозначный
    школа := лит_в_цел(строка[длин(строка) – 1 ], успех)
  иначе |Двузначный
    школа := лит_в_цел(строка[длин(строка) – 2 : длин(строка) – 1], успех)
все
```

После заполнения массива `всего` с общим числом учеников в каждой параллели на первом проходе находится его максимальный элемент, на втором — номера элементов (параллелей) с наибольшим числом учеников.

Общая структура программы решения обсуждаемой задачи такая:

```
1. Заполнение массива всего нулевыми значениями
2. Ввод значения N
3. Цикл для i от 1 до N |Ввод и обработка входных строк
   3.1. Ввод информации об i-м ученике
   3.2. Выделение номера параллели (ТЗ 2.1.6)
   3.3. Учет параллели класса этого ученика
   в массиве всего (ТЗ 2.2.5)
конец цикла
4. Определение максимального элемента массива всего (ТЗ 1.9.1)
5. Вывод найденного значения
6. Поиск и вывод номеров необходимых параллелей (ТЗ 1.11.6)
```

5.7. Вариант 10

Условие

На автозаправочных станциях (АЗС) продается бензин с маркировкой 92, 95 и 98. В городе N был проведен мониторинг цены бензина на различных АЗС.

Напишите эффективную, в том числе и по используемой памяти программу (укажите используемую версию языка программирования, например Borland Pascal 7.0), которая будет определять для бензина, с маркировкой 92, на какой АЗС его продают по второй по минимальности цене (считается, что самой низкой цене потребители не доверяют), а если таких АЗС несколько, то выдается только количество таких АЗС. Если все АЗС, у которых 92-й бензин есть, продают его по одной и той же цене, то эта цена является искомой и выдается либо число таких АЗС, когда их

несколько, либо конкретная АЗС, если она одна. Гарантируется, что хотя бы одна АЗС 92-й бензин продает.

На вход программе сначала подается число данных о стоимости бензина N. В каждой из последующих *N* строк находится информация в следующем формате:

```
<Компания> <Улица> <Марка> <Цена>
```

где `<Компания>` — строка, состоящая не более чем из 20 символов без пробелов; `<Улица>` — строка, состоящая не более чем из 20 символов без пробелов; `<Марка>` — одно из чисел 92, 95 или 98; `<Цена>` — целое число в диапазоне от 1000 до 3000, обозначающее стоимость одного литра бензина в копейках. `<Компания>` и `<Улица>`, `<Улица>` и `<Марка>`, а также `<Марка>` и `<Цена>` разделены ровно одним пробелом. Пример входной строки:

```
СуперБенз Цветочная 92 1950
```

Программа должна выводить через пробел компанию и улицу искомой АЗС или их количество, если искомых вариантов несколько.

Пример выходных данных:

```
Бензинчик Перспективная
```

Второй вариант выходных данных:

```
4
```

Анализ решения

Прежде чем проводить анализ задачи, предлагаем читателям вернуться к задаче С4 из [5] в *главе 4 (см. разд. 4.2)*.

В обсуждаемой задаче речь идет о "второй по минимальности цене". Если вспомнить *(см. ТЗ 2.4.1 и ТЗ 2.4.2)* о двух толкованиях понятий "второй максимум" и "второй минимум", то можно утверждать, что имеется в виду значение второго минимума во втором толковании (меньше него только минимальное значение).

Следовательно, для решения необходимо определить количество АЗС, цена на 92-й бензин на которых равна второму минимуму, а также сведения о единственной такой АЗС (название компании и улицу). Такая задача — типовая *(см. ТЗ 2.4.5 и 2.4.6)*.

Может понадобиться также информация об АЗС, продающей 92-й бензин по минимальной цене (первый минимум) — в случае, когда она единственная, продающая 92-й бензин вообще (см. условие задачи). Нахождение количества первых минимумов и информации о соответствующей АЗС проводится как при решении *ТЗ 2.4.4*.

Возникает вопрос: "Как можно зафиксировать тот факт, что все АЗС, у которых 92-й бензин есть, продают его по одной и той же цене?" Решение, что называется, "в лоб" — подсчитать количество первых минимумов и общее количество АЗС, продающих 92-й бензин (эти величины должны быть равны). Но в этом случае надо использовать и определять значения этих двух величин (в дополнение к подсчету количества вторых минимумов).

Можно обойтись без использования общего количества АЗС, продающих 92-й бензин, — если количество вторых минимумов равно нулю, то это означает, что все АЗС, у которых 92-й бензин есть, продают его по одной и той же цене.

После определения искомых значений кол_мин1, кол_мин2, АЗС_мин1 и АЗС_мин2 фрагмент, связанный с выводом ответа, оформляется так:

```
если кол_мин2 > 0   |АЗС, продающие бензин по второй
                    |минимальной цене, есть
  то
    если кол_мин2 = 1 |Такая АЗС единственная
      то
        |Выводим информацию о ней
        вывод нс, АЗС_мин2
      иначе
        |Выводим количество таких АЗС
        вывод нс, кол_мин2
    все
  иначе |кол_мин2 = 0 - все АЗС продают 92-й бензин по одной цене
    если кол_мин1 > 1
      то
        |Выводим количество таких АЗС
        вывод нс, кол_мин1
      иначе  |Такая АЗС единственная
        |Выводим информацию о ней
        вывод нс, АЗС_мин1
    все
все
```

где кол_мин1 и кол_мин2 — число АЗС, продающих 92-й бензин соответственно по первой и по второй минимальной цене; АЗС_мин1 и АЗС_мин2 — сведения (название компании и улица) об одной (или о единственной) из таких АЗС.

Представляя общую структуру программы решения задачи, обратим внимание на тот факт, что обрабатывается только входная информация, связанная с 92-м бензином:

```
1. Ввод значения N
2. Цикл для i от 1 до N |Ввод и обработка входных строк
   2.1. Ввод информации об i-й АЗС
   2.2. Выделение компании и улицы как единого целого (ТЗ 2.1.3)
   2.3. Определение марки и цены бензина (ТЗ 2.1.7)
   2.4. Для 92-й марки:
      Определение количества первых и вторых минимумов
      и запоминание информации об АЗС, продающих бензин
      по таким ценам
конец цикла
3. Вывод ответа
```

Можно также выделение компании и улицы проводить не для всех АЗС, а только для продающих 92-й бензин (что можно установить, выделив сначала марку бензина путем "вырезки" двух соответствующих символов строки):

```
1. Ввод значения N
2. Цикл для i от 1 до N |Ввод и обработка входных строк
   2.1. Ввод информации об i-й АЗС
   2.2. Определение марки бензина
   2.3. Для 92-й марки:
      2.3.1. Выделение компании и улицы как единого целого
      2.3.2. Выделение цены
      2.3.3. Определение количества первых и вторых минимумов
             и запоминание информации об АЗС, продающих бензин
             по таким ценам
конец цикла
3. Вывод ответа
```

В заключение — несколько замечаний по программе на языке Паскаль, приведенной в [16].

1. Величины `s` (компания и улица для текущей АЗС), `s1` (то же, для АЗС, продающей 92-й бензин по первой минимальной цене) и `s2` (то же, по второй минимальной цене) описаны без учета максимально возможной длины соответствующих значений (она равна 42):

   ```
   ...s, s1, s2: string;
   ```

2. Начальное значение ("пустое") величины `s1` — не задано, хотя величине `cnt1` нулевое значение присваивается.

> ***Примечание***
>
> Задачи вариантов 6, 8 и 9 имеют ряд особенностей, существенно отличающих их от рассмотренных задач, и будут анализироваться в *главе 7*.

ГЛАВА 6

Задачи С4 из книги [12]

6.1. Вариант 1

Условие

После единых выпускных экзаменов по информатике в район пришла информация о том, какой ученик какой школы сколько набрал баллов. Эта информация в том же виде была разослана в школы.

Завуч школы № 50 решила наградить двух учащихся, которые лучше всех в школе сдали информатику.

Программа должна вывести на экран фамилии и имена учеников.

Если наибольший балл набрало больше двух человек — вывести количество таких учеников.

Если наибольший балл набрал один человек, а следующий балл набрали несколько человек — нужно вывести только фамилию и имя лучшего.

Напишите эффективную, в том числе и по используемой памяти, программу (укажите используемую версию языка программирования, например Borland Pascal 7.0), которая должна вывести на экран требуемую информацию. Известно, что информатику сдавало больше пяти учеников школы № 50.

На вход программе сначала подается число учеников, сдававших экзамен. В каждой из следующих *N* строк находится информация об учениках в формате:

```
<Фамилия> <Имя> <Номер школы> <Количество баллов>
```

где `<Фамилия>` — строка, состоящая не более чем из 30 символов без пробелов; `<Имя>` — строка, состоящая не более чем из 20 символов без пробелов; `<Номер школы>` — целое число в диапазоне от 1 до 99; `<Количество баллов>` — целое число в диапазоне от 1 до 100. Эти данные записаны через пробел, причем ровно один между каждой парой (т. е. всего по три пробела в каждой строке).

Пример входной строки:

```
Иванов Иван 50 87
```

Пример выходных данных:

```
Круглов Василий
Тарасова Дарья
```

Другой вариант выходных данных:

```
7
```

Третий вариант выходных данных:

```
Гусарский Илья
```

Анализ решения

Чтобы учесть требование условия о выводе ответа, составим таблицу возможных вариантов:

Фамилия и имя	**Баллы в порядке убывания**			
Круглов Василий	77	77	77	77
Тарасова Дарья	77	70	70	77
Петров Николай	60	60	70	77
Бойко Сергей	59	59	59	59
...				
Должно быть выведено	Круглов Василий Тарасова Дарья		Круглов Василий	3
Вариант выходных данных в условии	1		3	2

Для вывода требуемой в условии информации надо знать следующие данные об учениках школы № 50:

1. Количество учащихся, набравших максимальное число (первый максимум) баллов.
2. Количество учащихся, набравших второй максимум баллов.
3. Фамилию и имя ученика, набравшего первый максимум баллов.
4. То же ученика, набравшего второй максимум баллов.

Правда, здесь возникает вопрос: когда наибольший балл набрали два ученика, то кого считать лучшим? Ведь в таком случае возможны два варианта ответа:

```
Круглов Василий
Тарасова Дарья
```

и

```
Тарасова Дарья
Круглов Василий
```

Поэтому важно договориться о терминологии.

Пусть имена перечисленных выше четырех величин — соответственно кол_макс1, кол_макс2, фам_макс1, фам_макс2. Используем также величины макс1 — максимальный балл (первый максимум) и макс2 — второй максимум.

Условимся во множестве баллов, набранных отдельными учениками школы № 50, называть первым максимумом тот балл, который стоял бы на последнем месте, если бы все баллы были отсортированы по неубыванию[1]. Если такой балл набрали несколько учащихся, то будем учитывать фамилию и имя ученика, который в списке находится *раньше* других таких учеников. А вот (внимание!) в количестве первых максимумов будем учитывать *всех* таких учеников (от этого значения зависит ответ). Кроме того, нужно запоминать фамилию лучшего ученика (фам_макс1) и ученика, так сказать, занявшего второе место (фам_макс2).

Соответственно, вторым максимумом будем называть балл, который стоял бы на предпоследнем месте в только что указанных условиях. Обращаем внимание на то, что на предпоследнем месте может стоять как элемент, равный максимальному, так и меньше его.

Анализ сказанного показывает, что при сделанных допущениях величина кол_макс2 не подходит ни под одно из двух толкований понятия "второй максимум" *(см. разд. 2.4.1 и 2.4.5)*, а носит некий условный характер.

Итак, с учетом всего сказанного, задача определения значений кол_макс1, кол_макс2, фам_макс1 и фам_макс2 решается так:

```
макс1 := 0
кол_макс1 := 0
фам_макс1 := ""
...
|Для каждого ученика школы № 50
если балл > макс1
  |Встретился балл больше макс1
  то
    |Бывший первый максимум станет вторым
    макс2 := макс1
    |фамилия – аналогично
    фам_макс2 := фам_макс1
    |а значение кол_макс2 станет равно
    |"старому" значению величины кол_макс1
    кол_макс2 := кол_макс1
    |Первым максимумом станет встреченное число
    макс1 := балл
    |Запоминаем фамилию этого ученика
    фам_макс1 := фиу
```

[1] Или, что то же самое, стоял бы на первом месте, если все баллы были отсортированы по невозрастанию.

```
    |Новый первый максимум встретился впервые
    кол_макс1 := 1
  иначе
    |Проверяем, не равен ли очередной балл
    |"старому" первому максимуму
    если балл = макс1
      то
        |Вторым максимумом остается первый максимум
        макс2 := макс1
        |Запоминаем фамилию этого ученика
        фам макс2 := фиу
        |Увеличиваем значение кол_макс1 (см. выше)
        кол_макс1 := кол_макс1 + 1
      иначе
        |Сравниваем очередной балл со вторым максимумом
        если балл > макс2
          то |Встретился балл больше макс2
            |Новое значение второго максимума
            макс2 := балл
            |Пока он – единственный
            кол_макс2 := 1
            |Запоминаем также фамилию этого ученика
            фам_макс2 := фиу
          иначе
            если балл = макс2
              то |Встретился еще один второй максимум
                кол_макс2 := кол_макс2 + 1
                |Значение фам_макс2 не меняем
           все
        все
    все
все
```

где фиу — фамилия и имя очередного ученика (как единое целое), балл — набранный тем или иным учеником балл.

Итак, значения, необходимые для вывода ответа, найдены.

Учитывая условие задачи, фрагмент программы, относящийся к выводу ответа, можно оформить так:

```
если кол_макс1 = 2 или кол_макс1 = 1 и кол_макс2 = 1
  |Если лучших учеников — два
  то |Первый вариант ответа
       вывод нс, фам_макс1
       вывод нс, фам_макс2
  иначе
    если кол_макс1 = 1 и кол_макс2 > 1 |Один лучший ученик
      то |Третий вариант ответа
          вывод нс, фам_макс1
```

```
      иначе |Второй вариант ответа
          вывод нс, кол_макс1
    все
все
```

Общая структура программы решения обсуждаемой задачи такая:

```
1. Ввод значения N
2. Цикл для i от 1 до N |Ввод и обработка входных строк
   2.1. Ввод информации об i-м ученике
   2.2. Выделение фамилии и имени i-го ученика
        как единого целого (величины фиу) — см. ТЗ 2.1.3
   2.3. Выделение номера школы (школа) и балла (балл) i-го ученика
        (ТЗ 2.1.70)
   2.4. Если номер школы равен 50
          то
            Определение значений, от которых зависит ответ (см. выше)
конец цикла
3. Вывод ответа (см. выше)
```

В заключение — два замечания к программе на языке Паскаль, приведенной в [12]:

1. В значения величин, хранящих фамилию и имя ученика как одно целое, входят также два пробела *(см. ТЗ 2.1.3)*, поэтому их целесообразно описать следующим образом:

   ```
   S, Smax, Smax2: string[52];
   ```

   — по условию фамилия ученика — это строка, состоящая не более чем из 30 символов без пробелов, его имя — строка, состоящая не более чем из 20 символов без пробелов.

2. Среди начальных присваиваний значений:

   ```
   max := -1; Smax := ''; nmax := 0; nmax2 := -1;
   ```

   нет связанных с величинами `max2` и `Smax2` и есть такое:

   ```
   nmax2 := -1
   ```

   которое является лишним *(см. обсуждение ТЗ 2.4.5)*.

6.2. Вариант 2

Условие

После единых выпускных экзаменов по информатике в район пришла информация о том, какой ученик какой школы сколько набрал баллов.

Районный методист решила выяснить номер школы, ученики которой набрали наибольший средний балл с точностью до целых.

Программа должна вывести на экран номер такой школы и ее средний балл.

Если наибольший средний балл набрало больше одной школы — вывести количество таких школ.

Напишите эффективную, в том числе и по используемой памяти, программу (укажите используемую версию языка программирования, например Borland Pascal 7.0), которая должна вывести на экран требуемую информацию. Известно, что информатику сдавало больше пяти учеников района. Также известно, что в районе школы с некоторыми номерами не существуют.

На вход программе сначала подается число учеников, сдававших экзамен. В каждой из следующих *N* строк находится информация об учениках в формате:

```
<Фамилия> <Имя> <Номер школы> <Количество баллов>
```

где `<Фамилия>` — строка, состоящая не более чем из 30 символов без пробелов; `<Имя>` — строка, состоящая не более чем из 20 символов без пробелов; `<Номер школы>` — целое число в диапазоне от 1 до 99; `<Количество баллов>` — целое число в диапазоне от 1 до 100. Эти данные записаны через пробел, причем ровно один между каждой парой (т. е. всего по три пробела в каждой строке).

Пример входной строки:

```
Иванов Иван 50 87
```

Пример выходных данных:

```
50 74
```

Другой вариант выходных данных:

```
7
```

Анализ решения

Для решения задачи надо получить массив со средними баллами по каждой школе. С этой целью используем два массива с индексами от 1 до 99:

1. `с_школа` — для хранения информации об общей сумме баллов, набранных учениками той или иной школы.
2. `кол_школа` — для хранения информации об общем числе сдававших экзамен учеников из той или иной школы.

Их нужно заполнять во время ввода данных после определения номера школы и балла ученика.

После ввода всех *N* данных результаты расчета среднего балла по каждой школе будем записывать в тот же массив `с_школа`:

```
нц для i от 1 до 99
  если кол_школа[i] > 0  |Только в этом случае
    то |рассчитываем средний балл по школе
      с_школа[i] := int(с_школа[i]/кол_школа[i])
  все
кц (см. ТЗ 5.2)
```

Далее для вывода ответа нужно знать:

1. Максимальное значение в массиве с_школа.
2. Его индекс (номер соответствующей школы).
3. Количество максимальных значений в массиве.

Как отмечалось при анализе *ТЗ 1.9.3*, вместо двух первых величин можно применить только вторую (номер_макс). Третью же величину можно найти согласно методике, описанной применительно к *ТЗ 2.4.3*. Целесообразно объединить нахождение двух необходимых величин в одном операторе цикла:

```
|Начальное присваивание значений искомым величинам
номер_макс := 1
кол_макс := 1
|Рассматриваем остальные школы
нц для i от 2 до 99
  если с_школа[i] > с_школа[номер_макс]
    то      |Встретился новый максимум
      |Запоминаем его номер
      номер_макс :=  i
      |Пока он — единственный
      кол_макс := 1
    иначе
      |Проверяем, не равно ли очередное значение
      |"старому" максимуму
      если с_школа[i] = с_школа[номер_макс]
        то  |Встретился еще один максимум
            |Учитываем это
            кол_макс := кол_макс + 1
      все
  все
кц
```

В результате фрагмент программы, относящийся к выводу ответа, имеет вид:

```
если кол_макс = 1
  то
    вывод нс, номер_макс, " ", с_школа[номер_макс]
  иначе
    вывод нс, кол_макс
все
```

Итак, общая структура программы решения задачи такая:

```
1. Заполнение массивов с_школа и кол_школа нулевыми значениями
2. Ввод значения N
3. Цикл для i от 1 до N |Ввод и обработка входных строк
   3.1. Ввод информации об i-м ученике
   3.2. Выделение номера школы (школа) и балла (балл)
        i-го ученика — см. ТЗ группы 2.1 и ТЗ 2.5.3
```

```
   3.3. Учет этих значений в массивах с_школа и кол_школа
        (ТЗ 2.5.1, ТЗ 2.2.5)
конец цикла
4. Расчет среднего балла по каждой школе (см. выше)
5. Определение значений номер_макс и кол_макс (см. выше)
6. Вывод ответа (см. выше)
```

В заключение обратим внимание на то, что в приведенном в [12] фрагменте программы на языке Паскаль для подсчета необходимых величин в массиве со средними значениями:

```
max := 1; nmax := 1;
for i := 2 to 99 do
  if s[i] > s[max] then
    begin
      max := i;
      nmax := 1
    end
  else
    if s[i] = max then {считаем количество максимумов}
       nmax := nmax + 1;
```

имя `max` имеет величина, относящаяся не к максимальному элементу массива, а к его индексу.

6.3. Вариант 3

Условие

После единых выпускных экзаменов по информатике в район пришла информация о том, какой ученик какой школы сколько баллов набрал.

Районный методист решила выяснить номер школы, ученики которой набрали средний балл по школе, больший, чем районный средний балл (все средние баллы вычисляются с точностью до целых).

Напишите эффективную, в том числе и по используемой памяти, программу (укажите используемую версию языка программирования, например Borland Pascal 7.0), которая должна вывести на экран требуемую информацию. Известно, что информатику сдавало больше пяти учеников района. Также известно, что в районе школы с некоторыми номерами не существуют.

На вход программе сначала подается число учеников, сдававших экзамен. В каждой из следующих *N* строк находится информация об учениках в формате:

`<Фамилия> <Имя> <Номер школы> <Количество баллов>`

где `<Фамилия>` — строка, состоящая не более чем из 30 символов без пробелов; `<Имя>` — строка, состоящая не более чем из 20 символов без пробелов; `<Номер школы>` — целое число в диапазоне от 1 до 99; `<Количество баллов>` — целое число в диапазоне от 1 до 100. Эти данные записаны через пробел, причем ровно один между каждой парой (т. е. всего по три пробела в каждой строке).

Пример входной строки:

```
Иванов Иван 50 87
```

Пример выходных данных:

```
5 50 74 84
```

Другой вариант выходных данных:

```
7
Средний балл = 74
```

Анализ решения

Прежде всего о том, что такое "районный средний балл", фигурирующий в условии. Правильный ответ — это частное от деления общей суммы баллов, набранной всеми учащимися, на их число *N*. Оперировать для расчета районного среднего балла средними значениями балла по каждой школе нельзя[1] (но находить их придется — см. далее).

Как и в предыдущей задаче, используем два массива с индексами от 1 до 99:

1. `с_школа` — для хранения информации об общей сумме баллов, набранных учениками той или иной школы.
2. `кол_школа` — для хранения информации об общем количестве числе сдававших экзаменов учеников из той или иной школы.

Их также нужно заполнять во время ввода данных после считывания номера школы и балла ученика.

Для решения рассматриваемой задачи надо получить массив со средними баллами по каждой школе и рассчитать средний балл для всего района. Все эти расчеты можно объединить так:

```
с_район := 0 |Сумма баллов по району
нц для i от 1 до 99
  если кол_школа[i] > 0  |Только в этом случае
    то
      с_район := с_район + с_школа[i] |Учитываем сумму баллов этой школы
                                      |в сумме баллов по району
      |и рассчитываем средний балл по данной школе
      с_школа[i] := int(с_школа[i]/кол_школа[i])
  все
кц
|Рассчитываем средний балл для всего района
с_район:= int(с_район/N)
```

Видно, что, как и для предыдущей задачи, после ввода всех *N* входных данных результаты расчета среднего балла по каждой школе записываются в тот же массив

[1] "Физическое" отступление ☺. В задаче "по физике": "Первую половину пути автомобиль ехал со средней скоростью 40 км/час, а вторую — 60 км/час. Какова средняя скорость движения на всем маршруте?" — ответ 50 является неверным.

с_школа, а искомое значение среднего балла по району обозначено тем же именем, что сумма средних баллов по району (с_район). Так сделано по аналогии с решением, приведенным в [12].

Осталось только вывести на экран номера тех школ, где средний балл выше районного.

Поскольку может понадобиться также количество таких школ и величина среднего балла для единственной из них (см. условие), то "попутно" будем считать и эти значения (обозначим их соответственно k и балл):

