

Программирование на языке С и С++

Массивы

Алгоритмы обработки массивов

Сортировка

Матрицы

Программирование на языке C++

Массивы

Что такое массив?



Как ввести 10000 переменных?

Массив – это группа переменных одного типа, расположенных в памяти рядом (в соседних ячейках) и имеющих общее имя. Каждая ячейка в массиве имеет уникальный номер (индекс).

Надо:

- выделять память
- записывать данные в нужную ячейку
- читать данные из ячейки

Выделение памяти (объявление)

! Массив = таблица!

```
int A[5];  
double V[8];  
bool L[10];  
char S[80];
```

число
элементов

! Элементы нумеруются
с нуля!

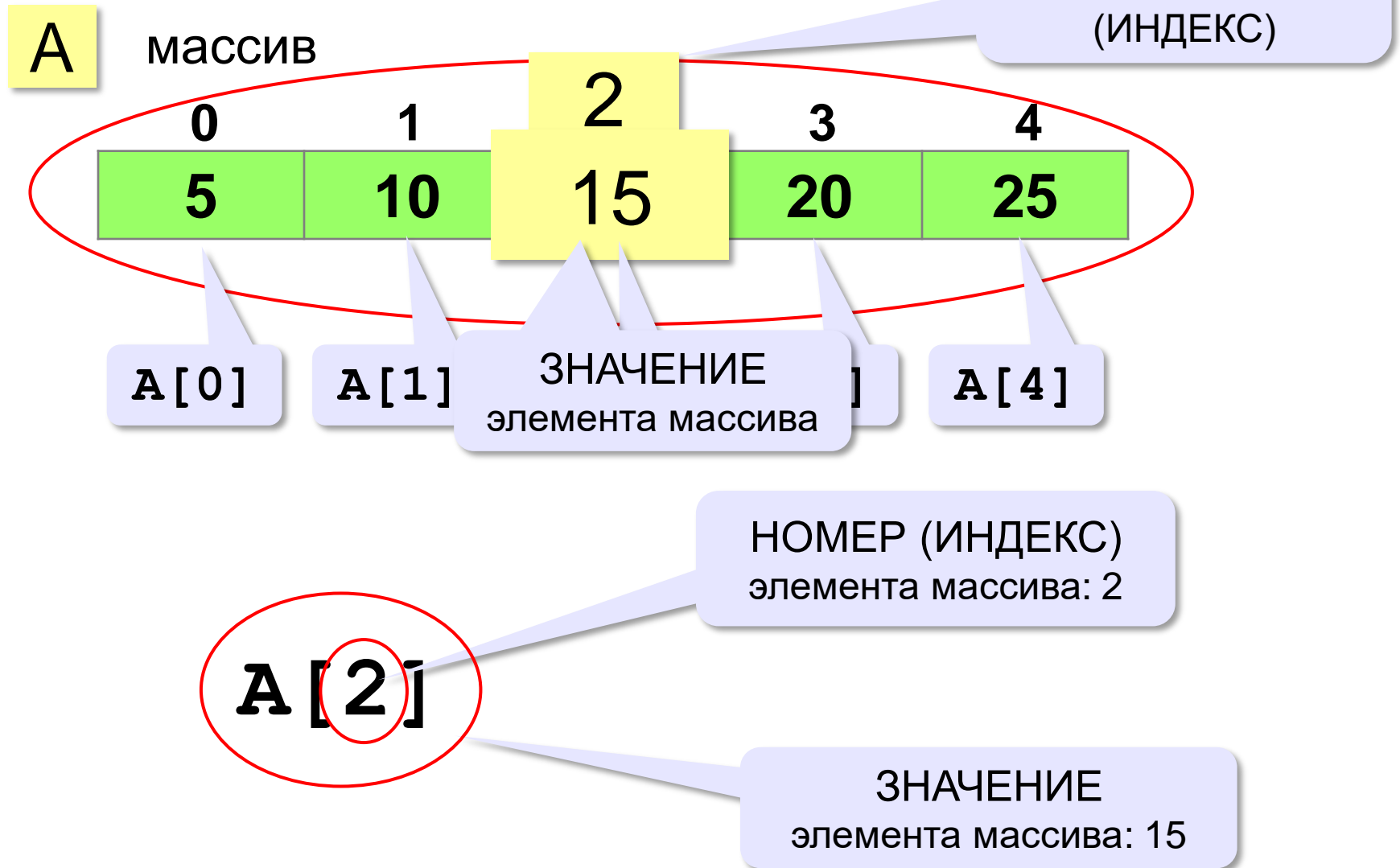
A[0], A[1], A[2], A[3], A[4]

размер через
константу

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

? Зачем?

Обращение к элементу массива



Как обработать все элементы массива?

Объявление:

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

Обработка:

```
// обработать A[0]  
// обработать A[1]  
// обработать A[2]  
// обработать A[3]  
// обработать A[4]
```



1) если N велико (1000, 1000000)?

2) при изменении N программа не должна меняться!

Как обработать все элементы массива?

