Предисловие[1]

Научиться программировать можно только программируя. Другого способа нет. Но прежде чем самому начать проектировать алгоритмы и записывать их на языках программирования, полезно разобрать большое число разнообразных алгоритмов и реализующих их программ.

Цели настоящего практикума:

- Скорейшее привлечение читателя к самостоятельному и осмысленному составлению законченных программ на популярном языке программирования Pascal;
- Привитие основных навыков алгоритмической и программистской грамотности:
    - ясного и понятного стиля;
    - надёжности решений;
    - экономии вычислений;
    - организации переборов и т.д.

Тематически практикум разбит на несколько разделов, охватывающих обработку числовой, текстовой и графической информации.

Примеры и задачи для самостоятельного решения в разделах подобраны по общности алгоритмических конструкций, употребляемых для их реализации:

- задачи без циклов;
- задачи на циклы с известным числом повторений;
- задачи на циклы с неизвестным числом повторений;
- задачи, реализуемые комбинациями этих двух видов циклов;
- задачи обработки текстовой информации;
- задачи обработки графической и звуковой информации.

Для каждой задачи практикума приводятся:

- система тестов;
- параллельная реализация алгоритма на **школьном алгоритмическом языке**, **языке блок-схем** и на языке **Turbo Pascal**;
- таблицы исполнения алгоритма на каждом из тестов.

Для многих задач приводятся **результаты работы программ**, выведенные на экран дисплея. Такое же изображение получит читатель, выполняя программу на своем компьютере.

Важное значение, придаваемое тестированию алгоритмов, объясняется следующим:

- на этом этапе детально изучается и уточняется условие задачи;
- происходит осмысление того, что является исходными данными и результатами;
- фиксируются все ситуации, которые могут возникнуть при решении задачи;

---

[1] Файлы созданы на основе интернет-версии издания: ***Шауцукова Л.З. Информатика 10 - 11. — М.: Просвещение, 2000 г. (http://www.tomsk.ru/Books/informatica/practice/index.html)***

- уточняются типы данных;
- даются имена переменным;
- продумываются формы представления и выдачи исходных данных и результатов.

Приводимые способы и программы решения задач по возможности являются рациональными, но не претендуют на то, чтобы быть наилучшими.

Так, в программах из-за соображений экономии объёма не предусмотрена защита от недопустимых данных, хотя это – обязательный элемент любой программы. Читатель может сам восполнить эти недочеты, воспользовавшись рекомендациями восьмой главы первой книги ("Теория") настоящего учебника, и в ряде случаев предложить более совершенное решение задачи.

Примечание:

Тексты программ, приведенные в электронном файле методического пособия, можно скопировать в файл Паскаля и выполнить. Так как редактор Турбо Паскаля не поддерживает работу с буфером обмена Windows, для создания файла следует использовать текстовый редактор файлового менеджера FAR.

*Для этого в документе Word выделить текст программы и скопировать в буфер обмена (команда меню **Правка/Копировать**). Перейти в окно FAR, открыть каталог, где хранятся текстовые файлы Паскаль-программ (C:\TP\BIN\) и создать текстовый файл (клавиши Ctrl+F4), присвоив ему имя с расширением .PAS (например, prg.pas). В открывшемся окне редактора FAR вставить из буфера обмена текст программы (клавиши Ctrl+V). Сохранить файл на диске (клавиша F2) и выйти из редактора FAR (клавиша Esc).*

Запустить Турбо Паскаль и вызвать с диска программу.

Настройки

Содержание учебника можно читать практически под любой аппаратно-программной платформой (ввиду кроссплатформной переносимости HTML), но демонстрационная часть, связанная с запуском таких внешних по отношению к браузеру программ как Turbo Pascal или Quick Basic, будет работать лишь под управлением операционных систем MS Windows 95/98, NT 4.0. Что в свою очередь требует дополнительной настройки.

Поскольку как Turbo Pascal, так и Quick Basic являются приложениями DOS, а не Windows, то требуется сконфигурировать их для корректной работы в Windows.

Настройка Turbo Pascal

Предположим, интегрированная среда Turbo Pascal находится в каталоге **C:\TP** (тогда файл turbo.exe должен находиться в каталоге **C:\TP\BIN**). Требуется выполнить следующую последовательность действий:

- щелкнуть *правой* кнопкой мыши на кнопке "Пуск" панели задач;
- в появившемся контекстном меню выбрать пункт "Проводник";
- зайти в каталог **C:\TP\BIN**;
- на файле turbo.exe щелкнуть *правой* кнопкой мыши;
- в появившемся контекстном меню выбрать "Свойства";
- во вновь появившемся окне выбрать закладку "Программа";
- щелкнуть на кнопке "Дополнительные параметры";
- в появившемся окне выставить все установки как показано на рис. 1;

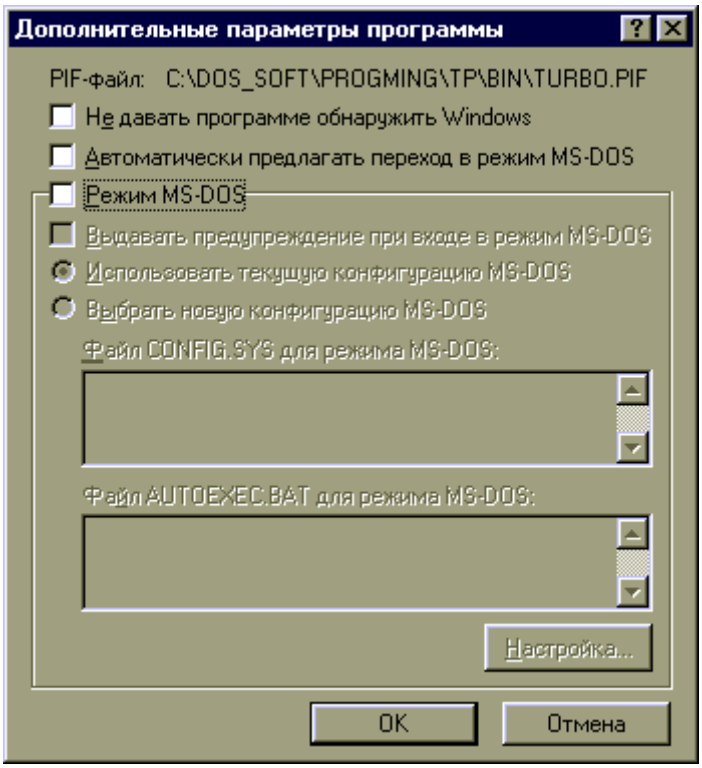


Рис. 1

- закрыть окно щелчком на кнопке "ОК";
- активизируется предыдущее окно, в котором установить пункт "Закрывать окно по завершении сеанса работы";

- закрыть окно щелчком на кнопке "ОК";
- закрыть Проводник.

Настройка Quick Basic

Предположим, интерпретатор Quick Basic находится в каталоге **C:\QB** (тогда файл basic.exe должен находиться в каталоге **C:\QB**). Требуется выполнить для файла basic.exe аналогичную описанной в предыдущем пункте для turbo.exe последовательность действий.

Настройки расширений файлов

Демонстрационные примеры программ хранятся в файлах с расширением .tpp (для Turbo Pascal) и .qbp (для Quick Basic). Для того, чтобы при их приеме браузер не показывал их текст в своем окне, а запускал внешние программы (компилятора Паскаля и интерпретатора Бейсика) необходимо зарегистрировать указанные типы файлов среди известных Windows типов файлов.

Настройка .tpp

Рекомендуется: установить среду Turbo Pascal 7.0 в каталог **C:\TP**. В этом случае запишите на свой диск и запустите на выполнение конфигурационный файл tpp.reg.

Внимание: воспользуйтесь этим файлом только в том случае, если у вас корректно установлен Turbo Pascal 7.0 **в каталог C:\TP**. В таком случае остаток данной секции можно пропустить. Если по какой-либо причине вы не можете воспользоваться **tpp.reg**, попробуйте провести настройки вручную как описано ниже.

Предположим, интегрированная среда Turbo Pascal находится в каталоге **C:\TP** (тогда файл turbo.exe должен находиться в каталоге **C:\TP\BIN**). Требуется выполнить следующую последовательность действий:

- щелкнуть *правой* кнопкой мыши на кнопке "Пуск" панели задач;
- в появившемся контекстном меню выбрать пункт "Проводник";
- в верхнем меню "Вид" выбрать пункт "Параметры...";
- в появившемся окне выбрать закладку "Типы файлов";
- нажать кнопку "Новый тип";
- в поле "Описание:" набрать **Turbo Pascal Program**;
- в поле "Тип (MIME):" набрать **application/x-turbo-pascal**;
- в поле "Стандартное расширение для типа:" набрать **.tpp**;
- щелкнуть на кнопке "Создать...";
- в появившемся окне в поле "Действие:" набрать **open** ;
- в поле "Приложение, исполняющее действие:" набрать **C:\TP\BIN\TURBO.EXE %1** ;
- закрыть окно щелчком на кнопке "ОК";
- закрыть появившееся предыдущее окно щелчком на кнопке "Закрыть";
- вновь закрыть появившееся предыдущее окно щелчком на кнопке "Закрыть";
- закрыть Проводник.

Настройка .qbp

Рекомендуется: установить среду Quick Basic в каталог **C:\QB**. В этом случае запишите на свой диск и запустите на выполнение конфигурационный файл qbp.reg.

Внимание: воспользуйтесь этим файлом только в том случае, если у вас корректно установлен Quick Basic **в каталог C:\QB**. В таком случае остаток данной секции можно пропустить. Если по какой-либо причине вы не можете воспользоваться **qbp.reg**, попробуйте провести настройки вручную как описано ниже.

Предположим, интерпретатор Quick Basic находится в каталоге **C:\QB** (тогда файл basic.exe должен находиться в каталоге **C:\QB**). Требуется выполнить следующую последовательность действий:

- щелкнуть *правой* кнопкой мыши на кнопке "Пуск" панели задач;
- в появившемся контекстном меню выбрать пункт "Проводник";
- в верхнем меню "Вид" выбрать пункт "Параметры...";
- в появившемся окне выбрать закладку "Типы файлов";
- нажать кнопку "Новый тип";
- в поле "Описание:" набрать **Quick Basic Program**;
- в поле "Тип (MIME):" набрать **application/x-quick-basic**;
- в поле "Стандартное расширение для типа:" набрать **.qbp**;
- щелкнуть на кнопке "Создать...";
- в появившемся окне в поле "Действие:" набрать **open** ;
- в поле "Приложение, исполняющее действие:" набрать **C:\QB\BASIC.EXE %1** ;
- закрыть окно щелчком на кнопке "ОК";
- закрыть появившееся предыдущее окно щелчком на кнопке "Закрыть";
- вновь закрыть появившееся предыдущее окно щелчком на кнопке "Закрыть";
- закрыть Проводник.

Настройка браузера

При принятии файла нестандартного типа браузер выдает запрос на сохранение файла, с возможностью открытия этого файла, если он имеет зарегистрированное расширение. Поэтому, каждый раз при попытке запуска демонстрации, браузер, получая .tpp или .qbp файл, будет выдавать подобный запрос. Во избежание этого, при первом получении запроса (т.е. при первом нажатии кнопки "Turbo Pascal" демонстрации) следует указать браузеру открыть файл и больше не задавать подобный вопрос, т.е. выставить установки как показано на рис. 2 и нажать "OK" (рассматривается настройка браузера Netscape Navigator 4.0).

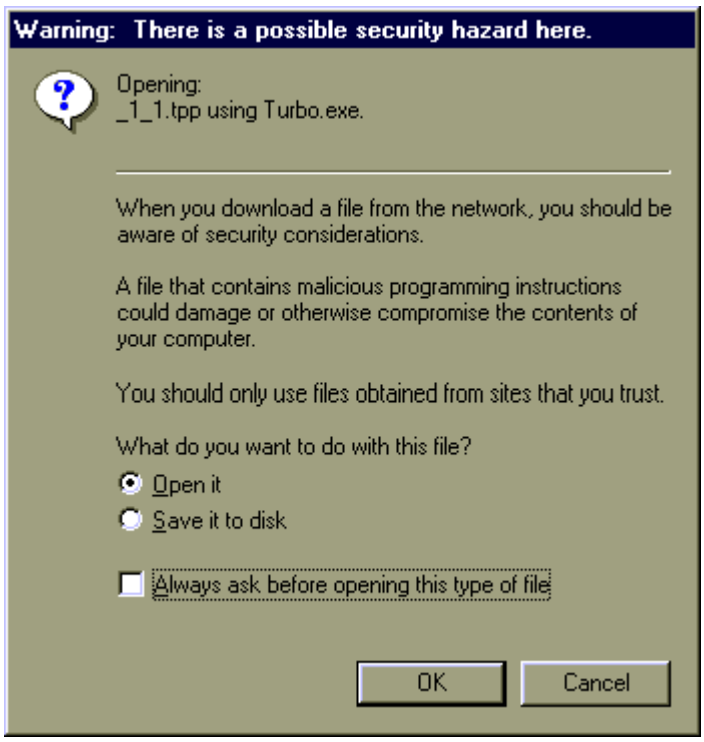


Рис. 2

При первом нажатии кнопки "Quick Basic" демонстрации и появлении подобного окна запроса, следует проделать аналогичные действия.

После этого браузер запустит Turbo Pascal или Quick Basic с передачей полученного файла в качестве параметра. При дальнейших нажатиях кнопок демонстрации более никаких запросов выдаваться не будет.

Внимание: все настройти вступят в силу только после перезагрузки компьютера.

Пример 1.1. ***Простейший алгоритм, запрашивающий имя и затем приветствующий его обладателя.***

Тест

Данные	Результат
Имя = "Тимур"	"Привет, Тимур!"

Школьный АЯ

```
алг Знакомство (арг лит Имя, рез лит t)
нач
  вывод "Как тебя зовут ?"
  ввод Имя
  t := "Привет, " + Имя + "!"     | "+" - операция сцепки
  вывод t
кон
```

Turbo Pascal

```
Program Hello;
Var Name: String;  {Описание переменной Name строкового типа}
BEGIN
  Write('Как тебя зовут ? '); {Вывод на экран текста вопроса}
  ReadLn(Name);               {Ввод с клавиатуры имени}
  WriteLn('Привет, ', Name, '!'); {Вывод на экран приветствия}
  ReadLn
END.
```

Здесь последний оператор ReadLn позволяет видеть на экране результаты работы программы, пока не будет нажата клавиша <Enter>.

Результаты работы Pascal-программы

```
Как тебя зовут ? Тимур   <Enter>
Привет, Тимур !
```

Пример 1.2. ***Определить объём и площадь боковой поверхности цилиндра с заданными радиусом основания R и высотой H.***

Тест

Данные		Результат	
R = 1	H = 1	V = 3.14	S = 6.28

Школьный АЯ

```
алг Цилиндр (арг вещ R, H, рез вещ V, S)
нач вещ Pi
   Pi := 3.14
   V := Pi * R**2 * H
   S := 2 * Pi * R * H
кон
```

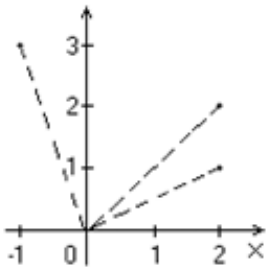
Turbo Pascal

```
Program Cylinder;
  Uses Crt; {Подключение библиотеки Crt}
  Var
    R, {радиус основания цилиндра}
    H, {высота цилиндра }
    V, {объем цилиндра }
    S: Real; {площадь боковой поверхности цилиндра}
BEGIN
  ClrScr; {Вызов из библиотеки Crt процедуры очистки экрана}
  Write('Введите высоту цилиндра : ');  ReadLn(H);
  Write('Введите радиус основания : '); ReadLn(R);
  V := Pi * R * R * H;
  S := 2 * Pi * R * H;  WriteLn;
  WriteLn('Объем цилиндра = ', V : 5 : 2); {Здесь 5 - общее
      количество позиций, занимаемых переменной V при выводе,
      а 2 - количество позиций в дробной части значения V}
  WriteLn('Площадь боковой поверхности = ', S : 5 : 2);
  ReadLn
END.
```

Пример 1.3. ***Даны три точки на плоскости. Определить, какая из них ближе к началу координат.***

Система тестов

Номер теста	Данные						Результат
	xA	yA	xB	yB	xC	yC	Otvet
1	2	1	2	2	-1	3	"Это точка A"
2	2	2	2	1	-1	3	"Это точка B"
3	2	2	-1	3	2	1	"Это точка C"/TR>



Школьный АЯ

```
алг Точки(арг вещ xA,yA,xB,yB,xC,yC, рез лит Otvet)
нач вещ DistA,DistB,DistC
  ввод xA,yA,xB,yB,xC,yC
  DistA := sqrt(xA**2 + yA**2)
  DistB := sqrt(xB**2 + yB**2)
  DistC := sqrt(xC**2 + yC**2)
  если (DistA < DistB) и (DistA < DistC)
    то Otvet := "Это точка A"
    иначе если DistB < DistC
            то Otvet := "Это точка B"
            иначе Otvet := "Это точка C"
          все
  все
  вывод Otvet
кон
```

Turbo Pascal

```
Program Points;
  Uses Crt;
  Var xA, yA, xB, yB, xC, yC, DistA, DistB, DistC : Real;
BEGIN ClrScr;
  WriteLn('Введите координаты точки A:');
  Write('x = '); ReadLn(xA); Write('y = '); ReadLn(yA);
  WriteLn('Введите координаты точки B:');
  Write('x = '); ReadLn(xB); Write('y = '); ReadLn(yB);
  WriteLn('Введите координаты точки C:');
  Write('x = '); ReadLn(xC); Write('y = '); ReadLn(yC);
  DistA := sqrt(sqr(xA) + sqr(yA));
  DistB := sqrt(sqr(xB) + sqr(yB));
  DistC := sqrt(sqr(xC) + sqr(yC));
  WriteLn; Write('Ответ : ');
  If (DistA < DistB) and (DistA < DistC)
     then WriteLn( 'Это точка A.')
     else If (DistB < DistC)
             then WriteLn('Это точка B.')
             else WriteLn('Это точка C.');
  ReadLn
END.
```

Пример 1.4. ***Найти произведение цифр заданного целого четырехзначного числа.***

Система тестов

Номер теста	Проверяемый случай	Число	Результат
1	Число положительное	2314	P = 24
2	Число отрицательное	-1245	P = 40

Школьный АЯ

```
алг Произведение цифр (арг цел Num, рез цел P)
нач цел i, j, k, l
  Num := abs(Num)              | abs - абсолютная величина
  i := div(Num, 1000)          | i - первая цифра
                               | div - частное от деления с
остатком
  j := mod(div(Num, 100), 10)  | j - вторая цифра
                               | mod - остаток от деления с
остатком
  k := mod(div(Num, 10), 10)   | k - третья цифра
  l := mod(Num, 10)            | l - четвертая цифра
  P := i * j * k * l;
кон
```

Turbo Pascal

```
Program DigitsProduct;
Uses Crt;
Var Number,       {заданное число}
    i, j, k, l,   {цифры числа}
    P : Integer; {произведение цифр}
BEGIN ClrScr;
  Write( 'Введите четырехзначное число : ' ); ReadLn(Number);
  Number:=Abs(Number);
  Write( 'Цифры числа ' , Number , ' : ' );
  i := Number div 1000; Write(i:3);        {первая цифра}
  j := Number div 100 mod 10; Write(j:3); {вторая цифра}
  k := Number div 10 mod 10; Write(k:3);  {третья цифра}
  l := Number mod 10; WriteLn(l:3);        {четвертая цифра}
  P := i * j * k * l ;
  WriteLn( 'О т в е т : произведение цифр равно ' , P );
  ReadLn
END.
```

Пример 1.5. ***Решить квадратное уравнение*** $ax^2 + bx + c = 0$.

Система тестов

Номер теста	Проверяемый случай	Коэффициенты			Результаты
		a	b	c	
1	d >0	1	1	-2	x1 = 1, x2 = - 2
2	d=0	1	2	1	Корни равны: x1 = - 1, x2 = - 1
3	d < 0	2	1	2	Действительных корней нет
4	a=0, b=0, c=0	0	0	0	Все коэффициенты равны нулю. x

					— любое число.
5	**a=0, b=0, c<>0**	**0**	**0**	**2**	**Неправильное уравнение**
6	**a=0, b<>0**	**0**	**2**	**1**	**Линейное уравнение. Один корень: x = - 0,5**
7	**a <> 0, b <> 0, c = 0**	**2**	**1**	**0**	**x1 = 0, x2 = - 0,5**

Школьный АЯ (упрощенный алгоритм)

```
алг Квур (арг вещ a, b, c, рез вещ x1, x2, рез лит t)
   дано a <> 0
нач вещ d
  d := b**2-4*a*c | d - дискриминант квадратного уравнения
  если d<0
    то t := "Действительных корней нет"
    иначе если d=0
            то t := "Корни равны"; x1 := -b/(2*a); x2 := x1
            иначе t := "Два корня"
                  x1 := (-b + sqrt(d)) / (2*a)
                  x2 := (-b - sqrt(d)) / (2*a)
          все
  все
кон
```

Turbo Pascal

```
Program QuadraticEquation;
  Uses Crt; { подключение библиотеки Crt }
  Var a, b, c      : Real;          {a, b, c - коэффициенты
уравнения}
      Discr        : Real;
      x1, x2       : Real;          {x1, x2 - корни }
      Test, NTest  : Integer;       {Ntest - количество тестов }
BEGIN
  ClrScr;
  Write('Введите количество тестов : ');
  ReadLn(NTest);
  For Test := 1 to NTest do {цикл по всем тестам задачи }
   begin
    Write('Тест ', Test, '. Введите коэффициенты a, b, c : ');
    ReadLn(a, b, c);
    If (a=0) and (b=0) and (c=0)
     then begin Write('Все коэффициенты равны нулю.');
                WriteLn('x - любое число ')
          end
     else
      If (a=0) and (b<>0)
       then WriteLn('Линейное уравнение. Один корень: x =', (-
c/b):6:2)
       else
```

```
          If (a=0) and (b=0) and (c<>0)
            then WriteLn('Неправильное уравнение.')
            else
             begin
              Discr := b*b - 4*a*c;
              If Discr > 0
               then begin
                      x1:=(-b + Sqrt(Discr)) / (2*a);
                      x2:=(-b - Sqrt(Discr)) / (2*a);
                      WriteLn('x1=' , x1:6:2 , '; x2=' , x2:6:2)
                    end
               else
                If Discr = 0
                 then begin
                        x1 := -b/(2*a);
                        WriteLn('Корни равны: x1=', x1:6:2, '
x2=', x1:6:2)
                      end
                 else WriteLn('Действительных корней нет.');
            end;
        WriteLn
      end;
    ReadLn
END.
```

Пример 1.6.

Две прямые описываются уравнениями

$$a_1 x + b_1 y + c_1 = 0;$$
$$a_2 x + b_2 y + c_2 = 0.$$

Напечатать координаты точки пересечения этих прямых, либо сообщить, что эти прямые совпадают, не пересекаются или вовсе не существуют.

Система тестов

Номер теста	Проверяемый случай	Коэффициенты прямых						Результаты
		a1	b1	c1	a2	b2	c2	
1	Первая прямая не существует	0	0	1	1	2	2	Это не прямая
2	Вторая прямая не существует	1	2	2	0	0	1	Это не прямая
3	Все коэффициенты одной или обеих прямых равны нулю	0	0	0	1	2	1	Это не прямая (прямые)
4	Коэффициенты попарно равны	1	2	1	1	2	1	Прямые совпадают
5	Коэффициенты попарно пропорциональны	1	2	1	2	4	2	Прямые совпадают

6	Прямые параллельны	2	3	-1	4	6	1	Прямые параллельны
7	Прямые пересекаются	1	2	-4	1	-2	1	x=1.50, y=1.25

Школьный АЯ

```
алг Пересечение (арг вещ a1, b1, c1, a2, b2, c2,
                 рез вещ x, y, рез лит t)
нач
  если (a1 = 0 и b1 = 0) или ( a2 = 0 и b2 = 0)
    то t := "Это не прямая (прямые)"
    иначе если (a1*b2 = a2*b1) и (a1*c2 = a2*c1)
            то t := "Прямые совпадают"
            иначе если a1*b2 = a2*b1
                    то t := "Прямые параллельны"
                    иначе x := (c1*b2-c2*b1)/(b1*a2-b2*a1)
                           y := (c2*a1-c1*a2)/(b1*a2-b2*a1)
                  все
          все
    все
кон
```

Turbo Pascal

```
Program Intersection;
  Uses Crt; {подключение библиотеки Crt }
  Var a1, b1, c1,  {коэффициенты уравнения первой прямой}
      a2, b2, c2,  {коэффициенты уравнения второй прямой}
      x, y : Real; {координаты точки пересечения }
      Test, NTest : Integer;
BEGIN
  ClrScr; {очистка экрана}
  Write('Введите количество тестов : ');
  ReadLn(NTest);
  For Test := 1 to NTest do {цикл по всем тестам задачи}
   begin
     Write('Тест ', Test, '. Введите a1, b1, c1 : ');
     ReadLn( a1, b1, c1);
     Write(' Введите a2, b2, c2 : ');
     ReadLn( a2, b2, c2);
     WriteLn; Write('О т в е т : ');
     If ( (a1=0) and (b1=0) ) or ( (a2=0) and (b2=0) )
      then WriteLn( 'это не прямая (прямые). ' )
      else
        if (a1*b2=a2*b1) and (a1*c2=a2*c1) {условие совпадения}
         then WriteLn( 'прямые совпадают.' )
         else
          if a1*b2 = a2*b1 {условие параллельности}
           then WriteLn('прямые параллельны.')
           else begin x:=(c1*b2-c2*b1)/(b1*a2-b2*a1);
                      y:=(c2*a1-c1*a2)/(b1*a2-b2*a1);
                      WriteLn('координаты точки пересечения :',
                              ' x = ', x : 5 : 2 , ', y = ', y :
5 : 2);
```

```
                end; WriteLn
    end;
   ReadLn
END.
```

Результаты работы Pascal-программы:

```
Введите количество тестов : 7
Тест 1.  Введите a1, b1, c1 : 0  0  1 <Enter>
Введите a2, b2, c2 : 1  2  2 <Enter>
О т в е т : это не прямая (прямые).

Тест 2. Введите a1, b1, c1 : 1  2  2
<Enter>
  Введите a2, b2, c2 : 0  0  1   <Enter>
О т в е т : это не прямая (прямые).

Тест 3. Введите a1, b1, c1 : 0  0  0
<Enter>
  Введите a2, b2, c2 : 1  2  1   <Enter>
О т в е т : это не прямая (прямые).

Тест 4. Введите a1, b1, c1 : 1  2  1
<Enter>
  Введите a2, b2, c2 : 1  2  1   <Enter>
О т в е т : прямые совпадают.

Тест 5. Введите a1, b1, c1 : 1  2  1
<Enter>
  Введите a2, b2, c2 : 2  4  2   <Enter>
О т в е т : прямые совпадают.

Тест 6. Введите a1, b1, c1 : 2  3 -1 <Enter>
  Введите a2, b2, c2 : 4  6  1 <Enter>
О т в е т : прямые параллельны.

Тест 7. Введите a1, b1, c1 : 1  2 -4 <Enter>
  Введите a2, b2, c2 : 1 -2  1 <Enter>
О т в е т : координаты точки пересечения : x =
1.50, y = 1.25
```

Задачи для самостоятельного решения

1.1. Вычислите длину окружности, площадь круга и объём шара одного и того же заданного радиуса.

1.2. Вычислите периметр и площадь прямоугольного треугольника по длинам двух его катетов.

1.3. По координатам трёх вершин некоторого треугольника найдите его площадь и периметр.

1.4. Вычислите дробную часть среднего геометрического трёх заданных вещественных чисел.

1.5. Определите, является ли заданное целое число A нечётным двузначным числом.

1.6. Определите, имеется ли среди заданных целых чисел A, B, C хотя бы одно чётное.

1.7. Даны три числа. Выберите те из них, которые принадлежат заданному отрезку [e, f].

1.8. Определите число, полученное выписыванием в обратном порядке цифр заданного целого трёхзначного числа.

1.9. Для заданных вещественных чисел a, b и c определите, имеет ли уравнение $ax^2 + bx + c = 0$ хотя бы одно вещественное решение.

1.10. Вычислите площадь кольца, ширина которого равна H, а отношение радиуса большей окружности к радиусу меньшей окружности равно D.

1.11. Определите, есть ли среди цифр заданного целого трёхзначного числа одинаковые.

1.12. Заданы площади круга и квадрата. Определите, поместится ли квадрат в круге.

1.13. Для задачи 1.12 определите, поместится ли круг в квадрате.

1.14. Заданы координаты двух точек. Определите, лежат ли они на одной окружности с центром в начале координат.

1.15. Определите, лежит ли заданная точка на одной из сторон треугольника, заданного координатами своих вершин.

1.16. Проверьте, можно ли построить треугольник из отрезков с длинами x, y, z и, если можно, то какой - остроугольный, прямоугольный или тупоугольный.

1.17. Проверьте, можно ли построить параллелограмм из отрезков с длинами x, y, v, w.

1.18. Даны координаты (как целые от 1 до 8) двух полей шахматной доски. Определите, может ли конь за один ход перейти с одного из этих полей на другое.

1.19. Треугольник задан величинами своих углов (град.) и радиусом описанной окружности. Вычислите стороны треугольника.

1.20. Смешали v_1 литров воды с температурой t_1 градусов Цельсия с v_2 литрами воды с температурой t_2 градусов Цельсия. Вычислите объем и температуру образовавшейся смеси.

1.21. Выберите наибольшее из трех заданных чисел.

1.22. Два прямоугольника заданы длинами сторон. Определите, можно ли первый прямоугольник целиком разместить во втором.

1.23. Значения заданных переменных a, b и c перераспределите таким образом, что a, b, c станут, соответственно, наименьшим, средним и наибольшим значениями.

1.24. Решите линейное уравнение $ax = b$.

1.25. Решите биквадратное уравнение $ax^4 + bx^2 + c = 0$.

1.26. Определите номер квадранта, в котором находится точка с заданными координатами (x, y).

1.27. Запишите заданное смешанное число в виде неправильной дроби.

1.28. Определите, пройдет ли кирпич с рёбрами a, b, c в прямоугольное отверстие со сторонами x и y. Просовывать кирпич в отверстие разрешается только так, чтобы каждое из его рёбер было параллельно или перпендикулярно каждой из сторон отверстия.

1.29. Идет k-ая секунда суток. Определите, сколько полных часов и полных минут прошло к этому моменту от начала суток.

1.30.* Найдите центр и радиус окружности, проходящей через три заданные точки на плоскости.

1.31.* Даны четыре точки на плоскости. Определите, можно ли построить треугольник с вершинами в этих точках такой, что оставшаяся точка окажется внутри треугольника.

1.32.* Определите, имеют ли общие точки две плоские фигуры - треугольник с заданными координатами его вершин и круг радиусом R с центром в начале координат.

1.33. В кубический, наполненный до краев аквариум со стороной a метров выпустили рыбу-шар диаметром b см. Вычислите, сколько процентов от первоначального объема воды выплеснется из аквариума (хвост и плавники рыбы не учитывайте).

1.34. Станции A, B и C расположены на n-м, m-м и p-м километрах железной дороги, соответственно. Какие из этих станций расположены наиболее близко друг к другу?

1.35. На карте координаты начала и конца строящегося прямолинейного участка шоссе обозначены как (x_1, y_1) и (x_1, y_2). Карьер, откуда можно брать гравий для стройки, имеет координаты (x_0, y_0), причем $x_0 <> x_1$. Определите минимальное расстояние от строящегося участка шоссе до карьера.

1.36. Составьте программу, играющую со своим автором в "Орел или решку".

1.37*. Любитель горнолыжного спорта собирается провести свой недельный отпуск на одном из трех курортов. Курорт A открыт с начала ноября по конец апреля, но из-за лавинной опасности его закрывают на весь январь. Курорт B открыт с начала декабря по конец марта. Его закрывают на соревнования с 1 по 15 февраля. Курорт C постоянно открыт с начала октября по конец мая. Стоимость отдыха на каждом из курортов, включая проезд, составляет, соответственно, P_1, P_2 и P_3 рублей. По дате начала отпуска определите, сможет ли он провести свой отпуск в горах, и какой минимальной суммой он должен располагать.

