Министерство образования и науки Российской Федерации
Департамент образования и молодежной политики Ханты-Мансийского автономного округа — Югры
Нижневартовский государственный гуманитарный университет
Факультет информационных технологий и математики
Кафедра информатики и МПИ

Т.Б.Казиахмедов

ПОДГОТОВКА ШКОЛЬНИКОВ К ОЛИМПИАДЕ ПО ПРОГРАММИРОВАНИЮ

Учебно-методическое пособие

Издательство
Нижневартовского государственного
гуманитарного университета
2011

ББК 74.263.2я7
К 14

Печатается по постановлению Редакционно-издательского совета
Нижневартовского государственного гуманитарного университета

Рецензент
кандидат физико-математических наук, зав. кафедрой математики
и информатики Нижневартовского филиала ТюмГУ *Н.П.Дмитриев*

Казиахмедов Т.Б.

К 14 **Подготовка школьников к олимпиаде по программированию:** Учебно-методическое пособие. — Нижневартовск: Изд-во Нижневарт. гуманит. ун-та, 2011. — 68 с.

ISBN 978–5–89988–806–9

В пособии рассматривается класс задач по программированию, аналоги которых предлагались на всероссийских и международных олимпиадах по программированию. К рассматриваемым задачам даны пояснения, приведена их программная реализация, а в некоторых случаях и варианты решений.

Для преподавателей вузов, аспирантов, учителей информатики и студентов.

ББК 74.263.2я7

Изд. лиц. ЛР № 020742. Подписано в печать 11.03.2011
Формат 60×84/8. Бумага для множительных аппаратов
Гарнитура Times. Усл. печ. листов 8,5
Тираж 500 экз. Заказ 1160

Отпечатано в Издательстве
Нижневартовского государственного гуманитарного университета
628615, Тюменская область, г.Нижневартовск, ул.Дзержинского, 11
Тел./факс: (3466) 43-75-73, E-mail: izdatelstvo@nggu.ru

ISBN 978–5–89988–806–9

Часть 1
РЕКУРСИИ КРУГОМ

Как правило, в школе данный стиль программирования практически игнорируется. Но существует класс задач, понимание решения которых без рекурсивного стиля невозможно.

Рекурсию можно довести до учащихся при помощи довольно простых и понятных задач.

Задача 1.1. Вывести на экран все натуральные числа от 1 до N. Не использовать циклы, условный или безусловный переход.

```
uses crt;
var n,m:byte;
procedure demorecur(m:byte);
begin
write(m,' ');
if m<n then demorecur(m+1);
end;
begin
writeln('N=');
readln(N);
demorecur(1);
readln;
end.
```

Здесь необходимо обратить внимание, что в рекурсии выполняется вывод числа и рекурсивный вызов. Можно заменить тип byte на тип word и испытывать рекурсию до переполнения. После замены введите значение N = 2645.

Задача 1.2. Вычислить сумму натуральных чисел от 1 до N. Не использовать циклы, условный или безусловный переход на метки.

Вариант-функция.

```
uses crt;
var m:integer; {переменная для ограничения рекурсии}
function sum(n:integer):longint;
begin
if n=0 then sum:=0;
if ((n>=1) and (n<=m)) then begin writeln(n);sum:=sum(n-1)+n;end;
end;
begin
readln(m);
writeln(sum(m):10);
readln;
end.
```

Вариант-процедура.

```
uses crt;
var m:integer; {переменная для ограничения рекурсии}
v:double; {переменная для запоминания суммы}
procedure sum(n:integer;var s:double);
begin
if n=0 then s:=0;
if ((n>=1) and (n<=m)) then  begin write(n,' ');sum(n-1,s);s:=s+n;end;
end;
begin
```

```
readln(m);
sum(m,v);write(v:10:0);
readln; end.
```

Задача 1.3. Заполнить одномерный массив случайными числами с помощью рекурсии.

Ниже приводится возможный вариант решения.

```
uses crt;
var m:integer;{переменная m<=100}
mas:array[1..100] of integer;

procedure fill(n:integer);{рекурсивное заполнение с выводом номера и значения}
элемента}
begin
mas[n]:=random(100);
if ((n>=1) and (n<=m)) then  begin writeln(n,' ',mas[n]);fill(n+1);end;
end;
begin
readln(m);
clrscr;
fill(1);
readln;
end.
```

Задача 1.4. Заполнить двумерный массив случайными числами с помощью рекурсии, причем в ней вызывается N раз рекурсия для заполнения строк (т.е. одномерных массивов).

Рассмотрим один из возможных вариантов решения.

```
uses crt;
var n,m,m1,m2,m3:integer;
mas:array[1..100,1..100] of integer;

{перепишем рекурсию из предыдущего задания следующим образом}
procedure fill(n:integer);
begin
if (n<=m1) then begin mas[n,m]:=random(100); write(mas[n,m],' ');
fill(n+1);end;
end;

{создаем рекурсию для заполнения двумерного массива. В ней вызывается процедура
fill}
procedure  fill2(m2:integer);
begin
mas[n,m]:=random(100);
if m2<=m3 then begin fill(m);m:=1;writeln;fill2(m2+1);end;
end;
begin
randomize;
n:=1;m:=1;
m1:=4; m3:=4;
fill2(n);
readln;
end.
```

Задача 1.5. Вывести на экран несколько семейств концентрических окружностей, используя рекурсивную процедуру okr(x, y, r), где x, y — центр окружностей, r — радиусы, причем условием рекурсивного вызова является r > 10 (ограничение по r и определяет количество концентрических окружностей).

```
uses crt,graph;
var dr,dm:integer;
procedure okr(x,y,r:integer);
begin
circle(x,y,r);
setcolor(random(14)+1);
if r>10 then okr(x,y,r-5);
end;
begin
dr:=detect;
initgraph(dr,dm,'');
{Вызовы рекурсии в цикле определят различные виды на семейства концентрических
окружностей. Экспериментируйте сами}
repeat
okr(200,100,100);
okr(315,200,50);
until keypressed;
 readln;
 end.
```

Задача 1.6. Задать движение окружности по контуру большой окружности, используя рекурсию.

Здесь усложнена процедура из предыдущего задания. Кроме того, учащиеся должны знать параметрическое уравнение окружности.

```
uses crt,graph;
var
dr,dm:integer;
x,y:integer;
h:real;

procedure okr(h:real);
begin
x:=round(100+30*cos(h));{Получаем координаты точек большой окружности}
y:=round(100+50*sin(h));
setfillstyle(1,4);
circle(x,y,8);floodfill(x,y,14);
cleardevice;
if h<2*pi then okr(h+0.009); {условие выхода из рекурсии}
end;
begin
dr:=detect;
initgraph(dr,dm,'');
setcolor(14);
repeat
okr(0);
until keypressed;
readln;
closegraph;
end.
```

Задача 1.7. Лабиринт. Может ли путник выйти из лабиринта? Если может, то напечатать путь от выхода до начального положения путника. Лабиринт задан массивом A размером 40×40, в котором:

A[k, m] = 0, если клетка [k, m] “проходима”;

A[k, m] = 1, если клетка [k, m] “непроходима”.

Начальное положение путника задается в проходимой клетке [i, j]. Путник может перемещаться из одной проходимой клетки в другую, если они имеют общую сторону. Путник выходит из лабиринта, когда попадает в граничную клетку (т.е. клетку [k, m], где k или m равны 1 или 40).

Это хорошо известная задача. Решение распадается на 2 части: поиск пути для выхода и печать обратного пути — от выхода до начального положения.

```
uses crt;
const mm=15;
       nn=15;
var m,n,i,j:integer;
v_yh:boolean;
a:array[1..mm,1..nn] of byte;
procedure l(i,j:integer);
begin
if not v_yh then
if a[i,j]=0 then
begin
if(i=1) or (i=m) or(j=1) or(j=n) then v_yh:=true;
a[i,j]:=1;L(i,j-1);L(i,j+1);l(i-1,j);l(i+1,j);
{Обратите внимание на рекурсивные вызовы}
if v_yh then writeln(i,' ',j);
end
end;
Begin
write('m,n=');readln(m,n);
for i:=1 to m do begin
for j:=1 to n do
begin
a[i,j]:=random(2);{Формирование лабиринта случайным образом. Нужно задавать
лабиринты самим}
write(a[i,j]:2);
end;
writeln;
end;
writeln('i,j');
readln(i,j);
L(i,j);
if not v_yh then writeln('нет выхода');
readln;
end.
```

Задача 1.8. Лабиринт. Лабиринт задан в виде матрицы размером n на m. Стенам лабиринта соответствуют единицы, проходам — нули. Определить, можно ли из точки с координатами (i1, j1) попасть в точку с координатами (i2, j2).

Для усложнения задачи можно предложить указать самый короткий путь из заданной точки, причем из всех путей одинаковой длины выбирать путь с наименьшим числом поворотов. Наиболее простым способом решения задач по поиску пути является рекурсивный поиск. Его алгоритм во многом аналогичен рассмотренному в задаче 1.7. Пусть *A(nxm)* — матрица, задающая лабиринт (1 — стена, 0 — проход). Напишем процедуру рекурсивного перемещения по лабиринту. Путь прохода будем отмечать цифрами, начиная с 2 (2, 3, 4 и т.д.). Пусть на *k*-том шаге мы попали на клетку с координатами *(i, j)*. Если это та клетка, путь до которой необходимо было отыскать (с координатами (i_2, j_2)), то задача

решена. Необходимо распечатать матрицу с отмеченным путем и прекратить выполнение программы. Если нет, то необходимо проверить, можно ли перейти на соседнюю клетку (справа, слева, сверху или снизу), и если возможно, то вызвать процедуру перемещения уже с этой клетки. Если перехода на соседнюю клетку нет, то необходимо очистить исходную клетку, чтобы ее можно было использовать в других вариантах при поиске пути. Обозначим **move (i, j, k)** — процедуру рекурсивного перемещения (**i, j** — номер клетки, **k** — номер шага), **writematr** — процедуру, распечатывающую матрицу A, тогда программа, реализующая предложенный алгоритм, может быть записана следующим образом:

```
const n=4; m=5;
type matr=array [1..n,1..m] of integer;
{Матрица, задающая варианты прохода. 1-стена; 0-проход.}
const labir:matr=((1,0,0,0,1),
                  (0,0,1,0,1),
                  (1,0,0,0,0),
                  (0,0,1,0,0));
var a:matr;
    i1,j1,i2,j2,f:byte;
    k:integer;
procedure writematr;
var i,j:byte;
begin
for i:=1 to n do
      begin
         for j:=1 to m do write(a[i,j]:3,' ');
         writeln;
      end
end;
procedure move(i,j:byte; k:integer);
begin
   a[i,j]:=k; k:=k+1;
   if (i=i2)and(j=j2) then begin writematr; f:=1; halt end
   else
   begin
     if (i+1<=n)and(a[i+1,j]=0) then move(i+1,j,k);
     if (j+1<=m)and(a[i,j+1]=0) then move(i,j+1,k);
     if (j-1>0)and(a[i,j-1]=0) then move(i,j-1,k);
     if (i-1>0)and(a[i-1,j]=0) then move(i-1,j,k);
   end;
   a[i,j]:=0; k:=k-1;
end;
begin
  a:=labir;
  writeln('Координаты i1, j1'); readln(i1,j1);
  writeln('Координаты i2, j2'); readln(i2,j2);
  f:=0; k:=2;
  if(a[i1,j1]=1)or(a[i2,j2]=1) then
  writeln ('Неверные координаты')
  else move(i1,j1,k);
  if f=0 then writeln('Прохода нет');
end.
```

Задача 1.9. Получить треугольник Серпинского k-го порядка (Ковер Серпинского).

```
 program trserpinskogo;
uses graph;
var
dr,dm:integer;
x1, y1, x2, y2, x3, y3: real;
n:integer;  {Процедура построения треугольника}
```

```
procedure triang (x1, y1, x2, y2, x3, y3: real);
begin
line (round(x1), round(y1) ,round(x2), round(y2));
line (round(x2), round(y2) ,round(x3), round(y3));
line (round(x3), round(y3) ,round(x1), round(y1));
end;
procedure tr(x1, y1, x2, y2, x3, y3: real; n: integer) ;
var
x1n, y1n, x2n, y2n, x3n, y3n: real;
begin
if n>0 then
begin
x1n:= (x1+x2)/2;{Найдем координаты середин сторон}
y1n:= (y1+y2)/2;
x2n:= (x2+x3)/2;{Найдем координаты середин сторон}
y2n:= (y2+y3)/2;
x3n:= (x3+x1)/2;{Найдем координаты середин сторон}
y3n:= (y3+y1)/2;
triang(x1n, y1n, x2n, y2n, x3n, y3n); {Рисуем треугольник в центре}
tr(x1, y1, x1n, y1n, x3n, y3n, n -1); {Обработка крайних треугольников}
tr(x2, y2, x1n, y1n, x2n, y2n, n -1);
tr(x3, y3, x2n, y2n, x3n, y3n, n -1);
end;
end;
begin
{write ('Введите координаты вершин треугольника:');
readln (x1, y1, x2, y2, x3, y3);}

write ('Введите порядок фигуры:');
readln (n);
dr:=detect;
initgraph(dr,dm,'');
triang (10, 10, 320, 440, 600,10);{Исходный треугольник}
tr (10, 10, 320, 440, 600, 10, n); {Обработка исходного треугольника}

readln;
closegraph;
end.
```

Часть 2
ЗАДАЧИ НА МАССИВЫ

Мы уже рассмотрели две задачи на массивы-лабиринты. Задачи на массивы можно разделить на следующие группы:

- заполнение массивов по определенной схеме или правилу;
- использование массивов как контейнеров данных, обычно для хранения параметров состояния объектов;
- преобразование массивов, упорядочивание, перебор и другие.

Приведем задачи на массивы, которые рассматривались на Всероссийских олимпиадах по программированию.

Задача 2.1. Заполнить квадратную матрицу размера n на n натуральными числами от 1 до n2 в указанном порядке:

Например:

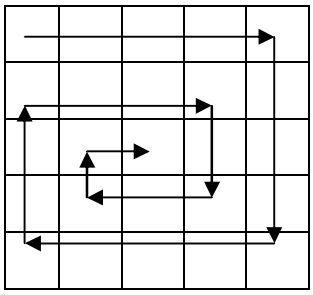
$$\begin{bmatrix} 1 & 2 & 3 & 4 \\ 8 & 7 & 6 & 5 \\ 9 & 10 & 11 & 12 \\ 16 & 15 & 14 & 13 \end{bmatrix}$$

Здесь направление заполнения зависит от четности строки массива: слева направо для нечетных строк и справа налево для четных строк.

Ниже приводится возможный вариант программы.

```
uses crt;
var
a: array [1..100,1..100] of integer;
n,m,i,j,k:byte;
begin
k:=1;
writeln('N');
readln(n);
for i:=1 to n do begin
if (i mod 2=1) then {если индекс строки нечетное число}
for j:=1 to n do begin a[i,j]:=k;k:=k+1;end;
if (i mod 2=0) then если индекс строки четное число}
 for j:=n downto 1 do begin a[i,j]:=k;k:=k+1;end;
 end;
 clrscr;
 for i:=1 to n do begin
 for j:=1 to n do write(a[i,j]:4,' ');
 writeln;
 end;
 readln;
 end.
```

Задача 2.2. Ввести число N и заполнить двумерный массив размером N×N числами 1, 2, …N∗N по спирали.



Разделим виток спирали на 4 прямолинейных участка: горизонтальный слева направо, вертикальный сверху вниз, горизонтальный справа налево, вертикальный снизу вверх. В программе каждый из них заполняется отдельно и используется «скатывание» одного участка на другой. Окончание проверяется по k = N×N. Создадим логическую функцию mov, которая присваивает элементу массива значение, ведет счет числу рассмотренных элементов. Функция принимает значение False, как только все элементы массива заполнены.

```
uses crt;
const nn=19;
var i,j,k,n:integer;
a:array[1..nn,1..nn] of integer;
function mov:boolean;
begin
mov:=false;
if k<=n*n then begin a[i,j]:=k;k:=k+1;mov:=true;end
end;

begin
write('N=');readln(n);k:=1;i:=1;j:=1;
repeat
while mov and (i+j<n+1) do j:=j+1;k:=k-1; {слева направо}
while mov and (i<j) do i:=i+1;k:=k-1;{сверху вниз}
while mov and (i+j>n+1) do j:=j-1;k:=k-1; {справа налево}
while mov and (i>j+1) do i:=i-1;k:=k-1;{снизу вверх}
until k=n*n;
For i:=1 to n do begin
for j:=1 to n do
write(a[i,j]:4);
writeln;
end;
readln;
end.
```

Задача 2.3. В заданном двумерном массиве A [1..m, 1..n] заменить нулями элементы, стоящие в строках или столбцах, где имеются нули.

