

Программирование на языке Си Часть II

Массивы

Массив – это группа однотипных элементов, имеющих общее имя и расположенных в памяти рядом.

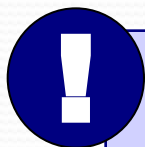
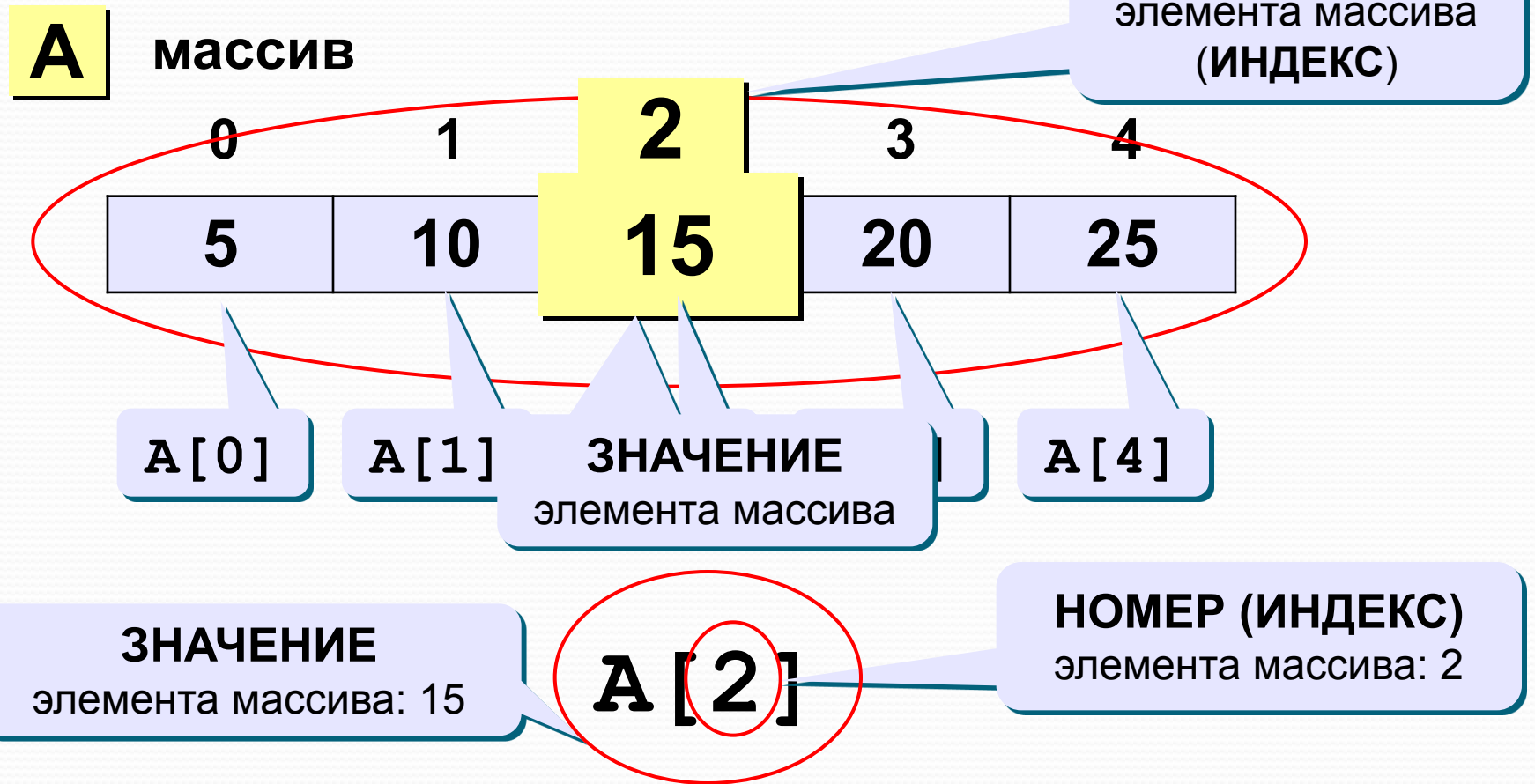
Особенности:

- все элементы имеют **один тип**
- весь массив имеет **одно имя**
- все элементы расположены в памяти **рядом**