1.38*.Стартовый номер участника соревнований по автомотоспорту определяется на квалификационных заездах. При этом фиксируется время начала и конца прохождения так называемого "быстрого" круга (часы, минуты, секунды). Проверьте, корректно ли зафиксированы данные участника, и найдите время прохождения им "быстрого" круга.

*Алгоритмы, реализуемые с помощью циклов типа **для***

Циклы типа **для** применяются, когда число повторений цикла известно к началу его выполнения.

Язык	Пример	Величина шага
Школьный АЯ	**нц для** i **от** 1 **до** N тело цикла **кц**	Всегда 1
Pascal	**For** i := 1 **to** N **do** тело цикла ;	1
	For i := N **downto** 1 **do** тело цикла ;	-1
Basic	**FOR** I = 1 **TO** N **STEP** H тело цикла **NEXT** I	Шаг H произвольный (по умолчанию равен 1)

Пример 2.1. ***Вычислить сумму элементов числового массива*** $A = (a_1, a_2, \ldots, a_N)$.

Тест

Данные		Результат
N=5	A=(3, 5, -2, 6, 3)	S=15.0

Школьный АЯ

```
алг Сумма (арг цел N, арг вещ
           таб A[1:N], рез вещ S)
  дано N>0
нач цел i
  S:=0
  нц для i от 1 до N
    S := S + A[i]
  кц
кон
```

Исполнение алгоритма

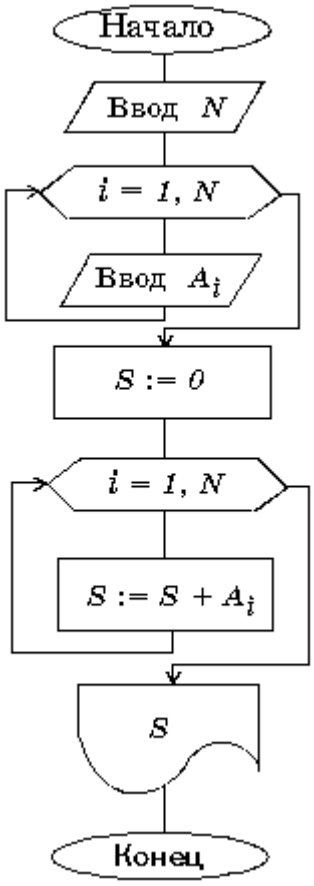
i	S
	0
1	$0 + a_1 = 0+3 = 5$
2	$a_1 + a_2 = 3+5 = 8$
3	$a_1+a_2+a_3 = 8-2 = 6$
4	$a_1+a_2+a_3+a_4 = 6+6 = 12$
5	$a_1+a_2+a_3+a_4+a_5 = 12+3=15$

Turbo Pascal

```
Program Summa;
  Uses Crt;
  Type Mas = Array [1..20] of Real;
  Var A    : Mas;
      i, N : Integer;
      S    : Real;
```

Блок-схема

```
BEGIN
  ClrScr;     {очистка экрана }
  Write('Введите N = ');
  ReadLn(N); {ввод значения N}
  For i := 1 to N do {цикл по элементам
массива}
    begin
      Write('A [ ', i , ' ] = ');
      ReadLn(A[i])   {ввод элементов массива}
    end;
  S := 0; {присваивание начального значения}
  For i := 1 to N do S := S+A[i];
{суммирование}
  WriteLn;
  WriteLn('Сумма равна ', S : 5 : 1);
  ReadLn
END.
```



Пример 2.2. ***Найти наибольший элемент числового массива $A = (a_1, a_2, ..., a_N)$ и его номер.***

Тест

Данные		Результаты	
N=4	A=(3, -1, 10, 1)	Amax=10	K=3

Школьный АЯ

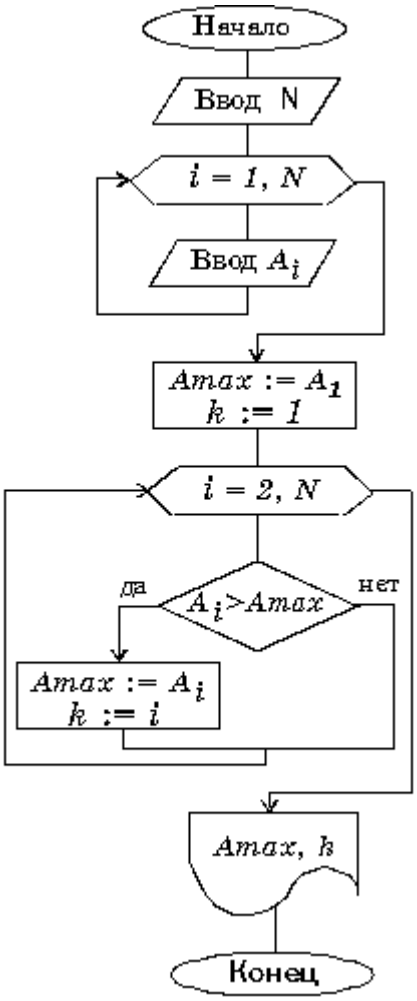
```
алг МаксЭлемент (арг цел N, арг вещ таб
A[1:N],
                 рез вещ Amax, рез цел k)
нач цел i
  Amax := A[1]; k := 1
  нц для i от 2 до N
    если A[i] > Amax
      то Amax:=A[i]; k := i
    все
  кц
кон
```

Блок-схема

Исполнение алгоритма

i	A[i] > Amax	Amax	k

2 3 4	- + -	3 10	1 3



Turbo Pascal

```
Program MaxElem;
  Uses Crt;
  Type Mas = Array [1..20] of Real;
  Var A     : Mas;
      i, N  : Integer;
      k     : Integer;
      Amax  : Real;
BEGIN
  ClrScr;
  Write('Введите N = ');
  ReadLn(N);
  For i := 1 to N do {Ввод значений элементов массива A}
    begin
      Write('A [ ', i, ' ] = '); ReadLn(A[i])
    end;
  Amax := A[1]; k:=1; {Поиск максимального элемента}
  For i := 2 to N do
    If A[i] > Amax then
      begin
        Amax := A[i]; k := i
      end;
  WriteLn; WriteLn('Наибольший элемент' , k , '-й');
  WriteLn('Его значение ', Amax : 5 : 1); ReadLn
END.
```

Пример 2.3. ***В баскетбольную команду могут быть приняты ученики, рост которых превышает 170 см. Составьте список кандидатов в команду из учеников класса.***

Система тестов

Номер теста	Проверяемый случай	Число учеников	Фамилии	Рост	Результаты
1	Есть кандидаты	3	Кулов Чехин Уваров	171 165 178	Кулов Уваров
2	Нет кандидатов	2	Ершов Иванов	170 165	Нет кандидатов

Школьный АЯ

```
алг Баскетбол (арг цел N, арг лит таб Фам[1:N], арг вещ
               таб Рост[1:N], рез лит таб Канд [1:N] )
нач цел i, k
  k:=0
  нц для i от 1 до N | запись фамилий кандидатов в таблицу Канд
    если Рост[i]>170
      то k:=k+1; Канд [k] := Фам [i]
    все
  кц
  если k=0
    то вывод "В КЛАССЕ НЕТ КАНДИДАТОВ В КОМАНДУ."
    иначе нц для i от 1 до k
            вывод Канд [i]
          кц
 все
кон
```

Исполнение алгоритма

N теста	i	Рост[i] > 170	K	Кондидаты в команду
1	1 2 3	+ - +	0 1 2	Кулов Уваров
2	1 2	- -	0	-

TurboPascal

```
Program BascetBall;
  Uses Crt;
  Var
     SurName : Array [1..30] of String;  { фамилии учеников }
     Height  : Array [1..30] of Real;    { рост учеников }
     Cand    : Array [1..30] of String;  { фамилии кандидатов }
```

```
     NPupil, i, K : Integer;                { NPupil - число
учеников,
                                              K - количество
зачисленных}
BEGIN ClrScr;
  Write('В КОМАНДУ ЗАЧИСЛЯЮТСЯ УЧЕНИКИ, ');
  WriteLn('РОСТ КОТОРЫХ ПРЕВЫШАЕТ 170 СМ.'); WriteLn;
  Write('Сколько всего учеников ? ');
  ReadLn(NPupil);
  WriteLn('Введите фамилии и рост учеников :');
  For i := 1 to NPupil do
    begin Write(i, '. Фамилия - '); ReadLn(SurName[i]);
          Write('     Рост - ');    ReadLn(Height[i]);
    end; WriteLn;
  K:=0; { Составление списка команды }
  For i := 1 to NPupil do
    If Height[i]>170 then
      begin K:=K+1; Cand[K] := SurName[i] end;
  If K=0 then WriteLn('В КЛАССЕ НЕТ КАНДИДАТОВ В КОМАНДУ.')
    else
      begin WriteLn('КАНДИДАТЫ В БАСКЕТБОЛЬНУЮ КОМАНДУ :');
            For i := 1 to K do WriteLn( i, '. ' , Cand[i]);
      end;
  ReadLn
END.
```

Пример 2.4. ***Для заданного x вычислить***

$$S = 1 - \frac{x}{1!} + \frac{x^2}{2!} - \frac{x^3}{3!} + \ldots + (-1)^n \frac{x^n}{n!}$$

Здесь n! = 1·2·3 ...· n (читается как "n-факториал").

Тест

Данные		Результат
X=1	n=3	$S = 1 - \frac{1}{1} + \frac{1^2}{1 \cdot 2} - \frac{1^3}{1 \cdot 2 \cdot 3} = 0.33$

Школьный АЯ

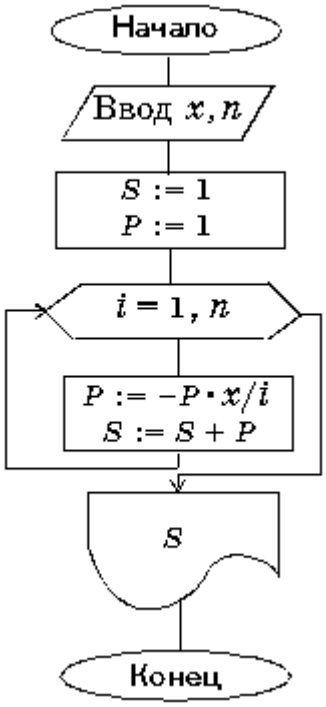
```
алг Сумма Ряда (арг вещ x, арг цел n, рез вещ S)
нач цел i, вещ P      | P - очередное слагаемое
  S := 1; P := 1
  нц для i от 1 до n
    P := - P*x /i     | получение очередного слагаемого
    S := S + P
  кц
кон
```

Turbo Pascal

```
Program SumUp;
```

Блок-схема

```
  Uses Crt;
  Var x, S, P : Real;
                {P - очередное слагаемое}
      i, n : Integer;
BEGIN ClrScr;
  Write('Введите n = ');  ReadLn(n);
  Write('Введите x = ');  ReadLn(x); WriteLn;
  S := 1; P := 1;
  For i := 1 to n do
    begin
      P := - P*x /i; {получение очередного
слагаемого}
      S := S + P
    end;
  WriteLn('O т в е т : S = ', S : 7 : 3 );
ReadLn
END.
```



Пример 2.5. ***Дан массив X(N). Получить новый массив Y(N) такой, что в нем сначала идут положительные числа, затем нулевые, и затем отрицательные из X.***

Тест

Данные	Результат
N=7 X=(-1, 2, 0, 4, -3,-2,0)	Y=(2, 4, 0, 0, -1, -3, -2)

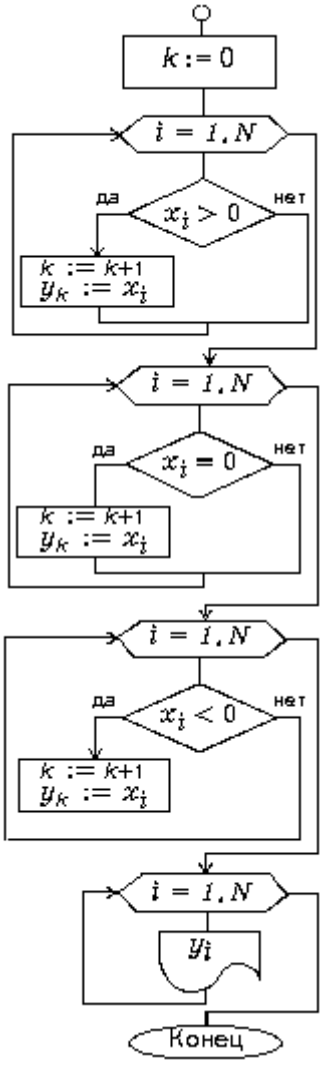
Школьный АЯ

```
алг Новый Порядок (арг цел N, арг вещ таб X[1:N], рез вещ таб
Y[1:N])
нач цел i, k    | k - индекс массива Y
  k := 0
  нц для i от 1 до N | Занесение в Y положительных чисел из X
    если X[i] > 0
       то k := k+1; Y[k] := X[i]
    все
  кц
  нц для i от 1 до N | Занесение в Y чисел, равных нулю, из X
    если X[i] = 0
      то k := k+1; Y[k] := X[i]
    все
  кц
  нц для i от 1 до N | Занесение в Y отрицательных чисел из X
    если X[i] < 0
      то k := k+1; Y[k] := X[i]
    все
  кц
кон
```

Turbo Pascal

```
Program NewOrder;
  Uses Crt;
  Var N, i, k : Integer;
      X, Y    : Array [1..20] of Real;
BEGIN
  ClrScr;
  Write('Введите N = '); ReadLn(N);
  For i := 1 to N do
    begin
      Write('X[ ', i, ' ] = '); ReadLn(X[i])
    end;
  k:=0;
  For i := 1 to N do
    If X[i]>0 then
      begin k:=k+1; Y[k]:=X[i]
      end;
  For i := 1 to N do
    If X[i]=0 then
      begin k:=k+1; Y[k]:=X[i]
      end;
  For i := 1 to N do
    If X[i]<0 then
      begin k:=k+1; Y[k]:=X[i]
      end;
  Write('О т в е т : полученный массив');
  For i := 1 to N do Write(Y[i] : 5 : 1);
  WriteLn; ReadLn
END.
```

Блок-схема
(фрагмент)



Задачи для самостоятельного решения

2.1. Подсчитайте число и сумму положительных, число и произведение отрицательных элементов заданного массива *A(N)*.

2.2. Заданные векторы *X(N)* и *Y(N)* преобразуйте по правилу: большее из x_i и y_i примите в качестве нового значения x_i , а меньшее — в качестве нового значения y_i .

2.3. Элементы заданного массива *B(N)* перепишите в новый массив *A(N)* в обратном порядке.

2.4. Из заданного вектора *A(3N)* получите вектор *B(N)*, очередная компонента которого равна среднему арифметическому очередной тройки компонент вектора *A*.

2.5. В заданном массиве *X(N)* замените нулями все отрицательные компоненты, непосредственно предшествующие его максимальной компоненте (первой по порядку, если их несколько).

2.6. Вычислите значения
а) $sin\ x\ +\ sin^2 x\ +\ ...\ +\ sin^n x$;
б) $sin\ x\ +\ sin\ x^2\ +\ ...\ +\ sin\ x^n$;
в) $sin\ x\ +\ sin^2 x^2\ +\ ...\ +\ sin^n x^n$;
г) $sin\ x\ +\ sin\ sin\ x\ +\ ...\ +\ sin\ sin...sin\ x$ (n раз).

2.7. Вычислите сумму квадратов всех элементов заданного массива *X(N)*, за исключением элементов, кратных пяти.

2.8. Вычислите значения функции $z = (a + b + c_i\ )\ /\ i$, если a изменяется от 0 с шагом 1, b изменяется от 5 с шагом 1, c_i является элементом массива $C(N)$. При этом a и b изменяются одновременно с i.

2.9. В заданном массиве *A(N)* поменяйте местами наибольший и наименьший элементы.

2.10. В заданном массиве *A(N)* определите количество элементов, которые меньше заданного значения.

2.11. Осуществите циклический сдвиг компонент заданного вектора $A(N)$ влево на одну позицию, то есть получите вектор $A\ =\ (a_2\ ,\ a_3\ ,\ ...,\ a_N\ ,\ a_1\ )$.

2.12. Осуществите циклический сдвиг компонент заданного вектора $A(N)$ вправо на две позиции, то есть получите вектор $A\ =\ (a_{N-1}\ ,\ a_N\ ,\ a_1\ ,\ a_2\ \ ,\ ...\ ,\ a_{N-2}\ )$.

2.13. Дан массив *A(N)*. Получите массив *B(N)*, i-й элемент которого равен среднему арифметическому первых i элементов массива A: $b_i = (a_1\ +\ a_2\ +\ ...\ +\ a_i\ )\ /\ i.$

2.14. Вычислите значения многочленов:
$P = a_n x^n\ +\ a_{n-1}\ x^{n-1}\ +\ ...\ +\ a_1\ x\ +\ a_0\ ;$
$Q = a_0 x^n\ +\ a_1\ x^{n-1}\ +\ ...\ +\ a_{n-1}\ x\ +\ a_n,$
используя формулу Горнера. Коэффициенты многочленов заданы в виде вектора $A = (a_0\ ,\ a_1\ ,\ ...\ ,\ a_n)$.

2.15. Запишите подряд в массив *A(N)* элементы заданного массива *B(2N)*, стоящие на чётных местах, а элементы, стоящие на нечетных местах, запишите в массив *C(N)*.

2.16. Выведите на печать номера элементов заданного массива *Y(N)*, удовлетворяющих условию $0 < y_i < 1$.

2.17. Выведите на печать номера точек, лежащих в круге радиусом R с центром в начале координат. Координаты точек заданы массивами $X(N)$ и $Y(N)$.

2.18. В заданном массиве $A(N)$ вместо a_1 запишите наибольший элемент массива, а вместо a_N — наименьший элемент массива.

2.19. В заданном массиве *A(N)*, все элементы которого попарно различны, найдите:
а) наибольший элемент из отрицательных;
б) наименьший элемент из положительных;
в) второй по величине элемент.

2.20. В заданном массиве *A(N)* определите число соседств:
а) двух положительных чисел;

б) двух чисел разного знака;
в) двух чисел одного знака, причем абсолютная величина первого числа должна быть больше второго числа;
г) чётного числа и нечётного с нечётным индексом.

2.21. В заданном массиве $A(N)$ положительные элементы уменьшите вдвое, а отрицательные замените на значения их индексов.

2.22. В заданном массиве $A(N)$ вычислите среднее геометрическое и среднее арифметическое значения для положительных элементов.

2.23. Вычислите $P = 1 \cdot 2 + 2 \cdot 3 \cdot 4 + 3 \cdot 4 \cdot 5 \cdot 6 + \ldots + N \cdot (N+1) \cdot \ldots \cdot 2N$.

2.24. Образуйте массив B, состоящий из положительных элементов заданного массива $A(N)$, больших пяти. Выведите на печать образованный массив и число его элементов.

2.25. Из заданных векторов $X(N)$ и $Y(N)$ получите вектор $Z(2N)$ с элементами $(x_1, y_1, x_2, y_2, \ldots, x_N, y_N)$.

2.26. Для заданного вектора $X(2N)$ вычислите $Y = x_1 - x_2 + x_3 - \ldots - x_{2N}$.

2.27. Дан вектор $A(N)$. Найдите порядковый номер того из элементов, который наиболее близок к какому-нибудь целому числу (первому по порядку, если таких несколько).

2.28. Элементы заданного массива $X = (x_1, x_2, \ldots, x_N)$ переупорядочите следующим образом: $X = (x_N, x_{N-1}, \ldots, x_1)$.

2.29. Для заданного набора коэффициентов a, b, c, d найдите наименьшее значение функции $y = a x^3 + b x^2 + cx + d$ и значение аргумента, при котором оно получено. Значение x изменяется от 0 до 2 с шагом 0,2.

2.30. Дано натуральное N. Вычислите сумму тех элементов серии $i^3 - 3 \cdot i \cdot N + N$, $i = 1, 2, \ldots, N$, которые являются удвоенными нечётными числами.

2.31*. Сожмите заданный массив $A(N)$ отбрасыванием нулевых элементов.

2.32. Дан массив $A(2N)$. Постройте массивы с элементами, соответственно равными:
а) $a_1, a_{N+1}, a_2, a_{N+2}, \ldots, a_N, a_{2N}$;
б) $a_{2N}, a_1, a_{2N-1}, a_2, \ldots, a_{N+1}, a_N$.

2.33. Дана матрица $A(3, N)$, элементы которой положительны. Определите, какие из троек a_{1i}, a_{2i}, a_{3i} $(i = 1, \ldots, N)$ могут служить сторонами треугольника. Выведите массив $b_1, \ldots, b_N$, состоящий из нулей и единиц. Если тройка a_{1i}, a_{2i}, a_{3i} может служить сторонами треугольника, то $b_i = 1$, если нет, то $b_i = 0$.

2.34. У кассы аэрофлота выстроилась очередь из N человек. Время обслуживания кассиром i-го клиента равно T_i $(i = 1, \ldots, N)$.
а) Определите время пребывания в очереди каждого клиента;
б) Укажите номер клиента, для обслуживания которого кассиру потребовалось больше всего время.

2.35. В соревнованиях по фигурному катанию N судей независимо выставляют оценки спортсмену. Затем из объявленных оценок удаляют самую высокую (одну, если самую высокую оценку выставили несколько судей). Аналогично поступают с самой низкой оценкой. Для оставшихся оценок вычисляется среднее арифметическое, которое и становится зачетной оценкой. По заданным оценкам судей определите зачетную оценку спортсмена.

2.36. Несколько однотипных спасательных катеров, находящихся в акватории в точках с координатами (x_i , y_i), $i = 1, 2, ..., N$, получили сигнал SOS от судна, находящегося в той же акватории в точке с координатами (x_0 , y_0). Определите, какой из катеров быстрей других сможет оказать помощь?

2.37. По данным о расписании движения пригородных поездов определите значение наибольшего интервала времени между отправлениями поездов.

2.38. Учитель объявил результаты контрольной работы. Определите процентное содержание выставленных им "пятерок", "четверок", "троек" и "двоек".

2.39. Фунт стерлингов, денежная единица Великобритании, до 1971 г. равнялся 20 шиллингам или 240 пенсам. С проходящего корабля в порту Ливерпуля сошли N путешественников, каждый из которых имел по одной десятифунтовой купюре. Они купили сувениры на сумму $p_1 , p_2 , ..., p_n$, соответственно. Сколько фунтов, шиллингов и пенсов сдачи получил каждый из путешественников?

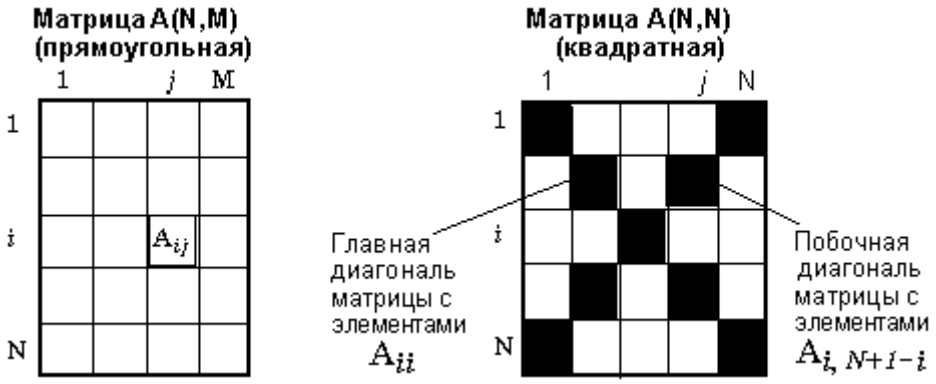
2.40. О каждом учащемся класса известны его пол, год рождения, рост и вес. Определите, сколько в классе мальчиков и сколько девочек. Найдите средний возраст мальчиков и средний возраст девочек. Определите, верно ли, что самый высокий мальчик весит больше всех в классе, а самая маленькая девочка является самой юной среди девочек.

Алгоритмы, реализуемые с помощью вложенных циклов типа **для**

Язык	*Схемы вложенных циклов типа* ***для***
Школьный АЯ	**нц для** i **от** A1 **до** B1 тело внешнего цикла . . . . . . **нц для** j **от** A2 **до** B2 тело внутреннего цикла . . . . . . **кц** . . . . . . **кц**

Pascal	**For** i := A1 **to** B1 **do**
	begin . . . . . .
	For j := A2 **to** B2 **do**
	begin
	. . . . . .
	end;
	. . . . . .
	end;

Вложенные циклы типа **для** особенно часто используются при обработке матриц (двумерных массивов, прямоугольных таблиц) и векторов (одномерных массивов, линейных таблиц):

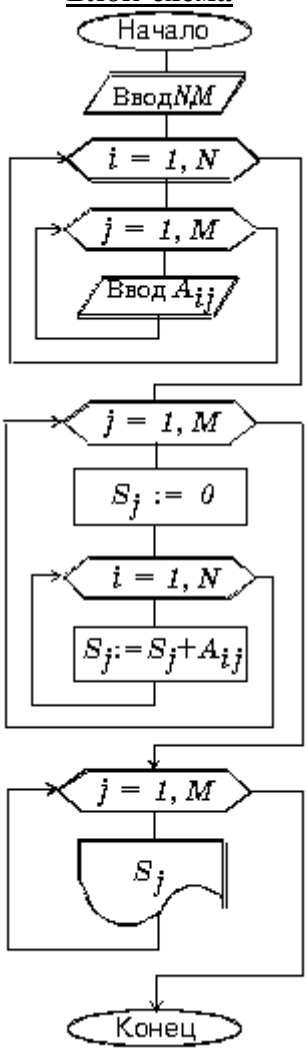


Пример 3.1. ***Вычислить суммы элементов столбцов заданной матрицы A(N, M).***

Тест

Данные		Результат
N=2 M=2	$A=\begin{pmatrix}2 & 1\\ 4 & 3\end{pmatrix}$	S=(6,74)

Блок-схема



Школьный АЯ

```
алг Суммы столбцов (арг цел N, M, арг вещ таб
A[1:N, 1:M],
                    рез вещ таб S[1:M])
  дано | N>0, M>0
нач цел i, j
  нц для j от 1 до M | цикл по столбцам
    S[j]:=0
    нц для i от 1 до N | цикл по элементам
      S[j]:=S[j] + A[i, j] |текущего столбца
    кц
  кц
кон
```

Исполнение алгоритма

j	i	S[i]
1	1 2	S1=0 S1=0+2=2 S1=2+4=6
2	1 2	S2=0 S2=0+1=1 S2=1+3=4

Turbo Pascal

```
Program SumColumn;
  Uses Crt;
  Var A          : Array [1..10, 1..10] of Real;
      N, M, i, j : Integer;
      S          : Array [1..10] of Real;
  {-----------------------------------------}
  Procedure InputOutput;
    Begin {описание процедуры ввода-вывода исходных данных}
      ClrScr;
      Write('Количество строк - '); ReadLn(N);
      Write('Количество столбцов - '); ReadLn(M);
      For i := 1 to N do {Ввод элементов матрицы}
        For j := 1 to M do
          begin Write('A[' , i , ', ' , j , ']= ? '); {вывод
запроса}
                ReadLn(A[i, j]) {ввод значения}
          end; WriteLn;
      ClrScr;
      WriteLn(' Матрица A');
      For i := 1 to N do {Вывод матрицы по строкам}
```

```
        begin
          For j := 1 to M do Write(A[i, j] : 5 : 1); {вывод i-ой
строки}
          WriteLn {перенос курсора на начало следующей строки}
        end; WriteLn
    End; { of InputOutput }
  {-----------------------------------------}
  Procedure SumCol;
    Begin {описание процедуры вычисления сумм элементов
столбцов}
      For j := 1 to M do {цикл по столбцам матрицы}
        begin
          S[j] := 0; {обнуление суммы элементов j-го столбца}
          For i := 1 to n do S[j] := S[j] + A[i, j] {накопление
суммы}
        end;
    End; { of SumCol }
  {-----------------------------------------}
  Procedure OutResult; {описание процедуры вывода результатов}
    Begin
      Write( 'О т в е т : Суммы элементов столбцов равны ');
      For j := 1 to M do Write(S[j] : 5 : 1); WriteLn; ReadLn
    End; { of OutResult }
  {-----------------------------------------}
BEGIN
  InputOutput; {вызов процедуры ввода-вывода исходных данных }
  SumCol; {вызов процедуры вычисления сумм }
  OutResult; {вызов процедуры вывода результатов}
END.
```

Пример 3.2. ***Подсчитать, сколько раз встречается в заданной целочисленной матрице A(N, M) максимальное по величине число.***

Тест

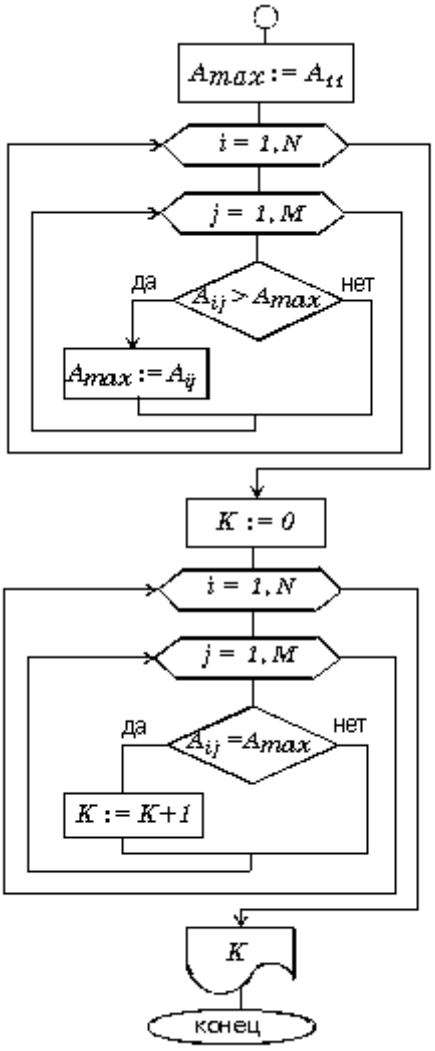
Данные		Результат
N=2 M=3	A=$\begin{pmatrix} 1 & 2 & 5 \\ 5 & 1 & 5 \end{pmatrix}$	K=3

Школьный АЯ

```
алг Количество максимумов(арг цел N,M, арг цел таб A[1:N,1:M],
рез цел K)
нач цел i, j, Amax
  Amax := A[1, 1] | Поиск максимального элемента матрицы
  нц для i от 1 до N
    нц для j от 1 до M
      если A[i, j] > Amax
        то Amax := A[i, j]
      все
    кц
  кц
```

```
  K := 0 | подсчет количества вхождений Amax
  нц для i от 1 до N
    нц для j от 1 до M
      если A[i, j] = Amax
        то K := K+1
      все
    кц
  кц
кон
```

Блок-схема (фрагмент)



Исполнение алгоритма

i	j	A[i,j]>Amax	Amax
			1
1	1	-	
	2	+	2
	3	+	5
2	1	-	
	2	-	
	3	-	

(продолжение)

i	j	A[i,j]=Amax	K
1	1	-	0
	2	-	
	3	+	1
2	1	+	2
	2	-	
	3	+	3

Turbo Pascal

```
Program NumberOfMaximums;
  Uses Crt;
  Type Mas = Array [1..10, 1..10] of Integer;
  Var A                          : Mas;
      N, M, K, Amax, i, j : Integer; {K - искомый результат}
{-----------------------------------------}
  Procedure InputOutput(Var A : Mas);
    Begin {описание процедуры ввода-вывода матрицы}
      ClrScr;
      Write('Количество строк - '); ReadLn(N);
```

```
      Write('Количество столбцов - '); ReadLn(M);
      For i := 1 to N do {Ввод матрицы}
        For j := 1 to M do
          begin Write('A[' , i , ', ' , j , ']= ? ');
                ReadLn(A[i, j])
          end; WriteLn;
      ClrScr;  WriteLn(' Матрица A');
      For i := 1 to N do {Вывод матрицы}
        begin
          For j := 1 to M do Write(A[i, j] : 5 );
          WriteLn
        end; WriteLn
    End; { of InputOutput }
    {-------------------------------------------}
  Procedure MaxElement(Var A : Mas; Var Amax : Integer);
    Begin {описание процедуры поиска Amax}
      Amax := A[1, 1];
      For i := 1 to N do
        For j := 1 to M do
          If A[i, j] > Amax then Amax := A[i, j];
    End; {of MaxElement}
    {-------------------------------------------}
  Procedure HowMuch(Var A : Mas; K : Integer);
    Begin {описание процедуры подсчета числа вхождений Amax}
      K:=0;
      For i := 1 to N do
        For j := 1 to M do
          if A[i, j] = Amax then K := K+1;
      WriteLn('Максимальное число ', Amax : 3 ,
              ' встречается ', K, ' раз(а)'); ReadLn;
    End; {of HowMuch}
{-------------------------------------------}
BEGIN
  InputOutput(A);       {Вызов процедуры ввода-вывода матрицы}
  MaxElement(A, Amax); {Вызов процедуры поиска макс. элемента}
  HowMuch(A, K)         {Вызов процедуры подсчета числа
                         вхождений максимального элемента }
END.
```

Пример 3.3. ***В заданной матрице A(N, M) поменять местами строки с номерами P и Q***
(1 <= P <= N, 1 <= Q <= N).

