«Массивы»

Лекция 4

Массив

- – это структурированный тип данных, состоящий из фиксированного числа элементов, имеющих один и тот же тип.
- Массивы описываются следующим образом:

<имя типа> = **ARRAY** [диапазоны индексов] **OF** <тип элемента массива>;

- Одномерные массивы

- Многомерные массивы (Матрицы)

Описание массива

- Перед использованием массив, как и любая переменная, должен быть объявлен в разделе объявления переменных. В общем виде объявление массива выглядит так:

Имя: array [нижний_индекс .. верхний_индекс] of тип

где

Имя - имя переменной-массива;

array - ключевое слово, обозначающее, что переменная является массивом;

нижний_индекс и верхний_индекс - целые числа, определяющие диапазон изменения индексов (номеров) элементов массива и, неявно, количество элементов (размер) массива;

тип - тип элементов массива.

типичные действия с массивами

- вывод массива;
- ввод массива;
- сортировка массива;
- поиск в массиве заданного элемента;

Вывод массива

- это вывод на экран значений элементов массива. Если в программе необходимо вывести значения всех элементов массива, то для этого удобно использовать инструкцию FOR, переменная-счётчик которой может быть реализована как индекс элемента массива.

программа, выводящая на печать номера и названия дней недели, хранящиеся в массиве day

```
var
    day: array [1..7] of string [11];
    i: integer;
  begin
    day [1]:=‘Понедельник’;
    day [2]:=‘Вторник’;
    day [3]:=‘Среда’;
    day [4]:=‘Четверг’;
    day [5]:=‘Пятница’;
    day [6]:=‘Суббота’;
    day [7]:=‘Воскресенье’;
    for i:= 1 to 7 do writeln (i, ‘   ’, day [i]);
    end.
```

Ввод массива

- Под вводом массива понимается ввод значений элементов массива. Как и вывод массива, ввод удобно реализовать при помощи инструкции FOR.
- Случайным образом
- Ввод с клавиатуры
- По формуле

Случайным образом

```
Program zadahca1;
Var A:array[1..15] of integer;
i:nteger;
Begin
Randomize;
For i:=1 to 15 do
Begin
A[i]:=random(27);
Write(A[i]:3);
End;
Readln;
End.
```

Ввод с клавиатуры

```
Program zadahca2;
Const n=10;
Var A:array[1..n] of integer;
i:nteger;
Begin
For i:=1 to n do
Begin
Readln(A[i]);
Write(A[i]:3);
End;
Readln;
End.
```

По формуле

```
Program zadahca3;
Var A:array[1..100] of integer;
n,i:integer;
Begin
Writeln(‘введите количество элементов в массиве’);
Readln(n);
For i:=1 to n do
Begin
A[i]:=sin(I)+cos(I);
Write(A[i]:3);
End;
Readln;
End.
```

Сортировка массива

- Под *сортировкой массива* подразумевается процесс перестановки элементов с целью упорядочивания их в соответствии с каким-либо критерием. Например, если имеется массив целых a , то после сортировки по возрастанию должно выполняться условие:

a [1] <= a [2] <= … <= a [SIZE]

SIZE - верхняя граница индекса массива.

методы сортировки

- Сортировка выбором
- Сортировка обменом (методом "пузырька")
- Шейкерная перестановка
- Сортировка включением
- Сортировка Хоара

Сортировка выбором по возрастанию массива A из N целых чисел

```
Program Sort_Vybor1;
var    A:array[1..100] of integer;    N,i,m,k,x : integer;
begin
write('количество элементов массива ');
read(N);
for i:=1 to n do read(A[i]);
for k:=n downto 2 do { k - количество элементов для поиска
   max }
begin
m:=1; { m - место max }
for i:=2 to k do if A[i]>A[m] then m:=i;
{меняем местами элементы с номером m и номером k}
x:=A[m]; A[m]:=A[k]; A[k]:=x;
end;
for i:=1 to n do write(A[i],' '); {упорядоченный массив}
end.
```

Сортировка обменом по возрастанию массива A из N целых чисел. (Базовый вариант)

```
Program Sort_Obmen1;
var   A: array[1..100] of integer;   N,i,k,x : integer;
begin
write('количество элементов массива ');
read(N);
for i:=1 to n do read(A[i]);
for k:=n-1 downto 1 do {k - количество сравниваемых
   пар}
for i:=1 to k do
if A[i]>A[i+1] then
{меняем местами соседние элементы}
begin
x:=A[i];   A[i]:=A[i+1];      A[i+1]:=x;
end;
for i:=1 to n do write(A[i],' '); {упорядоченный массив}
end.
```

