

**\* Массивы. Определение,  
описание, размещение  
в памяти,  
использование**

**Лекция 8**

**Массив** представляет собой совокупность данных (элементов) *одного типа* с *общим* для всех элементов *именем*.

Элементы массива *пронумерованы*, и обратиться к каждому из них можно по номеру. Номера элементов массива иначе называются *индексами*.

Характеристики массива:

- тип – *общий* тип всех элементов массива;
- размерность (ранг) – *количество индексов* массива;
- диапазон изменения индекса (индексов) – определяет *количество элементов в массиве*.

*Одномерный массив* — это массив, в котором элементы нумеруются одним индексом.

Если в массиве хранится *таблица значений (матрица)*, то такой массив называется *двумерным*, а его элементы нумеруются *двумя* индексами — номером *строки и столбца* соответственно.

При обращении к элементу массива индекс указывается в *квадратных* скобках после имени массива.

Например, *a[3]*, *c[1,2]*.

В памяти компьютера все элементы массива обязательно занимают одну непрерывную область (массив), отсюда и произошло это название. Двумерные массивы располагаются в памяти по строкам: сначала все элементы первой строки, затем — второй и т. д.

# \* Описание массивов

Элементы массива могут иметь любой тип, индексы — порядковый тип, чаще всего диапазон.

*Сам массив — это тип, определяемый пользователем* на основе типа его элементов и типа индексов. Поэтому имеются два основных способа описания массива:

- сначала описать тип массива в разделе описания типов, а затем описать переменную этого типа;
- описать массив непосредственно в разделе описания переменных. В этом случае используется *анонимный*, т. е. *безымянный* тип.

# \* Предварительное описание типа массива

*Предварительное описание типа* считается более строгим способом описания массива. Такое описание необходимо, например, при использовании имени массива в качестве параметра процедуры или функции.

Например, вот как выглядит описание одномерного массива:

**type**

ИмяТипа = **array**[ НижняяГраница .. ВерхняяГраница ] **of**  
ТипЭлементов;

**var**

ИмяМассива : ИмяТипа;

Описание массивов других размерностей выполняется аналогично, например для двумерного массива:

**type**

ИмяТипа = **array** [ НижняяГрИндекса1 .. ВерхняяГрИндекса1,  
НижняяГрИндекса2 .. ВерхняяГрИндекса2 ] of ТипЭлементов;

**var**

ИмяМассива : ИмяТипа;

Пример:

**type**

**matr** = **array** [1..5, 1..6] of **real**;

**var**

**m**: **matr**; {**matrix** - имя переменной, **matr** — имя ее типа}

# \* Описание массива в разделе VAR

Самый простой и короткий способ описания массива — это объявить переменную в разделе описания переменных var. В общем виде описание выглядит так:

□ для одномерного массива:

```
var ИмяМассива: array[НижняяГраница..ВерхняяГраница] of  
ТипЭлементов;
```

Например:

```
var a: array[1..100] of integer; { 100 элементов — целые числа }
```

□ для двумерного массива:

```
var ИмяМассива: array[НижняяГрИндекс1..ВерхняяГрИндекс1,  
НижняяГрИндекс2..ВерхняяГрИндекс2] of ТипЭлементов;
```

Например:

```
Var b: array[1..10, 1..10] of real; { матрица 10 на 10 из  
вещественных чисел }
```



## Следует знать:

- границы изменения индекса должны быть константами. Нельзя использовать переменные при описании границ массива, т. к. память под массив должна быть выделена до начала выполнения программы, а переменные получают значения при выполнении программы;
- верхнюю границу индекса обычно определяют исходя из максимально возможного количества элементов в массиве;
- поскольку в качестве индексов используются переменные и выражения, то возможна ситуация выхода индекса за границы массива.

Например, если массив `a` описан как `var a: array [1..100] of integer`, то обращение к `a[i]` при `i`, равном 0 или 200, означает выход индекса за границу массива. Ошибка выхода индекса за границы выдается, только если включена ди



# \* Действия над массивами

Над массивом как целым допускается только одна операция — массивы, идентичные по структуре, т. е. с одинаковыми типами индексов и элементов, могут быть операндами в операторе присваивания.

Например, если **a** и **b** являются именами массивов одного типа:

```
type mas = array[1..100] of integer;
```

```
type vec = array[1..100] of integer;
```

```
var a,b: mas;
```

```
c: vec;
```

то разрешено присваивание: **a:=b;**. В этом случае массив **a** будет представлять собой точную копию массива **b**. Присваивание **a:=c;** запрещено и вызовет сообщение об ошибке **Type mismatch** (Несоответствие типов).

Можно присваивать одной строке двумерного массива значения одномерного массива и наоборот. Например:

```
type
```

```
  TVec = array[1..10] of integer;
```

```
  TMatrix = array[1..10] of TVec;
```

```
var
```

```
  a: TMatrix;
```

```
  b: TVec;
```

```
begin
```

```
b:=a[5];
```

```
end.
```

Эта операция используется довольно редко. Гораздо чаще приходится обрабатывать массив последовательно, элемент за элементом, используя циклы.

# \* Заполнение массива данными

Заполнение массива можно выполнить следующими способами:

- вводом значений элементов с клавиатуры;
- присваиванием заданных или случайных значений;
- считывая значения элементов из файла.

