

Основы программирования (на языке Си)

Тема 10. Массивы

Массивы

Массив – тип данных, состоящий из группы однотипных элементов, имеющих общее имя и расположенных в памяти рядом.

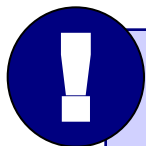
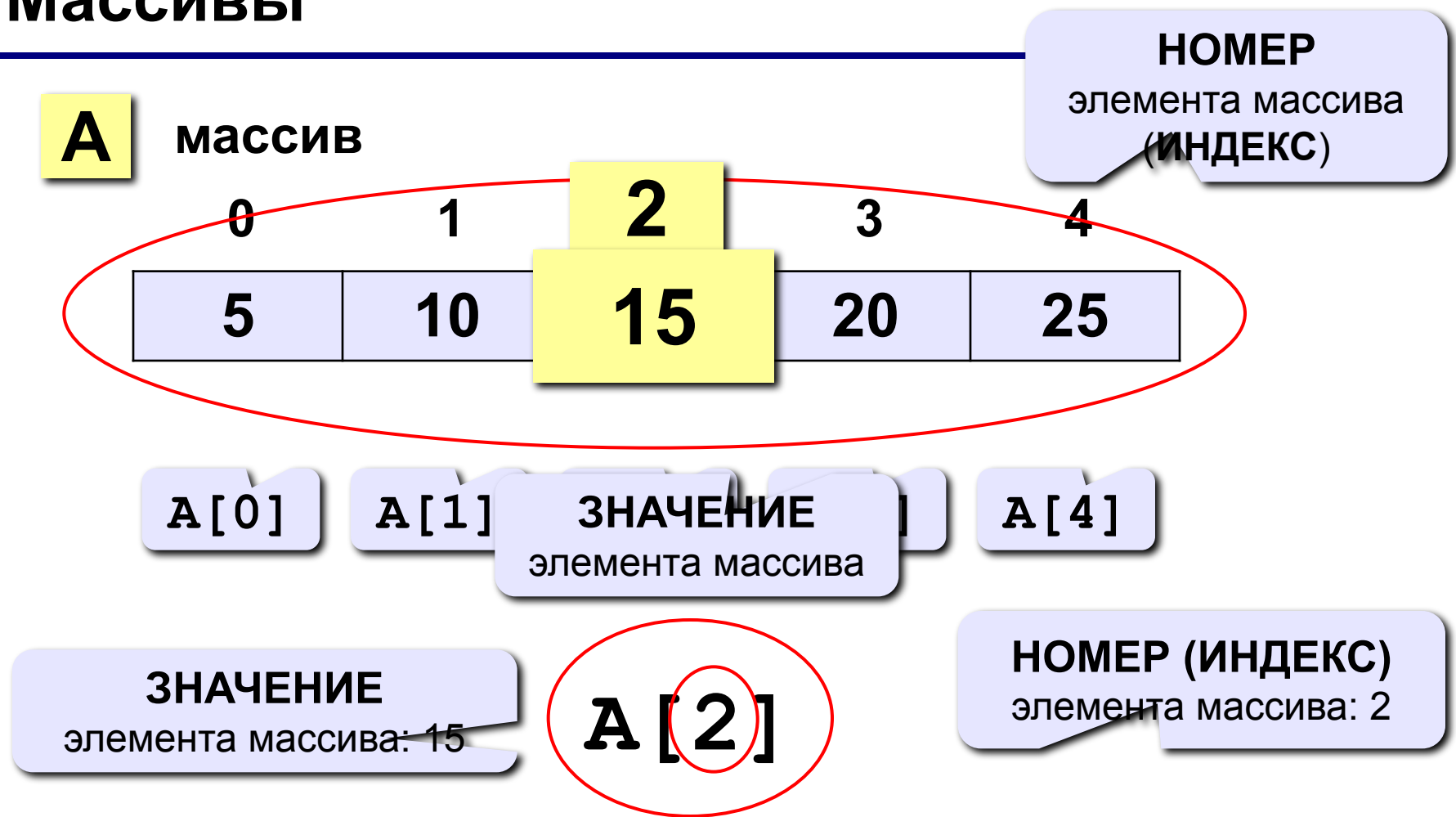
Особенности:

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

Примеры:

- список студентов в группе
- 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:

ТИП
элементов

ИМЯ

размер массива
(количество
элементов)

```
int A [ 5 ] ;
```

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?

```
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

«А»: 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

«В»: 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!

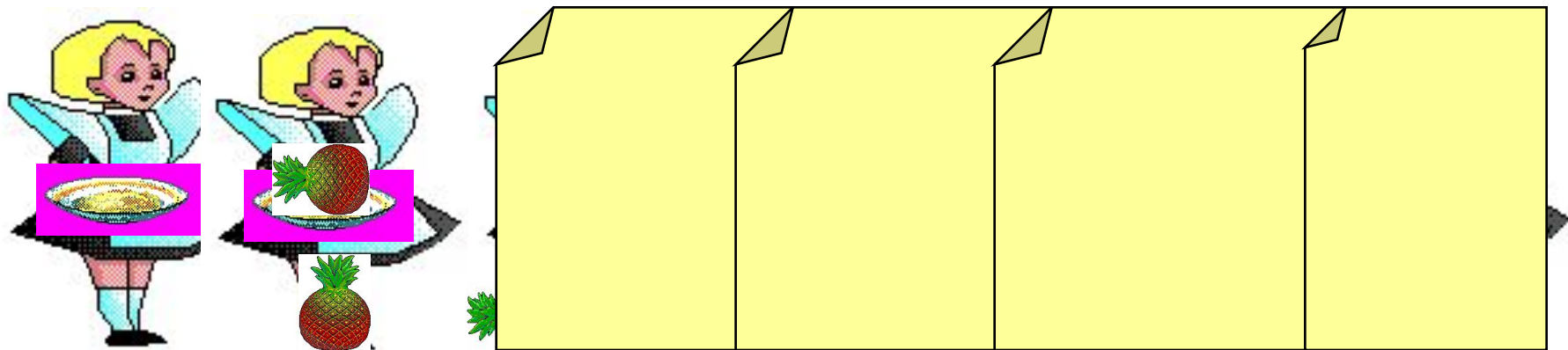
Основы программирования (на языке Си)

Тема 11. Поиск максимального элемента массива

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 ); // установка зерна генератора
```

```
srand ( time (NULL) ); // зерно равно
```

```
// количеству секунд, прошедших с 01.01.1970
```

необходимо подключить

```
#include <time.h>
```

Заполнение случайными числами

```
#include <stdio.h>
#include <stdlib.h>
main()
{
    const int N = 10;
    int A[N], i;
    srand ( 100 );
    printf("Исходный массив:\n");
    for (i = 0; i < N; i++) {
        A[i] = rand()%101 - 50;
        printf("%4d", A[i]);
    }
    ...
}
```



Какой интервал?



Чему равно зерно?

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

«А»: 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 $a[8]=-10$

«В»: 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

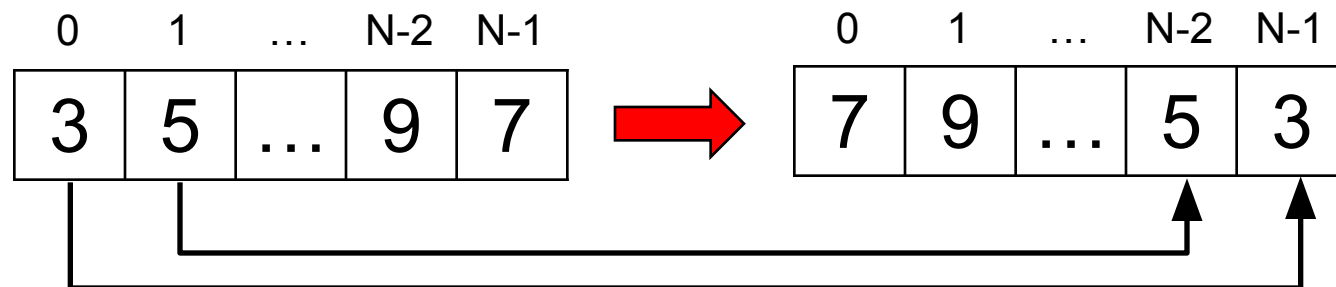
максимальные $a[4]=10$, $a[7]=8$

Основы программирования (на языке Си)

Тема 12. Обработка массивов

Реверс массива

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



Алгоритм:

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?