Обработка с переменной:

```
i = 0;  
// обработать A[i]  
i ++;  
// обработать A[i]  
i ++;  
// обработать A[i]  
i ++;  
// обработать A[i]  
i ++;  
// обработать A[i]  
i ++;
```



Обработка в цикле:

```
i = 0;  
while ( i < N )  
{  
    // обработать A[i]  
    i ++;  
}
```



Цикл с переменной:

```
for( i = 0; i < N; i++ )  
{  
    // обработать A[i]  
}
```

Заполнение массива

```
main()  
{  
    const int N = 10;  
    int A[N];  
    int i;  
    for ( i = 0; i < N; i++ )  
        A[i] = i*i;  
}
```



Чему равен $A[9]$?

Ввод с клавиатуры и вывод на экран

Объявление:

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

Ввод с клавиатуры:

```
for ( i = 0; i < N; i++ )  
{  
    cout << "A[" << i << "]=";  
    cin >> A[i];  
}
```

A[1] = 5
A[2] = 12
A[3] = 34
A[4] = 56
A[5] = 13

Вывод на экран:

```
cout >> "Массив A:\n";  
for ( i = 0; i < N; i++ )  
    cout << A[i] << " ";
```



Зачем пробел?

Заполнение случайными числами

Задача. Заполнить массив (псевдо)случайными целыми числами в диапазоне от 20 до 100.

```
int irand ( int a, int b )  
{  
    return a + rand() % (b - a + 1) ;  
}
```

```
for ( i = 0; i < N; i++ )  
{  
    A[i] = irand ( 20, 100 ) ;  
    cout << A[i] << " " ;  
}
```

Перебор элементов

Общая схема:

```
for ( i = 0; i < N; i++ )  
{  
    ... // сделать что-то с A[i]  
}
```

Подсчёт нужных элементов:

Задача. В массиве записаны данные о росте баскетболистов. Сколько из них имеет рост больше 180 см, но меньше 190 см?

```
count = 0;  
for ( i = 0; i < N; i++ )  
    if ( 180 < A[i] && A[i] < 190 )  
        count ++;
```

Перебор элементов

Среднее арифметическое:

```
int count, sum;
count = 0;
sum = 0;
for ( i = 0; i < N; i++ )
    if ( 180 < A[i] && A[i] < 190 ) {
        count ++;
        sum += A[i];
    }
cout << (float)sum / count;
```



Зачем **float**?

среднее
арифметическое

Задачи

«А»: Заполните массив случайными числами в интервале $[0, 100]$ и найдите среднее арифметическое его значений.

Пример:

Массив :

1 2 3 4 5

Среднее арифметическое 3.000

«В»: Заполните массив случайными числами в интервале $[0, 100]$ и подсчитайте отдельно среднее значение всех элементов, которые < 50 , и среднее значение всех элементов, которые ≥ 50 .

Пример:

Массив :

3 2 52 4 60

Ср. арифм. элементов $[0, 50) : 3.000$

Ср. арифм. элементов $[50, 100] : 56.000$

Задачи

«С»: Заполните массив из N элементов случайными числами в интервале $[1, N]$ так, чтобы в массив обязательно вошли все числа от 1 до N (постройте случайную перестановку).

Пример:

Массив :

3 2 1 4 5

Программирование на языке C++

Алгоритмы обработки массивов

Поиск в массиве

Найти элемент, равный X:

```
i = 0;  
while ( A[i] != X )  
    i ++;  
cout << "A[" << i << "]=" << X;
```



Что плохо?

```
i = 0;  
while ( i < N && A[i] != X )  
    i ++;  
if ( i < N )  
    cout << "A[" << i << "]=" << X;  
else  
    cout << "Не нашли!";
```

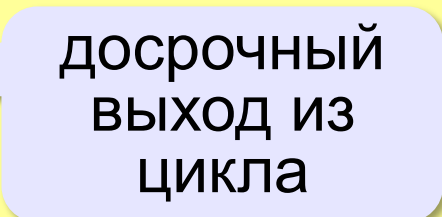


Что если такого нет?

Поиск в массиве

Вариант с досрочным выходом:

```
nX = -1;  
for ( i = 0; i < N; i++ )  
    if ( A[i] == X )  
    {  
        nX = i;  
        break;  
    }  
if ( nX >= 0 )  
    cout << "A[" << nX << "]=" << X;  
else  
    cout << "Не нашли!";
```



досрочный
выход из
цикла

Задачи

«А»: Заполните массив случайными числами в интервале $[0,5]$. Введите число X и найдите все значения, равные X .

Пример:

Массив :

1 2 3 1 2

Что ищем:

2

Нашли: $A[2]=2$, $A[5]=2$

Пример:

Массив :

1 2 3 1 2

Что ищем:

6

Ничего не нашли.

Задачи

«В»: Заполните массив случайными числами в интервале $[0,5]$. Определить, есть ли в нем элементы с одинаковыми значениями, стоящие рядом.