Проще всего завести вспомогательный массив C [1..n]. Массив A просматривается по строкам. Столбцы, где встречались нулевые элементы, отмечаются в массиве C, а встретился ли нуль в просматриваемой строке — отмечается в переменной z, т.е.

If A[i, j] = 0 then begin c[j] = true; z: = true;

После просмотра строки i она обнуляется, если z = true. В заключение просматривается массив C и обнуляются столбцы j, где C[j] = true.

```
uses crt;
const NN=20;
       MM=20;
var n,m,i,j:integer;
z:boolean;
```

```
A:array[1..nn,1..mm] of real;
c:array[1..nn] of boolean;
begin
writeln('M,N');
readln(n,m);
for i:=1 to m do
for j:=1 to n do
begin
write('a[',i,',',j,']=');
readln(a[i,j]);
end;
for j:=1 to n do c[j]:=False;
for i:=1 to m do
begin
z:=false;
for j:=1 to n do
if a[i,j]=0 then begin z:=true;c[j]:=true;end;
if z then for j:=1 to n do a[i,j]:=0;
end;
for j:=1 to n do
if c[j] then for i:=1 to n do a[i,j]:=0;
for i:=1 to m do
begin
for j:=1 to n do write(a[i,j]:4:0);
writeln;
end
readln;
end.
```

Задача 2.4. В массиве A [1: N] каждый элемент равен 0, 1, 2. Переставить элементы массива так, чтобы сначала были нули, затем 1, затем 2. (Дополнительного массива не заводить).

При решении этой задачи нужно сосчитать, сколько в массиве нулей, единиц и двоек, заполнить массив требуемым образом.

```
uses crt;
var a:array[1..100] of byte;
z0,z1,z2:integer;
i,n:integer;
begin
write('N=');readln(n);
clrscr;
for i:=1 to n do begin a[i]:=random(3);write(a[i]:2);end;
readln;
z0:=0;z1:=0;z2:=0;
for i:=1 to n do begin
if a[i]=0 then inc(z0);
if a[i]=1 then inc(z1);
if a[i]=2 then inc(z2);
end;
for i:=1 to z0 do a[i]:=0;
for i:=z0+1 to z0+z1 do a[i]:=1;
for i:=z0+z1+1 to n do a[i]:=2;
for i:=1 to n do write(a[i]:2);
readln;
end.
```

Задача 2.5. В массиве A [1..m, 1..n] все числа различны. В каждой строке находится минимальный элемент, затем среди этих чисел выбирается максимальное. Напечатать номер строки, в которой расположено найденное число.

1 вариант

```
uses crt;
const mm=10;
       nn=20;
label s1;
var i,j,k,m,n:integer;
a:array[1..mm,1..nn] of integer;
max,min:integer;
begin
write('m=');readln(m);
write('n=');readln(n);
for i:=1 to m do
for j:=1 to n do
begin
write('a[',i,',',j,']=');readln(a[i,j]);
end;
for i:=1 to m do
begin
for j:=1 to n do
begin
if (i>1) and (a[i,j] <= max) then goto s1;
if (j=1) or (a[i,j]<=min) then min:=a[i,j];
end;
max:=min;k:=i;
s1:end;
writeln(k);
readln
end.
```

2 вариант

Запомним все минимальные элементы в массиве C, а затем найдем максимальный элемент этого массива.

```
uses crt;
const mm=10;
       nn=20;
var i,j,k,m,n:integer;
a:array[1..nn,1..nn] of integer;
c:array[1..nn] of integer;
max,min:integer;
begin
{write('m=');readln(m);}
write('n=');readln(n);
for i:=1 to n do begin
for j:=1 to n do
begin
a[i,j]:=random(20);write(a[i,j]:3);
end;
writeln;
end;

for i:=1 to  n do
begin
min:=a[i,1];
for j:=1 to n do
```

```
begin
if min >a[i,j] then min:=a[i,j]; end;
c[i]:=min;
end;
max:=0;
writeln;
for i:=1 to n do write(c[i]:3);
for i:=1 to n do
if max< c[i] then begin max:=c[i];j:=i;end;
write('max=',max:3,'  номер строки=',j);
readln
end.
```

Задача 2.6. Даны два упорядоченных массива A, B. Образовать из элементов этих массивов упорядоченный массив C.

1 вариант

```
uses crt;
const mm=100;
       nn=100;
       mmpnn=200;
var m,n,i,j,k:integer;
a,b:array[1..mm] of integer;
c:array[1..mmpnn] of integer;

begin
write('m,n='); readln(m,n);
for i:=1 to m do begin a[i]:=i*3;write(a[i]:4);end;
for i:=1 to n do begin b[i]:=i*9;write(b[i]:4);end;
readln;
i:=1;j:=1;
for k:=1 to m+n do
begin
if((i>n) or (a[i]>b[j]))  and not(j>n) then
begin c[k]:=b[j];j:=j+1;end
else begin c[k]:=a[i];i:=i+1;end
end;
For i:=1 to m+n do write(c[i]:4);
readln;
end.
```

2 вариант

```
uses crt;
const mm=100;
       nn=100;
       mmpnn=200;
var m,n,i,j,k,k1:integer;
a,b:array[1..mm] of integer;
c:array[1..mmpnn] of integer;

begin
write('m,n='); readln(m,n);
for i:=1 to m do begin a[i]:=i*3;write(a[i]:4);end; writeln;
for i:=1 to n do begin b[i]:=i*9;write(b[i]:4);end;
readln;
i:=1;j:=1;
for k:=1 to m+n do
begin
if (a[i]>=b[j]) and (i<m)  then
```

```
begin c[k]:=a[i];i:=i+1;end
else begin c[k]:=b[j];j:=j+1;end ;

if i=m then  begin for k1:=j to n do c[k1]:=b[k1]; break; end;
if j=n then  begin for k1:=i to m do c[k1]:=a[k1]; break; end;
end;
For i:=1 to m+n do write(c[i]:4);
readln;
end.
```

Задача 2.7. Квадратики. Дан массив A [1..m, 1..m], каждый элемент которого равен 0, 1, 5 или 11. Подсчитать в нем количество четверок A[i, j], A[i + 1,j], A[i, j + 1], A[i + 1, j + 1], в каждой из которых все элементы различны.

```
uses crt;
const mm=50;
numb:array[1..4] of integer=(0,1,5,11);
var i,j,m,s:integer;
a:array[1..mm,1..mm] of integer;
begin
writeln('M=');readln(m);
for i:=1 to m do begin
for j:=1 to m do begin
a[i,j]:=numb[random(5)];write(a[i,j]:3);end;
writeln;
end;
s:=0;
for i:=1 to m-1 do
for j:=1 to m-1 do
if (a[i,j]+a[i,j+1]+a[i+1,j]+a[i+1,j+1]=17) then s:=s+1;
writeln(S);
readln;
end.
```

Задача 2.8. В круге стоят N человек. Они пронумерованы от 1 до N. Поочередно из круга начинает выходить каждый третий человек. Это продолжается до тех пор, пока в круге не останется последний человек. Определить его номер.

Например, если в круге стояло 7 человек, то его поочередно покинут 3, 6, 2, 7, 5, 1. Оставшимся будет человек, стоявший на 4 месте. Вывести последним этот номер.

Пояснение: заведем массив A [N], удаление будет означать присвоение элементу значения нуль.

```
uses crt;
const nn=30;
var
a:array[1..nn] of integer;
n,p,k,i,j:integer;
begin
write('n=');readln(n);
for i:=1 to n do begin a[i]:=i;write(i,' ');end;writeln;
k:=1;
p:=1;
i:=0;
while k<>n do begin
i:=i+1;
if a[i]<>0 then begin p:=p+1;end;
if p>3 then begin write(a[i],' ');a[i]:=0;k:=k+1;p:=1;end;
if i>n then i:=0;
end;
```

```
for i:=1 to n do if a[i]<>0 then begin write(a[i]);a[i]:=0;end;
readln;
end.
```

Задача 2.9. Написать программу, которая печатает только те элементы одномерного массива A (1000), индексы которых — числа Фибоначчи.

```
uses crt;
var kol,i,j:word;
a:array [1..1000] of word;
function fib(n:word):longint;
begin
if ((n=0) or (n=1)) then fib:=1;
if n>=2  then fib:=fib(n-2)+fib(n-1);
end;
begin
j:=1;
{определение количества чисел Фибоначчи для охвата всех индексов массива}
while fib(j)<1000 do begin
j:=j+1;
kol:=kol+1;
end;
clrscr;
{Заполнение массива генератором случайных чисел}
for i:=1 to 1000 do a[i]:=random(1500)+5;
writeln('Элемент','   Индекс ');
for i:=1 to 1000 do
begin
for j:=1 to kol+1 do
if i=fib(j) then writeln(a[i]:8,' ',i:6);
end;readln;end.
```

Задача 2.10. Мода. В целочисленном массиве A [1..n] найти число, повторяющееся максимальное количество раз. Если таких чисел несколько — одно из них.

```
uses crt;
var
x,y,n,i,j,ma,k:integer;
a:array[1..100] of integer;
begin
write('N');readln(n);
clrscr;randomize;
for i:=1 to n do begin a[i]:=random(10)+1;write(a[i]:3);end;
writeln;
k:=0;
for i:=1 to n do begin
x:=a[i];
for j:=1 to n do
if (a[j]<>200) and (x=a[j]) then begin k:=k+1;y:=a[j];a[j]:=200;end;
if ma<=k then begin ma:=k; writeln(y,' - ',ma);end;
k:=0;
end;
readln;
end.
```

Задача 2.11. Несоставляемое число. Задан массив натуральных чисел P [1..n]. Найти минимальное натуральное число, не представимое суммой никаких элементов массива P. Сумма может состоять и из одного слагаемого, но каждый элемент массива может входить в него только один раз.

Решение. Положим s = 1 и посмотрим, есть ли в массиве P какой-либо элемент P [j] ≤ s. Если такого элемента нет, то s будет решением. Если такой элемент найдется, то добавим его к значению s, т.е. положим s = s + P [j], удалим P [j] из массива и снова поищем, найдется ли среди оставшихся элемент, не превосходящий нового значения s, и т.д.

Фактически на очередном i = 1, 2,.. шаге надо просматривать элементы массива P, начиная с P [i], а найденный элемент P [j] заменить на P [i] (для экономии числа действий можно предварительно упорядочить массив P).

```
uses crt;
const nn=30;
label PI,BR;
var
n,i,j,s,s1:integer;
p:array[1..nn] of integer;
begin
write('N=');readln(N);
randomize;
s1:=0;
for i:=1 to n do begin p[i]:=random(100);inc(s1,p[i]);write(p[i]:3);end;
writeln;writeln('s1=',s1);
s:=1;
for i:=1 to n do
begin
for j:=i to n do if p[j]<=s then goto Pi;
goto BR;
PI: s:=s+P[j];P[j]:=P[i]
end;
BR:writeln(s);
readln;
end.
```

Задача 2.12. На квадратном листе бумаги в клетку размером 100×100 клеток нарисовано несколько прямоугольников. Каждый прямоугольник состоит из целых клеток, прямоугольники не накладываются друг на друга и не соприкасаются.

Задан массив размером 100×100, в котором элемент A [i, j] = 1 если клетка [i, j] принадлежит какому-либо прямоугольнику, в противном случае A [i, j] = 0. Написать программу, которая сосчитает и напечатает число прямоугольников.

Пояснение. Прямоугольников столько, сколько их северо-западных углов (верхних левых**).**

```
uses crt;
const mm=100;nn=100;

var m,n,i,j,s:integer;
A:array[0..mm,0..nn] of byte;
begin
write('M,N=');readln(m,n);
for i:=0 to m do
for j:=0 to n do
if (i=0) or(j=0) then A[i,j]:=0 else
begin
write('A[',i,',',j,']=');readln(A[i,j]);
end;
s:=0;
for i:=1 to m do
for j:=1 to n do
if(A[i,j]=1) and(A[i-1,j]+A[i,j-1]=0) then s:=s+1;
writeln(s);readln;
end.
```

Задача 2.13. Седловая точка. Задан числовой массив A [m, n]. Некоторый элемент этого массива назовем седловой точкой, если он является одновременно наименьшим в своей строке и наибольшим в своем столбце. Напечатать номера строки и столбца какой-нибудь седловой точки или число 0, если нет такой точки.

Пояснение: если mini — минимум элементов Aij в строке i, а Mj — максимум элементов Aij в столбце j, то mini ≤ Aij ≤ Mj, что позволяет сделать несколько выводов:

а) минимум в любой строке не более максимума в любом столбце, т.е. всегда mini ≤ Mj;

б) если для каких-нибудь i и j будет выполнено равенство mini = Mj, то максимальный минимум совпадает с минимальным максимумом, а на пересечении строки I со столбцом J будет стоять седловая точка mini = Aij = Mj.

```
uses crt;
const mm=20;nn=20;
label SI,BRI;

var m,n,i,j,i0,mi,ma:integer;
A:array[1..mm,1..nn] of integer;
begin
write('M,N=');readln(m,n);
for i:=1 to m do begin
for j:=1 to n do begin
A[i,j]:=random(50);write(a[i,j]:4);end;
writeln;
end;
for i:=1 to m do
begin
for j:=1 to n do
begin
if(i>1) and(A[i,j]<=ma) then goto SI;
if(j=1) or(A[i,j]<mi) then mi:=A[i,j]
end;
ma:=mi;i0:=i;
SI:end;
for j:=1 to n do begin
for i:=1 to m do
if A[i,j]>ma then goto BRI;
writeln(i0:3,J:3,ma:4);readln;exit;
Bri:end;
writeln(0);readln;
end.
```

Задача 2.14. Центральное селение. Имеется k селений. Если в селении i расположить пункт скорой помощи, то поездка по вызову в селение j займет время A [i, i] + A [i, j] (1 ≤ i, j ≤ k, i ≠ j). Найти номер селения i, от которого поездка в самое удаленное (по времени) селение занимало бы минимальное время. Массив A [k, k] задан. В нем все элементы A [i, j] > 0 и элемент A [i, j] может быть не равен A [j, i].

Пояснение. Сформулируем эту задачу следующим образом. В каждой строке i массива выберем максимальное среди чисел A [i, j] (j ≠ i) и сложим его с A [i, i]. Найти i, при котором соответствующая сумма будет минимальна.

```
uses crt;
const kk=20;

var i,j,k,i1,s,t:integer;
A:array[1..kk,1..kk] of integer;
begin
write('K=');readln(k);
```

```
randomize;
for i:=1 to k do begin
for j:=1 to k do begin
A[i,j]:=random(50);write(a[i,j]:4);end;
writeln;
end;
for i:=1 to k do
begin
s:=0;
for j:=1 to k do
if(i<>j) and(s<A[i,j]) then s:=A[i,j];
s:=s+A[i,i];
if (i=1) or (s<t) then begin i1:=i; t:=s end
end;
writeln(i1);
readln;end.
end;
ma:=mi;i0:=i;
SI:end;
for j:=1 to n do begin
for i:=1 to m do
if A[i,j]>ma then goto BRI;
writeln(i0:3,J:3,ma:4);readln;exit;
Bri:end;
writeln(0);readln;
end.
```

Задача 2.15. Рюкзак. Из заданных n предметов выбрать такие, чтобы их суммарный вес был менее 30 кг, а стоимость наибольшей. Напечатать суммарную стоимость выбранных предметов.

Точнее, заданы 2 массива положительных чисел A [1..n] и B [1..n]. Выбрать такие попарно различные числа i1, i2, i3,…, ik, чтобы сумма

A [i1] + A [i2] + A [i3] +… A [ik] < 30, а сумма

B [i1] + B [i2] + B[i3] +… B [ik] = max была максимальной.

Пояснение. После того как предметы, весящие 30 кг, удалены, а остальные расположены в каком-либо порядке, определим дерево вариантов следующим образом.

На очередном ходе i = 1, 2,…, n будем рассматривать предмет с номером i, а вариантов j хода i всегда будет два: j = 0 означает брать предмет, а j = 1 — не брать.

Кроме заданных массивов A, B заведем массив P [1..n] и несколько переменных:

i — номер очередного предмета,

S — вес предметов в рюкзаке,

Z — суммарная стоимость предметов в рюкзаке,

ZM — максимальная стоимость рассмотренных вариантов,

P [k] = 0 или P [k] = 1, если k ≤ I взят или не взят в рюкзак.

Вначале I, Z, S, ZM обнуляются.

При движении вперед мы пытаемся добавить предмет в рюкзак (если S + A [i] < 30). В этом случае мы идем по левой ветке:

S = S + A [i], Z = Z + B [i], P [i] = 0.

Если предмет добавить нельзя, то мы его не беспокоим (движемся по правой ветке, отбрасывая все дерево вариантов, идущих влево, и отмечаем P [i] = 1).

В обоих случаях продолжаем двигаться вперед, пока не будет рассмотрен последний предмет.

Если все предметы рассмотрены, то вариант получен. Он сравнивается с ZM.

IF ZM < Z THEN ZM: = Z и начинается движение назад.

При движении назад пропускается вся группа идущих подряд взятых предметов (у них P [i] = 0), поскольку изменения в одной этой группе могут лишь снизить суммарную стоимость предметов в рюкзаке. Просмотренные предметы попутно удаляются из рюкзака

If P[i] = 0 THEN S = S – A[i];Z = Z – B[i];

Далее пропускается вся группа не взятых ранее предметов (у них P [i] = 1), т.к. изменение в этой группе приводит к левой ветке, которая должна была быть оценена раньше. Короче говоря, мы движемся назад до достижения такого номера I, что P [i] = 0, P [I + 1] = 1. При этом движении из рюкзака удаляются имеющиеся там предметы. После этого мы движемся вперед. Если нужного i не окажется, то работа закончена.

```
uses crt;
const nn=100;
      T=30;
label V;
var i,s,z,Sm,zm,n:integer;
A,B:array[1..nn] of integer;
P:array[1..nn] of boolean;
begin
write('N=');readln(n);
for i:=1 to n do begin
write('A[',i,']=');readln(a[i]);
write('B[',i,']=');readln(b[i]);
end;
s:=0;z:=0;zm:=0;i:=0;
V: for i:=i+1 to n do
if s+A[i]>=T then P[i]:=false
else begin
s:=s+A[i];
z:=z+B[i];
P[i]:=true;end;
if zm<z then zm:=z;sm:=s;
for i:=n-1 downto 1 do
begin
if p[i+1] then
begin
s:=s-A[i+1];
z:=z-B[i+1];
end;
if P[i] and not p[i+1] then
begin
s:=s-A[i];
z:=z-B[i];
P[i]:=false;
goto V;
end;
end;
writeln(zm,' ',sm);
readln;
end.
```

Часть 3

СВОЙСТВА ЧИСЕЛ. ПОСЛЕДОВАТЕЛЬНОСТИ. СИСТЕМЫ СЧИСЛЕНИЯ

Задача 3.1. Разложить на слагаемые и напечатать все представления натурального числа N суммой натуральных чисел. Перестановка слагаемых нового способа не дает.

Например, при N = 5

5 = 5
5 = 4 + 1
5 = 3 + 2
5 = 3 + 1 + 1
5 = 2 + 2 + 1
5 = 2 + 1 + 1 + 1
5 = 1 + 1 + 1 + 1 + 1

Например, при N = 6

6 = 6
6 = 5 + 1
6 = 4 + 2
6 = 4 + 1 + 1
6 = 3 + 3
6 = 3 + 2 + 1
6 = 3 + 1 + 1 + 1
6 = 2 + 2 + 2
6 = 2 + 2 + 1 + 1
6 = 2 + 1 + 1 + 1 + 1
6 = 1 + 1 + 1 + 1 + 1 + 1

```
uses crt;
const nn=100;
label RA,BR;
var
n,i,k,t,s:integer;
M:array[1..nn] of integer;
begin
write('N');readln(n);
M[1]:=n;k:=1;i:=1;
RA:t:=M[k]-1;s:=t+i-k+1;
for i:=k to n do
if s>t then begin M[i]:=t;s:=s-t end
else begin M[i]:=s;goto BR end;
BR: for k:=1 to i do write(M[k],' ');writeln;
for k:=i downto  1 do if M[k]>1 then goto RA;
readln;
end.
```

Задача 3.2. Вывести все простые числа меньше M.

```
uses crt;
{простые до M}
var S, m, i,j:longint;
```

```
label met;
begin
clrscr;
writeln('Введите M');
readln(m);
for i:=2 to m do
begin
s:=0;
for j:=2 to m do
begin
if i mod j=0  then begin s:=s+1;if s>1 then goto met;end;
 end;
 if s<2 then write (i,'  ');
 met:
 end;
 readln;
 end.
```

Задача 3.3. С клавиатуры вводятся натуральные числа, меньшие M (M < 100000). Если число простое, то оно добавляется в массив целых чисел SS.

```
uses crt;
{простые до M}
var S, m, i,j,kol:longint;
SS:array[1..100] of longint;
label met,met1;
begin
clrscr; kol:=1;
met1:
writeln('Введите M');
readln(m);
begin
s:=0;
for j:=2 to m do
begin
if m mod j=0  then begin s:=s+1;end;
 end;
 if s<2 then begin ss[kol]:=m;kol:=kol+1;end;
 if kol<10 then goto  met1;
 met:
 end;
 for i:=1 to kol-1 do write(ss[i],' ');
 readln;
 end.
```

Задача 3.4. Определить простые делители для натурального числа N.

Будем менять число I = 2, 3,… до тех пор, пока I не станет делителем числа N. Поделим N на I (N = N/I) и будем повторять деление, пока N делится на I. После этого, если N >1, перейдем к следующему значению I, и т.д.

Введем еще одну переменную J, равную вначале 0, а в дальнейшем — последнему найденному делителю. Новый делитель в первый раз будет отличен от J.

Чтобы ускорить процесс, можно отдельно рассматривать случай I = 2, а в дальнейшем двигаться по нечетным значениям I.

```
uses crt;
var i,j,n:integer;
begin
writeln('N');
readln(n);
```

```
j:=0;
i:=2;
while n>1 do begin
while (n mod i<>0) do i:=i+1;
if i<>j then begin writeln(i); j:=i;end;
n:=n div i;
end;
end.
```

Задача 3.5. Быстрая степень. Ввести вещественное число A и натуральное k. Вычислить и напечатать a^k с выполнением следующих условий: операцией возведения в степень пользоваться нельзя; k может оказаться настолько большим, что недопустимо выполнять k умножений.

При обычном вычислении a^k заводят переменную b, вначале равную единице, и многократно выполняют операторы:

k = k – 1:b = b ∗ a

Когда переменная k станет равной нулю, b станет равной искомой величине a^k. Идея сокращения вычислений в следующем. Вначале положим b = 1. Если k нечетно, по-прежнему выполняем операторы k = k – 1:b = b ∗ a.

Но если k четно, то воспользовавшись тождеством $a^k = (a^2)^{k/2}$, сделаем замены k = k/2:a = a ∗ a. Теперь, когда k станет равной нулю, переменная b станет равна искомой величине.

```
uses crt;
var a,b:double;
k,n:integer;
begin
write('A,K=');readln (a,k);
b:=1;
while k>0 do
begin
n:=k div 2;
if (n+n<k) then b:=b*a;
k:=n;a:=a*a;
end;
writeln(b:72:0);
readln;
end.
```

Задача 3.6. Упорядоченные дроби. Напечатать в порядке возрастания все простые несократимые дроби, заключенные между 0 и 1, знаменатели которых не превышают 7.

Невозможно обойтись без вспомогательных массивов и без проверок на сократимость дробей. Для этого заведем дробь m/n и положим сначала m = 0, n = 1. Отпечатаем m/n. Теперь среди всех дробей a/b, больших, чем m/n, и таких что b ≤ P (P = 7), выберем минимальную. Пусть это будет i/j. Если окажется, что i/j < 1, то заменим дробь m/n дробью i/j и продолжим этот процесс.

Для заданного знаменателя b = 2, 3, ..7 числитель a не подбирается, а вычисляется по формуле a = m ∗ b\n + 1. Каждая найденная дробь m/n автоматически будет несократимой. Это вытекает из того, что среди всех дробей, равных ей, она имеет наименьший знаменатель. Отметим, что при сравнении дробей лучше сравнивать не частные, а произведения (не a/b < i/j, но a ∗ j < b ∗ i).

```
uses crt;
var
```

```
p,n,m,i,j,a,b:integer;
begin
write('P=');readln (p);
m:=0;n:=1;
repeat
writeln(m:4,'/',n,'=',m/n:16:12);
i:=1;j:=1;
for b:=2 to p do
begin
a:=m*b div n+1;
if (a*j<b*i) then  begin i:=a;j:=b end
end;
m:=i;n:=j;
until i>=j;
readln;
end.
```

Задача 3.7. Бит-реверс. Целое положительное число m записывается в двоичной системе счисления и разряды в этой записи переставляются в обратном порядке. Получившееся число принимается за значение функции B (m). Напечатать значения B (m) для m = 512, 513, 514,…1023.

Начало распечатки: 1, 513, 257…

Выпишем несколько значений m и B(m) в десятичной и в двоичной системах.

Система	10	2	2	10
Значения m и B (m)	512	1000000000	0000000001	1
	513	1000000001	1000000001	513
	514	1000000010	0100000001	257
	515	1000000011	1100000001	769
	516	1000000100	0010000001	129
	……			
	1023	1111111111	1111111111	1023

По числу B (m) построим число B (m + 1). Глядя на двоичную запись значений B (m), можно понять, что при построении B (m + 1) нужно двигаться по двоичной записи числа B (m) слева направо, заменяя единицы нулями до первого нуля – его нужно заменить единицей (т.е. вычитая из B (m) числа 512, 256, 128, 64, 32 и т.д.).

```
uses crt;
label m1;
var
m,b,k,p:integer;
begin
k:=512;m:=1;p:=1;
clrscr;
writeln(m);
while m<1024 do
begin
p:=p+1;
if p>25 then exit;
while m>=k do begin m:=m-k;k:=k div 2 end;
m:=m+k;k:=512;
writeln(m);
end;
readln;
end.
```

3.7.1. Бит-реверс. Целое положительное число m записывается в двоичной системе счисления и разряды в этой записи переставляются в обратном порядке. Получившееся число принимается за значение функции B (m). Напечатать значения B (m) для любого m.

```
uses crt;
var
x,y:integer;
i,j,p:integer;
s,s1:string[20];
function step(x,k:integer):integer;
begin
if k=0 then step:=1;
if k>0 then step:=step(x,k-1)*x;
end;
begin
write('x=');readln(x);
for i:=0 to 15 do
if x and step(2,i)=step(2,i) then s:=s+'1' else s:=s+'0';
 p:=length(s);
 i:=p;
while s[p]<>'1' do begin p:=p-1;end;
 writeln(p);
for i:=1 to p do s1:=s1+s[i];
s:='';
for i:=p downto 1 do s:=s+s1[i];
write('x=',x,' ',s,' ',s1,'y=');
p:=length(s1); y:=0;
for i:=p  downto 1 do begin
val(s1[i],j,x);
y:=Y+j*step(2,p-i);
end;
writeln(y);
readln;
end.
```

Задача 3.8. Совершенные числа. Натуральное число называется совершенным, если оно равно сумме собственных делителей, включая единицу. Напечатать все совершенные числа, меньшие N.

```
uses crt;
var
N,n1,S:integer;
i,j:integer;
begin
write('N=');readln(N);
S:=0;
for j:=1 to N  do begin
S:=0;n1:=j;
for i:=1 to n1-1 do
begin
if N1 mod i=0 then s:=S+i;
end;
if S=N1 then writeln(N1);
end;
readln;end.
```

Задача 3.9. Ввести натуральные числа m и n и напечатать период десятичной дроби m/n. Например, для дроби 1/7 периодом будет (142857), а если дробь конечная, то ее период состоит из одной цифры 0.

Пояснения: При делении натурального числа M на натуральное число N получается целая часть частного и остаток. I = M div N:k = M – I ∗ N.

Для решения задачи сначала удалим из дроби M/N целую часть, заменив числитель M остатком

M: = M – M div N ∗ N.

Теперь мы можем последовательно получать цифры частного I и остатки M получившейся дроби.

I = 10 ∗ M\N:M = 10 ∗ M – i ∗ N.

Следовательно, различных остатков будет не более, чем N, и они начнут повторяться. А когда повторится остаток, начнут повторяться и частные, и начнется период. Для запоминания остатков можно завести массив D (1..N), вписать в него остатки в порядке получения и каждый новый остаток сравнить со всеми предыдущими. Это решение хотя и верное, но займет очень много времени при вычислениях. Поэтому попробуем обойтись без массива. Сначала так же выделим из M часть, кратную N. Затем пропустим N цифр частного. И теперь, когда период заведомо начался, запомним один единственный остаток, и будем печатать цифры частного, пока он не повторится.

```
uses crt;
var
M,N,i,j,k:integer;
begin
write('M,N=');readln(M,N);
m:=m-(m div n)*n;
k:=1;
while(k<=n) or (j<>m)do
begin
if k=n then j:=m;
i:=10*m div n;
m:=10*m-i*n;
if k>=n then write(i);
k:=k+1;
end;
writeln;readln;
end.
```

Задача 3.10. Распечатать числовую последовательность, которая задается по следующим правилам:

— первое число последовательности — произвольное нечетное число от 3 до 99;

— каждый следующий элемент последовательности определяется через предыдущий элемент p, и равен

3p + 1, если p — нечётное число,

p/2, если p — чётное число.

Например: вводим 9.

28 14 7 22 11 34 17 52 26 13 40 20 10 5 16 8 4 2 1

Вычисления прекратить, когда очередной элемент последовательности станет равен 1. (Известно, что в любой такой последовательности рано или поздно встречается 1).

```
uses crt;
var p,p1:integer;
begin
```

```
write('P=');readln(p);
p1:=p1;
while p1<>1 do
begin
if p mod 2 =1 then p1:=3*p+1 else p1:=p div 2;
write(p1,' ');
p:=p1;
end;
readln;
end.
```

Задача 3.11. Для заданного натурального числа определить, образуют ли его цифры арифметическую прогрессию. Предполагается, что в числе не менее трех цифр.

Например: 1357,963.

```
uses crt;
var
a,z,sh,aa:integer;
c:array[1..9] of integer;
m,m1,i:byte;
s:string;
begin
write('A'); readln(A);
str(a,s);
m:=length(s);writeln(m);m1:=m;writeln;
for i:=1 to m do begin  val(s[i],c[i],aa);write(c[i],' ');end;
z:=c[2]-c[1]; sh:=1;writeln; writeln('z=',z);
for i:=m downto 2 do if (c[i]-c[i-1]=z) then sh:=sh+1;
writeln;
writeln(sh);
if sh=m1 then writeln('yes') else write('No');
readln;
end.
```

Задание 3.12. Циклический сдвиг числа. Имеется десятичное число. Необходимо начальную цифру запомнить в ячейке и сдвинуть число влево на 1 разряд, первую цифру сделать последней. Это необходимо выполнить k раз (k вводится с клавиатуры и должно быть меньше или равно числу знаков числа).

53478
1-й сдвиг
34785
2-й сдвиг
47853
3-й сдвиг
78534
4-й сдвиг
85347
5-й сдвиг
53478

```
uses crt;
var
x,y:longint;
z:integer;
m,j,i:byte;
s:string;
ch:char;
begin
```

```
write('X='); readln(X);
str(X,s);
m:=length(s);
for i:=1 to m+1 do
begin
str(x,s);
ch:=s[1];
for j:=2 to m do begin s[j-1]:=s[j];end;s[j]:=ch;
val(s,x,z);
writeln(i,'й-сдвиг=',x);
end;
readln;
end.
```

Задача 3.13. Напишите программу, которая переводит числа римской нумерации в десятичные числа арабской нумерации.

Входной файл
I
II
XX
LX
Выходной файл
1 2 20 60

```
uses crt;
var s:string;
    n,c,c1,i,a:integer;
begin writeln('введите число:');
      readln(s);
      c:=0;n:=0;
      for i:=1 to length(s) do
        begin
            c1:=c;
            if s[i]='I' then c:=1;
            if s[i]='V' then c:=5;
            if s[i]='X' then c:=10;
            if s[i]='L' then c:=50;
            if s[i]='C' then c:=100;
            if s[i]='D' then c:=500;
            if s[i]='M' then c:=1000;
            if c>c1 then a:=-2*c1
                    else a:=0;
            n:=n+a+c
        end;
      writeln('ваше число=',n)
end.
```

Задача 3.14. Вычислить значение арифметического выражения:

$$R=\sqrt{98+\sqrt{95+\sqrt{92+...\sqrt{5+\sqrt{2}}}}}\,.$$

Вычисление непрерывных радикалов производится в цикле, начиная от внутреннего радикала. В данной задаче начальное значение $R=\sqrt{2}$. Каждое следующее значение радикала будет вычисляться через предыдущее значение радикала по формуле $R=\sqrt{a+R}$, число изменяется от начального значения 5 до конечного значения 98 с шагом 3.

Программа для вычисления *R* будет иметь вид:

```
var r,a:real;
begin
  r:=sqrt(2); a:=5;
  while a<=98 do
  begin
     r:=sqrt(a+r);
     a:=a+3;
  end;
  writeln('R=',r);
end.
```

Вычисленное по данной программе значение $R \approx 10.4044$.

Задача 3.15. Вычислить значение дроби:

$$Q = \cfrac{1}{1+\cfrac{1}{2+\cfrac{1}{3+\cfrac{1}{\ddots\ \cfrac{1}{98+\cfrac{1}{99+\cfrac{1}{100}}}}}}}$$

Вычисление непрерывных дробей производится снизу вверх, начиная от последней. Значение $r \approx 0,69777$.

```
uses crt;
var r,a:real;
begin
a:=100;r:=1/100;
while a>1 do
begin
r:=(a-1)+1/r;
a:=a-1;
end;
r:=1/r;
writeln('r=',r:10:6);
readln;end.
```

Задача 3.16. Напечатать все натуральные четырехзначные числа, в десятичной записи которых нет одинаковых цифр, и разность двух натуральных двузначных чисел, составленных из двух последовательных первых цифр и двух последовательных последних цифр числа, равна сумме всех цифр числа.

Любое целое четырехзначное число можно представить в виде: $N = a \cdot 1\,000 + b \cdot 100 + c \cdot 10 + d$ (*a, b, c, d* — цифры числа, причем $a \neq 0$). Например: 1742 = 1 ∗ 1000 + 7 ∗ 100 + 4 ∗ 10 + 2.

То, что цифры числа не должны совпадать, можно записать на Паскале в виде условия: *(a<>b)and(a<>c)and(a<>d)and(b<>c)and(b<>d)and(c<>d)*. Условие на разность чисел, составленных из цифр числа: $a * 10 + b - (c * 10 + d) = a + b + c + d$. Тогда выполняемая часть программы будет иметь вид:

```
for a:=1 to 9 do
for b:=0 to 9 do
for c:=0 to 9 do
for d:=0 to 9 do
```

```
if (a<>b)and(a<>c)and(a<>d)and(b<>c)and(b<>d)and(c<>d)and
(a*10+b-(c*10+d)=a+b+c+d)
then writeln(a*1000+b*100+c*10+d);
```

Задача 3.17. Уравнения в целых числах. Для любого целого числа N > 7 найти все такие пары целых чисел x и y, чтобы 3x + 5y = N.

Разделим N нацело на 5 и получим k — максимальное значение для y (т.е. 0 <= y <= k). Организуем цикл по переменной y и будем рассматривать значения разности N – 5y. Если это число делится нацело на 3, то полученное частное и есть соответствующее значение x. Соответствующая программа будет иметь вид:

```
var x,y,n,k:integer;
begin
  writeln('Введите N');
  readln(n);
  k:=n div 5;
  for y:=0 to k do
  if (N-5*y) mod 3=0 then
  begin
    x:=(N-5*y) div 3;
    writeln('x=',x,'  y=',y);
  end;
end.
```

Задача 3.18. Распечатать числовую последовательность, которая задается по следующим правилам:

— первое число последовательности — натуральное число, кратное 3 (входной параметр задачи);

— каждый последующий элемент равен сумме кубов цифр предыдущего.

Например:

33

33 + 33 = 54

53 + 43 = 189

13 + 83 + 93 = 1242

13 + 23 + 43 + 23 = 81

83 + 13 = 513

53 + 13 + 33 = 153

Вычисления прекратить, когда очередной элемент последовательности станет равен 153. (Известно, что любая такая последовательность рано или поздно приводит к 153).

На каждом шаге данного алгоритма приходится разбивать целое число на отдельные цифры (причем количество цифр в числе неизвестно). Это можно выполнить, используя операции целочисленной арифметики (деления нацело — div и остатка от деления — mod). Процесс вычисления очередного члена последовательности p через предыдущий в рассматриваемой задаче будет иметь следующий вид (s и p1 — рабочие переменные, t — очередная цифра числа): s: = 0; p1: = p;

```
while p1<>0 do
begin
   t:=p1 mod 10;
   p1:=p1 div 10;
   s:=s+t*t*t;
end;
p:=s;
```

Сама программа может быть реализована таким образом.

```
uses crt;
var s,t,p,p1:integer;
label m1;
begin
p:=33;
m1:
s:=0;p1:=p;
while p1<>0 do
begin
t:=p1 mod 10;;
p1:=p1 div 10;
s:=s+t*t*t;
end;
p:=s;
writeln(s);
if p<>153 then goto m1;
readln;end.
```

Задача 3.19. В трехзначном числе зачеркнули старшую цифру. Когда полученное число умножили на 5, то получили исходное число. Найти эти числа.

```
uses crt;
var
x,y,i:integer;
begin
clrscr;
for i:=100 to 999 do begin
x:=i;
y:=x mod 100; {Отбрасываем старшую цифру}
if (y*5=x) then writeln(x);
end;
readln;
end.
```

Задача 3.20. Вычислите сумму всех чисел, на которые число N делится без остатка. Единица и само число также являются делителями (N <= 32000).

```
uses crt;
var N,i:integer;
S:longint;
begin
writeln('N');
readln(n);
for i:=1 to N do
if N mod i=0 then begin s:=s+i;write(i,' ');end;
writeln;
writeln('S=',S);
readln;
end.
```

Часть 4
СТРОКИ. ПРЕОБРАЗОВАНИЕ ТИПОВ

Задача 4.1. Для заданной строки символов проверить, является ли она симметричной или нет. (Симметричной считается строка, которая одинаково читается слева направо и справа налево).

Проще всего в этой задаче определить длину строки n, организовать цикл по номеру символа в строке и сравнивать попарно первый символ с последним, второй – с предпоследним и т.д. Если хотя бы одна пара различна, то строка не симметричная. Так как просматривается сразу пара символов, то в цикле будет m = n div 2 повторений. Для запоминания результата просмотра введем переменную k (k будет равна 0, если строка симметрична, иначе — 1). Программа, решающая эту задачу, будет иметь вид:

```
var   s:string;
i,k,n,m:integer;
begin
  readln(s);
  n:=length(s);
  k:=0;
  m:=n div 2;
  for i:=1 to m do
  if s[i]<>s[n-i+1] then k:=1;
  if k=0 then writeln('Строка симметрична')
  else writeln('Строка несимметрична');
end.
```

Задача 4.2. Для заданной строки символов определить сумму всех входящих в нее цифр.

Так как все цифры от 0 до 9 располагаются в таблице кодировки компьютера подряд, то проще всего проверить, является ли символ s [i] цифрой, можно с помощью неравенства '0' <= s[i] <= '9' (на Паскале (s[i] >= '0')and(s[i] <= '9')). Для преобразования строки в число в Паскале используется процедура val(s,v,k), где s — строка (или символ), v — переменная числового типа, куда будет занесен результат преобразования, k — переменная целочисленного типа, которая принимает значение 0, если преобразование строки в число прошло успешно.

```
uses crt;
var
st:string;
i:byte;
s,k,tip:integer;
begin
writeln('stroka?');
readln(st);
s:=0;
for i:=1 to length(st) do
if (st[i]>='0') and(st[i]<='9') then
begin
val(st[i],k,tip);
s:=s+k;
end;
```

```
writeln('s=',s);
readln;
end.
```

Задача 4.3. Для заданной строки символов вычислить произведение входящих в эту строку целых чисел (без учета их знаков). Например, для строки "kjjjkkj2.5jkjn,,,hfd45jgfvjlkfdii10,2hfhg" произведение 2 ∗ 5 ∗ 45 ∗ 10 ∗ 2 = 9000.

Пусть s — строка. Обозначим длину строки l. Организуем цикл, в котором будем проверять, является ли очередной символ цифрой, если да, то организуем новый цикл, в котором будем формировать строку sn, состоящую из цифр (очередное целое число). Потом преобразуем sn в число и вычислим произведение. Программа на Паскале, реализующая данный алгоритм, будет иметь следующий вид (переменная p в этой программе используется для накапливания значения произведения, переменная kod — для хранения кода результата преобразования строки в число):

```
var sn,s:string;
    l,k,kod:integer;
    v,p:real;
begin
 writeln('Введите строку');
 readln(s);
 l:=length(s);
 p:=1; k:=1;
 repeat
  sn:='';
  while (s[k]>='0')and(s[k]<='9')and(k<=l) do
   begin
    sn:=sn+s[k];
    k:=k+1;
   end;
   if sn<>'' then
   begin
    val(sn,v,kod);
    p:=p*v;
   end;
   k:=k+1;
   until k>l;
   writeln(' p=',p);end.
```

Задача 4.4. Задана строка символов. Среди литер этого текста особую роль играет знак #, появление которого означает отмену предыдущей литеры текста; k знаков # отменяют k предыдущих литер (если такие есть). Напечатать строку с учетом роли знака #. Например, строка "VR#Y##HELO#LO" должна быть напечатана в виде: "HELLO".

Обозначим исходную строку — s, длину этой строки — l. Для решения создадим еще одну строку s1 (вначале пустую). Далее организуем цикл по номеру символа в строке s. Если очередной символ не #, то добавим его к строке s1, если это знак # и строка s1 не пустая, то удалим из нее последний символ.

```
var s,s1:string;
  dl,i,k:integer;
begin
  writeln('Введите строку');
  readln(s);
  dl:=length(s);
  s1:='';
  k:=0;
```

```
  for i:=1 to dl do
  if (s[i]='#')and(k<>0) then
   begin
    delete(s1,k,1);
    k:=k-1;
   end
  else
    begin
     k:=k+1;
     s1:=s1+s[i];
    end;
 writeln(s1);
end.
```

Задача 4.5. Для заданной строки символов, состоящей из строчных букв и пробелов, определить слово наибольшей длины, которое начинается и заканчивается на одну и ту же букву. Например: строка — "программа на языке Паскаль", слово — "программа".

Пусть s — заданная строка. Для решения данной задачи определим длину строки l и организуем цикл по номеру символа i. Символ строки является первым символом некоторого слова в том случае, когда он сам не является пробелом, и либо он — первый символ строки, либо слева от него стоит пробел. Если мы нашли первый символ некоторого слова, то запомним его номер и организуем цикл, в котором найдем номер последнего символа этого слова (символ будет последним в слове либо тогда, когда после него стоит пробел, либо когда он последний символ строки). Если последний символ слова совпадает с первым символом этого слова, и длина слова наибольшая из всех найденных, запомним эту длину и номер первого символа этого слова. В приведенной программе введены следующие обозначения: max — переменная, в которой запоминается текущая максимальная длина найденного слова; k — переменная, в которой поочередно запоминается номер первого символа каждого слова; koord — переменная, в которой хранится номер первого символа слова с максимальной длиной.

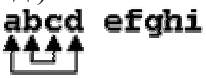
```
var s:string;
    koord,k,i,n,max:integer;
    fst:char;
begin
   writeln('Введите строку');
   readln(s);
   n:=length(s);
   max:=0; i:=1;
   while i<=n do
   if (s[i]<>' ')and((i=1)or(s[i-1]=' ')) then
   begin
       k:=i;
       while (i<n)and(s[i+1]<>' ') do  i:=i+1;
       if (s[i]=s[k])and(i-k+1>max) then
       begin
         koord:=k;
         max:=i-k+1;
         i:=i+1;
       end;
   end
   else i:=i+1;
  if max<>0 then
  begin
     writeln(' max=',max);
     for i:=0 to max-1 do write(s[koord+i]);
```

```
  end
  else writeln('Такого слова нет')
end.
```

Задача 4.6. Задана строка символов, содержащая два или более слова, разделенных пробелами. Написать программу, меняющую местами все четные и нечетные слова в строке, предполагая, что за один раз можно менять местами не более двух символов.

Имеется несколько путей решения этой задачи. Для упрощения предположим, что строка не начинается и не заканчивается пробелом и что между словами в строке стоит ровно по одному пробелу. Пусть известна пара слов, которую необходимо переставить, номера первой и последней букв в первом слове и номера первой и последней букв во втором слове. Рассмотрим способ, в котором для перестановки слов будем использовать следующий алгоритм:

Запишем буквы первого слова в обратном порядке (поменяв первую с последней, вторую с предпоследней и т.д.).

Например, из строки 

получим dcba efghi.

Потом аналогичным образом переставим буквы второго слова:

из строки 

получим dcba ihgfe.

Для получения окончательного результата необходимо записать буквы полученного словосочетания в обратном порядке:

Из строки 

получим efghi abcd (что и требовалось получить).

Таким образом, для перестановки двух слов достаточно написать подпрограмму, которая меняет в заданной строке порядок букв на противоположный (инвертирует строку), и вызвать эту подпрограмму для первого слова, второго слова и всего словосочетания.

Обозначим процедуру invert (k, l), которая записывает в заданной строке s символы с k-того по l-й в обратном порядке, тогда программа, решающая задачу, будет иметь вид:

```
var s:string;
    i,n,m1,m2,l1,l2:byte;
procedure invert(k,l:byte);
var i:byte;
    ch:char;
begin
   for i:=k to ((l+k) div 2) do
   begin
      ch:=s[i];
      s[i]:=s[l+k-i];
      s[l+k-i]:=ch;
   end;
end;
begin
writeln('Введите строку'); readln(s);
i:=0; n:=0;
m1:=1;m2:=1;l1:=1;l2:=1;
while i<length(s) do
begin
  i:=i+1;
  if (s[i]=' ')or(i=length(s)) then
  begin
    n:=n+1;
    if n=1 then
    begin
```

```
      m2:=i-1;
      l1:=i+1
    end
    else
    begin
      n:=0;
      if i=length(s) then l2:=i else l2:=i-1;
      invert(m1,m2);invert(l1,l2);invert(m1,l2);
      m1:=i+1
    end
  end
end;
writeln(s)
end.
```

Задача 4.7. Текст задан последовательностью букв (кириллица строчная). Написать программу, которая определяет, выполнено ли условие, что буквы упорядочены по алфавиту.

```
uses crt;
var st:string;
i,j,p:byte;
sh:char;
begin
writeln('Введите последовательность букв');
readln(st);i:=1;
for sh:='a' to 'я' do begin
while i<=length(st)-1 do begin
if copy(st,i+1,1)<=sh then  begin writeln('Нет');readln;exit;end;
i:=i+1;
end;
end;
writeln('Да');
readln; end.
```

Часть 5
МНОГОЗНАЧНАЯ АРИФМЕТИКА

Задача 5.1. Вывести на экран цифры числа 3^{1000}. Если попытаться получить число непосредственно умножением, компьютер выдаст сообщение об ошибке.

Для записи больших чисел удобно использовать массивы, записывая цифры числа как элементы массива. Оценим количество цифр, необходимых для записи 3^{1000} : $3^2 = 9$; $3^{1000} = (3^2)^{500} = 9^{500} \approx 10^{500}$.

Таким образом, в записи этого числа будет менее 500 цифр. Запишем вначале в массив только число 3 и далее будем умножать его поэлементно на 3 нужное число раз, учитывая перенос из разряда в разряд, возникающий при умножении. Программа, решающая эту задачу, может быть записана, например, так:

```
Uses crt;
const stp=1000;
var i,j,k,prn,x:integer;
    a:array [1..500] of integer;
begin
     a[500]:=3; prn:=0;
     for i:=2 to stp do
     for j:=500 downto 1 do
     begin
          x:=a[j]*3;
          a[j]:=(x+prn) mod 10;
          prn:=(x+prn) div 10;
     end;
     k:=1; while (a[k]=0) do k:=k+1;Подсчет пустых(нулей) элементов массива
     for i:=k to 500 do write(a[i]:1);
     writeln
end.
```

Задача 5.2. Написать программу сложения двух целых 100-значных чисел A и B из диапазона $0..10^{100}$.

Пояснение: Числа A и B нужно записать в текстовом формате, так как другое представление чисел невозможно.

```
 uses crt;
var
st1,st2,st3,st4:string[100];{переменные для чисел в строковом формате}
i,j,m,r1,r2,r3,r4,r5:byte; {вспомогательные переменные}
kl:integer;
begin
clrscr;
{чтение чисел из выходного файла}
writeln('A,B');
readln(st1);
readln(st2);
{выравнивание длины строк справа налево}
if length(st1)>  length(st2) then for i:=1 to length(st1)-length(st2) do
st2:='0'+st2;
if length(st1)<  length(st2) then for i:=1 to length(st2)-length(st1) do
st1:='0'+st1;
```

```
m:=length(st1);
{Выполнение сложения поразрядно с учетом перехода через десяток}
for i:=m downto 1 do begin
writeln(st1[i],' ',st2[i]);
val(st1[i],r1,kl);
val(st2[i],r2,kl);
r3:=(r1+r2+r5);
if r3>=10 then begin r4:=(r1+r2+r5) mod 10; r5:=1;end;
if r3<10  then begin r4:=(r1+r2+r5) mod 10; r5:=0;end;
str(r4,st4);
st3:=st4+st3;
end;
{сохранение старшего разряда}
if r5=1 then st3:='1'+st3;
{Запись в выходной файл и на экран}
{assign(f,'vyx.txt');
rewrite(f);
write(f,st3);
close(f);}
writeln(st3);
readln;
end.
```

Задача 5.3. Написать программу сложения двух целых чисел A и B из диапазона 0..200000000 ($2 * 10^9$).

Пример входного файла
Например

999999999
899999999

Выходной файл имеет структуру

ЧИСЛО В ТЕКСТОВОМ ФОРМАТЕ

Для нашего примера

1000000088

Аналогично предыдущей задаче.

```
uses crt;
var s1,s2:longint;{переменные для целых чисел указанного интервала}
f:text; {файловая переменная для входного и выходного файла}
st1,st2,st3,st4:string[10];{переменные для преобразования чисел в строки}
i,j,m,r1,r2,r3,r4,r5:byte; {вспомогательные переменные}
kl:integer;
begin
clrscr;
{чтение чисел из выходного файла}
assign(f,'vx.txt');{устройство ввода и вывода необходимо корректировать}
reset(f);
read(f,s1);
read(f,s2);
close(f);
{преобразование чисел в строки}
str(s1,st1);
str(s2,st2);
{выравнивание длины строк справа налево}
```

```
if length(st1)>  length(st2) then for i:=1 to length(st1)-length(st2) do
st2:='0'+st2;
if length(st1)<  length(st2) then for i:=1 to length(st2)-length(st1) do
st1:='0'+st1;

m:=length(st1);
{Выполнение сложения побайтно с учетом перехода через десяток}
for i:=m downto 1 do begin
writeln(st1[i],' ',st2[i]);
val(st1[i],r1,kl);
val(st2[i],r2,kl);
r3:=(r1+r2+r5);
if r3>=10 then begin r4:=(r1+r2+r5) mod 10; r5:=1;end;
if r3<10  then begin r4:=(r1+r2+r5) mod 10; r5:=0;end;
str(r4,st4);
st3:=st4+st3;
end;
{сохранение старшего разряда}
if r5=1 then st3:='1'+st3;
{Запись в выходной файл и на экран}
assign(f,'vyx.txt');
rewrite(f);
write(f,st3);
close(f);
writeln(st3);
readln;
end.
```

Задача 5.4. Написать программу нахождения наибольшего среди трех чисел. Причем числа принимают значения от 1 до 10^{100}.

Пояснение. Такие числа можно представить только в виде строк. Следовательно, вычисление максимума сводится к нахождению числа (строки) максимальной длины. В случае, когда количество знаков во всех числах одинаково — к сравнению разрядов, начиная со старшего разряда.

```
uses crt;
var
A,B,C:string[100];
l1,l2,l3:byte;
x,y,z:array[1..100] of byte;
k,i:integer;
label m;
begin
writeln('A');readln(A);
writeln('B');readln(B);
writeln('C');readln(C);
if (a=b) and(a=c) then begin writeln(a);goto m;end;
l1:=length(A);
l2:=length(B);
l3:=length(C);
if (l1>l2) and (l1>l3) then write(A);
if (l2>l1) and (l2>l3) then write(B);
if (l3>l1) and (l3>l2) then write(C);
if (l3=l1) and (l3=l2) then
begin
for i:=1 to l1 do begin
val(a[i],x[i],k);
val(b[i],y[i],k);
val(c[i],z[i],k);
```

```
end;
for i:=1 to l1 do begin
if (x[i]>y[i]) and(x[i]>z[i]) then begin write(A);goto m;end;
if (y[i]>x[i]) and(y[i]>z[i]) then begin write(B);goto m;end;
if (Z[i]>y[i]) and(Z[i]>x[i]) then  begin write(C);goto m;end;
end;
end;
m:
readln;
end.
```

Задача 5.5. Натуральное число X возводится в степень Y, причем Y > 5000. Определить последнюю цифру результата.

```
uses crt;
var
x:integer;
p:integer;
st,i:integer;
begin
writeln('число X');
readln(x);
writeln('Показатель степени>5000');
readln(st);
x:=x mod 10;{определение последней цифры}
p:=x;
for i:=1 to st-1 do
begin
x:=x*p;
x:=x mod 10;
end;
writeln('x=',x);
readln;
end.
```

Задача 5.6. Написать программу умножения двух 100-значных чисел.

Суть решения задачи нам уже известна из рассмотренных выше задач. Числа вводятся как строки и записываются в числовые массивы A, B. Выполняется поразрядное умножение чисел и результаты записываются в матрицу rez(100, 100). Далее выполняется сложение со сдвигом каждой строки влево на одну позицию. Сумма хранится в массиве Sum(200). Можно обойтись без матрицы. Оптимизируйте предложенную программу, а именно — часть, где складываются суммы произведений.

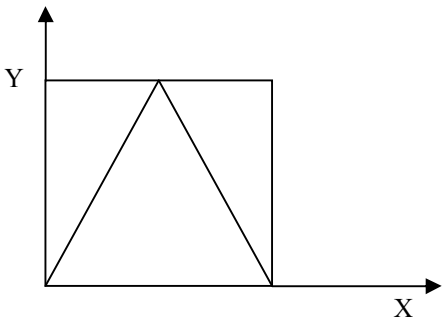
Основные подходы отображены в предложенной ниже программе.

```
uses crt;
var rez:array[1..100,0..100] of byte;
sum:array[1..200] of byte;
a,b:array[1..100] of byte;
s1,s2:string[100];
p,l1,l2,i,j:byte;
cel,ost,pr:byte;
mem:byte;
k:integer;
LOG:boolean;
begin
clrscr;
{Ввод чисел в виде строк}
writeln('a');readln(s2);
writeln('b');readln(s1);
```

```
l1:=length(s1);
l2:=length(s2);
{Запись чисел в массивы}
for i:=l2 downto 1 do  begin val(s2[i],b[i],k);end;
for i:=l1 downto 1 do  begin val(s1[i],a[i],k);end;
{отображение чисел в массивах}
for i:=1 to l2 do write(b[i]);writeln;
for i:=1 to l1 do write(a[i]);
{Умножение чисел}
readln; p:=0;
for i:=l1 downto 1 do  begin
pr:=0;cel:=0;ost:=0;
for j:=l2 downto 1 do begin
pr:=a[i]*b[j];
cel:=pr div 10;
ost:=pr mod 10;
rez[i,j-1]:=cel;
rez[i,j]:=rez[i,j]+ost;
if rez[i,j]>=10 then begin rez[i,j-1]:=rez[i,j-1]+rez[i,j] div 10;
rez[i,j]:=rez[i,j] mod 10;end;
writeln(pr,' ',cel,' ',ost,' ',rez[i,j]);
end;
Write(rez[i,j-1]);
writeln;
end;
{Суммирование произведений}
k:=200;p:=0;
for i:=l1 downto 1 do  begin
for j:=l2 downto 0  do begin sum[k]:=sum[k]+rez[i,j]+mem;
log:=false; MEM:=0;
if (sum[k]>=10) then begin
mem:=(sum[k] div 10);sum[k-1]:=mem;sum[k]:=(sum[k] mod 10); log:=true;end;
{write(rez[i,j]);}
k:=k-1;
end;
writeln;
p:=p+1;
k:=200-p;
end;
{Уточните вывод. Подсчитайте незначащие нули и поменяйте параметры цикла}
p:=4*p;
for i:=200-l1-l2 to 200 do write(sum[i]);
readln;
end.
```

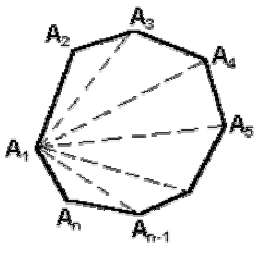
Часть 6
ГЕОМЕТРИЧЕСКИЕ ЗАДАЧИ

Задача 6.1. В квадрат со стороной 50 вписан равнобедренный треугольник (см. рисунок). По квадрату ведется стрельба. Если стрелок попадает в контур треугольника или в точку вне треугольника, то это считается промахом. Определить количество промахов и удач N = 100, 1000. (N — общее количество испытаний, вводится с клавиатуры)




```
uses crt,graph;
var
x,y,x1,x2,y1,y2:integer;
kl,kl1:longint;
dr,dm,i:integer;
begin
kl:=0;
dr:=9;
dm:=1;
initgraph(dr,dm,'c:\tp\bgi');
setcolor(15);
setbkcolor(12);
cleardevice;
rectangle(0,50,50,100);
for i:=1 to 10000 do
begin
x:=random(50);
y:=random(50);
if (x<0.5*y)or (x>-0.5*y+50) or (y=0)  then begin Putpixel(round(x),100-
round(y),14);kl:=kl+1;end;
{if (2*x-100>25 ) and( -2*x+100<25) then begin Putpixel(round(x),100-
round(y),4);kl1:=kl1+1;end;}
end;readln;closegraph;
writeln(kl,'  ',kl1);
readln;end.
```

Задача 6.2. Выпуклый многоугольник задан последовательностью координат своих вершин в порядке обхода: (x1, y1), (x2, y2), (x3, y3), . . . , (xn, yn). Вычислить площадь многоугольника.



Стандартный способ вычисления площади выпуклого многоугольника — разбиение исходного многоугольника на отдельные треугольники (см. рисунок) с последующим вычислением площадей полученных треугольников и их суммированием. Площадь отдельного

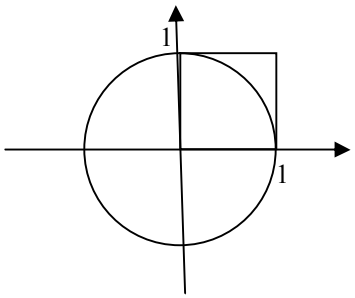
треугольника можно вычислить, например, по формуле Герона, но в данном случае более удобной будет формула расчета площади треугольника по координатам его вершин:

$$S = \frac{1}{2}\left|(x_2 - x_1)(y_3 - y_1) - (x_3 - x_1)(y_2 - y_1)\right|.$$

Пусть n — число вершин, X(n), Y(n) — массивы, содержащие координаты вершин, тогда основная часть программы для вычисления площади многоугольника будет иметь вид:

```
s:=0;
for i:=3 to n do
s:=s+0.5*abs((x[i-1]-x[1])*(y[i]-y[1])-
             (x[i]-x[1])*(y[i-1]-y[1]));
writeln('Площадь многоугольника s=',s);
```

Задача 6.3. По четверти круга единичной окружности ведется стрельба. Определить значение числа PI.



Будем стрелять по квадрату со стороной единица.

Площадь квадрата S = 1.Площадь четверти круга S1 = pi/4.

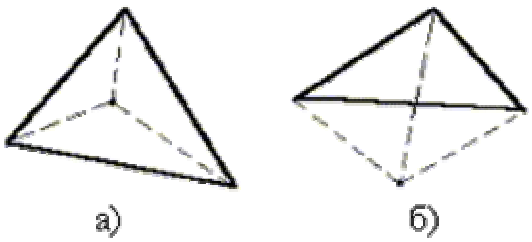
Общее число стрельб обозначим через N,а количество удач, т.е. попаданий в четверть круга — M. По методу Монте-Карло имеем соотношение S/S1 = N/M. Подставив известные значения(Pi считаем неизвестным), имеем:

1/pi/4 = N/M. Отсюда pi = 4 ∗ M/N.

```
uses crt;
var ppi,f,x,y,r:real;
n,m:integer;

begin
m:=0;
for n:=1 to 400 do
begin
x:=random;
y:=random;
if x*x+y*y<=1 then m:=m+1;
 end;
 ppi:=4*m/n;
 writeln(ppi:10:8);
 readln;
 end.
```

Задача 6.4. Задана точка с координатами (x, y) и треугольник с координатами вершин (x1, y1), (x2, y2), (x3, y3). Определить, лежит точка внутри или вне треугольника. Чтобы определить, лежит ли точка внутри треугольника, можно соединить эту точку отрезками с его вершинами и рассчитать площади получившихся треугольников (как в задаче 6.2). Если сумма вычисленных площадей равна площади исходной фигуры (рис. а), то точка лежит внутри, если нет (рис. б) — снаружи.

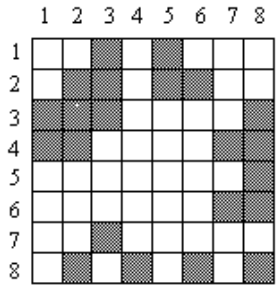


Задача 6.5. Диаметр колеса автомобиля 80 см. Колесо потребует замены через 200 000 оборотов. Определить, доедет ли колесо от города Нижневартовска до города N-ска, если расстояние между ними x километров.

Решение задачи сводится к определению длины окружности. Умножив длину на количество оборотов и поделив произведение на 1000, мы найдем пригодность колеса в километрах.

```
uses crt;
var s,l,dist:real;
const d=0.8;
begin
write('Расстояние до города в километрах');readln(dist);
l:=(0.8*pi*200000)/1000; {сколько  километров можно ехать на колесе}
if dist<=l then write('Можно доехать') else write('Возможно, будут
проблемы');
readln;
end.
```

Задача 6.6. На квадратном листе бумаги в клетку размером 8×8 клеток заштрихована часть клеток (пример на рисунке). Определить вписанный в решетку прямоугольник максимальной площади, не содержащий заштрихованных клеток. В качестве ответа вывести площадь прямоугольника и координаты его двух противоположных вершин. (Предполагается, что прямоугольник с максимальной площадью один).



Для приведенного примера координаты вершин (3, 4) и (7, 6), площадь 15 клеток. Представим лист бумаги в виде числовой матрицы A (8×8). Обозначим заштрихованные клетки единицами, а чистые — нулями. Данную программу удобно реализовать, используя подпрограммы. Напишем подпрограмму, которая проверяет, есть ли в прямоугольнике с координатами левой верхней вершины (i1, j1) и координатами правой нижней вершины (i2, j2) заштрихованные клетки (т.е. элементы, равные 1).

```
function prov(a:matr;i1,j1,i2,j2:integer):boolean;
var i,j:integer;
begin
  prov:=true;
  for i:=i1 to i2 do
  for j:=j1 to j2 do
  if a[i,j]=1 then prov:=false;
end;
```

Эта функция будет возвращать значение "истина" (true), если заштрихованных клеток в рассматриваемом прямоугольнике нет, и "ложь" (false) — в противном случае. В основной программе организуем два попарно вложенных цикла: в первом цикле будут изменяться координаты верхнего левого угла рассматриваемого прямоугольника, во втором — координаты нижнего правого угла. Если в прямоугольнике нет заштрихованных точек, то вычислим его площадь, и если она является максимальной — запомним ее и координаты противоположных вершин этого прямоугольника. Часть основной программы, реализующая данный алгоритм, будет иметь следующий вид:

```
maxs:=0;
 for i1:=1 to n do
 for j1:=1 to n do
   for i2:=i1 to n do
   for j2:=j1 to n do
```

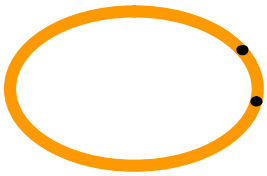
```
   begin
     s:=(i2-i1+1)*(j2-j1+1);
     if prov(a,i1,j1,i2,j2)and(maxs<s) then
     begin
       maxs:=s;
       maxi1:=i1;
       maxj1:=j1;
       maxi2:=i2;
       maxj2:=j2;
     end;
   end;
writeln(' s=',maxs);
writeln('(',maxi1,',',maxj1,')',     '(',maxi2,',',maxj2,')');
```

Здесь maxs — площадь полученного прямоугольника, (maxi1, maxj1) — координаты его левой верхней вершины, (maxi2, maxj2) — координаты его правой нижней вершины.

Задача 6.7. Вычислить интеграл функции $y = x^2$ в интервале от 0 до 1 методом Монте-Карло.

Пояснения к методу Монте-Карло даны в задаче 6.3.

```
{Метод стрельбы}
uses crt;
const NN=10000;
var x,y,xx,yy: real;
    n,i: integer;
function Funct(x:real):real;
begin Funct:=x*x; end;
BEGIN  {Основная программа}
  clrscr; Randomize; n:=0;
  for i:=1 to NN do
    begin
       x:=Random(1000)/1000;
       yy:=Random(1000)/1000;
       if yy<Funct(x) then n:=n+1;
    end;
  writeln('Интеграл равен ',n/NN);
  Repeat until KeyPressed;
END.
```



Задача 6.8. Два тела с разной скоростью совершают движение по одной и той же эллиптической орбите. Когда тела окажутся в одной точке, необходимо завершить работу программы.

Для программирования этой задачи просто необходимо напомнить уравнение эллипса.

X = X0 + R1 ∗ cos(f)

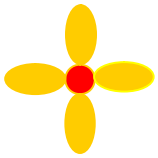
Y = Y0 + R2 ∗ sin(f), где f = 0..2pi. R1, R2 — полуоси эллипса.

Программа может быть реализована следующим образом.

```
uses crt, graph,dos;
var
x,y,x0,y0,xx,yy,xxx,yyy:integer;
h,h1,m,m1,s,s1,ms:word;
lr,lr1:longint;
c:byte;
f,f1:real;
dr,dm:integer;
r,r1,t:integer;
```

```
begin
dr:=detect;
initgraph(dr,dm,'');
f:=0;f1:=0;
gettime(h,m,s,ms);
lr:=h*3600+m*60+s;
x0:=getmaxx div 2;
y0:=getmaxy div 2;
while f<=2*pi do
begin
x:=round(x0+200 *cos(f));
y:=round(y0+200 *sin(f));
setcolor(14);
circle(x,y,4);
gettime(h1,m1,s1,ms);
lr1:=h1*3600+m1*60+s1;
if lr1-lr>=3.5 then
begin
xx:=round(x0+200 *cos(f1));
yy:=round(y0+200 *sin(f1));
setcolor(15);
circle(xx,yy,4);
f1:=f1+0.0001*1.85;
end;
f:=f+0.0001;
if ((xx=x) and (yy=y)) then begin outtext('Догнал');readln; halt; end;
end;
readln;
end.
```

Задача 6.9. Заполнить экран следующим узором.



```
uses crt,graph;
var
dr,dm:integer;
p:pointer;
size:word;
i,j:integer;
begin
dr:=detect;
dm:=0;
initgraph(dr,dm,'');
setfillstyle(1,12);
circle(50,55,10);
floodfill(50,55,15);
setfillstyle(1,14);
fillellipse(50,20,10,20);
fillellipse(50,90,10,20);
fillellipse(20,50,20,10);
fillellipse(80,50,20,10);
size:=imagesize(1,1,100,120);
readln;
getmem(p,size);
getimage(1,1,100,120,P^);
cleardevice;
setbkcolor(7);
repeat
```

```
{cleardevice;}
i:=random(640);
j:=random(480);
putimage(i,j,P^,0);
delay(1000);
until keypressed;
readln;
end.
```

Задача 6.10. Два квадрата с одинаковыми сторонами блуждают по графическому экрану. При касании сторон, при пересечении квадратов программа прекращает работу. Попробуйте программировать.

Ниже приводится возможный вариант программы.

```
Program Rect;
 uses Crt,Graph;
const dim = 35;{сторона квадрата в пикселях }
var
  grDriver: Integer;
  grMode: Integer;
  ErrCode: Integer;
  x,y,x1,y1:integer;
  Stop:boolean;
begin
  Stop:=False;
  randomize;
  grDriver := Detect;
  InitGraph(grDriver, grMode,'c:\tp\bgi');
  ErrCode := GraphResult;

  if ErrCode = grOk then
  begin  { Do graphics }
  repeat
  x:=random(340-dim);
  y:=random(480-dim);
  x1:=random(300+dim);
  y1:=random(480-dim);
  rectangle(x,y,x+dim,y+dim);
  rectangle(x1,y1,x1+dim,y1+dim);
  if (x<=x1) and (x1<=x+dim) and (y<=y1) and (y1<=y+dim) then Stop:=True;
  if (x1<=x) and (x<=x1+dim) and (y1<=y) and (y<=y1+dim) then Stop:=True;
  if (x<=x1) and (x1<=x+dim) and (y1<=y) and (y<=y1+dim) then Stop:=True;
  if (x1<=x) and (x<=x1+dim) and (y<=y1) and (y1<=y+dim) then Stop:=True;
  if stop=true then begin outtextxy(50,50,'Пересеклись');readkey;end;
  delay(5000);
  cleardevice;
  until stop=true;
   CloseGraph;
  end
  else
    Writeln('Ошибка графики:', GraphErrorMsg(ErrCode));
end.
```

Часть 7
РАЗНЫЕ ЗАДАЧИ

Задача 7.1. Из одного порта в другой необходимо перевезти 15 различных грузов. Грузоподъемность судна, на котором будет проходить перевозка, 50 тонн. Грузы пронумерованы, и информация о массах грузов хранится в массиве M(15). Определить, сколько рейсов необходимо сделать судну, если грузы неделимы и могут перевозиться только подряд в порядке их нумерации. (Предполагается, что масса отдельного груза не превышает 50 тонн).

Обозначим: k — номер рейса судна, i — номер очередного груза, s — масса груза на судне в k-том рейсе. Решать задачу будем так: если на судно в k-том рейсе можно поместить еще один груз, то мы грузим его и берем следующий, если груз не может быть размещен, то перевозим его следующим рейсом (увеличиваем k).

Соответствующая программа может иметь вид:

```
uses crt;
var
m:array[1..15] of byte;
k,i,s:byte;
begin
{m[1]:=50;m[2]:=25; m[3]:=25;m[4]:=15;m[5]:=35;}
randomize;
for i:=1 to 15 do begin m[i] :=random(50)+1;
write(m[i]:4);end;
k:=1; i:=1; s:=0;
repeat
    if s+m[i]<=50 then
    begin
      s:=s+m[i];
      i:=i+1;
    end
    else
    begin
      k:=k+1;
      s:=0;
    end;
until i>15;
writeln;
writeln('Всего потребовалось', k,' рейсов');
readln;
end.
```

Задача 7.2. Жизнь роботов. Сообщество роботов живет по следующим законам: один раз в год они объединяются в полностью укомплектованные группы по 3 или 5 роботов (причем число групп из 3 роботов — максимально возможное). За год группа из 3 роботов собирает 5, а группа из 5—9 новых собратьев. Каждый робот живет 3 года после сборки. Известно начальное количество роботов (K > 7), все они только что собраны. Определить, сколько роботов будет через N лет.

Рассмотрим следующий вариант решения. Создадим массив R(3), где R_1, R_2, R_3 — количество роботов соответствующего возраста. Тогда общее количество роботов $S = R_1 + R_2 + R_3$. Обозначим *x* — количество троек, *y* — количество пятерок, которое можно сформировать

из общего числа роботов (идея разбиения на тройки и пятерки рассмотрена в задаче 1.5). Каждый год роботы стареют, общее количество роботов увеличивается на $5x + 9y$ и уменьшается на R_3 (число роботов, проживших 3 года). Программа решения может быть записана так:

```
var k,i,n,p:integer;
    s,x,y:longint;
    r:array [1..3] of longint;
begin
  write('Начальное количество роботов k='); readln(k);
  write('Число лет n='); readln(n);
  r[1]:=k; r[2]:=0; r[3]:=0; s:=k;
  for i:=1 to n do
  begin
      x:=s div 3;
      p:=s mod 3;
      if p=0 then y:=0
      else if p=1 then begin x:=x-3; y:=2 end
      else begin x:=x-1; y:=1 end;
      r[3]:=r[2]; r[2]:=r[1]; r[1]:=5*x+9*y;
      s:=r[1]+r[2]+r[3];
  end;
  writeln('s=',s)
end.
```

В этой программе использовался тип **longint**, предназначенный для хранения больших целых чисел (до 2147483647). Это связано с тем, что общее число роботов увеличивается очень быстро. Так, например, если начальное число роботов — 10, то через 10 лет их будет 143702.

Задача 7.3. Функция f(n) для целых неотрицательных n определена так:

$$f(0) = 0,\ f(1) = 1,\ f(2n) = f(n),\ f(2n + 1) = f(n) + f(n + 1).$$

Для данного N найти и напечатать f(N). (N столь велико, что недопустимо заводить массив из N чисел).

Попытка решения обычным способом приведет к многозначности функции. Поэтому, наряду с заданной функцией f(n) одного аргумента введем функцию g(n, i, j) = i ∗ f(n) + j ∗ f(n + 1). Для этой функции проверяются формулы: g(2n, i, j) = g(n, I + j, j), g(2n + 1, i, j)=g(n, i, I + j).

Тогда f(n) = g(n,1,0), а при многократном применении рекуррентных формул получим

$$F(n) = g(n, 1, 0) = \ldots = g(0, i, j) = j.$$

```
uses crt;
var n,i,j:longint;
begin
write('N=');readln(n);
i:=1;j:=0;
while n>0 do
begin
if n mod 2=0 then i:=i+j else j:=j+i;
n:=n div 2;
end;
writeln(j);readln;
end.
```

Задача 7.4. Вася в школе изучил квадратные уравнения и понял, как они легко решаются путем вычисления дискриминанта. Но Петя поведал ему о решении кубических уравнений $A * X^3 + B * X^2 + C * X + D = 0$. Вася забыл формулы, о которых ему

рассказал Петя. Но Васе было известно, что все корни уравнения — целые числа и находятся в диапозоне [-100, 100]. Поэтому у Васи есть шанс найти их методом перебора. Но для этого нужно потратить очень много времени, т.к. необходимо будет осуществить перебор нескольких тысяч значений. Помогите Васе написать программу, которая поможет ему найти корни кубических уравнений.

```
uses crt;
var a,b,c,d:integer;
x:integer;
begin
write('a b c d='); readln(a,b,c,d);
x:=-100;
while x<=100 do begin
if a*x*x*x+b*x*x+c*x+d=0 then write(x,' ');
inc(x);
end;
readln;
end.
```

Задача 7.5. Числа. Рассмотрим числа вида $2^p - 1$, где p — простое число. Числами такого вида интересовались многие известные математики.

Необходимо определить, является ли заданное натуральное число N, представленное в шестнадцатеричной системе счисления, числом вида $2^p - 1$ (в десятичной системе счисления).

Данные вводятся с клавиатуры (одна непустая строка длиной не более 80 символов). В строке могут встречаться только цифры (0..9) и заглавные буквы английского алфавита (A..Z). Первым символом в строке не может быть цифра 0 (ноль).

Результат работы программы выводится на экран и может принимать одно из следующих значений:

- натуральное число p в десятичной системе счисления (если заданное шестнадцатеричное число является числом вида $2^p - 1$ и p — простое число);
- error1 (если входная строка содержит символы, не соответствующие шестнадцатеричному числу);
- error2 (если заданное число не представляется в виде $2^p - 1$, при любых значениях p);
- error3 (если заданное число представляется в виде $2^p - 1$, но p — не простое число).

Примеры ввода и вывода данных:

№	Ввод	Вывод	Комментарий
1	1F	5	$1F_{16} = 31_{10} = 2^5 - 1$
2	5AM2	error1	M не является цифрой шестнадцатеричного числа
3	3E	error2	$3E_{16} = 62_{10}$ не представляется в виде $2^p - 1$
4	3F	error3	$3F_{16} = 63_{10} = 2^6 - 1$, 6 — не простое число

Примечание. Цифры числа в шестнадцатеричной системе счисления: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F.

```
Uses crt;
var s:string[80];
    i,err:byte;
    p:word;
Function Prost(x:word):boolean;
var j:word;
begin
 Prost:=(x>1);
 for j:=2 to trunc(sqrt(x)) do
```

```
  if (x mod j)=0 then
   begin prost:=false; break; end;
end;

begin
 readln(s);
 while (s[1]='0') and (length(s)>1) do delete(s,1,1);
 case s[1] of
     '0':p:=0;
     '1':p:=1;
     '7':p:=3;
     else err:=2;
 end;

 if err=0 then
 for i:=2 to length(s) do
  if s[i]='F' then inc(p,4) else err:=2;

 if err=2 then
 begin
  for i:=1 to length(s) do
   if not (s[i] in ['0'..'9','A'..'F']) then
    begin err:=1; break; end;
 end;
 if err=0 then if not prost(p) then err:=2;
 case err of
  0:writeln(p);
  1:writeln('error1');
  2:writeln('error2');
 end;
 readln;
end.
```

Задача 7.6. Множества. Необходимо заполнить одномерный массив случайными неповторяющимися байтовыми числами таким образом, чтобы числа в нем были отсортированы. Методы сортировки использовать нельзя.

Неприменение в курсе программирования реализованных в языке типов множеств является непозволительным действием. Для этой задачи вначале сгенерированные байтовые числа занесем в множество, что само по себе не допускает равных элементов. Далее берем элементы множества по возрастанию и заносим в массив. В итоге имеем отсортированный массив.

```
uses crt;
{отсортированный массив без повторяющихся элементов}
var
s :set of byte;
k,i:integer;
x:byte;
b:array[1..255] of byte;{массив задаем с запасом}
begin
clrscr;
k:=1;
for i:=0 to 255 do  begin x:=random(255); s:=s + [x];end;{заполнение множества}
{Перепишем множества в массив}
for i:=1 to 255 do
if i in s then begin b[k]:=i;k:=k+1;write (i ,'  ');end;
clrscr;
writeln;
{печать элементов массива}
```

```
for i:=1 to k-1 do write(b[i]:4);
readln;
end.
```

Задача 7.7. Счет. Густаво знает, как считать, но он еще только учится писать. Он уже научился писать цифры 1, 2, 3 и 4. Правда, он еще не знает, что 4 отличается от 1, поскольку он думает, что 4 — это просто другой способ написания 1. Ему нравится простая игра, которую он придумал: он пишет число из четырех известных ему цифр и складывает их значения. Например:

132 = 1 + 3 + 2 = 6

112314 = 1 + 1 + 2 + 3 + 1 + 1 = 9 (для Густаво 4 = 1)

Густаво интересно, сколько таких чисел с суммой, равной n, он может составить. Для n = 2 он может составить 5 чисел: 11, 14, 41, 44 и 2. Тем не менее, он не может найти это количество для n, больших 2. Подскажите, сколькими способами можно получить сумму n(n < 100).

Пример входных данных

```
2
3
Пример выходных данных
5
13
{$N+}
program Gustava;
const B:array[1..4] of byte=(1,2,3,1);
var
    S,n:word;
    w:extended;
    A:array[0..1000] of extended;
Procedure Count;
var i:byte;
    w1:extended;
begin
 if s=n then w:=w+1 else
 if s<n then
 if A[n-s]>0 then w:=A[n-s]+w else
 begin
 w1:=w;
 For i:=1 to 4 do
 begin
  inc(S,B[i]);
  count;
  dec(s,B[i]);
 end;
 A[n-s]:=w-w1;
 end;
end;
begin
 S:=0;
 w:=0;
 readln(n);
 Count;
 writeln(w:0:0);
 readln;
end.
```

Задача 7.8. Самоописывающая последовательность.

Самоописывающая последовательность Соломона Голомба < f(1), f(2), f(3)…— это единственная неубывающая последовательность положительных целых чисел, обладающая тем свойством, что она содержит ровно f(k) экземпляров k для каждого k. Немного поразмыслив, можно прийти к выводу, что последовательность должна начинаться так:

n	1	2	3	4	5	6	7	8	9	10	11	12
f(n)	1	2	2	3	3	4	4	4	5	5	5	6

Требуется написать программу, которая будет рассчитывать значение f(n) по заданному n (0 < N < 1100)

Входные
100
150
200

Выходные данные
21
27
32

```
uses crt;
var m,i,k,n:integer;
kl,p:integer;
f:array[1..1000] of integer;
label m1,m2;
begin
f[1]:=1;
k:=1;
f[2]:=2;
f[3]:=2;
m1:
writeln('n-?>3');
readln(m);
n:=3;
while n<=m do begin
p:=0;
for i:=0 to 1000 do if f[i]>0 then p:=p+1;
for i:=1 to f[n] do
f[p+i]:=n;
{clrscr;}
n:=n+1;
end;
{for i:=1 to 1000 do write(f[i],' ');}
Writeln(f[m]);
writeln('0-EXIT');
readln(KL);
if  KL=0 then goto m2;
goto m1;
m2:
end.
```

Часть 8
АЛГОРИТМЫ НА БИНАРНЫХ ДЕРЕВЬЯХ И ГРАФАХ

С помощью теории графов решается большое количество задач из различных областей. Граф состоит из множества вершин и множества ребер, которые соединяют между собой вершины. С точки зрения теории графов не имеет значения, какой смысл вкладывается в вершины и ребра. Вершинами могут быть населенные пункты, а ребрами дороги, соединяющие их, или вершинами — подпрограммы, соединение вершин ребрами означает взаимодействие подпрограмм. Часто имеет значение направление дуги в графе. Если ребро имеет направление, оно называется дугой, а граф с ориентированными ребрами называется орграфом.

Дадим теперь основные определения теории графов. **Граф G** есть упорядоченная пара **(V, E),** где **V** — непустое множество вершин, **E** — множество пар элементов множества V, пара элементов из V называется ребром. Упорядоченная пара элементов из V называется дугой. Если все пары в E упорядочены, то граф называется **ориентированным.**

Путь — это любая последовательность вершин орграфа, такая, что в этой последовательности вершина b может следовать за вершиной a, только если существует дуга, следующая из a в b. Аналогично можно определить путь, состоящий из дуг. Путь, начинающийся в одной вершине и заканчивающийся в одной вершине, называется циклом. Граф, в котором отсутствуют циклы, называется ациклическим.

Важным частным случаем графа является дерево. **Деревом** называется орграф, для которого:

1. Существует узел, в который не входит ни одна дуга. Этот узел называется корнем.
2. В каждую вершину, кроме корня, входит одна дуга.

С точки зрения представления в памяти важно различать два типа деревьев: бинарные и сильноветвящиеся.

В бинарном дереве из каждой вершины выходит не более двух дуг. В сильноветвящемся дереве количество дуг может быть произвольным.

Бинарные деревья классифицируются по нескольким признакам. Введем понятия степени узла и степени дерева. Степенью узла в дереве называется количество дуг, которое из него выходит. Степень дерева равна максимальной степени узла, входящего в дерево. Исходя из определения степени понятно, что степень узла бинарного дерева не превышает числа 2. При этом листьями в дереве являются вершины, имеющие степень 0.

Другим важным признаком структурной классификации бинарных деревьев является **строгость бинарного дерева**. Строго бинарное дерево состоит только из узлов, имеющих степень 2 или степень 0. Нестрого бинарное дерево содержит узлы со степенью, равной одному.

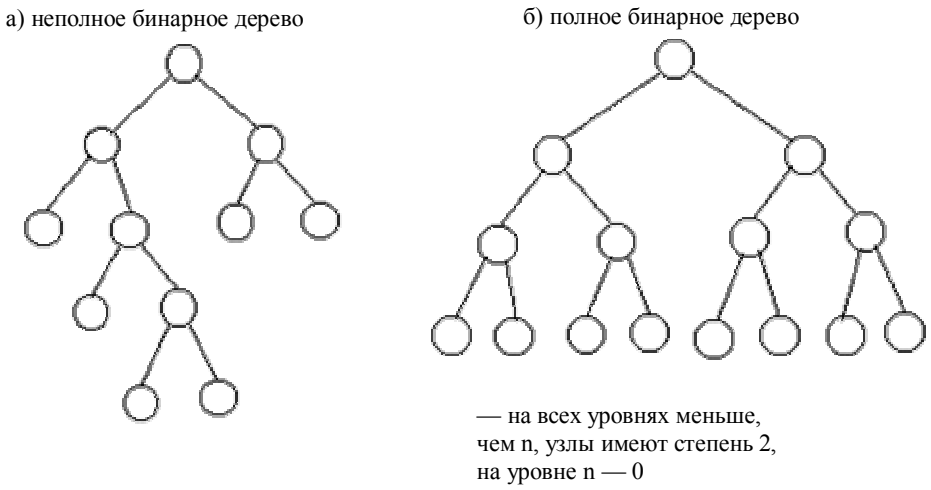


Рис. 8.1. Полное и неполное бинарные деревья

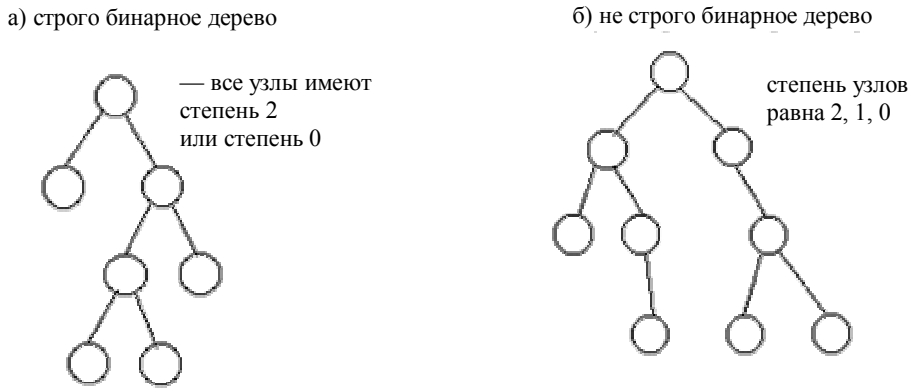


Рис. 8.2. Строго и нестрого бинарные деревья

8.1. Представление бинарных деревьев

Бинарные деревья достаточно просто могут быть представлены в виде списков или массивов. Списочное представление бинарных деревьев основано на элементах, соответствующих узлам дерева. Каждый элемент имеет поле данных и два поля указателей. Один указатель используется для связывания элемента с правым потомком, а другой — с левым. Листья имеют пустые указатели потомков. При таком способе представления дерева обязательно следует сохранять указатель на узел, являющийся корнем дерева.

Можно заметить, что такой способ представления имеет сходство с простыми линейными списками. И это сходство не случайно. На самом деле рассмотренный способ представления бинарного дерева является разновидностью мультисписка, образованного комбинацией множества линейных списков. Каждый линейный список объединяет узлы, входящие в путь от корня дерева к одному из листьев.

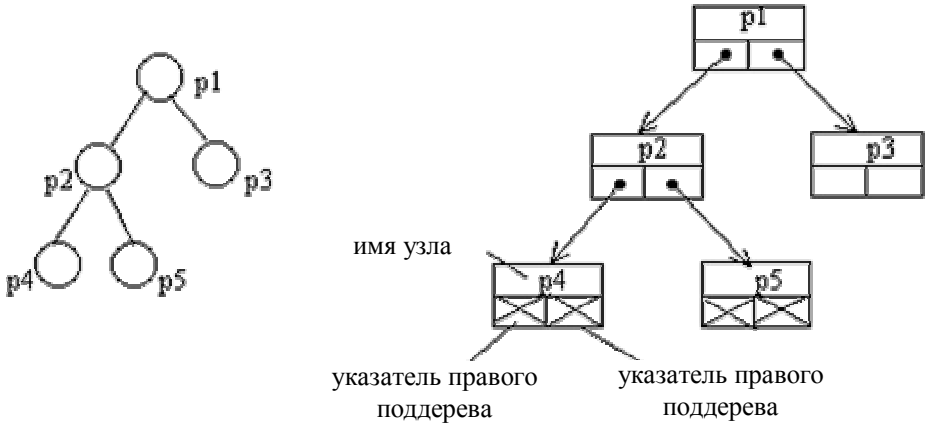


Рис. 8.3. Представление бинарного дерева в виде списковой структуры

Приведем пример программы, которая осуществляет создание и редактирование бинарного дерева, представленного в виде списковой структуры

```
program bin_tree_edit;
type node=record
                name: string;
                left, right: pointer;
       end;
var
        n:integer;
        pnt_s,current_s,root: pointer;
        pnt,current:^node;
        s: string;
procedure node_search (pnt_s:pointer; var current_s:pointer);
{Поиск узла по содержимому}
var
         pnt_n:^node;
begin
pnt_n:=pnt_s; writeln(pnt_n^.name);
if not (pnt_n^.name=s) then
         begin
                if pnt_n^.left <> nil then
                            node_search (pnt_n^.left,current_s);
               if pnt_n^.right <> nil then
                           node_search (pnt_n^.right,current_s);
         end
else current_s:=pnt_n;
end;
procedure node_list (pnt_s:pointer);
{Вывод списка всех узлов дерева}
var
          pnt_n:^node;
begin
pnt_n:=pnt_s; writeln(pnt_n^.name);
if pnt_n^.left <> nil then node_list (pnt_n^.left);
if pnt_n^.right <> nil then node_list (pnt_n^.right);
end;
procedure node_dispose (pnt_s:pointer);
{Удаление узла и всех его потомков в дереве}
var
           pnt_n:^node;
begin
if pnt_s <> nil then
```

```
          begin
                  pnt_n:=pnt_s; writeln(pnt_n^.name);
                  if pnt_n^.left <> nil then
                           node_dispose (pnt_n^.left);
                  if pnt_n^.right <> nil then
                           node_dispose (pnt_n^.right);
                 dispose(pnt_n);
         end
end;
begin
new(current);root:=current;
current^.name:='root';
current^.left:=nil;
current^.right:=nil;
repeat
            writeln('текущий узел -',current^.name);
            writeln('1-присвоить имя левому потомку');
            writeln('2-присвоить имя правому потомку');
            writeln('3-сделать узел текущим');
            writeln('4-вывести список всех узлов');
            writeln('5-удалить потомков текущего узла');
            read(n);
            if n=1 then
            begin {Создание левого потомка}
                     if current^.left= nil then new(pnt)
                     else pnt:= current^.left;
                     writeln('left ?');
                     readln;
                     read(s);
                     pnt^.name:=s;
                     pnt^.left:=nil;
                     pnt^.right:=nil;
                    current^.left:= pnt;
             end;
            if n=2 then
             begin {Создание правого потомка}
                       if current^.right= nil then new(pnt)
                      else pnt:= current^.right;
                      writeln('right ?');
                      readln;
                      read(s);
                      pnt^.name:=s;
                      pnt^.left:=nil;
                      pnt^.right:=nil;
                      current^.right:= pnt;
             end;
             if n=3 then
             begin {Поиск узла}
                      writeln('name ?');
                      readln;
                      read(s);
                      current_s:=nil; pnt_s:=root;
                      node_search (pnt_s, current_s);
                      if current_s <> nil then current:=current_s;
             end;
             if n=4 then
             begin {Вывод списка узлов}
                     pnt_s:=root;
                     node_list(pnt_s);
             end;
             if n=5 then
```

```
            begin {Удаление поддерева}
                      writeln('l,r ?');
                      readln;
                      read(s);
                      if (s='l') then
                                  begin {Удаление левого поддерева}
                                  pnt_s:=current^.left;
                                  current^.left:=nil;
                                  node_dispose(pnt_s);
                                 end
                             else
                                      begin {Удаление правого поддерева}
                                      pnt_s:=current^.right;
                                      current^.right:=nil;
                                      node_dispose(pnt_s);
                                       end;
            end;
until n=0
end.
```

В виде массива проще всего представляется полное бинарное дерево, так как оно всегда имеет строго определенное число вершин на каждом уровне. Вершины можно пронумеровать слева направо последовательно по уровням и использовать эти номера в качестве индексов в одномерном массиве.

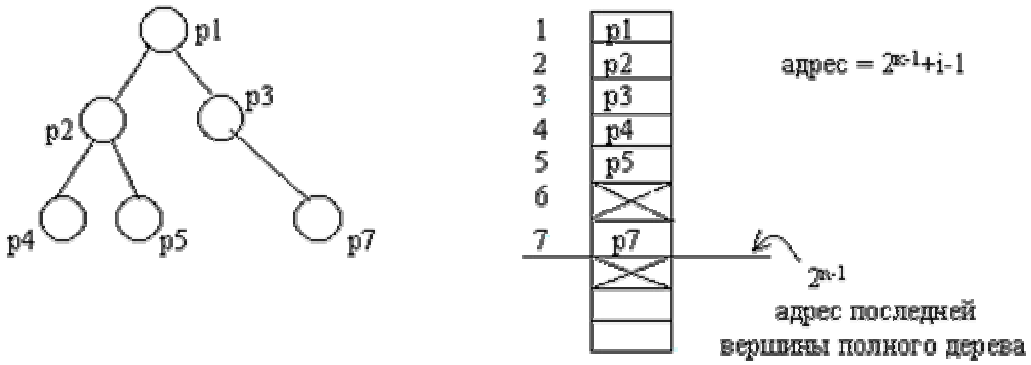


Рис. 8.4. Представление бинарного дерева в виде массива

Если число уровней дерева в процессе обработки не будет существенно изменяться, то такой способ представления полного бинарного дерева будет значительно более экономичным, чем любая списковая структура.

8.2. Прохождение бинарных деревьев.

В ряде алгоритмов обработки деревьев используется так называемое прохождение дерева. Под прохождением бинарного дерева понимают определенный порядок обхода всех вершин дерева. Различают несколько методов прохождения.

Прямой порядок прохождения бинарного дерева можно определить следующим образом:

1. попасть в корень;
2. пройти в прямом порядке левое поддерево;
3. пройти в прямом порядке правое поддерево.

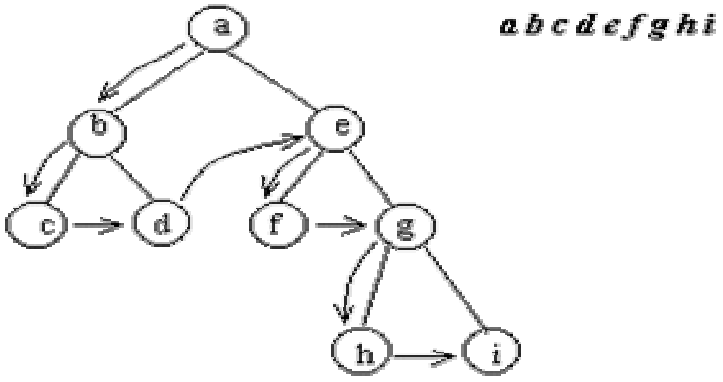


Рис. 8.5. Прямой порядок прохождения бинарного дерева

Прохождение бинарного дерева в обратном порядке можно определить в аналогичной форме:

1. пройти в обратном порядке левое поддерево;
2. пройти в обратном порядке правое поддерево;
3. попасть в корень.

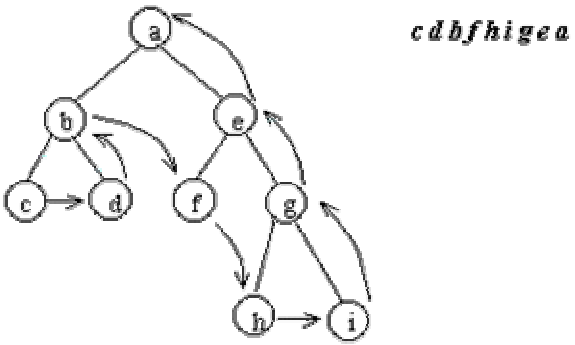


Рис. 8.6. Обратный порядок прохождения бинарного дерева

Определим еще один порядок прохождения бинарного дерева, называемый симметричным.

1. пройти в симметричном порядке левое поддерево;
2. попасть в корень;
3. пройти в симметричном порядке правое поддерево.

Порядок обхода бинарного дерева можно хранить непосредственно в структуре данных. Для этого достаточно ввести дополнительное поле указателя в элементе списковой структуры и хранить в нем указатель на вершину, следующую за данной вершиной при обходе дерева.

Представление деревьев в виде массивов также допускает хранение порядка прохождения дерева. Для этого вводится дополнительный массив, в который записывают адрес вершины в основном массиве, следующей за данной вершиной.

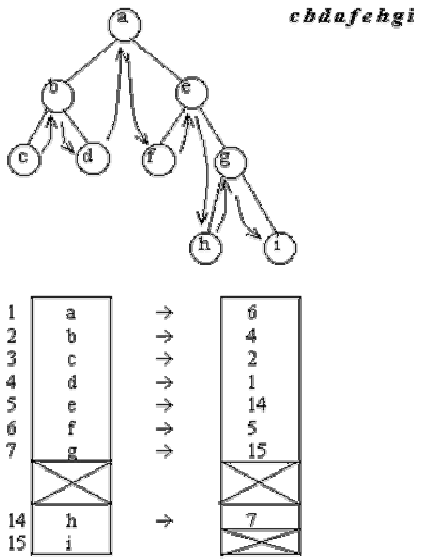


Рис. 8.7. Представление симметрично прошитого бинарного дерева в виде массивов

Такие структуры данных получили название прошитых бинарных деревьев. Указатели или адреса, определяющие порядок обхода, называют нитями. При этом в соответствии с порядком прохождения вершин различают право прошитые, лево прошитые и симметрично прошитые бинарные деревья.

8.3. Сортировка с прохождением бинарного дерева.

В качестве примера использования прохождения бинарного дерева можно привести один из способов сортировки. Допустим, мы имеем некоторый массив и пытаемся упорядочить его элементы по возрастанию. Сама сортировка при этом распадается на две фазы:

1. построение дерева;
2. прохождение дерева.

Дерево строится по следующим принципам. В качестве корня создается узел, в который записывается первый элемент массива. Для каждого очередного элемента создается новый лист. Если элемент меньше значения в текущем узле, то для него выбирается левое поддерево, если больше или равен — правое.

Рис. 8.8. Сортировка по возрастанию с прохождением бинарного дерева

Для создания очередного узла осуществляют сравнение элемента со значениями существующих узлов, начиная с корня.

Во время второй фазы происходит прохождение дерева в симметричном порядке. Результатом сортировки является последовательность значений элементов, извлекаемых их пройденных узлов.

Приведем пример программы сортировки с прохождением дерева. Данные находятся во входном файле inp2.pas

```
Program derevo;
uses crt;
const n=5;
type
keytype=integer;
Tpnode=^Tnode;
 Tnode=record
        Key:keytype;
        left,right:Tpnode;
       end;

Function CreateNode(key:keytype):Tpnode;
var node:Tpnode;
begin
 new(node);
 node^.key:=key;
 Node^.Left:=NIL;
 Node^.right:=NIL;
 CreateNode:=node;
end;

Procedure AddP(var RT:TPNode; key:keytype);
var pp,pc,nd:TPNode;
begin
Pp:=Rt;
PC:=Rt;
While PC<>nil do
begin
pp:=pc;
if key>=pc^.key then pc:=pc^.right else
pc:=pc^.left;
end;
nd:=CreateNode(key);
IF key>=pp^.key then pp^.right:=nd else
pp^.left:=nd;
end;

var
i:word;
k:integer;
root:TPNode;
F:text;
s:keytype;

Procedure Obhod(Rt:TPNode);
begin
If rt<>NIL then
begin
OBHOD(RT^.LEFT);
{OBHOD(RT^.Right); }
writeln(F,RT^.KEY,' ');
writeln(RT^.KEY,' ');
{OBHOD(RT^.LEFT);}
```

```
OBHOD(RT^.Right);
end;
end;
{Меняйте комментированные строки обхода с некомментированными. Получим сорти-
ровку по убыванию}
begin
assign(f,'inp2.pas');
reset(f);
readln(f,s);
Root:=CreateNode(S);
while not eof(f) do
begin
readln(f,s);
AddP(root,s);
end;
close(f);
Assign(F,'Out.pas');
rewrite(F);
Obhod(Root);
close(f);
end.
```

Задача 8.1. Измените программу так, чтобы можно было осуществить поиск числа в бинарном дереве.

8.4. Сортировка методом турнира с выбыванием.

Приведем другой алгоритм сортировки, основанный на использовании бинарных деревьев. Данный метод получил название турнира с выбыванием. Пусть мы имеем исходный массив 10, 20, 3, 1, 5, 0, 4, 8.

Сортировка начинается с создания листьев дерева. В качестве листьев бинарного дерева создаются узлы, в которых записаны значения элементов исходного массива.

Дерево строится от листьев к корню. Для двух соседних узлов строится общий предок, до тех пор, пока не будет создан корень. В узел-предок заносится значение, являющееся наименьшим из значений в узлах-потомках.

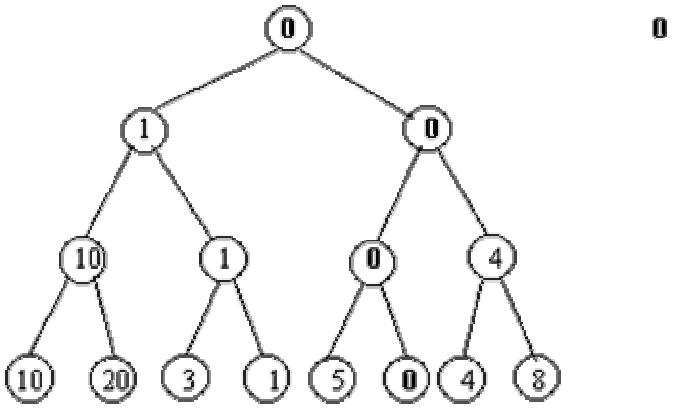


Рис. 8.9. Построение дерева сортировки

В результате построения такого дерева наименьший элемент попадает сразу в корень. Далее начинается извлечение элементов из дерева. Извлекается значение из корня. Данное значение является первым элементом в результирующем массиве. Извлеченное значение помещается в отсортированный массив и заменяется в дереве на специальный символ.

После этого происходит повторное занесение значений в родительские элементы от листьев к корню. При сравнениях специальный символ считается большим по отношению к любому другому значению.

После повторного заполнения из корня извлекается очередной элемент и итерация повторяется. Извлечения элементов продолжаются до тех пор, пока в дереве не останутся одни специальные символы.

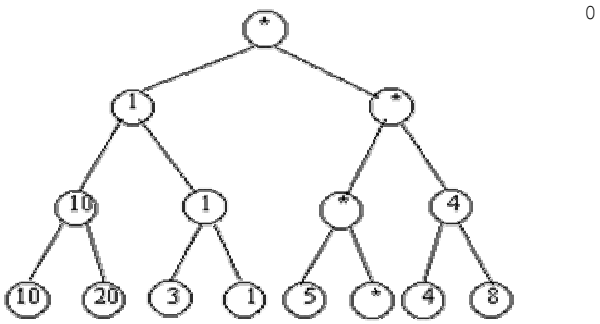


Рис. 8.10. Замена извлекаемого элемента на специальный символ

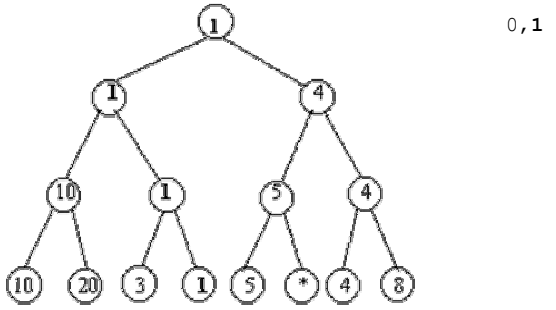
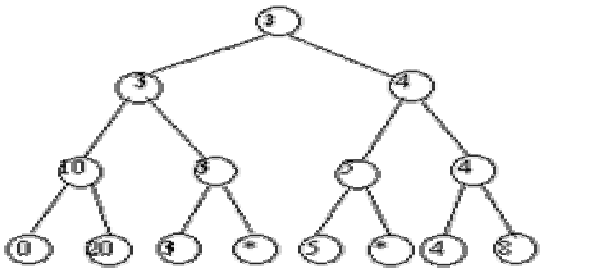
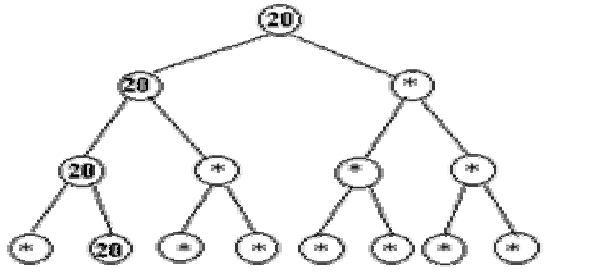
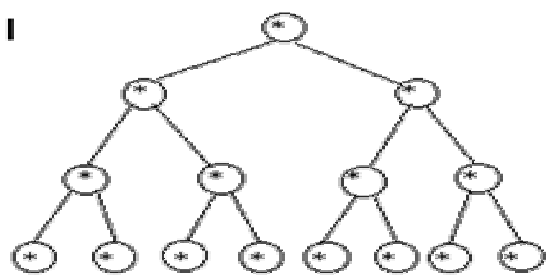


Рис. 8.11. Повторное заполнение дерева сортировки







В результате получим отсортированный массив 0, 1, 3, 4, 5, 8, 10, 20.

Задача 8.2. Напишите программу, реализующую сортировку массива этим методом.

8.5. Матричное представление графа.

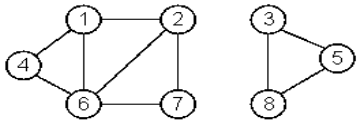
Очень распространенным является матричный способ представления графов. Для представления ненаправленных графов обычно используют матрицы смежности, а для ориентированных графов матрицы инцидентности. Обе матрицы имеют размерность n * n, где n — число вершин в графе. Вершины графа последовательно нумеруются.

Матрица смежности имеет нулевое значение в позиции m (i, j), если не существует ребра, связывающего вершину i с вершиной j, или имеет единичное значение в позиции m (i, j), если такое ребро существует.

Правила построения матрицы инцидентности аналогичны правилам построения матрицы смежности. Разница состоит в том, что единица в позиции m (i,j) означает выход дуги из вершины i и вход дуги в вершину j.

Представление графов. Матрица смежности

Граф:

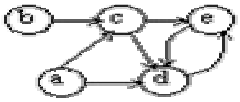


	1	2	3	4	5	6	7	8
1	0	1	0	1	0	1	0	0
2	1	0	0	0	0	1	1	0
3	0	0	0	0	1	0	0	1
4	1	0	0	0	0	1	0	0
5	0	0	1	0	0	0	0	1
6	1	1	0	1	0	0	1	0
7	0	1	0	0	0	1	0	0
8	0	0	1	0	1	0	0	0

Поскольку матрица смежности симметрична относительно главной диагонали, то можно хранить и обрабатывать только часть матрицы. Можно заметить, что для хранения одного элемента матрицы достаточно выделить всего один бит.

8.6. Алгоритмы на графах.

В некоторых матричных алгоритмах обработки графов используются так называемые матрицы путей. Определим понятие пути в ориентированном графе. Под путем длиной k из вершины i в вершину j мы будем понимать возможность попасть из вершины i в вершину j за k переходов, каждому из которых соответствует одна дуга. Одна матрица путей mk содержит сведения о наличии всех путей одной длины k в графе. Единичное значение в позиции (i, j) означает наличие пути длины k из вершины i в вершину j.



m_1

	a	b	c	d	e
a	0	0	1	1	0
b	0	0	1	0	0
c	0	0	0	1	1
d	0	0	0	0	1
e	0	0	0	1	0

m_2

	a	b	c	d	e
a	0	0	0	1	1
b	0	0	0	1	1
c	0	0	0	1	1
d	0	0	0	1	0
e	0	0	0	0	1

m_3

	a	b	c	d	e
a	0	0	0	1	1
b	0	0	0	1	1
c	0	0	0	1	1
d	0	0	0	0	1
e	0	0	0	1	0

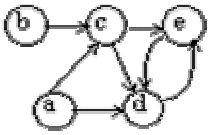
Рис. 8.13. Матрицы путей

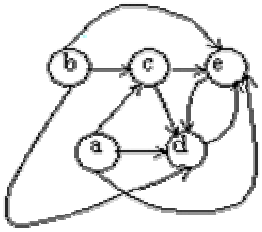
Матрица m1 полностью совпадает с матрицей инцидентности. По матрице m1 можно построить m2. По матрице m2 можно построить m3 и т.д. Если граф является ациклическим, то число мариц путей ограничено. В противном случае матрицы будут повторяться до бесконечности с некоторым периодом, связанным с длиной циклов. Матрицы путей не имеет смысла вычислять до бесконечности. Достаточно остановиться в случае повторения матриц.

Если выполнить логическое сложение всех матриц путей, то получится транзитивное замыкание графа.

$$M\,tr = m\,1\ OR\ m\,2\ OR\ m\,3$$

В результате матрица будет содержать все возможные пути в графе.





m_{tr}

	a	b	c	d	e
a	0	0	1	1	1
b	0	0	1	1	1
c	0	0	0	1	1
d	0	0	0	1	1
e	0	0	0	1	1

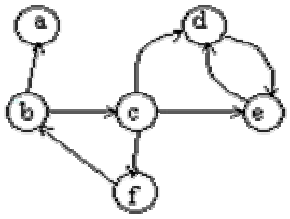
Рис. 8.14. Транзитивное замыкание в графе

Наличие циклов в графе можно определить с помощью эффективного алгоритма. Алгоритм может быть реализован как для матричного, так и для спискового способа представления графа.

Принцип выделения циклов следующий. Если вершина имеет только входные или только выходные дуги, то она явно не входит ни в один цикл. Можно удалить все такие вершины из графа вместе со связанными с ними дугами. В результате появятся новые вершины, имеющие только входные или выходные дуги. Они также удаляются. Итерации повторяются до тех пор, пока граф не перестанет изменяться. Отсутствие изменений свидетельствует об отсутствии циклов, если все вершины были удалены. Все оставшиеся вершины обязательно принадлежат циклам.

Сформулируем алгоритм в матричном виде

1. Для **i** от **1** до **n** выполнить шаги 1–2.
2. Если строка **M(*i* ,*) = 0**, то обнулить столбец ***i.***
3. Если столбец **M(*, i) = 0**, то обнулить строку ***i***.
4. Если матрица изменилась, то выполнить шаг 1.
5. Если матрица нулевая, то ***стоп, граф ациклический***, иначе ***матрица содержит вершины, входящие в циклы.***



начало	a	b	c	d	e	f
итерация 1	b	c	d	e	f	
итерация 2		c	d	e	f	
итерация 3		c	d	e		
итерация 4			d	e		
итерация 5			d	e		

итерация 1

	a	b	c	d	e	f	
a	0	0	0	0	0	0	< обнуляемая строка
b	1	0	0	0	0	0	
c	0	0	0	1	1	1	< обнуляемая строка
d	0	0	0	0	1	0	
e	0	0	0	1	0	0	
f	0	1	0	0	0	0	

Рис. 8.15. Поиск циклов в графе

Достоинством данного алгоритма является то, что происходит одновременное определение цикличности или ацикличности графа и формирование списка вершин, входящих в циклы. В матричной реализации после завершения алгоритма остается матрица инцидентности, соответствующая подграфу, содержащему все циклы исходного графа.

Задача 8.3. Определить ацикличность графов, имеющих следующие матрицы смежности:

a)

0	0	1	1	0
1	0	0	0	1
0	0	0	0	0
1	0	1	0	1
0	0	0	0	0

b)

0	1	1	0	0
1	0	0	0	0
1	0	0	1	1
0	0	1	0	0
0	0	1	0	0

Рассмотрим граф, представленный первой матрицей.

```
uses crt;
const a:array[1..5,1..5] of byte=((0,0,1,1,0),(1,0,0,0,1),
(0,0,0,0,0),(1,0,1,0,1),(0,0,0,0,0));
var
i,j,i1,j1,p:byte;
kol:integer;
label m1;
{процедура вывода матрицы смежности}
procedure writematr;
var i2,j2:byte;
begin
for i2:=1 to 5 do begin
for j2:=1 to 5 do write(a[i2,j2]);
writeln;
end;
end;
begin
clrscr;
writematr;
m1:
{обнуление столбцов, соответствующих нулевым строкам}
for i:=1 to 5 do begin
p:=0;
for j:=1 to 5 do
if a[i,j]=0 then p:=p+1;
if p>=5 then   for i1:=1 to 5 do a[i1,i]:=0;
end;
writeln;
writematr;
{обнуление строк, соответствующих нулевым столбцам}
for i:=1 to 5 do begin
p:=0;
for j:=1 to 5 do
if a[j,i]=0 then p:=p+1;
if p>=5 then   for i1:=1 to 5 do a[i,i1]:=0;
end;
writeln;
writematr;
kol:=kol+1;
if kol<5 then goto m1;
p:=0;
for i:=1 to 5 do
for j:=1 to 5 do if a[i,j]>0 then p:=p+1;
if p>0 then writeln('цикличный граф')else writeln('ацикличный');
readln;
end.
```

СПИСОК ЛИТЕРАТУРЫ

1. Брудно А.Л., Каплан Л.И. Московские олимпиады по программированию. — М.: Наука, 1996.
2. Васильев Н.Б., Гутенмахер В.Л., Работ Ж.М., Тоом А.Л. Заочные математические олимпиады. — М.: Наука, 1987.
3. Казиахмедов Т.Б. Программирование национальных орнаментов и узоров. — Нижневартовск: Изд-во НГГУ, 2007.
4. Кормен Т., Лейзерсон Ч., Ривест Р. Алгоритмы построение и анализ. — М.: Бином, 2004.

СОДЕРЖАНИЕ