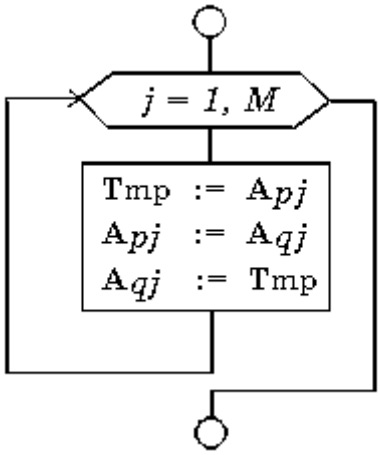
Тест

Данные	Результат
N=3 M=3 P=1 Q=3 $A=\begin{pmatrix}1 & 2 & 1\\ 2 & 2 & 2\\ 3 & 1 & 3\end{pmatrix}$	$A=\begin{pmatrix}3 & 1 & 3\\ 2 & 2 & 2\\ 1 & 2 & 1\end{pmatrix}$

Школьный АЯ

```
алг Поменять местами строки (арг цел N, M, арг цел P, Q,
                             арг рез вещ таб A[1:N, 1:M] )
нач цел j, вещ Tmp
  нц для j от 1 до M | цикл по элементам строк матрицы
    Tmp:=A[P, j]; A[P, j]:=A[Q, j]; A[Q, j]:=Tmp
  кц
кон
```

Блок-схема (фрагмент)



Исполнение алгоритма

j	Tmp	A[1,j]	A[3,j]
1	1	3	1
2	2	1	2
3	1	3	1

Turbo Pascal

```
Program Exchange;
  Uses Crt;
  Type Mas = Array [1..10, 1..10] of Real;
  Var A   : Mas;
  N, M, P, Q, i, j : Integer;
  Tmp   : Real;
{-----------------------------------------------------------------}
 Procedure InputOutput(Var A:Mas); {описание процедуры ввода-
вывода}
  Begin
    ClrScr;
    Write('Количество строк - '); ReadLn(N);
    Write('Количество столбцов - '); ReadLn(M);
    For i := 1 to N do
      For j := 1 to M do
        begin Write('A[' , i , ', ' , j , '] = ? ');
              Read(A[i, j])
        end; WriteLn;
    WriteLn('Номера строк, которые нужно поменять местами :');
    Write('P = '); ReadLn(P); Write('Q = '); ReadLn(Q);
WriteLn;
    ClrScr; WriteLn('Исходная матрица : ' ) ;
    For i := 1 to N do
      begin
        For j := 1 to M do Write(A[i, j] : 5 : 1);
        WriteLn
      end; WriteLn
  End; { of InputOutput}
{-----------------------------------------------------------}
```

```
 Procedure Change(P, Q: Integer); {описание процедуры замены
строк }
  Begin
    For j := 1 to M do
      begin Tmp:=A[P, j]; A[P, j]:=A[Q, j]; A[Q, j]:=Tmp end;
  End; { of Change}
{-------------------------------------------------------}
 Procedure OutRes(Var A:Mas); {описание процедуры вывода
результатов}
  Begin
    WriteLn('Матрица-результат :');
    For i := 1 to N do
      begin
        For j := 1 to M do Write(A[i, j] : 5 : 1) ;
        WriteLn
      end; ReadLn
  End; { of OutRes}
{-------------------------------------------------------}
BEGIN
  InputOutput(A); {вызов процедуры ввода-вывода исходных данных}
  Change(P, Q); {вызов процедуры замены строк }
  OutRes(A) {вызов процедуры вывода результатов}
END.
```

Пример 3.4. *Элементы заданного числового массива $a_1, a_2, ..., a_N$ упорядочить по возрастанию.*

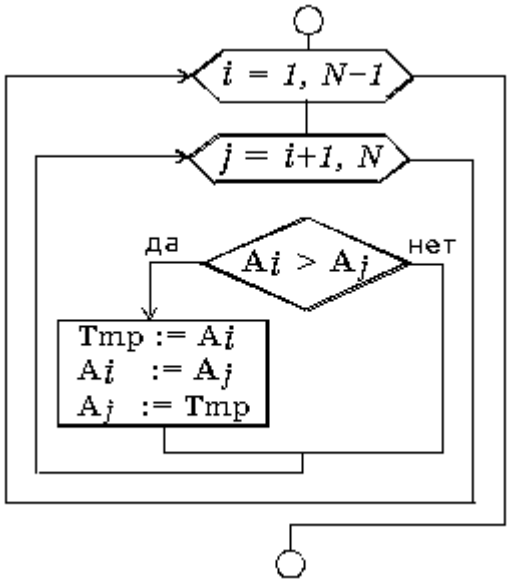
Тест

Данные		Результат
N=4	A=(5, 2, 7, 1)	A=(1, 2, 5, 7)

Школьный АЯ

```
алг Возрастание(арг цел N, арг
рез
                  вещ таб A[1:N])
нач цел i, j, вещ Tmp
  нц для i от 1 до N-1
    нц для j от i+1 до N
      если A[i] > A[j]
        то Tmp:=A[i]; A[i]:=A[j];
A[j]:=Tmp
      все
    кц
  кц
кон
```

Блок-схема (фрагмент)



Исполнение алгоритма

i	j	A[i]>A[j]	Массив A
1	2	+	2, 5, 7, 1
	3	-	
	4	+	

			1, 5, 7, 2
2	3 4	- +	1, 2, 7, 5
3	4	+	1, 2, 5, 7

Turbo Pascal

```
Program Regulation;
  Uses Crt;
  Type Mas = Array [1..10] of Real;
  Var A       : Mas;
      i, j, N : Integer;
      Tmp     : Real;
   {-------------------------------------------}
  Procedure Input; {описание процедуры ввода массива }
    Begin
      ClrScr;
      Write('Введите N = '); ReadLn(N);
      WriteLn('Введите элементы массива: ');
      For i := 1 to N do
        begin Write('A [ ' , i , ' ] = ');
              ReadLn(A[i])
        end;
    End; {of Input}
   {-------------------------------------------}
  Procedure Regulate; {описание процедуры упорядочения по
возрастанию}
    Begin
      For i := 1 to N-1 do
        For j := i+1 to N do
          If A[i] > A[j] then
            begin Tmp:=A[i]; A[i]:=A[j]; A[j]:=Tmp
            end;
    End; {of Regulate}
   {-------------------------------------------}
  Procedure Output; {описание процедуры вывода результата}
    Begin
      WriteLn('Упорядоченный массив :');
      For i:=1 to N do Write( A[i] : 6 : 1);
      WriteLn; ReadLn
    End; {of Output}
   {-------------------------------------------}
BEGIN
  Input;    {вызов процедуры ввода массива }
  Regulate; {вызов процедуры упорядочения по возрастанию}
  Output    {вызов процедуры вывода результата}
END.
```

Пример 3.5. ***В массиве A(N, N) вычислить две суммы элементов, расположенных ниже и выше главной диагонали.***

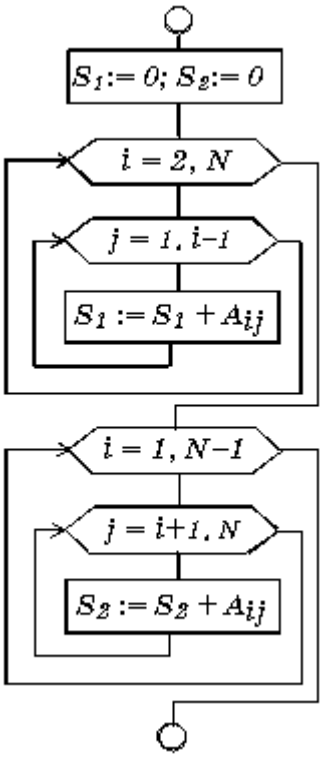
Тест

Данные	Результат
N=3 $A = \begin{pmatrix} 1 & 2 & 4 \\ 3 & 1 & 3 \\ 2 & 1 & 2 \end{pmatrix}$	$S_1=6$ $S_2=9$

Школьный АЯ

```
алг Две суммы (арг цел N, арг вещ таб A[1:N, 1:N], рез вещ S1,
S2)
  надо | S1 = сумма элементов ниже главной диагонали
       | S2 = сумма элементов выше главной диагонали
нач цел i, j
  S1:=0; S2:=0
  нц для i от 2 до N | циклы по элементам, расположенным
    нц для j от 1 до i-1 | ниже главной диагонали
      S1:=S1 + A[i, j]
    кц
  кц
  нц для i от 1 до N-1 | циклы по элементам, расположенным
    нц для j от i+1 до N | выше главной диагонали
      S2:=S2 + A[i, j]
    кц
  кц
кон
```

Блок-схема (фрагмент)



Исполнение алгоритма

i	j	S1	S2
			0
		0	
2	1	0+3=3	
3	1	3+2=5	
	2	5+1=6	
1	2		0+2=2
	3		2+4=6
2	3		6+3=9

Turbo Pascal

```
Program TwoSums;
  Uses Crt;
  Var A       : Array [1..10, 1..10] of Real;
      S1, S2  : Real; {S1, S2 - суммы элементов, расположенных
ниже
                           и выше главной диагонали,
соответственно}
      N, i, j : Integer;
{-----------------------------------------}
  Procedure InputOutput;{описание процедуры ввода-вывода
исходных данных}
    Begin ClrScr;
      Write('Количество строк и столбцов - '); ReadLn(N);
      For i := 1 to N do {Ввод матрицы}
        For j := 1 to N do
          begin Write('A[' , i , ', ' , j , '] = ? ');
                ReadLn(A[i, j])
          end; WriteLn;
      ClrScr; WriteLn(' Матрица A');
      For i := 1 to N do {Вывод матрицы}
        begin
          For j := 1 to N do Write(A[i, j] : 5 : 1);
          WriteLn
        end; WriteLn
    End; { of InputOutput }
{-----------------------------------------}
  Procedure Under;
    Begin {описание процедуры суммирования элементов, }
      S1 := 0; {расположенных ниже главной диагонали }
      For i := 2 to N do
        For j := 1 to i-1 do
          S1 := S1 + A[i, j];
      WriteLn('О т в е т :');
      WriteLn('Сумма элементов, лежащих ниже главной диагонали
=', S1:5:1);
    End;
{-----------------------------------------}
  Procedure Over;
    Begin {описание процедуры суммирования элементов,}
      S2 := 0; {расположенных выше главной диагонали}
      For i := 1 to N-1 do
        For j := i+1 to N do
          S2 := S2 + A[i, j];
      WriteLn('Сумма элементов, лежащих выше главной диагонали
=', S2:5:1);
      ReadLn
    End;
{-----------------------------------------}
BEGIN
  InputOutput; {Вызов процедуры ввода-вывода матрицы }
  Under; {Вычисление суммы элементов, лежащих ниже главной
диагонали}
```

```
  Over ; {Вычисление суммы элементов, лежащих выше главной
диагонали }
END.
```

Задачи для самостоятельного решения

3.1. Дана матрица *A(N, M)*. Найдите её наибольший элемент и номера строки и столбца, на пересечении которых он находится.

3.2. В каждой строке заданной матрицы *A(N, M)* вычислите сумму, количество и среднее арифметическое положительных элементов.

3.3. Для заданной целочисленной матрицы *A(N, M)* определите, является ли сумма её элементов чётным числом, и выведите на печать соответствующий текст.

3.4. Дана матрица *A(N, M)*. Найдите количество элементов этой матрицы, больших среднего арифметического всех её элементов.

3.5. Дана целочисленная матрица *A(N, M)*. Вычислите сумму и произведение тех её элементов, которые при делении на два дают нечётное число.

3.6. В заданной матрице *A(N, M)* поменяйте местами столбцы с номерами *P* и *Q*.

3.7. Дана матрица *A(N, M)*. Вычислите вектор *X(M)*, где значение X_j равно сумме положительных элементов *j*-го столбца матрицы *A*.

3.8. Дана матрица *A(N, M)*. Получите вектор *X(M)*, равный *P*-й строке матрицы, и вектор *Y(N)*, равный *Q*-му столбцу матрицы.

3.9. Дана матрица *A(N, M)*. Поменяйте местами её наибольший и наименьший элементы.

3.10. По заданному *n* постройте матрицы размером (*n*,*n*) вида

```
а)  1 0 ... 0      б)  n n-1 n-2 ... 1      в)  0 0 ...  0  1
    0 1 ... 0          0  n  n-1 ... 2          0 0 ...  1  2
     . . . .            .  .  . .  .             . . .  .  .
    0 0 ... 1          0  0   0  ... n          1 2 ... n-1 n
```

3.11. Даны две целочисленные матрицы *A(N, M)* и *B(N, M)*. Подсчитате (в отдельности) количество тех пар (a_{ij}, b_{ij}) , для которых:

а) $a_{ij} < b_{ij}$;
б) $a_{ij} = b_{ij}$;
в) $a_{ij} > b_{ij}$.

3.12. Дана матрица *A(N, N)*. Перепишите элементы её главной диагонали в одномерный массив *Y(N)* и разделите их на максимальный элемент главной диагонали.

3.13. Дана матрица *A(N, M)*. Получите $Y = X_1 \cdot X_N + X_2 \cdot X_{N-1} + \ldots + X_N \cdot X_1$, где X_i - наибольший элемент в строке с номером i матрицы A.

3.14. Постройте матрицу *A(N, N)*, элементы которой определяются равенствами $a_{ij} = i + 2 \cdot j$, а также найдите произведение чётных элементов этой матрицы, удовлетворяющих условию $a_{ij} < P$ $(0 < P < 3N)$.

3.15. Найдите наибольший элемент побочной диагонали заданной матрицы *A(N, N)* и выведите на печать всю строку, в которой он находится.

3.16. Дана целочисленная матрица *A(N, M)*. Вычислите сумму и произведение нечётных отрицательных элементов матрицы, удовлетворяющих условию $| a_{ij} | < i$.

3.17. Для заданной матрицы *A(N, N)* найдите:
а) сумму всех элементов;
б) сумму элементов главной диагонали;
в) значения наибольшего и наименьшего из элементов главной диагонали.

3.18. По трём заданным матрицам *A(N, N)*, *B(N, N)* и *C(N, N)* постройте матрицу *X* того же размера, каждый элемент которой вычисляется по формуле $x_{ij} = max \{ a_{ij}, b_{ij}, c_{ij} \}$.

3.19. Дана матрица *A(N, N)* и целое *P*. Преобразуйте матрицу по правилу: строку с номером *P* сделайте столбцом с номером *P*, а столбец с номером *P* сделайте строкой с номером *P*.

3.20. Для заданной матрицы *A(N, N)* найдите сумму элементов, расположенных в строках с отрицательным элементом на главной диагонали.

3.21. Дана матрица *A(N, M)*. Определите:
а) число ненулевых элементов в каждой строке матрицы;
б) общее число ненулевых элементов в матрице;
в) отношение числа ненулевых элементов в каждой строке матрицы к общему числу ненулевых элементов в матрице.

3.22. Вычислите матрицу *C(N, N)*, являющуюся суммой матриц *A(N, N)* и *B(N, N)*. Матрица *A* задана, а элементы матрицы *B* вычисляются по формуле

$$b_{ij} = \begin{cases} a_{ij}, \text{если } a_{ij} \geq 0, \\ 1, \text{если } a_{ij} < 0. \end{cases}$$

3.23. Из заданной матрицы *A(N, M)* удалите строку с номером *K* и столбцы с номерами *P* и *Q*. Полученную матрицу уплотните.

3.24. В заданном массиве *X(N, M)* все числа различны. В каждой строке выбирается минимальный элемент, затем среди этих чисел выбирается максимальное. Напечатайте номер строки массива *X*, в которой расположено выбранное число.

3.25. В заданном массиве *A(N, M)* переставьте строки так, чтобы суммы их элементов возрастали.

3.26. В заданном массиве *A(N, N)* вычислите две суммы элементов, расположенных выше и ниже побочной диагонали.

3.27. Дана матрица *A(N, M)*. Переставляя её строки и столбцы, добейтесь того, чтобы наибольший элемент (один из них) оказался в верхнем левом углу.

3.28. Расстояние между двумя множествами точек — это расстояние между наиболее близко расположенными точками этих множеств. Найдите расстояние между двумя заданными множествами точек на плоскости.

3.29. В заданном множестве точек на плоскости найдите пару точек с максимальным расстоянием между ними.

3.30. Задан список участников соревнований по плаванию и их результаты. Расположите результаты и фамилии участников в соответствии с занятым местом.

3.31. На основе сведений о ежедневном пробеге на тренировке спортсменов команды рассчитате среднесуточный и общий пробег каждого спортсмена за 20 дней.

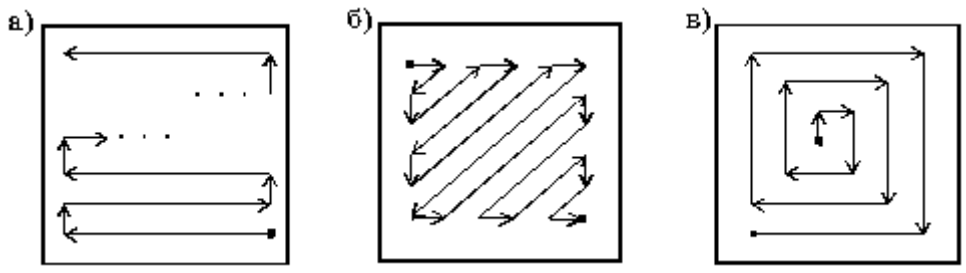
3.32. Известен расход по *N* видам горючего в каждом из *M* автохозяйств. Определите для каждого хозяйства вид горючего с наибольшим и с наименьшим расходом.

3.33. На основе сведений об отметках учеников класса в последней четверти
а) вычислите средние баллы по каждому предмету;
б) определите группу из пяти лучших учеников;
в) определите группу из пяти самых слабых учеников.

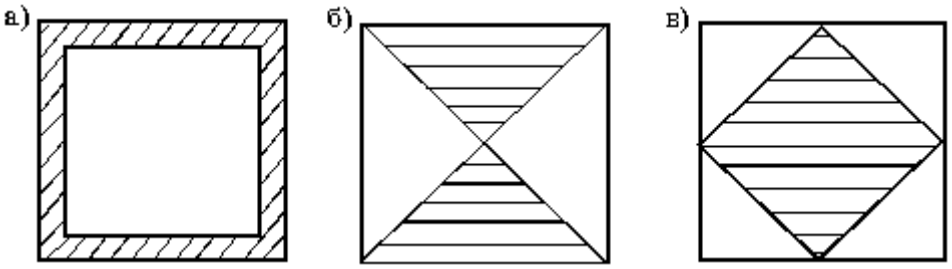
3.34. Заданы запасы по *N* видам топлива в каждом из *M* районов города и нормы минимально допустимого запаса по каждому виду топлива. Определите:
а) запасы каждого вида топлива на все районы;
б) запасы всех видов топлива для каждого района;
в) в каких районах запас того или иного вида топлива меньше минимально допустимого и каких видов топлива запасено недостаточно в каждом районе.

3.35. Последовательно (в лексикографическом порядке) напечатайте всевозможные перестановки целых чисел *1, 2, ..., n*. Значение *n* задано.

3.36*. Напечатать элементы заданной матрицы *A(N, N)* в следующем порядке:



3.37*. Дана матрица *A(N, N)*, где *N* — нечётное число. Вычислите сумму её элементов из заштрихованной области.



3.38. Шестизначный номер автобусного билета называют "счастливым", если равны суммы его первых трёх и последних трёх цифр. Подсчитате количество "счастливых" билетов.

3.39. Дана последовательность целых чисел $a_1, a_2, ..., a_m$. Постройте на ее основе новую последовательность, содержащую только те числа, которые в исходную последоватеьность входят по одному разу.

3.40. Даны два множества целых чисел: $a_1, a_2, ..., a_n$ и $b_1, b_2, ..., b_m$. *Среди* $a_1, a_2, ..., a_n$ нет повторяющихся чисел, нет их и среди $b_1, b_2, ..., b_m$. Постройте:
а) пересечение и объединение этих множеств;
б) множество, содержащее все члены множества $b_1, b_2, ..., b_m$, *которые не входят в множество* $a_1, a_2, ..., a_n$.

3.41. Вычислите $P = 1 \cdot 2 + 2 \cdot 3 \cdot 4 + 3 \cdot 4 \cdot 5 \cdot 6 + ... + N \cdot (N+1) \cdot ... \cdot 2N$.

3.42. Дана квадратная таблица *A(N, N)*, элементами которой являются нули и единицы. Подсчитайте, сколько в ней содержится квадратов, состоящих из единиц, со стороной из двух элементов таблицы и развернутых по отношению к таблице на 45 градусов.

3.43. Дана квадратная таблица *A(N, N)*, элементами которой являются нули и единицы. Не проверяя значений элементов таблицы, замените каждый из нулей на единицу, а каждую из единиц — на ноль.

3.44. Имеется *N* партий микросхем одного вида. Из каждой партии для контроля отобрали *M* микросхем. Тестовый контроль определил годность или негодность каждой микросхемы. Для того, чтобы вся партия была забракована, достаточно обнаружить в этих *M* выбранных микросхемах *K* негодных. По данным тестового контроля определите количество негодных микросхем в каждой партии и число забракованных партий.

3.45. Числом Армстронга называется целое *n*-значное число, сумма *n*-х степеней цифр которого равна самому этому числу. Например, числом Армстронга является число 407, так как $407 = 4^3 + 0^3 + 7^3$. Найдите все числа Армстронга для заданного $n <= 10$.

Алгоритмы, реализуемые с помощью циклов типа ПОКА

С помощью циклов типа ***пока*** можно запрограммировать любые повторяющиеся фрагменты алгоритмов. Но на практике цикл типа ***пока*** чаще всего используют в двух следующих случаях:

- **Число повторений заранее не известно** (например, цикл до достижения требуемой точности результата, цикл до первого отрицательного элемента массива и т.п.). Такой цикл называется циклом типа ***пока* с прерыванием**.
- **Число повторений заранее известно, но шаг параметра цикла не равен 1** (в школьном АЯ) **или 1, –1** (в Pascal). Такой цикл называется циклом типа ***пока* без прерывания**.

Цикл типа *пока* с прерыванием

Язык	Пример	Пояснения

Школьный АЯ	`i:=1; Flag:="Нет"`
	`нц пока (i<=N) и (Flag="Нет")`
	`  если A[i]<0`
	`  то Flag:="Да"; k:=i`
	`  иначе i:=i+1`
	`  все`
	`кц`
		Решается задача: определить номер первого отрицательного элемента массива A(N). Здесь **Flag** — "управляющая" переменная **литерного** типа (можно с успехом использовать также **логический** или **целый** типы)

Pascal	`i:=1; Flag:=FALSE;`
	`While (i<=N) and not Flag do`
	`  If A[i]<0`
	`  then begin`
	`  Flag:=TRUE; k:=i`
	`  end`
	`  else i:=i+1;`
		Здесь Flag — переменная **логического** типа, принимающая значение TRUE (истина) или FALSE (ложь),
		and - операция '**и**', **not** - операция '**не**'

Цикл типа *пока* без прерывания

Язык	Пример	Пояснения

Школьный АЯ	`i:=1; S:=0`
	`нц пока i<=N`
	`  S:=S+A[i]`
	`  i:=i+2`
	`кц`
		Вычисляется **сумма элементов массива**

Pascal	i:=1; S:=0;
	While i<=N do
	begin S:=S+A[i];
	i:=i+2
	end;
		A(N)
		с нечетными индексами.
		Число таких элементов заранее известно. Шаг параметра цикла равен **двум**

Для организации циклов типа ***пока*** можно также использовать:

- в языке **Pascal** оператор цикла с постусловием **Repeat...until**:

Repeat **тело цикла** **until <условие завершения>**	**Повторять тело цикла до тех пор, пока не выполнится условие завершения цикла.**

Пример 4.1. ***Определить, является ли заданная последовательность чисел $a_1, a_2, \ldots, a_N$ монотонно убывающей.***

Система тестов

Номер теста	Проверяемый случай	Данные		Результат
		N	Вектор A	Otvet
1	Является	3	(3, 2, 1)	'Да'
2	Не является	3	(2, 3, 1)	'Нет'

Школьный АЯ

```
алг Убывание (арг цел N, арг вещ таб
A[1:N],
              рез лит Otvet)
нач цел i
  i:=1; Otvet:="Да"
  нц пока (i<=N-1) и (Otvet="Да")
    если A[i] < A[i+1]
       то Otvet := "Нет"
       иначе i:=i+1
    все
  кц
кон
```

Блок-схема (фрагмент)

Исполнение алгоритма

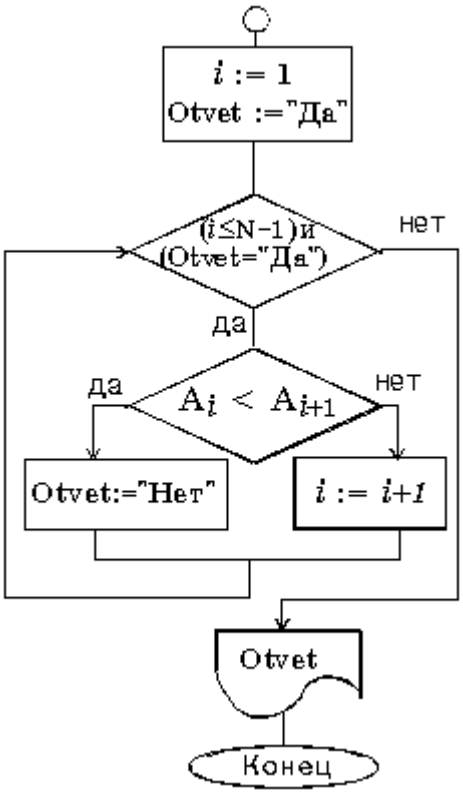
Обозначения проверяемых условий:

(i <= N-1) и (Otvet = "Да") => (1)

A[i] < A[i+1] => (2)

N	i	Otvet	(1)	(2)

теста				
1	1 2 3	"Да"	+ + - (кц)	- -
2	1	"Да" "Нет"	+ - (кц)	+



Turbo Pascal

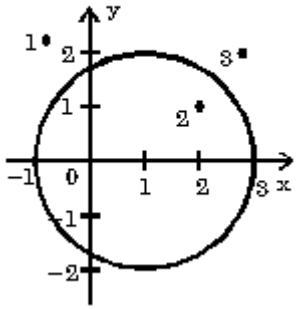
```
Program Decrease;
  Uses Crt;
  Var A    : Array [1..10] of Real;
      N, i : Integer;
      Otvet: Boolean;
{-------------------------------------------}
Procedure InputOutput; {описание процедуры ввода-вывода данных}
 Begin
  ClrScr;
  Write('Количество элементов - '); ReadLn(N);
  For i := 1 to N do
   begin Write('A[' , i , '] = ');
         ReadLn(A[i])
   end; WriteLn;

  WriteLn('Заданная последовательность чисел');
  For i := 1 to N do Write(A[i] : 5 : 1);
  WriteLn
 End; { of InputOutput }
{-------------------------------------------}
Procedure Processing( Var Otvet: Boolean);
 Begin                {описание процедуры проверки на убывание
элементов}
  Otvet := TRUE; i:=1;
  While (i<=N-1) and Otvet do
   If (A[i]<A[i+1]) then Otvet := FALSE
                    else i := i+1;
 End; { of Processing }
{-------------------------------------------}
```

```
Procedure Result(Otvet: Boolean);  {описание процедуры вывода
результата}
 Begin
  If Otvet then Write('образует ')
           else Write('не образует ');
  WriteLn('монотонно убывающую последовательность.');
  ReadLn
 End;
{-----------------------------------------}
BEGIN
  InputOutput;        {вызов процедуры ввода-вывода}
  Processing(Otvet); {вызов процедуры проверки на убывание}
  Result(Otvet);      {вызов процедуры вывода результата}
END.
```

Пример 4.2. ***Задано множество точек на плоскости. Определить, принадлежит ли хотя бы одна точка множества внутренней области круга с центром в точке (a, b) и радиусом R.***

Тест 1



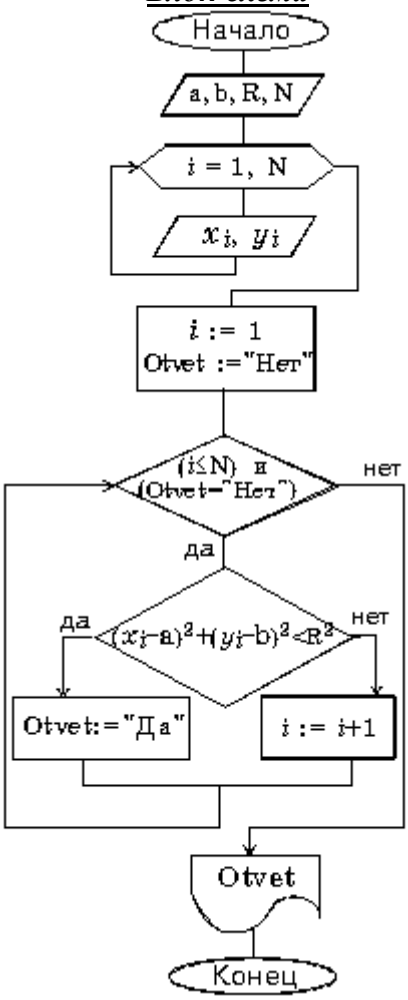
Система тестов

Номер теста	Проверяемый случай	Данные					Результат
		a	b	R	Кол. точек	Координаты точек	Otvet
1	Принадлежит	1	0	2	3	X=(-1, **2**, 3) Y=(2, **1**, 2)	"Да"
2	Не принадлежит	1	0	2	2	X=(-1, 3) Y=(2, 2)	"Нет"

Школьный АЯ

```
алг Точки (арг цел N, арг вещ таб X [1 : N] , Y [1 : N] ,
           арг вещ a, b, R, рез лит Otvet)
нач цел i
  i:=1;   Otvet:="Нет"
  нц пока (i<=N) и (Otvet="Нет") | условие продолжения цикла
    если (X[i]-a)**2 + (Y[i]-b)**2 <R*R | условие прерывания
цикла
      то Otvet := "Да"
      иначе i:=i+1
    все
  кц
кон
```

Блок-схема



Исполнение алгоритма

Обозначения проверяемых условий:

(i <= N) и (Otvet = "Нет") => (1)

(X[i]-1)2 + (Y[i]-b)**2 < R*R** => (2)

N теста	i	Otvet	(1)	(2)
1	1 2 3	"Нет" "Да"	+ + - (кц)	- +
2	1 2 3	"Нет"	+ + - (кц)	- -