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

«А»: 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
----	----	---	----	---	---	---	-----	---	----

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

Пример:

Исходный массив:

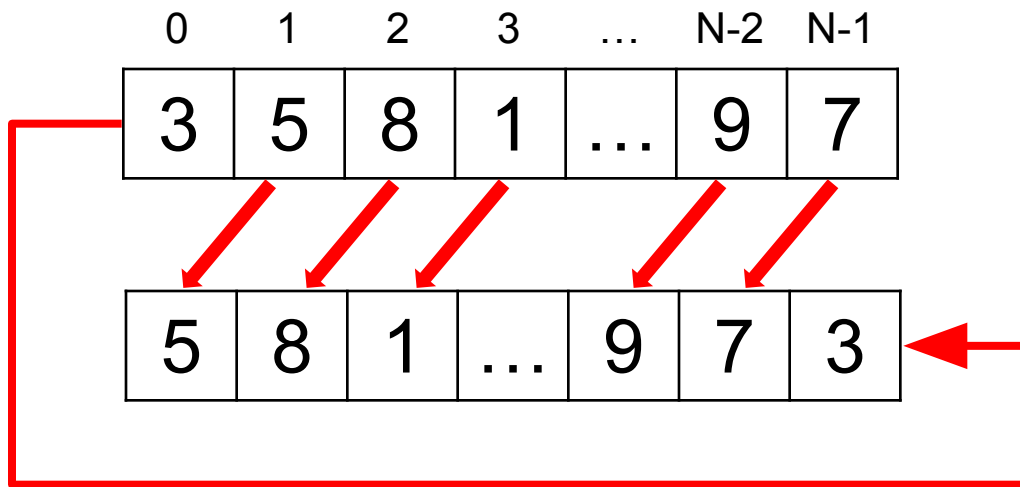
4	-5	3	10	-4	-6	8	-10	1	0	5	7
---	----	---	----	----	----	---	-----	---	---	---	---

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

Основы программирования (на языке Си)

Тема 13. Сортировка массивов

Сортировка

Сортировка – это расстановка элементов массива в заданном порядке (по возрастанию, убыванию, последней цифре, сумме делителей, ...).

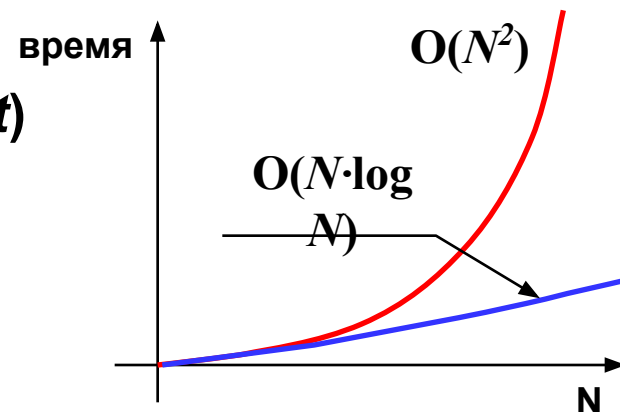
Задача: переставить элементы массива в порядке возрастания.

Алгоритмы:

- простые и понятные, но неэффективные для больших массивов
 - **метод пузырька**
 - **метод выбора**
- сложные, но эффективные
 - **«быстрая сортировка» (*Quick Sort*)**
 - **сортировка «кучей» (*Heap Sort*)**
 - **сортировка слиянием**
 - **пирамидальная сортировка**

сложность $O(N^2)$

сложность $O(N \cdot \log N)$



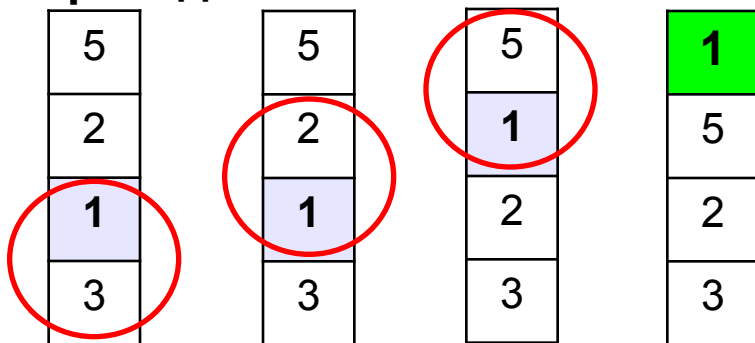
Метод пузырька

Идея – пузырек воздуха в стакане воды поднимается со дна вверх.