Пример:

Массив :

1 2 3 3 2 1

Есть : 3

Пример:

Массив :

1 2 3 4 2 1

Нет

Задачи

«С»: Заполните массив случайными числами. Определить, есть ли в нем элементы с одинаковыми значениями, не обязательно стоящие рядом.

Пример:

Массив :

3 2 1 3 2 5

Есть: 3, 2

Пример:

Массив :

3 2 1 4 0 5

Нет

Максимальный элемент

```
M = A[0];  
for ( i = 1; i < N; i++ )  
    if ( A[i] > M )  
        M = A[i];  
cout << M;
```



Как найти его номер?

```
M = A[0]; nMax = 0;  
for ( i = 1; i < N; i++ )  
    if ( A[i] > M ) {  
        M = A[i];  
        nMax = i;  
    }
```



Что можно улучшить?

```
cout << "A[" << nMax << "]= " << M;
```

Максимальный элемент и его номер



По номеру элемента можно найти значение!

```
nMax = 0;  
for ( i = 1; i < N; i++ )  
    if ( A[i] > A[nMax] )  
        nMax = i;  
cout << "A[" << nMax << "]=" << A[nMax] ;
```

Задачи

«А»: Заполнить массив случайными числами и найти минимальный и максимальный элементы массива и их номера.

Пример:

Массив :

1 2 3 4 5

Минимальный элемент: $A[1]=1$

Максимальный элемент: $A[5]=5$

«В»: Заполнить массив случайными числами и найти два максимальных элемента массива и их номера.

Пример:

Массив :

5 5 3 4 1

Максимальный элемент: $A[1]=5$

Второй максимум: $A[2]=5$

Задачи

«С»: Введите массив с клавиатуры и найдите (за один проход) количество элементов, имеющих максимальное значение.

Пример:

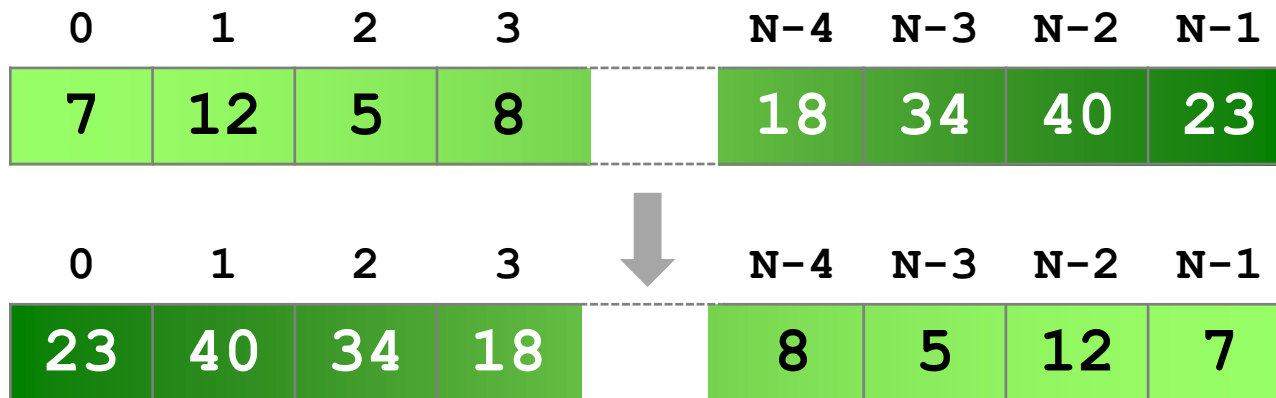
Массив :

3 4 5 5 3 4 5

Максимальное значение 5

Количество элементов 3

Реверс массива



«Простое» решение:

остановиться на середине!

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

?

Что плохо?

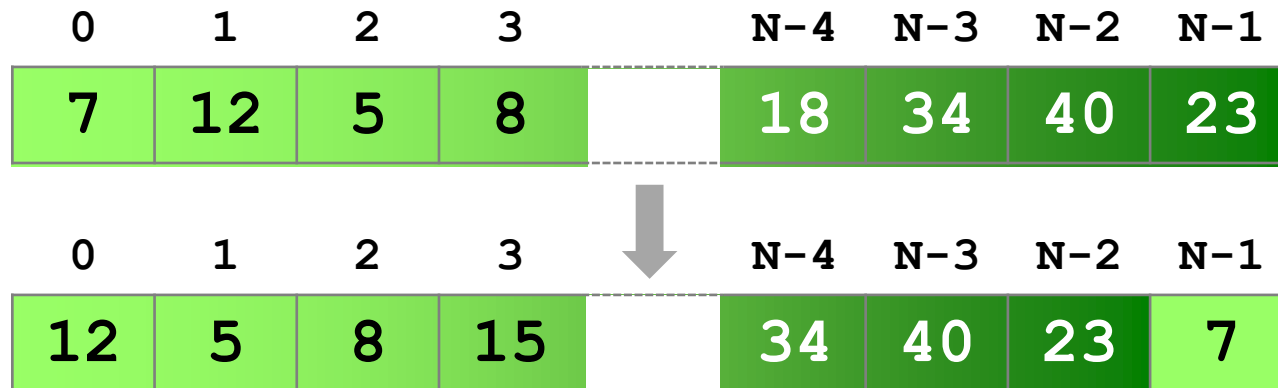
Реверс массива

```
for ( i = 0; i < (N/2); i++ )  
{  
    c = A[i];  
    A[i] = A[N-1-i];  
    A[N-1-i] = c;  
}
```



*Как обойтись без переменной c?

Циклический сдвиг элементов



«Простое» решение:

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



Почему не до N-1?



Что плохо?

Задачи

«А»: Заполнить массив случайными числами и выполнить циклический сдвиг элементов массива вправо на 1 элемент.

Пример:

Массив :

1 2 3 4 5 6

Результат :

6 1 2 3 4 5

«В»: Массив имеет четное число элементов. Заполнить массив случайными числами и выполнить реверс отдельно в первой половине и второй половине.

Пример:

Массив :

1 2 3 4 5 6

Результат :

3 2 1 6 5 4

Задачи

«С»: Заполнить массив случайными числами в интервале $[-100, 100]$ и переставить элементы так, чтобы все положительные элементы стояли в начала массива, а все отрицательные и нули – в конце. Вычислите количество положительных элементов.

Пример:

Массив:

20 -90 15 -34 10 0

Результат:

20 15 10 -90 -34 0

Количество положительных элементов: 3

Отбор нужных элементов

Задача. Отобрать элементы массива **A**,
удовлетворяющие некоторому условию, в массив **B**.

«Простое» решение:

сделать для i от 0 до $N-1$
если условие выполняется для $A[i]$ то
 $B[i] := A[i]$

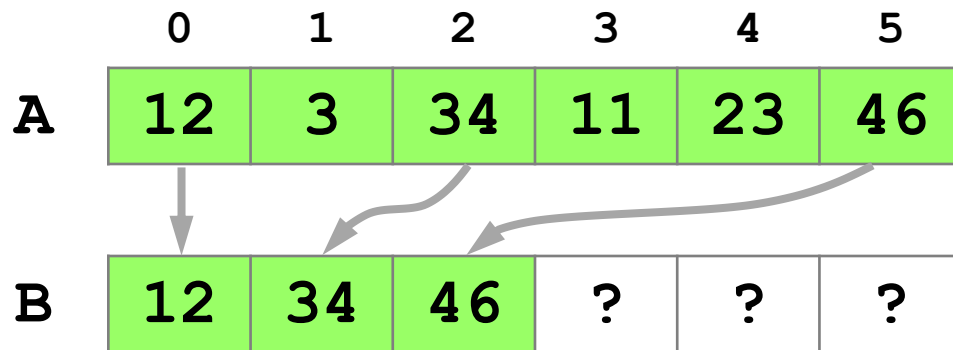


Что плохо?

	0	1	2	3	4	5
A	12	3	34	11	23	46
	↓		↓			↓
B	12	?	34	?	?	46

выбрать чётные
элементы

Отбор нужных элементов



выбрать чётные
элементы

```
count = 0;
for ( i = 0; i < N; i++ )
    if ( A[i] % 2 == 0 )
    {
        B[count] = A[i];
        count++;
    }
```

?

Если A и B – один и
тот же массив?

?

Как вывести на экран?

```
for ( i = 0; i < count; i++ )
    printf ( "%d ", B[i] );
```

Задачи

«А»: Заполнить массив случайными числами в интервале $[-10, 10]$ и отобрать в другой массив все чётные отрицательные числа.

Пример:

Массив А:

-5 6 7 -4 -6 8 -8

Массив В:

-4 -6 -8

«В»: Заполнить массив случайными числами в интервале $[0, 100]$ и отобрать в другой массив все простые числа. Используйте логическую функцию, которая определяет, является ли переданное ей число простым.

Пример:

Массив А:

12 13 85 96 47

Массив В:

13 47

Задачи

«С»: Заполнить массив случайными числами и отобразить в другой массив все числа Фибоначчи. Используйте логическую функцию, которая определяет, является ли переданное ей число числом Фибоначчи.

Пример:

Массив А:

12 13 85 34 47

Массив В:

13 34

Программирование на языке C++

Сортировка

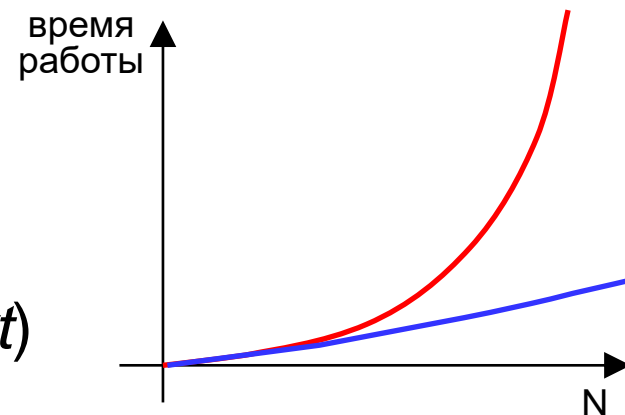
Что такое сортировка?

Сортировка – это расстановка элементов массива в заданном порядке.

...по возрастанию, убыванию, последней цифре, сумме делителей, по алфавиту, ...

Алгоритмы:

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

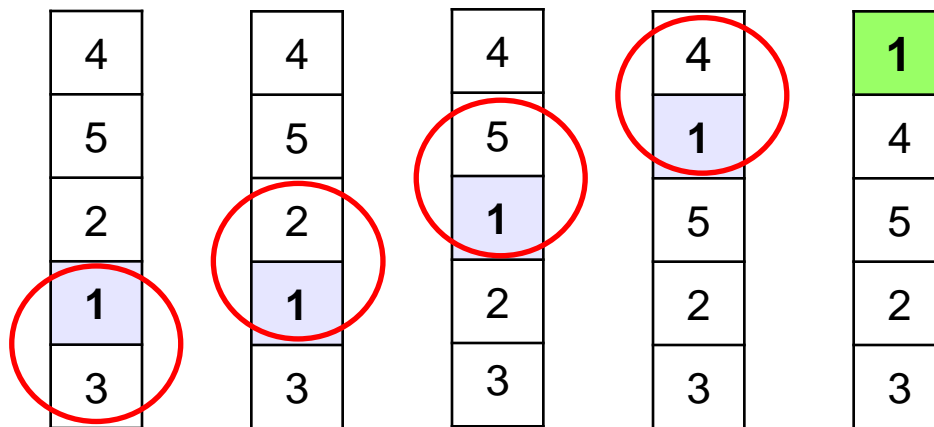


Метод пузырька (сортировка обменами)

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

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

1-й проход:



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

Метод пузырька

2-й проход:

1	1	1	1
4	4	4	2
5	5	2	4
2	2	5	5
3	3	3	3

3-й проход:

1	1	1
2	2	2
4	4	3
5	3	4
3	5	5

4-й проход:

1	1
2	2
3	3
4	4
5	5



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

Метод пузырька

1-й проход:

```
сделать для j от N-2 до 0 шаг -1
    если A[j+1] < A[j] то
        // поменять местами A[j] и A[j+1]
```

2-й проход:

```
сделать для j от N-2 до 1 шаг -1
    если A[j+1] < A[j] то
        // поменять местами A[j] и A[j+1]
```

единственное
отличие!

Метод пузырька

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



Как написать метод «камня»?



Как сделать рекурсивный вариант?

Задачи

- «А»: Напишите программу, в которой сортировка выполняется «методом камня» – самый «тяжёлый» элемент опускается в конец массива.
- «В»: Напишите вариант метода пузырька, который заканчивает работу, если на очередном шаге внешнего цикла не было перестановок.
- «С»: Напишите программу, которая сортирует массив по убыванию суммы цифр числа. Используйте функцию, которая определяет сумму цифр числа.

Метод выбора (минимального элемента)

Идея: найти минимальный элемент и поставить его на первое место.

```
сделать для i от 0 до N-2
    // найти номер nMin минимального
    // элемента из A[i]..A[N]
    если i != nMin то
        // поменять местами A[i] и A[nMin]
```

Метод выбора (минимального элемента)

```
for ( i = 0; i < N-1; i++ )  
{  
    nMin = i;  
    for ( j = i+1; j < N; j++ )  
        if ( A[j] < A[nMin] )  
            nMin = j;  
    if ( i != nMin )  
    {  
        // поменять местами A[i] и A[nMin]  
    }  
}
```



Как поменять местами два значения?

Задачи

«А»: Массив содержит четное количество элементов. Напишите программу, которая сортирует первую половину массива по возрастанию, а вторую – по убыванию. Каждый элемент должен остаться в «своей» половине.

Пример:

Массив :

5 3 4 2 1 6 3 2

После сортировки :

2 3 4 5 6 3 2 1

Задачи

«В»: Напишите программу, которая сортирует массив и находит количество различных чисел в нем.

Пример:

Массив :

5 3 4 2 1 6 3 2 4

После сортировки:

1 2 2 3 3 4 4 5 6

Различных чисел: 5

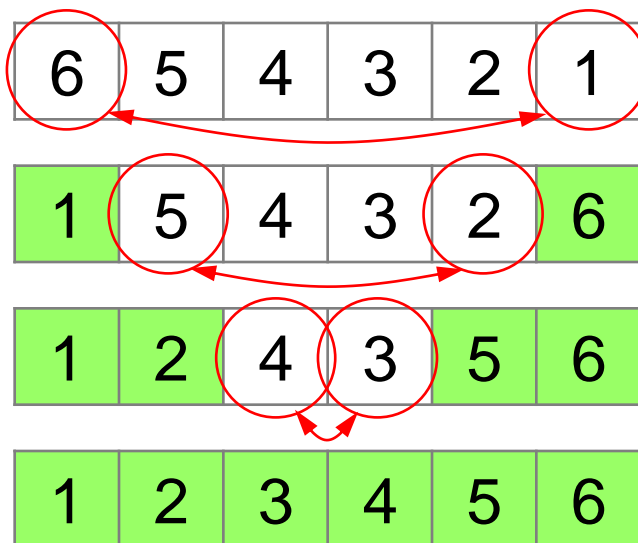
«С»: Напишите программу, которая сравнивает число перестановок элементов при использовании сортировки «пузырьком» и методом выбора. Проверьте ее на разных массивах, содержащих 1000 случайных элементов, вычислите среднее число перестановок для каждого метода.

Быстрая сортировка (*QuickSort*)



Ч.Э.Хоар

Идея: выгоднее переставлять элементы, который находятся дальше друг от друга.

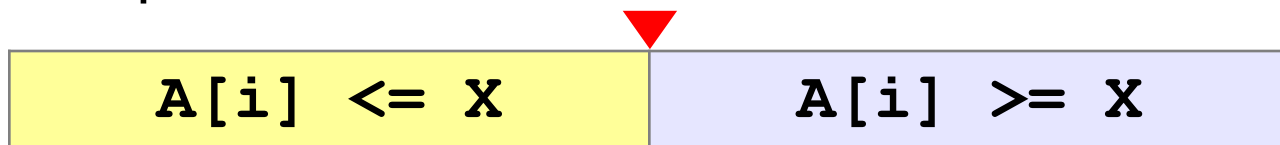


Для массива из N элементов нужно всего $N/2$ обменов!

Быстрая сортировка

Шаг 1: выбрать некоторый элемент массива X

Шаг 2: переставить элементы так:



при сортировке элементы не покидают « свою область »!

Шаг 3: так же отсортировать две получившиеся области

Разделяй и властвуй (англ. *divide and conquer*)

78	6	82	67	55	44	34
----	---	----	----	----	----	----

?

Как лучше выбрать X ?

Медиана – такое значение X , что слева и справа от него в отсортированном массиве стоит одинаковое число элементов (*для этого надо отсортировать массив...*).

Быстрая сортировка

Разделение:

1) выбрать средний элемент массива ($x=67$)