Turbo Pascal

```
Program SetOfPoints;
  Uses Crt;
  Type Mas = Array [1..20] of Real;
  Var X, Y          : Mas;      {массивы координат точек }
      i, NPoints    : Integer;  {NPoints – количество точек}
      a, b, Radius  : Real;     {координаты центра и радиус}
      Flag          : Boolean;
{-------------------------------------------}
Procedure Input; {описание процедуры ввода данных}
 Begin
  ClrScr;
  Write('Введите координаты центра круга: '); ReadLn(a, b);
  Write('Введите радиус круга: ');            ReadLn(Radius);
  Write('Введите количество точек: ');        ReadLn(NPoints);
  For i := 1 to NPoints do
    begin
       WriteLn(i : 4, '-ая точка ');
       Write('X = ');  ReadLn(X[i]);
       Write('Y = ');  ReadLn(Y[i]);
    end; WriteLn
 End;            {of Input}
{-------------------------------------------}
```

```
Procedure Inside(Var Flag : Boolean); {описание процедуры
проверки }
 Begin                                  {принадлежности точек
области}
   Flag := FALSE ; i := 1;
   While (i<=NPoints) and not Flag do
     If Sqr(X[i]-a)+Sqr(Y[i]-b) < Sqr(Radius) {Sqr - возведение
в квадрат}
        then Flag := TRUE
        else i:=i+1;
 End; {of Inside}
{-----------------------------------------}
Procedure Output( Flag : Boolean); {описание процедуры }
 Begin                                 {вывода результатов }
  Write('О т в е т : в множестве точек ');
  If Flag then WriteLn('содержатся')
          else WriteLn('не содержатся');
  WriteLn(' точки, принадлежащие заданной области.');
  ReadLn
 End; {of Output}
{-----------------------------------------}
BEGIN
  Input;          {вызов процедуры ввода данных }
  Inside(Flag); {вызов процедуры проверки принадлежности}
  Output(Flag)  {вызов процедуры вывода результатов }
END.
```

Пример 4.3. ***Определить, имеется ли среди элементов главной диагонали заданной целочисленной матрицы A(N, N) хотя бы один положительный нечётный элемент.***

Система тестов

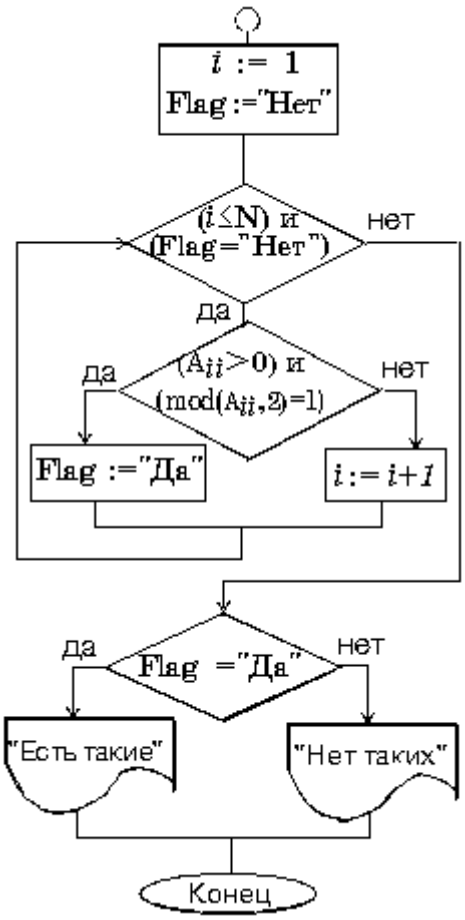
Номер теста	Проверяемый случай	Данные		Результат
		N	Матрица A	Текст
1	Имеется	3	$\begin{pmatrix} 2 & 2 & 2 \\ 2 & \mathbf{3} & 2 \\ 2 & 2 & 5 \end{pmatrix}$	"Есть такие"
2	Не имеется	2	$\begin{pmatrix} 2 & 1 \\ 1 & 2 \end{pmatrix}$	"Нет таких"

Школьный АЯ

```
алг Диагональ (арг цел N, арг цел таб A[1:N, 1:N], рез лит
Текст)
нач цел i, лит Flag
  i:=1; Flag:="Нет"
  нц пока (i<=N) и (Flag="Нет")    | условие продолжения цикла
```

```
    если (A[i, i]>0) и (mod(A[i, i], 2)=1) | условие завершения
цикла
      то Flag := "Да"
      иначе i:=i+1
    все
  кц
  если Flag = "Да"
    то Текст := "Есть такие"
    иначе Текст := "Нет таких"
  все
кон
```

Блок-схема (фрагмент)



Исполнение алгоритма

Обозначения проверяемых условий:
`(i <= N) и (Flag = "Нет")` => (1)
`(A[i, i] > 0) и (mod(A[i, i], 2) = 1)` => (2)

N теста	i	Flag	(1)	(2)	Текст
1	1 2	"Нет" "Да"	+ + - (кц)	- +	"Есть такие"
2	1 2 3	"Нет"	+ + - (кц)	- -	"Нет таких"

Turbo Pascal

```
Program Diagonal;
  Uses Crt;
  Type Mas = Array [1..10, 1..10] of Integer;
  Var  A          : Mas;
       N, i, j    : Integer;
       Flag       : Boolean;
{-----------------------------------}
Procedure InputOutput(Var A : Mas);    {описание процедуры ввода-
}
 Begin                                 {вывода исходных данных
}
   ClrScr;
   Write('Количество строк и столбцов - '); Read(N);
   For i := 1 to N do
```

```
      For j := 1 to N do
        begin Write('A[' , i , ', ' , j , '] = ? ');
              ReadLn(A[i, j])
        end;  WriteLn;
    WriteLn('Заданная матрица :');
    For i := 1 to N do
      begin
        For j := 1 to N do Write(A[i, j] : 5);
        WriteLn
      end; WriteLn
 End;    { of InputOutput }
{-----------------------------------}
Procedure Solution(Var A : Mas); {описание процедуры поиска
решения}
  Var Flag : Boolean;
 Begin
   Flag:=FALSE; i:=1;
   While (i<=N) and not Flag do
     If (A[i, i]>0) and (A[i, i] mod 2 = 1)
        then Flag:=TRUE
        else i:=i+1;
   WriteLn('О т в е т :');
   Write('Среди элементов главной диагонали ');
   If Flag then WriteLn ('есть нечетные положительные.')
           else WriteLn('нет нечетных положительных.');
   ReadLn;
  End;     { of Solution }
{-----------------------------------}
BEGIN
  InputOutput(A); {вызов процедуры ввода-вывода данных}
  Solution(A);    {вызов процедуры поиска решения задачи}
END.
```

Пример 4.4. ***Числа Фибоначчи (F_i) определяются по формулам $F_0 = F_1 = 1$; $F_i = F_{i-1} + F_{i-2}$ при i = 2, 3, ... (каждое очередное число равно сумме двух предыдущих). Вычислить сумму всех чисел Фибоначчи, которые не превосходят заданного натурального числа M.***

Тест

Номер теста	Данные	Результат
1	M=10	S=1+1+2+3+5+8=20
2	M=1	S=1+1=2

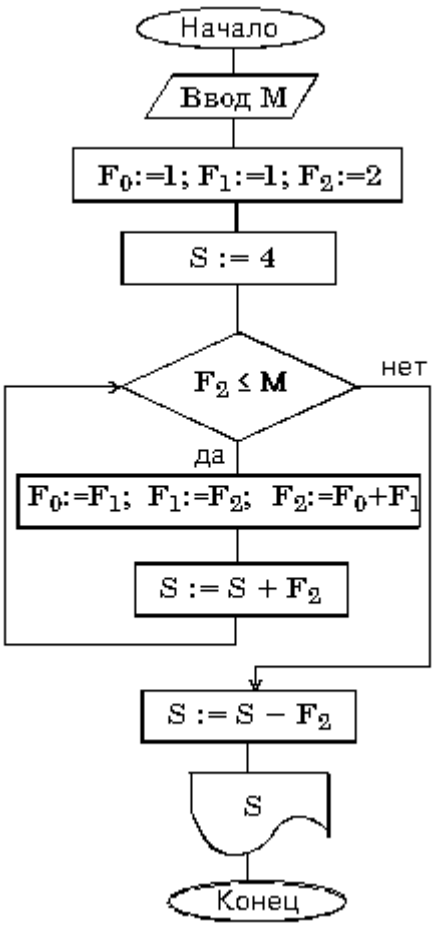
Демонстрация

Школьный АЯ

```
алг Фибоначчи (арг цел M, рез цел S)
  дано       | M>0
нач цел F0, F1, F2
```

```
  F0:=1; F1:=1; F2:=2
  S:=4    | 4 – сумма первых трех чисел Фибоначчи
  нц пока F2<=M
    F0:=F1; F1:=F2; F2:=F0+F1   | серия переприсваиваний
    S:=S+F2;
  кц
  S:=S-F2    | из S вычитается последнее значение F2,
превосходящее M
кон
```

Блок-схема



Исполнение алгоритма

F0	F1	F2	S	F2<M
1 1 2 3 5	1 2 3 5 8	2 3 5 8 13	4 4+3=7 7+5=12 12+8=20 20+13=33	+ + + + -(кц)
			33-13=20	

Turbo Pascal

```
Program SummaFib;
  Uses Crt;
  Var M,                    {заданное число }
      F0, F1, F2,           {три последовательных числа Фибоначчи}
      S : Integer;          {сумма чисел Фибоначчи}
BEGIN
  ClrScr;
  Write('Введите натуральное M : ');
  ReadLn(M);
  F0:=1; F1:=1; F2:=2;
  S:=4;              {4 – сумма первых трех чисел Фибоначчи}
  Write('Числа Фибоначчи, не превосходящие ', M, ' :', F0:4,
F1:4);
  While F2<=M do
    begin
      F0:=F1; F1:=F2; Write(F1 : 4);
```

```
      F2:=F0+F1; S:=S+F2;
    end;
  S:=S-F2; {вычитание из суммы последнего числа, которое
превосходит M}
  WriteLn; WriteLn;
  WriteLn('О т в е т : Сумма этих чисел равна ', S); ReadLn
END.
```

Результаты работы Pascal-программы

```
Введите натуральное M>0 :  10  <Enter>
Числа Фибоначчи, не превосходящие 10 :  1  1  2  3
5  8
О т в е т : Сумма этих чисел равна 20
```

Пример 4.5. ***Включить заданное число D в массив A(N), упорядоченный по возрастанию, с сохранением упорядоченности.***

Система тестов

Номер теста	Проверяемый случай	Данные		Результат
		D	Массив A	
1	D <= a_1	0	A=(1, 3, 5)	A=(0, 1, 3, 5)
2	a_1< D <= a_N	4	A=(1, 3, 5)	A=(1, 3, 4, 5)
3	a_N < D	6	A=(1, 3, 5)	A=(1, 3, 5, 6)

Школьный АЯ

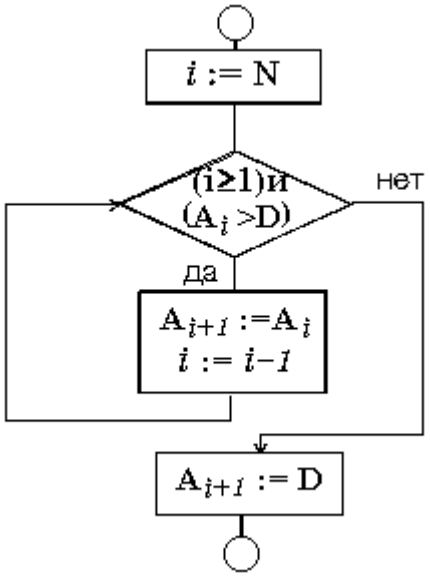
```
алг Включение (арг цел N, арг вещ D, арг рез вещ таб A[1:N+1])
  дано | A – упорядоченная по возрастанию последовательность
  надо | в A включено число D с сохранением упорядоченности
нач цел i
  i:=N
  нц пока (i>=1) и (A[i]>D)
    A[i+1] := A[i] | сдвиг очередного элемента вправо на одну
позицию
    i := i-1
  кц
  A[i+1] := D       | включение числа D в последовательность
кон
```

Исполнение алгоритма

Обозначение проверяемого условия:
(i >= 1) и (A[i] > D) => (1)

Блок-схема *(фрагмент)*

Номер теста	i	(1)	Массив A
1	3 2 1	+ + + -(кц)	(1, 3, 5) (1, 3, 5, 5) (1, 3, 3, 5) (1, 1, 3, 5) (0, 1, 3, 5)
2	3 2	+ -(кц)	(1, 3, 5) (1, 3, 5, 5) (1, 3, 4, 5)
3	3	-(кц)	(1, 3, 5) (1, 3, 5, 6)



Turbo Pascal

```
Program Insertion;
Uses Crt;
Var A    : Array [1..20] of Real;
    D    : Real;
    N, i : Integer;
{-----------------------------------------}
Procedure InputOutput;  {описание процедуры ввода-вывода}
 Begin   ClrScr;
   Write('Количество элементов массива - ');       ReadLn(N);
   WriteLn('Введите элементы массива, упорядоченные по
возрастанию:');
   For i := 1 to N do
     begin   Write('A[' , i , '] = '); ReadLn(A[i])
     end;    WriteLn;
   Write('Введите число, которое требуется включить в массив:
');
   ReadLn(D);
   ClrScr;   Write('Исходный массив :');
   For i := 1 to N do Write(A[i] : 5 : 1);  WriteLn;
   WriteLn('Включаемый элемент - ', D : 5 : 1);
End;    { of InputOutput }
{-----------------------------------------}
Procedure Insert; {описание процедуры включения нового элемента}
Begin
  i:=N;
  While (i>=1) and (A[i]>D) do
    begin A[i+1] := A[i];     {сдвиг очередного элемента вправо}
          i:=i-1
    end;
  A[i+1] := D {включение числа D в последовательность}
End;
{-----------------------------------------}
Procedure Result; {описание процедуры вывода результатов}
Begin   WriteLn;
  Write('О т в е т : массив с включенным элементом ');
```

```
  For i := 1 to N+1 do Write( A[i] : 5 : 1);     WriteLn;
  ReadLn
End;
{-----------------------------------------}
BEGIN
  InputOutput; {вызов процедуры ввода-вывода }
  Insert;      {вызов процедуры включения нового элемента}
  Result;      {вызов процедуры вывода результатов }
END.
```

Задачи для самостоятельного решения

4.1. Вычислите сумму $Z = 1 + 2 + 3 + \ldots$. Вычисления прекратите, когда значение Z превысит заданное значение A.

4.2. Проверьте, есть ли в заданной целочисленной последовательности a_1 , a_2 , ... , a_N элементы, равные нулю. Если есть, найдите номер первого из них, если нет - выдайте соответствующий текст.

4.3. Для заданного числа x вычислите первое из чисел последовательности *sin x, sin sin x, sin sin sin x,* ..., меньшее по модулю 10^{-2}.

4.4. Выясните, имеются ли в заданном векторе *A(N)* два подряд идущих нулевых элемента.

4.5. Выясните, имеются ли в заданном целочисленном векторе *A(N)* три подряд идущих элемента одного знака.

4.6. Множество точек в пространстве задано своими целочисленными координатами. Определите, совпадает ли хотя бы одна из точек с началом координат.

4.7. Если у заданного вектора *A(N)* есть хотя бы один элемент, меньший, чем -5, то все отрицательные элементы замените их квадратами, оставив остальные элементы без изменения; в противном случае вектор домножьте на 0,1 .

4.8. Имеется последовательность чисел a_1 , a_2 , ... , a_N. Найдите сумму первых из них (считая слева направо), произведение которых не превышает заданного числа M.

4.9. Задано целое $A>1$. Найдите наименьшее целое неотрицательное k, при котором $5^k > A$.

4.10. Все элементы заданного вектора *A(N)*, начиная с первого по порядку положительного элемента, уменьшите на единицу.

4.11. Числа Фибоначчи (F_i) определяются по формулам
$F_0 = F_1 = 1$; $F_i = F_{i-1} + F_{i-2}$ при $i = 2, 3, \ldots$
Найдите первое из чисел Фибоначчи, которое превосходит заданное число M ($M>0$).

4.12. Выясните, имеется ли среди чисел $i^3 - 17 \le i \le n^2 + n^3$, $i= 1, \ldots, n$, хотя бы одно число, которое кратно заданному числу A и не кратно заданному числу B (A<>B). При существовании такого числа вычислите сумму всех тех элементов, которые предшествовали ему.

4.13. Определите, имеются ли среди элементов побочной диагонали заданной целочисленной матрицы *A(N, N)* числа, равные нулю.

4.14. Найдите любое трёхзначные число, кратное заданному *P* и не равное ему.

4.15. Вычислите приближённое значение бесконечной суммы:

$$1-\frac{x^2}{2!}+\frac{x^4}{4!}-\frac{x^6}{6!}+\ldots+(-1)^{n-1}\frac{x^{2n-2}}{(2n-2)!}+\ldots$$

Суммирование производите до тех пор, пока очередное слагаемое не станет по абсолютной величине меньше заданного числа e>0.

4.16. Если в заданном целочисленном векторе *A(N)* есть элементы со значением, равным заданному числу *B*, то переменной *C* присвойте значение, равное сумме всех элементов, предшествующих первому по порядку такому элементу; в противном случае вывести соответствующий текст.

4.17. Дана последовательность из *N* целых чисел. Определите, со скольких положительных чисел она начинается.

4.18. Среди чисел

$$\cos(i^3)\cdot\sin(100+\frac{i}{100}),\quad i=0,1,\ldots,N$$

найдите номер и значение первого по порядку числа, абсолютная величина которого меньше заданного eps > 0. Если таких чисел нет, выведите на печать соответствуюший текст.

4.19. Определите, имеется ли в заданном массиве *A(N)* хотя бы одна пара соседних чисел, являющихся взаимнообратными.

4.20. Определите, выполняются ли для заданного вектора *A(2N)* условия:

$a_1 = a_{2N},\ \ a_2 = a_{2N-1},\ \ldots,\ \ a_N = a_{N+1},$

т.е. является ли он симметричным относительно своей середины.

4.21. Имеется монотонно убывающая последовательность чисел $a_1, a_2, \ldots, a_N$. Определите квадрат суммы положительных членов этой последовательности.

4.22. Если в заданном целочисленном векторе *A(N)* есть элементы со значением, равным заданному числу *B*, то переменной *C* присвойте значение, равное произведению всех элементов, следующих за первым по порядку таким элементом; в противном случае выведите соответствующий текст.

4.23. Определите, имеется ли в заданном целочисленном массиве *X(N)* число, кратное заданным числам *A* и *B*, и не кратное числу *C*.

4.24. Дано натуральное *N*. Выясните, сколько цифр оно содержит.

4.25. Найдите сумму цифр заданного натурального числа.

4.26. Цифры заданного натурального числа запишите в обратном порядке.

4.27. Проверьте, все ли элементы заданного массива *A(N)* положительны.

4.28. Найдите наименьший делитель заданного натурального числа *A* (не считая единицы).

4.29. Определите, является ли заданное натуральное число палиндромом (палиндром - число, одинаково читаемое слева направо и справа налево).

4.30. Определите по данным музейного каталога, есть ли в музее хотя бы одна картина Левитана или Шишкина. Если есть, выдайте ее название, в противном случае выдайте соответствующий текст.

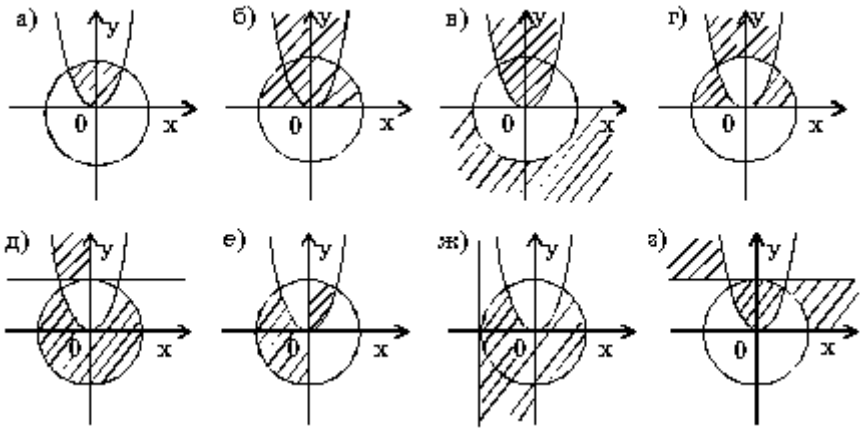
4.31. Определите по прейскуранту, можно ли подобрать в спортивном магазине велосипед, стоимость которого не превышает имеющуюся у покупателя сумму.

4.32. Известен начальный вклад клиента в банк и процент годового дохода. Определите, через сколько лет вклад превысит заданный размер и каков при этом будет размер вклада.

4.33. Торговая фирма в первый день работы реализовала товаров на *P* тыс. руб., а затем ежедневно увеличивала выручку на 3%. Какой будет выручка фирмы в тот день, когда она впервые превысит заданное значение *Q* ? Сколько дней придется торговать фирме для достижения этого результата?

4.34. Малое предприятие в первый день работы выпустило *P* единиц товарной продукции. Каждый последующий день оно выпускало продукции на *Q* единиц больше, чем в предыдущий. Сколько дней потребуется предприятию, чтобы общее количество выпущенной продукции за все время работы впервые превысило запланированный объем?

4.35. Определите, имеется ли в заданном множестве точек на плоскости хотя бы одна, принадлежащая заштрихованной на рисунке области (на рисунках даны окружности с единичным радиусом и парабола $y=x^2$):



4.36. Даны два натуральных числа *M* и *N* - числитель и знаменатель дроби *M*/*N*. Требуется сократить дробь, насколько это возможно.

4.37. На плоскости даны две точки *A(2, 2)* и *B(2, 6)*, а также *N* точек со своими координатами. Определите, есть ли среди этих *N* точек хотя бы одна, которая является:

а) вершиной равнобедренного треугольника с основанием AB;

б) вершиной прямоугольного треугольника с катетом AB.

4.38. Дано натуральное число *N*. Получите его запись в двоичной, восьмеричной и шестнадцатеричной системах счисления.

4.39. На соревнованиях по воздухоплаванию доля тепловых шаров от общего количества шаров обычно составляет от 93,4% до 97,5% всех шаров. При каком наименьшем общем количестве шаров возможно такое процентное соотношение?

4.40*.По перечню редких и исчезающих видов животных и растений, содержащемуся в "Красной Книге" вашего региона, определите, верно ли, что в нем содержится не менее пяти подвидов папоротников и ни более двух подвидов фазанов.

4.41*.Городок состоит из N многоквартирных котеджей, расположенных вдоль прямой дороги с одной ее стороны на равных расстояниях друг от друга. В городке проводят телефонную связь. Известно, сколько телефонных аппаратов надо установить в каждом доме. Определите, в каком из домов надо установить АТС, чтобы суммарное расстояние от АТС до всех телефонных аппаратов было минимальным. Если таких домов несколько, достаточно найти любой из них. Учтите, что каждый телефон связан с АТС отдельным проводом.

4.42*. Вокруг считающего стоят N человек, один из которых назван первым, а остальные занумерованы по часовой стрелке числами от 2 до *N*. Считающий ведет счет до M, начиная с первого. Человек, на котором остановился счет, выходит из круга. Счет возобновляется с человека, следовавшего за выбывшим, и так до тех пор, пока не останется один человек. Определите первоначальный номер последнего оставшегося человека.
&nspb;

Алгоритмы, реализуемые с помощью вложенных циклов типа ПОКА

Схема вложенных циклов типа пока:

```
нц пока  <условие 1>
  тело внешнего цикла
  . . . . . . .
      нц пока  <условие 2>
        тело внутренного цикла
        . . . . . .
      кц
  . . . . . . .
кц
```

Пример 5.1. Определить, имеется ли в заданном целочисленном массиве A(N) хотя бы одна пара совпадающих по значению чисел.

Система тестов

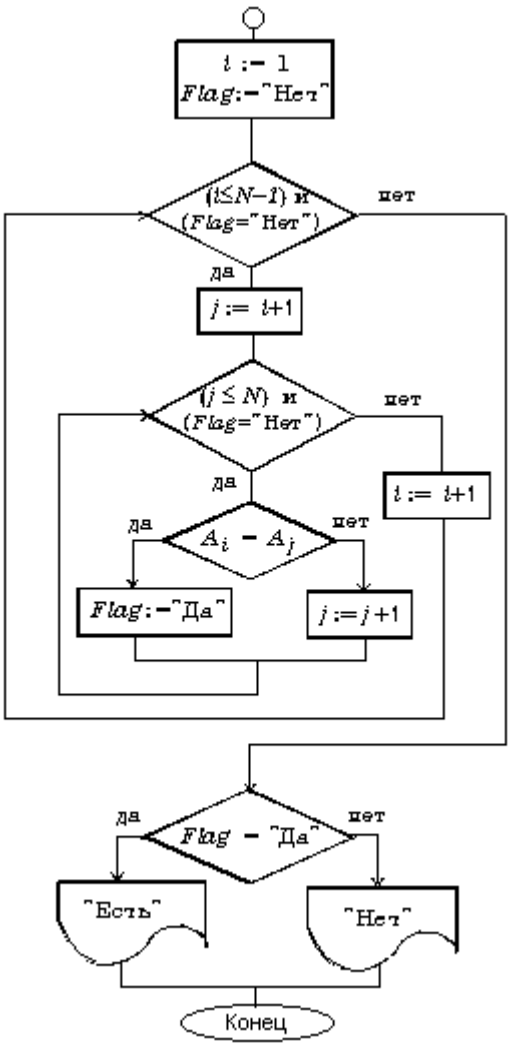
Номер теста	Проверяемый случай	Данные		Результат
		N	Массив A	Otvet
1 2	Имеется Не имеется	4 3	(1,3,2,3) (1,2,3)	"Есть совпадающие числа" "Нет совпадающих чисел"

Школьный АЯ

```
алг Равенство(арг цел N, арг цел
таб A[1:N],
                          рез лит
Otvet)
нач цел i, j, лит Flag
  i:=1;  Flag:="Нет"
  нц пока (i<=N-1) и (Flag="Нет")
      | цикл по первому числу из
пары
      j:=i+1
      нц пока (j<=N) и (Flag="Нет")
         | цикл по второму числу из
пары
        если A[i]=A[j] | проверка
равенства чисел
          то Flag:="Да"
          иначе  j:=j+1
        все
      кц
      i:=i+1
  кц
  если Flag="Да"
```

Блок-схема (фрагмент)

```
      то Otvet:="Есть совпадающие
числа"
      иначе  Otvet:="Нет
совпадающих чисел"
  все
кон
```



Исполнение алгоритма

Обозначения проверяемых условий:

(i <= N-1) и (Flag = "Нет") => (1)

(i <= N) и (Flag = "Нет") => (2)

N теста	i	Flag	(1)	j	(2)	A[i]=A[j]	Otvet
1	1	"Нет"	+	2 3 4 5	+ + + -(кц)	- - -	
	2	"Да"	+	3 4	+ + -(кц)	- +	
	3		- (кц)				"Есть совп.числа"
2	1 2 3	"Нет"	+ + -(кц)	2 3 4 3 4	+ + -(кц) + -(кц)	- - -	"Нет

							совп.чисел"

Turbo Pascal

```
Program Equal;
  Uses Crt;
  Type Mas = Array [1..20] of Integer;
  Var A       : Mas;
      i, j, N : Integer;
      Flag    : Boolean;
{-----------------------------------------}
Procedure InputOutput; {Описание процедуры ввода-вывода данных}
 Begin ClrScr;
   Write('N = '); ReadLn(N);
   For i := 1 to N do
     begin Write('A[' , i , '] = ') ; ReadLn(A[i]) end;
   WriteLn; WriteLn('Массив A');
   For i := 1 to N do Write(A[i] : 4);
   WriteLn; WriteLn
 End;
{-----------------------------------------}
Procedure Search(Var A:Mas; Var Flag:Boolean); {Описание
процедуры}
 Begin                                          {поиска решения
  }
   i:=1; Flag:= FALSE;
   While (i<=N-1) and not Flag do   {цикл по первому числу из
пары}
    begin
      j:=i+1;
      While (j<=N) and not Flag do  {цикл по второму числу из
пары}
         If A[i]=A[j] then Flag:=TRUE else j:=j+1;
      i:=i+1
    end;
 End;
{-----------------------------------------}
BEGIN
  InputOutput;     {вызов процедуры ввода-вывода данных  }
  Search(A, Flag); {вызов процедуры поиска решения задачи}
  WriteLn( 'О т в е т : ');
  If Flag then WriteLn('Есть совпадающие числа.' )
          else WriteLn('Нет совпадающих чисел.');
  ReadLn
END.
```

Пример 5.2. Дана целочисленная матрица A(N, N). Определить, имеются ли среди её элементов, лежащих ниже главной диагонали, отрицательные числа.

Система тестов

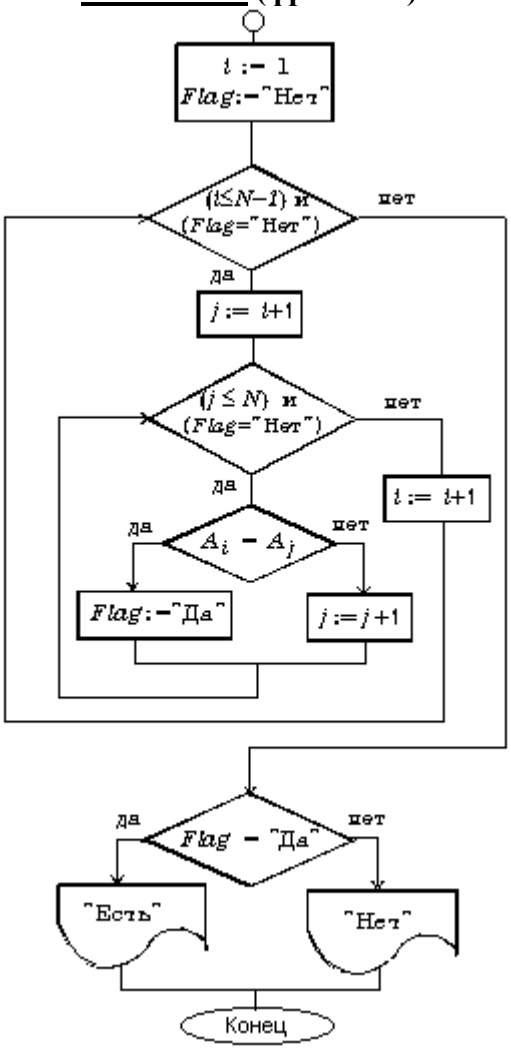
Номер	Проверяемый	Данные	Результат

теста	случай	N	Массив A	Otvet
1	Имеются	4	1 -1 2 1 **2 3 1 0** **1 -1** 2 -1 **-2 1 0** 1	"Есть отрицательные числа"
2	Не имеются	3	1 -1 2 **1** 0 1 **2 1** 1	"Нет отрицательных чисел"

Школьный АЯ

```
алг Ниже диагонали (арг цел N, арг
цел таб
                    A[1:N, 1:N], рез
лит Otvet)
нач цел i, j, лит Flag
  Flag:="Нет"; i:=2
  нц пока (i< =N) и (Flag="Нет") |
цикл по строкам
    j:=1
    нц пока (j<i) и (Flag="Нет")
              | цикл по элементам
строки
      если A[i, j]<0 | условие
прерывания циклов
        то Flag:="Да"
        иначе  j:=j+1 |
продвижение по строке
      все
    кц
    i:=i+1 | переход на новую
строку
  кц
 если Flag="Да"
  то Otvet:="Есть отрицательные
ниже диагонали"
  иначе Otvet:="Нет отрицательных
ниже диагонали"
 все
кон
```

Блок-схема (фрагмент)



Исполнение алгоритма

Обозначения проверяемых условий:

(i <= N) и (Flag = "Нет") => (1)
(j < i) и (Flag = "Нет") => (2)

N теста	i	Flag	(1)	j	(2)	A[i]=A[j]	Otvet
1	2	"Нет"	+	1 2	+ - (кц)	-	

	3	"Да"	+	1 2 3	+ + - (кц)	- +	
	4		- (кц)				"Есть отрицательные"
2	2	"Нет"	+	1 2	+ - (кц)	-	
	3		+	1 2 3	+ + - (кц)	- -	
	4		-(кц)				"Нет отрицательных"

Turbo Pascal

```
Program UnderDiagonal;
  Uses Crt;
  Type Mas = Array [1..10, 1..10] of Integer;
  Var A       : Mas;
      N, i, j : Integer;
      Flag    : Boolean;
{-----------------------------------}
Procedure InputOutput(Var A : Mas);  {описание процедуры }
 Begin                               {ввода-вывода данных}
  ClrScr;
  Write('Количество строк и столбцов - '); ReadLn(N);
  For i := 1 to N do
    For j := 1 to N do
      begin Write('A[' , i , ', ' , j , ']= ? ');
            ReadLn(A[i, j])
      end; WriteLn;
  WriteLn('Матрица :');
  For i := 1 to N do
   begin
     For j := 1 to N do Write(A[i, j] : 5);
     WriteLn
   end; WriteLn
End; { of InputOutput }
{-----------------------------------}
Procedure Solution(Var A : Mas); {описание процедуры поиска
решения}
 Begin
  i := 2 ; Flag := FALSE;
  While (i<=N) and not Flag do
    begin
      j:=1;
      While (j<i) and not Flag do
        If (A[i, j]<0)
```

```
            then Flag:=TRUE
            else j:=j+1;
        i:=i+1
      end;
End; { of Solution }
{-----------------------------------------}
Procedure OutResult;
Begin
  WriteLn('O т в е т :');
  Write('Среди элементов, лежащих ниже главной диагонали, ');
  If Flag then WriteLn('есть отрицательные.')
          else WriteLn('нет отрицательных.');
  ReadLn
End; { of OutResult }
{-----------------------------------------}
BEGIN
  InputOutput(A); {вызов процедуры ввода-вывода данных }
  Solution(A); {вызов процедуры поиска решения задачи}
  OutResult {вызов процедуры вывода результата }
END.
```

Пример 5.3. **Выяснить, есть ли в баскетбольных командах "Спартак" и "Зенит" игроки одинакового роста.**

Система тестов

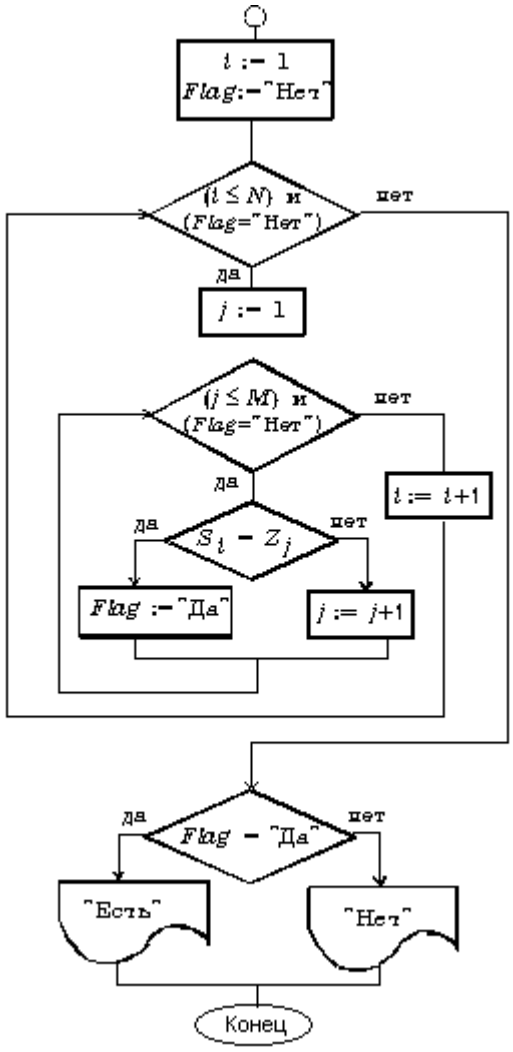
Обозначения:
N - количество игроков в команде "Спартак";
M - количество игроков в команде "Зенит";
S(N) - массив ростов игроков команды "Спартак" (см);
Z(N) - массив ростов игроков команды "Зенит" (см).

