Программирование на языке C++

Программирование на языке C++

§ 62. 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!

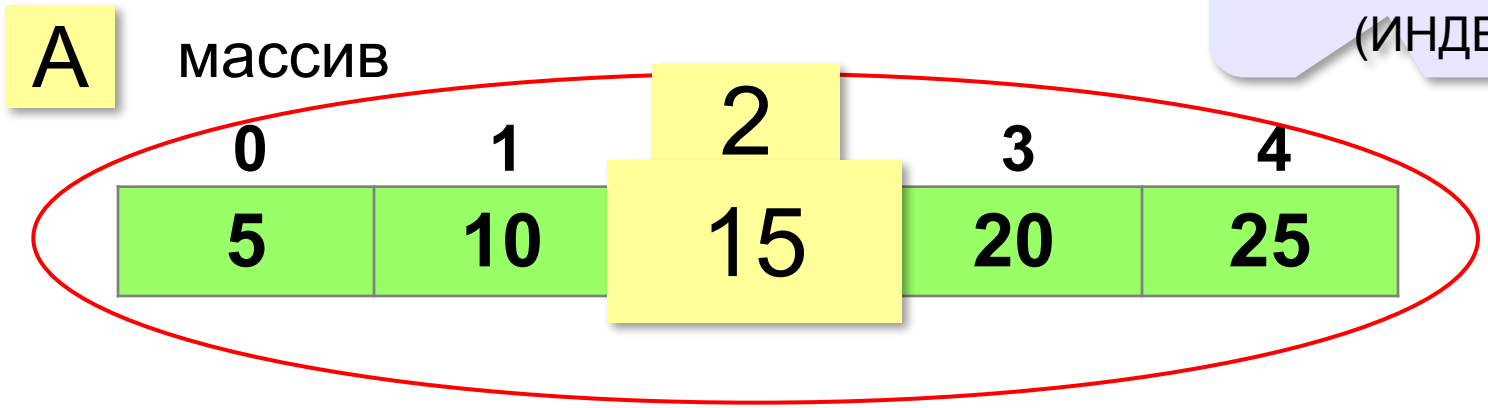
`A[0], A[1], A[2], A[3], A[4]`

размер через константу

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

Зачем?

Обращение к элементу массива



НОМЕР
элемента массива
(ИНДЕКС)

ЗНАЧЕНИЕ
элемента массива

НОМЕР (ИНДЕКС)
элемента массива: 2

A[2]

ЗНАЧЕНИЕ
элемента массива: 15

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

это внутренняя переменная цикла

Заполнение массива

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

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

Вывод на экран:

```
cout >> "Массив A:\n";
for ( int 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 randInt ( int a, int b )
{
  return a + rand() % (b - a + 1);
}
```

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

Перебор элементов

Общая схема:

```
for ( int 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?

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

```
int count = 0,
    sum = 0;
```

? Зачем `float`?

среднее арифметическое

Задачи

«A»: 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
```

«B»: 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++

§ 63. Алгоритмы обработки массивов

Поиск в массиве

Найти элемент, равный X:

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

? Что плохо?

```
int 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:

```
int nX = -1;
for ( int 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

«A»: 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

«B»: 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

«C»: 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

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

? Как найти его номер?

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

? Что можно улучшить?

Максимальный элемент и его номер

По номеру элемента можно найти значение!

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

Задачи

«A»: 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
```

«B»: 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
```

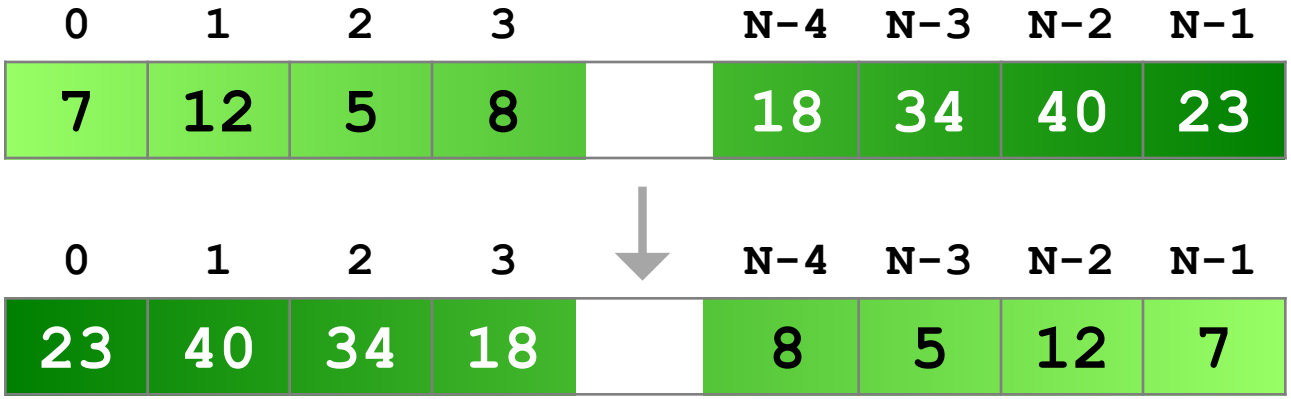
Задачи

«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]
   }
```

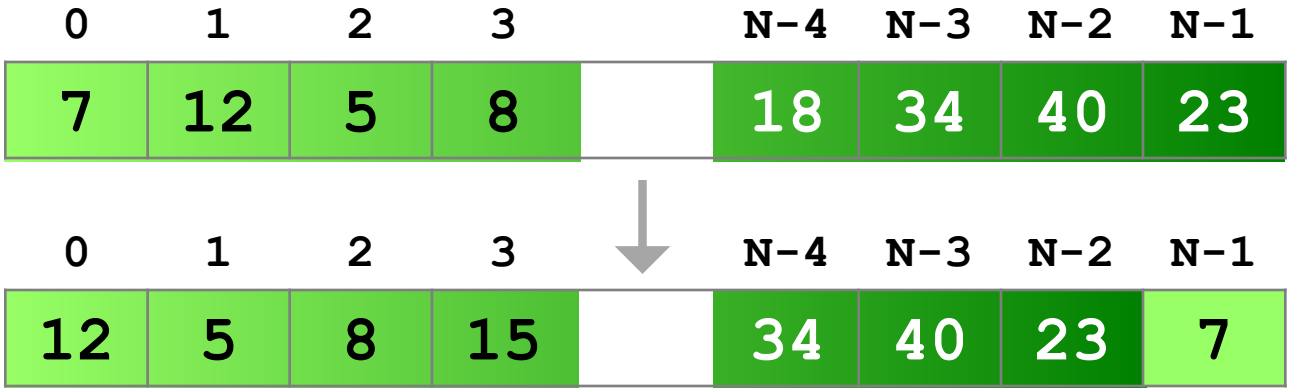
? Что плохо?

Реверс массива

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

? Почему не до N-1?

? Что плохо?

Задачи

«A»: 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
```

«B»: 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
```

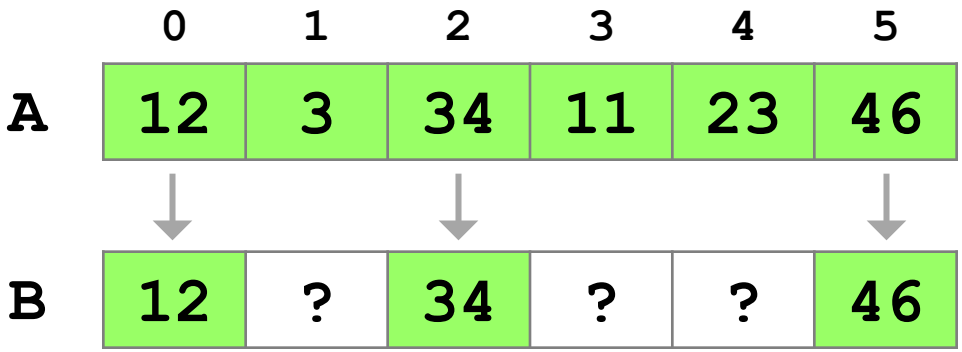
Отбор нужных элементов

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

«Простое» решение:

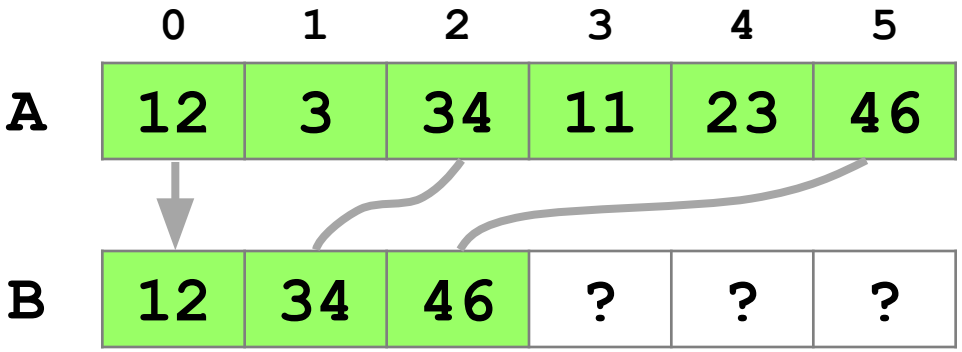
```
сделать для i от 0 до N-1
  если условие выполняется для A[i] то
    B[i]:=A[i]
```

? Что плохо?



выбрать чётные элементы

Отбор нужных элементов



выбрать чётные элементы

? Если A и B – один и тот же массив?

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

? Как вывести на экран?

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

Задачи

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

Пример:

```
Массив A:
-5 6 7 -4 -6 8 -8
Массив B:
-4 -6 -8
```

«B»: 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 A:
12 13 85 96 47
Массив B:
13 47
```

Задачи

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

Пример:

```
Массив A:
12 13 85 34 47
Массив B:
13 34
```

Программирование на языке C++

§ 64. Сортировка

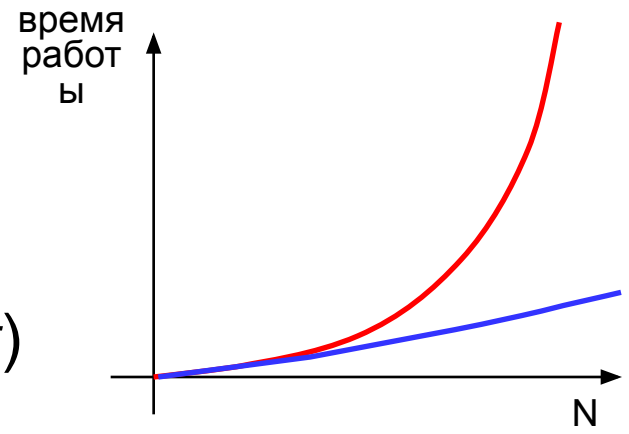
Что такое сортировка?

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

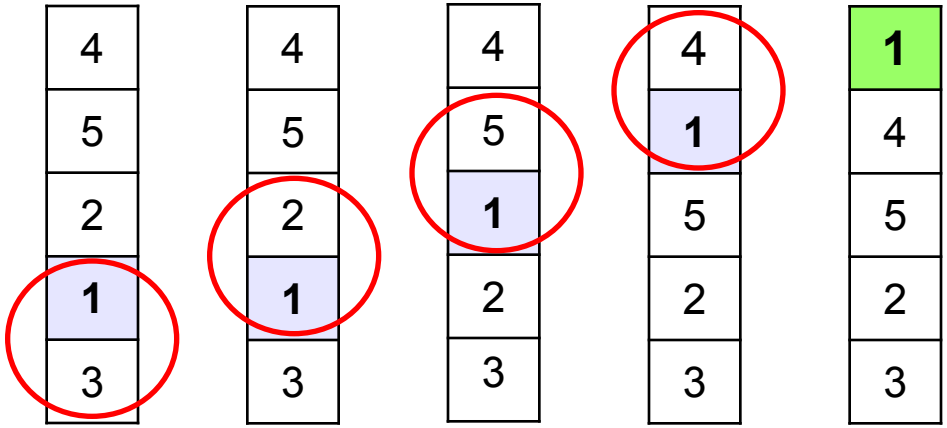


Метод пузырька (сортировка обменами)

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

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

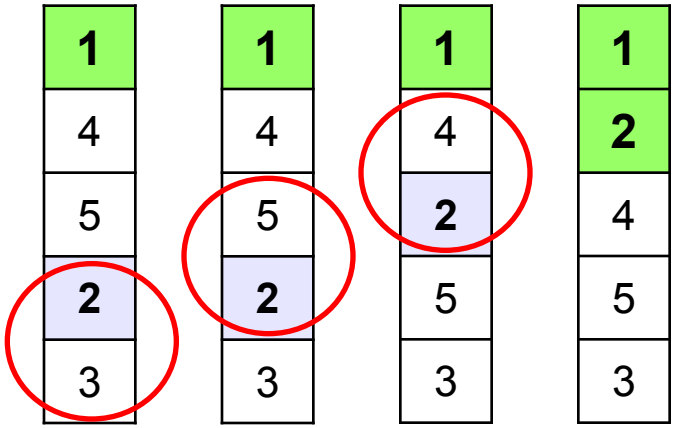
1-й проход:



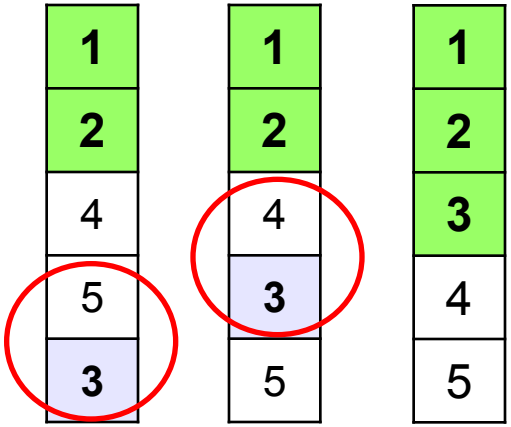
- сравниваем два соседних элемента; если они стоят «неправильно», меняем их местами
- за 1 проход по массиву один элемент (самый маленький) становится на свое место

Метод пузырька

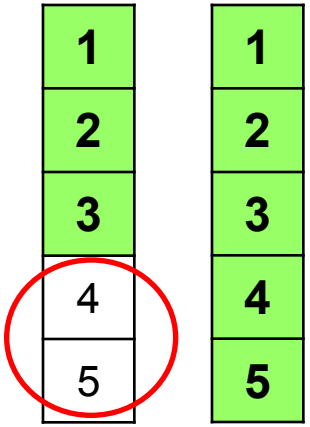
2-й проход:



3-й проход:



4-й проход:



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]
```

единственное отличие!

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]
```

Метод пузырька

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

«A»: Напишите программу, в которой сортировка выполняется «методом камня» – самый «тяжёлый» элемент опускается в конец массива.

«B»: Напишите вариант метода пузырька, который заканчивает работу, если на очередном шаге внешнего цикла не было перестановок.

«C»: 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-1]
  если i != nMin то
    // поменять местами A[i] и A[nMin]
```

Метод выбора (минимального элемента)

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

«A»: 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

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

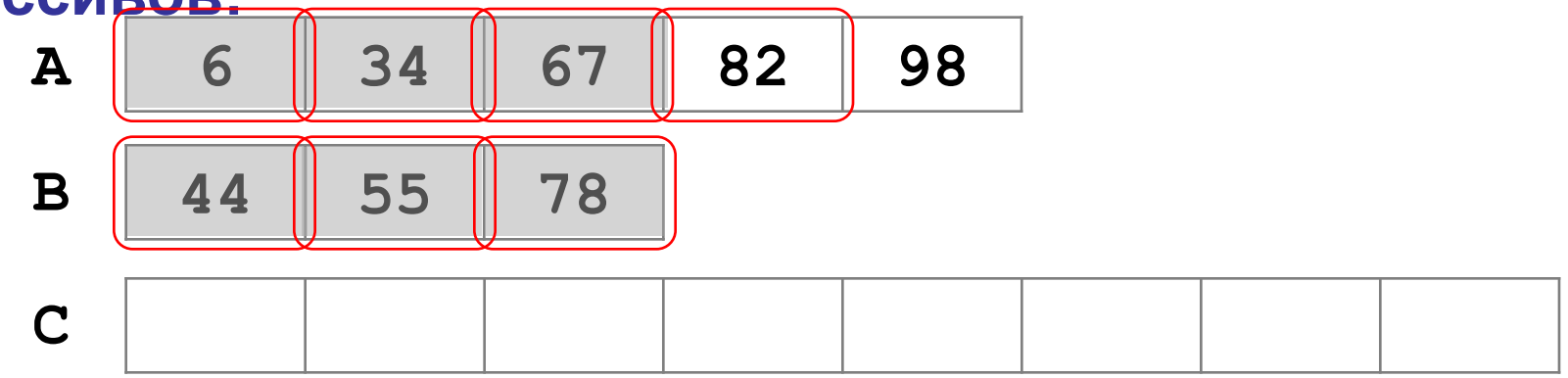
Пример:

```
Массив:
5 3 4 2 1 6 3 2 4
После сортировки:
1 2 2 3 3 4 4 5 6
Различных чисел: 5
```

«C»: 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.

Сортировка слиянием

Слияние отсортированных массивов:



новый тип

динамический массив

```
#include <vector>
using vecInt = vector<int>;
int iA = 0, iB = 0;   // позиции в A и B
int j = 0;            // позиция в C
int nA = A.size();
int nB = B.size();
```

Процедура слияния (I)

```
vecInt C( nA + nB ); // это результат

  // пока в обоих массивах остались данные
while ( iA < nA  &&  iB < nB ) {

  if ( A[iA] <= B[iB] ) { // взять из A
    C[j] = A[iA]; iA++;   // C[j] = A[iA++];
    }
  else { // взять из B
    C[j] = B[iB++]; // C[j] = A[iB]; iB++;
    }

  j++;  // к следующему элементу C
}
```

Процедура слияния (II)

Дописываем «хвост» одного массива:

```
while ( iA < nA ) {
   C[j] = A[iA];
   iA++;
   j++;
   }
while ( iB < nB )
   C[j++] = B[iB++];
```

Процедура слияния (III)

Оформляем функцию:

```
vecInt merge( vecInt A, vecInt B )
{
      // всё, что написано раньше
  return C;
}
```

Сортировка слиянием