Сортировка обменом с проверкой факта перестановки.

```
Program Sort_Obmen2;
var   A: array[1..100] of integer;     N,i,k,x :
   integer; p: boolean;
begin
write('количество элементов массива ');
read(N);
for i:=1 to n do read(A[i]);
k:=n-1; {количество пар при первом проходе}
p:=true; {логическая переменная p истинна,
   если были перестановки, т.е. нужно
   продолжать сортировку}
while p do
begin
p:=false;
```

```
{Начало нового прохода. Пока перестановок не
  было.}
for i:=1 to k do
if A[i]>A[i+1] then
begin
x:=A[i]; A[i]:=A[i+1]; A[i+1]:=x;
{меняем элементы местами}
p:=true; {и запоминаем факт перестановки}
end;
k:=k-1;
{уменьшаем количество пар для следующего
  прохода}
end;
for i:=1 to n do write(A[i],' '); {упорядоченный
  массив}
end.
```

Сортировка обменом с запоминанием места последней перестановки.

```
Program Sort_Obmen3;
var  A: array[1..100] of integer;   N,i,k,x,m :
  integer;
begin
write('количество элементов массива ');
read (N);
for i:=1  to n do read(A[i]);
k:=n-1; {количество пар при первом проходе}
while k>0 do
begin
m:=0;
```

```
{пока перестановок на этом проходе нет, место равно 0}
for i:=1 to k do
if A[i]>A[i+1] then
begin
x:=A[i]; A[i]:=A[i+1]; A[i+1]:=x; {меняем элементы местами}
m:=i; {и запоминаем место перестановки}
end;
k:=m-1; {количество пар зависит от места последней перестановки}
end;
for i:=1 to n do write(A[i],' '); {упорядоченный массив}
end.
```

Шейкерная сортировка по возрастанию массива A из N целых чисел.

```
Program Shaker;
var   A: array[1..100] of integer;  N,i,k,x,j,d : integer;
begin
write('количество элементов массива ');
read(N);
for i:=1 to n do read(A[i]);
d:=1; i:=0;
for k:=n-1 downto 1 do {k - количество сравниваемых
   пар }
 begin
i:=i+d;
for j:=1 to k do
```

```
begin
if (A[i]-A[i+d])*d>0 then
{меняем местами соседние элементы}
begin x:=A[i]; A[i]:=A[i+d]; A[i+d]:=x;
end;
i:=i+d;
end;
d:=-d;
{меняем направление движения на
  противоположное}
end;
for i:=1 to n do write(A[i],' '); {упорядоченный
  массив}
end.
```

Сортировка по возрастанию массива A из N целых чисел включением с линейным поиском.

```
Program Sort_Include1;
var    A: array[1..100] of integer;   N,i,k,x : integer;
begin
write('количество элементов массива ');
read(N);
read(A[1]); {for i:=1 to n do read(A[i]);}
{k - количество элементов в упорядоченной части массива}
for k:=1 to n-1 do
begin
read(x); {x:=A[k+1];}
i:=k;
while (i>0)and(A[i]>x) do
begin
A[i+1]:=A[i];
i:=i-1;
end;
A[i+1]:=x;
end;
for i:=1 to n do write(A[i],' '); {упорядоченный массив}
end.
```

Быстрая сортировка по возрастанию массива A из N целых чисел.

- **Program** Quick_Sort;
- **var** A:**array**[1..100] **of** integer;
- N,i : integer;
- {В процедуру передаются левая и правая границы сортируемого фрагмента}
- **procedure** QSort(L,R:integer);
- **var** X,y,i,j:integer;
- **begin**
- X:=A[(L+R) **div** 2];
- i:=L; j:=R;
- **while** i<=j **do**
- **begin**
- **while** A[i]<X **do** i:=i+1;
- **while** A[j]>X **do** j:=j-1;

```
if i<=j then
begin
y:=A[i]; A[i]:=A[j]; A[j]:=y;
i:=i+1; j:=j-1;
end;
end;
if L<j then QSort(L,j);
if i<R then QSort(i,R);
end;
begin
write('количество элементов массива ');
read(N);
for i:=1 to n do read(A[i]);
QSort(1,n);                    {упорядочить элементы с первого до n-го}
for i:=1 to n do write(A[i],' ');   {упорядоченный массив}
end.
```

поиск в массиве заданного элемента

- Наиболее простой - это алгоритм перебора. Поиск осуществляется последовательным сравнением элементов массива с образцом до тех пор, пока не будет найден элемент, равный образцу, или не будут проверены все элементы. Алгоритм простого перебора применяется, если элементы массива не упорядочены.