Номер теста	Проверяемый случай	Данные				Результат
		Спартак		Зенит		Otvet
		N	S(N)	M	Z(M)	
1	Есть	3	200 195 205	4	198 200 206 192	"Есть игроки одинакового роста"
2	Нет	2	200 195	2	198 201	"Нет игроков одинакового роста"

Школьный АЯ **Блок-схема (фрагмент)**

```
алг Рост (арг цел N, M, арг целтаб
S[1:N],
          Z[1:M], резлит Otvet)
```

```
нач цел i, j, лит Flag
  i:=1; Flag:="Нет"
  нц пока (i<=N) и (Flag="Нет")
          |цикл по игрокам
"Спартака"
     j:=1
     нц пока (j<=M) и (Flag="Нет")
          |цикл по игрокам "Зенита"
       если S[i]=Z[j] |проверка
равенства ростов
         то Flag:="Да"
         иначе  j:=j+1
       все
     кц
     i:=i+1
  кц
  если Flag="Да"
    то Otvet:="Есть игроки
одинакового роста"
     иначе Otvet:="Нет игроков
одинакового роста"
  все
кон
```



Исполнение алгоритма

Обозначения проверяемых условий:

(i <= N) и (Flag = "Нет") => (1)

(j < i) и (Flag = "Нет") => (2)

Номер теста	i	Flag	(1)	j	(2)	S[i]=Z[j]	Otvet
1	1	"Нет" "Да"	+	1 2	+ - -(кц)	- +	
	2		-(кц)				"Есть"
2	1	"Нет"	+	1 2 3	+ + -(кц)	- -	
	2		+	1 2 3	+ + -(кц)	- -	
	3		-(кц)				"Нет"

Turbo Pascal

```
Program EqualHeight;
  Uses Crt;
  Type Mas = Array [1..20] of Integer;
  Var
    Spart, Zenit : Mas;      {массивы ростов игроков}
    N, M, i, j   : Integer; {N - к-во игроков "Спартака", M -
"Зенита"}
    Flag         : Boolean;
    Name         : String;  {название команды}
{----------------------------------------}
Procedure Input(NCommand : Integer; Var Number : Integer; Var
Rost:Mas);
                {NCommand - номер команды (равен 1 или 2)}
 Begin {описание процедуры ввода данных по команде}
    If NCommand=1 then Name:='Спартак' else Name:='Зенит';
    Write('Введите количество игроков команды ', Name, ': ');
    ReadLn(Number);
    WriteLn('Введите роста игроков:');
    For i := 1 to Number do
      begin Write(i, ' игрок - '); ReadLn(Rost[i]) end;
    WriteLn
 End;
{----------------------------------------}
Procedure Search; {описание процедуры поиска решения задачи}
 Begin
   i:=1; Flag:=FALSE;
   While (i<=N) and not Flag do {цикл по игрокам Спартака}
     begin
       j:=1;
       While (j<=M) and not Flag do {цикл по игрокам Зенита}
         If Spart[i]=Zenit[j] then Flag:=TRUE else j:=j+1;
       i:=i+1
     end;
 End;
{-----------------------------------------------------------}
Procedure OutResult; {описание процедуры вывода результата}
  Begin
    Write('О т в е т : в командах Спартак и Зенит ');
    If Flag then Write('есть игроки ') else Write('нет игроков
');
    WriteLn('одинакового роста.');
    ReadLn
  End;
{----------------------------------------}
BEGIN ClrScr; {вызов процедур}
  Input(1, N, Spart); {ввод данных для первой команды}
  Input(2, M, Zenit); {ввод данных для второй команды}
  Search;             {поиск решения задачи}
  OutResult           {вывод результата}
END.
```

Пример 5.4. **Из партии шин отобрать две шины, диаметры которых отличаются не более, чем на D см, а вес — не более, чем на W грамм.**

Система тестов

N теста	Проверяемый случай	Данные					Результат
		N шины	Диаметр	Вес	Допуски		Otvet
					диам.	вес	
1	Есть такие шины	1 2 3 4	103 100 99 101	98 100 101 99	1	1	"2-я и 3-я шины"
2	Нет таких шин	1 2 3	100 98 100	100 100 98	1	1	"Подходящих шин нет"

Школьный АЯ

```
алг МоиШины (арг цел N, арг вещ таб Диам[1 : N] , Вес[1 : N] ,
              арг вещ ДопДиам, ДопВес, рез цел Шина1, Шина2,
              рез лит Otvet)
нач цел i, j, лит Flag
  i:=1; Flag:="Нет"
  нц пока (i< =N-1) и (Flag="Нет") | цикл по первой шине из
пары
    j:=i+1
    нц пока (j< =N) и (Flag="Нет") | цикл по второй шине из
пары
      если (abs(Диам[i] - Диам[j]) <= ДопДиам)   | условие сове-
          и (abs(Вес[i] - Вес[j]) <= ДопВес )    | таемости шин
        то Flag:="Да"; Шина1:=i; Шина2:=j
        иначе  j:=j+1
      все
    кц
    i:=i+1
  кц
  если Flag="Да"
    то Otvet := "По параметрам подходят друг другу "
              + Шина1 + "-ая и " + Шина2 + "-ая шины."
    иначе  Otvet := "Шин, подходящих друг другу, в партии нет."
  все
кон
```

Исполнение алгоритма

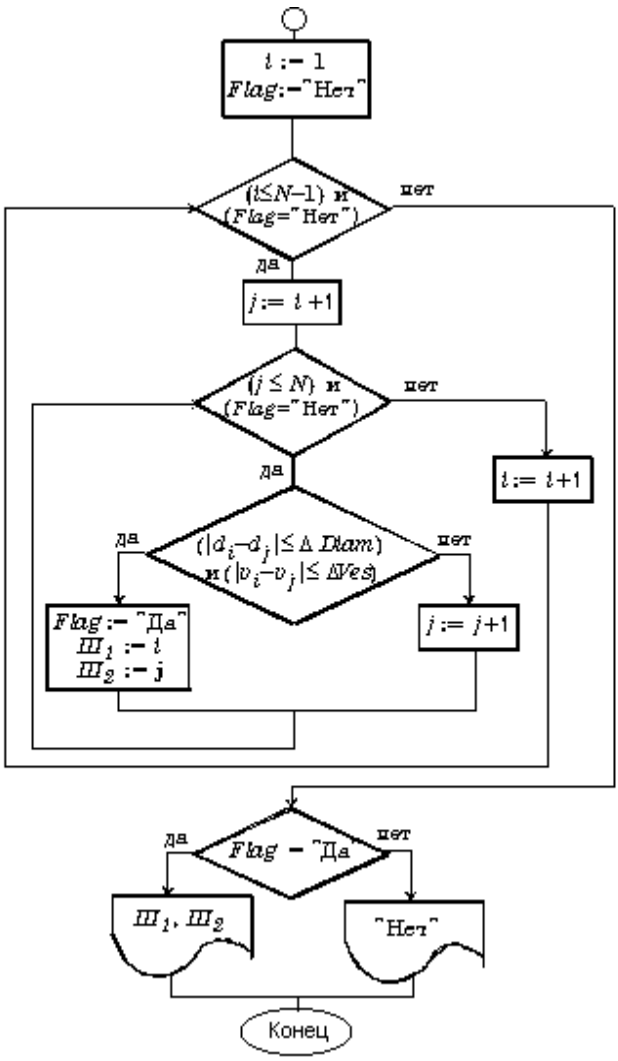
Обозначения проверяемых условий:

(i <= N-1) и (Flag = "Нет") => (1)
(i < N) и (Flag = "Нет") => (2)
(abs(Диам[i] - Диам[j]) <= ДопДиам)

Блок-схема (фрагмент)

и (abs(Вес[i] - Вес[j]) <= ДопВес) =>
(3)

N теста	i	Flag	(1)	j	(2)	(3)	Шина 1	Шина 2
1	1	"Нет"	+	2 3 4 5	+ + + - (кц)	- - -		
	2	"Да"	+	3	+ - (кц)	+	2	3
	3		- (кц)					
2	1	"Нет"	+	2 3 4	+ + - (кц)	- -		
	2		+	3 4	+ +	-		
	3		- (кц)					



Turbo Pascal

```
Program MyTyres;
  Uses Crt;
  Type Mas = Array [1..100] of Real;
  Var
    Number, i, j      : Integer; { Number - количество шин     }
    Diameter, Weight  : Mas;     { массивы параметров шин     }
    First, Second     : Integer; { номера отобранных шин      }
    Flag              : Boolean;
    D, W              : Real;    {D, W - допуски по параметрам}
{-------------------------------------------------------}
Procedure InputOutput; {описание процедуры ввода-вывода данных}
 Begin
  ClrScr;
  Write('Количество шин : '); ReadLn(Number);
  WriteLn('Параметры шин : ');
  For i := 1 to Number do
    begin
      Write(i, '-ая шина: Диаметр - '); ReadLn(Diameter[i]);
      Write('           Вес - ');  ReadLn(Weight[i])
    end; WriteLn;
  Write('Допуск по диаметру : '); ReadLn(D);
  Write('Допуск по весу : ');     ReadLn(W);
```

```
  WriteLn; WriteLn(' Параметры шин ');
  WriteLn('N шины   Диаметр   Вес');
  For i := 1 to Number do
    WriteLn(i:4, Diameter[i]:10:1, Weight[i]:10:1);
  WriteLn
End; { of InputOutput }
{-------------------------------------------------------------}
Procedure YesNo(Var First, Second : Integer; Var Flag :
Boolean);
 Begin {описание процедуры поиска решения задачи}
  i:=1;  Flag := FALSE;
  While (i<=Number-1) and not Flag do {цикл по первой шине из
пары}
    begin
      j := i+1;
      While (j<=Number) and not Flag do {цикл по второй шине из
пары}

        If (Abs(Diameter[i]-Diameter[j]) <= D)
            and (Abs(Weight[i]-Weight[j]) <= W)
          then begin Flag:=TRUE; First:=i; Second:=j end
          else j := j+1;
      i:=i+1
    end;
End; {of YesNo }
{-------------------------------------------------------------}
BEGIN
  InputOutput; {Вызов процедуры ввода-вывода исходных данных}
  YesNo(First, Second, Flag);{Вызов процедуры поиска решения
задачи}

  WriteLn('О т в е т :');
  If Flag then WriteLn('По параметрам подходят друг другу ',
                       First, '-ая и ', Second, '-ая шины.')
          else WriteLn('Шин, подходящих друг другу, в партии
нет.');
  ReadLn
END.
```

Задачи для самостоятельного решения

5.1. В заданной целочисленной матрице *A(N, M)* выведите на печать индексы первого положительного элемента, кратного заданному числу *K*. Если таких элементов в матрице нет, то выведите соответствующий текст. Элементы матриц просматривайте слева направо и сверху вниз.

5.2. В заданной целочисленной матрице *A(N, M)* замените первый отрицательный элемент максимальным элементом матрицы. Если отрицательных элементов нет, то выведите соответствующий текст. Элементы матриц просматривайте слева направо и сверху вниз.

5.3. Из заданной матрицы *A(N, N)* удалите строку, в которой находится первый отрицательный элемент. Элементы матриц просматривайте слева направо и сверху вниз.

5.4. В заданной матрице *A(N, N)* найдите индексы первого элемента, превосходящего среднее арифметическое всех элементов. Элементы матриц просматривайте слева направо и сверху вниз.
5.5. Из заданной матрицы *A(N, N)* удалите строку и столбец, в которых находится первый элемент, равный нулю. Полученную матрицу уплотните. Элементы матриц просматривайте слева направо и сверху вниз.
5.6. Если в заданной матрице *A(N, N)* есть хотя бы один элемент, больший ста, то элементы обеих диагоналей замените нулями.
5.7. Дана целочисленная матрица *A(N, N)*. Найдите номер первой из её строк, которые начинаются с К положительных чисел подряд.
5.8. Элементы заданной матрицы *A(N, N)* переписывайте построчно в одномерный массив до тех пор, пока не встретится нулевой элемент.
5.9. Заданное натуральное число *M* представьте в виде суммы квадратов двух неравных натуральных чисел. В случае, если это невозможно, выведите соответствующее сообщение.
5.10. Дана целочисленная матрица *A(N, N)*. Просматривая её элементы в заданном порядке, найдите первый чётный элемент и поменяйте его местами с диагональным элементом той строки, в которой он находится. Порядок просмотра:
а) сверху вниз и справа налево;
б) снизу вверх и слева направо;
в) справа налево и снизу вверх.
5.11. Проверьте, удовлетворяет ли заданная матрица *A(N, N)* следующему условию: для всех $i>1$ и для всех $j>1$ верно неравенство

$$a_{i\,j} >= a_{i-1,\,j} + a_{i,\,j-1} .$$

5.12. В заданном множестве точек на плоскости найдите пару точек, удалённых друг от друга на расстояние, большее заданного *D*.
5.13. Для заданной матрицы *A(N, N)* найдите хотя бы одно *k*, такое, что *k*-ая строка матрицы совпадает с *k*-м столбцом.
5.14. Даны три целочисленных массива *A(N)*, *B(M)* и *C(L)*. Найдите хотя бы одно число, встречающееся во всех трех массивах. Если таких чисел нет, выдайте соответствующее сообщение.
5.15*. Выберите три различные точки из заданного множества точек на плоскости так, чтобы внутри треугольника с вершинами в выбранных точках лежала ровно одна точка.
5.16. В школе имеется три параллельных десятых класса. Даны списки десятиклассников, содержащие фамилию и имя каждого ученика. Выясните:
а) в каких классах имеются однофамильцы;
б) в каких классах имеются тезки;
в) имеются ли в параллельных десятых классах однофамильцы;
г) в каких классах имеются ученики, у которых совпадают и имя и фамилия;
д) есть ли в десятых классах однофамильцы первого космонавта.
5.17. В детском саду есть *N* мячей. Имеются сведения о диаметре и цвете каждого мяча. Выясните:
а) есть ли среди мячей такой, что он не пройдет в квадратное окошко площадью 900 см^2;
б) есть ли мячи одинакового цвета или диаметра;
в) есть ли среди красных мячей такой, что его диаметр превосходит средний диаметр всех мячей.
5.18*. В заданном множестве точек на плоскости найдите три точки, которые могут служить вершинами остроугольного треугольника.
5.19*. В заданном множестве точек на плоскости найдите четыре точки, которые могут служить вершинами квадрата.
5.20*. В заданном множестве точек на плоскости найдите четыре точки, которые могут служить вершинами ромба.

5.21*. В заданном множестве точек на плоскости найдите четыре точки, которые могут служить вершинами выпуклого четырёхугольника.

5.22. Дана целочисленная квадратная матрица *A(N, N)*. Проверьте, являются ли все числа, расположенные выше главной и побочной диагоналей:

а) различными;

б) одинаковыми.

5.23. Дана целочисленная квадратная матрица *A(N, N)*. Определите, имеется ли среди элементов, расположенных ниже ее главной и побочной диагоналей хотя бы одно составное двузначное число.

5.24. На плоскости даны две точки *A(1, 1)* и *B(8, 1)*, а также *N* точек со своими координатами. Определите, есть ли среди этих *N* точек хотя бы одна пара точек, которые являлись бы вершинами трапеции с большим основанием AB.

5.25*. Дана квадратная таблица *A(N, N)*, элементами которой являются нули и единицы. Установите наличие в ней квадрата, строны которого состоят из *M* единиц *(M<=N)* и параллельны строкам или столбцам таблицы. Если такой квадрат найдется, то нужно вывести координаты его верхнего левого угла.

5.26*. Известно, что в пачке банкнот есть одна фальшивая банкнота и ее подлинник, серии и номера которых совпадают. Найдите эти банкноты в пачке, замените их двумя резервными банкнотами с известными сериями и номерами, и затем перенумеруйте всю пачку, расположив банкноты в соответствии с алфавитным порядком их серий, а банкноты с одинаковыми сериями — по возрастанию номеров.

5.27*.Имеется список учеников класса (все фамилии различны). Каждый ученик представил список одноклассников, у которых он был в гостях. Определите:

а) есть ли в классе ученик, который побывал в гостях у всех одноклассников, кроме одного;

б) есть ли в классе хотя бы одна пара учеников, которые не были в гостях друг у друга.

Схема циклов типа пока / для

```
нц пока <условие>
  тело внешнего цикла
    . . . . . .
  нц для i от A до B
    тело внутреннего цикла
  кц
    . . . . . .
кц
```

Схема циклов типа для / пока

```
нц для i от A до B
  тело внешнего цикла
    . . . . . .
  нц пока <условие>
    тело внутреннего цикла
  кц
    . . . . . .
кц
```

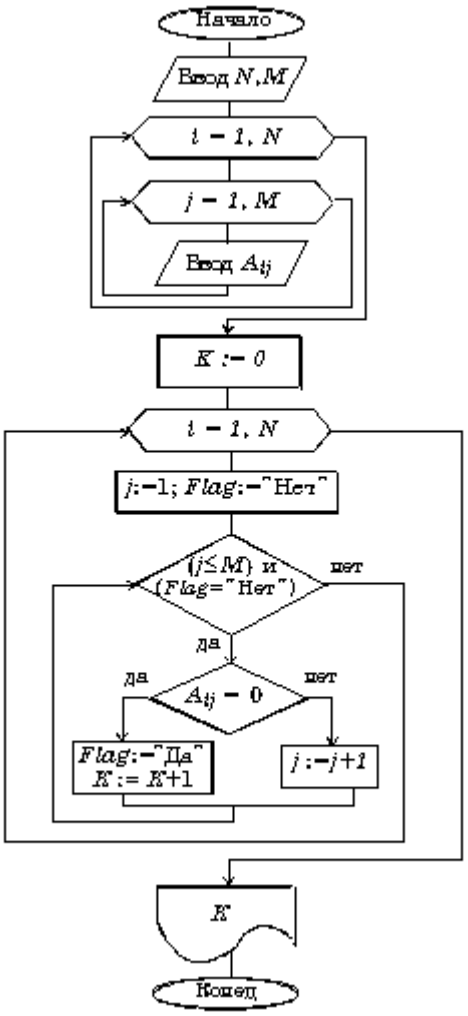
Пример 6.1. В заданной целочисленной матрице A(N, M) найти количество строк, содержащих нули.

Тест

Данные			Результат
N	M	Матрица A	K
3	3	$\begin{pmatrix} 1 & 0 & 1 \\ 2 & 2 & 1 \\ 0 & 1 & 0 \end{pmatrix}$	2

Школьный АЯ

```
алг Строки с нулями (арг цел N, M,
        арг цел таб A[1:N, 1:M], рез
цел K)
   дано | N>0, M>0
нач цел i, j, лит Flag
  K := 0
  нц для i от 1 до N | цикл по всем
строкам
    j:= 1; Flag := "Нет"
    нц пока (j <= M) и (Flag = "Нет")
           | цикл до нулевого элемента
строки
       если A[i, j] = 0
          то Flag:="Да"; K:=K+1
          иначе j:=j+1
       все
    кц
  кц
кон
```



Исполнение алгоритма

Обозначение проверяемого условия:

(j<=M) **и** (Flag = "Нет") => (1)

i	Flag	j	(1)	A[i,j]=0	K
1	"Нет" "Да"	1 2	+ + - (кц)	- +	0 1
2	"Нет"	1 2 3 4	+ + + - (кц)	- - -	
3	"Нет" "Да"	1	+ - (кц)	+	2

Turbo Pascal

```
Program ContainZero;
  Uses Crt;
  Var A          : Array[1..10, 1..10] of Integer;
```

```
      N, M, i, j : Integer;
      K          : Integer; {K - количество строк, содержащих
нули}
{-----------------------------------------}
Procedure InputOutput; {описание процедуры ввода-вывода данных}
Begin
  ClrScr;
  Write('Количество строк - '); ReadLn(N);
  Write('Количество столбцов - '); ReadLn(M);
  For i := 1 to N do
    For j := 1 to M do
      begin Write('A[' , i , ' , ' , j , ']= ? ');
            ReadLn(A[i, j])
      end;
  WriteLn;  WriteLn('Исходная матрица :');
  For i := 1 to N do
    begin
      For j := 1 to M do Write(A[i, j] : 5);
      WriteLn
    end; WriteLn
End; { of InputOutput }
{-----------------------------------------}
Function Zero(i:Integer):Boolean; {описание функции,
принимающей         }
   Var Flag : Boolean;            {значение Истина, если в
строке есть  }
Begin                             {нули, и Ложь, если в строке
нет нулей}
  j:=1; Flag:=FALSE;
  While (j<=M) and not Flag do
    If A[i, j]=0 then Flag:=TRUE else j:=j+1;
  Zero:=Flag                {значение функции присваивается имени
функции}
End;
{-----------------------------------------}
BEGIN
  InputOutput; {вызов процедуры ввода-вывода}
  K:=0;
  For i := 1 to N do
    If Zero(i) then K:=K+1;       {Zero(i) - указатель функции
Zero}
  WriteLn('Количество строк, содержащих нули, равно ', K);
ReadLn
END.
```

Пример 6.2. ***Дана целочисленная матрица A(N, M). Определить, встречается ли заданное целое K среди максимальных элементов столбцов этой матрицы.***

Система тестов

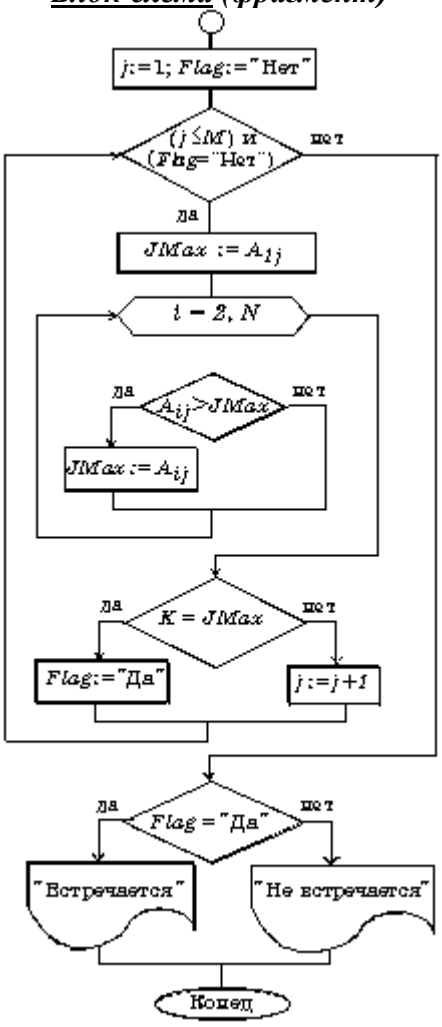
Номер	Проверяемый	Данные	Результат

теста	случай	K	N	M	Матрица A	Otvet
1	Встречается	5	3	3	$\begin{pmatrix} 1 & 5 & 8 \\ 4 & 2 & 0 \\ 1 & 2 & 3 \end{pmatrix}$	"Встречается"
2	Не встречается	1	2	2	$\begin{pmatrix} 2 & 1 \\ 1 & 2 \end{pmatrix}$	"Не встречается"

Школьный АЯ

```
алг Да или Нет(арг цел N,M,K, арг цел
таб
                A[1:N, 1:M], рез лит
Otvet)
   дано | N, M >0
нач цел i, j, JMax, лит Flag
  Flag:="Нет"; j:=1
  нц пока (j<=M) и (Flag="Нет")
          | цикл по столбцам с
прерыванием
    JMax:=A[1,j]
    нц для i от 2 до N | цикл нахождения
JMax
      если A[i,j]>JMax
        то JMax:=A[i, j]
      все
    кц
    если K=JMax | условие прерывания
цикла
      то Flag:="Да"
      иначе j:=j+1
    все
  кц
  если Flag="Да"
    то Otvet := "Встречается"
    иначе Otvet := "Не встречается"
  все
кон
```

Блок-схема (фрагмент)



Исполнение алгоритма

Обозначение проверяемого условия:

(j<=M) и (Flag = "Нет") => (1)

N теста	Flag	j	(1)	Jmax	I	A[i,j]>Jmax	K=Jmax

1	"Нет"	1	+	1 4	2 3	+ -	-
	"Да"	2	+ - (кц)	5	2 3	- -	+
2	"Нет"	1 2 3	+ + - (кц)	2 1 2	2 2	- +	- -

Turbo Pascal

```
Program Checking;
  Uses Crt;
  Var A           : Array[1..10, 1..10] of Integer;
      N, M, i, j  : Integer;
      K           : Integer; {заданное число}
      JMax        : Integer; {максимальный элемент столбца}
      Flag        : Boolean;
{-------------------------------------------------}
Procedure InputOutput; {описание процедуры ввода-вывода}
 Begin
  ClrScr;
  Write('Введите целое K = '); ReadLn(K); WriteLn;
  WriteLn('Введите целочисленную матрицу A');
  Write('Количество строк - '); ReadLn(N);
  Write('Количество столбцов - '); ReadLn(M);
  For i := 1 to N do
    For j := 1 to M do
      begin Write('A[' , i , ', ' , j , '] = ');
            ReadLn(A[i, j])
      end; ClrScr;
  WriteLn('Исходная матрица :'); WriteLn;
  For i := 1 to N do
    begin
      For j := 1 to M do Write(A[i, j] : 4);
      WriteLn
    end; WriteLn;
End; { of InputOutput }
{-----------------------------------------}
Procedure YesOrNot(Var Flag:Boolean); {описание процедуры
решения задачи}
 Begin
  Flag:=FALSE; j:=1;
  While (j<=M) and not Flag do {цикл по столбцам с прерыванием}
    begin JMax:=A[1, j];
      For i := 2 to N do {цикл нахождения JMax}
        If A[i, j]>JMax then JMax:=A[i, j];
      If K=JMax then Flag:=TRUE else j:=j+1 {условие
прерывания}
    end;
 End;
```

```
{-----------------------------------------}
BEGIN
  InputOutput;     {вызов процедуры ввода-вывода исходных
данных}
  YesOrNot(Flag); {вызов процедуры решения задачи}
  Write('О т в е т : число ', K );
  If Flag then Write(' встречается')
          else Write(' не встречается');
  WriteLn(' среди максимальных элементов столбцов матрицы');
  ReadLn
END.
```

Пример 6.3. ***Проверить, является ли заданная целочисленная матрица A(N, N) "магическим квадратом" (это значит, что суммы чисел во всех её строках, всех столбцах и двух диагоналях одинаковы).***

Система тестов

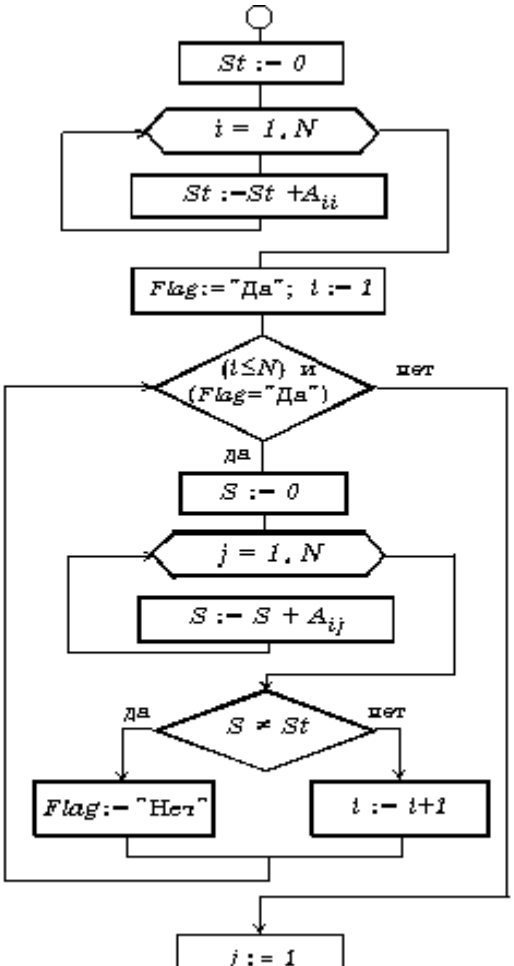
Номер теста	Проверяемый случай	Данные		Результат
		N	Матрица A	Otvet
1	Является	3	$\begin{pmatrix} 1 & 2 & 3 \\ 4 & 2 & 0 \\ 1 & 2 & 3 \end{pmatrix}$	"Магический квадрат"
2	Не является	2	$\begin{pmatrix} 2 & 1 \\ 1 & 2 \end{pmatrix}$	"Не магический квадрат "

Школьный АЯ

```
алг Магический квадрат (арг цел N, арг цел таб A[1:N, 1:N], рез
лит Otvet)
     дано | N>0
нач цел i, j, St, S, лит Flag
  St:=0                | вычисление суммы элементов главной
диагонали
  нц для i от 1 до N | в качестве эталонной суммы St
    St:=St+A[i, i]
  кц
  Flag:="Да"; i:=1
  нц пока (i<=N) и (Flag="Да") | вычисление сумм элементов
строк
     S:=0
     нц для j от 1 до N
       S:=S+A[i, j]
     кц
     если S<>St | сравнение суммы элементов текущей строки с
эталонной
       то Flag:="Нет"
       иначе i:=i+1
```

```
    все
  кц
  j:=1
  нц пока (j<=N) и (Flag="Да") | вычисление сумм элементов
столбцов
    S:=0
    нц для i от 1 до N
      S:=S+A[i, j]
    кц
    если S<>St                  |сравнение суммы элементов
текущего
      то Flag:="Нет"            |столбца с эталонной суммой
      иначе j:=j+1
    все
  кц
  если Flag="Да"
    то S:=0 | вычисление суммы элементов побочной диагонали
      нц для i от 1 до N
        S:=S+A[i, N+1-i]
      кц
      если S<>St | сравнение суммы с эталонной
        то Flag:="Нет"
      все
  все
  если Flag="Да"
    то Otvet := "Это магический квадрат."
    иначе Otvet := "Это не магический квадрат."
  все
кон
```

Блок-схема (фрагмент)



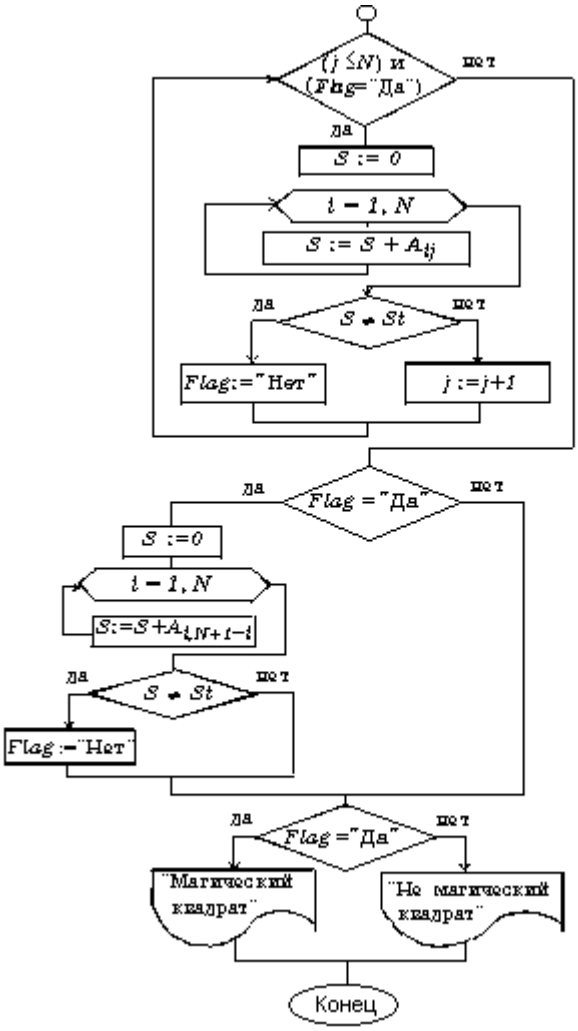
Вычисление суммы элементов главной диагонали в качестве эталонной суммы

Вычисление сумм элементов строк и сравнение их с эталонной суммой

Блок-схема (продолжение)

Вычисление сумм элементов столбцов и сравнение их с эталонной суммой

Вычисление суммы элементов побочной диагонали и сравнение ее с эталонной суммой



Исполнение алгоритма

(в таблице отражен только конечный результат работы циклов типа для, в которых вычисляются суммы)

Обозначения проверяемых условий:

(i<=N) и (Flag = "ДА")=> (1)

(j<=N) и (Flag = "ДА")=> (2)

Flag = "ДА" => (3)

N теста	St	Flag	i	(1)	j	(2)	(3)	S	S<>St	Otvet
	6	"Да"	1	+	1,2,3			6	-	

1			2 3 1,2,3 1,2,3 1,2,3 1,2,3	+ +	1,2,3 1,2,3 1 2 3	+ + +	+	6 6 6 6 6 6	- - - - - -	"Магический квадрат"
2	4	"Да" "Нет"	1	+ - (кц)	1,2 1	- (кц)	-	3	+	"Не магический квадрат"

Turbo Pascal

```
Program MagicSquare;
Uses Crt;
Var A : Array [1..20, 1..20] of Integer;
    i, j, N : Integer;
    Standard, S : Integer; {Standard - сумма-эталон, S - текущая
сумма}
    Flag : Boolean;
{--------------------------------------}
Procedure InputOutput; {описание процедуры ввода-вывода
матрицы}
Begin
  ClrScr;
  Write('Количество строк и столбцов - ');
  ReadLn(N);
  For i := 1 to N do
    For j := 1 to N do
      begin Write('A[' , i , ', ' , j , '] = ');
            ReadLn(A[i, j])
      end;
  ClrScr;
  WriteLn('Исходная матрица :'); WriteLn;
  For i := 1 to N do
    begin
      For j := 1 to N do Write(A[i, j] : 5);
      WriteLn
    end; WriteLn
End; { of InputOutput }
{-----------------------------------------}
Procedure MagicOrNot(Var Flag : Boolean);          {описание
процедуры, }
                  {в которой выясняется, является ли квадрат
"магическим"}
 Begin   {вычисление суммы элементов главной диагонали}
         {в качестве эталонной суммы}
  Standard:=0;
  For i := 1 to N do Standard := Standard + A[i,i];
  Flag:=TRUE; i:=1;
  While (i<=N) and Flag do {вычисление сумм элементов строк}
    begin
      S:=0;
      For j := 1 to N do S := S+A[i, j];
      If S<>Standard then Flag := FALSE else i:=i+1
```

```
    end;
  j:=1;
  While (j<=N) and Flag do {вычисление сумм элементов столбцов}
    begin
      S:=0;
      For i := 1 to N do S:=S+A[i, j];
      If S<>Standard then Flag := FALSE else j := j+1
    end;
  If Flag then
    begin
      S:=0; {вычисление суммы элементов побочной диагонали}
      For i := 1 to N do S := S+A[i, N+1-i];
      If S<>Standard then Flag := FALSE;
    end;
End;
{-------------------------------------------------------}
BEGIN
  InputOutput; {Вызов процедуры ввода-вывода }
  MagicOrNot(Flag); {Вызов процедуры решения задачи }
  If Flag then WriteLn('Это магический квадрат.')
          else WriteLn('Это не магический квадрат.');
  ReadLn
END.
```

Пример 6.4. **Дана матрица A(N, N). Если хотя бы один элемент строки матрицы отрицателен, то все элементы этой строки заменить нулями.**

Тест

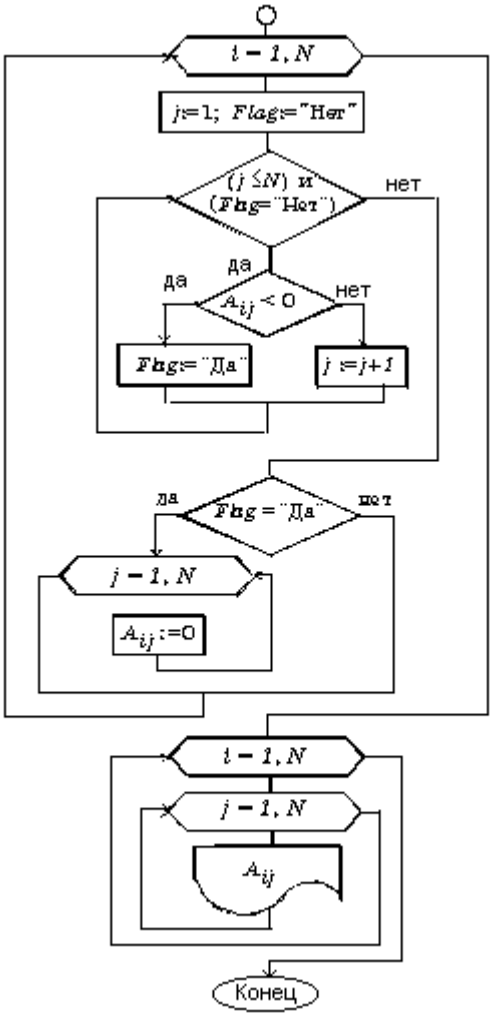
Данные		Результат
N	Матрица A	Матрица A
3	$\begin{pmatrix} 1 & -2 & 1 \\ 1 & 2 & 1 \\ -1 & 2 & -2 \end{pmatrix}$	$\begin{pmatrix} 0 & 0 & 0 \\ 1 & 2 & 1 \\ 0 & 0 & 0 \end{pmatrix}$

Школьный АЯ ***Блок-схема (фрагмент)***

(в этом алгоритме отражены процессы ввода исходных данных и вывода результатов)

```
алг Модификация(арг цел N, арг
рез
                вещ таб A[1:N,
1:N])
  дано | N>0
  надо | элементы строк,
содержащих отрица-
       | тельные числа,
заменены на нули
нач цел i, j, лит Flag
  ввод N
  нц для i от 1 до N
```

```
    нц для j от 1 до N
       ввод A[i,j]
    кц
  кц
  нц для i от 1 до N | цикл по
строкам
    j := 1; Flag := "Нет"
    нц пока (j<=N) и (Flag =
"Нет") |цикл до
       если A[i, j]<0
|первого отрицат.
         то Flag := "Да"
|элемента строки
         иначе j:=j+1
       все
    кц
    если Flag = "Да"
|обнуление строки
      то нц для j от 1 до N
          A[i, j]:=0
         кц
    все
  кц
  нц для i от 1 до N
    нц для j от 1 до N
       вывод A[i,j]
    кц
  кц
кон
```



Исполнение алгоритма

Обозначение проверяемого условия:
(j<=N) и (Flag = "Нет")=> (1)

i	Flag	j	(1)	A[i,j]<0	Flag="Да"	A[i,j]
1	"Нет" "Да"	1 2 1 2 3	+ + -(кц)	- +	+	 A[1,1]=0 A[1,2]=0 A[1,3]=0
2	"Нет"	1 2 3 4	+ + + -(кц)	- - -	-	
3	"Нет" "Да"	1 1 2 3	+ -(кц)	+	+	A[3,1]=0 A[3,2]=0 A[3,3]=0

Turbo Pascal

```
Program Modify;
 Uses Crt;
```

```
 Var A       : Array[1..10, 1..10] of Real;
     N, i, j : Integer;
{-------------------------------------}
Procedure InputOutput; {описание процедуры ввода-вывода матрицы}
 Begin ClrScr;
   Write('Количество строк и столбцов - '); ReadLn(N);
   For i := 1 to N do
     For j := 1 to N do
       begin Write('A[' , i , ', ' , j , '] = ');
             ReadLn(A[i, j])
       end; ClrScr;
   WriteLn(' Исходная матрица :'); WriteLn;
   For i := 1 to N do
     begin
       For j := 1 to N do Write(A[i, j] : 5 : 1);
       WriteLn
     end; WriteLn
 End; { of InputOutput }
{-----------------------------------------}
Procedure Line(Var i : Integer);          {описание процедуры
обработки}
 Var Flag : Boolean;                      {строки
матрицы              }
 Begin
   j := 1; Flag := FALSE;
   While (j<=N) and not Flag do            {цикл до первого
отрицательного}
     If A[i, j]<0 then Flag:=TRUE else j:=j+1;
{элемента строки}
   If Flag then                          {обнуление строки,
содержашей}
     For j := 1 to N do A[i, j] := 0 {отрицательные
элементы      }
End;
{-----------------------------------------}
Procedure OutRes; {описание процедуры вывода матрицы-результата}
 Begin
   WriteLn(' Матрица-результат :'); WriteLn;
   For i := 1 to N do
     begin
       For j := 1 to N do Write(A[i, j]:5:1);
       WriteLn
     end; ReadLn
 End; { of OutRes }
{-----------------------------------------}
BEGIN
  InputOutput; {вызов процедуры ввода-вывода матрицы}
  For i := 1 to N do Line(i);{циклический вызов процедуры
обработки строк}
  OutRes;       {вызов процедуры вывода матрицы-результата}
END.
```

Задачи для самостоятельного решения

6.1. Дана матрица *A(N, N)* . Переменной *B* присвойте значение, равное количеству строк матрицы *A*, содержащих хотя бы одну нулевую компоненту.

6.2. Дана матрица *B(N, N)* . Получите вектор *A(N)* , компоненты которого находятся по правилу: A_i равно первому по порядку положительному элементу в *i*-ой строке матрицы (если таких элементов в строке нет, то примите $A_i = -1$).

6.3. Дана матрица *B(N, N)*. Получите вектор *A(N)*, компоненты которого находятся по правилу: A_i равно количеству отрицательных чисел, с которых начинается *i*-ая строка.

6.4. Среди строк заданной целочисленной матрицы, содержащих только нечётные элементы, найдите строку с максимальной суммой модулей элементов.

6.5. Среди столбцов заданной целочисленной матрицы, содержащих только такие элементы, которые по модулю не больше 10, найдите столбец с минимальным произведением элементов.

6.6. Задано два множества точек на плоскости. В первом множестве найдите хотя бы одну точку, сумма расстояний от которой до точек второго множества превышала бы заданную величину.

6.7. В заданной матрице *A(N, M)* найдите количество строк, не содержащих отрицательных чисел.

6.8. Дана целочисленная матрица *A(N, N)* . Сформируйте результирующий вектор *B* , элементами которого являются суммы элементов только тех строк матрицы *A* , которые начинаются с *K* положительных чисел подряд.

6.9. Подсчитайте количество столбцов заданной целочисленной матрицы *A(N, N)* , в которых имеются взаимнопротивоположные соседние числа.

6.10. Дана матрица *A(N, M)* . Постройте вектор *B(N)* , элементы B_i которого равны единице, если элементы *i*-ой строки образуют упорядоченную по убыванию или по возрастанию последовательность, и нулю во всех остальных случаях.

6.11. Определите, сколько строк заданной матрицы *A(N, M)* содержат хотя бы один элемент из заданного числового диапазона.

6.12. Найдите номера строк заданной целочисленной матрицы *A(N, M)* , в которых:

а) на всех нечётных позициях стоят нули;

б) на нечетных позициях встречаются нули.

6.13. Найдите номера столбцов заданной целочисленной матрицы *A(N, M)*, которые составлены из попарно различных чисел, и подсчитайте количество таких столбцов.

6.14. Подсчитайте количество различных (не повторяющихся) чисел, встречающихся в заданном целочисленном массиве *A(N)*.

6.15. Подсчитайте количество различных (не повторяющихся) чисел, встречающихся в заданной целочисленной матрице *A(N, M)* .

6.16. Даны сведения о количестве забитых голов каждого футболиста команды в каждом из матчей чемпионата. Проверьте, сколько в команде есть футболистов:

а) забивших хотя бы два гола;

б) забивавших голы в каждом матче;

в) не забивших ни одного гола.

6.17. Используя сведения о ежемесячных выплатах зарплаты сотрудникам фирмы, выясните, не оказалась ли годовая зарплата кого-либо из сотрудников ниже годового минимума, оговоренного в его контракте.

6.18. Используя сведения о результатах сдачи *n* вступительных экзаменов *m* абитуриентами, определите, сколько абитуриентов сдали все экзамены на "отлично".

6.19. Используя сведения о размере обуви каждого члена баскетбольной команды, а также сведения о наличии в обувном магазине размеров спортивной обуви, определите, сколько членов команды можно обуть в этом магазине.

6.20*. Найдите максимальное из чисел, встречающихся в заданной матрице более одного раза.

6.21*. Подсчитайте количество строк заданной целочисленной матрицы $A(5,5)$, являющихся перестановкой чисел 1, 2, ..., 5.

6.22. Из массива $A(N)$, состоящего из натуральных чисел, получите массив $B(N)$, элементами которого являются разрядности элементов массива $A(N)$.

6.23. Задана последовательность N вещественных чисел. Определите, можно ли так переставить ее элементы, чтобы они образовали геометрическую прогрессию.

6.24. В заданном одномерном массиве удалите каждый третий положительный элемент.

6.25. В каждой строке матрицы $A(N, N)$ определите наибольшее простое число. Если в строке нет простых чисел, выдайте соответствующее сообщение.

6.26. Задана квадратная таблица $A(N, N)$, элементами которого являются нули и единицы. Подсчитайте в ней:

а) количество квадратов размером 3 на 3, в которых есть не менее пяти нулей;

б) количество строк, в которых есть три нуля, расположенных рядом;

в) количество столбцов, в которых нули и единицы чередуются;

г) количество единичных столбцов и количество нулевых строк.

6.27. Пифагоровыми называются тройки натуральных чисел a, b, c, удовлетворяющие условию: $a^2 + b^2 = c^2$. Например, пифагоровой является тройка чисел 6, 8, 10. Найдите все тройки пифагоровых чисел, не превышающих 25.

6.28. Совершенными называются числа, равные сумме своих делителей. Например, совершенным является число 28, равное 1 + 2 + 4 + 7 + 14. Найдите все совершенные числа в интервале [1, 1000].

Символьная информация — это информация, отображаемая с помощью символов (букв, цифр, знаков операций и др.).

IBM-совместимые компьютеры обрабатывают 256 различных символов, каждый из которых кодируется одним байтом. Соответствие символов и байтов задается ***таблицей кодировки***, в которой для каждого символа указывается соответствующий байт.
Символы с кодами от 0 до 127 построены по **стандарту ASCII** (**A**merican **S**tandard **C**ode for **I**nformation **I**nterchange — Американский стандартный код обмена информацией, читается "аски"). Вторая половина таблицы (коды 128 ... 255) в нашей стране содержит русские буквы (кириллицу) и символы псевдографики.

Коды 0...127 (кодировка ASCII)

	000	016	032	048	064	080	096	112	
00		◆		0	@	P	`	p	00
01		◆	!	1	A	Q	a	q	01
02		↕	"	2	B	R	b	r	02
03	♥	‖	#	3	C	S	c	s	03
04	◆	¶	$	4	D	T	d	t	04
05	♣	§	%	5	E	U	e	u	05
06	♠		&	6	F	V	f	v	06
07	•		'	7	G	W	g	w	07
08		↑	(	8	H	X	h	x	08
09	°	↓	)	9	I	Y	i	y	09
10		→	*	:	J	Z	j	z	10
11		←	+	;	K	[	k	{	11
12		⌙	,	<	L	\	l	|	12
13		•	-	=	M	]	m	}	13
14		•	.	>	N	^	n	~	14
15	Ц	◆	/	?	O	_	o	§	15

Коды 128...255 (модифицированный альтернативный вариант)

	128	144	160	176	192	208	224	240	
00	А	Р	а	░	└	╨	р	≡	00
01	Б	С	б	▒	┴	╤	с	±	01
02	В	Т	в	▓	┬	╒	т	≥	02
03	Г	У	г	|	├	╓	у	≤	03
04	Д	Ф	д	┤	─	╘	ф	⌠	04
05	Е	Х	е	╡	┼	╒	х	⌡	05
06	Ж	Ц	ж	╢	╞	╫	ц	÷	06
07	З	Ч	з	╖	╟	╪	ч	≈	07
08	И	Ш	и	╕	╚	╪	ш	°	08
09	Й	Щ	й	╣	╔	┘	щ	∙	09
10	К	Ъ	к	‖	╩	┌	ъ	·	10
11	Л	Ы	л	╗	╦	▉	ы	√	11
12	М	Ь	м	╝	╠	▄	ь	ⁿ	12
13	Н	Э	н	╜	¥	▌	э	²	13
14	О	Ю	о	╛	╬	▐	ю	■	14
15	П	Я	п	┐	╧	▀	я		15

Для того, чтобы определить по этим таблицам код того или иного символа, нужно **сложить номер строки с номером столбца**, в которых он расположен. Так, код цифры 5 равен 05+048 = 053.
Символьная информация в алгоритмах и программах описывается данными двух типов: ***символьным*** и ***литерным***. Они отличаются друг от друга тем, что **значением символьной переменной является один символ, а литерной — строка символов**.

Типы данных, используемые для обработки символьной информации

Язык	Тип, ключевое слово	Примеры использования
Школьный АЯ	Символьный **сим**	a := "f " ; b := "+" ; c := "5" If a = " " then k := k + 1
	Литерный **лит**	t := "Литерная величина" s := "" (пустая строка)
Turbo Pascal	Символьный **Char**	a := ' f ' ; b := '+' ; c := '5' ; If a = ' ' then k := k + 1
	Литерный **String**	t := 'Литерная величина' ; f := ' ' ; (пустая строка)

Для данных символьного и литерного типов применимы ***операции сцепки*** (соединения, конкатенации) и ***сравнения*** (<, >, <=, >=, =, <>). Сравнивать можно строки разной длины. Сравнение осуществляется слева направо в соответствии с ASCII-кодами соответствующих символов. Так, строка "*стол*" меньше строки "*стул*", строка "*teacher*" больше строки "*pupil*" , а строка "*пар*" меньше строки "*парад*".

Функции и команды обработки строк

Школьный АЯ

Функция **длин(S)** Возвращает количество символов в строке S.
Операция **вырезка** позволяет "вырезать" из строки группу соседних символов. Вырезка из строки S подстроки, начинающейся с i-ой и кончающейся j-ой позицией, обозначается S [i : j]. Вырезка из строки S одного i-го символа обозначается S[i].
Команда **присваивания вырезке S[N : M] := SubS**. Часть строки S, начиная с позиции N и кончая позицией M, заменяется на подстроку SubS такой же длины.

Turbo Pascal

Процедуры

Delete (Var S : String; N, M : Integer) Удаляет M символов из строки S, начиная с позиции N.
Insert (SubS : String; Var S : String; N : Integer) Вставляет подстроку SubS в строку S, начиная с позиции N.
Str (X : Integer; Var S : String) Возвращает представление числа X в его символьной форме S.
Val (S : String; Var X, Code : Integer) Возвращает представление символов строки S в ее числовой форме X. Параметр Code содержит признак ошибки преобразования (если Code = 0, ошибки нет).

Функции

Chr (X : Byte) : Char Возвращает символ с заданным порядковым номером X.
Concat (S_1 [, S_2 , ... , S_N]): String Выполняет сцепку (конкатенацию) последовательности строк.
Copy (S : String; N , M : Integer) : String Возвращает подстроку из строки S, начиная с позиции N и длиной M символов.
Length (S : String) : Byte Возвращает количество символов в строке S.

Ord (X : Char) : LongInt Возвращает порядковый номер символа X в таблице кодов символов.
Pos (SubS , S : String) : Byte Возвращает номер позиции, начиная с которой в строке S располагается подстрока SubS (если значение функции равно нулю, то S не содержит SubS).

Пример 7.1. ***Определить количество слов в заданном тексте.***
Если слова в тексте разделены **одним пробелом**, то задача сводится к подсчету числа пробелов. Количество слов при этом равно числу пробелов плюс 1. Если же **число пробелов** между соседними словами **произвольное**, как обычно и бывает, то алгоритм усложняется. Рассмотрим оба варианта решения этой задачи.

Вариант 1. Слова в тексте разделены одним пробелом.

Тест

Данные	Результат
"Кот на крыше"	**N=3**

Школьный АЯ

```
алг Число слов (арг лит Text,
рез цел N)
  дано | В непустом тексте
Text слова
       | разделены одним
пробелом
  надо | N — количество слов
нач цел i
  N:=1
  нц для i от 1 до длин(Text)
              | цикл по буквам
текста
    если Text[i] = " "
      то N:=N+1
    все
  кц
кон
```

Исполнение алгоритма

i	Text[i]	Text[i]=" "	N
1	к	+	1
2	о	-	
3	т	-	
4	_	-	2
5	н	+	
6	а	-	3
7	_	-	
8	к	+	
9	р	-	
10	ы	-	
11	ш	-	
12	е	-	

Turbo Pascal

```
Program Probel;
  Uses Crt;
  Var Text       : String;  {заданный непустой текст}
      i, Number  : Integer; {Number — количество слов в тексте}
      Letter     : Char;    {текущая буква }
BEGIN ClrScr;
  WriteLn('Введите текст :'); ReadLn(Text);
  Number:=1;
  For i:=1 to Length(Text) do {цикл по буквам текста}
   begin
     Letter:=Text[i];
     If (Letter = ' ' ) then Number:=Number+1;
   end;
  WriteLn('О т в е т : количество слов в тексте равно ',
Number);
END.
```

Вариант 2. Слова в тексте разделены произвольным количеством пробелов.

Тест

Данные	Результат
"Кот на крыше"	**N=3**

Школьный АЯ

```
алг Число слов (арг лит Text, рез цел N)
  дано | В тексте Text слова могут быть разделены
       | произвольным количеством пробелов
  надо | N — количество слов в тексте Text
нач цел i, лог Flag
  N:=0; Flag:=да
  нц для i от 1 до длин(Text)        | цикл по буквам текста
     если (Text[i]<>" ") и (Flag=да) | это условие выполняется
только
       то N:=N+1                     | для первой буквы каждого
слова
     все
     Flag := (Text[i]=" ")     | Flag=да, если очередная буква —
пробел,
  кц                           | в противном случае Flag = нет
кон
```

Исполнение алгоритма

Обозначение проверяемого условия:

(Text[i]<>" ") и (Flag = да)=>(1)

i	Text[i]	(1)	N	Flag
1	К	+	0	да
2	о	-	1	нет
3	т	-		нет
4	_	-		нет
5	н	+	2	нет
6	а	-		да
7	_	-		нет
8	_	-		нет
9	_	-	3	нет
10	к	+		нет
11	р	-		да
12	ы	-		нет
13	ш	-		нет
14	е	-		нет
				нет

Turbo Pascal

```
Program KolSlov;
  Uses Crt;
  Var Text       : String;  {заданный текст}
      i, Number  : Integer; {Number - количество слов в тексте}
      Flag       : Boolean;
      Letter     : Char;    {текущая буква }
BEGIN
  ClrScr;
  WriteLn('Введите текст :');
```

```
  ReadLn(Text);
  Number := 0; Flag := TRUE;
  For i := 1 to Length(Text) do {цикл по буквам текста}
    begin
      Letter := Text[i];         {текущая буква текста }
      If (Letter <> ' ') and Flag
        then Number := Number+1;
      Flag := (Letter=' ')      {(Letter=' ') – логическое
выражение,}
    end;                        {принимающее значения TRUE или
FALSE }
  WriteLn;
  WriteLn('О т в е т : количество слов в тексте равно ',
Number); ReadLn
END.
```

Пример 7.2. ***Определить, является ли заданное слово "перевёртышем" (слово называется "перевёртышем", если совпадает с собой после переворачивания).***

Система тестов

N теста	Данные	Результат
1	Slovo = "казак"	Otvet = "Перевертыш"
2	Slovo = "коза"	Otvet = "Не перевертыш"

Школьный АЯ

```
алг Перевертыш (арг лит Slovo, рез лит Otvet)
  надо | Otvet = "Перевертыш", если Slovo совпадает с собой
       | после переворачивания
нач цел Dlina, i, лог Flag
  Dlina:=длин(Slovo)
  i:=1; Flag:=да
  нц пока (i<=Dlina/2) и Flag          | цикл пока с прерыванием
до
    Flag:=(Slovo[i]=Slovo[Dlina-i+1])  | первой несовпавшей пары
букв,
    i:=i+1                             | если такая имеется в
слове
  кц
  если Flag
    то Otvet:="Перевертыш"
    иначе Otvet:="Не перевертыш"
  все
кон
```

Исполнение алгоритма

Обозначения проверяемых условий:

(i<=Dlina/2) и Flag => (1)

Slovo[i]=Slovo[Dlina-i+1] => (2)

Номер теста	i	(1)	(2)	Flag	Otvet
1	1 2	+ +	+ +	да да	"Перевертыш"

	3	-(кц)		да	
2	1 2	+ -(кц)	-	да нет	"Не перевертыш"

Turbo Pascal

```
Program TurnOver;
  Uses Crt;
  Var Slovo     : String;
      Dlina, i  : Integer;
      Flag      : Boolean;
BEGIN
  ClrScr;
  Write('Введите слово : ');  ReadLn(Slovo);
  Dlina:= Length(Slovo);
     {Сравниваются пары букв: первая буква с последней, }
     {вторая буква с предпоследней и т.д. }
  i:=1;  Flag := TRUE;
  While (i <= Dlina/2) and Flag do          {цикл до первой
несовпавшей }
    begin                                   {пары букв (если такая
есть)}
      Flag := (Slovo[i]=Slovo[Dlina-i+1]);
      i := i+1
    end;
  WriteLn; Write( 'О т в е т : слово ', Slovo);
  If Flag then WriteLn(' — перевертыш. ')
          else WriteLn(' — не перевертыш');
  ReadLn
END.
```

Пример 7.3. ***В заданном тексте одно заданное слово везде заменить на другое заданное слово такой же длины.***

Тест

Данные			Результат
Текст	Слово1	Слово2	
"2sinx+siny"	"sin"	"cos"	"2cosx+cosy"

Школьный АЯ

```
алг Замена (арг рез лит Текст, арг лит Слово1, Слово2)
  дано | длины Слово1 и Слово2 совпадают
  надо | в строке Текст подстрока Слово1 везде
       | заменена на подстроку Слово2
нач цел i, DS
  DS:=длин(Слово1)
  нц для i от 1 до длин(Текст)-DS+1
    если Текст[i : i+DS-1] = Слово1 | если вырезка равна
Слово1,
      то Текст[i : i+DS-1] :=Слово2 | то вырезке присваиваем
Слово2
```

```
          i:=i+DS                        | и продвигаемся на длину
слова

    все
  кц
кон
```

Исполнение алгоритма

Обозначение проверяемого условия:

Текст[i : i+DS-1] = Слово1 => (1)

Для тестовых данных имеем: DS=3, длин (Текст)-DS+1= 8.

i	Текст[i : i+2]	(1)	Текст
1	"2si"	-	"2sinx+siny"
2	"sin"	+	"2cosx +siny"
5	"x+s"	-	
6	"+si"	-	"2cosx+cosy"
7	"sin"	+	
10			

Turbo Pascal

(эта программа, использующая стандартную функцию Pos , не требует, чтобы длины заменяемого и вставляемого слов были одинаковыми)

```
Program Replace;
  Uses Crt;
  Var Text, Slovo1, Slovo2 : String;
      i, DlinaSlova, P     : Integer;
BEGIN ClrScr;
  Write('Введите строку : '); ReadLn(Text);
  Write('Какое слово заменить ? '); ReadLn(Slovo1);
  Write('На какое слово заменить ? '); ReadLn(Slovo2);
  WriteLn; WriteLn('О т в е т : ');
  WriteLn('Исходный текст: ', Text); DlinaSlova:=Length(Slovo1);
  DlinaSlova:=Length(Slovo1);
  P:=Pos(Slovo1,Text); {номер позиции, с которой в строке Text
}
                       {в первый раз встречается подстрока
Slovo1  }
  While P>0 do         {цикл продолжается до тех пор,пока
подстрока}
                       {Slovo1 встречается в строке Text
}
   begin
     Delete(Text, P, DlinaSlova); {удаление подстроки Slovo1,
начинаю-}
                                  {щейся с позиции P, из строки
Text  }
     Insert(Slovo2, Text, P);  {вставка подстроки Slovo2  }
                               { в строку Text с позиции P}
     P:=Pos(Slovo1, Text); {номер позиции, с которой подстрока
Slovo1}
                           {встречается в строке Text в
очередной раз}
   end;
```

```
  WriteLn('Новый текст: ', Text);
  ReadLn
END.
```

Пример 7.4. ***Заданную последовательность слов переупорядочить в алфавитном порядке (то есть выполнить лексикографическое упорядочение).***

Тест

Данные	Результат
Words=("стул", "гора", "яма", "стол")	Words=("гора", "стол", "стул", "яма")

Школьный АЯ (АЯ расширен добавлением типа данных **лит таб** и операций отношения для литерных переменных)

```
алг Расположить по алфавиту(арг цел NWords, арг рез лит таб Words[1:NWords])
  надо | Таблица Words упорядочена лексикографически
нач цел i, j, лит Tmp
  нц для i от 1 до NWords-1
    нц для j от i+1 до NWords
      если Words[i]>Words[j]          | условие перестановки слов
        то Tmp:=Words[i]; Words[i]:=Words[j]; Words[j]:=Tmp
      все
    кц
  кц
кон
```

Исполнение алгоритма

i	j	Words[i]>Words[j]	Массив Words
			"стул", "гора", "яма", "стол"
1	2 3 4	+ - -	"гора", "стул", "яма", "стол"
2	3 4	- +	"гора", "стол", "яма", "стул"
3	4	+	"гора", "стол" , "стул", "яма"

Turbo Pascal

```
Program LexOrder;
  Uses Crt;
  Var Words         : Array[1..10] of String; {массив слов}
      Tmp           : String;        {Tmp – вспомогательная
переменная}
      i, j, NWords : Integer;       {NWords – количество слов}
BEGIN
  ClrScr;
  Write('Количество слов в тексте – ');
  ReadLn(NWords);
  For i := 1 to NWords do
    begin Write(i, '-ое слово : ');
          ReadLn(Words[i])
    end;
```

```
  For i := 1 to NWords-1 do {лексикографическое упорядочение
слов}
    For j := i+1 to NWords do
      If Words[i]>Words[j] then
        begin
          Tmp := Words[i]; Words[i]:=Words[j]; Words[j]:=Tmp
        end;
  WriteLn;  WriteLn('О т в е т');
  WriteLn('Лексикографически упорядоченный массив слов:');
  For i := 1 to NWords do Write(Words[i], ' ');
  WriteLn; ReadLn
END.
```

Пример 7.5. ***Проверить, имеется ли в линейной записи заданной математической формулы баланс открывающих и закрывающих скобок.***

Система тестов

Номер теста	Проверяемый случай	Данные	Результат
1	При просмотре линейной записи слева направо первой встречается закрывающая скобка	"a)b+1("	"Нет баланса"
2	Первой встречается открывающая скобка, но число открывающих и закрывающих скобок не совпадает	"(a+b))"	"Нет баланса"
3	Есть баланс скобок	"(a+b/(c*d))"	"Есть баланс"

Школьный АЯ

```
алг Баланс скобок(арг лит S, рез лит Otvet)
нач цел Dlina, Flag, i
  i:=1; Flag:=0; Dlina:=длин(S)
  нц пока (Flag>=0) и (i<=Dlina)
     если S[i] = "("
       то Flag:=Flag+1
     все
     если S[i] = ")"
       то Flag:=Flag-1
     все
     i:=i+1
  кц
  если Flag=0
     то Otvet := "Есть баланс"
     иначе Otvet := "Нет баланса"
  все
кон
```

Turbo Pascal

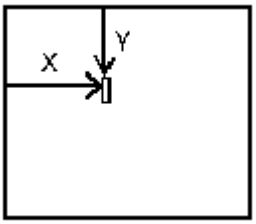
```
Program Balance;
  Uses Crt;
  Var S               : String;
      Dlina, Flag, i : Integer;
```

```
BEGIN ClrScr;
  GotoXY(15, 5);
  Write('Введите линейную запись математической формулы :');
  GotoXY(32,7); ReadLn(S);
  i:=1; Flag:=0; Dlina:=Length(S);
  While (Flag>=0) and (i<=Dlina) do
    begin
      If S[i] = '(' then Flag:=Flag + 1;
      If S[i] = ')' then Flag:=Flag - 1;
      i:=i+1
    end;
  GotoXY(32, 9); WriteLn('О т в е т');
  GotoXY(15,11);
  If Flag=0 then Write('Есть баланс ') else Write('Нет баланса
');
  WriteLn('открывающих и закрывающих скобок');
  ReadLn
END.
```

Задачи для самостоятельного решения

7.1. Подсчитайте количество запятых в заданном тексте.
7.2. Подсчитайте, сколько раз в заданном тексте встречается заданный символ.
7.3. Определите долю пробелов в заданной строке.
7.4. Проверьте, является ли заданное слово названием времени года на русском языке.
7.5. Замените в заданном тексте буквосочетание "*min*" на "*max*".
7.6. В заданном тексте подсчитате общее количество букв *"х"* и *"у"*.
7.7. В заданном тексте везде букву *"а"* замените на букву *"б"*, а букву *"б"* — на букву *"а"*.
7.8. Удвойте каждую букву в заданном тексте.
7.9. В заданном слове каждую букву *"б"* замените буквосочетанием *"ку"*.
7.10. Вычеркните из заданного слова все буквы *"а"*.
7.11. Подсчитате, сколько раз в заданном слове встречается буквосочетание *"аб"*.
7.12. Заданную строку *A* перепишите в обратном порядке в строку *B*.
7.13. Выясните, есть ли в заданном предложении буква *"ы"*.
7.14. Выясните, верно ли, что в заданном предложении *P* есть все буквы, входящие в заданное слово *S*.
7.15. Определите количество предложений в заданном тексте (предложение заканчивается либо точкой, либо вопросительным или восклицательным знаком).
7.16. Определите долю гласных букв в заданном тексте на русском (английском) языке.
7.17. Определите, является ли одно заданное слово обращением другого заданного слова.
7.18. Из заданного текста удалите те его части, которые заключены в кавычки (вместе с кавычками).
7.19. Каждые *n* символов во введенном тексте отделите знаком *"!"*.
7.20. Выясните, верно ли, что в заданном предложении есть пара соседствующих одинаковых символов.
7.21. Найдите хотя бы одно слово, которое встречается в каждом из трех заданных предложений.
7.22. Отредактируйте заданное предложение, удаляя из него все слова с чётными номерами.
7.23. В заданном предложении укажите слово, в котором доля гласных (*A, E, I, O, U* — строчных или прописных) максимальна.