```
vecInt mergeSort( vecInt A ) {
  vecInt L, R;
  if ( A.size() == 1 ) return A;

  int mid = A.size() / 2;
  L.resize( mid );  // левая
  R.resize( A.size()- mid ); // правая

  for ( int i = 0; i < L.size(); i++ )
    L[i] = A[i];
  for ( int i = 0; i < R.size(); i++ )
    R[i] = A[mid+i];

  L = mergeSort(L);
  R = mergeSort(R);
  return merge(L, R);
}
```

выход из рекурсии

номер среднего элемента

сортировка половин

Сортировка слиянием

- работает быстро

- нужен дополнительный массив

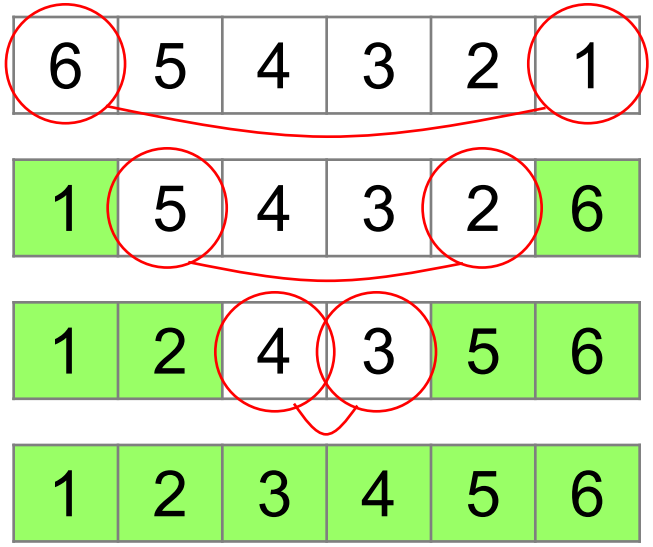
«Разделяй и властвуй» (*divide and conquer*):

1) задача разбивается на несколько подзадач меньшего размера;
2) эти подзадачи решаются с помощью рекурсивных вызовов того же (или другого) алгоритма;
3) решения подзадач объединяются, и получается решение исходной задачи.

Быстрая сортировка (*QuickSort*)

Ч.Э.Хоар

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

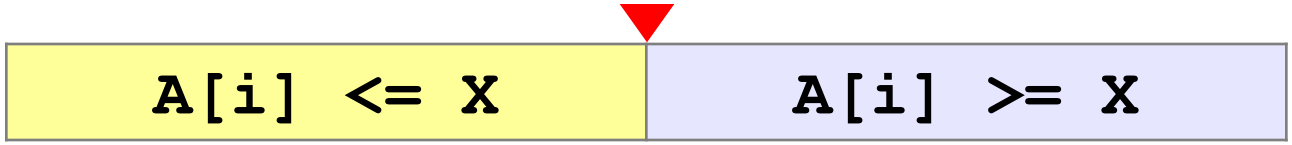


! Для массива из **N** элементов нужно всего **N/2** обменов!

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*)

? Как лучше выбрать X?

Медиана – такое значение **X**, что слева и справа от него в отсортированном массиве стоит одинаковое число элементов (*это довольно сложно…*).

Быстрая сортировка

Разделение:

1)выбрать любой элемент массива (**X=67**)

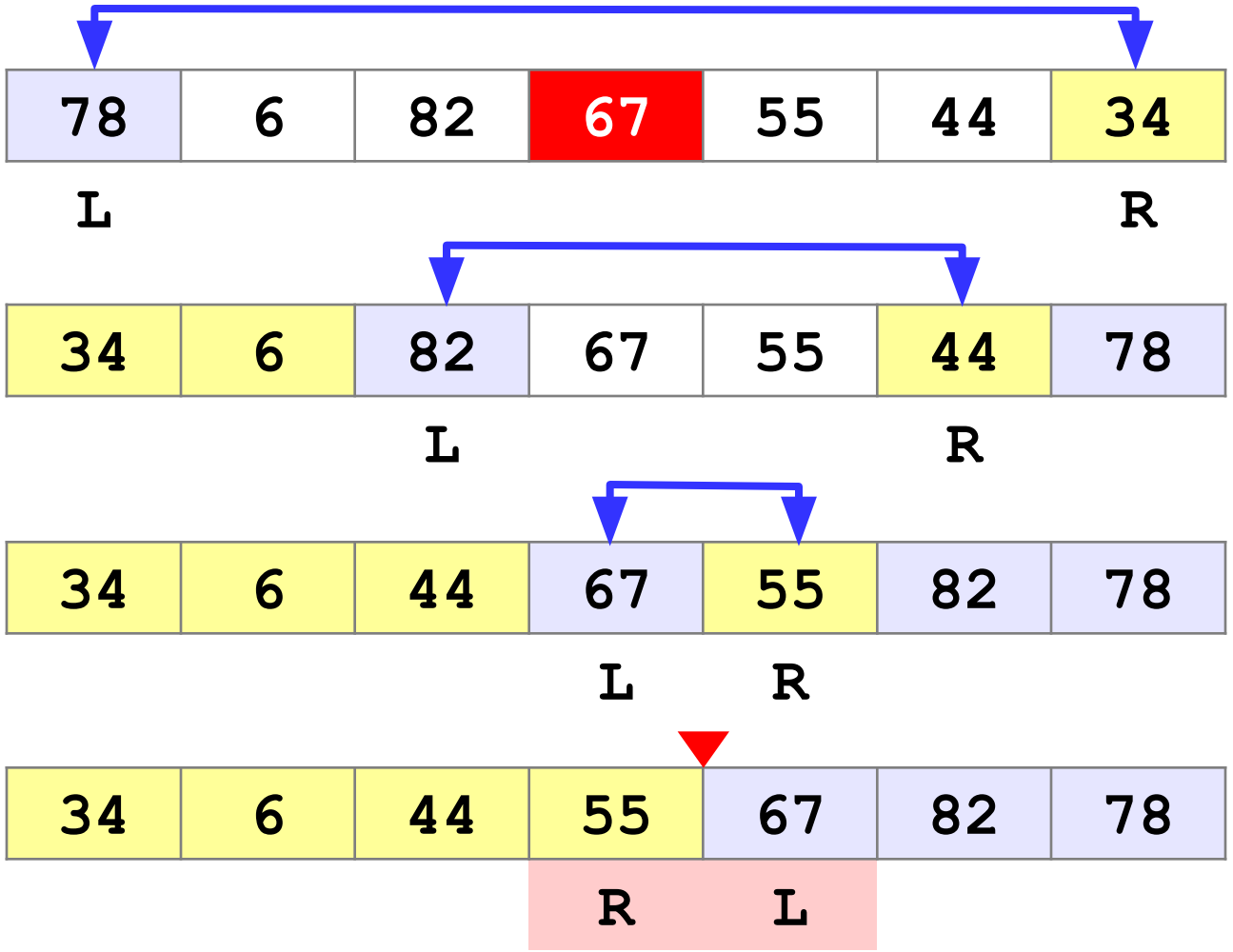
2)установить **L = 0, R = N-1**

3)увеличивая **L**, найти первый элемент **A[L]**, который **>= X** (должен стоять справа)

4)уменьшая **R**, найти первый элемент **A[R]**, который **<= X** (должен стоять слева)

5)если **L<=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:

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

процедура сортировки

массив-параметр

Быстрая сортировка