Для массивов – самый маленький («легкий») элемент перемещается вверх («всплывает»).

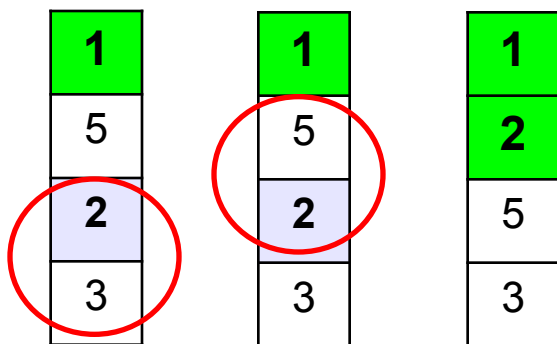
1-ый

проход

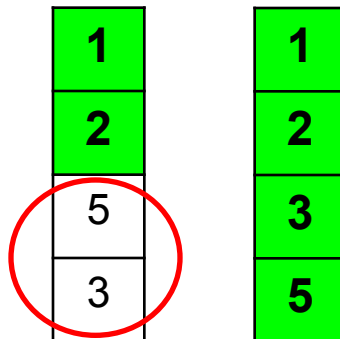


- начиная снизу, сравниваем два соседних элемента; если они стоят «неправильно», меняем их местами
- за 1 проход по массиву **один** элемент (самый маленький) становится на свое место

2-ой проход



3-ий проход



Для сортировки массива из N элементов нужен N-1 проход (достаточно поставить на свои места N-1 элементов).

Программа (1-ый проход)

0	5
1	2
...	...
N-2	6
N-1	3

сравниваются пары

$A[N-2]$ и $A[N-1]$,

$A[N-3]$ и $A[N-2]$

...

$A[0]$ и $A[1]$

$A[j]$ и $A[j+1]$

```
for ( j = N-2; j >= 0 ; j-- )
    if ( A[j] > A[j+1] ) {
        c = A[j];
        A[j] = A[j+1];
        A[j+1] = c;
    }
```

Программа (следующие проходы)

2-ой
проход

0	1
1	5
...	...
N-2	3
N-1	6

(i+1)-ый
проход



A[0] уже на своем месте!

```
for ( j = N-2; j >= 1; j-- )  
    if ( A[j] > A[j+1] ) {  
        c = A[j];  
        A[j] = A[j+1];  
        A[j+1] = c;  
    }
```

```
for ( j = N-2; j >= i; j-- )  
    ...
```

Программа

```
main()
```

```
{
```

```
    const int N=10;
```

```
    int A[N], i, j, c;
```

```
    // заполнить массив
```

```
    // вывести исходный массив
```

```
    for (i = 0; i < N-1; i++) {
```

```
        for (j = N-2; j >= i; j --)
```

```
            if (A[j] > A[j+1]) {
```

```
                c = A[j];
```

```
                A[j] = A[j+1];
```

```
                A[j+1] = c;
```

```
            }
```

```
        }
```

```
    // вывести полученный массив
```

```
}
```



Почему цикл для $i < N-1$,
а не $i < N$?

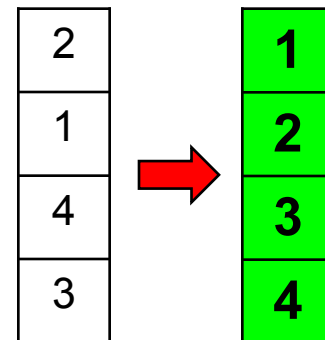
элементы выше
 $A[i]$ уже
поставлены

меняем $A[j]$
и $A[j+1]$

Метод пузырька с флажком

Идея – если при выполнении метода пузырька не было обменов, массив уже отсортирован и остальные проходы не нужны.

Реализация: переменная-флаг, показывающая, был ли обмен; если она равна **0**, то выход.



```

do {
    flag = 0; // сбросить флаг
    for (j = N-2; j >= 0; j --)
        if (A[j] > A[j+1]) {
            c = A[j];
            A[j] = A[j+1];
            A[j+1] = c;
            flag = 1; // поднять флаг
        }
    }
while (flag != 0); // выход при flag = 0
  
```



Как улучшить?