```
k := 0
нц для i от 1 до 99
  |Ищем школы, где средний балл выше районного
  если с_школа[i] > с_район |Встретилась такая школа
    то
      вывод i, " "        |Выводим ее номер
      k := k + 1          |Учитываем ее в значении k
      балл := с_школа[i] |Запоминаем ее средний балл,
                          |который может понадобиться,
                          |если эта школа будет единственной
  все
кц
|При необходимости выводим также значение балл
если k = 1
  то
    вывод нс, "Средний балл = ", балл
все
```

Нетрудно заметить, что имя балл теперь использовано для величины среднего балла по отобранным школам (или по единственной такой школе).

Итак, общая структура программы решения обсуждаемой задачи такая:

```
1. Заполнение массивов с_школа и кол_школа нулевыми значениями
2. Ввод значения N
3. Цикл для i от 1 до N |Ввод и обработка входных строк
   3.1. Ввод информации об i-м ученике
   3.2. Выделение номера школы (школа) и балла (балл) i-го
        ученика — см. ТЗ группы 2.1 и ТЗ 2.5.3
   3.3. Учет этих значений в массивах с_школа и кол_школа
        (ТЗ 2.5.1, ТЗ 2.2.5)
конец цикла
4. Расчет среднего балла по каждой школе и среднего балла по району (см. выше)
5. Поиск номеров необходимых школ (с подсчетом значений k и балл)
   и вывод ответа (см. выше)
```

В заключение обратим внимание на то, что в приведенной в [12] программе на языке Паскаль сумма баллов по району неудачно обозначена именем avg (*average* в переводе с английского — *средний*):

```
avg := avg + s[i];
```

что может привести к неправильному пониманию методики расчета районного среднего балла.

6.4. Вариант 4

Условие

После единых выпускных экзаменов по информатике в район пришла информация о том, какой ученик какой школы сколько набрал баллов.

Районный методист решила выяснить фамилии учеников, которые набрали наибольший балл, по каждой школе в отдельности, но только если из школы информатику сдавало не меньше трех человек. Если в школе информатику сдавало меньше трех человек, информацию по этой школе выводить не нужно.

Программа должна вывести на экран информацию в виде:

```
<Номер школы> <Фамилия ученика>
```

в отдельной строке для каждой школы.

Напишите эффективную, в том числе и по используемой памяти, программу (укажите используемую версию языка программирования, например Borland Pascal 7.0), которая должна вывести на экран требуемую информацию. Известно, что информатику сдавало больше пяти учеников района. Также известно, что в районе школы с некоторыми номерами не существуют.

На вход программе сначала подается число учеников, сдававших экзамен. В каждой из следующих *N* строк находится информация об учениках в формате:

```
<Фамилия> <Имя> <Номер школы> <Количество баллов>
```

где `<Фамилия>` — строка, состоящая не более чем из 30 символов без пробелов; `<Имя>` — строка, состоящая не более чем из 20 символов без пробелов; `<Номер школы>` — целое число в диапазоне от 1 до 99; `<Количество баллов>` — целое число в диапазоне от 0 до 100. Эти данные записаны через пробел, причем ровно один между каждой парой (т. е. всего по три пробела в каждой строке).

Пример входной строки:

```
Иванов Иван 50 87
```

Пример выходных данных:

```
5 Иванов
50 Петров
74 Сидоров
```

Анализ решения

Прежде всего, обратим внимание на тот факт, что в условии не говорится о том, фамилию какого ученика школы нужно вывести, если в этой школе максимальный балл получили несколько учащихся. Анализ приведенного в книге [12] ответа показывает, что это должна быть фамилия ученика, информация о котором в общем списке находится раньше. Поэтому и мы учтем это обстоятельство.

Используем три массива с индексами от 1 до 99:

1. макс_школа — для хранения информации о максимальном балле, набранном отдельными учениками той или иной школы (тип данных — целый).
2. фам_макс — для хранения информации о фамилии ученика с максимальной суммой баллов в данной школе (тип данных — литерный[1]).
3. кол_школа — для хранения информации об общем количестве сдававших экзаменов учеников из той или иной школы (тип данных — целый).

Их нужно заполнять во время ввода данных после формирования фамилии (фам), номера школы (школа) и балла (балл) ученика:

```
|Выделяем из строки значения фам, школа и балл
...
если балл > макс_школа[школа] |Если балл очередного ученика больше
                              |максимального балла по его школе
  то
    макс_школа[школа] := балл  |Запоминаем новое значение
                               |максимального балла
    фам_макс[школа] := фам     |и фамилию этого ученика
все
|Общее количество учеников из школы считаем в любом случае
кол_школа[школа] := кол_школа[школа] + 1
```

После заполнения массивов можно получить ответ:

```
нц для школа от 1 до 99
  |Из всех школ выбираем только те,
  |из которых сдавало не меньше трех учеников
  если кол_школа[школа] >= 3
    то
      вывод нс, школа, " ", фам_макс[школа]
  все
кц (см. ТЗ 1.11.6 и ТЗ 2.3.2)
```

Общая структура программы решения задачи следующая:

```
1. Заполнение массивов макс_школа и кол_школа нулевыми значениями
2. Ввод значения N
3. Цикл для i от 1 до N |Ввод и обработка входных строк
   3.1. Ввод информации об i-м ученике
   3.2. Выделение фамилии, номера школы и балла i-го ученика —
        (ТЗ 2.1.1 и ТЗ 2.1.7)
   3.3. Использование этих значений для заполнения
        массивов макс_школа, фам_макс и кол_школа (см. выше)
конец цикла
4. Вывод ответа (см. выше)
```

[1] В терминах школьного алгоритмического языка (в терминах языка Паскаль — строковый).

В заключение обратим внимание на то, что в приведенном в [12] решении на языке Паскаль:

- величина `s`, связанная с фамилией очередного ученика, и элементы массива `name` с фамилиями учеников, набравших максимальный балл в той или иной школе, описаны длиной 52 символа, хотя, согласно условию, максимально возможная длина фамилии — 30 символов (и еще должен быть учтен пробел после нее);
- при выводе ответа в комментариях ошибочно указывается: "выбираем только школы, из которых сдавало больше трех учеников" (должно быть "выбираем только школы, из которых сдавало не меньше трех учеников");
- имя `bal` имеет массив с данными о максимальном балле, набранном отдельными учениками той или иной школы, а имя `num` — массив с информацией об общем количестве сдававших экзаменов учеников из школы.

6.5. Вариант 5

Условие

После единых выпускных экзаменов по информатике в район пришла информация о том, какой ученик какой школы сколько баллов набрал.

В районе считается подозрительной ситуация, когда в школе более двух учащихся набирают одинаковый наибольший балл по школе.

Районный методист решила выяснить номера таких школ.

Программа должна вывести номера этих школ, в любом порядке.

Если такая школа окажется одна, нужно вывести наибольший балл в этой школе, с указанием того, что это наибольший балл.

Если таких школ не окажется, нужно вывести об этом сообщение.

Напишите эффективную, в том числе и по используемой памяти, программу (укажите используемую версию языка программирования, например Borland Pascal 7.0), которая должна вывести на экран требуемую информацию. Известно, что информатику сдавало больше пяти учеников района. Также известно, что в районе школы с некоторыми номерами не существуют.

На вход программе сначала подается число учеников, сдававших экзамен. В каждой из следующих *N* строк находится информация об учениках в формате:

```
<Фамилия> <Имя> <Номер школы> <Количество баллов>
```

где `<Фамилия>` — строка, состоящая не более чем из 30 символов без пробелов; `<Имя>` — строка, состоящая не более чем из 20 символов без пробелов; `<Номер школы>` — целое число в диапазоне от 1 до 99; `<Количество баллов>` — целое число в диапазоне от 0 до 100. Эти данные записаны через пробел, причем ровно один между каждой парой (т. е. всего по три пробела в каждой строке).

Пример входной строки:

```
Иванов Иван 50 87
```

Пример выходных данных:

```
5 50 74 87
```

Другой пример выходных данных:

```
7
Наибольший балл = 74
```

Третий пример выходных данных:

```
Нет таких школ
```

Анализ решения

Здесь для ответа надо для каждой школы знать количество учеников, набравших наибольший балл среди свои "товарищей", и также максимальный балл. Нахождение этих значений проводится так, как описано в *ТЗ 2.4.3* (но применительно не к массиву, а к данным, выделяемым из входных строк). Найденные значения записываются соответственно в массивы кол_макс и макс с индексами от 1 до 99 каждый. Соответствующий фрагмент:

```
...
|Сравниваем балл очередного ученика (балл)
|с максимальным для данной школы (ее номер – школа)
если балл > макс[школа]
  то |Встретился новый максимум
    |Запоминаем это балл в массиве макс для соответствующей школы
    макс[школа] := балл
    |Пока он — единственный
    кол_макс[школа] := 1
  иначе
    |Проверяем, не равно ли значение балл
    |"старому" максимуму для данной школы
    если балл = макс[школа]
      то |Встретился еще один максимум
        |Учитываем это
        кол_макс[школа] := кол_макс[школа] + 1
    все
все
```

После заполнения массивов необходимо подсчитать (в массиве кол_макс) число школ, в которых одинаковый наибольший балл по школе набрали более двух учащихся *(см. ТЗ 2.3.1)*. При этом одновременно будем выводить номера таких школ *(см. ТЗ 1.11.6)* и запоминать максимальный балл по школе. Если число таких школ больше одного, то требуемый ответ будет в результате выведен (см. первый вариант выходных данных). В противном случае выводится либо максимальный балл по единственной такой школе (см. второй вариант выходных данных), либо сообщение о том, что таких школ нет:

```
всего_школ := 0
вывод нс
```

```
нц для i от 1 до 99
  |Отбираем только школы, у которых лучший балл
  |набрали больше двух учеников
  если кол_макс[i] > 2
    то
      вывод i, " "                    |Выводим номер данной школы
      всего_школ := всего_школ + 1    |Учитываем ее в числе школ, в которых
                                      |одинаковый наибольший балл по школе
                                      |набрали более двух учащихся
      макс_балл := макс[i]            |Запоминаем максимальный балл
                                      |(любой из них)
  все
кц
|Рассматриваем возможность других вариантов вывода ответа
если всего_школ = 1
  то
    вывод нс, "Наибольший балл = ", макс_балл
  иначе
    если всего_школ = 0
      то
        вывод нс, "Нет таких школ"
    все
все
```

Нетрудно увидеть, что в приведенном фрагменте именем макс_балл обозначен максимальный балл в одной из учитываемых школ (или в единственной такой школе).

Итак, общая структура программы решения задачи такая:

```
1. Заполнение массивов кол_макс и макс нулевыми значениями
2. Ввод значения N
3. Цикл для i от 1 до N |Ввод и обработка входных строк
   3.1. Ввод информации об i-м ученике
   3.2. Выделение номера школы (школа) и балла (балл)
        i-го ученика (ТЗ 2.1.7)
   3.3. Сравнение значения балл с максимальным для данной школы
         и изменение при необходимости соответствующих
         элементов массивов макс и кол_макс (см. выше)
конец цикла
4. Вывод номеров школ, в которых одинаковый наибольший балл по школе набрали
более двух учащихся, подсчет значения количества таких школ (всего_школ) и
запоминание максимального балла в одной из них (макс_балл) — см. выше
5. Вывод при необходимости одного из двух других вариантов ответа (см. выше)
```

Дополнение

Задачи С4 вариантов 7 и 8 из книги [12], наряду с общими признаками с задачами, рассмотренными ранее, имеют особенности, которые обуславливают их отдельный анализ.

Вариант 7

Условие

После единых выпускных экзаменов по информатике в район пришла информация о том, какой ученик какой школы сколько баллов набрал. По положению об экзамене каждый район сам определяет, за какой балл нужно поставить какую оценку.

Районный методист решила, что оценку "отлично" должны получить 20% участников (целое число, с отбрасыванием дробной части).

Для этого она должна определить, какой балл должен набрать ученик, чтобы получить "отлично".

Если невозможно определить такой балл, чтобы "отлично" получили ровно 20% участников, "отлично" должны получить меньше участников, чем 20%.

Если таких участников не окажется (наибольший балл набрали более 20% участников) — эти и только эти ученики должны получить "отлично".

Напишите эффективную, в том числе и по используемой памяти, программу (укажите используемую версию языка программирования, например Borland Pascal 7.0), которая должна вывести на экран наименьший балл, который набрали ученики, получившие "отлично". Известно, что информатику сдавало больше пяти учеников. Также известно, что есть такое количество баллов, которое не набрал ни один участник.

На вход программе сначала подается число учеников, сдававших экзамен. В каждой из следующих *N* строк находится информация об учениках в формате:

```
<Фамилия> <Имя> <Номер школы> <Количество баллов>
```

где `<Фамилия>` — строка, состоящая не более чем из 30 символов без пробелов; `<Имя>` — строка, состоящая не более чем из 20 символов без пробелов; `<Номер школы>` — целое число в диапазоне от 1 до 99; `<Количество баллов>` — целое число в диапазоне от 1 до 100. Эти данные записаны через пробел, причем ровно один между каждой парой (т. е. всего по три пробела в каждой строке).

Пример входной строки:

```
Иванов Иван 50 87
```

Пример выходных данных:

```
78
```

Анализ решения

При анализе используем следующие переменные величины:

- `балл` — набранный тем или иным учеником балл;
- `кол` — массив из 100 элементов с индексами от 1 до 100[1] для подсчета числа участников, набравших тот или иной балл;
- `норм5` — число учащихся, которые должны получить оценку "отлично".

[1] Это соответствует условию (в решении, приведенном в [12], массив описан с диапазоном индексов от 0 до 100).

Балл ученика можно выделить так, как описано при разборе типовых задач в *разд. 2.1*, или используя "вырезки" (для однозначного или двухзначного балла, или балла, равного 100):

```
если строка[длин(строка) — 1] = " "
  то |Балл однозначный (именно в таком порядке!)
    балл := лит_в_цел(строка[длин(строка)], успех)
  иначе
    если строка[длин(строка) — 2] = " "
      то |Балл двухзначный
        балл := лит_в_цел(строка[длин(строка) — 1 : длин(строка)], успех)
      иначе |Балл равен 100
        балл := 100
    все
все
```

где `строка` — входная строка с информацией об ученике.

Заполнения массива `кол` – *ТЗ 2.2.5*.

Значение величины `норм5` равно 20% от общего числа учеников:

```
норм5 := div(N, 5)
```

где `div` — функция, возвращающая целую часть от деления первого параметра на второй.

После заполнения массива `кол` и определения значения `норм5`, следует проверить, можно ли "набрать" ровно `норм5` учеников из показавших лучшие результаты. Для этого необходимо исследовать массив `кол` с его конца, применив, например, оператор цикла с предусловием:

```
сумм := 0 |Общее число учеников – "претендентов" на оценку "отлично"
i := 101
нц пока сумм < норм5
  |Рассматриваем очередной балл (начиная с конца массива)
  i := i – 1
  |Соответствующее этому баллу количество учеников
  |учитываем в значении сумм
  сумм := сумм + кол[i]
кц
```

Далее рассмотрим возможные варианты.

Вариант 1. Если после окончания работы оператора цикла мы остановимся на таком значении балла `i`, при котором `сумм = норм5`, то это значит, что ровно 20% участников могут получить "отлично", и балл для такой оценки равен `i`. Например, при `норм5 = 20` это может быть в таком случае:

Количество		5	5	7	0	3	0	0	0
Балл *i*	...	93	94	95	96	97	98	99	100

при котором ответ — 93.

Вариант 2. В противном случае — сначала проверим, не набрали ли наибольший балл более 20% участников. Это может быть в ситуации, например, такой (также при `норм5 = 20`):

Количество		21	0	0	0	0	0
Балл *i*	...	95	96	97	98	99	100

Такой ситуации соответствует условие:

`кол[i] > норм5`[1]

и в ней ответ — 21 (`кол[i]`).

Если этого нет, то "отлично" должны получить меньше участников, чем 20%. Пример такой ситуации:

Количество		8	0	0	5	4	7	3	0
Балл *i*	...	93	94	95	96	97	98	99	100

Здесь ответ — минимальный балл, больший, чем `i`, с ненулевым значением количества учеников, т. е. — 96.

Следовательно, фрагмент, связанный с выводом ответа, оформляется так:

```
если сумм = норм5
  то |"Отлично" можно поставить ровно 20% участникам
    вывод нс, i
  иначе
    если кол[i] > норм5
      то |Наибольший балл более 20% участников
         вывод нс, i
      иначе |"Отлично" должны получить меньше участников, чем 20%
        i := i + 1 |они набрали больше, чем i баллов
        |Ищем соответствующий балл
        нц пока кол[i] = 0
          i := i + 1
        кц
        |и выводим его
        вывод нс, i
    все
все
```

Общая структура программы решения задачи:

```
1. Заполнение массива кол нулевыми значениями
2. Ввод значения N
```

[1] Такое условие, на наш взгляд, более логично и понятно, чем приведенное в [12]: `k[i]=s`.

```
3. Цикл для i от 1 до N |Ввод и обработка входных строк
   3.1. Ввод информации об i-м ученике
   3.2. Выделение балла i-го ученика — (см. выше)
   3.3. Учет этого ученика в количестве участников,
        набравших такой балл в массиве кол (ТЗ 2.2.5)
конец цикла
4. Определение значения норм5 (см. выше)
5. Нахождение величины сумм (общего числа учеников – "претендентов" на оценку
   "отлично") и предельного значения балла i (см. выше)
6. Вывод одного из трех вариантов ответа (см. выше)
```

Вариант 8

Условие

После единых выпускных экзаменов по информатике в район пришла информация о том, какой ученик какой школы сколько баллов набрал. По положению об экзамене оценку "2" (неудовлетворительно) получают ученики, набравшие меньше 40 баллов. Оценку "3" (удовлетворительно) получает 30% учеников среди оставшихся[1], за исключением тех, кто набрал больше 60 баллов.

Если количество "троешников" оказывается больше 30%, то следует выбрать меньшую границу для оценки "4" (но только если при этом "3" получит хоть кто-нибудь).

Напишите эффективную, в том числе и по используемой памяти, программу (укажите используемую версию языка программирования, например Borland Pascal 7.0), которая должна вывести на экран наибольший балл, который набрали ученики, получившие "удовлетворительно", и количество таких учеников. Известно, что информатику сдавало больше 50 учеников. Также известно, что есть такое количество баллов, которое не набрал ни один участник.

На вход программе сначала подается число учеников, сдававших экзамен. В каждой из следующих *N* строк находится информация об учениках в формате:

```
<Фамилия> <Имя> <Номер школы> <Количество баллов>
```

где `<Фамилия>` — строка, состоящая не более чем из 30 символов без пробелов; `<Имя>` — строка, состоящая не более чем из 20 символов без пробелов; `<Номер школы>` — целое число в диапазоне от 1 до 99; `<Количество баллов>` — целое число в диапазоне от 0 до 100[2]. Эти данные записаны через пробел, причем ровно один между каждой парой (т. е. всего по три пробела в каждой строке).

Пример входной строки:

```
Иванов Иван 50 87
```

[1] Как и в задаче варианта 7, имеется в виду целое число, с отбрасыванием дробной части.

[2] В оригинале в условии указан диапазон баллов от 1 до 100, а в приведенной в нем программе массив описан и обрабатывается с диапазоном индексов от 0 до 100. Мы приняли последний вариант.

Пример выходных данных:

```
45 703
```

Анализ решения

При анализе используем следующие основные переменные величины:

- балл — набранный тем или иным учеником балл; его выделение из входной строки проводится так же, как и при решении предыдущей задачи;
- кол — массив из 101 элемента с индексами от 0 до 100 для подсчета числа участников, набравших тот или иной балл. Он заполняется аналогично предыдущей задаче;
- норм3 — число учащихся, которые должны получить оценку "3".

Последнее значение можно получить следующим образом (30% учеников, получивших оценку не ниже "3"):

```
норм3 := div((N - n2) * 30, 100)
```

где n2 — число, равное числу учащихся, получивших оценку "2".

Определить же число учащихся, получивших оценку "2", можно так:

```
n2 := 0
нц для i от 0 до 39[1]
  |Учитываем кол[i] в значении n2
  n2 := n2 + кол[i]
кц
```

Часть массива кол с индексами от 0 до 39 дальше не рассматриваем.

Для получения ответа необходимо среди "оставшихся" элементов массива кол найти границу оценки "3". Это должно быть такое минимальное значение балла i, при котором общее количество учащихся, набравших от 40 до i баллов включительно, больше или равно норм3, но при этом баллы больше 60 не должны учитываться.

Искомое значение можно найти, применив, например, оператор цикла с предусловием:

```
i := 39 |Начальный балл (учтен не будет)
сумм := 0
нц пока сумм < норм3 и i < 60
  |Переходим к очередному баллу  |Начинаем учитывать с 40 баллов
  i := i + 1
  |Учитываем в значении сумм
  сумм := сумм + кол[i]
кц |60 баллов еще учитываем
```

Далее рассмотрим возможные варианты. Для этого обсудим вопрос: "Когда может закончиться работа указанного оператора цикла?" Это может произойти в пяти случаях, представленных в таблице:

[1] В [12] ошибочно учитывается и значение 40 баллов, что не соответствует условию.

1	сумм = норм3	*i* < 60
2	сумм = норм3	*i* = 60
3	сумм > норм3	*i* < 60
4	сумм > норм3	*i* = 60
5	сумм < норм3	*i* = 60

Вариант 1. В двух первых случаях число троечников будет равно установленной "норме" — норм3 (30% участников с положительной оценкой). Один из примеров при норм3 = 20 (для i < 60):

Количество учеников	3	0	10	4	3	
Балл *i*	40	41	42	43	44	

в котором i = 44, и ответом будут значения 44 и 20.

Соответствующий фрагмент программы с выводом ответа:

```
если сумм = норм3
  то
   вывод нс, i, " ",  сумм
    ...
```

Вариант 2. Если же мы остановились раньше, чем были "набраны" норм3 учащихся, а могло это произойти, когда i = 60 (пятый случай в таблице), как, например, в такой ситуации (при норм3 = 20):

сумм = 15

Количество учеников	2	0	4	0		0	1	5	3	4	1
Балл *i*	40	41	42	43		57	58	59	60	61	62

ответом должны быть значения 60 и 15 (учащиеся с 61 и 62 баллами тройку не получают, общее число "троешников" — не 20, а 15). Сказанное можно записать так:

```
если сумм < норм3 и i = 60
  то
    вывод нс, 60, " ",  сумм
```

Вариант 3. Остались варианты, когда "троешников" больше 30% (больше норм3 человек). Напомним, что в этом случае следует выбрать меньшую границу для оценки "4". Такая граница — ближайший "слева" от i-го элемент массива с ненулевым значением. Ее можно найти так:

```
сумм := сумм – кол[i]
i := i – 1        |Балл i не учитываем
```

```
нц пока кол[i] = 0
 i := i - 1
кц
```

Но есть одно исключение — когда все "троешники" набрали одинаковый балл, например (при норм3 = 20):

Количество учеников	0	0	0	0	0	21	14
Балл *i*	40	41	42	43	44	45	46

В этом случае всех исключить из числа "троешников" нельзя (по условию тройку должен кто-то получить), и оценку "3" получат все такие ученики. В этом случае ответом являются значения i и сумм (для приведенного примера — 45 и 21). Условие для такого случая:

```
кол[i] = сумм
```

Итак, весь фрагмент, связанный с выводом ответа:

```
если сумм = норм3
  то |Оценку "3" можно поставить ровно 30% участников
     |с положительной оценкой
     |вывод нс, i, " ",  сумм
  иначе
    если сумм < норм3 и i = 60
      то
        вывод нс, 60, " ",  сумм
      иначе | сумм >  норм3
        если кол[i] = сумм
          то   |Все "троешники" набрали одинаковый балл
            вывод нс, i, " ",  сумм
          иначе |Ищем меньшую границу для оценки "4"
            |Значения для i баллов не учитываем
            сумм := сумм - кол[i]
            |и переходим к предыдущему баллу
            i := i - 1
            |Ищем первое слева ненулевое значение в массиве кол
            нц пока кол[i] = 0
              i := i - 1
            кц
           |и выводим соответствующие значения
           вывод нс, i, " ",  сумм
        все
    все
все
```

Общая структура программы решения задачи такая:

```
1. Заполнение массива кол нулевыми значениями
2. Ввод значения N
```

```
3. Цикл для i от 1 до N |Ввод и обработка входных строк
   3.1. Ввод информации об i-м ученике
   3.2. Выделение балла i-го ученика — (см. выше)
   3.3. Учет этого ученика в количестве участников,
        набравших такой балл в массиве кол (ТЗ 2.2.5)
конец цикла
4. Определение числа учащихся, получивших оценку "2" (см. выше)
5. Расчет значения числа норм3 (см. выше)
6. Поиск границы оценки "3" (см. выше)
7. Вывод ответа (см. выше)
```

ПРИМЕЧАНИЕ

Задачи вариантов 6, 9 и 10 имеют ряд особенностей, существенно отличающих их от рассмотренных задач, и будут анализироваться в *главе 7*.

ГЛАВА 7

Задачи на обработку последовательности латинских букв

В *главах 4–6* были рассмотрены задачи С4, содержание которых было связано с нахождением максимальных/минимальных значений (в том числе второго и третьего), их индексов, количеств и т. п.

В данной главе обсуждается методика решения задач, касающихся обработки последовательности букв латинского алфавита. При анализе используются две функции школьного алгоритмического языка:

1. код — возвращающая код символа — ее аргумента.
2. символ — возвращающая символ, код которого указан в качестве ее аргумента.

Соответствующие функции в других языках программирования представлены в таблице:

Функция	**Язык Паскаль**	**Язык Бейсик**
1. Возвращающая код символа	`Ord`	`Asc`
2. Возвращающая символ по его коду	`Chr`	`Chr$`

Напомним также, что в школьном языке программирования и в языке Паскаль величина строкового типа рассматривается как массив, элементами которого являются отдельные символы. Это позволяет обратиться к тому или иному символу строки как к элементу массива — по имени строковой величины и его (символа) номеру. Рассмотреть же все символы строки можно, подсчитав их количество, что, в свою очередь, можно сделать с помощью функции длин (`Length`).

7.1. Задача варианта 8 из [16]

Условие

На вход программе сначала подаются строчные английские буквы. Ввод символов заканчивается точкой (другие символы, отличные от "a".."z", во входных данных отсутствуют; в программе на языке Бейсик символы можно вводить по одному в

строке, пока не будет введена точка). Требуется написать как можно более эффективную программу (укажите используемую версию языка программирования, например Borland Pascal 7.0), которая будет печатать буквы, встречающиеся во входной последовательности, в порядке увеличения их встречаемости. Каждая буква должна быть напечатана один раз. Точка при этом не учитывается.

Если какие-то символы встречаются одинаковое количество раз, то они выводятся в алфавитном порядке. Например, пусть на вход подаются следующие символы:

```
baobaba.
```

В данном случае программа должна вывести:

```
oab
```

Анализ решения

Обсудим сначала проблему определения частоты вхождения каждой буквы, причем сделаем это на примере обработки заданной строки символов (ее имя в программе — `строка`).

Для хранения искомой информации используем массив из 26 элементов (общее число букв латинского алфавита) с именем `кол_букв`. Возникает вопрос: "Какой элемент массива соответствует некоторой букве заданной строки?"

Так как в школьном алгоритмическом языке, в языке программирования Бейсик и др. индексами элементов массива могут быть только числа, то напрашивается идея определения кода каждого символа заданной строки и использование кодов строчных латинских букв в качестве индексов массива `кол_букв`. При этом можно поступить по-разному. Если коды букв известны, то все их и можно использовать в виде диапазона индексов. Например, в программе на школьном алгоритмическом языке описание массива будет таким:

```
цел таб кол_букв[97:122]
```

Представляя соответствующую программу, заметим, что в ней код `i`-й буквы определяется с помощью одноименной функции (см. выше) и именно для элемента массива с индексом, равным коду, происходит увеличение значения на 1:

```
алг Подготовительная_задача
нач лит строка, цел таб кол_букв[97:122], цел i
  |Обнуляем массив кол_букв
  нц для i от 97 до 122
    кол_букв[i] := 0
  кц
  |Вводим и обрабатываем строку символов
  ввод строка
  нц для i от 1 до длин(строка)  |Для каждого символа строки
    |Увеличиваем его "счетчик" в массиве кол_букв
    кол_букв[код(строка[i])] := кол_букв[код(строка[i])] + 1
  кц
```

```
  |Выводим результат (символ определяется по его коду)
  нц для i от 97 до 122
    вывод нс, символ(i), " ", кол_букв[i]
  кц
кон
```

Можно также, не зная (или не помня) кодов букв, использовать код (пусть — неизвестный) латинской буквы "a" и тот факт, что коды букв идут подряд. В этом случае диапазон индексов может быть принят равным 1:26. Переход от кода i-й буквы к индексу массива может быть проведен так:

```
код(строка[i]) — код("a") + 1
```

Соответствующая программа:

```
алг Подготовительная_задача_вариант_2
нач лит строка, цел таб кол_букв[1:26], цел i
  нц для i от 1 до 26
    кол_букв[i] := 0
  кц
  ввод строка
  нц для i от 1 до длин(строка)
    кол_букв[код(строка[i]) — код("a") + 1] := кол_букв[код(строка[i]) –
        код("a") + 1] + 1
  кц
  нц для i от 1 до 26
    вывод нс, символ(код("a") + i — 1), " ", кол_букв[i]
  кц
кон
```

Обратим внимание на то, что при выводе символов происходит "обратный" перерасчет кодов.

> ***Примечание***
>
> Можно также использовать диапазон индексов 0:25, что позволит отказаться от прибавления и вычитания единицы при перерасчете.

В программе на языке Паскаль задача упрощается благодаря тому, что в этом языке значениями индексов элементов массива могут быть данные символьного типа:

```
kol_bykv: array ['a'..'z'] of byte;
```

и для i-й буквы строки (stroka[i]) может быть увеличен на 1 ее "счетчик" kol_bykv[stroka[i]].

Программа решения задачи на этом языке:

```
var
stroka: string; bykva: char; kol_bykv: array['a'..'z'] of byte; i: byte;
BEGIN
  {Обнуляем массив kol_bykv}
  for bykva := 'a' to 'z' do kol_bykv[i] := 0;
```

```
  {Вводим и обрабатываем строку символов}
  readln(stroka);
  for i := 1 to Length(stroka) do
    kol_bykv[stroka[i]] := kol_bykv[stroka[i]] + 1;
  {Выводим результат}
  for bykva := 'a' to 'z' do writeln(bykva, ' ', kol_bykv[i])
END.
```

Вернемся к "основной" задаче. Из ее условия и из приведенных в [16] примеров программ и комментариев в них следует, что заданные символы необходимо вводить в программу по одному (в программах на школьном алгоритмическом языке и на языке Бейсик — нажимая клавишу <Enter> после ввода каждого символа; в программе на языке Паскаль, как отмечалось в *главе 2*, имеется возможность считывания отдельных символов во время ввода заданной последовательности строки — при использовании оператора `read` с параметром символьного типа). Условие окончания ввода — введена точка. Если в массиве `кол_букв` диапазон индексов определить равным 0:25 (см. ранее), то фрагмент, относящийся к подсчету частоты вхождения каждой буквы, может быть оформлен в виде:

```
нц для i от 0 до 25
  кол_букв[i] := 0
кц
ввод симв
нц
  кол_букв[код(симв) — код("a")] := кол_букв[код(симв) — код("a")] + 1
  ввод симв
кц при симв = "."
```

где `симв` — очередной введенный символ.

Теперь о том, как напечатать буквы, имеющиеся во входной последовательности, в порядке увеличения их встречаемости.

Используем второй массив с именем `буквы`, в который запишем все строчные латинские буквы. При его наличии можно, сортируя числовой массив `кол_букв` (т. е. располагая значения в порядке возрастания), одновременно соответственно менять и массив с буквами.

Далее (после сортировки), так как нужно вывести только те буквы, которые имелись во входной последовательности, то необходимо пропустить буквы, число вхождений которых равно нулю, и вывести остальные (они будут напечатаны в порядке увеличения их встречаемости).

Вся программа имеет вид:

```
алг Задача_С4_варианта_8  |из книги [16]
нач сим симв, цел таб кол_букв[0:25], сим таб буквы[0:25],
    цел i , лев, всп, сим всп_букв
  |Обнуляем массив кол_букв
  нц для i от 0 до 25
    кол_букв[i] := 0
```

```
    |и заполняем массив буквы
    буквы[i] := символ(код("а") + i)
  кц
  |Вводим и обрабатываем строку, записывая количество букв в массив кол_букв
  ввод симв
  нц
    кол_букв[код(симв) — код("а")] := кол_букв[код(симв) — код("а")] + 1
    ввод симв
  кц при симв = "."
  |Сортируем массивы
  нц для i от 1 до 25 |25 раз
    |Сравниваем пары соседних элементов массива кол_букв
    нц для лев от 0 до 25 – i |Индексы левого элемента в паре сравниваемых
      если кол_букв[лев] > кол_букв[лев + 1]
        то |Меняем местами лев-й и (лев + 1)-й элементы двух массивов
          всп := кол_букв[лев]; всп_букв := буквы[лев]
          кол_букв[лев] := кол_букв[лев + 1]; буквы[лев] := буквы[лев + 1]
          кол_букв[лев + 1] := всп; буквы[лев + 1] := всп_букв
      все
    кц
  кц
  |Пропускаем буквы, число вхождений которых равно нулю
  i := 0
  нц пока кол_букв[i] = 0
    i := i + 1
  кц
  |Выводим остальные буквы из массива буквы
  нц для j от i до 25
    вывод буквы[j]
  кц
кон
```

Примечания

1. Для сортировки массива кол_букв использован метод обмена ("пузырьковая" сортировка). При проходах по массиву (всего их 25) длина обрабатываемого участка уменьшается *(см. приложение 4)*.
2. Величины всп (числового типа) и всп_букв (символьного типа) — вспомогательные для обмена значениями двух соседних элементов массивов кол_букв и буквы соответственно.

7.2. Задача варианта 10 из [12]

Условие

На вход программе сначала подается последовательность символов, заканчивающаяся символом #. Другие символы # во входной последовательности отсутствуют.

Программа должна вывести на экран символы латинского алфавита, в порядке увеличения частоты встречаемости во входной последовательности.

Если буква во входной последовательность не встречается, ее выводить не нужно.

Если несколько букв встречаются одинаковое количество раз, программа должна вывести их в алфавитном порядке.

Строчные и прописные буквы не различаются.

Напишите эффективную, в том числе и по используемой памяти, программу (укажите используемую версию языка программирования, например Borland Pascal 7.0), которая должна решать поставленную задачу.

Пример входных данных:

```
Aced, ccedaa f#
```

Пример выходных данных:

```
FDEAC
```

Анализ решения

Основное отличие данной задачи от предыдущей — во входной последовательности присутствуют не только строчные латинские буквы, но и прописные, а также другие символы.

Тот факт, что очередной введенный символ `симв` — строчная латинская буква, можно проверить с помощью условия:

```
симв >= "a" и симв <= "z"
```

а прописная —

```
симв >= "A" и симв <= "Z"
```

Теперь о подсчете числа вхождений каждой буквы. Так как по условию строчные и прописные буквы не различаются, то должно быть найдено общее число вхождений букв "a" и "A", "b" и "B" и т. д. Можно, конечно, найти отдельно число вхождений строчных и прописных букв, а затем просуммировать значения соответствующих элементов, но это нерационально.

Лучше использовать один массив `кол_букв` и учитывать в нем как строчные, так и прописные буквы:

```
если симв >= "a" и симв <= "z"
  то
    кол_букв[код(симв) – код("a")] := кол_букв[код(симв) – код("a")] + 1
  иначе
    если симв >= "A" и симв  <= "Z"
      то
        кол_букв[код(симв) – код("A")] :=
          кол_букв[код(симв) – код("A")] + 1
    все
все
```

Видно, что и для прописных букв определяется "относительный код" (по отношению к коду латинской буквы "A").

Программа на языках Паскаль и Бейсик несколько упрощается за счет того, что в них имеются функции, "меняющие" регистр букв (соответственно `UpCase`/`LowerCase` и `UCASE$`/`LCASE$`). Они позволяют "объединить" проверку того факта, что очередной введенный символ — латинская буква (строчная или прописная).

Например, в программе на Паскале подсчет букв проводится так:

```
{Если очередной символ - латинская буква}
if (UpCase(simv) >= 'A') and (UpCase(simv) <= 'Z') then
  {увеличиваем счетчик количества вхождений этой буквы}
  Inc(kol_bykv[UpCase(simv)]);
```

А какие буквы — строчные или прописные — печатать при выводе результата? Так как в примере, приведенном в условии, фигурируют прописные буквы, то так же поступим и мы:

```
|Обнуляем массив кол_букв
нц для i от 0 до 25
  кол_букв[i] := 0
  |и заполняем массив буквы прописными латинскими буквами
  буквы[i] := символ(код("A") + i)
кц
```

В остальном программа аналогична программе решения предыдущей задачи (с учетом того, что входная последовательность заканчивается символом "#").

7.3. Задача варианта 9 из [12]

Условие

На вход программе сначала подается последовательность символов, заканчивающаяся символом `#`. Другие символы `#` во входной последовательности отсутствуют.

Программа должна вывести на экран латинскую букву, встречающуюся во входной последовательности наибольшее количество раз, и число этих букв (во второй строке).

Если таких букв во входной последовательности окажется несколько, программа должна вывести на экран все их, через пробел, в алфавитном порядке.

Строчные и прописные буквы не различаются.

Напишите эффективную, в том числе и по используемой памяти, программу (укажите используемую версию языка программирования, например Borland Pascal 7.0), которая должна решать поставленную задачу.

Пример входных данных:

```
Day, mice"Year" - a mistake#
```

Пример выходных данных:

```
A
4
```

Другой вариант

Пример входных данных:

```
ABCD ABCE ABCF#
```

Пример выходных данных:

```
A B C
3
```

Анализ решения

Ясно, что и здесь понадобится подсчет числа вхождений каждой латинской буквы (без учета регистра) и запись их в массив кол_букв (см. ранее).

Из условия следует, что ответ зависит от количества букв с максимальным числом вхождений, т. е. от количества максимальных элементов в массиве кол_букв.

Такая задача, как отмечалось в *разд. 1.9.5*, может быть решена двумя способами:

1) за два прохода по массиву;

2) за один проход по массиву.

Напомним идею второго, более рационального, способа: проходя по массиву, кроме значения максимума контролировать также количество элементов, равных максимальному (кол_макс). Если очередной элемент оказывается больше текущего максимума — он принимается в качестве максимального значения, а величина кол_макс — равной 1. Если же очередной элемент не больше максимального, то сравниваем его с максимумом. Если они равны, то встретился еще один максимум, и значение кол_макс увеличиваем на 1:

```
|Ищем максимумы в массиве кол_букв
номер_макс := 1 |Индекс максимального значения
кол_макс := 1 |Счетчик количества максимумов в массиве кол_букв
нц для i от 1 до 25
  если кол_букв[i] > кол_букв[номер_макс]
    то |Встретился новый максимум
      |Запоминаем также его индекс
      номер_макс := i
      |Количество максимумов пока равно 1
      кол_макс := 1
    иначе
      если кол_букв[i] = кол_букв[номер_макс]
        то |Встретился еще один максимум
          кол_макс := кол_макс + 1
      все
  все
кц
```

Примечание

В приведенном фрагменте значение максимального элемента массива не запоминается, а используется его индекс (номер_макс).

Фрагмент, относящийся к выводу ответа, имеет вид:

```
если кол_макс = 1 |Если максимум один
  то |Выводим соответствующую букву
    вывод символ(код("A") + номер_макс)
  иначе
    |Выводим все буквы с максимальным числом вхождений
    нц для i от 0 до 25
      если кол_букв[i] = кол_букв[номер_макс]
        то
          вывод символ(код("A") + i), " "
      все
    кц
все
|Выводим, сколько раз встречается буква(буквы)
вывод нс, кол_букв[номер_макс]
```

Общая структура программы решения обсуждаемой задачи такая:

```
1. Обнуление массива кол_букв
2. Обработка заданной строки и заполнение массива кол_букв
3. Определение количества максимальных элементов
   в массиве кол_букв (см. выше)
4. Вывод ответа (см. выше)
```

7.4. Задача вариантов 6 и 9 из [16]

7.4.1. Задача варианта 6

Условие

На вход программе подается текст заклинания, состоящего не более чем из 200 символов, заканчивающийся точкой (символ "точка" во входных данных единственный). Оно было зашифровано юным волшебником следующим образом. Сначала волшебник определил количество букв в самом коротком слове, обозначив полученное число K (словом называется непрерывная последовательность латинских букв, слова отделяются друг от друга другими символами, длина слова не превышает 20 символов). Потом он заменил каждую латинскую букву в заклинании на букву, стоящую в алфавите на K букв ранее (алфавит считается циклическим, т. е. перед буквой "A" стоит буква "Z"), оставив другие символы неизменными.

Строчные буквы при этом остались строчными, прописные — прописными. Требуется написать программу на языке Паскаль или Бейсик, которая будет выводить на экран текст расшифрованного заклинания.

Например, если зашифрованный текст был таким:

```
Zb Ra Ca Dab Ra.
```

то результат расшифровки должен быть следующим:

```
Bd Tc Ec Fcd Tc.
```

7.4.2. Задача варианта 9

Условие

На вход программе подается текст заклинания, состоящего не более чем из 200 символов, заканчивающийся точкой (символ "точка" во входных данных единственный). Оно было зашифровано юным волшебником следующим образом. Сначала волшебник определил количество букв в самом коротком слове, обозначив полученное число K (словом называется непрерывная последовательность латинских букв, слова отделяются друг от друга другими символами, длина слова не превышает 20 символов). Потом он заменил каждую латинскую букву в заклинании на следующую за ней K-й по счету в алфавите (алфавит считается циклическим, т. е. после буквы "Z" стоит буква "A"), оставив другие символы неизменными. Строчные буквы при этом остались строчными, прописные — прописными. Требуется написать как можно более эффективную программу (укажите используемую версию языка программирования, например Borland Pascal 7.0), которая будет выводить на экран текст расшифрованного заклинания. Например, если зашифрованный текст был таким:

```
Bd Tc Ec Fcd Tc.
```

то результат расшифровки должен быть следующим:

```
Zb Ra Ca Dab Ra.
```

Прежде чем решать эти две задачи, обсудим ряд "подготовительных" задач.

7.4.П1. Дано предложение, заканчивающееся точкой. Слова в нем разделены одним пробелом. Найти длину самого большого слова

Задачу можно решить, выделив все слова и записав их в массив, после чего решить задачу нахождения максимального значения его элементов. Соответствующая методика описана в *разд. 1.15.6*.

Однако в заданиях ЕГЭ требуется решать подобные задачи без использования массивов. Поэтому мы при решении данной и других задач будем соблюдать это требование.

Можно выделять каждое отдельное слово, не записывая его в массив, подсчитывать количество букв в нем и после окончания слова сравнить это количество с максимумом длины для ранее обработанных слов. Тот факт, что некоторое слово кончилось, можно зафиксировать, сравнив очередной символ с пробелом и точкой.

В приведенной ниже программе используются следующие основные величины:

- строка — заданное предложение;
- слово — отдельное слово в нем;
- длина_слова — количество букв в некотором слове;
- макс — искомая величина максимальной длины слов.

Общая структура программы такая:

```
1. Ввод предложения
2. Цикл для каждого символа предложения
   если очередной символ — не пробел и не точка
     то
       Слово начинается или продолжается -
       учитываем символ-букву в длине слова
     иначе
       Слово кончилось:
       1) сравниваем его длину со "старым" значением величины макс
       если длина_слова > макс
         то Встретился новый максимум -
           принимаем значение длина_слова в качестве макс
       все
       2) значение величины длина_слова принимаем равным нулю

   все
конец цикла
3. Вывод ответа
```

Соответствующая программа:

```
алг Задача_7_4_П1
нач лит строка, цел длина_слова, макс, i
  вывод нс, "Введите предложение"
  ввод строка
  макс := 0
  длина_слова := 0
  нц для i от 1 до длин(строка)
    если строка[i] <> " " и строка[i] <> "."
      то |Слово продолжается.
        |Учитываем букву в длине слова
        длина_слова := длина_слова + 1
      иначе |Слово кончилось
        |Сравниваем его длину со значением макс
        если длина_слова > макс
          то
            макс := длина_слова
        все
```

```
        |Новое значение величины длина_слова
        длина_слова := 0
    все
  кц
  |После обработки всех символов предложения выводим ответ
  вывод нс, "Максимальная длина слова: ", макс
кон
```

Можно не использовать сложное условие начала или продолжения слова (строка[i] <> " " и строка[i] <> "."). Для этого конечное значение параметра цикла нужно сделать равным длин(строка) - 1:

```
нц для i от 1 до длин(строка) - 1
  если строка[i] <> " "
    то |Слово продолжается.
      ...
```

Однако в этом случае нужно проверить последнее слово отдельно:

```
  ...
кц
|Проверяем последнее слово
если длина_слова > макс
  то
    макс := длина_слова
все
```

7.4.П2. Дано предложение, заканчивающееся точкой. Слова в нем разделены пробелами (одним или несколькими). Найти длину самого большого слова

Для решения данной задачи может быть использована программа Задача_7_4_П1.

7.4.П3. Дано предложение, заканчивающееся точкой. Слова в нем разделены одним пробелом. Найти длину самого короткого слова

Задача решается аналогично *задаче 7.4.П1* (с учетом того, что ищется не максимум, а минимум):

```
алг Задача_7_4_П3
нач лит строка, цел длина_слова, мин, i
  вывод нс, "Введите предложение"
  ввод строка
  мин :=  20 |Условно
  длина_слова := 0
  нц для i от 1 до длин(строка) - 1
    если строка[i] <> " "
```

```
      то |Слово продолжается
        длина_слова := длина_слова + 1
      иначе
        если длина_слова < мин
          то
            мин := длина_слова
        все
        длина_слова := 0
    все
  кц
  |Проверяем последнее слово
  если длина_слова < мин
    то
      мин := длина_слова
  все
  вывод нс, "Минимальная длина слова: ", мин
кон
```

7.4.П4. Дано предложение, заканчивающееся точкой. Слова в нем разделены пробелами (одним или несколькими). Найти длину самого короткого слова

Здесь, в отличие от *задачи 7.4.П2*, программу решения предыдущей задачи применить нельзя. Особенность в том, что когда рассматривается второй (из нескольких подряд идущих) пробел, то в программе `Задача_7_4_П3` фиксируется факт окончания слова, а текущее значение величины `длина_слова` принимается равным 0, и именно это значение будет принято в качестве искомого.

Фрагмент программы, связанный с обработкой отдельных символов строки, правильно учитывающий факт окончания слова, имеет вид:

```
нц для i от 1 до длин(строка) — 1
  если строка[i] <> " "
    то |Слово продолжается
       длина_слова := длина_слова + 1
    иначе |Очередной символ — пробел
      если строка[i — 1] <> " " |Только в этом случае!
        то
          |Слово кончилось
          |Сравниваем его длину со значением мин
          если длина_слова < мин
            то
              мин := длина_слова
            все
            |Новое значение величины длина_слова
            длина_слова := 0
          все
```

```
      все
  все
кц
|Проверяем последнее слово
|Считаем, что перед точкой пробелов нет
если длина_слова < мин
  то
    мин := длина_слова
все
```

Можно также применить величину `начато` логического типа, определяющую, начато ли очередное слово. С ее использованием общая схема фрагмента программы, относящегося к нахождению минимальной длины слова, будет такой:

```
1. Цикл для каждого символа предложения, кроме последней точки
   если слово было начато
     то
       если очередной символ — не пробел
         то
           Слово продолжается — учитываем символ в длине слова
         иначе |Очередной символ — пробел
           Слово кончилось:
           1) сравниваем его длину со "старым" значением
              величины мин;
           2) значение величины начато становится ложным
             (новое слово еще не началось)
       все
    иначе |Слово не было начато
      если очередной символ — не пробел
       то
         Началось новое слово:
         1) значение величины начато становится истинным;
         2) значение величины длина_слова становится равным 1
    все
  все
конец цикла
2. Проверка длины последнего слова
```

Соответствующая программа:

```
алг Задача_7_4_П4_вариант_2
нач лит строка, цел длина_слова, мин, i, лог начато
  вывод нс, "Введите предложение"
  ввод строка
  мин := 20 |Условно
  начато := нет
  длина_слова := 0
  нц для i от 1 до длин(строка) — 1
```

```
    |Если слово было начато
    если начато
      то
        если строка[i] <> " "
          то  |Оно продолжается
            |Уточняем его длину
            длина_слова := длина_слова + 1
          иначе |Слово кончилось
            |Проверяем его длину
            если длина_слова < мин
              то
                мин := длина_слова
            все
            |Следующее слово еще не началось
            начато := нет
        все
      иначе  |Слово не началось
        если строка[i] <> " "
          то |Началось новое слово
            начато := да
            |Его длина пока равна 1
            длина_слова := 1
        все
    все
  кц
  |Проверяем последнее слово
  если длина_слова < мин
    то
      мин := длина_слова
  все
  вывод нс, "Минимальная длина слова: ", мин
кон
```

7.4.П5. Дано предложение на английском языке, заканчивающееся точкой. Найти длину самого короткого слова (словом будем называть непрерывную последовательность латинских букв, слова друг от друга отделены другими символами)

Здесь удобно использовать второй вариант решения предыдущей задачи (с применением величины логического типа `начато`). Особенности решения данной задачи выделены жирным начертанием:

```
1. Цикл для каждого символа предложения, кроме последней точки
     если слово было начато
       то
         если очередной символ — латинская буква
           то
             Слово продолжается — учитываем символ в длине слова
```

```
          иначе
            Слово кончилось:
            1) сравниваем его длину со "старым" значением
               величины мин;
            2) значение величины начато становится ложным
               (новое слово еще не началось)
        все
      иначе |Слово не было начато
        если очередной символ — латинская буква
          то
            Началось новое слово:
            1) значение величины начато становится истинным;
            2) значение величины длина_слова становится равным 1
        все
    все
конец цикла
2. Проверка длины последнего слова
```

Тот факт, что i-й символ предложения это латинская буква (элемент слова), определяет следующее условие:

```
строка[i] >= "a" и строка[i] <= "z" или строка[i] >= "A" и строка[i] <= "Z"
```

Примечания

1. В приведенной чуть ранее общей схеме фрагмента программы проверка того факта, что очередной символ — латинская буква, происходит дважды, а условие проверки является сложным. Можно несколько упростить фрагмент, если для каждого символа сначала проверять указанный факт, а затем рассматривать возможные значения величины начато. Соответствующий вариант использован при решении задачи С4 варианта 6 из [12] (см. далее).
2. В программе на языке Паскаль можно не считывать, а затем обрабатывать строку символов (предложение), а считывать и сразу обрабатывать по одному символу во время ввода предложения. Для этого следует использовать оператор read с параметром символьного типа. Все символы можно считать, применив оператор цикла с постусловием:

   ```
   repeat
   read(c);
   ...
   until c = '.'
   ```

   Кроме того, для проверки того факта, что очередной символ — латинская буква, можно использовать множества:

   ```
   if c in ['a'..'z', 'A'..'Z'] then ...
   ```

7.4.П6. Дан текст на английском языке, состоящий из прописных букв (других символов в тексте нет). Получить текст, в котором каждая буква исходного текста заменена на букву, стоящую в алфавите на *k* букв правее. Алфавит считается циклическим, т. е. после буквы "Z" стоит буква "A"

Если исходную букву обозначить `буква`, то код заменяющей ее буквы, учитывая, что буквы в таблице кодировки идут непрерывно, будет равен `код(буква) + k`, а соответствующая буква `(новая) — символ(код(буква) + k)`. При этом если новый код будет больше кода буквы "Z" (или, что то же самое, но нагляднее — `новая` > "Z"), то, согласно правилу цикличности, необходимо уменьшить код на общее число букв в латинском алфавите (26).

Поэтому программа решения задачи может быть оформлена так:

```
алг Задача_7_4_П6
  нач лит текст, текст2, сим новая, цел k, i
  вывод нс, "Введите текст: "
  ввод текст
  вывод нс, "Задайте величину сдвига: "
  ввод k
  текст2 := "" |Новый текст
  |Для каждой буквы текста
  нц для i от 1 до длин(текст)
    |Получаем символ, код которого на k больше
    новая := символ(код(текст[i]) + k)
    |При необходимости уточняем его
    если новая > "Z"
      то
        новая := символ(код(новая) — 26)
    все
    |и "добавляем" к "старому" значению текст2
    текст2 := текст2 + новая
  кц
  вывод нс, "Новый текст: ", текст2
кон
```

7.4.П7. Дан текст на английском языке, состоящий из строчных букв (других символов в тексте нет). Получить текст, в котором каждая буква исходного текста заменена на букву, стоящую в алфавите на *k* букв правее. Алфавит считается циклическим, т. е. после буквы "z" стоит буква "a"

Задача решается аналогично предыдущей.

7.4.П8. Дан текст на английском языке, состоящий из прописных букв (других символов в тексте нет). Заменить каждую букву текста на букву, стоящую в алфавите на *k* букв левее. Алфавит считается циклическим, т. е. перед буквой "A" стоит буква "Z"

Фрагмент программы решения данной задачи, отличающийся от программы решения *задачи 7.4.П6* (учитывающий другое "направление" сдвига), такой:

```
нц для i от 1 до длин(текст)
  |Получаем символ, код которого на k меньше
  новая := символ(код(текст[i]) - k)
  |При необходимости уточняем его
  если новая < "A"
    то
      новая := символ(код(новая) + 26)
  все
кц
```

Кроме того, поскольку в данной задаче требуется заменить исходный текст, то величину `текст2` можно не использовать, а после определения каждой новой буквы заменять ею "старую":

```
...
  |и заменяем соответствующую букву
  текст[i] := новая
кц
вывод нс, "Новый текст: ", текст
```

7.4.П9. Дан текст на английском языке, состоящий из строчных букв (других символов в тексте нет). Заменить каждую букву текста на букву, стоящую в алфавите на *k* букв левее. Алфавит считается циклическим, т. е. перед буквой "a" стоит буква "z"

Задача решается аналогично предыдущей.

7.4.П10. Дан текст на английском языке, состоящий из букв (других символов в тексте нет). Заменить каждую букву текста на букву, стоящую в алфавите на *k* букв правее. Алфавит считается циклическим, т. е. после буквы "Z" стоит буква "A", а после буквы "z" — "a"

Сразу же заметим, что попытка объединить условия уточнения символов из *задач 7.4.П6* и *7.4.П8* с помощью логической связки **или**:

```
|Для каждой буквы текста
нц для i от 1 до длин(текст)
  |Получаем символ, код которого на k больше
  новая := символ(код(текст[i]) + k)
```

```
  |При необходимости уточняем его
  если новая > "Z" или новая > "z"
    то
      новая := символ(код(новая) — 26)
  все
кц
```

приведет к неправильному результату (убедитесь в этом!).

Необходимо в программе рассматривать отдельно прописные и строчные буквы:

```
алг Задача_7_4_П10
нач лит текст, сим новая, цел k, i
  вывод нс, "Введите текст: "
  ввод текст
  вывод нс, "Задайте величину k: "
  ввод k
  текст2 := ""
  нц для i от 1 до длин(текст)
    новая := символ(код(текст[i]) + k)
    если текст[i] >= "A" и текст[i] <= "Z"
      то
        если новая > "Z"
          то
            новая := символ(код(новая) — 26)
        все
    все
    если текст[i] >= "a" и текст[i] <= "z"
      то
        если новая > "z"
          то
            новая := символ(код(новая) — 26)
        все
    все
    текст[i] := новая
  кц
  вывод нс, "Новый текст: ", текст
кон
```

В приведенном фрагменте в любом случае происходят две проверки каждого символа (строчная буква или прописная). Целесообразно оформить эти два неполных условных оператора как один полный:

```
если текст[i] >= "A" и текст[i] <= "Z"
  то
    если новая > "Z"
      то
        новая := символ(код(новая) — 26)
    все
```

```
  иначе
    если текст[i] >= "a" и текст[i] <= "z"
      то
        если новая > "z"
          то
            новая := символ(код(новая) — 26)
        все
    все
все
```

В программах на языках Паскаль и Бейсик, как и при решении задачи варианта 10 из книги [12] *(см. разд. 7.2)*, можно упростить проверку того факта, что очередной введенный символ — строчная или прописная латинская буква, используя функции, "меняющие" регистр.

При циклическом сдвиге влево задача решается аналогично.

После рассмотрения перечисленных вспомогательных задач решение задач С4 вариантов 6 и 9 из книги [16] (см. ранее) становится прозрачным. Можно выделить два основных этапа их решения:

1. Определение длины самого короткого слова в зашифрованном заклинании — такая задача уже решалась ранее *(см. задачу П7.4.5)*.
2. В задаче варианта 9 — замена каждой латинской буквы (прописной и строчной) на букву, стоящую в алфавите на *k* букв правее (с учетом "цикличности" алфавита); в задаче варианта 6 — то же, но левее. Эти задачи также рассмотрены как "подготовительные".

Прежде чем представлять в качестве примера программу решения задачи варианта 9, заметим, что в ней не формируется расшифрованное заклинание, а происходит вывод его символов по одному. Обратим также внимание на то, что новый (после сдвига) символ целесообразно определять для каждого символа заданного зашифрованного текста, а выводить — только латинские буквы.

```
алг Задача_С4_варианта_9    |Из книги [16]
нач лит шифр, цел длина_слова, k, i, лог начато, сим новая
  вывод нс, "Введите зашифрованный текст: "
  ввод шифр
  |1. Определение длины k самого короткого слова
  k := 21
  длина_слова := 0
  начато := нет
  нц для i от 1 до длин(шифр) - 1
    ...
  кц
  если длина_слова < k
    то
      k := длина_слова
  все
```

```
  |2.  Определение и вывод исходных символов
нц для i от 1 до длин(шифр)
  |Определяем новый символ (для всех символов зашифрованного текста)
  новая := символ(код(шифр[i]) — k)
  |Для букв латинского алфавита новые буквы выводим
  |(при необходимости уточняя их)
  если шифр[i] >= "A" и шифр[i] <= "Z"
    то
      если новая < "A"
        то
          вывод символ(код(новая) + 26)
        иначе
          вывод новая
      все
    иначе
      если шифр[i] >= "a" и шифр[i] <= "z"
        то
          если новая < "a"
            то
              вывод символ(код(новая) + 26)
            иначе
              вывод новая
          все
        иначе |Очередной символ — не латинская буква
          |Выводим его
          вывод шифр[i]
      все
  все
кц
кон
```

В заключение обсуждения данной задачи заметим, что в программе на языке Паскаль, представленной в [16], происходит обработка отдельных символов во время их ввода (такая возможность отмечалась ранее), но при этом одновременно формировалось все вводимое зашифрованное заклинание[1], которое затем расшифровывалось.

7.5. Задача варианта 6 из [12]

Условие

При программировании школьной тестирующей системы по английскому языку выяснилось, что файлы с вопросами к тестам легко доступны и каждый может перед тестом открыть их и заранее узнать вопросы. Было решено закодировать файлы. Для этого придумали следующий алгоритм.

---

[1] Соответствующая величина *s* описана в программе как `s: string`, хотя, согласно условию, текст заклинания состоит из не более чем 200 символов.

Каждая строка файла кодируется отдельно.

В каждой строке ищутся отдельные слова, и все символы слова сдвигаются по алфавиту циклически вправо на длину слова.

Словом считается любая последовательность подряд идущих символов латинского алфавита, строчных или прописных.

Циклический сдвиг символа по алфавиту вправо на *X* — замена символа на символ, стоящий в алфавите на *X* позиций дальше. Если при этом происходит выход за пределы алфавита, счет начинается с начала алфавита.

Пример циклического сдвига символа на три позиции: буква "E" превращается в букву "H", буква "t" — в букву "w", буква "Y" — в букву "B".

Напишите эффективную, в том числе и по используемой памяти, программу (укажите используемую версию языка программирования, например Borland Pascal 7.0), которая должна закодировать строку по указанному алгоритму.

На вход программе подается строка, состоящая из не более чем 250 символов латинского алфавита, пробелов, знаков препинания, разного рода скобок, кавычек и других символов. Строка заканчивается символом "#"Других символов "#" в строке нет.

Программа должна вывести закодированную по указанному алгоритму строку.

Пример входных данных:

```
Day, mice“Year” — a mistake#
```

Пример выходных данных:

```
Gdb, qmgi“Ciev” — b tpzahrl#
```

Анализ решения

Общая идея — для каждого слова строки необходимо определить его длину, а затем зашифровать его по заданному алгоритму.

Фрагмент, относящийся к определению длины каждого слова, может быть оформлен в виде (применен вариант, указанный при анализе *задачи 7.4.П5*):

```
вывод нс, "Введите строку"
ввод строка
длина_слова := 0
начато := нет
нц для i от 1 до длин(строка)
  |Если очередной символ — латинская буква
  если строка[i] >= "a" и строка[i] <= "z" или
        строка[i] >= "A" и строка[i] <= "Z"
    то
      если начато |Если ранее слово было начато
        то |Слово продолжается (это не первая буква)
          |Учитываем букву в длине слова
          длина_слова := длина_слова + 1
```

```
        иначе |Слово не начиналось
          |Началось новое слово
          начато := да
          |Его длина пока равна 1
          длина_слова := 1
      все
    иначе |Встретилась не латинская буква
      если начато
        то |Слово кончилось
          |Шифруем данное слово
          ...
          начато := нет |Новое слово еще не начиналось
      все
  все
кц
```

Итак, мы зафиксировали факт окончания некоторого слова. Это произошло на символе–"небукве" с порядковым номером `i`, а длина этого слова равна `длина_слова`. Какие буквы надо шифровать? Анализ примера:

	H	**E**	**L**	**L**	**O**	**!**
						i

показывает, что это буквы с номерами `i - 1`, `i - 2` ... `i - длина_слова`. Методика шифрования букв с циклическим смещением вправо обсуждена в *задаче 7.4.П10*. На ее основе фрагмент, связанный с заменой букв конкретного слова, оформляется так:

```
нц для j от i — длина_слова до i — 1
  |Определяем новую букву
  новая := символ(код(строка[j]) + длина_слова)
  если строка[j] >= "A" и строка[j] <= "Z"
    то
      если новая > "Z"
        то
          новая := символ(код(новая) — 26)
      все
    иначе
      если строка[j] >= "a" и строка[j] <= "z"
        то
          если новая > "z"
            то
              новая := символ(код(новая) — 26)
          все
      все
  все
```

```
  |Заменяем ею "старую"
  строка[j] := новая
кц
```

Этот фрагмент должен быть вставлен в предыдущий фрагмент вместо многоточия (после комментария `Шифруем данное слово`).

И, конечно, надо вывести зашифрованную строку:

```
вывод нс, "Зашифрованная строка: ", строка
```

ПРИЛОЖЕНИЯ

ПРИЛОЖЕНИЕ 1

О задачах С1

На Едином государственном экзамене, начиная с 2009 года, стали появляться задачи (обозначенные как С1), в которых требуется написать программу (точнее, исправить предложенную неверную программу, но сути дела это не меняет), которая определяет принадлежность точки с заданными координатами некоторой области, границами которой являются графики функций. В качестве примера перечислим четыре такие задачи, приведенные в демонстрационных вариантах ЕГЭ.

Примеры задач

2009 — С1[1]

Требовалось написать программу, при выполнении которой с клавиатуры считываются координаты точки па плоскости (x, y — действительные числа) и определяется принадлежность этой точки заданной заштрихованной области (включая границы) — рис. П1.1.

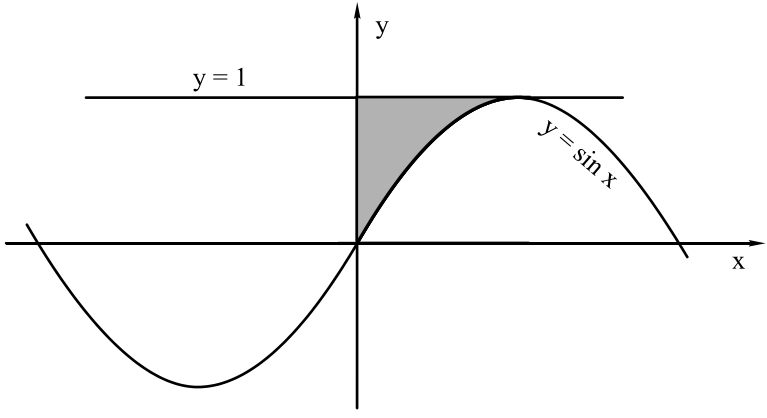


Рис. П1.1

[1] Здесь и далее принято следующее обозначение для заданий: сначала записан год, а затем через тире обозначение задания в демонстрационном варианте ЕГЭ соответствующего года.

Программист торопился и написал программу неправильно.

Программа на Паскале

```
var x,y: real;
begin
readln(x,y);
if y<=1 then
if x>=0 then
if y>=sin(x) then
write('принадлежит')
else
write('не принадлежит')
end.
```

Программа на Бейсике

```
INPUT x, y
IF y<=1 THEN
IF x>=0 THEN
IF y>=SIN(x) THEN
PRINT "принадлежит"
ELSE
PRINT "не принадлежит"
ENDIF
ENDIF
ENDIF
END
```

2010 — С1

Требовалось написать программу, при выполнении которой с клавиатуры считываются координаты точки па плоскости (*x*, *y* — действительные числа) и определяется принадлежность этой точки заданной заштрихованной области (включая границы) — рис. П1.2.

Программист торопился и написал программу неправильно.

Программа на Паскале

```
var x,y: real;
begin
readln(x,y);
if x*x+y*y>=4 then
if x>-2 then
if y<=-x then
write('принадлежит')
else
write('не принадлежит')
end.
```

Программа на Бейсике

```
INPUT x,y
IF x*x+y*y>=4 THEN
IF x>=-2 THEN
IF y<=-x THEN
PRINT "принадлежит"
ELSE
PRINT "не принадлежит"
ENDIF
ENDIF
ENDIF
END
```

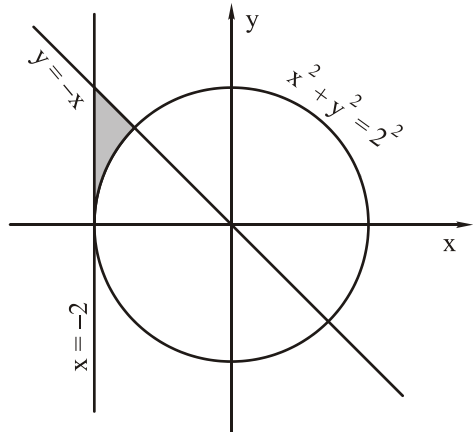


Рис. П1.2

2011 — С1

Требовалось написать программу, при выполнении которой с клавиатуры считываются координаты точки па плоскости (*x*, *y* — действительные числа) и определяется принадлежность этой точки заданной заштрихованной области (включая границы) — рис. П1.3.

Программист торопился и написал программу неправильно.

Программа на Паскале

```
var x,y: real;
begin
readln(x,y);
if y<=x then
if y<=-x then
if y>=x*x-2 then
write('принадлежит')
else
write('не принадлежит')
end.
```

Программа на Бейсике

```
INPUT x,y
IF y<=x THEN
IF y<-x THEN
IF y>=x*x-2 THEN
PRINT "принадлежит"
ELSE
PRINT "не принадлежит"
ENDIF
ENDIF
ENDIF
END
```

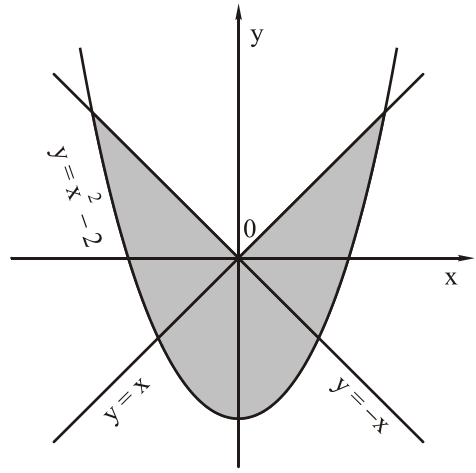


Рис. П1.3

2012 — С1

Требовалось написать программу, при выполнении которой с клавиатуры считываются координаты точки па плоскости (*x*, *y* — действительные числа) и определяется принадлежность этой точки заданной заштрихованной области (включая границы) — рис. П1.4.

Программист торопился и написал программу неправильно.

Алгоритмический язык

```
алг
нач
 вещ x,y
 ввод x,y
 если y>=-x то
   если y<=1 то
      если (x*x+y*y<=1) то
           вывод 'Принадлежит'
         иначе
           вывод 'Не принадлежит'
      все
   все
 все
кон
```

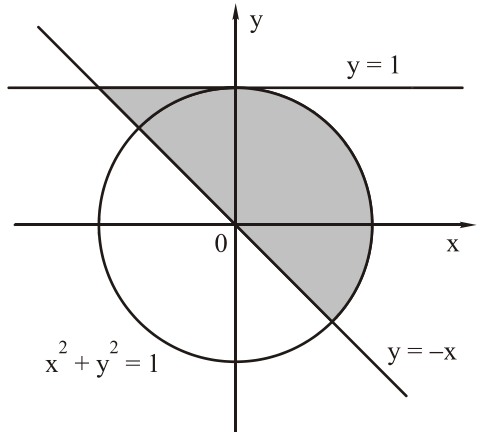


Рис. П1.4

Программа на Паскале