В любом случае для заполнения массива используется *цикл*. Наиболее удобен цикл `for`, причем для многомерных массивов применяются *вложенные циклы*.

Например, **ввод с клавиатуры** может выглядеть так:

**Одномерный массив:**

```
writeln('Введите размерность массива');  
readln(n);  
for i:=1 to n do  
begin  
writeln('Введите ', i, '-й элемент массива');  
readln(a[i]);  
end;
```

Довольно часто массив заполняется при помощи присваивания элементам определенных или случайных значений. Например, **фрагмент программы заполнения одномерного массива x из n элементов случайными числами в диапазоне от 0 до 10** выглядит так:

```
randomize;  
for i:=1 to n do x[i]:=random(11);
```

# \* Вывод элементов массива

Например, запишем вывод одномерного массива из  $n$  элементов:

□ в столбец:

```
for i:=1 to n do writeln(a[i]);
```

□ в одну строку, через пробел-разделитель:

```
for i:=1 to n do write(a[i], ' '); writeln;
```

□ или с заданием формата, где под каждый элемент отводится 4 позиции:

```
for i:=1 to n do write(a[i]:4); writeln;
```

# \* Обработка одномерных массивов

Рассмотрим типовые задачи обработки на примере одномерных массивов. Для массивов других размерностей применяются аналогичные алгоритмы, которые несколько усложняются использованием вложенных циклов.

Условимся, что в массиве  $a$  содержится  $n$  элементов. Над  $n$  элементами массива  $a$  выполним следующие действия.

## 1. Вычисление суммы элементов:

$s:=0;$

for  $i:=1$  to  $n$  do  $s:=s+a[i];$

## 2. Вычисление произведения элементов:

$s:=1;$

for  $i:=1$  to  $n$  do  $s:=s*a[i];$



3. Подсчет количества элементов, удовлетворяющих какому-либо условию (например, подсчет количества четных чисел в целочисленном массиве):

```
k:=0;
```

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

```
if a[i] mod 2=0 then k:=k+1;
```

4. Поиск элемента с заданным значением. Найти элемент — это значит выяснить его номер в массиве (например, найдем номер первого нулевого элемента массива  $a$ . Если таких элементов нет, то выведем соответствующее сообщение).

```
i:=0; { номер элементов массива }
```

```
repeat
```

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

```
until(a[i]=0) or (i=n) ;
```

```
if a[i]=0 then writeln('Номер первого нулевого эл-та ',i)  
  else writeln('Таких элементов нет');
```

## 5. Поиск максимального элемента и его номера.

Переменная **max** хранит значение максимума, **k** — его номер в массиве:

```
max:=a[1];
```

```
k:=1;
```

```
for i:=2 to n do
```

```
  if a[i]>max then
```

```
    begin
```

```
      max:=a[i];
```

```
        k:=i; { запоминаем значение и номер элемента,  
который больше всех предыдущих}
```

```
    end;
```

## 6. Изменение значений элементов.

Например, пусть в массиве  $a$  хранятся зарплаты  $n$  сотрудников. Тем сотрудникам, у которых зарплата меньше минимально возможной суммы, поднимем зарплату до этого минимального значения  $minzp$ .

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

```
if a[i]<minzp then a[i]:=minzp;
```

Пример. Пользователь вводит количество учащихся в группе и их оценки. Определить средний балл по группе.

type

```
mas=array[1..30] of integer;
```

```
var a:mas;    i,n,s:integer;    sr:real;
```

```
begin
```

```
writeln('Введите количество учащихся');
```

```
readln(n);
```

```
s:=0;
```

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

```
begin
```

```
writeln('Введите оценку ',i,'-го учащегося');
```

```
readln(a[i]);
```

```
s:=s+a[i];
```

```
end;
```

```
sr:=s/n;
```

```
writeln('Средний балл по группе 'sr:3:1);
```

```
readln;
```

```
end.
```

# \*Схема Горнера

Схема Горнера — алгоритм вычисления значения многочлена, записанного в виде суммы мономов, при заданном значении переменной.

## Описание алгоритма

Задан многочлен  $P(x)$ :

$$P(x) = a_0 + a_1x + a_2x^2 + a_3x^3 + \dots + a_nx^n, \quad a_i \in \mathbb{R}$$

Пусть требуется вычислить значение данного многочлена при фиксированном значении  $x = x_0$ . Представим многочлен  $P(x)$  в следующем виде:

$$P(x) = a_0 + x(a_1 + x(a_2 + \dots x(a_{n-1} + a_nx) \dots))$$



Определим следующую последовательность:

$$b_n = a_n$$

$$b_{n-1} = a_{n-1} + b_n x$$

...

$$b_i = a_i + b_{i+1} x$$

...

$$b_0 = a_0 + b_1 x$$

Искомое значение  $P(x_0) = b_0$ .

# \* Домашнее задание

1. Составить опорный конспект лекции по теме «Массивы. Определение, описание, размещение в памяти, использование» на основе презентации.

2. Программирование на языке Pascal. Рапаков Г. Г., Ржеуцкая С. Ю. СПб.: БХВ-Петербург, 2004, стр. стр. 221-233.

Составить программы:

- Пользователь вводит размерность массива и его элементы. Найти минимальный элемент массива.
- Пользователь вводит размерность массива и его элементы. Найти произведение четных элементов массива.
- В заданном массиве необходимо умножить все его четные элементы на значение его максимального элемента.