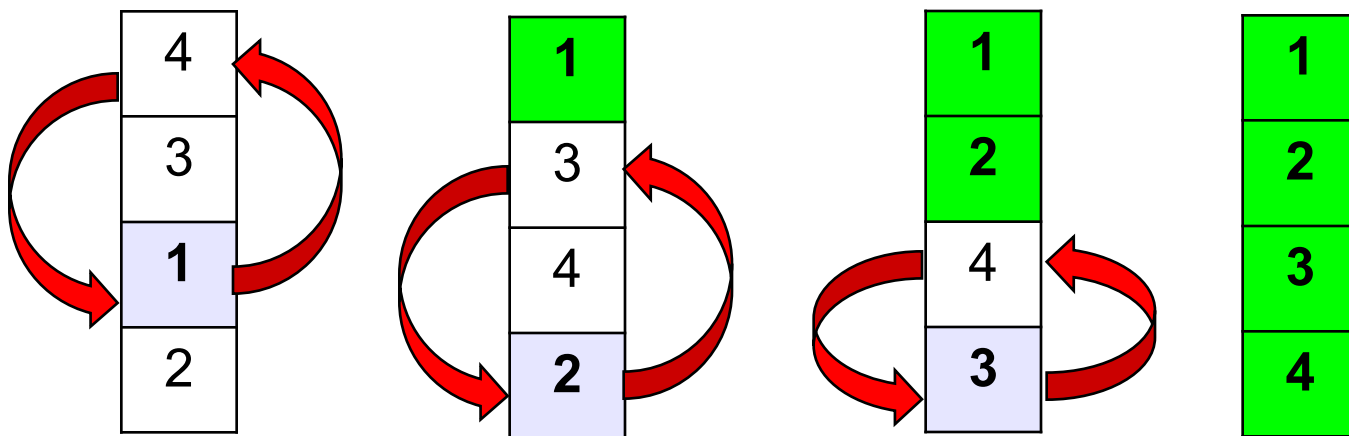
Метод пузырька с флажком

```
i = 0;
do {
    flag = 0; // сбросить флаг
    for ( j = N-2; j >= i; j-- )
        if ( A[j] > A[j+1] ) {
            c = A[j];
            A[j] = A[j+1];
            A[j+1] = c;
            flag = 1; // поднять флаг
        }
    i++;
} while ( flag ); // выход при flag = 0
```

Метод выбора

Идея:

- найти минимальный элемент и поставить на первое место (поменять местами с $A[0]$)
- **из оставшихся** найти минимальный элемент и поставить на второе место (поменять местами с $A[1]$), и т.д.



Метод выбора

нужно $N-1$ проходов

```
for ( i = 0; i <  $N-1$ ; i++ ) {  
    nMin = i;  
    for ( j = i; j < N; j++ )  
        if ( A[j] < A[nMin] ) nMin = j;  
    if ( nMin != i ) {  
        c = A[i];  
        A[i] = A[nMin];  
        A[nMin] = c;  
    }  
}
```

ПОИСК МИНИМАЛЬНОГО
ОТ $A[i]$ ДО $A[N-1]$

если нужно,
переставляем



Можно ли убрать if?

Задания

«А»: Заполнить массив из 10 элементов случайными числами в интервале [0..100] и отсортировать его по последней цифре.

Пример:

Исходный массив:

14 25 13 30 76 58 32 11 41 97

Результат:

30 11 41 32 13 14 25 76 97 58

«В»: Заполнить массив из 10 элементов случайными числами в интервале [0..100] и отсортировать первую половину по возрастанию, а вторую – по убыванию.

Пример:

Исходный массив:

14 25 13 30 76 | 58 32 11 41 97

Результат:

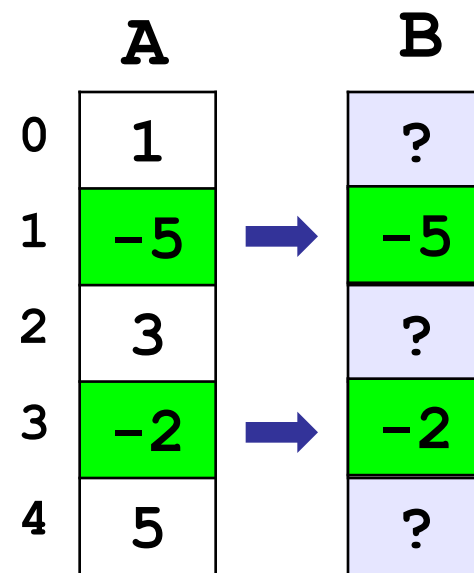
13 14 25 30 76 | 97 58 41 32 11

Формирование массива по условию

Задача – найти в массиве элементы, удовлетворяющие некоторому условию (например, отрицательные), и скопировать их в другой массив.