```
void qSort( int A[] , 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 ( A, nStart, R );    // рекурсивные вызовы
  qSort ( A, L, nEnd );
}
```

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

«A»: 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

«B»: 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

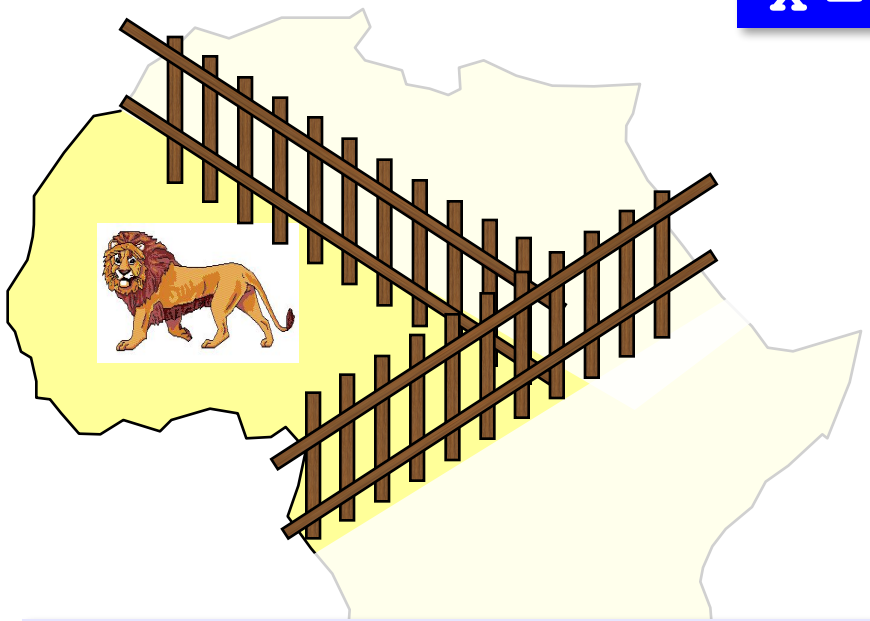
«C»: Напишите программу, которая сравнивает число перестановок элементов при использовании сортировки «пузырьком», методом выбора и алгоритма быстрой сортировки. Проверьте ее на разных массивах, содержащих 1000 случайных элементов, вычислите среднее число перестановок для каждого метода.

«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).

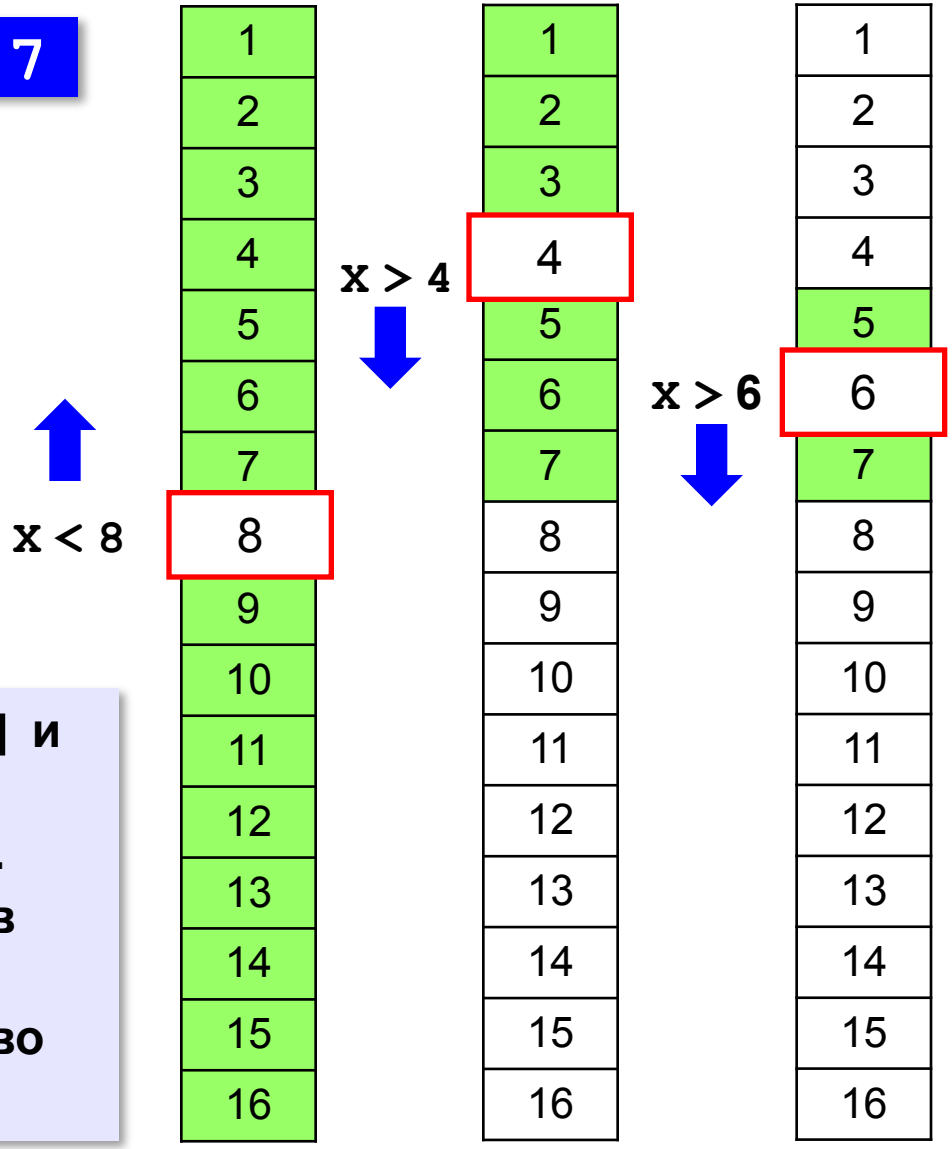
Программирование на языке C++

§ 65. Двоичный поиск

Двоичный поиск

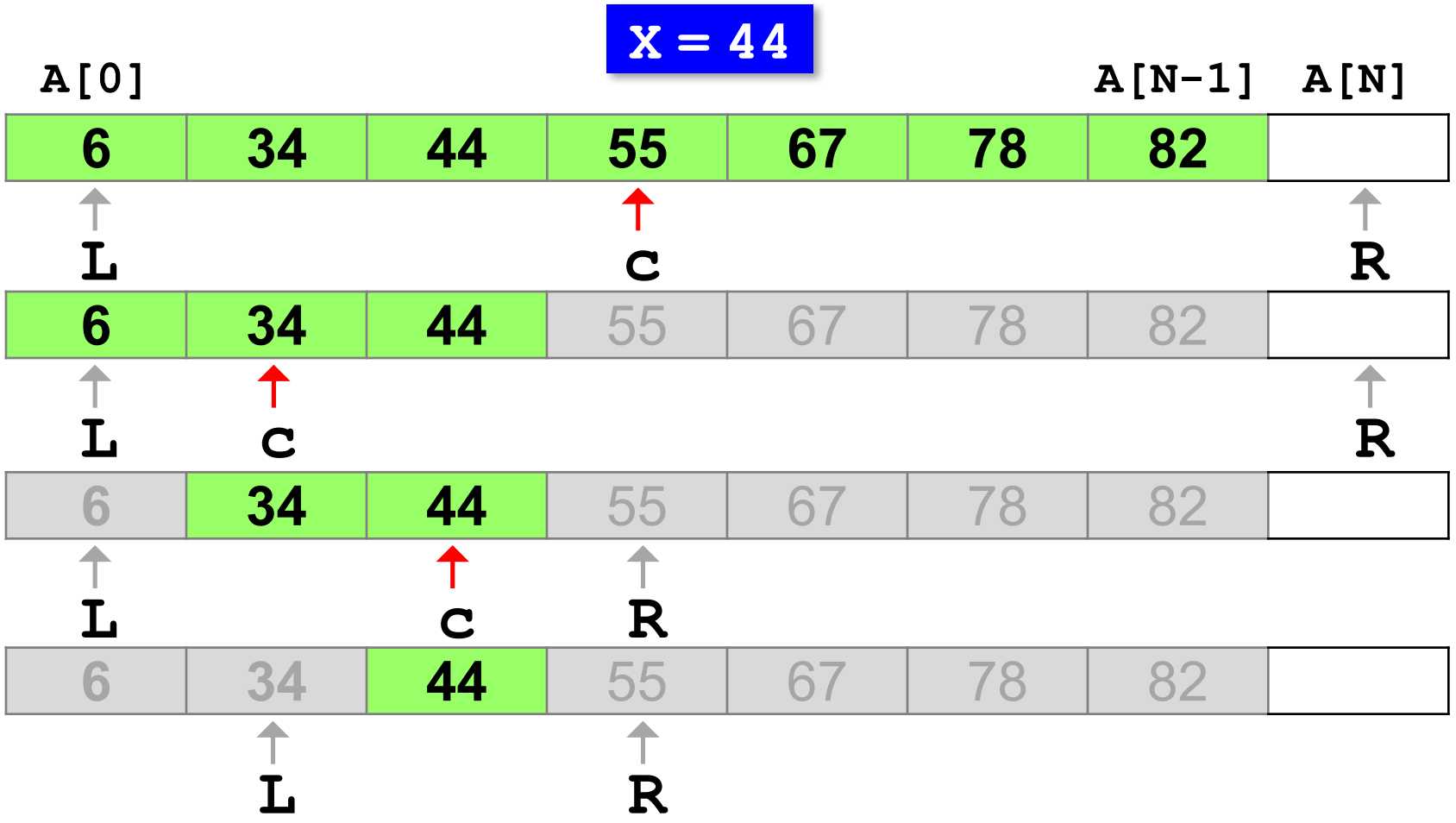


X = 7



1. **Выбрать средний элемент `A[c]` и сравнить с X.**
2. **Если `X = A[c]`, то нашли (стоп).**
3. **Если `X < A[c]`, искать дальше в первой половине.**
4. **Если `X > A[c]`, искать дальше во второй половине.**

Двоичный поиск



! L = R−1 : поиск завершен!

Двоичный поиск

```
int X;
// ввести или получить X
int L=0, R=N;     // начальный отрезок
while ( L<R-1 )
   {
   int c=(L+R)/2; // середина
   if ( X<A[c] )   // сжатие отрезка
         R=c;
   else L=c;
   }
if ( A[L]==X )
      printf ( "A[%d]=%d", L, X );
else printf ( "Не нашли!" );
```

Двоичный поиск

Число сравнений:

N	линейный поиск	двоичный поиск
2	2	2
16	16	5
1024	1024	11
1048576	1048576	21

- скорость выше, чем при линейном поиске

- нужна предварительная сортировка

Когда нужно применять?

Задачи

«A»: Заполнить массив случайными числами и отсортировать его. Ввести число X. Используя двоичный поиск, определить, есть ли в массиве число, равное X. Подсчитать количество сравнений.

Пример:

```
Массив:
1 4 7 3 9 2 4 5 2
После сортировки:
1 2 2 3 4 4 5 7 9
Введите число X:
2
Число 2 найдено.
Количество сравнений: 2
```

Задачи

«B»: Заполнить массив случайными числами и отсортировать его. Ввести число X. Используя двоичный поиск, определить, сколько чисел, равных X, находится в массиве.

Пример:

```
Массив:
1 4 7 3 9 2 4 5 2
После сортировки:
1 2 2 3 4 4 5 7 9
Введите число X:
4
Число 4 встречается 2 раз(а).
```

Пример:

```
Массив:
1 4 7 3 9 2 4 5 2
После сортировки:
1 2 2 3 4 4 5 7 9
Введите число X:
14
Число 14 не встречается.
```

Задачи

«С»: Заполнить массив случайными числами и ввести число и отсортировать его. Ввести число X. Используя двоичный поиск, определить, есть ли в массиве число, равное X. Если такого числа нет, вывести число, ближайшее к X.

Пример:

```
Массив:
1 4 7 3 9 2 4 5 2
После сортировки:
1 2 2 3 4 4 5 12 19
Введите число X:
12
Число 12 найдено.
```

Пример:

```
Массив:
1 4 7 3 9 2 4 5 2
После сортировки:
1 2 2 3 4 4 5 12 19
Введите число X:
11
Число 11 не найдено. Ближайшее число 12.
```

Программирование на языке C++

§ 66. Символьные строки

Зачем нужны символьные строки?

```
char s[10];  // массив символов
```

- элементы массива – отдельные объекты
- сложно работать со строками переменной длины

Хочется:

- строка – единый объект
- длина строки может меняться во время работы программы

```
string s;  // символьная строка
```

строка

Символьные строки

Начальное значение:

```
string s = "Привет!";
```

Присваивание:

```
s = "Привет!";
```

Вывод на экран:

```
cout << s;
```

А если массив?

Символьные строки

Ввод с клавиатуры:

```
cin >> s;
```

только до пробела!

```
getline( cin, s );
```

до перевода строки (Enter)

Отдельный символ:

```
s[4] = 'a';
```

Символы в строке нумеруются с нуля!

Длина строки:

```
int n = s.size();
```

метод для объектов типа `string`

Символьные строки

Задача: заменить в строке все буквы **'а'** на буквы **'б'**.

```
#include <iostream>
using namespace std;
int main()
{
  string s;
  cout << "Введите строку: ";
  getline ( cin, s );
  for ( int i=0; i<s.size(); i++ )
    if ( s[i]=='а' )
      s[i]='б';
  cout << s;
}
```

цикл по всем символам строки

Задачи

«А»: Ввести с клавиатуры символьную строку и заменить в ней все буквы «а» на «б» и все буквы «б» на «а» (заглавные на заглавные, строчные на строчные).

Пример:

```
Введите строку:
ааббААББссСС
Результат:
ббааББААссСС
```

Задачи

«B»: Ввести с клавиатуры символьную строку и определить, сколько в ней слов. Словом считается последовательности непробельных символов, отделенная с двух сторон пробелами (или стоящая с краю строки). Слова могут быть разделены несколькими пробелами, в начале и в конце строки тоже могут быть пробелы.

Пример:

```
Введите строку:
  Вася  пошел      гулять
Найдено слов: 3
```

Задачи

«С»: Ввести с клавиатуры символьную строку и найдите самое длинное слово и его длину. Словом считается последовательности непробельных символов, отделенная с двух сторон пробелами (или стоящая с краю строки). Слова могут быть разделены несколькими пробелами, в начале и в конце строки тоже могут быть пробелы.

Пример:

```
Введите строку:
  Вася   пошел  гулять
Самое длинное слово: гулять, длина 6
```

Операции со строками

Объединение (*конкатенация*):

```
string s, s1, s2;
s1 = "Привет";
s2 = "Вася";
s = s1 + ", " + s2 + "!";
```

"Привет, Вася!"

Срез (подстрока):

```
s = "0123456789";
s1 = s.substr( 3, 5 );    // "34567"
```

откуда

с какого символа

сколько символов

5

```
s = "0123456789";
s1 = s.substr( 3 );    // "3456789"
```

Операции со строками

Удаление:

```
s = "0123456789";
s.erase ( 3, 6 ); // "0129"
```

с какого символа

сколько символов

Вставка:

```
s = "0123456789";
s.insert( 3,"ABC" ); // "012ABC3456789"
```

куда

с какого символа

что

Поиск символа в строке

```
string s = "Здесь был Вася.";
int n = s.find( 'c' );  // 3
```

find – искать

Вернёт **string::npos**, если не нашли!

no position

```
if ( n != string::npos )
     cout << "Номер символа 'c': "
          << n << endl;
else cout << "Символ не найден.\n";
```

Поиск подстроки

```
string s = "Здесь был Вася.";
int n=s.find( "Вася" );  // 10
```

```
if ( n!= string::npos )
   cout << "Слово начинается с s["
        << n << "]\n";
else
   cout << "Слово не найдено.\n";
```

`s.rfind()` – поиск с конца строки!

Пример обработки строк

Задача: Ввести имя, отчество и фамилию. Преобразовать их к формату «фамилия-инициалы».

Пример:

```
Введите имя, отчество и фамилию:
Василий Алибабаевич Хрюндиков
Результат:
Хрюндиков В.А.
```

Алгоритм:

- найти первый пробел и выделить имя
- удалить имя с пробелом из основной строки
- найти первый пробел и выделить отчество
- удалить отчество с пробелом из основной строки
- «сцепить» фамилию, первые буквы имени и фамилии, точки, пробелы…

`Алибабаевич Хрюндиков`

`Хрюндиков`

`Хрюндиков В.А.`

Пример обработки строк

```
int main()
{
  string s, name, name2;
  int n;
  cout << "Введите имя, отчество и фамилию: ";
  getline ( cin, s );
  name = s.substr(0,1) + '.';// начало имени
  n = s.find(' ');           // найти пробел
  s = s.substr ( n+1 );      // удалить имя
  n = s.find(' ');           // найти пробел
  name2 = s.substr(0,1) + '.';// начало отчества
  s = s.substr( n+1 );       // осталась фамилия
  s = s + ' ' + name + name2; // результат
  cout << s;
}
```

Задачи

«А»: Ввести с клавиатуры в одну строку фамилию, имя и отчество, разделив их пробелом. Вывести фамилию и инициалы.

Пример:

```
Введите фамилию, имя и отчество:
Иванов Петр Семёнович
П.С. Иванов
```

Задачи

«B»: Ввести адрес файла и «разобрать» его на части, разделенные знаком `'/'`. Каждую часть вывести в отдельной строке.

Пример:

```
Введите адрес файла:
C:/Фото/2013/Поход/vasya.jpg
C:
Фото
2013
Поход
vasya.jpg
```

Задачи

«С»: Напишите программу, которая заменяет во всей строке одну последовательность символов на другую.

Пример:

```
Введите строку:
(X > 0) and (Y < X) and (Z > Y) and (Z <> 5)
Что меняем: and
Чем заменить: &
Результат
(X > 0) & (Y < X) & (Z > Y) & (Z <> 5)
```

Преобразования «строка» – «число»

Из строки в число:

```
#include <cstdlib>
...
string s="123";
int N=atoi( s.c_str() );     // N = 123
```

«12x3» → 12

в строку языка Си

```
string s="123.456";
float X;
X=atof( s.c_str() );  // X = 123.456
```

Преобразования «строка» – «число»

Из числа в строку:

Идея: направить выходной поток в строку!

```
#include <sstream>
```

строковые потоки

```
ostringstream ss;
string s;
int N=123;
ss << N;
s=ss.str();    // s = "123"
```

строковый поток вывода

из потока в строку

Преобразования «строка» – «число»

Вещественное число в строку:

```
ostringstream ss;
string s;
double X=123.456;
ss.width(10);     // ширина поля
ss.precision(3); // знаков в дробной части
ss << X;
s=ss.str();       // s ="   123.456"
```

Научный формат:

```
ss.str("");       // очистка потока
ss.width(10);     // ширина поля
ss.precision(6); // знаков в дробной части
ss << scientific << X; // научный формат
s=ss.str();       // s = "1.234560E+002"
```

Задачи

«A»: Напишите программу, которая вычисляет сумму трех чисел, введенную в форме символьной строки. Все числа целые.

Пример:

```
Введите выражение:
12+3+45
Ответ: 60
```

«B»: Напишите программу, которая вычисляет выражение, состоящее из трех чисел и двух знаков (допускаются только знаки «+» или «–»). Выражение вводится как символьная строка, все числа целые.

Пример:

```
Введите выражение:
12-3+45
Ответ: 54
```

Задачи

«C»: Напишите программу, которая вычисляет выражение, состоящее из трех чисел и двух знаков (допускаются знаки «+», «–», «*» и «/»). Выражение вводится как символьная строка, все числа целые. Операция «/» выполняется как целочисленное деление (**div**).

Пример:

```
Введите выражение:
12*3+45
Ответ: 81
```

Задачи

«D»: Напишите программу, которая вычисляет выражение, состоящее из трех чисел и двух знаков (допускаются знаки «+», «–», «*» и «/») **и круглых скобок**. Выражение вводится как символьная строка, все числа целые. Операция «/» выполняется как целочисленное деление.

Пример:

```
Введите выражение:
2*(3+45)+4
Ответ: 100
```

Строки в процедурах и функциях

Задача: построить процедуру, которая заменяет в строке **s** все вхождения слова-образца **wOld** на слово-замену **wNew**.

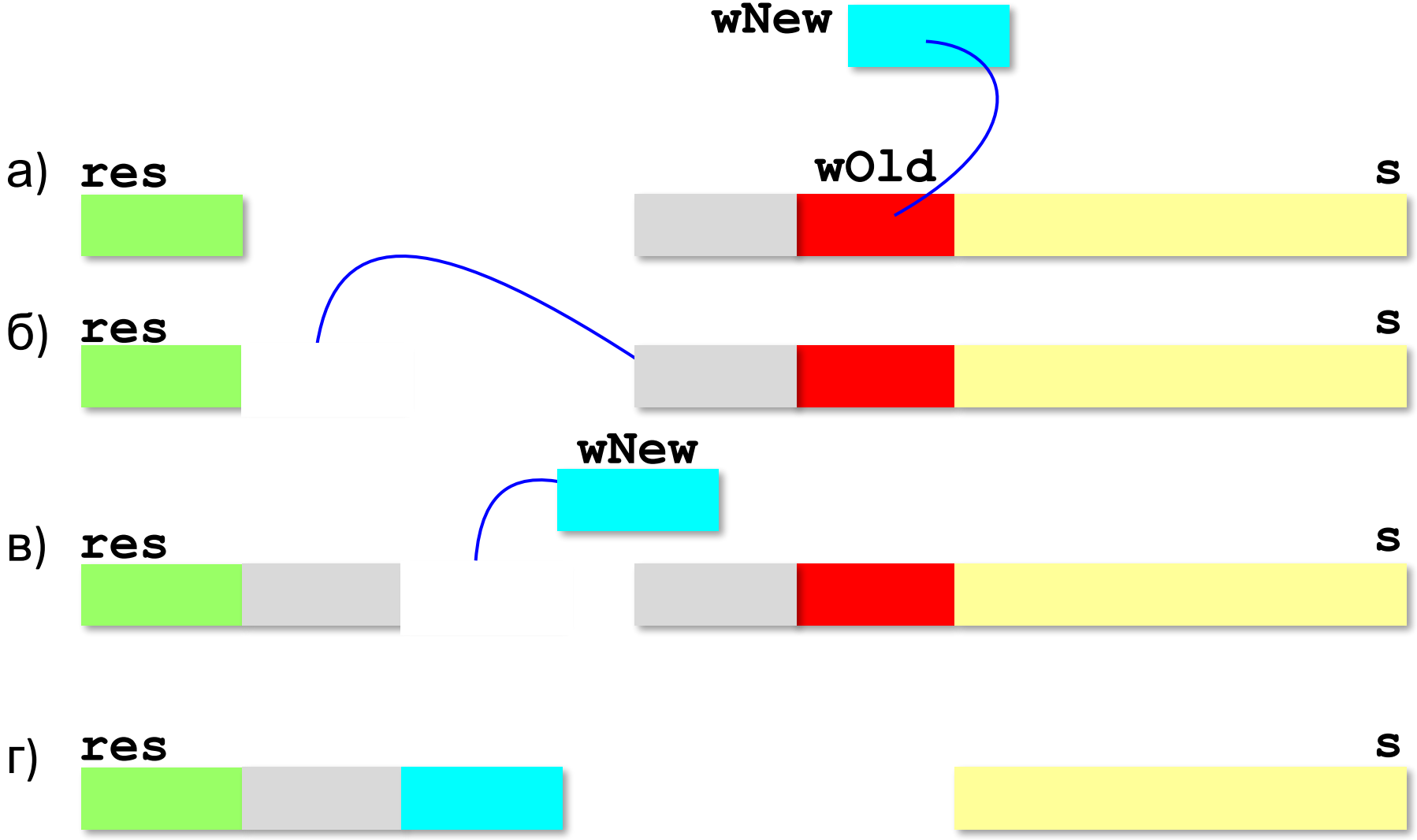
```
пока // слово wOld есть в строке s
  // удалить слово wOld из строки
  // вставить на это место слово wNew
```

Что плохо?

```
wOld:  '12'
wNew:  'A12B'
```

зацикливание

Замена всех экземпляров подстроки



Замена всех экземпляров подстроки

```
int main()
{
   string s="12.12.12";
   replaceAll( s, "12", "A12B" );
   cout << s;
}
```

рабочая строка s	результат res
"12.12.12"	""

Замена всех экземпляров подстроки

```
void replaceAll ( string &s, string wOld,
                  string wNew )
{
 string res = "";
 int p, len = wOld.size();
 while ( s.size() > 0 )
   {
   p = s.find ( wOld ); // искать образец
   if ( не нашли ) { // прицепить хвост и выйти }
   if ( p > 0 ) { // скопировать часть до образца }
   res = res + wNew; // добавить слово-замену
   if ( p + len > s.size() ) s = "";
   else s.erase ( 0, p + len );
   }
 s = res;
}
```

длина строки-образца

удалить начало

Замена всех экземпляров подстроки

Если образец не найден:

```
p = s.find ( wOld );
if ( p == string::npos )
   {
   res = res + s;
   break;
   }
```

если не нашли

прицепить «хвост»

выйти из цикла

Если перед образцом что-то есть:

```
if ( p > 0 )
   res = res + s.substr ( 0, p );
```

то, что перед образцом

Замена: из процедуры в функцию

```
string replaceAll( string s, string wOld,
                   string wNew )
{
   ...
   return res;
}
```

```
int main()
{
   string s="12.12.12";
   string sNew=replaceAll( s, "12", "A12B" );
   cout << sNew;
}
```

Задачи

«A»: Напишите функцию, которая возвращает первое слово переданной ей строки.

Пример:

```
Введите строку: Однажды в студёную зимнюю пору...
Первое слово: Однажды
```

Задачи

«B»: Напишите функцию, которая заменяет расширение файла на заданное новое расширение.

Пример:

```
Введите имя файла: qq
Введите новое расширение: tmp
Результат: qq.tmp
```

Пример:

```
Введите имя файла: qq.exe
Введите новое расширение: tmp
Результат: qq.tmp
```

Пример:

```
Введите имя файла: qq.work.xml
Введите новое расширение: tmp
Результат: qq.work.tmp
```

Задачи

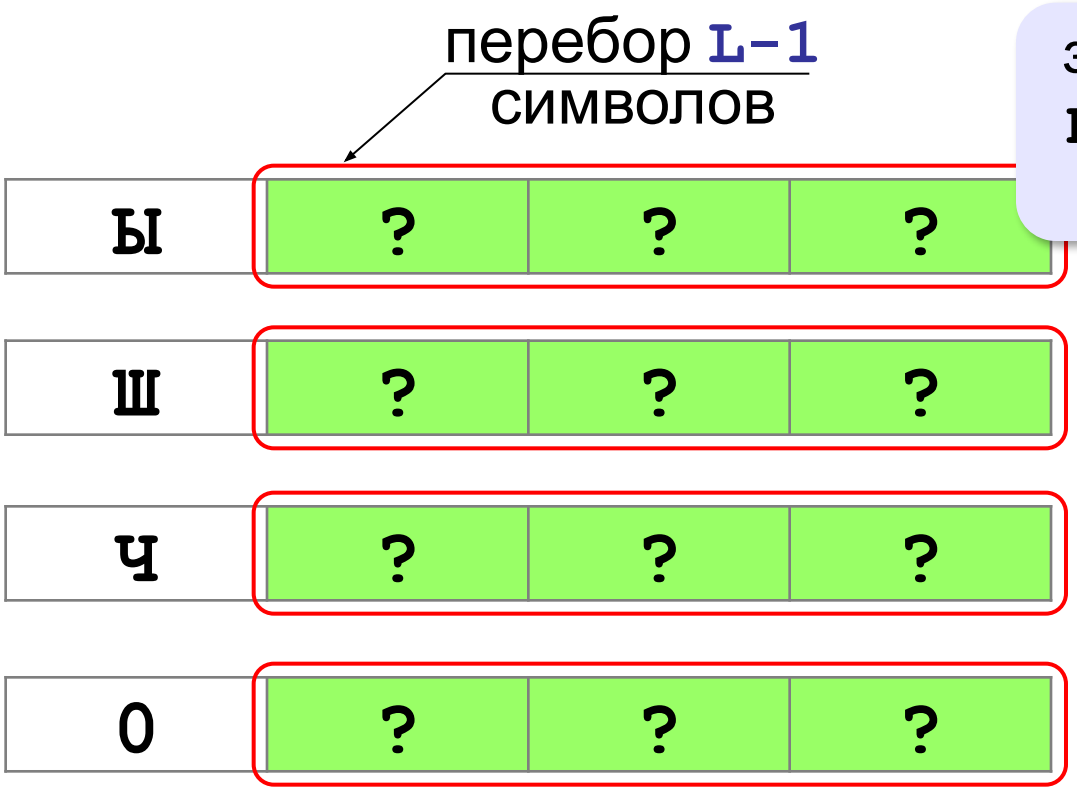
«С»: Напишите функцию, которая заменяет во всей строке все римские числа на соответствующие десятичные числа.

Пример:

```
Введите строку:
В MMXIII году в школе CXXIII состоялся очередной
    выпуск XI классов.
Результат:
В 2013 году в школе 123 состоялся очередной выпуск
    11 классов.
```

Рекурсивный перебор

Задача. В алфавите языка племени «тумба-юмба» четыре буквы: «**Ы**», «**Ш**», «**Ч**» и «**О**». Нужно вывести на экран все слова, состоящие из **L** букв, которые можно построить из букв этого алфавита.



задача для слов длины **K** сведена к задаче для слов длины **L-1**!

Рекурсивный перебор

```
перебор L символов
   w[0]='Ы';
    // перебор последних L-1 символов
   w[0]='Ш';
    // перебор последних L-1 символов
   w[0]='Ч';
    // перебор последних L-1 символов
   w[0]='О';
    // перебор последних L-1 символов
```

Рекурсивный перебор

```
void TumbaWords( string A, string &w, int N )
{
  int i;
  if ( N==w.size() ) {
    cout << w << endl;
    return;
    }
  for ( i=0; i<A.size(); i ++ ) {
    w[N]=A[i];
    TumbaWords ( A, w, N+1 );
    }
}
```

алфавит

слово

уже установлено

когда все символы уже установлены

по всем символам алфавита

```
int main()
{
  string word = "...";
  TumbaWords ( "ЫШЧО", word, 0 );
}
```

любая строка длины L

Задачи

«A»: В алфавите языке племени «тумба-юмба» четыре буквы: «Ы», «Ш», «Ч» и «О». Нужно вывести на экран все возможные слова, состоящие из K букв, в которых вторая буква «Ы». Подсчитайте количество таких слов.

«B»: В алфавите языке племени «тумба-юмба» четыре буквы: «Ы», «Ш», «Ч» и «О». Нужно вывести на экран все возможные слова, состоящие из K букв, в которых есть по крайней мере две одинаковые буквы, стоящие рядом. Подсчитайте количество таких слов.
Программа не должна строить другие слова, не соответствующие условию.

Задачи

«С»: В алфавите языке племени «тумба-юмба» четыре буквы: «Ы», «Ш», «Ч» и «О». Нужно вывести на экран все возможные слова, состоящие из K букв, в которых есть по крайней мере две одинаковые буквы, не обязательно стоящие рядом.

Программа не должна строить другие слова, не соответствующие условию.

Сравнение строк

Пар ? пар ? парк

Сравнение по кодам символов:

	0	1	...	8	9
CP-1251	48	49	...	56	57
UNCODE	48	49	...	56	57
	A	**B**	...	**Y**	**Z**
CP-1251	65	66	...	89	90
UNCODE	65	66	...	89	90
	a	**b**	...	**y**	**z**
CP-1251	97	98	...	121	122
UNCODE	97	98	...	121	122

Сравнение строк

	А	Б	...	Ё	...	Ю	Я
CP-1251	192	193	...	168	...	222	223
UNCODE	1040	1041	...	1025	...	1070	1071

	а	б	...	ё	...	ю	я
CP-1251	224	225	...	184	...	254	255
UNCODE	1072	1073	...	1105	...	1102	1103

5STEAM < STEAM < Steam < steam

steam < ПАР < Пар < пАр < пар < парк

Сортировка строк

```
int main()
{
  const int N=10;
  string s1, S[N];
  cout << "Введите строки: \n";
  for ( int i=0;
    getline ( cin,
    ...
  cout << "После
  for ( int i=0;
    cout << S[i]
}
```

массив строк

```
for ( int i=0; i<N-1; i++ )
  for ( int j=N-2; j>=i; j-- )
    if ( S[j] > S[j+1] )
      {
      s1 = S[j];
      S[j] = S[j+1];
      S[j+1] = s1;
      }
```

? Какой метод?

Задачи

«А»: Вводится 5 строк, в которых сначала записан порядковый номер строки с точкой, а затем – слово. Вывести слова в алфавитном порядке.

Пример:

```
Введите 5 строк:
1. тепловоз
2. арбуз
3. бурундук
4. кефир
5. урядник
Список слов в алфавитном порядке:
арбуз, бурундук, кефир, тепловоз, урядник
```

Задачи

«B»: Вводится несколько строк (не более 20), в которых сначала записан порядковый номер строки с точкой, а затем – слово. Ввод заканчивается пустой строкой. Вывести введённые слова в алфавитном порядке.

Пример:

```
Введите слова:
1. тепловоз
2. арбуз

Список слов в алфавитном порядке:
арбуз, тепловоз
```

Задачи

«С»: Вводится несколько строк (не более 20), в которых сначала записаны инициалы и фамилии работников фирмы. Ввод заканчивается пустой строкой. Отсортировать строки в алфавитном порядке по фамилии.

Пример:

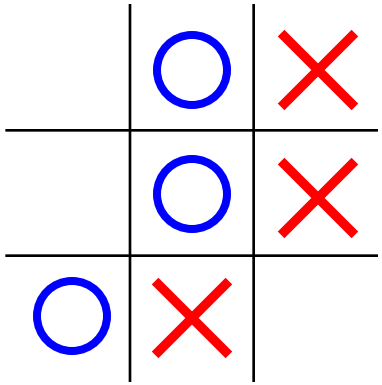
```
Введите ФИО:
А.Г. Урядников
Б.В. Тепловозов
В.Д. Арбузов

Список в алфавитном порядке:
В.Д. Арбузов
Б.В. Тепловозов
А.Г. Урядников
```

Программирование на языке C++

§ 67. 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?

Простые алгоритмы

Заполнение случайными числами:

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

Вложенный цикл!

Суммирование:

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

Задачи

«A»: 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 A:
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

«B»: 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)*

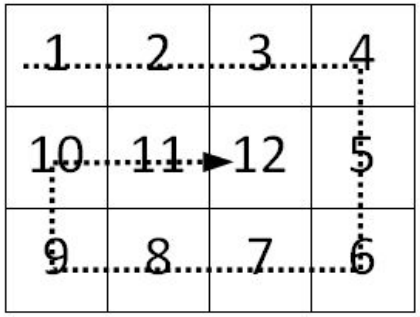
Пример:

```
Матрица A:
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
  0 255 255 255
255 255   0   0
255   0   0   0
```

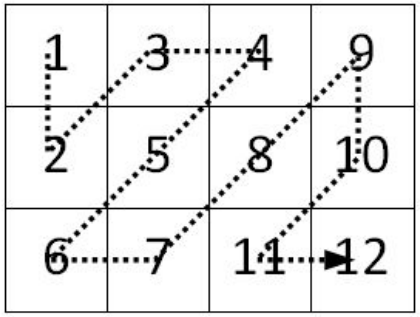
Задачи

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

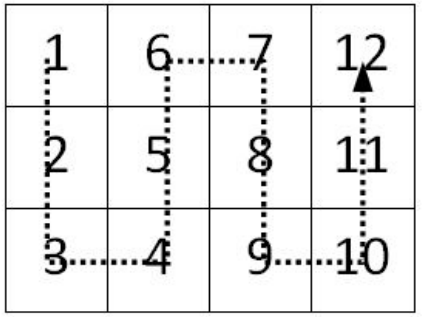
а)



б)



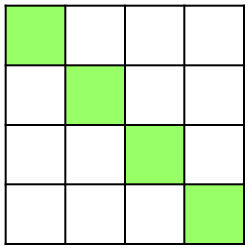
в)



Перебор элементов матрицы

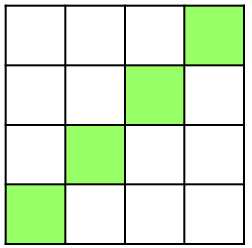
Главная диагональ:

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



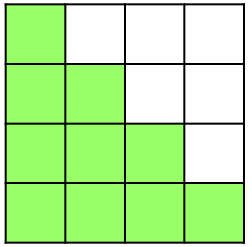
Побочная диагональ:

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



Главная диагональ и под ней:

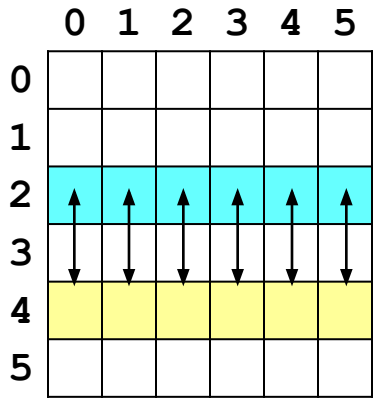
```
for ( int i=0; i<N; i++ )
  for ( int j=0; j<= i ; j++ )
    {
    // работаем с A[i][j]
    }
```



Перестановка строк

2-я и 4-я строки:

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



Задачи

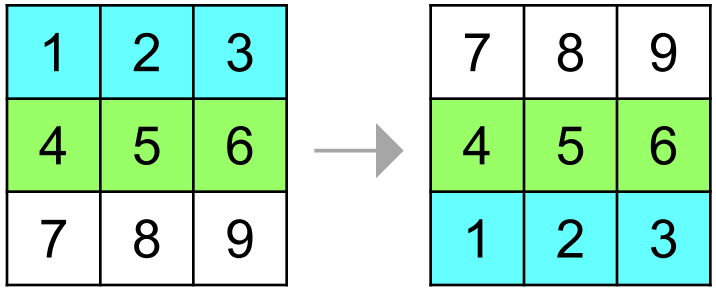
«A»: Напишите программу, которая заполняет квадратную матрицу случайными числами в интервале [10,99], а затем записывает нули во все элементы выше главной диагонали. Алгоритм не должен изменяться при изменении размеров матрицы.

Пример:

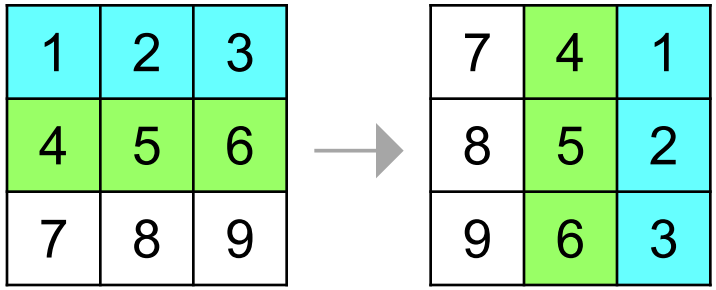
```
Матрица A:
12 14 67 45
32 87 45 63
69 45 14 30
40 12 35 65
Результат:
12  0  0  0
32 87  0  0
69 45 14  0
40 12 35 65
```

Задачи

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



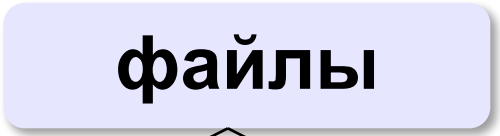
«C»: Пиксели рисунка закодированы числами (обозначающими цвет) в виде матрицы, содержащей N строк и M столбцов. Выполните поворот рисунка вправо на 90 градусов:



Программирование на языке C++

§ 68. Работа с файлами

Какие бывают файлы?



текстовые

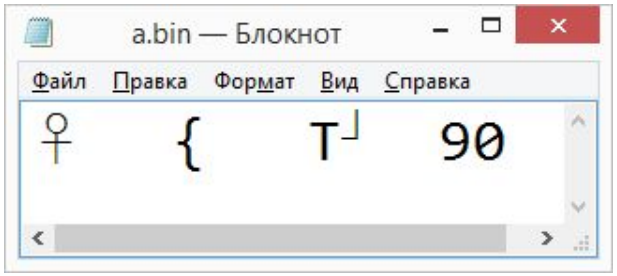
«***plain text***»:

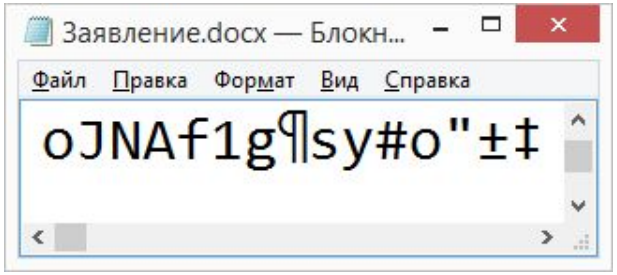
- для чтения человеком
- текст, разбитый на строки;
- из специальных символов только символы перехода на новую строку

```
12
123
1234
```

двоичные

- любые символы
- рисунки, звуки, видео, …





Принцип сэндвича

файловые потоки

```
#include <fstream>
```

```
ifstream Fin;  // поток ввода
ofstream Fout; // поток вывода
```

```
Fin.open ( "input.txt" );
Fout.open ( "output.txt" );
  // здесь работаем с файлами
Fin.close();
Fout.close();
```

Обработка ошибок

В случае неудачи поток нулевой!

```
ifstream F;
F.open ( "input.txt" );
if ( F )
   {
   // здесь работаем с файлом
   }
else
   printf ( "Открыть файл не удалось." );
```

```
if ( F.is_open() )
```

Когда такое может быть?

Ввод данных

```
int a, b;
ifstream Fin;
Fin.fopen ( "input.txt" );
 Fin >> a >> b;
Fin.close();
```

Переход к началу открытого файла:

```
Fin.close();
Fin.open( "input.txt" );
```

Определение конца файла:

```
if ( Fin.eof() )
  printf("Данные кончились");
```

eof = *end of file*, конец файла

Вывод данных в файл

```
int a = 1, b = 2;
ofstream Fout;
Fout.open( "output.txt" );
Fout << a << "+" << b << "=" << a+b;
Fout.close();
```

Режим добавления:

```
ofstream Fout( "output.txt",
               ios_base::app );
```

Чтение неизвестного количества данных

Задача. В файле записано в столбик неизвестное количество чисел. Найти их сумму.

```
пока не конец файла
  // прочитать число из файла
  // добавить его к сумме
```

```
int sum = 0, x;
while( ! Fin.eof() )
  {
  if ( Fin >> x )
    sum += x;
  }
```

Если удалось прочитать число, …

```
int sum = 0, x;
while( Fin >> x )
  sum += x;
```

Задачи

«A»: Напишите программу, которая находит среднее арифметическое всех чисел, записанных в файле в столбик, и выводит результат в другой файл.

«B»: Напишите программу, которая находит минимальное и максимальное среди чётных положительных чисел, записанных в файле, и выводит результат в другой файл. Учтите, что таких чисел может вообще не быть.

«C»: В файле в столбик записаны целые числа, сколько их – неизвестно. Напишите программу, которая определяет длину самой длинной цепочки идущих подряд одинаковых чисел и выводит результат в другой файл.

Обработка массивов

Задача. В файле записано не более 100 целых чисел. Вывести в другой текстовый файл те же числа, отсортированные в порядке возрастания.

? В чем отличие от предыдущей задачи?

! Для сортировки нужно удерживать все элементы в памяти одновременно.

```
const int MAX = 100;
int A[MAX];
```

Обработка массивов

Ввод массива:

Зачем?

```
N = 0;
while ( N < MAX  &&  !Fin.eof() )
   {
   if( Fin >> A[N] ) N++;
   }
```

Вывод результата:

```
Fout.open( "output.txt" );
for ( int i = 0; i < N ; i++ )
   Fout << A[i] << endl;
Fout.close();
```

Задачи

«A»: В файле записано не более 100 чисел. Отсортировать их по возрастанию последней цифры и записать в другой файл.

«B»: В файле записано не более 100 чисел. Отсортировать их по возрастанию суммы цифр и записать в другой файл. Используйте функцию, которая вычисляет сумму цифр числа.

«C»: В двух файлах записаны отсортированные по возрастанию массивы неизвестной длины. Объединить их и записать результат в третий файл. Полученный массив также должен быть отсортирован по возрастанию.

Обработка строк

Задача. В файле записано данные о собаках: в каждой строчке кличка собаки, ее возраст и порода:

```
Мухтар 4 немецкая овчарка
```

Вывести в другой файл сведения о собаках, которым меньше 5 лет.

```
пока не конец файла(Fin)
  // прочитать строку из файла Fin
  // разобрать строку – выделить возраст
  если возраст < 5 то
    // записать строку в файл Fout
```

Чтение строк из файла

Чтение одной строки:

```
string s;
getline( Fin, s );
```

строка

входной поток

При неудаче `getline` вернет `NULL`!

Чтение всех строк:

```
while ( getline(Fin, s) )
   {
   // обработать строку s
   }
```

Обработка строк

Разбор строки:

```
// найти в строке пробел
// удалить из строки кличку с первым пробелом
// найти в строке пробел
// выделить возраст перед пробелом
// преобразовать возраст в числовой вид
```

```
string s, s1;
int p, age;
...
p = s.find ( ' ' );
s1 = s.substr ( p + 1 );
age = atoi ( s1.c_str() );
```

p+1

Мухтар	4 немецкая овчарка

не влияет!

до конца строки

к формату строк Си

Задачи

«A»: В файле записаны данные о результатах сдачи экзамена. Каждая строка содержит фамилию, имя и количество баллов, разделенные пробелами:

```
<Фамилия> <Имя> <Количество баллов>
```

Вывести в другой файл фамилии и имена тех учеников, которые получили больше 80 баллов.

«B»: В предыдущей задаче добавить к полученному списку нумерацию, сократить имя до одной буквы и поставить перед фамилией:

```
1. Иванов
2. Петров
...
```

Задачи

«C»: В файле записаны данные о результатах сдачи экзамена. Каждая строка содержит фамилию, имя и количество баллов, разделенные пробелами:

```
<Фамилия> <Имя> <Количество баллов>
```

Вывести в другой файл данные учеников, которые получили больше 80 баллов. Список должен быть отсортирован по убыванию балла. Формат выходных данных:

```
П. Иванов   98
И. Петров   96
...
```

Конец фильма

ПОЛЯКОВ Константин Юрьевич
д.т.н., учитель информатики
ГБОУ СОШ № 163, г. Санкт-Петербург
kpolyakov@mail.ru

ЕРЕМИН Евгений Александрович
к.ф.-м.н., доцент кафедры мультимедийной дидактики и ИТО ПГГПУ, г. Пермь
eremin@pspu.ac.ru

Источники иллюстраций

1. www.mcdonalds.com
2. иллюстрации художников издательства «Бином»
3. авторские материалы