Поиск максимума в массиве

```
Max:=a[1];
nMax:=1;
For i:=1 to n do
Begin
If a[i]>Max then begin
Max:=a[i];
nMax:=i;
End;
```

Нахождение суммы (произведения, количество и т.д.) четных элементов

```
Num:=0;
Sum:=0;
For i:=1 to n do
  If a[i] mod 2=0 then
begin
  Num:=Num+1;
  Sum:=Sum+a[i];
End;
```

Нахождение среднего арифметического четных (нечетных и др) элементов массива

```
Num:=0;
Sum:=0;
Sr:=0;
For i:=1 to n do
  if a[i] mod 2=0 then
Begin
  Num:=Num+1;
Sum:=Sum+a[i];
End;
Sr:=Sum/Num;
```

Выбор из массива элементов, удовлетворяющих некоторому условию, и формирование из них нового массива

```
j:=0;
For i:=1 to n
  If a[i] mod 2 = 0 then
Begin
  j:=j+1;
  b[j]:=a[i];
End;
```

Обмен значений 1-го элемента массива с n-м, 2-го – с (n-1)-м и т.д.

- Конструкция **(n div 2)+(n mod 2)** означает номер центрального элемента массива (для массива с нечетной длинной) или номер последнего элемента первой половины массива (для массива с четной длиной)

```
For i:=1 to (n div 2)+(n mod 2) do
Begin
   k:=a[i];
   a[i]:=a[n+1-i];
   a[n+1-i]:=k;
End;
```

Матрицы

- Способ организации данных, при котором каждый элемент определяется номером строки и номером столбца, на пересечении которых он расположен, называется двумерным массивом или матрицей.

Описание массива

```
Const
  n=20; m=30;
Type
  MyArray2 = array [1..n] of array [1..m] of
   integer;
Var
  A : MyArray2;
```

- Еще более краткое описание массива A можно получить, указывая имя массива и диапазоны изменения индексов для каждой размерности массива

```
Const
  n=20; m=30;
Type
  MyArray2 = array [1..n, 1..m] of integer;
Var
  A : MyArray2;
```

- удобно использовать объявление массива в разделе описания переменных

```
Const
  n=20; m=30;
Var
  A : array [1..n, 1..m] of integer;
```

Или

```
Var
  A : array [1..5, 1..3] of integer;
```

A [i,j] – обращение к элементу

Первый индекс - это номер строки, а второй - номер столбца, где расположен элемент массива.

Вывод элементов массива в виде таблицы

```
For i:=1 to n do
Begin
For j:=1 to m do
Begin
Write(a[i,j]:5);
End;
Writeln;
End;
```

Соотношение индексов в квадратной матрице

- Если номер строки совпадает с номером столбца **(i=j),** это означает что элемент **лежит на главной диагонали**,
- Если номер строки превышает номер столбца **(i>j),** это означает что элемент находится **ниже главной диагонали,**
- Если номер столбца больше номера строки **(i<j),** это означает что элемент находится **выше главной диагонали**,

- Элемент лежит **на побочной диагонали,** если его индексы удовлетворяют равенству **i+j-1=n**,
- Элемент находится **над побочной диагональю** если его индексы удовлетворяют неравенству **i+j<n+1**,
- Элемент находится **под побочной диагональю** если его индексы удовлетворяют неравенству **i+j>n+1**.

Перебор элементов главной/побочной диагонали матрицы

```
For i:=1 to n do
Begin
A[i,i]:={элементы главной диагонали}
A[i,n+1-i]:={элементы побочной  диагонали}
End;
```

Перебор элементов, расположенных выше/ниже главной диагонали

```
{выше}
 For i:=1 to n-1 do
For j:=i+1 to n do
 a[i,j]:=…

{ниже};
For i:=2 to n do
For j:=i to n-1 do
 a[i,j]:=…
```

Перебор элементов, расположенных выше/ниже побочной диагонали матрицы

```
{выше}
For i:=1 to n-1 do
For j:=1 to n-i do
 a[i,j]:=…

{ниже}
For i:=2 to n do
For j:=n+2-i to n do
 a[i,j]:=...
```

Нахождение суммы/произведения элементов каждой строки

```
For i:=1 to n do
Begin
Sum:=0;{p:=1;}
For j:=1 to m do
 Sum=Sum+a[i,j]; {p=p*a[i,j]; }
Writeln(‘сумма элементов’,sum);
End;
```

Нахождение суммы/произведения элементов каждого столбца

```
For i:=1 to n do
Begin
Sum:=0;{p:=1;}
For j:=1 to m do
 Sum=Sum+a[j,i]; {p=p*a[j,i]; }
Writeln('сумма элементов',sum);
End;
```