Примитивное решение:

```
const int N = 5;  
int A[N], B[N];  
// здесь заполнить массив A  
for( i=0; i<N; i++)  
    if( A[i] < 0 ) B[i] = A[i];
```

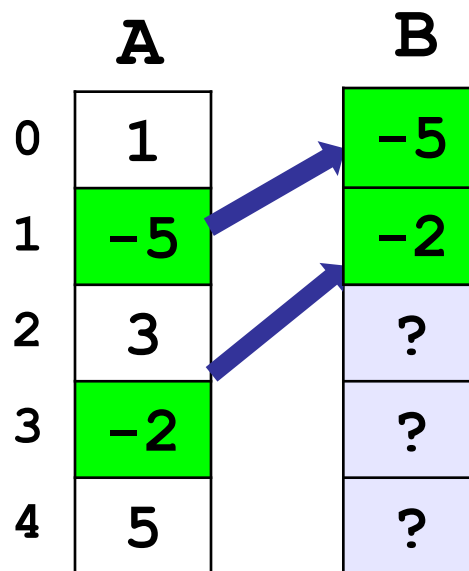


- выбранные элементы не рядом, не в начале массива
- непонятно, как с ними работать

Формирование массива по условию

Решение: ввести счетчик найденных элементов `count`, очередной элемент ставится на место `B[count]`.

```
int A[N], B[N], count = 0;  
// здесь заполнить массив A  
for( i=0; i < N; i++ )  
    if( A[i] < 0 ) {  
        B[count] = A[i];  
        count ++;  
    }  
  
// вывод массива B  
for( i=0; i < count; i++ )  
    printf( "%d\n", tB[i] );
```



Задания

«А»: Заполнить массив случайными числами и отобразить в другой массив все числа, у которых вторая с конца цифра (число десятков) – ноль.

Пример:

Исходный массив:

40 105 203 1 14

Результат:

105 203 1

«В»: Заполнить массив случайными числами и выделить в другой массив все числа, которые встречаются более одного раза.

Пример:

Исходный массив:

4 1 2 1 11 2 34

Результат:

1 2

Основы программирования (на языке Си)

Тема 14. Матрицы

Матрицы

Задача: запомнить положение фигур на шахматной доске.



1



2



3



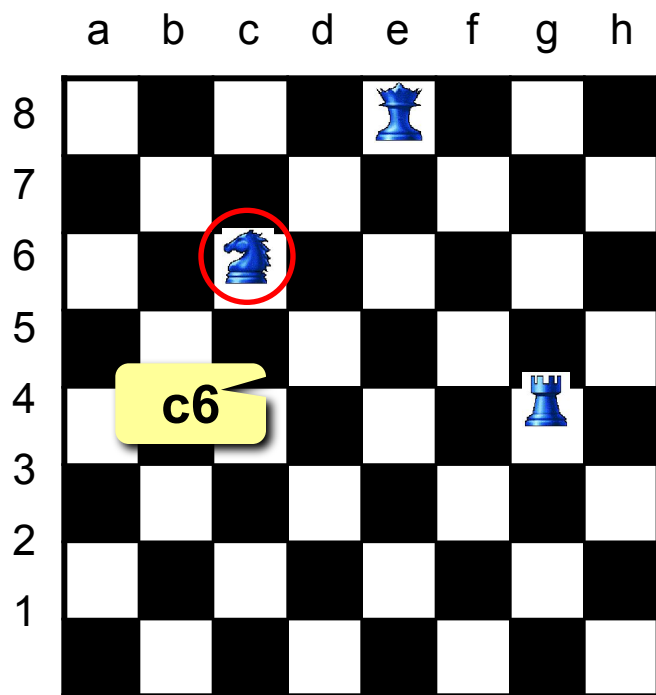
4



5



6



	0	1	2	3	4	5	6	7
7	0	0	0	0	2	0	0	0
6	0	0	0	0	0	0	0	0
5	0	0	3	0	0	0	0	0
4	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	4	0
2	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0

A[5][2]

Матрицы

Матрица – это прямоугольная таблица однотипных элементов.