```
var x,y: real;
begin
readln(x,y);
if y>=-x then
if y<=1 then
if (x*x+y*y<=1) then
write('принадлежит')
else
write('не принадлежит')
end.
```

Во всех перечисленных задачах предлагалось последовательно выполнить следующее:

1. Привести пример таких чисел *x*, *y*, при которых программа неправильно решает поставленную задачу.
2. Указать, как нужно доработать программу, чтобы не было случаев ее неправильной работы. (Это можно сделать несколькими способами, поэтому можно указать любой способ доработки исходной программы.)

В задаче демонстрационного варианта 2012 года в п. 1 предлагалось также объяснить, почему для указанных чисел программа неверно решает поставленную задачу.

Рассмотрим методику решения подобных задач на примере задачи 2011 года, где требовалось проанализировать область, показанную на рис. П1.3 [1].

Главное, что желательно научиться делать, — это записывать условие, определяющее заданную заштрихованную область. Здесь надо выделить три основных момента.

В первую очередь надо уметь выделять на рисунке "подобласти" (в том числе прилегающие друг к другу или перекрывающиеся), границы которых соответствуют линиям (графикам функций) на рисунке. В нашем случае можно выделить две таких подобласти, одна из которых ограничена параболой и прямой $y = -x$, а вторая — параболой и прямой $y = x$ (рис. П1.5).

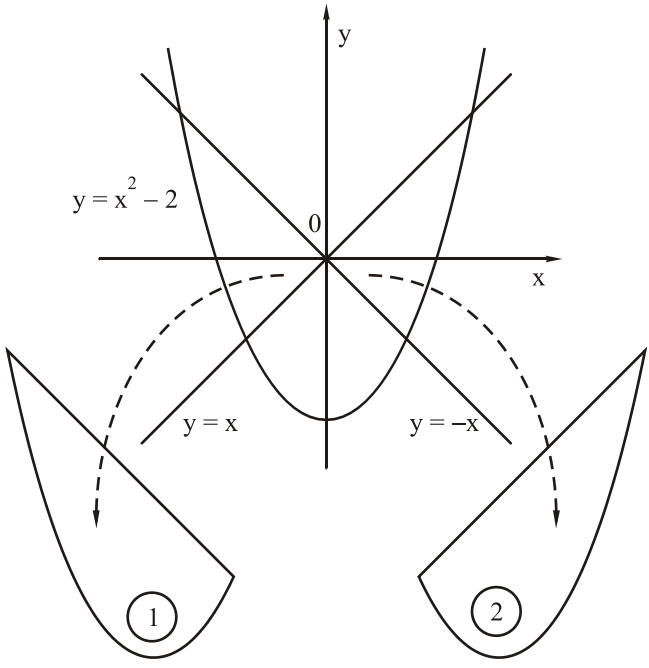


Рис. П1.5

Второй важный момент касается построения условий, описывающих каждую выделенную подобласть. Здесь надо вспомнить из курса алгебры следующее:

1. Для линии, применительно к которой можно сказать, что она разделяет плоскость на две части ("полуплоскости"[1]), расположенные выше и ниже линии графика:
   - все, что расположено ниже линии, соответствует условию $y \le f(x)$, где $f(x)$ — функция, по которой построен график;
   - все, что расположено выше линии, соответствует условию $y \ge f(x)$.

[1] Строго говоря, с математической точки зрения такие части плоскости нельзя называть полуплоскостями, но мы будем это делать для лучшего понимания методики решения задачи.

2. Для линии, применительно к которой можно сказать, что она разделяет плоскость на полуплоскости, находящиеся левее и правее линии графика:
   - все, что расположено левее линии, соответствует условию $x \le f^{-1}(y)$, где $f^{-1}(y)$ — функция, обратная заданной для графика; для ее получения достаточно в уравнении графика выразить x через y;
   - все, что расположено правее линии графика, соответствует условию $x \ge f^{-1}(y)$.
3. Для окружности:
   - все, что расположено внутри нее, соответствует условию $x^2 + y^2 \le R^2$, где R — радиус окружности;
   - все, что расположено вне ее, соответствует условию $x^2 + y^2 \ge R^2$.

Примечания

1. Нестрогость операций сравнения в приведенных условиях следует из того, что, согласно условию задачи, точки на линиях границ области (графиков) включаются в эту область.
2. Для ряда линий можно говорить о полуплоскостях как выше и ниже линии, так и левее и правее. Примером таких линий являются прямые $y = x$ и $y = -x$.

Подобласть, ограниченная несколькими линиями — графиками, представляет собой пересечение соответствующих полуплоскостей каждого графика. Очевидно, что каждая точка, принадлежащая этой подобласти, должна одновременно принадлежать всем "отдельным" полуплоскостям, — следовательно, условие, определяющее эту область, можно получить из ранее выведенных "элементарных" условий, используя логическую операцию конъюнкции (*и*, `and`).

Применительно к нашей задаче:

1. Подобласть 1 ограничена графиком параболы ($y = x^2 - 2$) и прямой $y = -x$. Причем нас интересует все, что расположено выше параболы (следовательно, первое "элементарное" условие: $y \ge x^2 - 2$), и все, что расположено ниже прямой (следовательно, второе "элементарное" условие: $y \le -x$). Значит, данная подобласть определяется логическим условием:

$$y \ge x^2 - 2 \text{ и } y \le -x.$$

2. Подобласть 2 ограничена графиком параболы ($y = x^2 - 2$) и прямой $y = x$. Причем нас интересует все, что расположено опять-таки выше параболы (следовательно, здесь первое "элементарное" условие: $y \ge x^2 - 2$), и все, что расположено ниже прямой (следовательно, второе "элементарное" условие: $y < x$). То есть эта подобласть определяется условием:

$$y \ge x^2 - 2 \text{ и } y \le x.$$

Третье, что нужно знать, касается объединения выделенных подобластей в одну требуемую в условии задачи область. Причем, как уже было сказано, эти подобласти могут прилегать друг к другу или перекрываться (как в обсуждаемом случае).

Очевидно, что для записи условия, обозначающего принадлежность точки любой из выделенных нами подобластей, нужно соединить записи условий, определяющих каждую подобласть, при помощи логической операции дизъюнкции (или, or). В нашем случае:

$$y \geq x^2 - 2 \text{ и } y \leq -x \text{ или } y \geq x^2 - 2 \text{ и } y \leq x^1.$$

Собственно, это и есть решение задачи — то самое условие, которое нужно проверить в программе и записать в единственном (но не вложенном) условном операторе.

Однако в задании ЕГЭ требуется найти ошибку в приведенном листинге, а также указать пример точек, для которых приведенная в условии программа будет работать неправильно. Поэтому далее нужно проанализировать фрагмент листинга, приведенный в условии задачи. Запишем соответствующий фрагмент на школьном алгоритмическом языке:

```
если y <= x
 то
   если y <= x
     то
       если y <= x * x – 2
         то
           вывод нс, "принадлежит"
         иначе
           вывод нс, "не принадлежит"
       все
   все
все
```

Из приведенного фрагмента видно, что ветвь **иначе** относится к последнему из стоящих перед ней служебных слов **если.**

Аналогично в листинге на языке Паскаль:

```
if y <= x then
if y <= -x then
if y >= x * x - 2 then
write('принадлежит')
else
write('не принадлежит')
```

— если цепочка из нескольких условных операторов `if ... then ...` завершается одной записью `else`, то эта ветвь `else` *всегда* относится к *последнему* из стоящих перед ней операторов `if`.

Следовательно, в ряде случаев программа вообще ничего не напечатает.

---

[1] Так как логическая операция конъюнкции (и) выполняется раньше, чем операция дизъюнкции (или), то использовать скобки не нужно. Напомним также, что в программе на языке Паскаль каждое простое условие берется в скобки.

Второе же важное правило, которое с очевидностью следует из анализа работы программы с такой цепочкой последовательно записанных вложенных друг в друга условных операторов, — что такая запись эквивалентна соединению условий, записанных в этих операторах, с помощью логической операции конъюнкции (**и, and**).

Учитывая все это, можно записать условие, запрограммированное в приведенном ранее листинге, следующим образом:

$$y \le x \text{ и } y \le -x \text{ и } y \ge x^2 - 2.$$

Если сравнить два полученных условия (сформулированное нами из анализа чертежа и "извлеченное" из листинга программы), то сразу бросается в глаза отсутствие во втором условии операции **или**.

Именно в этом и заключается ошибка листинга в условии: интуитивно (зная, что соединение "элементарных" условий через и более "строго", чем через или) нетрудно догадаться, что тем самым в программе "потеряны" какие-то части заданной области. Чтобы понять, какие именно, расшифруем второе (ошибочное) условие в виде чертежа, используя рассуждения, как бы "обратные" тем, которые мы применяли при составлении условия по чертежу:

- условие $y \le f(x)$, где $f(x)$ — функция, по которой построен график, соответствует всему, что расположено ниже линии графика;
- условие $y \ge f(x)$ соответствует всему, что расположено выше линии графика;
- условие $x \le f^{-1}(y)$, где $f^{-1}(y)$ — функция, обратная заданной для графика (для ее получения достаточно в уравнении графика выразить x через y), соответствует всему, что расположено левее линии графика;
- условие $x \ge f^{-1}(y)$ соответствует всему, что расположено правее линии графика;
- соединение этих "элементарных" условий через операцию и соответствует пересечению соответствующих полуплоскостей, а через операцию или — объединению областей (в том числе — с их взаимным наложением).

Зная все это, можно определить, что в программе было заложено условие, обозначающее принадлежность заданной точки области, которая расположена одновременно выше графика параболы и ниже обеих прямых. То есть речь идет о подобласти, которая на рис. П1.6 показана в нижней части.

А уже отсюда нетрудно догадаться, что в качестве примера точек, для которых приведенная в условии задачи программа будет давать неправильный ответ, можно брать любые "удобные" точки из боковых частей области, которые на рис. П1.7 закрашены.

Например, это могут быть точки, расположенные на оси x: (1, 0) или (–1, 0). Или, как указано в ответе к данной задаче в материалах демонстрационного варианта ЕГЭ, точка (2, 2): если решить уравнение $x^2 - 2 = 0$, чтобы определить координаты точек пересечения параболы с осью x, а также решить системы уравнений:

$$\begin{cases} y = x^2 - 2 \\ y = x \end{cases} \text{ и } \begin{cases} y = x^2 - 2 \\ y = -x \end{cases}$$

чтобы определить координаты точек пересечения заданных прямых с графиком параболы, то можно соответствующим образом "разметить" чертеж и убедиться, что точка (2, 2) находится как раз на пересечении графиков и потому попадает в правую боковую часть областей, которые ошибочная программа, так сказать, "упускает из вида" (рис. П1.8).

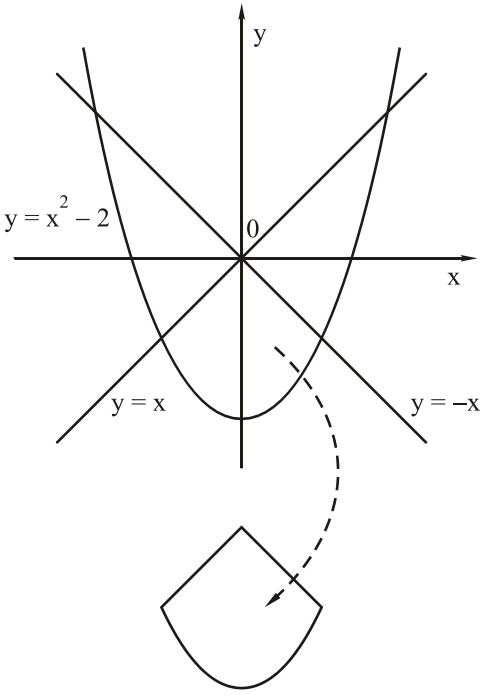


Рис. П1.6

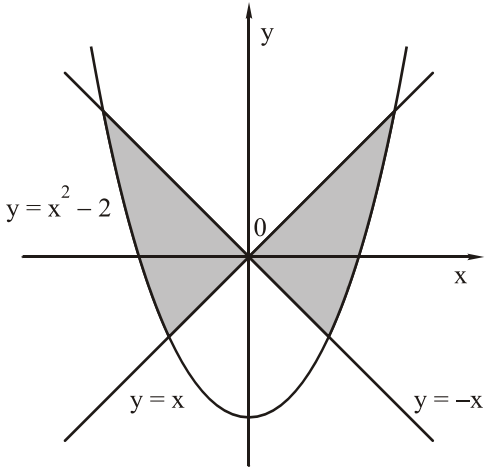


Рис. П1.7

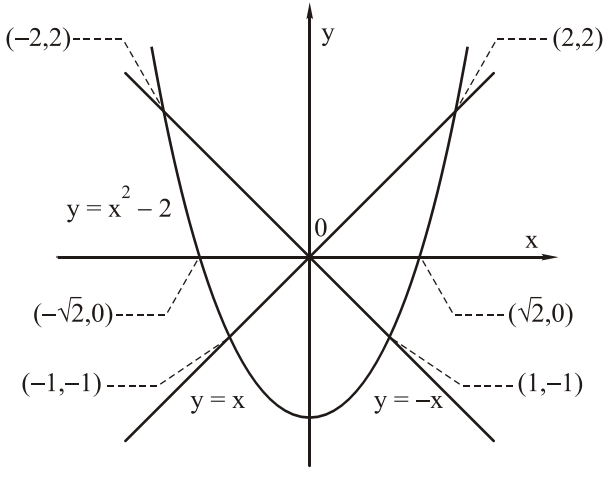


Рис. П1.8

Вторая группа точек, для которых программа выдаст неправильный результат, вообще ничего не напечатает, это точки с координатами, при которых первое или второе простое условие — ложные. Примеры таких точек:

- (2, 3) — здесь ложным является условие $y \le x$;
- (3, 2) — ложным является условие $y \le -x$.

Тем самым мы ответили на первый вопрос задачи.

А чтобы ответить на ее второй вопрос (о необходимой доработке программы), выведенное нами ранее условие принадлежности точки заданной области следует записать в одном полном условном операторе:

```
если y ≥ x - 2 и y ≤ -x или y ≥ x - 2 и y ≤ x
   то
     вывод нс, "принадлежит"
   иначе
     вывод нс, "не принадлежит"
все
```

Заметим, что приведенная в качестве примера в [6] возможная доработка:

```
if (y >= x * x - 2) and (y <= x) and (x >= 0)
  or (y >= x * x - 2) and (y <= -x) and (x <= 0)
then
write('принадлежит')
else
write('не принадлежит')
```

менее рациональная, чем наша — в ней использованы шесть простых условий (авторы в своей записи использовали еще одну границу при разбиении подобластей — ось y, которая записывается уравнением $x = 0$).

Приведем решение оставшихся задач из демовариантов ЕГЭ за 2009, 2010 и 2012 годы.

2009 — C1

Заштрихованная область образована пересечением полуплоскостей:

- ниже прямой $y = 1$;
- правее прямой (оси y) $x = 0$;
- выше синусоиды $y = \sin(x)$;
- левее вертикали $x = \pi/2 = 1{,}57$.

Обратим внимание на последнюю полуплоскость — прямая $x = \pi/2$ на чертеже не показана, поэтому о необходимости добавления этого условия часто забывают; однако если этого не сделать, то решение будет ошибочным, т. к. в него войдут и другие такие же области над синусоидой и под горизонтальной прямой $y = 1$, расположенные правее указанной области, — ведь, как нетрудно вспомнить, функция $\sin(x)$ имеет периодический характер.

Условие, определяющее принадлежность точки этой области:

$$y \ge \sin(x) \text{ и } y \le 1 \text{ и } x \ge 0 \text{ и } x \le 1.57.$$

Запись условия, запрограммированного в приведенной в условии задачи программе:

$$y \le 1 \text{ и } x \ge 0 \text{ и } y \ge \sin(x).$$

Сравнение двух условий показывает, что одна из ошибок в программе заключается в неучтенном ограничении "левее вертикали x = 1,57".

Пример точки, для которой программа дает ошибочный ответ (ошибочно относит эту точку к заданной области): (π, 1) — точка, лежащая на горизонтальной прямой y = 1. Пригодна в качестве ответа и точка (3, 0.5), приведенная в качестве примера правильного ответа в демонстрационном варианте ЕГЭ.

Пример исправления программы:

```
алг Задача_С1_2009
нач вещ x, y
  ввод x, y
  если y ≥ x – 2 и y ≤ –x или y ≥ x – 2 и y ≤ x
   то
     вывод нс, "принадлежит"
   иначе
     вывод нс, "не принадлежит"
  все
кон
```

2010 — С1

Область образована пересечением полуплоскостей:

- ниже прямой $y = -x$;
- правее прямой $x = -2$;
- вне окружности $x^2 + y^2 = 2^2$;
- выше оси x, т. е. прямой $y = 0$.

Условие, определяющее принадлежность точки этой области:

$$y \le -x \text{ и } x \ge -2 \text{ и } x^2 + y^2 \ge 4 \text{ и } y \ge 0.$$

Запись условия, запрограммированного в приведенной в условии задачи программе:

$$y \le -x \text{ и } x \ge -2 \text{ и } x^2 + y^2 \ge 4 \text{ и } y \ge 0.$$

Очевидно, что одна из ошибок в программе заключается в неучтенном ограничении "выше оси x".

Пример точки, для которой программа даст ошибочный ответ (ошибочно относит эту точку к заданной области): (0, –3) — точка, лежащая вне окружности, правее

прямой $x = -2$, ниже прямой $y = -x$, но не попадающей в указанную область. Пригодна в качестве ответа и точка (–1, –3), приведенная в качестве примера правильного ответа в демонстрационном варианте ЕГЭ.

Пример исправления программы:

```
алг Задача_С1_2010
нач вещ x, y
  ввод x, y
  если y <= -x и x >= -2 и x + y >= 4 и y >= 0
   то
     вывод нс, "принадлежит"
   иначе
     вывод нс, "не принадлежит"
  все
кон
```

2012 — С1

Область образована двумя подобластями. Первая образована пересечением полуплоскостей:

- внутри окружности $x^2 + y^2 = 1$;
- выше прямой $y = -x$;

вторая — полуплоскостей:

- вне окружности $x^2 + y^2 = 1$;
- выше прямой $y = -x$;
- ниже прямой $y = 1$;
- левее оси y, т. е. прямой $x = 0$ (обратите на это внимание!).

Условие, определяющее принадлежность точки этой области:

$$x^2 + y^2 \le 1 \text{ и } y \ge -x \text{ или } x^2 + y^2 \ge 1 \text{ и } y \ge -x \text{ и } y \le 1 \text{ и } x \le 0.$$

Можно условие $x^2 + y^2 \ge 1$ не использовать (достаточным является условие $x \le 0$):

$$x^2 + y^2 \le 1 \text{ и } y \ge -x \text{ или } y \ge -x \text{ и } y \le 1 \text{ и } x \le 0.$$

Если в условие для первой подобласти (см. чуть выше) искусственно добавить еще одно условие $x \le 0$:

$$x^2 + y^2 \le 1 \text{ и } y \ge -x \text{ и } x \le 0 \text{ или } y \ge -x \text{ и } y \le 1 \text{ и } x \le 0,$$

то, применив дистрибутивный закон логики, можно получить компактный вариант:

$$y \ge -x \text{ и } y \le 1 \text{ и } (x^2 + y^2 \le 1 \text{ или } x \le 0),$$

что совпадает с правильным ответом в демонстрационном варианте ЕГЭ.

Запись условия, запрограммированного в приведенной в условии задачи программе:

$$y \ge -x \text{ и } y \le 1 \text{ и } (x^2 + y^2 \le 1).$$

Анализ последнего условия показывает, что ему не удовлетворяют точки закрашенной области, лежащие вне окружности.

Пример точки, для которой программа даст ошибочный ответ (ошибочно не относит эту точку к заданной области): (–1, 1). Такой ответ ("не принадлежит") будет выведен для всех точек закрашенной области, о которых говорилось в п. 3.

Для всех точек, у которых $y < -x$, программа вообще не выведет никакого сообщения. Пример такой точки: (–1, 0). Это связано с тем, что при указанных значениях ложным является первое простое условие в листинге (`y >= -x`), и поэтому дальнейшие сравнения и вывод сообщений проводиться не будут.

Аналогично, вывода сообщений не будет для всех точек, у которых $y > 1$ (ложным является второе простое условие в листинге). Пример такой точки: (0, 2).

Пример исправления программы (использован "наш" вариант условия):

```
алг Задача_С1_2012
нач вещ x, y
  ввод x, y
  если x * x + y * y <= 1 и y >= -x
    или x * x + y * y >= 1 и y >= -x и y <= 1 и x <= 0
   то
     вывод нс, "принадлежит"
   иначе
     вывод нс, "не принадлежит"
  все
кон
```

В заключение напомним основные этапы решения задач типа С1:

1. Найти линии (графики функций), ограничивающие заданную область.
2. Определить уравнение каждой линии (в рассмотренных демонстрационных вариантах уравнения приводятся на рисунке).
3. Выделить на рисунке подобласти (в том числе прилегающие друг к другу или перекрывающиеся), границы которых соответствуют линиям графиков, причем желательно, чтобы было минимальным как количество таких подобластей, так и количество графиков, "участвующих" в построении каждой подобласти.
4. Записать условия, описывающие каждую полуплоскость того или иного графика (согласно приведенным выше указаниям, касающихся множества точек, находящихся выше графика, ниже, левее, правее и т. п.).
5. Объединить "элементарные" условия, указанные в п. 4, с помощью логической операции конъюнкции (**и**, **and**) — получить сложное условие, определяющее принадлежность точки той или иной подобласти.
6. Если заданная область состоит из нескольких подобластей — соединить записи условий, определяющих каждую подобласть, с помощью логической операции дизъюнкции (**или**, **or**). В результате будет получено условие, определяющее при-

надлежность точки с заданными координатами выделенной области. В ряде случаев его можно упростить, применив законы логики.

7. Записать все простые условия, приведенные в листинге программы в условии задачи, в виде одного сложного условия, используя логическую операцию конъюнкции (**и, and**).

8. Сравнить сложные условия, полученные на этапах 6 и 7, и определить "потерянную" часть заданной области. Одну из точек этой части можно привести в качестве примера чисел *x, y*, при которых программа неправильно решает поставленную задачу. Обратим также внимание на тот факт, что для ряда точек программа в условии задачи вообще не выведет никаких сообщений (см. ранее).

9. Исходная программа, измененная так, чтобы не было случаев ее неправильной работы, должна иметь следующий фрагмент, связанный с выводом ответа:

```
если <условие>
  то
    вывод нс, "принадлежит"
  иначе
    вывод нс, "не принадлежит"
все
```

где <условие> — условие, полученное на этапе 6.

Задачи для самостоятельной работы ([12])

Решите задачи типа задачи С1 из ЕГЭ для следующих рисунков и листингов программ.

1.

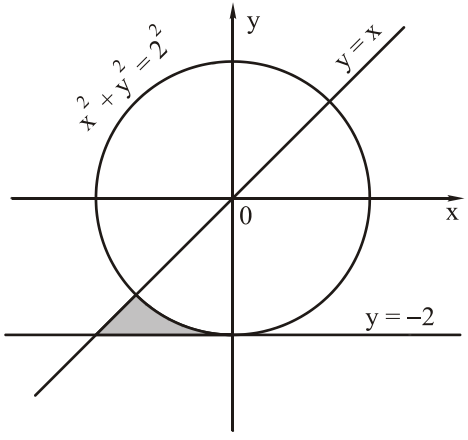


Рис. П1.9

Школьный алгоритмический язык

```
алг
нач вещ x, y
ввод x, y
если x * x + y * y >= 4
то
 если y >= -2
  то
  если y <= x
   то
    вывод "принадлежит"
   иначе
    вывод "не принадлежит"
  все
 все
все
кон
```

Язык Паскаль

```
var x, y: real;
begin
readln(x, y);
if x * x + y * y >= 4 then
if y >= -2 then
    if y <= x then
       write('принадлежит')
    else
       write('не принадлежит')
end.
```

2.

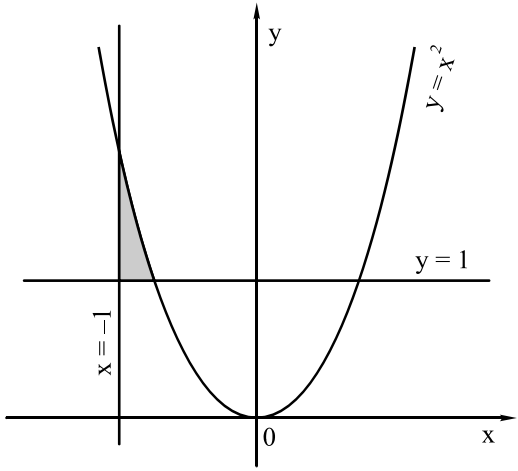


Рис. П1.10

Школьный алгоритмический язык

```
алг
нач вещ x, y
ввод x, y
если y <= x * x
то
 если x >= -1
  то
  если y >= 1
   то
    вывод "принадлежит"
   иначе
    вывод "не принадлежит"
  все
 все
все
кон
```

Язык Паскаль

```
var x, y: real;
begin
readln(x, y);
if y <= x * x then
if x >= -1 then
    if y >= 1 then
       write('принадлежит')
    else
       write('не принадлежит')
end.
```

3.

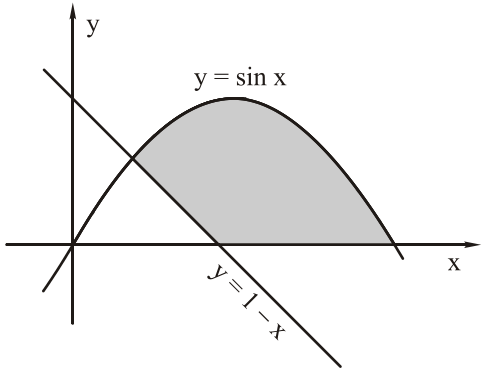


Рис. П1.11

Школьный алгоритмический язык

```
алг
нач вещ x, y
ввод x, y
если y <= sin(x)
то
 если y >= 1 − x
  то
  если y >= 0
   то
    вывод "принадлежит"
   иначе
    вывод "не принадлежит"
  все
 все
все
кон
```

Язык Паскаль

```
var x, y: real;
begin
readln(x, y);
if y <= sin(x)then
if y >= 1 − x then
    if y >= 0 then
       write('принадлежит')
    else
   write('не принадлежит')
end.
```

4.

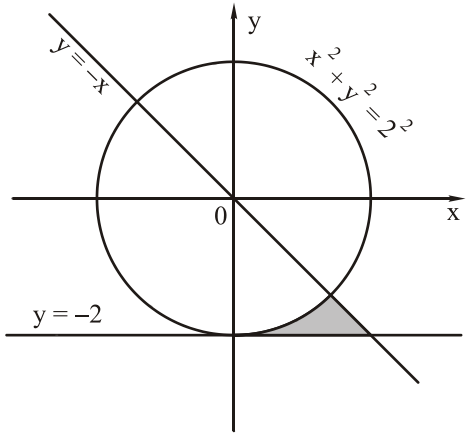


Рис. П1.12

Школьный алгоритмический язык

```
алг
нач вещ x, y
ввод x, y
если x * x >= 4
то
 если y >= -2
  то
  если y <= -x
   то
    вывод "принадлежит"
   иначе
    вывод "не принадлежит"
  все
 все
все
кон
```

Язык Паскаль

```
var x, y: real;
begin
readln(x, y);
if x * x >= 4 then
if y >= -2 then
 if y <= -x then
      write('принадлежит')
    else
   write('не принадлежит')
end.
```

5.

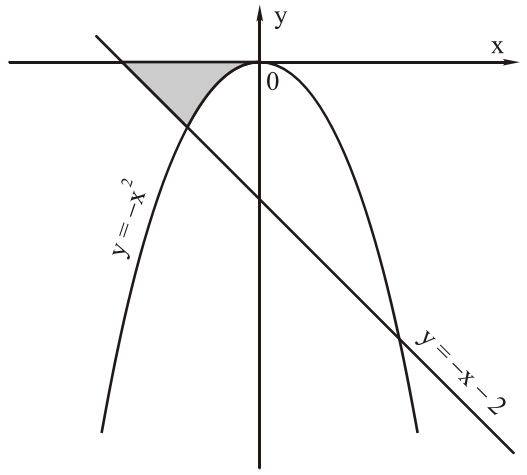


Рис. П1.13

Школьный алгоритмический язык

```
алг
нач вещ x, y
ввод x, y
если y >= -x * x
то
 если y >= -x - 2
  то
  если y <= 0
   то
    вывод "принадлежит"
   иначе
    вывод "не принадлежит"
  все
 все
все
кон
```

Язык Паскаль

```
var x, y: real;
begin
readln(x, y);
if y >= -x * x then
if y >= -x - 2 then
 if y <= 0 then
       write('принадлежит')
    else
   write('не принадлежит')
end.
```

6.

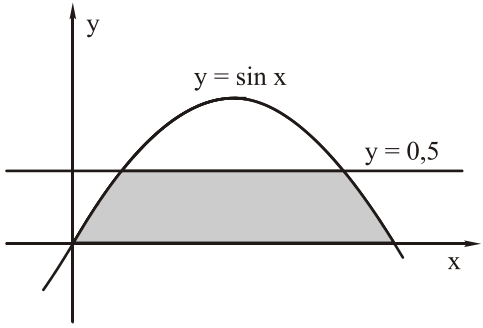


Рис. П1.14

Школьный алгоритмический язык

```
алг
нач вещ x, y
ввод x, y
если y <= sin(x)
то
 если y <= 0.5
  то
  если y >= 0
   то
    вывод "принадлежит"
   иначе
    вывод "не принадлежит"
  все
 все
все
кон
```

Язык Паскаль

```
var x, y: real;
begin
readln(x, y);
if y <= sin(x) then
if y <= 0.5 then
 if y >= 0 then
      write('принадлежит')
    else
   write('не принадлежит')
end.
```

7.

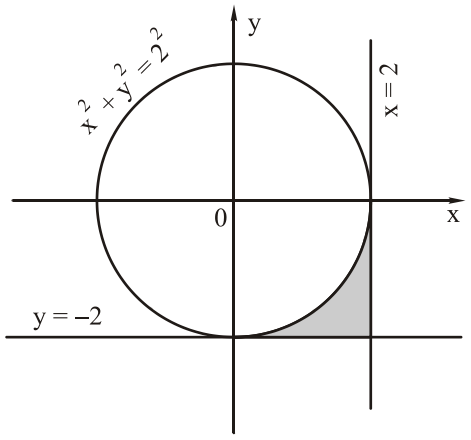


Рис. П1.15

Школьный алгоритмический язык

```
алг
нач вещ x, y
ввод x, y
если x * x + y * y >= 4
то
 если y >= -2
  то
  если x <= 2
   то
    вывод "принадлежит"
   иначе
    вывод "не принадлежит"
  все
 все
все
кон
```

Язык Паскаль

```
var x, y: real;
begin
readln(x, y);
if x * x + y * y >= 4 then
if y >= -2 then
 if x <= 2  then
      write('принадлежит')
    else
   write('не принадлежит')
end.
```

8.

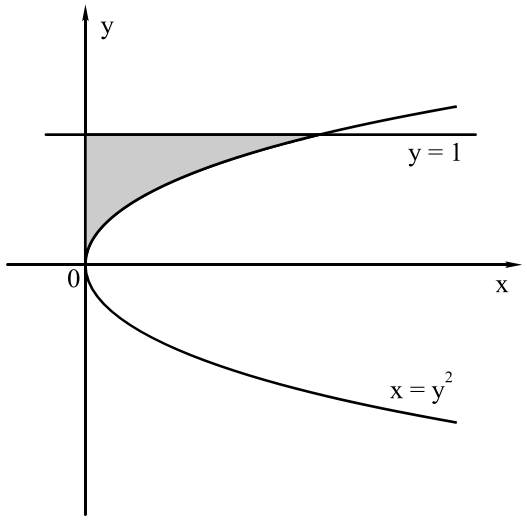


Рис. П1.16

Школьный алгоритмический язык

```
алг
нач вещ x, y
ввод x, y
если x <= y * y
то
 если x >= 0
  то
  если y <= 1
   то
    вывод "принадлежит"
   иначе
    вывод "не принадлежит"
  все
 все
все
кон
```

Язык Паскаль

```
var x, y: real;
begin
readln(x, y);
if x <= y * y then
if x >= 0 then
 if y <= 1 then
       write('принадлежит')
    else
   write('не принадлежит')
end.
```

9.

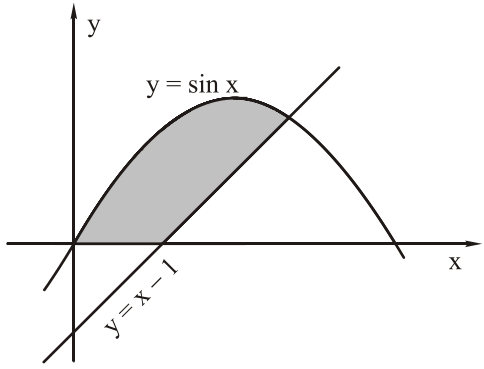


Рис. П1.17

Школьный алгоритмический язык

```
алг
нач вещ x, y
ввод x, y
если y <= sin(x)
то
 если y >= x - 1
  то
  если y >= 0
   то
    вывод "принадлежит"
   иначе
    вывод "не принадлежит"
  все
 все
все
кон
```

Язык Паскаль

```
var x, y: real;
begin
readln(x, y);
if y <= sin(x) then
if y >= x - 1 then
 if y >= 0 then
       write('принадлежит')
    else
   write('не принадлежит')
end.
```

10.

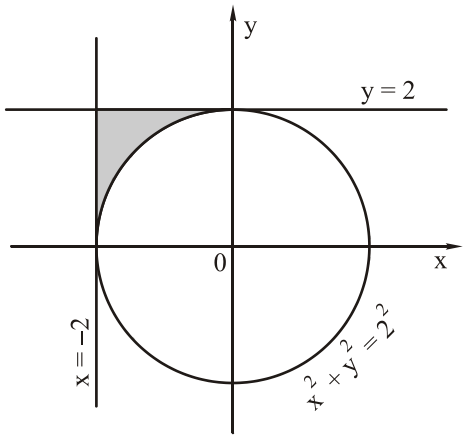


Рис. П1.18

Школьный алгоритмический язык

```
алг
нач вещ x, y
ввод x, y
если x * x + y * y >= 4
то
 если y >= −2
  то
  если x <= 2
   то
     вывод "принадлежит"
   иначе
     вывод "не принадлежит"
  все
 все
все
кон
```

Язык Паскаль

```
var x, y: real;
begin
readln(x, y);
if x * x + y * y >= 4 then
if y >= −2 then
 if x <= 2 then
      write('принадлежит')
    else
   write('не принадлежит')
end.
```

ПРИЛОЖЕНИЕ 2

Задачи на определение значений переменных величин

В частях 1 и 2 ЕГЭ представлены задачи, в которых требуется определить значение переменной величины или элементов массива после выполнения некоторого фрагмента программы или блок-схемы алгоритма. Их можно разделить на ряд групп.

П2.1. Задачи, реализующие линейный алгоритм

ПРИМЕР 1. Определите значение переменной `c` после выполнения следующего фрагмента программы:

Школьный алгоритмический язык

```
a := 5
a := a + 6
b := –a
c := a – 2 * b
```

Язык Паскаль

```
a := 5;
a := a + 6;
b := –a;
c := a – 2 * b;
```

Решение

Задачи такого типа удобно решать, используя таблицу, имитирующую "трассировку"[1] программы, в которую последовательно записывать значения переменных после выполнения каждого оператора (табл. П2.1).

Таблица П2.1

a	b	c
5	0	0
5+6=11	0	0
11	–11	0
11	–11	11–2*11

Ответ: –11.

[1] Трассировка — это процесс выполнения программы по шагам, инструкция за инструкцией.

П2.2. Задачи, реализующие разветвляющийся алгоритм

ПРИМЕР 2. Определите значение переменной `c` после выполнения следующего фрагмента программы:

Школьный алгоритмический язык

```
a := 40
b := 10
b := -a/2 * b
если a < b
 то c := b — a
 иначе c: = a - 2 * b
все
```

Язык Паскаль

```
a := 40;
b := 10;
b := -a/2 * b;
if a < b then c := b — a
else c := a - 2 * b
```

Решение

Для таких задач таблица, имитирующая "трассировку" программы, должна учитывать результат проверки условия, представленного в заданном фрагменте программы, и соответствующую ветвь условного оператора (табл. П2.2).

Таблица П2.2

a	b	c
40	0	0
40	10	0
40	–40/2*10=–200	
		a<b? — нет
		40 –2*(–200)=440

Ответ: 440.

ПРИМЕР 3. Определите значение переменной `a` после выполнения следующего фрагмента алгоритма (рис. П2.1).

Здесь следует последовательно проверить все (при необходимости) условия (табл. П2.3).

Таблица П2.3

a	b	
56	77	a = b? — нет
		a > b? — нет
	77-56=21	

Ответ: 56.

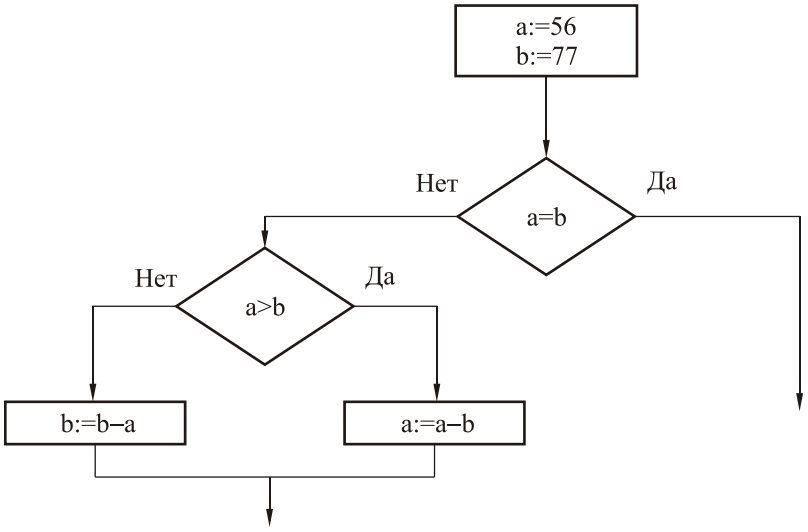


Рис. П2.1

П2.3. Задачи, реализующие циклический алгоритм

ПРИМЕР 4. Определите значение переменной `s` после выполнения следующего фрагмента алгоритма (рис. П2.2).

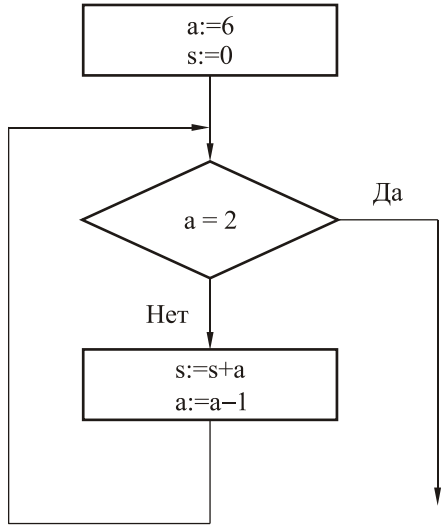


Рис. П2.2

Решение

Анализ показывает, что на рис. П2.2 применен цикл с условием. Поэтому в таблице "трассировки" следует проверять соответствующее условие многократно (табл. П2.4).

Таблица П2.4

a	s	a = 2?
6	0	Нет
	0+6=6	
6–1=5		Нет
	6+5=11	
5–1=4		Нет
	11+4=15	
4–1=3		Нет
	15+3=18	
3–1=2		Да

Ответ: 18.

При большой разности между значением переменной в условии для проверки и ее исходным значением и при небольшом шаге изменения этой переменной таблица становится громоздкой. Например, это имеет место в следующем примере.

ПРИМЕР 5. Определите, что будет напечатано в результате работы следующего фрагмента программы:

Школьный алгоритмический язык

```
Нач
цел k, s
s := 0
k := 0
нц пока s < 1024
   s := s + 10; k := k + 1
кц
вывод k
кон
```

Язык Паскаль

```
var k, s: integer;
begin
s := 0;
k := 0;
while s < 1024 do
begin
s := s + 10; k := k + 1
end;
write(k)
end.
```

Решение

Видно, что значение `s` превысит 1024 после достаточно большого числа повторений тела оператора цикла. Следует определить — после какого?

Ответ — после 103-го (после 102 повторений значение `s` будет равно 1020, после чего тело оператора выполнится еще 1 раз, и условие `s < 1024` станет ложным). А поскольку при каждом выполнении тела оператора цикла значение `k` увеличивается на 1, то его окончательное значение будет равно 103.

Ответ: 103.

Возможны и "промежуточные варианты" сложности заданий.

ПРИМЕР 6. Определите значение переменной `b` после выполнения следующего фрагмента алгоритма (рис. П2.3).

Решение

Здесь количество повторений тела цикла можно определить "в уме" (оно равно 8), однако выражение для расчета значения переменной `b` таково, что необходимые расчеты приходится проводить с помощью таблицы (табл. П2.5).

Таблица П2.5

a	**b**	**a = 1?**
256		
	0	Нет
128		
	0 + 128 + 1 = 129	Нет
64		
	129 + 64 + 1 = 194	Нет
32		
	194 + 32 + 1 = 227	Нет
16		
	227 + 16 + 1 = 244	Нет
8		
	244 + 8 + 1 = 253	Нет
4		
	253 + 4 + 1 = 258	Нет
2		
	258 + 2 + 1 = 261	Нет
1		
	261 + 1 + 1 = 263	Да

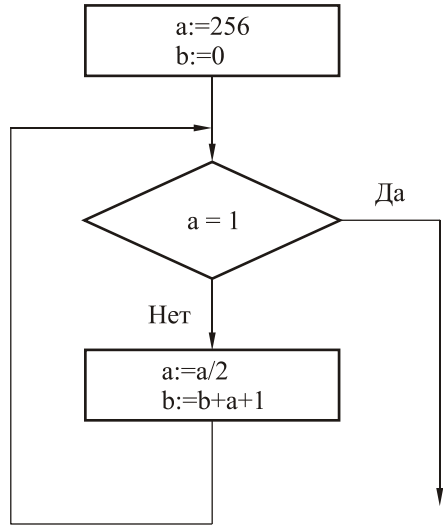


Рис. П2.3

Ответ: 263.

Обратим внимание, что последняя (как и все другие) проверка условия `a = 1` проводится *после* расчета значений `a` и `b`. Заметим также, что в случае введения компьютерного варианта ЕГЭ для расчетов, аналогичных проводимым согласно табл. П2.5, можно будет применить электронную таблицу Microsoft Excel или подобную.

П2.4. Задачи, реализующие алгоритмы различных типов

Решение задач такого типа можно проводить, используя таблицы, описанные применительно к задачам, рассмотренным ранее.

Мы же подробно обсудим методику решения задач, аналогичных задаче А14 из демонстрационного варианта ЕГЭ 2012 года [7].

ПРИМЕР 7. Определите, какое число будет напечатано после выполнения следующего фрагмента программы:

Школьный алгоритмический язык

```
алг
нач вещ d, a, b, t, M, R
a := -3; b := 3
d := 0.1
t := a; M := a; R:=F(a)
нц пока t < b
  если F(t) < R
    то
     M := t; R := F(t)
  все
  t := t + d
кц
вывод M
кон
алг вещ F (вещ x)
нач
  знач := (x - 1) * (x - 3)
вещ
```

Язык Паскаль

```
var d, a, b, t, M, R: real;
function F(x: real): real;
begin
F := (x - 1) * (x - 3)
end;
begin
a := -3; b := 3;
d := 0.1;
t := a; M := a; R:=F(a);
while t < b do
  begin
    if F(t) < R then
      begin
       M := t; R := F(t)
      end;
   t := t + d
  end;
write(M)
end.
```

Решение

Заметим прежде всего, что здесь впервые в задании ЕГЭ по программированию появилась задача, связанная с использованием вспомогательных функций.

Полная трассировка программы в данном случае крайне трудоемка — тело оператора цикла будет выполняться 60 раз (начальное значение `t` равно –3, и оно увеличивается до 2,9 с шагом 0,1; величина `b` не меняется).

Итак, условие `t < b` будет истинным пока $t \le 2{,}9$. Но означает ли это, что окончательное значение величины `M` также будет равно 2,9? Для ответа на этот вопрос следует знать, сколько раз будет меняться значение `M` в "теле" условного оператора, или, по-другому, при каком значении `t` прекратит выполняться это "тело", т. е. когда станет ложным условие `F(t) < R`.

Проанализируем это условие. Из программы видно, что значение `R` тоже равно значению функции, т. е. тоже зависит от `t`, но при каждом очередном выполнении тела

цикла (на каждой итерации) — от предыдущего значения `t`. Значит, нужно определить, когда значение функции $F(t)$ станет больше, чем на предыдущем шаге[1]. Для этого необходимо исследовать заданную функцию, которую можно представить в виде $x^2 - 4x + 3 = 0$. Подробное исследование мы проводить не будем (11-классники должны уметь сделать это самостоятельно), а приведем только график указанной функции (рис. П2.4).

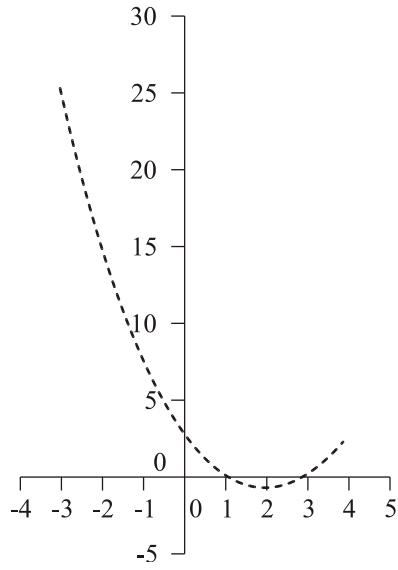


Рис. П2.4

Анализ графика показывает, что последнее значение `t`, при котором `F(t) < R`, равно 2,0. Это и будет окончательным значением величины `M`.

Ответ: 2.

П2.5. Задачи на заполнение и изменение одномерного массива

ПРИМЕР 8. В программе используется одномерный целочисленный массив `A` с индексами от 0 до 9. Ниже представлен фрагмент программы, в котором значения элементов сначала задаются, а затем меняются.

Школьный алгоритмический язык