7.24. Дан текст из цифр и строчных латинских букв, за которым следует точка. Определите, каких букв — гласных (*A, E, I, O, U*) или согласных — больше в этом тексте.
7.25. В заданную упорядоченную в лексикографическом (алфавитном) порядке последовательность слов вставьте новое заданное слово так, чтобы лексикографический порядок сохранился.
7.26. В заданной последовательности слов найдите все слова, начинающиеся с заданной приставки.
7.27. В заданной последовательности слов найдите все слова, имеющие заданное окончание.
7.28. Найдите самое длинное и самое короткое слово в заданном предложении.
7.29. В заданном тексте подсчитайте наибольшее количество подряд идущих пробелов.
7.30. Для каждого из слов заданного предложения укажите, сколько раз оно встречается в предложении.
7.31. Найдите самое длинное симметричное слово заданного предложения.
7.32. Из заданного текста выберите и напечатайте те символы, которые встречаются в нём ровно один раз.
7.33. Определите частоту вхождения в заданный текст каждой буквы русского алфавита.
7.34. Дана последовательность слов, в которой слова разделены запятыми, а за последним словом — точка. Напечатайте:
а) эту же последовательность слов, но в обратном порядке;
б) все различные слова, указав для каждого из них число его вхождений в последовательность.
7.35*. Отредактируйте заданное предложение, удаляя из него все лишние пробелы.
7.36*. Из заданного предложения удалите те слова, которые уже встречались в предложении раньше.
7.37*. Преобразуйте заданное десятичное натуральное число в римскую систему счисления.
7.38*. Преобразуйте число, заданное в римской системе счисления, в число десятичной системы.
7.39. Известны фамилии и имена учеников класса. Напечатайте список класса с указанием для каждого ученика количества его однофамильцев и тезок.
7.40. В заданном предложении найдите такие слова, которые, не имея приставки, начинаются с заданного корня, содержат или не содержат произвольный суффикс и имеют одно из трех заданных окончаний. Найденные слова распечатайте в алфавитном порядке.
7.41. Составьте целочисленный массив, элементами которого являются количества слов заданного текста на русском языке, начинающихся с соответствующей шипящей согласной ("ж", "ш", "ч"), прописной или строчной. Слова, содержащие менее трех букв, не учитывайте.
7.42*.Упорядочьте слова заданного предложения по возрастанию количества букв. Затем слова с одинаковым количеством букв упорядочьте по алфавиту (лексикографически).
7.43*. Определите, сколько слов заданного текста составлено из букв русского алфавита, а сколько — из букв латинского алфавита. Слова, в которых встречаются буквы обоих алфавитов, замените словом "Error".
7.44*. В заданном тексте на русском языке найдите структуры, которые могут обозначать фамилии и инициалы упоминаемых в тексте лиц (например, Павлов И.П. или И.П. Павлов), приведите их к стандартной форме <инициалы> <пробел> <фамилия> и занесите в отдельный массив.



Прежде, чем приступить к созданию графических программ на Turbo Pascal, необходимо ознакомиться с богатейшими графическими возможностями этого языка, сосредоточенными в стандартных модулях (библиотеках) ***GRAPH*** и ***CRT*** (название CRT происходит от *Cathode-Ray Tube* — электронно-лучевая трубка). Эти модули содержат описания стандартных констант, процедур и функций, используемых при работе с монитором в текстовом и графическом режимах.

- При работе в **текстовом режиме** экран монитора разбивается на строки, строки — на позиции, в каждую из которых можно поместить один символ. Положение символа на экране задается двумя координатами — вертикальной X и горизонтальной Y. X — это номер позиции в строке, а Y — номер строки. Чаще всего на экране в текстовом режиме размещается 25 строк по 80 позиций.

- При работе в **графическом режиме** весь экран разбивается на отдельные точки — **"пикселы"**. Положение пиксела также задается двумя координатами — X и Y. Координата X увеличивается слева направо, а координата Y — сверху вниз. Количество пикселов на экране зависит от типа графического адаптера и для распространённого адаптера VGA составляет 640 x 480.

Подключение модулей CRT и GRAPH к программе осуществляется с помощью ключевого слова Uses (англ. *uses* — использует) :
Uses Crt, Graph;

В системе программирования Turbo Pascal имеется хорошо развитая встроенная **служба помощи**, позволяющая получать подробное описание стандартных подпрограмм с примерами их применения. Поэтому ниже приведены только названия, описание параметров и назначение наиболее употребительных процедур и функций.

8.1. Модуль C R T

```
Модуль CRT содержит описания констант,
процедур и функций, обеспечивающих управление текстовым режимом
работы монитора и звуковым генератором.
```

Процедуры

ClrScr Очищает экран или окно и помещает курсор в верхний левый угол.

Delay(D: Word) Приостанавливает работу программы на указанное число D миллисекунд. Практически время задержки зависит от тактовой частоты процессора.
GotoXY(X, Y: Byte) Перемещает курсор в позицию X строки Y экрана.
NoSound Выключает источник звука.
Sound(F: Word) Запускает источник звука с частотой F (Гц).
TextBackGround(Color:Byte) Устанавливает цвет фона.
TextColor(Color: Byte) Устанавливает цвет символов.
Window(X1, Y1, X2, Y2: Byte) Определяет текстовое окно на экране. X1, Y1 — координаты левого верхнего угла окна, X2, Y2 — правого нижнего угла окна.

Функции

KeyPressed: Boolean Анализирует нажатие клавиши. Результат TRUE, если на клавиатуре нажата клавиша (кроме Alt, Ctrl и т.п.), и FALSE в противном случае. Не задерживает исполнение программы.
ReadKey: Char Читает символ с клавиатуры без эхоповтора на экране. Приостанавливает исполнение программы до нажатия на любую клавишу, кроме Alt, Ctrl и т.п.

8.1. Модуль G R A P H

Модуль Graph содержит константы, процедуры и функции для управления графическим режимом работы монитора.

Константы цвета

```
Black = 0; {Черный}          DarkGray = 8;   {Темносерый}
Blue = 1;   {Синий}          LightBlue = 9; {Яркосиний}
Green = 2; {Зеленый}         LightGreen = 10;  {Яркозеленый}
Cyan = 3; {Голубой}          LightCyan = 11;   {Яркоголубой}
Red = 4; {Красный}           LightRed = 12; {Розовый}
Magenta = 5;  {Фиолетовый}   LightMagenta = 13; {Малиновый}
Brown = 6;   {Коричневый}    Yellow = 14;   {Желтый}
LightGray = 7; {Светлосерый} White = 15; {Белый}
```

Константы типов и толщины линий

```
SolidLn = 0;   {Сплошная}          DashedLn = 3;  {Пунктирная}
DottedLn = 1; {Точечная}           NormWidth=1; {Нормальная толщина}
CenterLn = 2; {Штрихпунктирная}    ThickWidth = 3; {Тройная толщина}
```

Константы шаблона штриховки

```
EmptyFill = 0;          {Заполнение цветом фона}
```

SolidFill = 1;	{Сплошная штриховка}
LineFill = 2;	{Горизонтальная штриховка}
LtSlashFill = 3;	{/// штриховка}
SlashFill = 4;	{/// штриховка толстыми линиями}
BkSlashFill = 5;	{\\\ штриховка толстыми линиями}
LtBkSlashFill = 6;	{\\\ штриховка}
HatchFill = 7;	{Заполнение прямой клеткой}
XHatchFill = 8;	{Заполнение косой клеткой}
InterleaveFill = 9;	{Заполнение частой сеткой}
WideDotFill = 10;	{Заполнение редкими точками}
CloseDotFill = 11;	{Заполнение частыми точками}
UserFill = 12.	{Тип задается пользователем}

Процедуры

Arc(X, Y: Integer; U1, U2, R: Word) Строит дугу окружности текущим цветом с текущими параметрами линии. X, Y — координаты центра дуги, U1 — угол до начальной точки дуги, отсчитываемый против часовой стрелки от горизонтальной оси, направленной слева направо, U2 — угол до конечной точки дуги, отсчитываемый так же, как U1, R — радиус дуги.

Bar(X1, Y1, X2, Y2: Integer) Строит прямоугольник, закрашенный текущим цветом с использованием текущего стиля (орнамента, штриховки). X1, Y1, X2, Y2 — координаты левого верхнего и правого нижнего углов прямоугольника.

Bar3D(X1, Y1, X2, Y2: Integer; Glubina: Word; Top: Boolean) Строит параллелепипед, используя текущий стиль и цвет. X1, Y1, X2, Y2 — координаты левого верхнего и правого нижнего углов передней грани; Glubina — ширина боковой грани (отсчитывается по горизонтали), Top — признак включения верхней грани (если True — верхняя грань вычерчивается, False — не вычерчивается).

Circle(X, Y: Integer; R: Word) Рисует текущим цветом окружность радиуса R с центром в точке (X,Y).

ClearDevice Очищает графический экран, закрашивает его в цвет фона.

ClearViewPort Очищает выделенное графическое окно, закрашивает его в цвет фона.

CloseGraph Закрывает графический режим, т.е. освобождает память, распределенную под драйверы графики и файлы шрифтов, и восстанавливает текстовый режим работы экрана.

Ellipse(X, Y: Integer; U1, U2, XR, YR: Word) Рисует дугу эллипса текущим цветом; X, Y — координаты центра эллипса; U1, U2 — углы до начальной и конечной точек дуги эллипса (см. процедуру Arc); XR, YR — горизонтальная и вертикальная полуоси эллипса.

FillEllipse(X, Y: Integer; XR, YR: Word) Рисует заштрихованный эллипс, используя X,Y как центр и XR,YR как горизонтальную и вертикальную полуоси эллипса.

FillPoly(N: Word; Var PolyPoints) Рисует и штрихует многоугольник, содержащий N вершин с координатами в PolyPoints.
InitGraph(Var Driver, Mode: Integer; Path: String) Организует переход в графический режим. Переменные Driver и Mode содержат тип графического драйвера и его режим работы. Третий параметр определяет маршрут поиска графического драйвера. Если строка пустая (т.е. равна "), считается, что драйвер находится в текущем каталоге.
Line(X1, Y1, X2, Y2: Integer) Рисует линию от точки X1, Y1 до точки X2,Y2.
LineTo(X, Y: Integer) Рисует линию от текущего указателя к точке X,Y.
MoveTo(X, Y: Integer) Смещает текущий указатель к точке X,Y.
OutTextXY(X, Y: Integer; TextString: String) Выводит текст в заданное место экрана.
PieSlice(X, Y: Integer; U1, U2, Radius: Word) Строит сектор круга, закрашенный текущей штриховкой и цветом заполнения. X, Y — координаты центра сектора круга; U1 и U2 — начальный и конечный углы сектора, отсчитываемые против часовой стрелки от горизонтальной оси, направленной вправо; Radius — радиус сектора.
PutPixel(X, Y: Integer; Color: Word) Выводит точку цветом Color с координатами X, Y.
Rectangle(X1, Y1, X2, Y2) Рисует контур прямоугольника, используя текущий цвет и тип линии. X1, Y1 — координаты левого верхнего угла прямоугольника, X2, Y2 — координаты правого нижнего угла прямоугольника.
Sector(X, Y: Integer; U1, U2, XR, YR: Word) Рисует и штрихует сектор эллипса радиусами XR, YR с центром в X, Y от начального угла U1 к конечному углу U2.
SetBkColor(Color: Word) Устанавливает цвет фона.
SetColor(Color: Word) Устанавливает основной цвет, которым будет осуществляться рисование.
SetFillStyle(Pattern, Color: Word) Устанавливает образец штриховки и цвет.
SetLineStyle(LineStile, Pattern, Thickness: Word) Устанавливает толщину и стиль линии.
SetTextStyle(Font, Direction, CharSize: Word) Устанавливает текущий шрифт, направление (горизонтальное или вертикальное) и размер текста.
SetViewPort(X1, Y1, X2, Y2: Integer; ClipOn: Boolean) Устанавливает прямоугольное окно на графическом экране. Параметр ClipOn определяет "отсечку" элементов изображения, не умещающихся в окне.

Функции

GetMaxX и GetMaxY Возвращает значения максимальных координат экрана в текущем режиме работы, соответственно, по горизонтали и вертикали.

GraphResult Возвращает значение GrOk, соответствующее коду 0, если все графические операции программы выполнились без ошибок, или возвращает числовой код ошибки (от —1 до —14).

8.3. Примеры графических программ

Эти примеры иллюстрируют основные моменты, возникающие при написании графических программ:

- установку и закрытие графического режима;
- задание графических окон;
- вывод точек, линий, текста;
- использование различных шрифтов;
- установку цвета, палитры, типа штриховки;
- построение графических фигур (прямоугольников, многоугольников, дуг, окружностей, эллипсов, секторов);
- простейшие приемы анимации и звукового оформления.

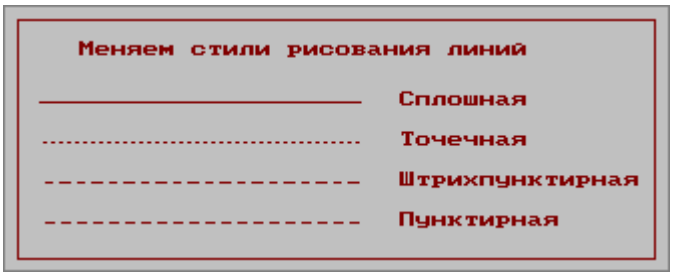
Даются окончательные подробно откомментированные тексты программ, которые могут служить основой для программ читателя. *Для их работы необходимо наличие библиотечного файла GRAPH.TPU, драйвера видеорежима EGAVGA.BGI (или другого, в зависимости от типа монитора) и файлов шрифтов (*.chr).*
Из-за недостатка места некоторые программы не содержат действий по выдаче сообщений о возможных ошибках графики, хотя они очень важны.

Пример 8.1. ***Эта программа демонстрирует работу процедур управления текстовым выводом на экран дисплея.***

```
Program ColorTable;
  Uses Crt; {подключение к программе библиотеки Crt}
  Const P = ' ';
  Var i, j : Integer;
BEGIN
  ClrScr; {очистка экрана}
  Window(1, 1, 80, 7); {определение окна для заголовочной части таблицы}
  TextColor(Yellow); {установка желтого цвета символов}
  GoToXY(24, 1); WriteLn('ТЕКСТОВЫЙ ВЫВОД НА ЭКРАН ДИСПЛЕЯ');
  GoToXY(30, 2); WriteLn('ТАБЛИЦА ЦВЕТНОСТИ');
  TextColor(LightCyan); {установка яркоголубого цвета символов}
  WriteLn('0-Черный ',P,'4-Красный ',P,'8-Темносерый ',P,'12-Розовый ');
  WriteLn('1-Синий ',P,'5-Фиолетовый ',P,'9-Яркосиний ',P,'13-Малиновый ');
  WriteLn('2-Зеленый ',P,'6-Коричневый ',P,'10-Яркозеленый ',P,'14-Желтый ');
  Write ('3-Голубой ',P,'7-Светлосерый',P,'11-Яркоголубой',P,'15-Белый ');
  TextColor(3+128); WriteLn(' i+128-Мерцание'); TextColor(White);
  For i := 0 to 9 do {цикл по цветам фона таблицы цветности}
    begin
```

```
      Window(i*8+1, 7, i*8+8, 25); {определение окна для столбца таблицы}
      GoToXY(1, 1); {курсор в верхнем левом углу окна}
      TextBackGround(Black); {установка черного цвета фона}
      WriteLn(' Фон', i:2);
      WriteLn('----------');
      TextBackGround(i); {установка текущего цвета фона окна }
      For j := 0 to 15 do
        begin
          TextColor(j); {установка текущего цвета надписей в окне }
          WriteLn('цвет', j:2);
        end;
    end; NormVideo; ReadLn
END.
```

Пример 8.2. ***Эта программа демонстрирует возможности изображения линий в графическом режиме.***



Внимание: будет работать только если Turbo Pascal установлен в каталог **C:\TP** и каталог **C:\TP\BGI** содержит необходимый файл **egavga.bgi**.

```
Program Lines;
  Uses Graph, Crt; {подключение к программе библиотек Crt и Graph}
  Var
    Key               : Char;
    LineStyle         : Word; {номер стиля рисования линии}
    Style             : String; {название стиля }
    GrDriver, GrMode  : Integer; {тип и режим работы графического драйвера}
    GrError           : Integer; {код ошибки графики}
BEGIN
  GrDriver := Detect; {автоопределение типа графического драйвера}
  InitGraph(GrDriver, GrMode, 'C:\TP\BGI'); {установка графического режима}
  GrError := GraphResult;
  If GrError<>GrOk then begin Writeln('Обнаружена ошибка!'); Halt
                        end;
  SetBkColor(LightGray); SetColor(Red); {цвет фона и цвет рисования }
  {-------------------------------------------------------------}
  OutTextXY(120, 100, 'Рисуем линию от точки (200,200) к точке (400,280)');
  Line(200, 200, 400, 280);
  Key:=ReadKey; {приостановление исполнения программы}
  ClearViewPort; {очистка окна}
  {-------------------------------------------------------------}
  OutTextXY(240, 80, 'Рисуем ломанную');
  Rectangle(110, 120, 520, 400); {рисование рамки }
  MoveTo(Round(GetMaxX/2), Round(GetMaxY/2)); {указатель в центре окна}
  Repeat {цикл прерывается нажатием любой клавиши}
    LineTo(Random(GetMaxX-250)+120, Random(GetMaxY-210)+120);
    Delay(100);
  until KeyPressed;
  Key := ReadKey; ClearViewPort;
  {-------------------------------------------------------------}
  OutTextXY(190, 80, 'Меняем стили рисования линий');
  For LineStyle := 0 to 3 do
```

```
    begin
      SetLineStyle(LineStyle, 0, 1);
      Case LineStyle of
        0: Style:='Сплошная';
        1: Style:='Точечная';
        2: Style:='Штрихпунктирная';
        3: Style:='Пунктирная'
      end;
      Line(120, 150+LineStyle*50, 430, 150+LineStyle*50);
      OutTextXY(450, 145+LineStyle*50, Style);
    end;
  Key:=ReadKey; ClearViewPort; {очистка окна}
  {-------------------------------------------------------}
  OutTextXY(180, 80, 'Меняем толщину рисования линий');
  SetLineStyle(0, 0, 1); {толщина 1 пиксел }
  Line(140, 200, 430, 200); OutTextXY(450, 195, 'Нормальная');
  SetLineStyle(0, 0, 3); {толщина 3 пиксела}
  Line(140, 250, 430, 250); OutTextXY(450, 245, 'Тройная');
  ReadLn; CloseGraph; {закрытие графического режима}
END.
```

Пример 8.3. ***Эта программа демонстрирует возможности изображения символов в графическом режиме (требует наличия в текущем каталоге файлов шрифтов *.chr).***

Внимание: будет работать только если Turbo Pascal установлен в каталог **C:\TP** и каталог **C:\TP\BGI** содержит необходимый файл **egavga.bgi**.

```
Program Symbols;
  Uses Graph, Crt; {подключение к программе библиотек Crt и Graph}
  Var
    Key               : Char;
    Font              : String; {названия шрифтов }
    Size, MyFont      : Word;
    GrDriver, GrMode  : Integer; {тип и режим работы графического драйвера}
BEGIN
  GrDriver := Detect; {автоопределение типа графического драйвера}
  InitGraph(GrDriver, GrMode, 'C:\TP\BGI'); {установка графического режима }
  If GraphResult <> GrOk then Halt;
  {-------------------------------------------------------}
  SetTextStyle(DefaultFont, HorizDir, 2);
  OutTextXY(140, 80, 'Меняем размер символов');
  OutTextXY(220, 100, 'и цвет фона');
  For Size := 0 to 13 do {Size - цвет фона и размер символов}
    begin SetBkColor(Size); {изменение цвета фона }
      Rectangle(135, 425, 470, 450); {рисование рамки }
      SetTextStyle(DefaultFont, HorizDir, 1);
      OutTextXY(150, 435, 'Для продолжения нажмите любую клавишу !');
      SetTextStyle(DefaultFont, HorizDir, Size);
```

```
      OutTextXY(250-Size*15, 200, 'HELLO');
      Key := ReadKey; ClearViewPort;
    end; ReadLn;
  {-------------------------------------------------------------}
  SetBkColor(LightGray); SetColor(Red);{цвет фона и цвет рисования }
  SetTextStyle(DefaultFont, HorizDir, 2);
              {установка шрифта, направления и размера символов}
  OutTextXY(70, 100, 'Располагаем строку горизонтально');
  SetTextStyle(DefaultFont, VertDir, 2);
  OutTextXY(310, 150, 'и вертикально');
  Key:=ReadKey; ClearViewPort;
  {-------------------------------------------------------------}
  SetTextStyle(DefaultFont, HorizDir, 2);
              {установка шрифта, направления и размера символов}
  OutTextXY(220, 30, 'Меняем шрифты');
  For MyFont := 0 to 9 do {цикл по номерам шрифтов}
    begin
      Case MyFont of
        0: Font:='0 - Точечный (Default)';
        1: Font:='1 - Утроенный (Triplex)';
        2: Font:='2 - Уменьшенный (Small)';
        3: Font:='3 - Прямой (SansSerif)';
        4: Font:='4 - Готический (Gothic)';
        5: Font:='5 - Рукописный';
        6: Font:='6 - Курьер';
        7: Font:='7 - Красивый (Таймс Italic)';
        8: Font:='8 - Таймс Roman';
        9: Font:='9 - Курьер увеличенный';
      end;
      SetTextStyle(MyFont, HorizDir, 2);
      OutTextXY(40, 70+MyFont*35, 'abcdfxyz 0123456789');{вывод текста}
      SetTextStyle(DefaultFont, HorizDir, 1);
      OutTextXY(410, 80+MyFont*35, Font) {вывод названия шрифта}
    end;
  OutTextXY(380, 60, 'N шрифта Описание'); ReadLn;
  CloseGraph; {закрытие графического режима}
END.
```

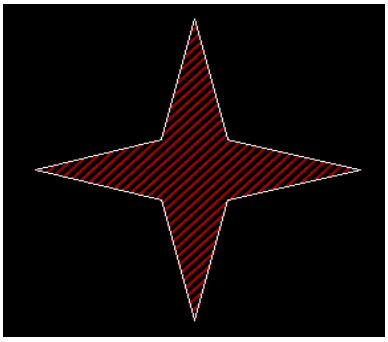
Пример 8.4. ***Эта программа рисует закрашенный прямоугольник, меняя случайным образом цвет, тип штриховки и высоту тона звукового сопровождения.***

Внимание: будет работать только если Turbo Pascal установлен в каталог **C:\TP** и каталог **C:\TP\BGI** содержит необходимый файл **egavga.bgi**.

```
Program MusicColor;
  Uses Crt, Graph; {подключение к программе библиотек Crt и Graph}
  Var
    GrDriver, GrMode: Integer; {тип и режим работы графического драйвера}
BEGIN
  GrDriver := Detect; {автоопределение типа графического драйвера}
  InitGraph(GrDriver, GrMode, 'C:\TP\BGI'); {установка графического режима}
  SetColor(White); {установка белого цвета рамки }
  Rectangle(130, 130, 460, 370); {рисование рамки }
  Randomize; {инициализация датчика случайных чисел}
  Repeat {цикл прерывается нажатием любой клавиши}
    Sound(Random(2000)); {изменение высоты звука }
    Delay(Random(1000)); {задержка }
    SetFillStyle(Random(4), Random(16)); {смена типа штриховки и цвета}
    Bar(140, 140, 450, 360); {рисование закрашенного прямоугольника}
  until KeyPressed;
  NoSound; {отмена звука }
  CloseGraph; ReadLn; {закрытие графического режима}
```

```
END.
```

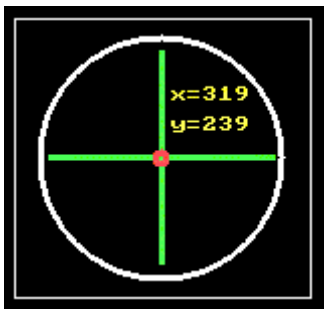
Пример 8.5. ***Эта программа рисует на экране звезду и закрашивает её, используя 12 типов штриховки.***



Внимание: будет работать только если Turbo Pascal установлен в каталог **C:\TP** и каталог **C:\TP\BGI** содержит необходимый файл **egavga.bgi**.

```
Program Star;
  Uses Crt, Graph;
       {подключение к программе библиотек Crt и Graph}
  Const { массив координат вершин многоугольника (звезды) }
    TopsStar: Array[1..18] of Integer = (300, 125, 325, 225, 425, 250,
          325, 275, 300, 375, 275, 275, 180, 250, 275, 225, 300, 125);
  Var
    i, j, GrDriver, GrMode : Integer;
BEGIN
  GrDriver := Detect;
  InitGraph(GrDriver, GrMode, 'C:\TP\BGI'); {установка графического режима}
  SetTextStyle(DefaultFont, HorizDir, 2); {установка шрифта,
                                           направления и размера символов}
  OutTextXY(220, 60, 'S T A R ');
  SetTextStyle(DefaultFont, VertDir, 2);
  OutTextXY(140, 150, 'S T A R ');
  SetTextStyle(DefaultFont, VertDir, 2);
  OutTextXY(500, 150, 'S T A R ');
  i:=0;
  Repeat
    j:=i mod 12; { j - остаток от деления i на 12 }
    SetFillStyle(j, Random(13)); { штриховка и фон }
    FillPoly(9, TopsStar); {рисование и штриховка звезды}
    Inc(i); {увеличение i на 1}
    Delay(500)
  until KeyPressed; {завершение цикла нажатием любой клавиши}
  CloseGraph
END.
```

Пример 8.6. ***Программа демонстрирует получение эффекта движения изображения прицела под управлением клавишей-стрелок клавиатуры с выводом координат центра прицела.***



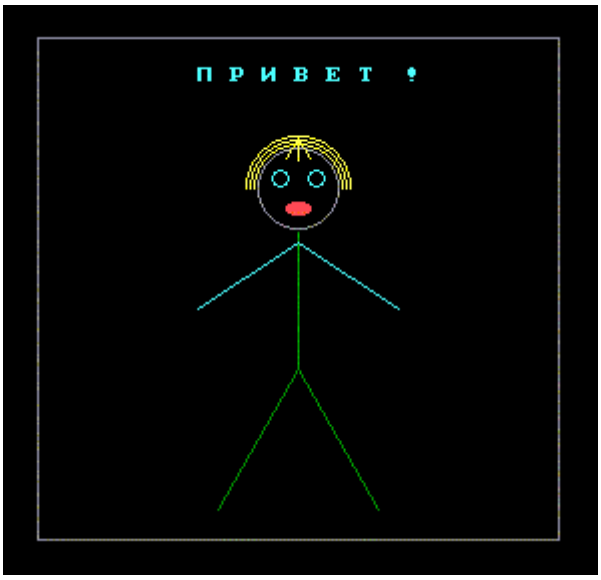
Внимание: будет работать только если Turbo Pascal установлен в каталог **C:\TP** и каталог **C:\TP\BGI** содержит необходимый файл **egavga.bgi.**

```
Program Sight;
  Uses Crt, Graph; {подключение к программе

                    библиотек Crt и Graph}
  Const Step = 5; {шаг изменения координат центра прицела }
    Instr = 'УПРАВЛЕНИЕ ДВИЖЕНИЕМ ПРИЦЕЛА - СТРЕЛКИ, ВЫХОД - ESC';
  Var
    GrDriver, GrMode : Integer; {тип и режим работы графического драйвера}
    X, Y             : Integer; {координаты центра прицела}
    XStr, YStr       : String;
    Ch               : Char;
{-------------------------------------------------------------}
Procedure MakeSight(X, Y : Integer); {процедура рисования прицела}
  Begin SetColor(White);
    Circle(X, Y, 80);
    SetColor(LightGreen);
    Line(X-80, Y, X+80, Y); Line(X, Y-63, X, Y+63); {вывод осей прицела}
    SetColor(LightRed); Circle(X, Y, 2); {окружность в центре прицела}
    Str(X, XStr); Str(Y, YStr); {перевод координат в строковый тип}
    SetColor(Yellow);
    OutTextXY(X+5, Y-35, 'x=' + XStr); {вывод координат центра прицела }
    OutTextXY(X+5, Y-20, 'y=' + YStr)
  End;
{-------------------------------------------------------------}
BEGIN
  GrDriver := Detect;
  InitGraph(GrDriver, GrMode, 'C:\TP\BGI');
  SetColor(LightGray);
  X := GetMaxX div 2; Y := GetMaxY div 2; {координаты центра экрана}
  Rectangle(50, 425, 600, 460); {рисование рамки }
  OutTextXY(120, 440, Instr);
  MakeSight(X, Y); {рисование прицела в центре экрана}
  While TRUE do {цикл работы программы до прерывания по клавише ESC}
    begin
      Ch := ReadKey;
      Case Ch of
        #27: begin CloseGraph; Halt(1) end; {выход по клавише ESC}
        #75: X: = X-Step; {изменение координат x, y нажатием стрелок}
        #77: X: = X+Step; {"влево", "вправо", "вверх", "вниз" }
        #72: Y: = Y-Step;
        #80: Y: = Y+Step
      end;
      ClearViewPort; { очистка графического экрана }
      SetColor(LightGray); {восстановление рамки с надписью}
      Rectangle(50, 425, 600, 460);
      OutTextXY(120, 440, Instr);
      MakeSight(X, Y) {рисование прицела в текущих координатах}
    end; CloseGraph;
```

```
END.
```

Пример 8.7. ***Программа рисует человечка, делающего утреннюю зарядку.***

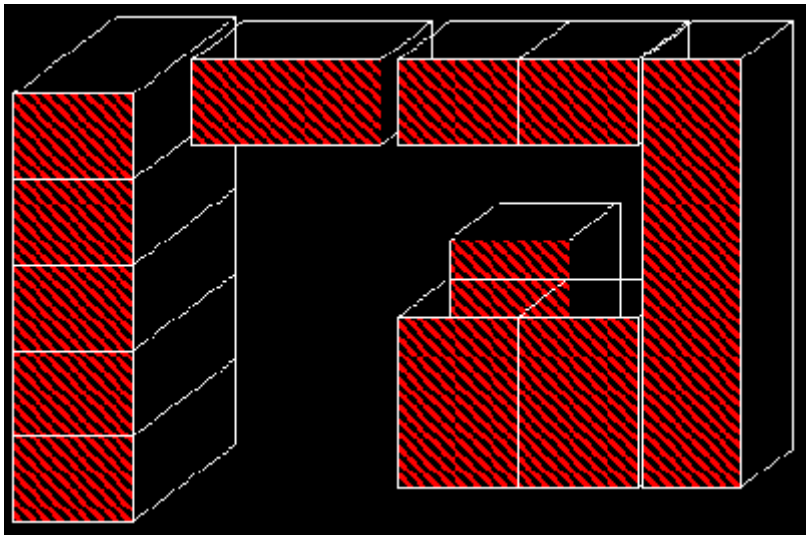


Внимание: будет работать только если Turbo Pascal установлен в каталог **C:\TP** и каталог **C:\TP\BGI** содержит необходимый файл **egavga.bgi**.

```
Program Animation;
  Uses Crt, Graph;
       {подключение к программе библиотек Crt и Graph}
  Const {вертикальные и горизонтальные координаты положения рук}
    Vert     : Array[1..3] of Integer = (190, 157, 120);
    Horizont : Array[1..3] of Integer = (200, 190, 200);
  Var
    GrDriver, GrMode, GrError, i, j : Integer;
BEGIN
  GrDriver := Detect; InitGraph(GrDriver, GrMode, 'C:\TP\BGI');
  GrError := GraphResult; If GrError <> GrOk then Halt;
  SetColor(LightGray); { установка светлосерого цвета для рамки}
  Rectangle(20, 20, 480, 400); {рисование рамки}
  SetColor(LightCyan); {установка яркоголубого цвета для текста}
  OutTextXY(200, 40, 'П Р И В Е Т !');
  SetColor(LightGray); Circle (250, 130, 20); {голова}
  SetColor(Yellow); Arc(250, 130, 0, 180, 26); {волосы}
  Arc(250, 130, 0, 180, 24); Arc(250, 130, 0, 180, 22);
  Line(250, 105, 244, 115); Line(250, 105, 250, 116); {чубчик}
  Line(250, 105, 256, 115);
  SetColor(LightCyan); Circle(241, 125, 4); {левый глаз }
  Circle(259, 125, 4); {правый глаз}
  SetColor(LightRed);
  SetFillStyle(SolidFill, LightRed);
  FillEllipse(250, 140, 6, 3); {рот }
  Setcolor(Green);
  Line(250, 152, 250, 220); {туловище }
  Line(250, 220, 210, 290); {левая нога }
  Line(250, 220, 290, 290); {правая нога}
  Repeat {цикл прерывается нажатием любой клавиши}
    For i := 1 to 3 do {Последовательный вывод трех положений рук:}
      begin { вниз, на уровне плеч, вверх }
        SetColor(LightCyan); Sound(200*i);
        Line(250, 157, Horizont[i], Vert[i]); {левая рука}
        Line(250, 157, 500-Horizont[i], Vert[i]); {правая рука}
        Delay(300); {задержка}
```

```
        SetColor(Black); {смена цвета на черный для повторного
                          рисования рук в том же положении
                          ("стирания" их с экрана) }
        Line(250, 157, Horizont[i], Vert[i]); {левая рука }
        Line(250, 157, 500-Horizont[i], Vert[i]); {правая рука}
      end
  until Keypressed;
  SetColor(LightCyan);
  Line(250, 157, Horizont[3], Vert[3]); {левая рука поднята }
  Line(250, 157, 500-Horizont[3], Vert[3]); {правая рука поднята}
  For i := 1 to 10 do { звуковая трель }
    begin
      Sound(1000);
      Delay(50);
      Sound(1500);
      Delay(50)
    end;
  NoSound; { выключение звука }
  CloseGraph;
END.
```

Пример 8.8. ***Эта программа демонстрирует возможности изображения объёмных предметов и столбиковых диаграмм.***

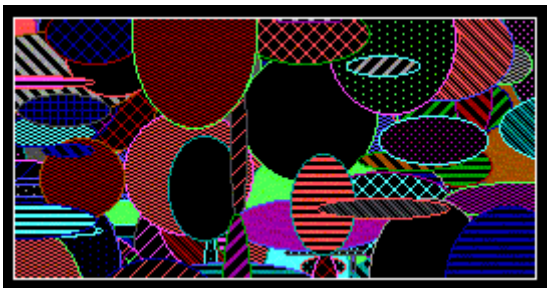


Внимание: будет работать только если Turbo Pascal установлен в каталог **C:\TP** и каталог **C:\TP\BGI** содержит необходимый файл **egavga.bgi**.

```
Program Design;
  Uses
    Graph, Crt; {подключение к программе библиотек Crt и Graph}
  Const
    Height           : Array[1..8] of Integer=(40,150,90,240,190,120,50,90);
                             {массив высот столбиков диаграммы}
  Var
    Color            : Word; {код цвета}
    Key              : Char;
    i, x, y, y1, h   : Integer;
    GrDriver, GrMode : Integer; {тип и режим работы графического драйвера}
    GrError          : Integer; {код ошибки графики}
BEGIN
  GrDriver := Detect; InitGraph(GrDriver, GrMode, 'C:\TP\BGI');
  GrError := GraphResult; If GrError <> GrOk then Halt;
```

```
  y := 120; h := 50; y1 := 140;
  SetTextStyle(DefaultFont, HorizDir, 2); {шрифт, направление, размер}
  OutTextXY(160, 20, 'Конструируем интерьер');
  SetFillStyle(5, LightRed); {тип штриховки и цвет (ярко красный)}
  For i := 4 downto 1 do
    begin {рисование параллелепипедов заданного размера}
      Bar3D(75, y1+i*h, 145, y1+(i+1)*h, 60, TopOff); Delay(200);
    end;
  Bar3D(75 , y1 , 145, y1+h , 60, TopOn); Delay(200);
  Bar3D(180, y , 290, y+h , 30, TopOn); Delay(200);
  Bar3D(330, 225 , 400, y+4*h , 30, TopOn); Delay(200);
  Bar3D(300, y+3*h, 370, y+5*h , 30, TopOn); Delay(200);
  Bar3D(370, y+3*h, 440, y+5*h , 30, TopOn); Delay(200);
  Bar3D(300, y , 370, y+h , 30, TopOn); Delay(200);
  Bar3D(370, y , 440, y+h , 30, TopOn); Delay(200);
  Bar3D(442, y , 500, y+5*h , 30, TopOn); Delay(200);
  Rectangle(135, 425, 470, 450); {рисование рамки для сообщения}
  SetTextStyle(DefaultFont, HorizDir, 1);
  OutTextXY(150, 435, 'Для продолжения нажмите любую клавишу !');
  Key := ReadKey; ClearViewPort; {очистка окна}
{-------------------------------------------------------------------}
  SetTextStyle(DefaultFont, HorizDir, 2);
  OutTextXY(100, 20, 'Рисуем столбиковую диаграмму');
  x := 50; Randomize; {инициализация датчика случайных чисел}
  For i := 1 to 8 do {цикл по столбикам диаграммы}
    begin
      Color := Random(12)+1; {задание кода цвета (кроме черного)}
      SetFillStyle(i, Color); {задание типа штриховки и цвета}
      SetColor(Color);
      Bar3D(x, 350-Height[i], x+50, 380, 20, TopOn); {рисование столбика}
      x := x+70; {изменение координаты x };
      Delay(200) {задержка}
    end;
  Key := ReadKey; CloseGraph; {Закрытие графического режима}
END.
```

Пример 8.9. ***Эта программа демонстрирует работу с пикселами, случайными эллипсами и секторами.***



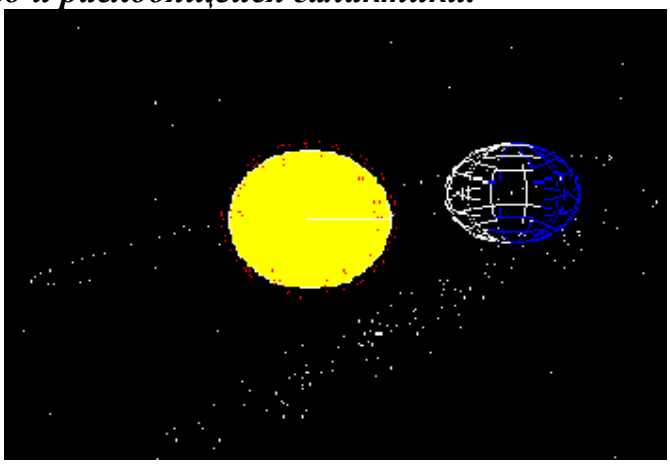
Внимание: будет работать только если Turbo Pascal установлен в каталог **C:\TP** и каталог **C:\TP\BGI** содержит необходимый файл **egavga.bgi**.

```
Program RandomFigures;
  Uses Graph, Crt;
  Var
    Key                       : Char;
    GrDriver, GrMode          : Integer;
    Radius, MaxX, MaxY, Ugol : Word; {параметры процедур}
BEGIN
  GrDriver := Detect; {автоопределение типа графического драйвера}
  InitGraph(GrDriver, GrMode, 'C:\TP\BGI'); {установка графического режима}
  SetTextStyle(DefaultFont, HorizDir, 2);
```

```
                {установка шрифта, направления и размера символов}
  OutTextXY(160, 50, 'Рисуем звездное небо');
  Rectangle(110, 90, 520, 380); {рисование рамки }
  Randomize; {инициализация датчика случайных чисел}

  Repeat {цикл прерывается нажатием любой клавиши}
    PutPixel(Random(GetMaxX-250)+120, Random(GetMaxY-210)+100,
    Random(15)); {вывод пикселя в области, ограниченной рамкой}
    Delay(5) {задержка}
  until KeyPressed;
  Key:=ReadKey; ClearDevice; {очистка графического экрана}
{-----------------------------------------------------------------}
  SetColor(White); {цвет рисования}
  OutTextXY(140, 30, 'Рисуем случайные эллипсы');
  Rectangle(100, 70, 560, 420); { рисование рамки }
  MaxX := GetMaxX;
  MaxY := GetMaxY;
  Radius := MaxY div 10;
  SetLineStyle(0, 0, 1); {толщина и стиль линии}
  SetViewPort(101, 71, 559, 419, ClipOn); {установка окна внутри рамки}
  Randomize; {инициализация датчика случайных чисел}
  Repeat {цикл прерывается нажатием любой клавиши}
    SetBkColor(Black); {цвет фона }
    SetColor(Random(13)+1); {цвет рисования}
    SetFillStyle(Random(12), Random(13)+1); {образец и цвет штриховки}
    FillEllipse(Random(MaxX), Random(MaxY), {координаты центра эллипса}
    Random(Radius), Random(Radius)); {полуоси эллипса}
  until KeyPressed;
  Key:=ReadKey;
  ClearDevice; {очистка графического экрана}
{-----------------------------------------------------------------}
  SetColor(White); SetViewPort(1, 1, GetMaxX, GetMaxY, ClipOn);
  OutTextXY(140, 20, 'Рисуем случайные секторы');
  Rectangle(90, 60, 570, 420); {рисование рамки}
  SetViewPort(92, 62, 569, 419, ClipOn); {установка окна внутри рамки}
  Repeat {цикл прерывается нажатием любой клавиши}
    SetFillStyle(Random(12), Random(13)+1); {изменение штриховки и цвета}
    Ugol := Random(360); {угол сектора}
    Sector(Random(MaxX-200), Random(MaxY-180), Random(Ugol), Ugol,
    Random(Radius*2), Random(Radius*2)); {рисование сектора}
  until KeyPressed;
  ClearViewPort; {очистка окна}
  CloseGraph; {закрытие графического режима}
END.
```

Пример 8.10. ***Программа изображает планету, вращающуюся вокруг Солнца на фоне мерцающих звезд и расходящейся галактики.***



Перемещение и изменение размеров изображений на экране можно организовать по разному. Так, в примере 8.6 эффект движения изображения достигается следующим образом: **выводится изображение, затем оно стирается с экрана с помощью процедуры ClearViewPort, повторно выводится с некоторым перемещением и т.д.**

В примере 8.7 применяется другой способ. **Сначала на экран выводится рисунок, затем тот же рисунок повторно**

изображается цветом фона, отчего он становится невидимым, далее рисунок выводится в исходном цвете, но с некоторым перемещением и т.д.

Оба способа имеют одинаковый недостаток —
движение изображения является прерывистым и вызывает мелькание экрана.

В этой программе для организации более плавного движения изображения по экрану **используется возможность формировать изображение на разных страницах видеопамяти** (обычно их две или четыре, в зависимости от типа графического адаптера).

Изображение сначала создается на странице с нулевым номером, видимой по умолчанию, а на невидимой странице с номером 1 формируется изображение с небольшим перемещением. Затем страница с номером 1 делается видимой, а новое изображение формируется на ставшей невидимой странице с нулевым номером и т.д.

Внимание: будет работать только если Turbo Pascal установлен в каталог **C:\TP** и каталог **C:\TP\BGI** содержит необходимый файл **egavga.bgi**.

```
Program Space; {составил студент Тетуев Р., мат.фак. КБГУ}
  Uses Graph, Crt;
  Const
    RadOrb = 250 {радиус орбиты Земли}; RadSun = 70 {радиус Солнца};
    RadGal = 100 {радиус галактики }; RadZem = 18 {радиус Земли };
    Naklon = 0.2 {коэффициент наклона плоскости орбиты Земли};
    PressZem = 0.65 {коэффициент сплющенности полюсов Земли};
    Compress = 0.8 {коэффициент сжатия при переходе из };
                   {расширения режима VGA в режим CGA }
  Var
    ZemX, ZemY, UgMer, PixelY, DUgZem , UpDown,
    XRad, Grad, UgZem, PixelX, StAngle, Ua, Ub,
    ParallelY , Color, ZemPix, EndAngle,
    VisualPage, GrMode, GrError, GrDriver, i : Integer;
    Ugol, CompressZem, Expansion,
    DUgol, Projection, PolUgol : Real;
BEGIN
  {установка графического режима и проверка возможных ошибок}
  GrDriver := EGA; GrMode := EGAHi;
  InitGraph(GrDriver, GrMode, 'C:\TP\BGI');
  GrError := GraphResult; If GrError<>GrOk then Halt;
  SetBkColor(Black);
  SetFillStyle(1, Yellow); {установка стиля заполнения и цвета Солнца}
  Ugol := 0; DUgol := 2*Pi/180; {орбитальное угловое смещение Земли}
  UgZem := 0; DUgZem := 14; {осевое угловое смещение Земли}
  {-----------------------------------------------------------------}
  VisualPage := 1;
  Repeat {цикл прерывается нажатием любой клавиши}
    SetVisualPage(1- (VisualPage mod 2));
         {установка номера видимой видеостраницы}
    VisualPage := VisualPage+1; {листание видеостраниц}
    SetActivePage(1 - (VisualPage mod 2));
         {установка номера невидимой (активной) видеостраницы,}
         {используемой для построения смещенного изображения }
    ClearDevice; {очистка графического экрана}
     {-------------------------------------------------------------}
    {Рисование "расходящейся" галактики}
    RandSeed:=1; {исходное значение датчика случайных чисел}
    Expansion:=VisualPage/100; {скорость расширения галактики}
    For i:= 1 to VisualPage do
      begin XRad := Trunc(Expansion*RadGal*Random);
```

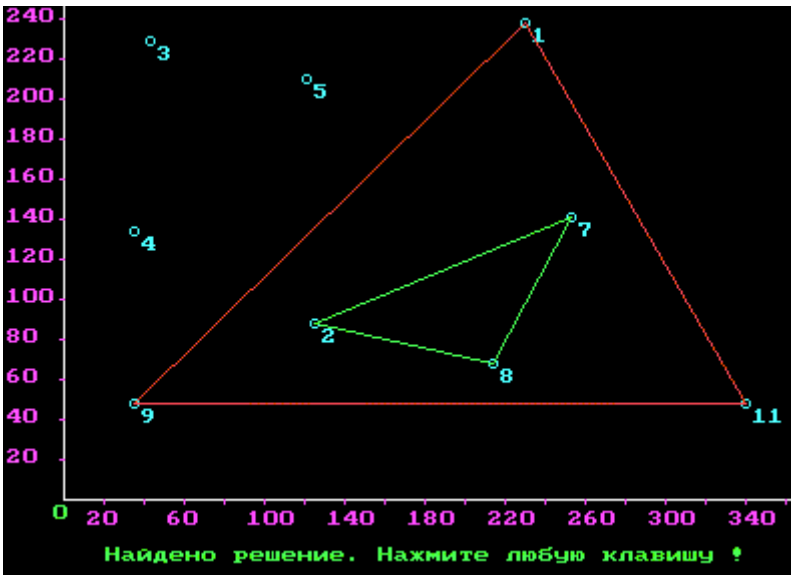
```
            {текущее расстояние от звезды до центра галактики}
          PolUgol:= 2*Pi*Random-VisualPage/30;
               {текущий центральный угол положения звезды галактики}
          PixelX := 370+Trunc(XRad*cos(PolUgol+1.8)); {координаты}
          PixelY := 250+Trunc(XRad*0.5*sin(PolUgol)); { звезды }
          PutPixel(PixelX, PixelY, White) {рисование звезды}
        end;
       {-------------------------------------------------------------}
      {Рисование мерцающих звезд}
      Randomize; {инициализация датчика случайных чисел}
      For i:=1 to 70 do
        PutPixel(Random(640),Random (350),White); {вспыхивающие звезды}
       {-------------------------------------------------------------}
      For i := 1 to 100 do {Рисование орбиты}
        PutPixel(320+Round(RadOrb * cos((i+VisualPage/5)*Pi/50+0.3)),
        160+Round(RadOrb*Naklon*sin((i+VisualPage/5)*Pi/50-Pi/2)),15);
       {-------------------------------------------------------------}
      PieSlice(310, 160, 0, 360, RadSun); {Рисование Солнца}
       {-------------------------------------------------------------}
      {Рисование Земли (ее параллелей и меридианов)}
      Ugol := Ugol+DUgol ; {угол поворота Земли относительно Солнца}
      Grad := Round(180*Ugol/Pi) mod 360; {в рад.(Ugol) и в град.(Grad)}
      ZemX := 320+Round(RadOrb*cos((Ugol+Pi/2+0.3))); { координаты }
      ZemY:=160+Round(RadOrb*Naklon*sin(Ugol)); {центра Земли}
      CompressZem := 2.5-cos(Ugol+0.3);
             {коэффициент учета удаленности Земли от наблюдателя}
      ZemPix := Round(RadZem*CompressZem); {текущий радиус Земли}
      UgZem := UgZem+DUgZem; {угол поворота Земли относительно своей оси}
      For i := 0 to 11 do { рисование меридианов }
        begin
          UgMer := (UgZem+i*30) mod 360;
          If (90<UgMer) and (UgMer<270) {установка начального и конечного}
            then begin StAngle := 90; EndAngle := 270 end { углов дуги }
            else begin StAngle := 270; EndAngle := 90 end; {эллипса
меридиана}
          Ua := (Grad+220) mod 360; Ub := (Grad+400) mod 360;
             {установка цветов рисования затененной и освещенной
              частей меридиана}
          Color := LightBlue;
          If Ua<=Ub then if (Ua<UgMer) and (UgMer<Ub) then Color := White;
          If Ua >Ub then if (Ua<UgMer) or (UgMer<Ub) then Color := White;
          SetColor(Color);
          XRad := round((ZemPix*cos(UgMer*Pi/180)));

Ellipse(ZemX,ZemY,StAngle,EndAngle,abs(XRad),round(PressZem*ZemPix));
        end;
      For i := 2 to 7 do {рисование параллелей}
        begin
          XRad := abs(Round(ZemPix*sin(i*Pi/9)));
             {большая полуось эллипса параллели}
          UpDown := Round(ZemPix*PressZem*cos(i*Pi/9));
             {высота параллели над плоскостью экватора}
          ParallelY := ZemY+UpDown; {координата Y центра эллипса
параллели}
          SetColor(LightBlue);
          Ellipse(ZemX, ParallelY, 0, 360, XRad, Round(Naklon*XRad));
             {затененная часть параллели}
          SetColor(White);

Ellipse(ZemX,ParallelY,Grad+220,Grad+400,XRad,Round(Naklon*XRad));
             {освещенная часть параллели}
        end;
       {-------------------------------------------------------------
}
```

```
    {Повторное рисование Солнца, если оно ближе к наблюдателю, чем
Земля}
    If CompressZem<2 then PieSlice(310, 160, 0, 360, RadSun);
     {-------------------------------------------------------------------
}
    RandSeed := VisualPage mod 12;
    For i := 1 to 250 do {Рисование протуберанцев}
      begin
        Projection := (1-sqr(Random))*Pi/2;
        XRad := RadSun+Round((20)*sin(Projection))-15;
        PolUgol := 2 * Pi * Random+VisualPage/20;
        {PolUgol, XRad - полярные координаты протуберанца}
        PixelX := 310 + Round( XRad * cos(PolUgol));
        PixelY := 160 + Round( Compress * XRad * sin(PolUgol));
        PutPixel(PixelX, PixelY, LightRed)
      end;
  until KeyPressed
END.
```

Пример 8.11. ***Программа рисует прямоугольную систему координат, отображает в ней заданное множество точек и строит все возможные пары треугольников с вершинами в этом множестве такие, чтобы один треугольник лежал строго внутри другого.***



Для работы программы необходимо предварительно создать в текущем каталоге текстовый файл **dan.dat**, содержащий координаты точек множества. Файл должен иметь структуру:

$x_1\ y_1\ x_2\ y_2 \ldots\ x_n\ y_n$, где $0 < x_i < 400$, $0 < y_i < 600$.

Пример файла dan.dat, содержащего координаты десяти точек:

20 20 150 40 90 300 500 400 50 380 110 130 370 290 300 140 70 60 500 170

Пустых строк в файле dan.dat быть не должно.

Внимание: будет работать только если Turbo Pascal установлен в каталог **C:\TP** и каталог **C:\TP\BGI** содержит необходимый файл **egavga.bgi**.

```
Program Triangles; {Составил студент Тезадов С., 1 к. мат. фак. КБГУ}
  Uses Crt,Graph;
  Const DemoN = 10;
    DemoX: array [1..DemoN] of Integer =
(20,150,90,500,50,110,370,300,70,500);
    DemoY: array [1..DemoN] of Integer =
(20,40,300,400,380,130,290,140,60,170);
  Var X, Y       : Array[1..50] of Integer; {координаты точек множества}
      InX, InY   : Array[1..50] of Integer; {координаты вершин внутренних}
      Flag       : Boolean; {треугольников}
      Ch         : Char;
      Coord, Num : String;
      i, j, k, p, i1, j1, k1, n, n1 : Integer;
      GrDriver, GrMode, GrError     : Integer;
{-------------------------}
Procedure InputOutput; {Описание процедуры считывания координат точек
                        множества из текстового файла dan.dat в массивы
                        X и Y и вывода точек на графический экран }
  Var f   : Text;
      a,b : Real;
 Begin
   Assign(f, 'dan.dat'); {установление связи между физическим }
                         {файлом dan.dat и файловой переменной f}
   {$I-}  {- отключаем автоматическую проверку существования файла}
   Reset(f); i:=0; {открытие файла f для чтения}
   {$I+}
   If IOResult = 0 then begin {если файл существует}
     While not eof(f) do {цикл "пока не будет достигнут конца файла"}
       begin Read(f,a,b); Inc(i); {считывание из файла f пары координат}
         X[i]:=Trunc(a-1); Y[i]:=Trunc(428-b) {преобразование декартовых}
       end; {координат в координаты графического экрана}
     n:=i; {n - количество введенных точек множества}
     Close(f); {закрытие файла f}
   end
   Else begin {если файла не существует, то используем множество точек,}
     n := DemoN; {заданное в DemoN, DemoX, DemoY.}
     For i:=1 to DemoN do begin
       x[i] := DemoX[i];
       y[i] := 428 - DemoY[i];
     end;
   end;
   SetColor(LightCyan);
   OutTextXY(200,30,'ИСХОДНОЕ МНОЖЕСТВО ТОЧЕК');
   For i:=1 to n do {рисование и нумерация точек множества}
     begin Circle(X[i], Y[i], 2);
        Str(i, Num); OutTextXY(X[i]+4, Y[i]+3, Num)
     end;
   Ch:=ReadKey; ClearViewPort; {очистка графического окна}
 End; {of InputOutput}
{-------------------------}
Procedure Drawing_Axes; {описание процедуры рисования осей координат}
  Begin SetColor(White);
    MoveTo(30,0); LineTo(30,430); LineTo(639,430); {оси OX,OY}
    OutTextXY(27,0,'^'); OutTextXY(630,427,'>'); {стрелки осей OX, OY}
    SetColor(LightGreen);
    OutTextXY(18,0,'y'); OutTextXY(630,434,'x');
    OutTextXY(25,433,'0');
    SetColor(LightMagenta); {установка розового цвета}
    For i:=1 to 20 do {нанесение делений и числовых отметок на ось OY}
      begin Str(20*(21-i), Coord); j:=i*20+10;
```

```
            OutTextXY(2, j-5, Coord);
            Line(28, j, 30, j)
      end;
    For i:=1 to 29 do {нанесение делений и числовых отметок на ось OX}
      begin Str(20*i,Coord); j:=i*20+30;
            If Odd(i) then OutTextXY(j-8, 436,Coord); Line(j,430, j,432)
      end;
    SetViewPort(31,4,630,429,FALSE) {установка текущего графического окна}
 End; {of Drawing_Axes}
{-------------------------}
Function Inside(i, j, k, p : Integer ) : Boolean;
   {функция Inside возвращает TRUE, если точка с номером p
    находится внутри треугольника с вершинами в точках i, j, k}
  Var S1, S2 : Real;
      {-------------------------------------------------}
  Function Area(x1, y1, x2, y2, x3, y3 : Real) : Real;
    {функция вычисления площади треугольника}
    {с вершинами в точках (x1,y1), (x2,y2), (x3,y3)}
    Begin Area:=abs((x1*(y2-y3)+x2*(y3-y1)+x3*(y1-y2))/2)
    End; {of Area}
 {-------------------------------------------------------}
Begin S1:=Area(X[i], Y[i], X[j], Y[j], X[k], Y[k]);
      {S1 - площадь треугольника с вершинами в точках i, j, k}
   S2 := Area(X[i], Y[i], X[j], Y[j], X[p], Y[p]) +
         Area(X[j], Y[j], X[k], Y[k], X[p], Y[p]) +
         Area(X[k], Y[k], X[i], Y[i], X[p], Y[p]);
      {S2 - сумма площадей трех треугольников с вершинами
       в точках (i,j,p), (j,k,p), (i,k,p) }
   Inside:=S1>S2 - 0.001
End; {of Inside}
{-------------------------}
Procedure Triangle(x1, y1, x2, y2, x3, y3 : Integer; Color : Byte);
  Begin {описание процедуры рисования треугольника цвета Color}
    SetColor(Color);
    Line(x1, y1, x2, y2);
    Line(x2, y2, x3, y3);
    Line(x3, y3, x1, y1)
  End; {of Triangle}
{-------------------------}
BEGIN
  GrDriver:=Detect;
  InitGraph(GrDriver, GrMode, 'C:\TP\BGI');
  GrError:= GraphResult;
  If GrError<>GrOk then begin WriteLn(' Ошибка графики!'); Halt end;
  Drawing_Axes; {вызов процедуры рисования осей координат}
  InputOutput; {вызов процедуры ввода и вывода исходных данных}
  Flag:=FALSE;
  For i:=1 to n -2 do {циклы по номерам вершин внешнего треугольника}
    For j:=i+1 to n -1 do
      For k:=j+1 to n do
        begin
          SetColor(LightCyan); {установка яркоголубого цвета}
          For p:=1 to n do {рисование и нумерация точек множества}
            begin Circle(X[p], Y[p], 2); {рисование точки}
                  Str(p, Num);
                  OutTextXY(X[p]+4, Y[p]+3, Num) {вывод номера точки}
            end;
          n1:=0; {занесение координат точек, находящихся
                  внутри треугольника, в массивы InX и InY}
          For i1:=1 to n do
            begin
              If (i1<>i) and (i1<>j) and (i1<>k) and Inside(i,j,k,i1)
                then begin Inc(n1); InX[n1]:=X[i1]; InY[n1]:=Y[i1]
                     end;
```

```
            end;
          If n1>=3 then {если число точек внутри треугольника не меньше
трех,}
           begin Flag:=TRUE; {то строятся внутренние треугольники}
            For i1:=1 to n1-2 do {циклы по номерам вершин внутренних}
             For j1:=i1+1 to n1-1 do {треугольников}
              For k1:=j1+1 to n1 do
               begin {рисование внешнего треугольника красным цветом}
                Triangle(X[i],Y[i],X[j],Y[j],X[k],Y[k],LightRed);
                  {рисование внутреннего треугольника зеленым цветом}
                Triangle(InX[i1],InY[i1],InX[j1],InY[j1],InX[k1],InY[k1],
                         LightGreen);
                OutTextXY(80,450,'Найдено решение. Нажмите любую клавишу!');
                Ch:=ReadKey;
                SetColor(Black); {"стирание" сообщения}
                OutTextXY(80,450,'Найдено решение. Нажмите любую клавишу!');
                  {"стирание" внутреннего треугольника}
                Triangle(InX[i1],InY[i1],InX[j1],InY[j1],InX[k1],InY[k1],
                         Black)
               end {конец циклов по номерам вершин внутренних треугольников}
           end;
           {"стирание" внешнего треугольника}
          Triangle(X[i], Y[i], X[j], Y[j], X[k], Y[k], Black)
        end; {конец циклов по номерам вершин внешнего треугольника}
  SetColor(White);
  If not Flag then OutText('Для данного множества нет решений задачи')
              else OutText('РАБОТА ПРОГРАММЫ ЗАВЕРШЕНА');
  OutTextXY(80,450,' Нажмите любую клавишу ...');
  Ch:=ReadKey;
  CloseGraph {закрытие графического режима}
END.
```