Примеры:

- список учеников в классе
- квартиры в доме
- школы в городе
- данные о температуре воздуха за год

Массивы



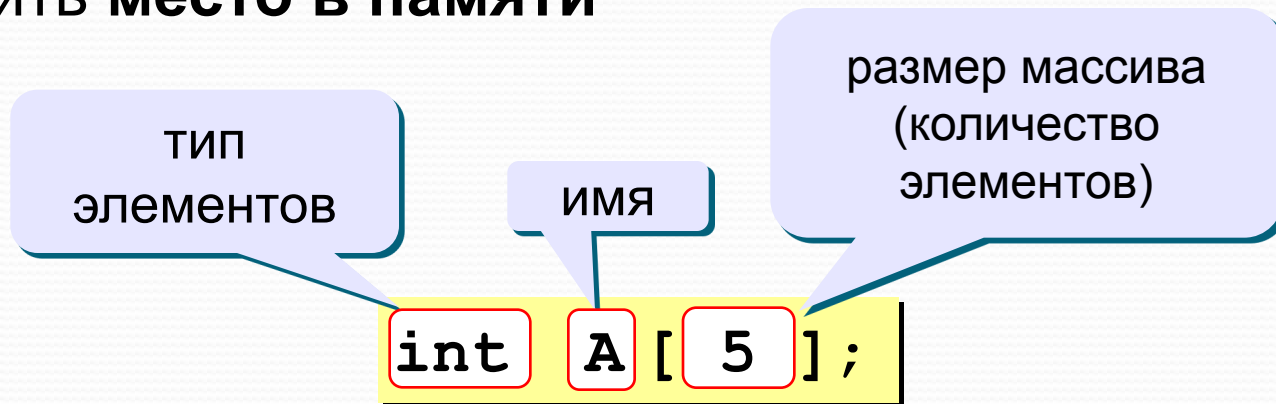
Нумерация элементов массива в Си начинается с **НУЛЯ!**

Объявление массивов

Зачем объявлять?

- определить **ИМЯ** массива
- определить **ТИП** массива
- определить **ЧИСЛО ЭЛЕМЕНТОВ**
- выделить **МЕСТО В ПАМЯТИ**

Пример:



Размер через константу:

```
const int N =  
5;  
int A [ N ] ;
```

Объявление массивов

Еще примеры:

```
int X[10], Y[10];  
float zz, A[20];  
char s[80];
```

С присвоением начальных значений:

```
int A[4] = { 8, -3, 4, 6 };  
float B[2] = { 1. };  
char C[3] = { 'A', '1', 'Ю' };
```

остальные
нулевые!



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

Что неправильно?

```
const int N = 10;  
float A[N];
```

```
int X[4.5];
```

```
int A[10];  
A[10] = 0;
```

```
float X[5];  
int n = 1;  
X[n-2] = 4.5;  
X[n+8] = 12.;
```

**ВЫХОД ЗА ГРАНИЦЫ
МАССИВА**
(стираются данные
в памяти)

дробная часть
отбрасывается
(ошибки нет)

```
int X[4];  
X[2] = 4.5;
```

```
float A[2] = { 1, 3.8 };
```

```
float B[2] = { 1., 3.8, 5.5 };
```

Массивы

Объявление:

```
const int N = 5;  
int A[N], i;
```

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

По

Вь

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

```
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>
#include <conio.h>
main()
{
  const int N = 5;
  int A[N], i;
  // ввод элементов массива
  // обработка массива
  // вывод результата
  getch();
}
```

на предыдущих
слайдах

Задания

«4»: Ввести с клавиатуры массив из 5 элементов, найти среднее арифметическое всех элементов массива.

Пример:

Введите пять чисел:

4 15 3 10 14

среднее арифметическое 9.200

«5»: Ввести с клавиатуры массив из 5 элементов, найти минимальный из них.

Пример:

Введите пять чисел:

4 15 3 10 14

минимальный элемент 3

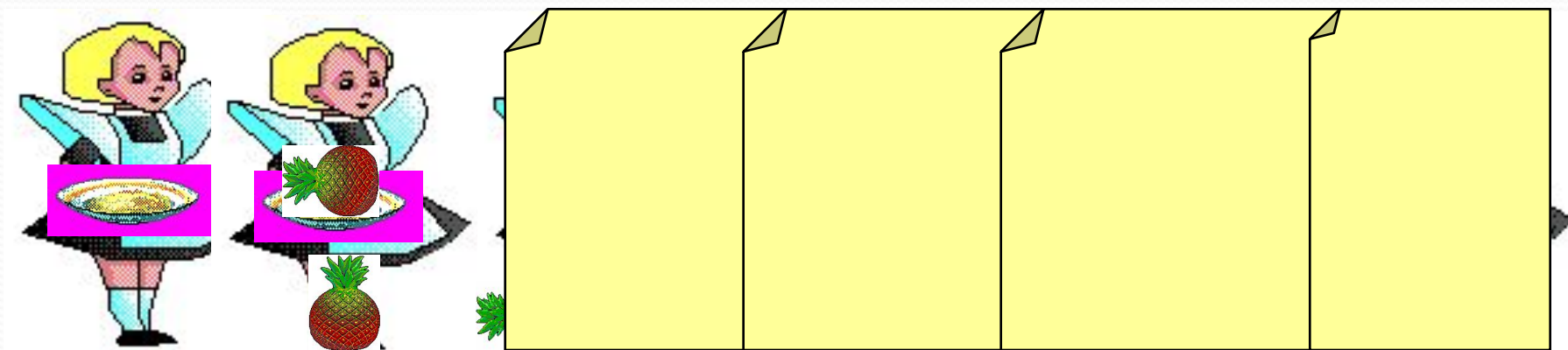


При изменении константы N остальная программа не должна изменяться!

Максимальный элемент

Задача: найти в массиве максимальный элемент.

Алгоритм:



Псевдокод:

```
// считаем, что элемент A[0] – максимальный
for ( i=1; i < N; i++ )
    if ( A[i] > максимального )
        // запомнить новый максимальный элемент A[i]
```



Почему цикл от $i=1$?

Максимальный элемент

Дополнение: как найти номер максимального элемента?

```

// пока A[0] – максимальный
iMax = 0;
for ( i=1; i < N; i++ ) // проверяем остальные
    if ( A[i] > A[iMax] ) { // нашли новый
        // запомнить A[i]
        iMax = i; // запомнить i
    }
```



Как упростить?

По номеру элемента **iMax** всегда можно найти его значение **A[iMax]**. Поэтому везде меняем **max** на **A[iMax]** и убираем переменную **max**.

Заполнение случайными числами

```
#include <stdlib.h> // случайные числа
```

RAND_MAX – максимальное случайное целое число
(обычно `RAND_MAX = 32767`)

Случайное целое число в интервале `[0,RAND_MAX]`

```
x = rand(); // первое число
```

```
x = rand(); // уже другое число
```

Установить начальное значение последовательности:

```
srand ( 345 ); // начнем с 345
```

Целые числа в заданном интервале

Целые числа в интервале $[0, N-1]$:

```
int random(int N) {  
    return rand() % N;  
}
```

Примеры:

```
x = random ( 100 ) ;    // интервал [0, 99]  
x = random ( z ) ;     // интервал [0, z-1]
```

Целые числа в интервале $[a, b]$:

```
x = random ( z ) + a ;    // интервал [a, z-1+a]  
x = random ( b - a + 1 ) + a ; // интервал [a, b]
```

Программа

```
#include <stdio.h>
```

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

```
main()
```



Что дает `const`?

```
{
```

```
const int N = 5;
```

```
int A[N], i, iMax;
```

```
    // заполнить случайными числами [100,150]
```

```
    // найти максимальный элемент и его номер
```

```
printf("\nМаксимальный элемент A[%d] = %d",  
        iMax, A[iMax]);
```

```
getch();
```

```
}
```

на предыдущих
слайдах

Задания

«4»: Заполнить массив из 10 элементов случайными числами в интервале $[-10..10]$ и найти в нем максимальный и минимальный элементы и их номера.

Пример:

Исходный массив:

4 -5 3 10 -4 -6 8 -10 1 0

максимальный $a[4]=10$

минимальный $a[8]=-10$

«5»: Заполнить массив из 10 элементов случайными числами в интервале $[-10..10]$ и найти в нем два максимальных элемента и их номера.

Пример:

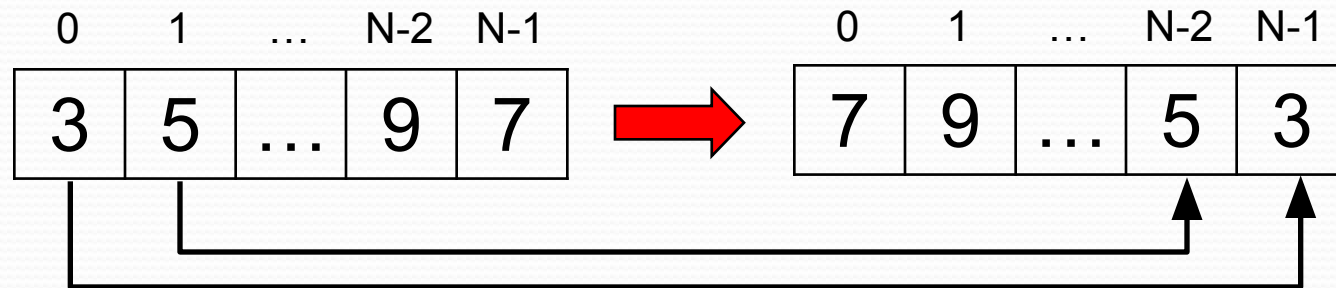
Исходный массив:

4 -5 3 10 -4 -6 8 -10 1 0

максимальные $a[4]=10, a[7]=8$

Реверс массива

Задача: переставить элементы массива в обратном порядке (выполнить инверсию).



Алгоритм:

сумма индексов $N-1$

поменять местами $A[0]$ и $A[N-1]$, $A[1]$ и $A[N-2]$, ...

Псевдокод:

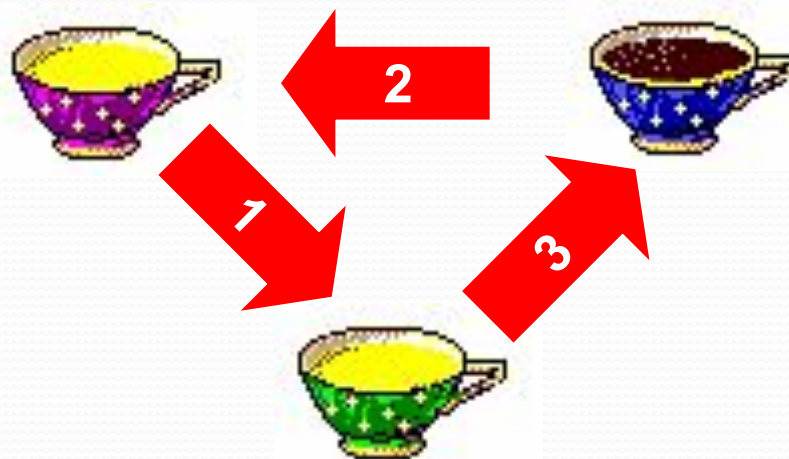
```
for ( i = 0; i < N / 2 ; i++ )  
    // поменять местами A[i] и A[N-1-i]
```



Что неверно?

Как переставить элементы?

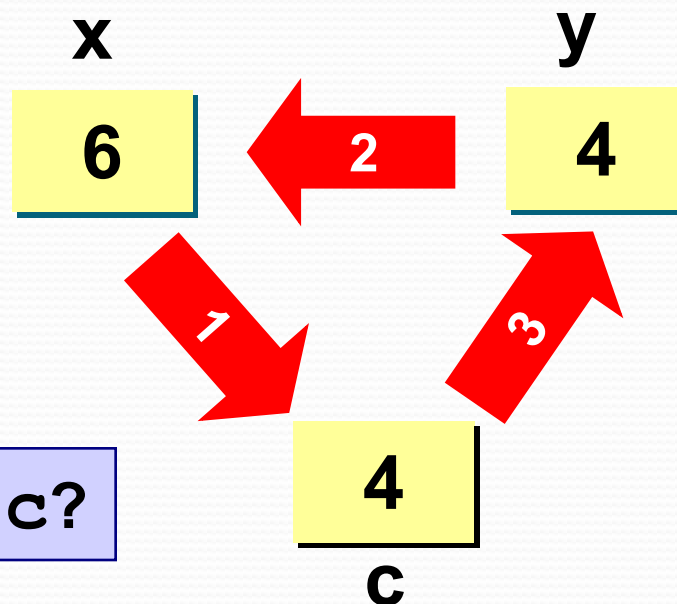
Задача: поменять местами содержимое двух чашек.



Задача: поменять местами содержимое двух ячеек памяти.



```
c = x;  
x = y;  
y = c;
```



Можно ли обойтись без `c`?

Программа

```
main ()
{
    const int N=10;
    int A[N], i, c;
    // заполнить массив
    // вывести исходный массив
    for ( i=0; i<N/2; i++ ) {
        c=A[i];
        A[i]=A[N-1-i];
        A[N-1-i]=c;
    }
    // вывести полученный массив
}
```

Задания

«4»: Заполнить массив из 10 элементов случайными числами в интервале $[-10..10]$ и выполнить инверсию отдельно для 1-ой и 2-ой половин массива.

Пример:

Исходный массив:

4 -5 3 10 -4 -6 8 -10 1 0

Результат:

-4 10 3 -5 4 0 1 -10 8 -6

«5»: Заполнить массив из 12 элементов случайными числами в интервале $[-12..12]$ и выполнить инверсию для каждой трети массива.

Пример:

Исходный массив:

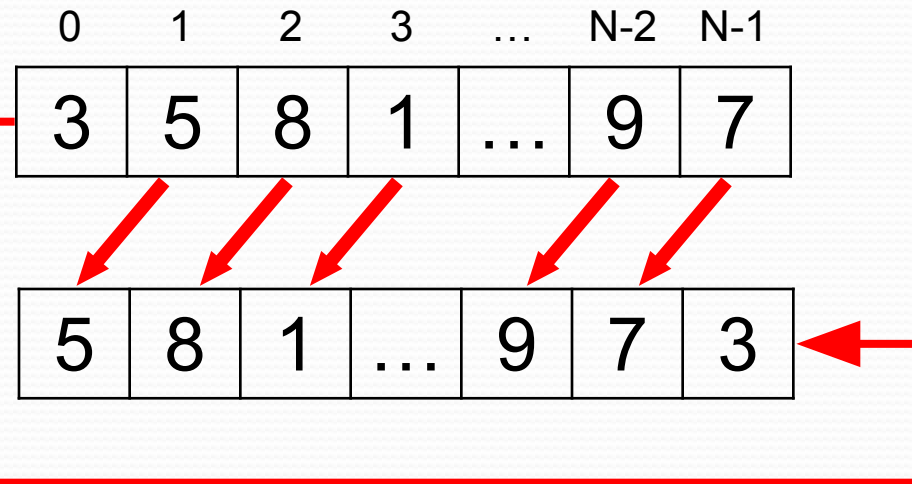
4 -5 3 10 -4 -6 8 -10 1 0 5 7

Результат:

10 3 -5 4 -10 8 -6 -4 7 5 0 1

Циклический сдвиг

Задача: сдвинуть элементы массива влево на 1 ячейку, первый элемент становится на место последнего.



Алгоритм:

$A[0]=A[1]; A[1]=A[2]; \dots A[N-2]=A[N-1];$

Цикл:

почему не N ?

```
for ( i = 0; i < N-1; i++)  
    A[i] = A[i+1];
```



Что неверно?

Программа

```
main()
{
    const int N = 10;
    int A[N], i, c;
    // заполнить массив
    // вывести исходный массив
    c = A[0];
    for ( i = 0; i < N-1; i++)
        A[i] = A[i+1];
    A[N-1] = c;
    // вывести полученный массив
}
```

Задания

«4»: Заполнить массив из 10 элементов случайными числами в интервале $[-10..10]$ и выполнить циклический сдвиг ВПРАВО.

Пример:

Исходный массив:

4 -5 3 10 -4 -6 8 -10 1 0

Результат:

0 4 -5 3 10 -4 -6 8 -10 1

«5»: Заполнить массив из 12 элементов случайными числами в интервале $[-12..12]$ и выполнить циклический сдвиг ВПРАВО на 4 элемента.

Пример:

Исходный массив:

4 -5 3 10 -4 -6 8 -10 1 0 5 7

Результат:

1 0 5 7 4 -5 3 10 -4 -6 8 -10