```
нц для i от 0 до 9
  A[i] := 9 - i
кц
нц для i от 0 до 4
  k := A[i]
  A[i] := A[9 - i]
  A[9 - i] := k
кц
```

Язык Паскаль

```
for i := 0 to 9 do
  A[i] := 9 - i;
for i := 0 to 4 do
 begin
  k := A[i];
  A[i] := A[9 - i];
  A[9 - i] := k
 end
```

Чему будут равны элементы этого массива после выполнения фрагмента программы?

[1] Иными словами, нужно найти минимум функции $F(t)$, точнее — значение аргумента, при котором этот минимум достигается.

Решение

Задачи такого типа целесообразно решать в два этапа.

1. Проанализировать первый оператор цикла — в нем проводится заполнение массива `A`. Оформим таблицу, моделирующую массив, и запишем в нее значения элементов после выполнения первого оператора цикла:

i	0	1	2	3	4	5	6	7	8	9
***A*[*i*]**	9	8	7	6	5	4	3	2	1	0

Примечание

Строку с номерами индексов желательно сделать первой.

2. Исследовать второй оператор цикла. Его анализ показывает, что в нем проводится обмен значениями элементов с индексами `i` ($i = 0, 1, 2, 3, 4$) и $(9 - i)$. На этом этапе целесообразно:

   - выписать пары индексов элементов, для которых будет проводиться обмен значениями:

     0 и 9, 1 и 8, 2 и 7, 3 и 6, 4 и 5;

   - к приведенной только что таблице добавить еще одну строку и провести в ней обмен значениями соответствующих элементов:

i	0	1	2	3	4	5	6	7	8	9
***A*[*i*]**	9	8	7	6	5	4	3	2	1	0
***A*[*i*]**	0	1	2	3	4	5	6	7	8	9

Ответ: 0 1 2 3 4 5 6 7 8 9.

Примечание

При достаточном опыте можно также сразу обнаружить, что поскольку индекс `i` во втором операторе цикла меняется до "половины массива", то в результате произойдет перестановка элементов массива в обратном порядке *(см. задачу 1.8.3)*.

ПРИМЕР 9. В программе используется одномерный целочисленный массив `A` с индексами от 0 до 10. Ниже представлен фрагмент программы, в котором значения элементов сначала задаются, а затем меняются.

Школьный алгоритмический язык

```
нц для i от 0 до 10
  A[i] := 10 - i
кц
нц для i от 0 до 10
  A[10 - i] := A[i]
  A[i] := A[10 - i]
кц
```

Язык Паскаль

```
for i := 0 to 10 do
  A[i] := 10 - i;
for i := 0 to 10 do
 begin
  A[10 - i] := A[i];
  A[i] := A[10 - i]
 end
```

Чему будут равны элементы этого массива после выполнения фрагмента программы?

Решение

Согласно рекомендациям, сделанным применительно к предыдущей задаче, таблица, моделирующая массив A после выполнения первого оператора цикла, имеет вид:

i	0	1	2	3	4	5	6	7	8	9	10
***A*[*i*]**	10	9	8	7	6	5	4	3	2	1	0

а в ходе и после выполнения второго:

i	0	1	2	3	4	5	6	7	8	9	10	
***A*[*i*]**	10	9	8	7	6	5	4	3	2	1	0	
***A*[*i*]**	...	...	...	...	...	...	...	...	...	9	10	После выполнения оператора A[10 – i] := A[i]
***A*[*i*]**	10	9	...	...	...	...	...	...	...	...	...	После выполнения оператора A[i] := A[10 – i]

Видно, что в последней таблице добавлен один столбец.

Ответ: 10 9 8 6 5 4 3 2 1 0.

> ***ПРИМЕЧАНИЕ***
>
> Здесь также можно сразу заметить, что каждый элемент будет менять значения дважды, и в результате массив не изменится *(см. задачу 1.8.3)*.

П2.6. Задачи на обработку одномерного массива

ПРИМЕР 10. Дан фрагмент программы, обрабатывающей массив A из n элементов с индексами от 1 до n:

Школьный алгоритмический язык

```
j := 1
нц для i от 2 до n
 если A[i] > A[j]
  то
    j := i
 все
кц
s := A[j]
```

Язык Паскаль

```
j := 1;
for i := 2 to n do
 if A[i] > A[j] then
    j := i;
s := A[j]
```

Чему будет равно значение переменной s после выполнения данного фрагмента при любых значениях элементов массива A:

1. Индексу максимального элемента в массиве A (первому из них, если максимальных элементов несколько).

2. Второму максимальному элементу в массиве A.
3. Максимальному элементу в массиве A.
4. Минимальному элементу в массиве A.

Решение

Здесь надо вспомнить типовые задачи обработки одномерных массивов *(см. разд. 1.9)*.

Анализ показывает, что во фрагменте происходит запоминание индекса элемента с максимальным значением среди рассмотренных, а после прохода по массиву выводится соответствующее максимальное значение.

Ответ: 3.

Другие возможные варианты значения переменной s в задачах такого типа (кроме вариантов 1–4, приведенных в условии):

1. Индекс максимального элемента в массиве (последнему из них, если максимальных элементов несколько).
2. Индекс минимального элемента в массиве (первому из них, если максимальных элементов несколько).
3. Индекс минимального элемента в массиве (последнему из них, если максимальных элементов несколько).
4. Минимальный элемент в массиве.
5. Второй минимальный элемент в массиве.
6. Сумма значений элементов массива, удовлетворяющих заданному условию.
7. Количество элементов массива, удовлетворяющих заданному условию.
8. Среднее арифметическое значений элементов массива, удовлетворяющих заданному условию.

П2.7. Задачи на заполнение двух массивов

ПРИМЕР 11. Значения двух массивов A и B с индексами от 1 до 100 задаются с помощью следующего фрагмента программы:

Школьный алгоритмический язык

```
нц для i от 1 до 100
 A[i] := i * i
кц
нц для i от 1 до 100
 B[i] := A[i] - 100
кц
```

Язык Паскаль

```
for i := 1 to 100 do
  A[i] := i * i;
for i := 1 to 100 do
  B[i] := A[i] — 100;
```

Какое количество элементов массива B будут иметь положительные значения после выполнения данного фрагмента?

Решение

Сначала надо заполнить таблицу, моделирующую массив A после выполнения первого оператора цикла:

i	1	2	3	...	99	100
***A*[*i*]**	1	4	9	...	9801	10000

Анализ второго оператора цикла показывает, что следует дополнительно исследовать "окрестности" массива в районе индекса, равного 10 (для которого `A[i] = 100`), дополнив таблицу еще одной строкой:

i	1	2	3	...	9	10	11	...	99	100
***A*[*i*]**	1	4	9	...	81	100	121	...	9801	10000
***B*[*i*] = *A*[*i*] – 100**	–99	–96	–91	...	–19	0	21	...		

Из последней таблицы следует ответ: 100 – 10 = 90.

Ответ: 90.

П2.8. Задачи на заполнение и изменение двумерного массива

ПРИМЕР 12. Значения двумерного массива A размером 9×9 задаются с помощью вложенного оператора цикла в представленном фрагменте программы:

Школьный алгоритмический язык

```
нц для i от 1 до 9
  нц для j от 1 до 9
    A[i, j] := n + k – 1
  кц
кц
```

Язык Паскаль

```
for i := 1 to 9 do
  for j := 1 to 9 do
    A[i, j] := n + k – 1
```

Сколько элементов массива A будут принимать четное значение?

Решение

Использовать для решения таблицу — модель массива и заполнять ее согласно заданному выражению — трудоемко (общее число рассчитываемых значений равно 81). Задание можно выполнить методом рассуждений.

Значение выражения $n + k - 1$ будет четным, когда:

1) n — четное, k — нечетное;

2) k — четное, n — нечетное.

Число четных значений n равно 4, нечетных значений k — 5. Значит, в первом случае общее число элементов массива с четным значением будет равно 20. Аналогично и для второго случая.

Ответ: 40.

ПРИМЕР 13. Элементы двумерного массива A размером N × N первоначально были равны 1000. Затем значения некоторых из них меняются с помощью вложенного оператора цикла в представленном фрагменте программы:

Школьный алгоритмический язык

```
k := 0
нц для i от 1 до N
  нц для j от N – i + 1 до N
    k := k + 1
    A[i, j] := k
  кц
кц
```

Язык Паскаль

```
k := 0;
for i := 1 to N do
  for j := N – i + 1 to N do
    begin
      k := k + 1;
      A[i, j] := k
    end
```

Какой элемент массива в результате будет иметь минимальное значение?

Решение

В данном случае не все элементы массива меняют значение. Необходимо проанализировать — какие?

Ответ на этот вопрос можно получить с помощью таблицы:

***i* = 1**		...			√
***i* = 2**		...		√	√
***i* = 3**		...	√	√	√
...		...			
i* = *N	√	√	√	√	√

При этом первый меняющийся элемент равен 1, каждый очередной — на 1 больше предыдущего. Значит, минимальное значение (равное 1) будет у элемента в верхнем правом углу таблицы.

Ответ: A[1, N].

ПРИМЕР 14. Дан фрагмент программы, обрабатывающей двумерный массив A размера n × n:

Школьный алгоритмический язык

```
k := 1
нц для i от 1 до n
     c := A[i, i]
     A[i, i] := A[k, i]
     A[k, i] := c
кц
```

Язык Паскаль

```
k := 1;
for i := 1 to n do
    begin
      c := A[i, i];
      A[i, i] := A[k, i];
      A[k, i] := c
    End
```

Представим массив в виде квадратной таблицы, в которой для элемента массива `A[i, j]` величина `i` является номером строки, а величина `j` — номером столбца, в котором расположен элемент. Какую задачу решает данный фрагмент?

Решение

Видно, что происходит обмен значениями ряда элементов. Каких именно? Так как один из элементов, меняющих значения, имеет индексы `[i, i]`, то это элемент главной диагонали[1]. У второго меняющего значения элемента первый индекс (`k`) постоянен, значит, это элемент `k`-й строки.

Ответ: фрагмент программы решает задачу обмена значениями в каждом столбце массива элементов главной диагонали и `k`-й строки таблицы.

Приведем краткую информацию, позволяющую определить, какие элементы меняются значениями при различных вариантах (табл. П2.6):

Таблица П2.6

№	Элементы, используемые в теле цикла при обмене	Происходит обмен значениями
1	$A[k, i]$ и $A[s, i]$	Элементов *k*-й и *s*-й строк
2	$A[i, k]$ и $A[i, s]$	Элементов *k*-го и *s*-го столбцов
3	$A[i, i]$ и $A[i, n - i + 1]$	Элементов главной и побочной[2] диагоналей в каждой строке
4	$A[i, i]$ и $A[n - i + 1, i]$	Элементов главной и побочной диагоналей в каждом столбце
5	$A[i, i]$ и $A[k, i]$	Элементов главной диагонали и *k*-й строки в каждом столбце (см. выше)
6	$A[i, i]$ и $A[i, k]$	Элементов главной диагонали и *k*-го столбца в каждой строке
7	$A[n - i + 1, i]$ и $A[k, i]$	Элементов побочной диагонали и *k*-й строки в каждом столбце
8	$A[n - i + 1, i]$ и $A[i, k]$	Элементов побочной диагонали и *k*-го столбца в каждой строке

Примечание

Во всех случаях параметр цикла `i` меняется от 1 до `n`.

[1] Главную диагональ квадратного двумерного массива образуют элементы, расположенные между верхним левым и нижним правым элементами (включая эти два элемента).

[2] Побочную диагональ квадратного двумерного массива образуют элементы, находящиеся между верхним правым и нижним левым элементами (включая эти два элемента).

Задания для самостоятельной работы

1. Определите значение переменной `c` после выполнения следующего фрагмента программы:

Школьный алгоритмический язык

```
b := -5
b := b + 6
a := –b
c := a + 2 * b
```

Язык Паскаль

```
b := -5;
b := b + 6;
a := –b;
c := a + 2 * b
```

2. Определите значение целочисленных переменных `a` и `b` после выполнения следующего фрагмента программы:

Школьный алгоритмический язык

```
a := 42
b := 14
a := div(a, b)
b := a * b
a := div(b, a)
```

Язык Паскаль

```
a := 42;
b := 14;
a := a div b;
b := a * b;
a := b div a
```

3. Определите значение переменной `x` после выполнения следующего фрагмента программы:

Школьный алгоритмический язык

```
x := 10
y := 30
x := y – x * 2
если x < y
   то
     x := y – x
   иначе
     x := x – y
все
```

Язык Паскаль

```
x := 10;
y := 30;
x := y – x * 2;
if x < y then
     x := y – x
   else
     x := x – y
```

4. Определите значение переменной `x` после выполнения следующего фрагмента алгоритма, заданного в виде блок-схемы (рис. П2.5).

5. Определите значение переменной `a` после выполнения следующего фрагмента алгоритма, заданного в виде блок-схемы (рис. П2.6).

Решите задачу двумя способами:

1) применив полную трассировку программы;

2) не используя полную трассировку *(см. разд. П2.3)*.

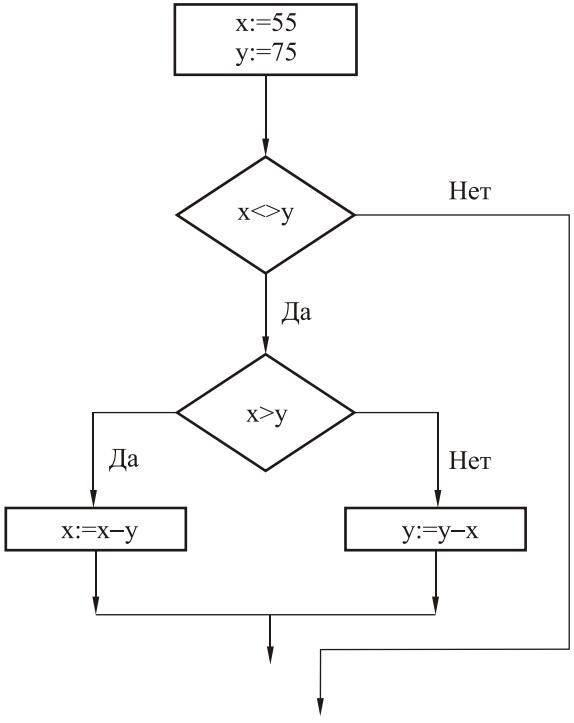


Рис. П2.5

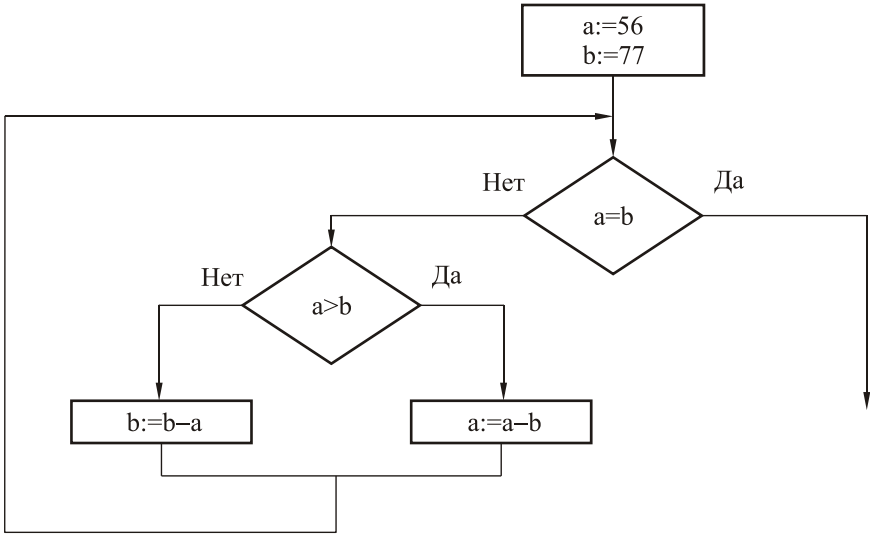


Рис. П2.6

6. Определите значение переменной `s` после выполнения следующего фрагмента алгоритма, заданного в виде блок-схемы (рис. П2.7).

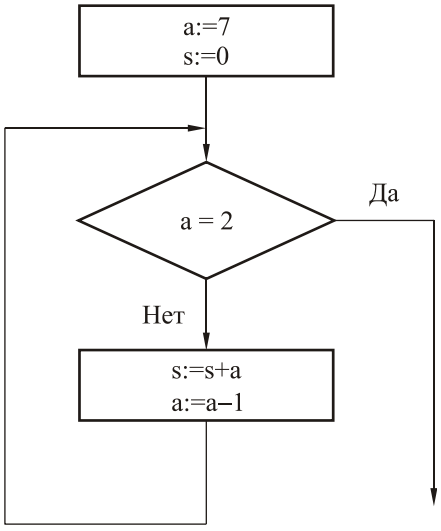


Рис. П2.7

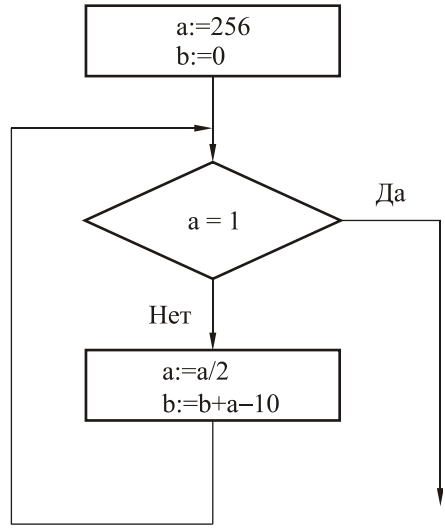


Рис. П2.8

7. Определите значение переменной `b` после выполнения следующего фрагмента алгоритма, заданного в виде блок-схемы (рис. П2.8).

8. Определите, какое число будет напечатано после выполнения следующего фрагмента программы:

Школьный алгоритмический язык

```
алг
нач вещ k, d, a, b, z, N, R
a := –2; b := 2
d := 0.1
z := a; N := a; R := F(a)
нц пока z < b
  если F(z) < R
    то
     N := z; R := F(z)
  все
  z := z + d
кц
вывод N
кон
алг вещ F (вещ x)
нач
  знач := x * x – 2 * x – 1
кон
```

Язык Паскаль

```
var k, d, a, b, z, N, R: real;
function F(x: real): real;
begin
  F := x * x – 2 * x – 1
end;
begin
a := –2; b := 2;
d := 0.1;
z := a; N := a; R := F(a);
while z < b do
  begin
   if F(z) < R then
     begin
       N := z; R := F(z)
     end;
   z := z + d
  end;
write(K)
end.
```

9. Определите, какое число будет напечатано после выполнения следующего фрагмента программы:

Школьный алгоритмический язык

```
алг
нач вещ d, a, b, y, K, S
a := -1; b := 5
d := 0.1
y := a; K := a; S := F(a)
нц пока y < b
  если F(y) > S
   то
     K := y; S := F(y)
  все
  y := y + d
кц
вывод K
кон
алг вещ F (вещ x)
нач
  знач := x * (4 — x)
кон
```

Язык Паскаль

```
var d, a, b, y, K, S: real;
function F(x: real): real;
begin
  F := x * (4 - x)
end;
begin
a := -1; b := 5;
d := 0.1;
y := a; K := a; S := F(a);
while y < b do
  begin
   if F(y) > S then
     begin
       K := y; S := F(y)
     end;
     y := y + d
  end;
write(K)
end.
```

10. В программе описан одномерный целочисленный массив A с индексами от 0 до 10 и целочисленные переменные k, i. Ниже представлен фрагмент программы, в котором значения элементов сначала задаются, а затем меняются.

Школьный алгоритмический язык

```
нц для i от 0 до 10
  A[i] := i - 1
кц
нц для i от 10 до 1 шаг -1
  A[i - 1] := A[i]
Кц
```

Язык Паскаль

```
for i := 0 to 10 do
  A[i] := i - 1;
for i := 10 downto 1 do
  A[i - 1] := A[i ];
```

Чему будут равны элементы этого массива после выполнения фрагмента программы?

11. Дан фрагмент программы, обрабатывающей массив A из n элементов с индексами от 1 до n (известно, что в массиве имеются положительные элементы):

Школьный алгоритмический язык

```
s := 0
k := 0
нц для i от 1 до n
 если A[i] > 0
  то
   s := s + A[i]
   k := k + 1
 все
кц
s := s/k
```

Язык Паскаль

```
s := 0;
k := 0;
for i := 1 to n do
  if A[i] > 0 then
    begin
      s := s + A[i]
      k := k + 1
    end;
s := s/k
```

Чему будет равно значение переменной s после выполнения данного фрагмента при любых значениях элементов массива A:

1) количеству положительных элементов в массиве A;
2) среднему арифметическому всех элементов массива A;
3) среднему арифметическому всех положительных элементов массива A;
4) сумме значений всех положительных элементов массива A?

12. Значения двух массивов A и B с индексами от 1 до 100 задаются с помощью следующего фрагмента программы:

Школьный алгоритмический язык

```
нц для n от 1 до 100
 A[n] := (n - 80) * (n - 80)
кц
нц для n от 1 до 100
 B[101 - n] := A[n]
кц
```

Язык Паскаль

```
for n := 1 to 100 do
   A[n] := (n - 80) * (n - 80)
for n := 1 to 100 do
   B[101 — n] := A[n]
```

Какой элемент массива B будет наибольшим?

13. Значения двумерного массива B размера 7×7 задаются с помощью вложенного оператора цикла в представленном фрагменте программы:

Школьный алгоритмический язык

```
нц для n от 1 до 7
  нц для k от 1 до 7
    B[n, k] := k - n
  кц
кц
```

Язык Паскаль

```
for n := 1 to 7 do
for k := 1 to 7 do
   B[n, k] := k - n;
```

Сколько элементов массива B будут иметь положительные значения?

14. Значения двумерного массива A размером 9×9 задаются с помощью вложенного оператора цикла в представленном фрагменте программы:

Школьный алгоритмический язык

```
нц для i от 1 до 10
  нц для j от 1 до 10
    A[i, j] := n + k + 1
  кц
кц
```

Язык Паскаль

```
for i := 1 to 10 do
  for j := 1 to 10 do
    A[i, j] := n + k + 1
```

Сколько элементов массива A будут принимать нечетное значение?

15. Элементы двумерного массива A размером N × N первоначально были равны –10. Затем значения некоторых из них меняются с помощью вложенного оператора цикла в представленном фрагменте программы:

Школьный алгоритмический язык

```
k := 0
нц для i от 1 до N
  нц для j от 1 до i
    k := k + 1
    A[i, j] := k
  кц
кц
```

Язык Паскаль

```
k := 0;
for i := 1 to N do
  for j := 1 to i do
    begin
     k := k + 1;
     A[i, j] := k
    end
```

Какой элемент массива в результате будет иметь максимальное значение?

16. Дан фрагмент программы, обрабатывающей двумерный массив A размера n × n:

Школьный алгоритмический язык

```
k := 1
нц для i от 1 до n
   c := A[n – i + 1, i]
   A[n – i + 1, i] := A[k, i]
   A[k, i] := c
кц
```

Язык Паскаль

```
k := 1;
for i := 1 to n do
    begin
      c := A[n – i + 1, i];
      A[n – i + 1, i] := A[k, i];
      A[k, i] := c
    end
```

Представим массив в виде квадратной таблицы, в которой для элемента массива A[i, j] величина i является номером строки, а величина j — номером столбца, в котором расположен элемент. Какую задачу решает данный фрагмент?

ПРИЛОЖЕНИЕ 3

Методы заполнения числовых массивов

Возможны несколько способов заполнения числовых массивов.

П3.1. Заполнение массива разными значениями, не подчиняющимися общему закону

Если все значения элементов разные и не подчиняются общему закону, то для заполнения массива используются `n` операторов присваивания:

```
m[1] := ...
m[2] := ...
...
m[n] := ...
```

Здесь и далее `n` — число элементов массива (индексы элементов от 1 до `n`).

Если при каждом запуске программы значения элементов меняются, целесообразно для заполнения использовать оператор ввода данных и делать это следует в цикле:

```
нц для i от 1 до n
  ввод m[i]
кц
```

Однако при таком оформлении пользователю нужно контролировать количество уже введенных значений. Более рациональный вариант:

```
нц для i от 1 до n
  вывод нс, "Введите значение ", i, "-го элемента массива"
  ввод m[i]
кц
```

Для двумерных массивов соответствующие фрагменты:

```
m[1, 1] := ...
m[1, 2] := ...
...
m[1, s] := ...
```

```
m[2, 1] := ...
m[2, 2] := ...
...
m[2, s] := ...
...
m[n, 1] := ...
m[n, 2] := ...
...
m[n, s] := ...
```

где n — число строк, s — число столбцов (эти же величины будут использоваться и далее).

```
нц для i от 1 до n
  нц для j от 1 до s
    ввод m[i, j]
  кц
кц

нц для i от 1 до n
  нц для j от 1 до s
    вывод нс, "Введите значение элемента в ", i, "-й строке "
    вывод "в ", j, "-м столбце"
    ввод m[i, j]
  кц
кц
```

П3.2. Заполнение массива одинаковыми значениями

В этом случае фрагмент программы для заполнения имеет вид:

а) одномерного массива:

```
нц для i от 1 до n
  m[i] := ...
кц
```

б) двумерного массива:

```
нц для i от 1 до n
  нц для j от 1 до s
    m[i, j] := ...
  кц
кц
```

Например, при заполнении одномерного массива нулями:

```
нц для i от 1 до n
  m[i] := 0
кц
```

П3.3. Заполнение массива последовательностью чисел, закон построения которой известен

Когда зависимость значения элемента массива от его индекса $m[i] = f(i)$ указана явно, задача является простой:

```
m[1] := 2
нц для i от 2 до n
  m[i] := m[i-1]+3
кц
```

где вместо многоточия должно быть записано выражение, соответствующее заданной зависимости. Например, для случая заполнения массива по закону $m[i] = i^2$:

```
нц для i от 1 до n
  m[i] := i * i
кц
```

Для двумерного массива:

```
нц для i от 1 до n
  нц для j от 1 до s
    m[i, j] := ...
  кц
кц
```

где вместо многоточия должно быть записано выражение, соответствующее заданной зависимости. Эта может быть зависимость от номера строки и/или номера столбца. Например, для случая заполнения массива по закону $m[i, j] = i + j$:

```
нц для i от 1 до n
  нц для j от 1 до s
    m[i, j] := i + j
  кц
кц
```

Другим примером последовательностей, закон построения которых известен, являются значения членов прогрессий (арифметической или геометрической). В таких случаях первый элемент массива равен первому члену соответствующей прогрессии, а остальные рассчитываются с использованием формул:

- i-го члена арифметической прогрессии: $a_i = a_{i-1} + d$,

  где d — разность прогрессии;

- i-го члена арифметической прогрессии: $a_i = a_{i-1} \cdot z$,

  где z — знаменатель прогрессии.

ПРИМЕР 1. Заполнить одномерный массив из `n` элементов членами арифметической прогрессии, первый член которой равен 2, а разность — 3.

Решение. В соответствии с определением арифметической прогрессии можно так оформить фрагмент программы, в котором проводится заполнение массива:

```
m[1] := 2
нц для i от 1 до n
  m[i] := m[i − 1] + 3
кц
```

ПРИМЕР 2. Заполнить одномерный массив из n элементов членами геометрической прогрессии, первый член которой равен 1, а знаменатель — 1,5.

Решение. Учитывая закон построения геометрической прогрессии, можем записать:

```
m[1] := 1
нц для i от 2 до n
  m[i] := m[i − 1] * 1.5
кц
```

Прогрессии являются частным случаем так называемых "рекуррентных" последовательностей — последовательностей чисел, каждый элемент которых зависит от одного или нескольких предшествующих элементов[1].

П3.4. Заполнение массива случайными значениями

Для получения случайных чисел во всех современных языках программирования имеется стандартная функция. В школьном алгоритмическом языке ее имя — `rnd`, в языке Бейсик — `RND` (верхний регистр букв, естественно, не обязателен), в языке Паскаль — `random`. Формулы для расчета случайного числа x различного типа в программах на этих языках программирования приведены в табл. П3.1–П3.3.

Таблица П3.1. *Школьный алгоритмический язык*[2]

Тип величины x	Диапазон возможных значений	Формула
Вещественный	$0 \le x < 1$	`x = rnd(1)`
	$0 \le x < A$	`x = rnd(A)`
	$A \le x < B$	`x = A + rnd(B − A)`
Целый	$0 \le x \le A$	`x = int(rnd(A + 1))`
	$A \le x \le B$	`x = A + int(rnd(B − A + 1))`

[1] Формулы, выражающие очередной член последовательности через один или несколько предыдущих членов, называют *рекуррентными соотношениями*.

[2] В новой версии системы КуМир [13].

Таблица П3.2. *Язык Паскаль*

Тип величины x	Диапазон возможных значений	Формула
Вещественный	$0 \le x < 1$	`x = random`
	$0 \le x < A$	`x = random * A`
	$A \le x < B$	`x = A + random * (B − A)`
Целый	$0 \le x \le A$	`x = random(A + 1)`
	$A \le x \le B$	`x = A + random(B − A + 1)`

В случае заполнения массива случайными значениями следует использовать оператор цикла:

а) для одномерного массива:

```
нц для i от 1 до n
  m[i] := ...
кц
```

б) для двумерного массива:

```
нц для i от 1 до n
  нц для j от 1 до s
    m[i, j] := ...
  кц
кц
```

где вместо многоточия должно быть записано выражение, соответствующее типу и диапазону случайных значений.

> ***Внимание!***
>
> В программах на языках Бейсик и Паскаль, в случае, когда в них указанные функции используются несколько раз (например, в операторе цикла), при каждом новом запуске программы будут генерироваться одни и те же случайные числа. Чтобы исключить это, необходимо записать (до первого использования функций):
>
> - в программах на языке Бейсик оператор RANDOMIZE с параметром TIMER (в виде RANDOMIZE TIMER);
> - в программах на языке Паскаль — оператор `randomize` (без параметров).

ПРИЛОЖЕНИЕ 4

Простейшие методы сортировки массивов

В данном приложении описаны два простых метода сортировки массивов. В обоих случаях рассматривается сортировка одномерного массива *a*, состоящего из *n* элементов. Сортировка проводится в порядке возрастания значений.

Заметим, что запомнить всю программу сортировки сложно, поэтому следует знать особенности описанных методов, на основе которых можно разработать соответствующую программу.

Сортировка обменом

Сортировка обменом — метод, при котором все соседние элементы массива попарно сравниваются друг с другом и меняются местами в том случае, если предшествующий элемент больше последующего (при сортировке в порядке возрастания). Этот процесс повторяется $(n - 1)$ раз.

Например, требуется провести сортировку массива:

30, 17, 73, 47, 22, 11, 65, 54.

Методика сортировки отражена на рис. П4.1 (представлены первые два прохода обработки).

Если индексы "левых" элементов в паре сравниваемых обозначить лев, то фрагмент программы на школьном алгоритмическом языке, реализующий описанные действия на одном проходе, выглядит так:

```
нц для лев от 1 до n — 1
  |Если "левый" элемент в паре сравниваемых
  |больше "правого"
  если a[лев] > a[лев + 1]
    то |Проводим их обмен
      всп := a[лев]
      a[лев] := a[лев + 1]
      a[лев + 1] := всп
  все
кц
```

Сравниваемые элементы		Обмен
Первый проход по массиву:		
1)	30 и 17	Проводится
2)	30 и 73	Нет
3)	73 и 47	Проводится
4)	73 и 22	Проводится
5)	73 и 11	Проводится
6)	73 и 65	Проводится
7)	73 и 54	Проводится
Полученный массив: 17, 30, 47, 22, 11, 65, 54, 73		
Второй проход по массиву:		
1)	17 и 30	Нет
2)	30 и 47	Нет
3)	47 и 22	Проводится
4)	47 и 11	Проводится
5)	47 и 65	Нет
6)	65 и 54	Проводится
7)	65 и 73	Нет
Полученный массив: 17, 30, 22, 11, 47, 54, 65, 73		

Рис. П4.1

а вся процедура сортировки оформляется следующим образом:

```
|Проходим по массиву n — 1 раз,
нц для номер_прохода от 1 до n - 1
  |сравнивая пары элементов
  нц для лев от 1 до n — 1
    если a[лев] > a[лев + 1]
      то |Проводим обмен
       всп := a[лев]
       a[лев] := a[лев + 1]
       a[лев + 1] := всп
    все
  кц
кц
```

Язык Паскаль

```
{Проходим по массиву n — 1 раз,}
for nomer_proxoda := 1 to n - 1 do
  {сравнивая пары элементов}
  for lev := 1 to n - 1 do
```

```
if a[lev] > a[lev + 1] then {Проводим обмен}
  begin
    vsp := a[lev];
    a[lev] := a[lev + 1];
    a[lev + 1] := vsp
 end;
```

Лирическое отступление. Если последовательность сортируемых чисел расположить вертикально (первый элемент массива — внизу) и проследить за перемещением элементов (рис. П4.2), то можно увидеть, что большие элементы, подобно пузырькам воздуха в воде[1], "всплывают" на соответствующую им позицию. Поэтому сортировку таким способом называют еще сортировкой методом "пузырька" или "пузырьковой" сортировкой.

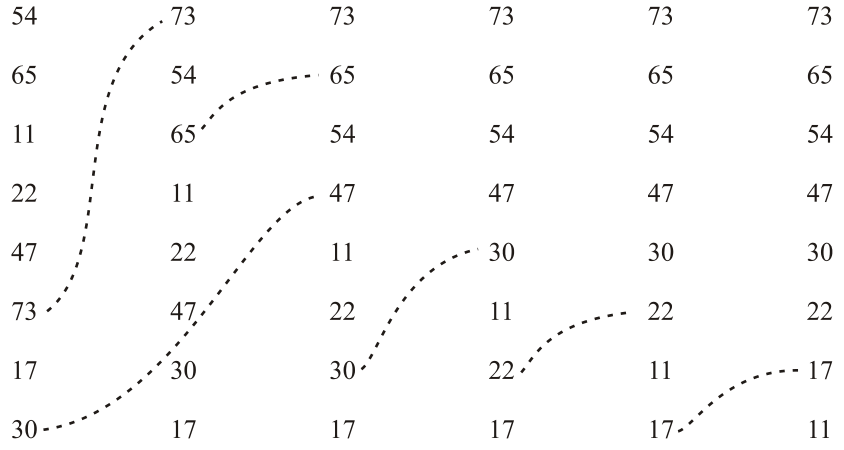


Рис. П4.2

Можно усовершенствовать программу, учитывая следующее обстоятельство. В ходе первого прохода максимальный элемент постепенно смещается вправо и, в конце концов, занимает свое (которое он должен занимать в упорядоченном массиве — крайнее правое) место в массиве (см. рис. П4.1). После этого его можно исключить из дальнейшей обработки. Затем процесс повторяется, и свое место занимает второй по величине элемент, который также исключается из дальнейшего рассмотрения. Так продолжается до тех пор, пока весь массив не будет упорядочен.

Выпишем пары индексов элементов, сравниваемых на каждом проходе с учетом сказанного только что, в виде табл. П4.1.

Если индекс "левого" элемента в последней паре сравниваемых на каждом проходе чисел обозначить `посл_лев` (соответствующие значения в табл. П4.1 выделены полужирным начертанием), то можно так оформить фрагмент программы, осуществляющей сортировку:

[1] В воде всплывают "легкие" пузырьки.

```
нц для посл_лев от n - 1 до 1 шаг -1
  нц для лев от 1 до посл_лев
    если a[лев] > a[лев + 1]
      то |Проводим обмен
        ...
    все
  кц
кц
```

Таблица П4.1

Номер прохода по массиву	Индексы				
1-й	1 – 2	2 – 3	...	$(n-2)-(n-1)$	$(\boldsymbol{n-1})-n$
2-й	1 – 2	2 – 3		$(\boldsymbol{n-2})-(n-1)$	
...					
$(n-1)$-й	**1** – 2				

Язык Паскаль

```
for posl_lev := n - 1 downto 1 do
  for lev :=1 to posl_lev do
    if a[lev] > a[lev + 1] then {Проводим обмен}
      begin
        vsp := a[lev];
        a[lev] := a[lev + 1];
        a[lev + 1] := vsp
     end;
```

Можно также в качестве параметра "наружного" оператора цикла использовать номер прохода, а конечное значение параметра "внутреннего" оператора цикла связать с номером прохода согласно табл. П4.1.

```
нц для номер_прохода от 1 до n - 1
  нц для лев от 1 до n — номер_прохода
    если a[лев] > a[лев + 1]
      то |Проводим обмен
        ...
    все
  кц
кц
```

По нашему мнению, такой вариант более "запоминаем".

Язык Паскаль

```
for nomer_proxoda := 1 to n - 1 do
  for lev :=1 to n - nomer_proxoda do
    if a[lev] > a[lev + 1] then
      …
```

Можно также прекратить проходы по массиву, как только он станет отсортированным (в общем случае это может произойти менее чем за $n - 1$ проходов). Признак, по которому целесообразно установить факт отсортированности массива, — на каком-то проходе обменов элементов местами не было. Однако при этом программа усложняется.

Сортировка выбором

Сортировка выбором состоит в том, что сначала в неупорядоченном массиве выбирается минимальный элемент[1]. Этот элемент исключается из дальнейшей обработки, а оставшаяся последовательность элементов принимается за исходную, и процесс повторяется до тех пор, пока все элементы не будут выбраны. Очевидно, что выбранные элементы образуют упорядоченную последовательность.

Выбранный в исходном массиве минимальный элемент может быть размещен на предназначенном ему месте упорядоченной последовательности разными способами. Один из них заключается в использовании дополнительного массива — после первого просмотра найденный минимальный элемент размещается на первом месте в этом массиве (при сортировке в порядке возрастания) (рис. П4.3).

Рис. П4.3

Но если после этого вновь найти минимальный элемент, то им окажется тот же самый (найденный ранее) элемент. Чтобы исключить это, надо в исходном массиве на месте выбранного записать число, превосходящее любой элемент сортируемого

[1] Разумеется, можно выбрать и максимальный. В этом случае алгоритм претерпевает минимальные изменения.

массива. Тогда, вновь найдя при втором проходе минимальный элемент, можно разместить его на втором месте в дополнительном массиве и т. д. Итак, общая схема алгоритма сортировки рассмотренным методом имеет вид:

```
нц для i от 1 до n
  1. Определить минимальный элемент массива.
  2. Записать его на i-е место в дополнительном массиве.
  3. В исходном массиве на место минимального элемента
     записать "большое" число.
кц
```

Что нужно знать, чтобы выполнить этап 3? Во-первых, число, которое превосходит любой элемент сортируемого массива. Будем считать, что такое число известно. Во-вторых, индекс минимального элемента. Значит, на этапе 1 следует также найти этот индекс *(см. задачу 1.9.4)*.

В приведенном далее фрагменте программы сортировки массива обсуждаемым методом используем величины:

- мин — минимальный элемент массива;
- индмин — его индекс;
- макс — число, которым заменяются минимальные элементы исходного массива.

```
нц для i от 1 до n
  |1. Находим минимальный элемент массива и его индекс
  мин := a[1]; индмин := 1
  нц для j от 2 до n
    если a[j] < a[индмин]
      то
       мин := a[j]
       индмин := j
    все
  кц
  |2. Записываем минимальный элемент на i-е место в массиве b
  b[i] := мин
  |3. Заменяем минимальный элемент исходного массива "большим" числом
  a[индмин] := макс
кц
```

> ***Примечание***
>
> Величину мин можно не использовать — значение минимального элемента массива можно получить, зная величину индмин *(см. разд. 1.9.3 и 1.9.4)*.

Язык Паскаль

```
for i := 1 to n do
  begin
    {1. Находим минимальный элемент массива и его индекс}
    min := a[1]; indmin := 1;
```

```
    for j := 2 to n do
      if a[j] < min then
        begin
          min := a[j];
          indmin := j
        end;
      {2. Записываем минимальный элемент на i-е место в массиве b}
       b[i] := min;
      {3. Заменяем минимальный элемент исходного массива "большим" числом}
       a[indmin] := max
  end;
```

Рассмотренный вариант сортировки массива методом выбора обладает двумя недостатками:

- для его реализации требуется дополнительный массив;
- при нахождении минимального элемента и его индекса на каждом проходе приходится рассматривать все элементы массива.

Указанные недостатки устраняются, если все изменения проводить в исходном массиве. Отсортировать его можно следующим образом:

1. Найти минимальный элемент среди всех элементов массива и поменять его местами с первым элементом.
2. Найти минимальный элемент среди второго, третьего ... последнего элементов массива и поменять его местами со вторым элементом.

...

На последнем этапе находится минимальный элемент среди двух последних, найденный элемент меняется местами с предпоследним элементом исходного массива. Поскольку после каждого просмотра упорядоченные элементы исключаются из дальней обработки, размер каждого последующего обрабатываемого участка массива на 1 меньше размера предыдущего.

Общая схема программы, основанной на сделанных рассуждениях, такая:

```
нц для i от 1 до n – 1
  1. Ищем минимальный элемент мин и его индекс индмин
     среди элементов с индексами i, i + 1, i + 2, ..., n
  2. Меняем местами минимальный и i-й элементы[1]
кц
```

Итак, мы пришли к задаче нахождения минимального элемента и его индекса среди элементов с индексами *i*, *i* + 1, *i* + 2 ... *n*. Эта задача решается аналогично такой задаче применительно ко всему массиву:

```
|Начальные значения
мин := a[i]
индмин := i
```

[1] Этап 2 можно выполнять только в случае, когда найденный минимальный элемент меньше *i*-го.

```
|Рассматриваем остальные элементы
нц для j от i + 1 до n
  если a[j] < a[индмин]
    то
     мин := a[j]
     индмин := j
  все
кц
```

С учетом сказанного фрагмент, реализующий сортировку, имеет вид:

```
нц для i от 1 до n - 1
  |Этап 1
  мин := a[i]
  индмин := i
  нц для j от i + 1 до n
    если a[j] < a[индмин]
      то
       мин := a[j]
       индмин := j
    все
  кц
  |Этап 2
  |Меняем местами минимальный и i-й элементы
  a[индмин] := a[i]
  a[i] := мин
кц
```

Примечание

Здесь также величину мин можно не использовать.

Язык Паскаль

```
for i := 1 to n - 1 do
  begin
    {Этап 1}
    min := a[i];
    indmin := i;
    for j := i + 1 to n do
      if a[j] < min then
        begin
          min := a[j];
          indmin := j
        end;
    {Этап 2. Меняем местами минимальный и i-й элементы}
    a[indmin] := a[i];
    a[i] := min
  end;
```

В заключение заметим, что сортировку методом обмена можно применить к массивам как числового, так и символьного и строкового типов. В последнем случае значения сортируются в так называемом лексикографическом порядке, например:

```
бит бот ботаник ботаника код кодировка карта линейка …
```

При сортировке методом выбора использование данных символьного и строкового типов нецелесообразно (хотя и возможно).

Список литературы

1. Богомолова О. Б., Усенков Д. Ю. Задачи на пересечение областей (С1): "на стыке алгебры и логики". — Информатика, № 5, 2011.
2. Демонстрационный вариант контрольных измерительных материалов Единого государственного экзамена по информатике и ИКТ 2007 года. **http://www.fipi.ru/binaries/395/inf dem.doc**.
3. Демонстрационный вариант контрольных измерительных материалов Единого государственного экзамена по информатике и ИКТ 2008 года. **http://www.fipi.ru/binaries/518/inform.rar**.
4. Демонстрационный вариант контрольных измерительных материалов Единого государственного экзамена по информатике и ИКТ 2009 года. **http://www.fipi.ru/binaries/731/infZIP - WinRAR.zip**.
5. Демонстрационный вариант контрольных измерительных материалов Единого государственного экзамена по информатике и ИКТ 2010 года. **http://www.fipi.ru/binaries/895/inf.zip**.
6. Демонстрационный вариант контрольных измерительных материалов Единого государственного экзамена 2011 года по информатике и ИКТ. **http://www.fipi.ru/binaries/1079/inf_10_11_10.zip**.
7. Демонстрационный вариант контрольных измерительных материалов Единого государственного экзамена 2012 года по информатике и ИКТ (проект). **http://www.fipi.ru/binaries/1232/infEGE2012.zip**.
8. Златопольский Д. М. Программирование: типовые задачи, алгоритмы, методы. — М.: Бином. Лаборатория знаний, 2007.
9. Златопольский Д. М. Сборник задач по программированию — 3-е изд. — СПб.: БХВ-Петербург, 2011.
10. Кодификатор элементов содержания и требований к уровню подготовки выпускников общеобразовательных учреждений для проведения в 2012 году Единого государственного экзамена по информатике и ИКТ (проект). / Подготовлен Федеральным государственным научным учреждением "Федеральный институт педагогических измерений", 2011. **http://www.fipi.ru/binaries/1232/infEGE2012.zip**.

11. Кушниренко А. Г., Лебедев Г. В., Зайдельман Я. Н. Информатика. 7–9 классы. — М.: Изд-во "Дрофа", 2002.
12. Самое полное издание типовых вариантов реальных заданий ЕГЭ: 2010: Информатика / авт. и сост. П. А. Якушкин, Д. М. Ушаков. — М.: Астрель, 2010. (Федеральный институт педагогических измерений).
13. Система программирования КуМир. **http://www.niisi.ru/kumir/**.
14. Спецификация контрольных измерительных материалов Единого государственного экзамена 2012 года по информатике и ИКТ (проект). **http://www.fipi.ru/binaries/1232/infEGE2012.zip**.
15. Усенков Д. Ю. Поиск наибольшего и наименьшего значений среди двух, трех и четырех чисел. — Информатика, № 4, 2011.
16. Якушкин П. А., Лещинер В. Р., Кириенко Д. П. ЕГЭ 2010. Информатика. Типовые тестовые задания. — М.: Изд-во "Экзамен", 2010.