Пузырьковая сортировка

Пузырьковая сортировка по возрастанию – проходит по массиву снизу вверх *(от последнего элемента к первому)*, сравнивая каждый элемент массива с расположенным выше, и если верхний больше, то меняет их местами. При этом проходе наименьший элемент – "всплывет" наверх. Операция продолжается пока наименьший элемент не станет первым.

Затем операция повторяется над подмножеством массива с номерами (индексами) элементов от 2 до N, затем над подмножеством от 3 до N и так до подмножества N-1, N. То есть, до тех пор пока массив не будет отсортирован по возрастанию элементов.

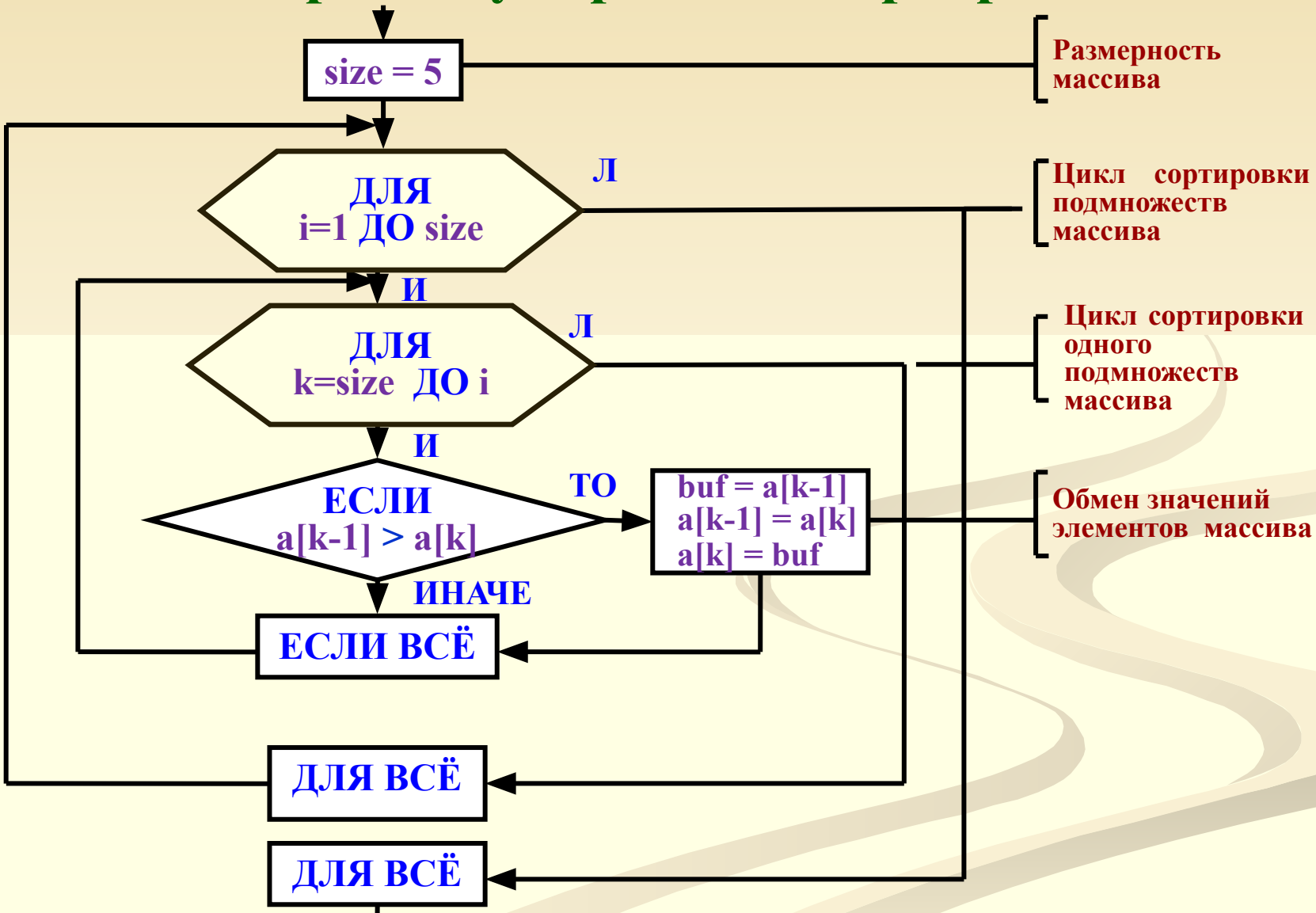
(При формировании условия сравнения "наибольший наверх" будет происходить сортировка по убыванию элементов массива).

Индекс	Исходный массив	1 шаг	2 шаг	3 шаг
1	6	2	2	2
2	4	6	3	3
3	3	4	6	4
4	2	3	4	6

На каждом шаге происходит три перестановки значений элементов.

МАССИВЫ

Алгоритм пузырьковой сортировки



Написать программу пузырьковой сортировки в стиле C++

Пузырьковая сортировка
в стиле C++

Предварительных замечаний относительно задачи:

Все функции надо определить как шаблоны встроенных функций. Цель использования шаблонов – обеспечить возможность обработки разнообразных типов данных.

Причина объявления функций в качестве встроенных имеет двоякий характер: во-первых, некоторые из них будут более эффективны; во-вторых, такое объявление позволяет размещать их, если это желательно, в заголовочных файлах.

Программа представлена .cpp-файлом, который включает заголовок sort.h. Этот заголовок содержит шаблоны двух функций:

- **print <T> (T*, int) печатает значения массива в линейной последовательности,**
- **swap <T> (T*, int, int) меняет местами два элемента массива.**

О массивах: Массив – это самый ясный и простой способ представления линейной последовательности элементов. Доступ к массиву data[n] является простейшим способом ссылки на n-ный элемент массива data.

Т.о. программа должна состоять из 3-х шаблонных функций (две в заголовке sort.h и одна - собственно сортировка пузырьком - в основном файле с программой.

В головной программе надо:

- **задать 6 массивов целых чисел: 7 3 8 2 1 5 4; 7 3 8 2 1 5 4 9 75 -5; 1 2 3; 3 2 1; 3 2 1 3; 3 3 3,**
- **отсортировать их и вывести на экран исходный массив, а под ним отсортированный массив,**

дополнительное задание:

- **задать массив символов, отсортировать его и вывести на экран.**

Пузырьковая сортировка

Файл sort.h, содержит вспомогательные функции для процедур сортировки:

```
#include <iostream>
using namespace std;
// Файл sort.h содержит две полезных вспомогательных функции:
// Swap( )  обменивает элементы в позициях pos1 и pos2 в пределах массива
   template <class T>
    inline  void  swap(T array[], int pos1, int pos2)
     {
       T temp;
       temp = array[pos1];
       array[pos1] = array[pos2];
       array[pos2] = temp;
      }
// Print( ) распечатывает элементы массива в линейной последовательности
    template <class T>
     inline  void  print(T array[], int size)
      {
        int i;
        for (i=0; i < size; ++i)
               { cout << array[i] << " ";  }
        cout << endl;
       }
```

Пузырьковая сортировка

Файл сортировки bs.cpp содержит шаблон bubble_sort() и головную функцию main():

```
/* Файл bs.cpp реализует шаблон функции bubble_sort(), которая сортирует элементы своего входного
массива в восходящем порядке.  Тип T должен поддерживать operator=() и operator<().  Для
инициализации может потребоваться копирование. Если требуется печать, необходима operator<<().
Аргумент array[] содержит элементы, требующие сортировки, а size является числом элементов этого
массива. Сортировка делается "по месту." Допускаются повторяющиеся элементы. */
#include "psort.h"
template <class T>
inline void bubble_sort(T array[], int size)
    {
     int i, j;
  /* Верхний предел внешнего цикла равен size-1,
а не  size, так как если все прочие элементы
заняли свои места,  наибольший автоматически
оказывается в правильной позиции */
     for (i=0; i < size-1; ++i)
       {
         for  (j=size-1; j > i; --j )
           {
             if (array[j-1] > array[j])
                swap(array, j-1, j);
           }
       }
    }                          // см. продолжение
```

Функция bubble_sort(T*, int) принимает параметр-массив и его размер. Вся функция состоит из внешнего цикла, определяющего, какой из подмассивов обрабатывается в данный момент, и внутреннего цикла, который сканирует данный подмассив и обменивает при необходимости соседние значения. Сортировка проводится «по месту». Повторяющиеся значения не вызывают никаких неприятностей.

Внешний цикл выполняется size-1, а не size раз.

Как только все остальные элементы заняли правильные места, наибольший элемент также должен автоматически занять свое место.

Внутренний цикл for проходит каждый из подмассивов в обратном порядке, в направлении, противоположном внешнему циклу. Можно сделать наоборот, при этом не наименьший элемент будет «всплывать», а наибольший — «тонуть» на «дно» текущего подмассива.

Вывод программы показывает, что повторяющиеся значения обрабатываются правильно.

Применение ключевого слова inline не является существенным, и его всегда можно опустить (inline - модификатор определения функции. Предписывает компилятору помещать расширение тела функции везде, где происходит обращение к ней, вместо того, чтобы генерировать код вызова. Встроенные функции позволяют увеличить быстродействие программы. Однако применение больших inline - функций может привести к увеличению объема кода программы)

Пузырьковая сортировка

```
int main()                          // продолжение
{
   int array_1[] = {7, 3, 8, 2, 1, 5, 4};
 print(array_1, 7);
  bubble_sort (array_1, 7);
 print(array_1, 7);
  cout << endl;
   int array_2[] = {7, 3, 8, 2, 1, 5, 4, 9, 75, -5};
 print(array_2, 10);
  bubble_sort(array_2, 10);
 print(array_2, 10);
  cout << endl;
   int array_3[] = {1, 2, 3};
 print(array_3, 3);
  bubble_sort (array_3, 3);
 print(array_3, 3);
  cout << endl;
  int array_4[] = {3, 2, 1};
 print(array_4, 3);
  bubble_sort (array_4 , 3);
 print(array_4, 3);
  cout << endl;
  int array_5[] = {3, 2, 1, 3};
 print(array_5, 4);
  bubble_sort (array_5, 4);
 print(array_5, 4);
  cout << endl;
  int array_6[] = {3, 3, 3};
 print(array_6, 3);
  bubble_sort(array_6, 3);
 print(array_6, 3);
  cout << endl;
return 0;
}
```

Вывод программы пузырьковой сортировки bs.cpp.

Каждая пара строк показывает целый массив до и после сортировки.

```
7 3 8 2 1 5 4
1 2 3 4 5 7 8

7  3 8 2 1 5 4 9 75 -5
-5 1 2 3 4 5 7 8  9  75

1 2 3
1 2 3

3 2 1
1 2 3

3 2 1 3
1 2 3 3

3 3 3
3 3 3
```