Матрица – это массив, в котором каждый элемент имеет два индекса (номер строки и номер столбца).

A

	0	1	2	3	4
0	1	4	7	3	6
1	2	-5	0	15	10
2	8	9	11	12	20

столбец 2

строка 1

ячейка **A[2][3]**

Матрицы

Объявление:

```
const int N = 3, M = 4;
int A[N][M];
float a[2][2] = {{3.2, 4.3}, {1.1, 2.2}};
char sym[2][2] = { 'a', 'b', 'c', 'd' };
```

Ввод с клавиатуры:

```
for ( j = 0; j < M; j ++ )
    for ( i = 0; i < N; i ++ ) {
        printf ( "A[%d][%d]=", i, j );
        scanf ( "%d", &A[i][j] );
    }
```

i	j		
		A[0][0]	2
		A[0][1]	5
		A[0][2]	4
]=	4
		A[2][3]	5
]=	4



Если переставить циклы?

Матрицы

Заполнение случайными числами

```
for ( i = 0; i < N; i ++ )
    for ( j = 0; j < M; j ++ )
        A[i][j] = random(25) - 10;
```

цикл по строкам

интервал?

цикл по столбцам

Вывод на экран

```
for ( i = 0; i < N; i ++ ) {
    for ( j = 0; j < M; j ++ )
        printf("%5d", A[i][j]);
    printf("\n");
}
```

ВЫВОД строки

12	25	1	13
156	1	12	447
1	456	222	23

в той же строке

перейти на
новую строку



Если переставить циклы?

Обработка всех элементов матрицы

Задача: заполнить матрицу из 3 строк и 4 столбцов случайными числами и вывести ее на экран. Найти сумму элементов матрицы.

```
main()  
{  
    const int N=3, M=4;  
    int A[N][M], i, j, S=0;  
    ... // заполнение матрицы и вывод на экран  
    for ( i=0; i<N; i++)  
        for ( j=0; j<M; j++)  
            S += A[i][j];  
    printf("Сумма элементов матрицы S=%d", S);  
}
```

Задания

Заполнить матрицу из 8 строк и 5 столбцов случайными числами в интервале $[-10, 10]$ и вывести ее на экран.

«4»: Найти минимальный и максимальный элементы в матрице их номера. Формат вывода:

Минимальный элемент $A[3][4] = -6$

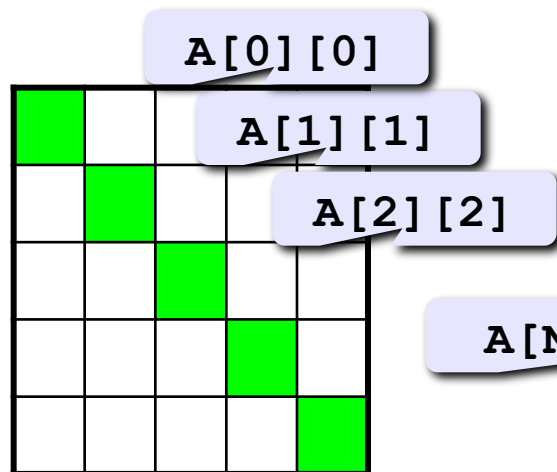
Максимальный элемент $A[2][2] = 10$

«5»: Вывести на экран строку, сумма элементов которой максимальна. Формат вывода:

Строка 2: 3 5 8 9 8

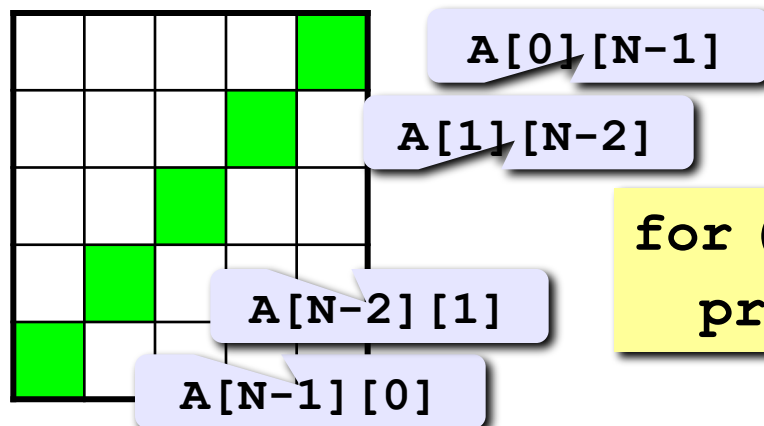
Операции с матрицами

Задача 1. Вывести на экран главную диагональ квадратной матрицы из N строк и N столбцов.



```
for ( i = 0; i < N; i ++ )
    printf ( "%5d", A[i][i] );
```

Задача 2. Вывести на экран вторую диагональ.

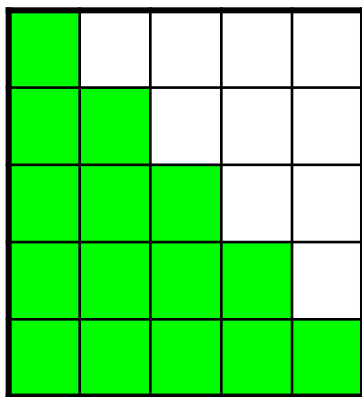


сумма номеров строки и столбца N-1

```
for ( i = 0; i < N; i ++ )
    printf ( "%5d", A[i][N-1-i] );
```

Операции с матрицами

Задача 3. Найти сумму элементов, стоящих на главной диагонали и ниже ее.



Одиночный цикл или вложенный?

строка 0: $A[0][0]$

строка 1: $A[1][0] + A[1][1]$

...

строка i : $A[i][0] + A[i][2] + \dots + A[i][i]$

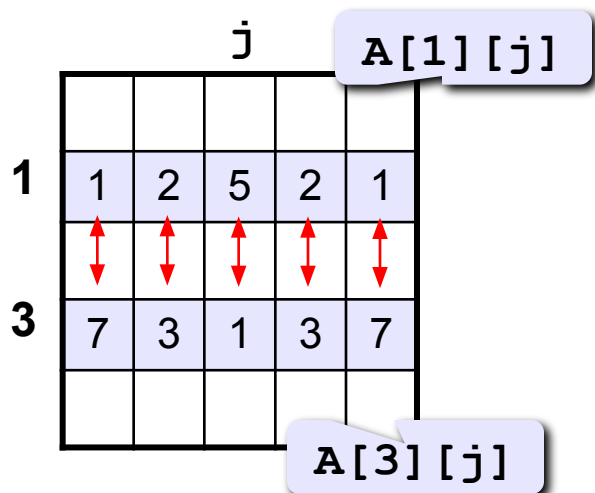
цикл по всем строкам

```
S = 0;
for ( i = 0; i < N; i ++ )
    for ( j = 0; j <= i; j ++ )
        S += A[i][j];
```

складываем нужные
элементы строки i

Операции с матрицами

Задача 4. Перестановка строк или столбцов. В матрице из N строк и M столбцов переставить 1-ую и 3-ю строки.

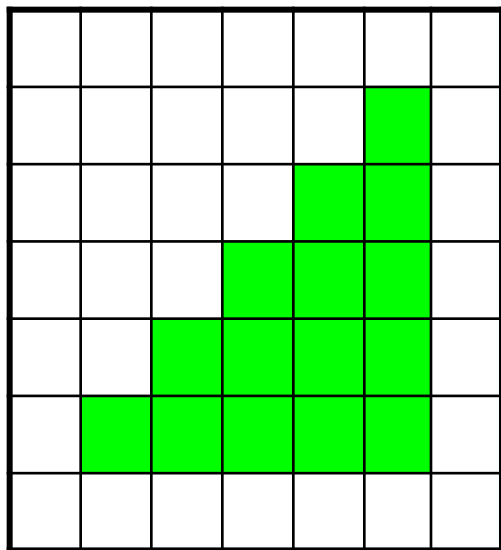


```
for ( j = 0; j <= M; j ++ ) {
    c = A[1][j];
    A[1][j] = A[3][j];
    A[3][j] = c;
}
```

Задача 5. К третьему столбцу добавить шестой.

```
for ( i = 0; i < N; i ++ )
    A[i][3] += A[i][6];
```

«A»:



«B»:

