Лекция 9

Массивы

Преподаватель Палехова Ольга Александровна,
кафедра О7 БГТУ «Военмех»

Массивы

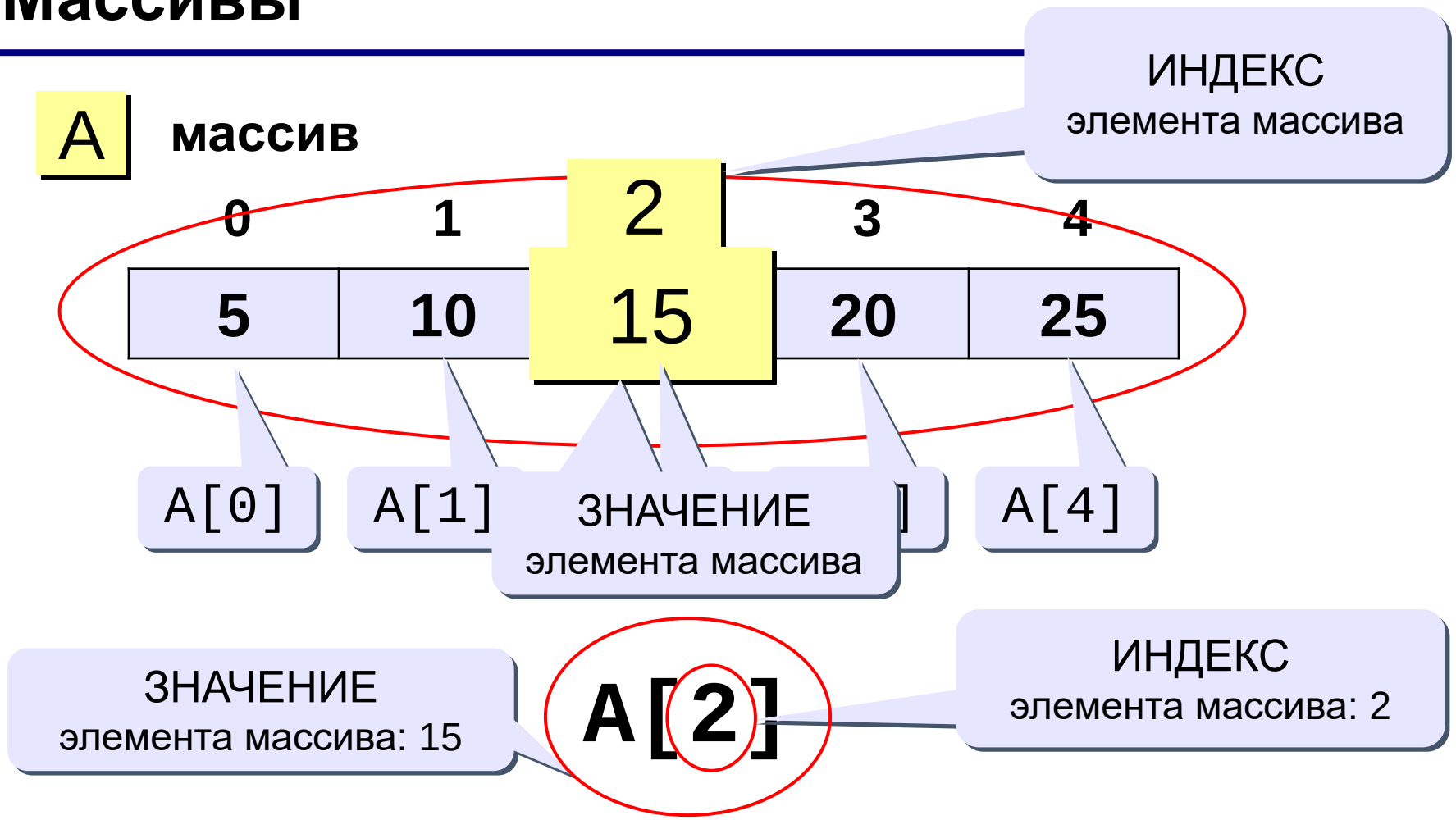
Массив – это группа однотипных элементов, имеющих общее имя и расположенных в памяти непосредственно друг за другом.

Особенности:

- все элементы имеют **один тип**
- весь массив имеет **одно имя**
- все элементы расположены в памяти **рядом**
- у каждого элемента есть свой **индекс**
- действия выполняются **отдельно над каждым элементом**

С массивом как единым целым в языке Си ничего сделать нельзя!

Массивы



! Индексация элементов массива в Си начинается с **НУЛЯ**!

Массивы в языке Си

Особенности:

- с массивом как единым целым ничего сделать нельзя
- для обработки массивов используют **циклы**
- **имя** массива – синоним **адрес**а первого элемента

```
ar   ≡   & ar [0]
```

- **индекс** элемента массива – это **смещение** элемента относительно начала массива

```
ar + 3   ≡   & ar [3]

* ( ar + 3 )   ≡   ar [3]
```

- **выход** за пределы массива **не контролируется**

Статические и динамические массивы

По способу выделения памяти массивы делятся на:

- **статические** – память под массив рассчитывается (выделяется) во время компиляции программы

a

- **динамические** – память под массив выделяется в процессе работы программы

!

Алгоритмы обработки массивов и способы обращения к элементам массива не зависят от способа выделения памяти

Статический массив

- **Размер** массива N может быть задан **только константой**. Реально используемое количество элементов может изменяться от 0 до N.
- Есть **ограничения** на максимально возможный размер массива.
- **Имя** массива по сути является **указателем**-константой на первый элемент массива

```
тип_элементов имя [ размер ];
```

Тип массива

! **Имя массива – УКАЗАТЕЛЬ-константа!**

Объявление статических массивов

Примеры объявлений:

```
int x[10], y[10];
double a[20];
char s[80];
```

С предопределенной константой:

```
#define N 5
int m [N];
```

С присвоением начальных значений:

```
int a[4] = { 8, -3, 4, 6 };
double b[10] = { 1.1 };
int q[] = { 4, 0, -11 };
```

остальные нулевые!

размер массива определяется списком инициализации

! **Если начальные значения не заданы, в ячейках находится «мусор»!**

Массивы

Объявление:

```
#define N 5
int a[N], i;
```

! Для обработки массивов всегда используются циклы

Ввод с клавиатуры:

```
printf("Введите 5 элементов массива:\n");
for( i=0; i < N; i++ )
{
  printf ("A[%d] = ", i );
  scanf ("%d", &a[i] );
}
```

```
a[0] = 5
a[1] = 12
a[2] = 34
a[3] = 56
a[4] = 13
```

Поэлементные операции:

```
for( i=0; i < N; i++ ) a[i] = a[i]*2;
```

Вывод на экран:

```
printf("Результат:\n");
for( i=0; i < N; i++ )
  printf("%4d", a[i]);
```

```
Результат:
  10   24   68 112   26
```

Программа

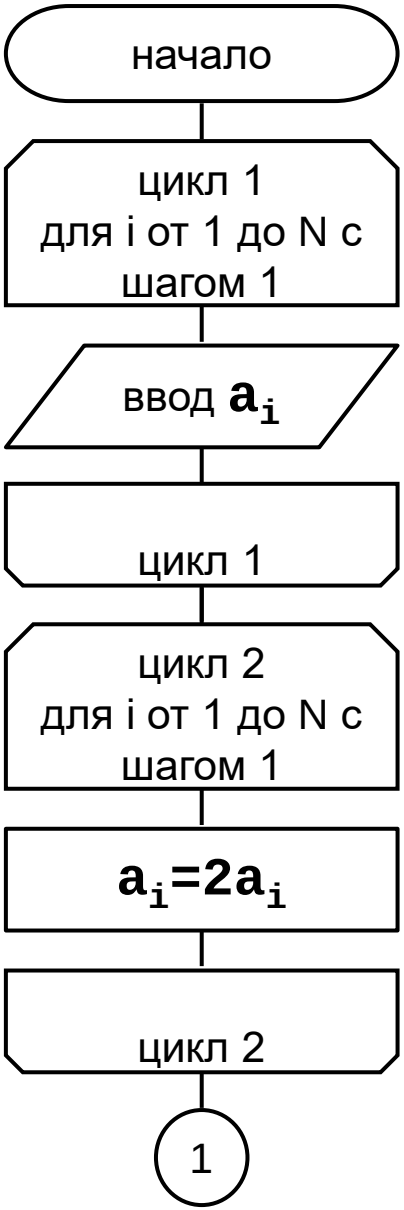
Задача: ввести с клавиатуры массив из 5 элементов, умножить все элементы на 2 и вывести полученный массив на экран.

```
#include <stdio.h>
#define N 5
int main()
{
   int a[N], i;
   /* ввод элементов массива */
   /* обработка массива */
   /* вывод результата */
   return 0;
}
```

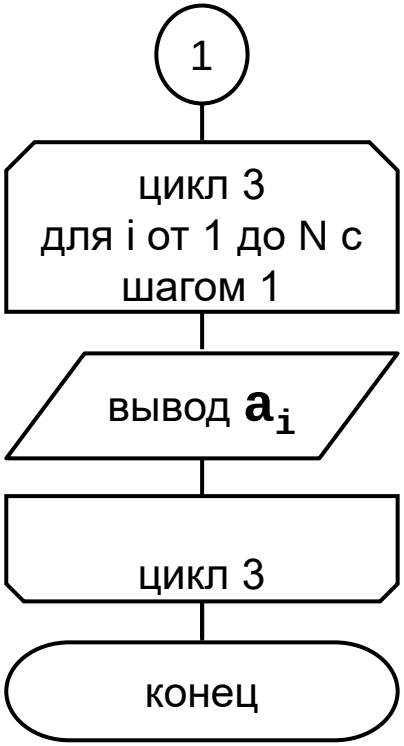
на предыдущем слайде

При изменении константы N остальная программа не должна изменяться!

Схема программы



Выделение памяти и способ обращения к элементам массива в языке программирования к алгоритму не относятся



Динамический массив

- **Размер** массива может быть задан **во время выполнения** программы.
- **Размер** массива **можно изменять**.
- Максимально возможный размер массива определяется объемом свободной оперативной памяти.
- **Именем массива** служит имя **указателя**, в котором хранится адрес начала блока
- После того, как массив перестал быть нужным, память необходимо освободить!

Программа с динамическим массивом

```
#include <stdio.h>
#include <stdlib.h>  /* #include <alloc.h>*/
#define N 5
int main()
{
    int * a, i;
    a = malloc ( N * sizeof(int));
    /* ввод элементов массива */
    /* обработка массива */
    /* вывод результата */
    free (a);
    return 0;
}
```

если функции для работы с динамической памятью не в stdlib

выделение памяти

то же, что и в предыдущей версии

освобождение памяти

Функции выделения памяти

```
char * pC;
int * pI;
```

```
#include <alloc.h>
#include <stdlib.h>
```

Выделение произвольного блока памяти

malloc (размер_блока_в_байтах)

```
pC = (char*) malloc (10);
```

выделили 10 байт, в блоке мусор

Выделение блока под массив

calloc (количество_элементов, размер_элемента)

```
pI = (int*) calloc (5, sizeof(int));
```

выделили блок под массив из 5 целых чисел, блок обнулен

! **При недостатке памяти функции возвращают NULL**

Перевыделение и освобождение памяти

Перевыделение памяти

realloc (адрес_имеющегося_блока, новый_размер)

```
char * tmp_p = (char*) realloc (pC, 100);
if ( tmp_p != NULL )
    pC = tmp_p;
else
    /* что делать, если память не выделилась */
```

увеличили размер блока до 100 байт, данные переписали в новый блок

Освобождение памяти

free (адрес_блока)

```
free (pC);
free (pI);
```

После освобождения памяти не забудьте обнулить указатель!

Ошибки при работе с памятью

Обращение к «чужой» области памяти:

память не была выделена, а массив используется.

Что делать: проверять указатель на NULL.

Блок памяти освобожден повторно:

структура памяти нарушена, может быть все, что угодно.

Что делать: в указатель, хранящий адрес удаленного блока, записывать NULL.

Утечка памяти:

ненужная память не освобождается.

Что делать: убирать «мусор».

Статические и динамические массивы

критерий	массив	
	статический	динамический
создание	`int a [N]`	`int *a=malloc(N*sizeof(int))`
адрес первого элемента	`&a[0]` `a`	`&a[0]` `a`
адрес i-го элемента	`&a[i]` `a+i`	`&a[i]` `a+i`
обращение к элементу	`a[i]` `*(a+i)`	`a[i]` `*(a+i)`
`sizeof(a)`	`N*sizeof(int)`	`sizeof(int*)`
тип `&a`	`int(*)[N]`	`int**`