



# Лекция №6

## Массивы

# Определение Представление в памяти

**Массив** – упорядоченная последовательность переменных одного типа, имеющая общее имя.



В языке Си массивы нумеруются с 0 до N-1, где N – **размер массива** или количество занимаемых массивом ячеек памяти.

Номер элемента в массиве называется **индексом** этого элемента.

# Объявление одномерного массива

<имя\_типа> <имя\_массива> [<количество\_элементов>];

<Количество\_элементов> - всегда **ЦЕЛОЕ** и задается *константным выражением*.

*Примеры:*

```
int mas[10];
```

```
char m [20 * 45];
```

```
float A[N];
```

```
int massive[2 * N];
```

***N обязательно определено ранее как константа!***

***N НЕ может быть переменной!***

# Объявление констант в Си

Константа – это нечто постоянное, неменяющееся.

*Примеры констант:*

- 10 целое число 10;
- 20,5 вещественное число 20,5;
- 'a' символ 'a';
- "abc" слово "abc".

## Как объявить именованную константу?

`const <имя_типа> <имя_константы> = <константное_выражение>;`

*Примеры объявления именованных констант:*

`const int N = 10;` N – именованная константа целого типа int со значением 10

`const float x = 20,5;` x – именованная константа вещественного типа float со значением 20,5

`const float PI = 3.141593;` PI – именованная константа веществ. типа float со значением 3.141593

`const char symbol = 'a';` symbol – именованная константа символьного типа char, в которой хранится символ 'a' (код символа 'a')

# Константное выражение

`<имя_типа> <имя_массива>[<константное_выражение>];`

*Замечание:* тут под «константным выражением» следует понимать «**ЦЕЛОЕ** константное\_выражение»

`<Константное_выражение>` может быть:

- **константой**

10

- **именованной константой**

N (ранее была объявлена как `const int N = <целое_значение>`)

- **выражением, содержащим константы и именованные константы**

20 \* 45

2 \* N

N + M (N и M ранее были объявлены как

`const int N = <целое_значение>;`

`const int M = <целое_значение>;`)

# Объявление одномерного массива

*Пример 1:*

```
const int N = 50;  
int A [N];  
int B [2 * N];
```

*Пример 2:*

```
float M[128];  
const int mas_size = 128 * 128;  
double mas[mas_size];
```

**Еще один способ объявить именованную константу**

**#define <имя\_константы> <значение>**

*Замечание:* так можно объявить не только целочисленную константу, но в случае массивов **<значение>** должно быть **ЦЕЛЫМ ЧИСЛОМ**.

*Пример 3:*

```
#define N 50  
...  
int A [N];  
int B [2 * N];
```

Ниже в тексте программы везде N будет замещено на 50.

*Замечание:* если написать

```
#define N =50
```

то ниже в тексте программы везде N будет замещено на =50

# Объявление одномерного массива с инициализацией

## Способ 1

`<имя_типа> <имя_массива>[<константное_выражение>] =  
{значения элементов массива, разделенные “;”};`

## Способ 2

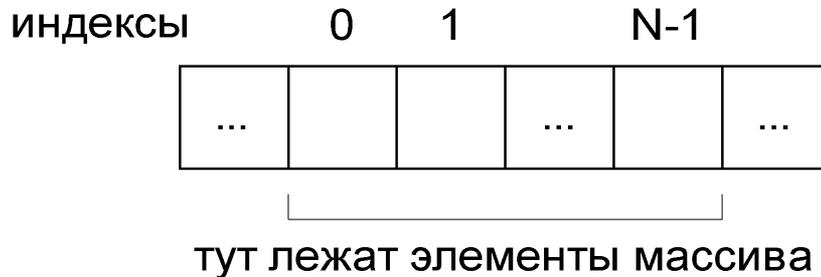
`<имя_типа> <имя_массива>[] =  
{значения элементов массива, разделенные “;”};`

```
int myMas1[4] = {0, 12, 10, 4};  
char myMas2[] = {'c', 'h', 'e', 'c', 'k'};  
double myMas3[4] = {1, 2};  
double myMas4[4] = {0};
```

## Замечание:

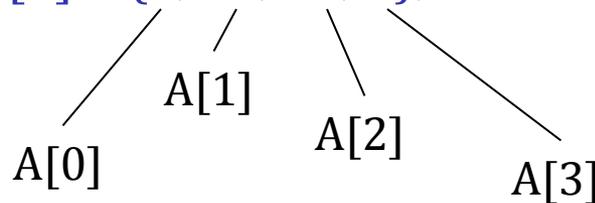
**double myMas4[ ]; - нельзя!!!** Необходимо либо указать количество элементов, либо присвоить значения.

# Представление в памяти Обращение к элементам



*Пример:*

```
int A[4] = {0, 12, 10, 4};
```



Обращение к произвольному элементу массива:

`<имя_массива>[<индекс_элемента>]`

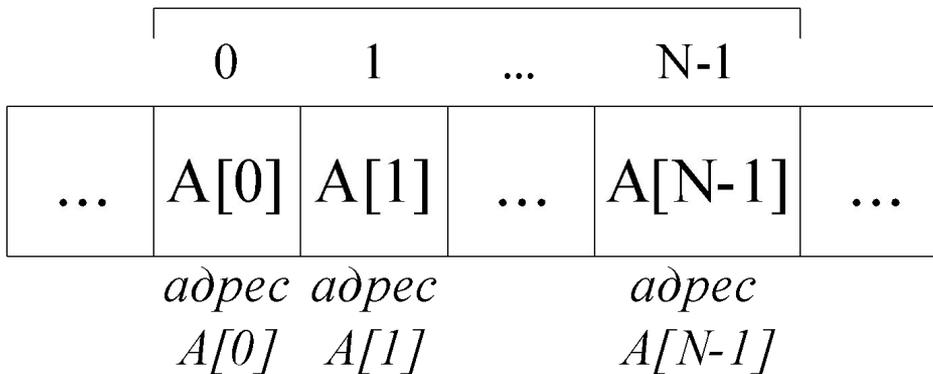
*Замечание:*

`<индекс_элемента>` должен быть только целым числом и должен быть  $\geq 0$  и  $\leq$  `<размер_массива>`

# Представление в памяти

```
int A[N];
```

Элементы массива A



Каждая ячейка имеет размер, соответствующий типу элементов массива. В данном примере - int

Имя массива – это адрес начала массива, а так же адрес элемента с индексом 0.

Т.е. обращение **A** и **&A[0]** – обращение к адресу, по которому начинается массив.

Одновременно работать со всем массивом нельзя, т.е. нельзя сложить два массива **A** и **B** вот так: **A + B**, необходимо все операции с массивами выполнять **поэлементно**.

# Операции с массивами

*Пример 1 (все элементы массива увеличиваются на 1):*

```
int A[3] = {1, 2, 3};
```

## Вариант 1

```
A[0] = A[0]+1;  
A[1]+=1;  
A[2]++;
```

## Вариант 2

```
A[0]++;  
A[1]++;  
A[2]++;
```

## Вариант 3

```
for ( int i = 0; i < 3; i++ )  
    A[i]++;
```

*Пример 2:*

```
double m[100], a, b;  
...  
b = 3 * m[2];  
a = m[50] / b;  
m[99]++;
```

*Пример 3 (сложение двух массивов):*

```
int A[4] = { 2, 3, 4};  
int B[] = {1, -1, 5};  
int C[4] = {0};
```

## Вариант 1

```
C[0] = A[0]+B[0];  
C[1] = A[1]+B[1];  
C[2] = A[2]+B[2];
```

## Вариант 2

```
for (int i=0; i<3; i++)  
    C[i] = A[i]+B[i];
```

# Использование массивов

- Используются для хранения различных последовательностей
- Подробнее об использовании массивов поговорим на практике

# Многомерные массивы

`float A[size];` одномерный массив

`char B[size1][size2];` двумерный массив

`int mas[N1][N2]...[Nk];` k-мерный массив,  
 $N1 * N2 * \dots * Nk$  – количество элементов

`<имя_типа> <имя_массива>[N1] [N2]...[Nk];`

$N1, N2, \dots, Nk$  - константные выражения



# Объявление многомерного массива с инициализацией

*Пример 1:*

```
int mas[N1][N2]...[Nn] = {0};
```

*Пример 2:*

```
int mas[2][3] = { {2, 0, 1}, {1, 5, 3} };
```

**Обращение к элементу многомерного массива**  
<имя\_массива>[индекс 1][индекс 2]...[индекс k]

*Пример 3 (обращение к элементам):*

```
double A[4][2], a, b;
```

...

```
b = 3 * A[2][0];
```

```
a = A[1][1] / b;
```

```
A[1][1]++;
```

# Двумерный массив

*Пример* (увеличить все элементы матрицы на 1):

```
int M[2][3] = { {0, 3, 1}, {6, 1, 5} };
```

```
int i, j;
```

```
for (i=0; i<2; i++)
```

```
    for (j=0; j<3; j++)
```

```
        M[i][j]++;
```