78	6	82	67	55	44	34
----	---	----	----	----	----	----

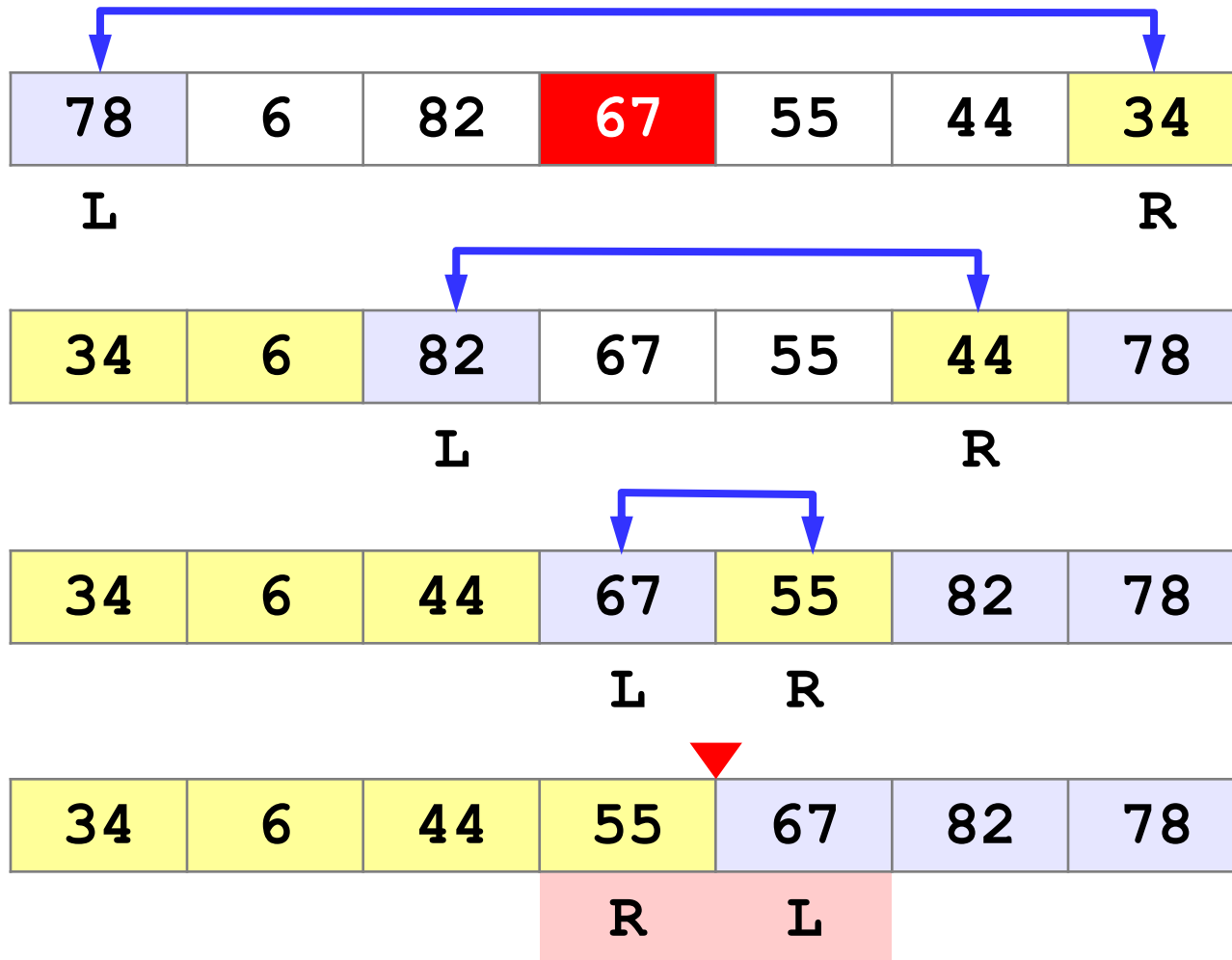
2) установить $L = 1$, $R = N$

3) увеличивая L , найти первый элемент $A[L]$,
который $\geq x$ (должен стоять справа)

4) уменьшая R , найти первый элемент $A[R]$,
который $\leq x$ (должен стоять слева)

5) если $L \leq R$ то поменять местами $A[L]$ и $A[R]$
и перейти к п. 3
иначе **СТОП**.

Быстрая сортировка



L > R : разделение закончено!

Быстрая сортировка

Основная программа:

```
const int N = 7; }  
int A[N];  
  
...  
main()  
{  
    // заполнить массив  
    qSort( 0, N-1 ); // сортировка  
    // вывести результат  
}
```

глобальные
данные

процедура
сортировки

Быстрая сортировка

```
void qSort( int nStart, int nEnd )
{
    int L, R, c, X;
    if ( nStart >= nEnd ) return; // готово
    L = nStart; R = nEnd;
    X = A[ (L+R)/2 ]; // или X = A[irand(L,R)];
    while ( L <= R ) { // разделение
        while ( A[L] < X ) L++;
        while ( A[R] > X ) R--;
        if ( L <= R ) {
            c = A[L]; A[L] = A[R]; A[R] = c;
            L++; R--;
        }
    }
    qSort ( nStart, R ); // рекурсивные вызовы
    qSort ( L, nEnd );
}
```



Что плохо?

Быстрая сортировка

Передача массива через параметр:

```
void qSort ( int A[], int nStart,
             int nEnd )
{
    ...
    qSort ( A, nStart, R );
    qSort ( A, L, nEnd );
}
```

```
main()
{ // заполнить массив
  qSort ( A, 0, N-1 ); // сортировка
  // вывести результат
}
```

Быстрая сортировка

Сортировка массива случайных значений:

N	метод пузырька	метод выбора	быстрая сортировка
1000	0,24 с	0,12 с	0,004 с
5000	5,3 с	2,9 с	0,024 с
15000	45 с	34 с	0,068 с

Задачи

«А»: Массив содержит четное количество элементов.

Напишите программу, которая сортирует по возрастанию отдельно элементы первой и второй половин массива.

Каждый элемент должен остаться в «своей» половине.

Используйте алгоритм быстрой сортировки.

Пример:

Массив :

5 3 4 2 1 6 3 2

После сортировки :

2 3 4 5 6 3 2 1

Задачи

«В»: Напишите программу, которая сортирует массив и находит количество различных чисел в нем. Используйте алгоритм быстрой сортировки.

Пример:

Массив :

5 3 4 2 1 6 3 2 4

После сортировки:

1 2 2 3 3 4 4 5 6

Различных чисел: 5

Задачи

- «C»: Напишите программу, которая сравнивает число перестановок элементов при использовании сортировки «пузырьком», методом выбора и алгоритма быстрой сортировки. Проверьте ее на разных массивах, содержащих 1000 случайных элементов, вычислите среднее число перестановок для каждого метода.
- «D»: Попробуйте построить массив из 10 элементов, на котором алгоритм быстрой сортировки показывает худшую эффективность (наибольшее число перестановок). Сравните это количество перестановок с эффективностью метода пузырька (для того же массива).

Программирование на языке C++

Матрицы

Что такое матрица?

	○	×
	○	×
○	×	

нет знака

НОЛИК

крестик

строка 1,
столбец 2



Как закодировать?

Матрица — это прямоугольная таблица, составленная из элементов одного типа (чисел, строк и т.д.). Каждый элемент матрицы имеет два индекса — номера строки и столбца.

Объявление матриц

```
const int N=3, M=4;  
int A[N][M];  
double X[10][12];
```

строки

столбцы

строки

столбцы



Нумерация строк и столбцов с нуля!



Если удобна нумерация с 1?

Заполнение случайными числами

Задача. Заполнить массив (псевдо)случайными целыми числами в диапазоне от 20 до 100.

```
int irand ( int a, int b )  
{  
    return a + rand() % (b - a + 1) ;  
}
```

```
for ( i = 0; i < N; i++ )  
{  
    A[i] = irand ( 20, 100 ) ;  
    cout << A[i] << " " ;  
}
```

Простые алгоритмы

Заполнение случайными числами:

```
for ( i = 0; i < N; i++ ) {  
    for ( j = 0; j < M; j++ ) {  
        A[i][j] = irand(20, 80);  
        cout << width(3);  
        cout << A[i][j];  
    }  
    cout << endl;  
}
```



Вложенный цикл!

Суммирование:

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

Простые алгоритмы

Заполнение случайными числами:

```
for ( i = 0; i < N; i++ ) {  
    for ( j = 0; j < M; j++ ) {  
        A[i][j] = irand(20, 80);  
        printf ( "%3d", A[i][j] );  
    }  
    printf ( "\\n" );  
}
```



Вложенный цикл!

Суммирование:

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

Задачи

«А»: Напишите программу, которая заполняет квадратную матрицу случайными числами в интервале $[10,99]$, и находит максимальный и минимальный элементы в матрице и их индексы.

Пример:

Матрица А:

12 14 67 45

32 87 45 63

69 45 14 11

40 12 35 15

Максимальный элемент $A[2,2]=87$

Минимальный элемент $A[3,4]=11$

Задачи

«В»: Яркости пикселей рисунка закодированы числами от 0 до 255 в виде матрицы. Преобразовать рисунок в черно-белый по следующему алгоритму:

- 1) *вычислить среднюю яркость пикселей по всему рисунку*
- 2) *все пиксели, яркость которых меньше средней, сделать черными (записать код 0), а остальные – белыми (код 255)*

Пример:

Матрица А:

12 14 67 45

32 87 45 63

69 45 14 11

40 12 35 15

Средняя яркость 37.88

Результат:

0 0 255 255

0 255 0 255

255 255 0 0

255 0 0 0

Задачи

«С»: Заполните матрицу, содержащую N строк и M столбцов, натуральными числами по спирали и змейкой, как на рисунках:

а)

1	2	3	4
10	11	12	5
9	8	7	6

б)

1	3	4	9
2	5	8	10
6	7	11	12

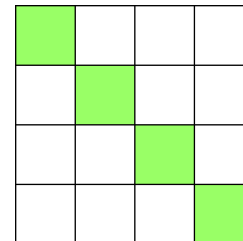
в)

1	6	7	12
2	5	8	11
3	4	9	10

Перебор элементов матрицы

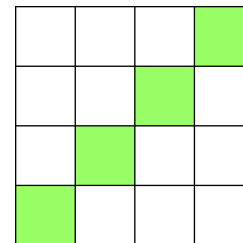
Главная диагональ:

```
for ( i = 0; i < N; i++ ) {  
    // работаем с A[i][i]  
}
```



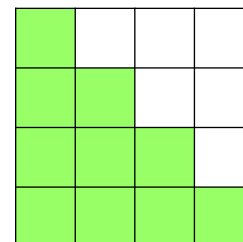
Побочная диагональ:

```
for ( i = 0; i < N; i++ ) {  
    // работаем с A[i][N-1-i]  
}
```



Главная диагональ и под ней:

```
for ( i = 0; i < N; i++ )  
    for ( j = 0; j <= i; j++ )  
    {  
        // работаем с A[i][j]  
    }
```



Перестановка строк

2-я и 4-я строки:

```
for ( j = 0; j < M; j++ )  
{  
    c = A[2][j];  
    A[2][j] = A[4][j];  
    A[4][j] = c;  
}
```

	0	1	2	3	4	5
0						
1						
2						
3						
4						
5						

Задачи

«А»: Напишите программу, которая заполняет квадратную матрицу случайными числами в интервале $[10,99]$, а затем записывает нули во все элементы выше главной диагонали. Алгоритм не должен изменяться при изменении размеров матрицы.

Пример:

Матрица А:

12 14 67 45

32 87 45 63

69 45 14 30

40 12 35 65

Результат:

12 0 0 0

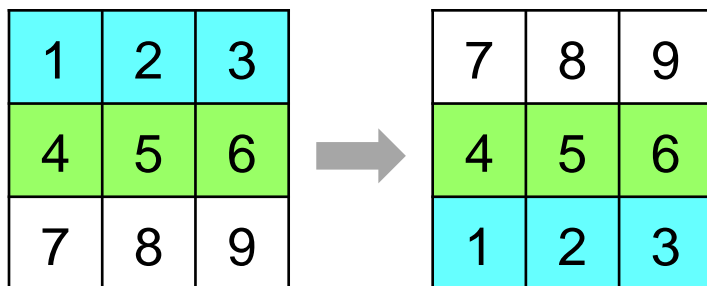
32 87 0 0

69 45 14 0

40 12 35 65

Задачи

«В»: Пиксели рисунка закодированы числами (обозначающими цвет) в виде матрицы, содержащей N строк и M столбцов. Выполните отражение рисунка сверху вниз:



«С»: Пиксели рисунка закодированы числами (обозначающими цвет) в виде матрицы, содержащей N строк и M столбцов. Выполните поворот рисунка вправо на 90 градусов:

