Федеральное агентство связи

Федеральное государственное бюджетное образовательное

учреждение высшего образования

Поволжский государственный университет

телекоммуникаций и информатики

Лабораторная работа №5

По дисциплине «Алгоритмы и структуры данных»

Тема: «Сортировка одномерных массивов»

Выполнил студент

группы ИСТ-82 Логинов Ю.

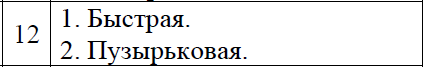
Проверил

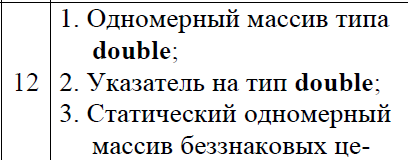
Назаренко П. А.

Самара 2021

***Цель работы***

Изучить основные алгоритмы сортировки массивов и освоить их на практике. Проверить работу алгоритмов на различных наборах данных.



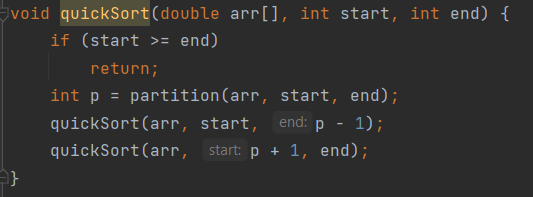


***Порядок выполнения работы***

1. Определить массив, элементы которого будут упорядочиваться. Тип массива выбрать по таблице №1 – массив №1.



2. Разработать функцию сортировки массива методами, выбранными по таблице 3 в соответствии с вариантом.

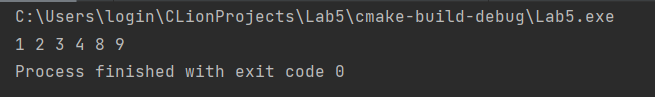


3. Любым способом заполнить элементы массива значениями.



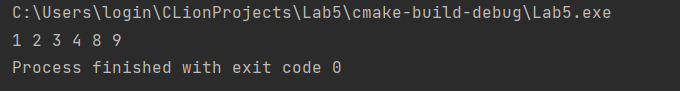
4. Выполнить сортировку массива первым алгоритмом и проконтролировать ее результат.



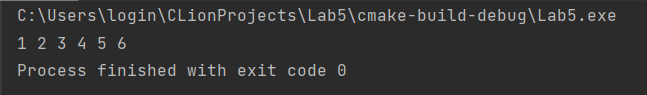


Проверить все варианты исходного заполнения массива: случайным образом,



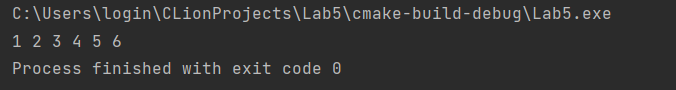
 отсортированного в обратном порядке,





отсортированного в требуемом порядке.



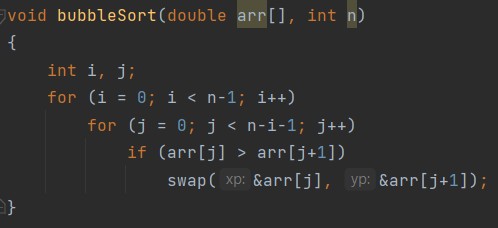


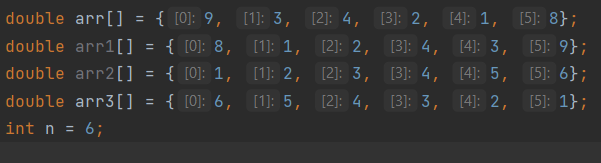
Убедиться в правильности сортировки во всех случаях. Сделать выводы.

**Сортировка во всех случаях прошла успешно.**

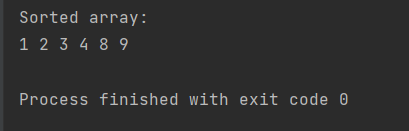
5. Повторить пункты 3 и 4 для второго алгоритма сортировки.

**Пузырьковая сортировка.**

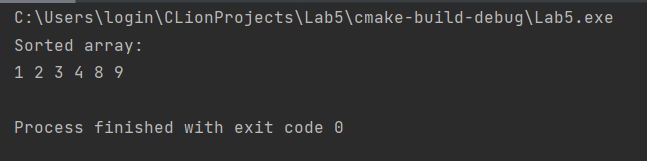
****

****

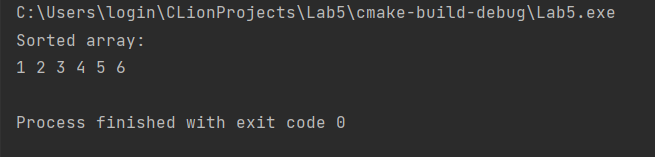
Сортировка arr:



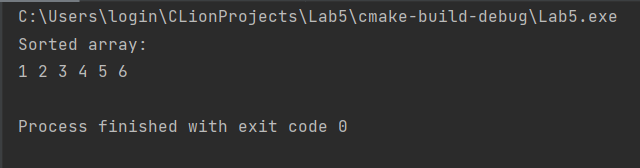
Сортировка arr1:



Сортировка arr2:



Сортировка arr3:



**Сортировка во всех случаях прошла успешно.**

Листинг программы:

#include <iostream>  
  
using namespace std;  
  
double arr[] = {9, 3, 4, 2, 1, 8};  
double arr1[] = {8, 1, 2, 4, 3, 9};  
double arr2[] = {1, 2, 3, 4, 5, 6};  
double arr3[] = {6, 5, 4, 3, 2, 1};  
int n = 6;  
  
  
int partition(double arr[], int start, int end) {  
 int pivot = arr[start];  
 int count = 0;  
 for (int i = start + 1; i <= end; i++) {  
 if (arr[i] <= pivot)  
 count++;  
 }  
 int pivotIndex = start + count;  
 swap(arr[pivotIndex], arr[start]);  
 int i = start, j = end;  
 while (i < pivotIndex && j > pivotIndex) {  
 while (arr[i] <= pivot) {  
 i++;  
 }  
 while (arr[j] > pivot) {  
 j--;  
 }  
 if (i < pivotIndex && j > pivotIndex) {  
 swap(arr[i++], arr[j--]);  
 }  
 }  
 return pivotIndex;  
}  
  
void quickSort(double arr[], int start, int end) {  
 if (start >= end)  
 return;  
 int p = partition(arr, start, end);  
 quickSort(arr, start, p - 1);  
 quickSort(arr, p + 1, end);  
}  
  
  
void swap(double \*xp, double \*yp) {  
 int temp = \*xp;  
 \*xp = \*yp;  
 \*yp = temp;  
}  
  
void bubbleSort(double arr[], int n) {  
 int i, j;  
 for (i = 0; i < n - 1; i++)  
 for (j = 0; j < n - i - 1; j++)  
 if (arr[j] > arr[j + 1])  
 swap(&arr[j], &arr[j + 1]);  
}  
  
void printArray(double arr[], int size) {  
 int i;  
 for (i = 0; i < size; i++)  
 cout << arr[i] << " ";  
 cout << endl;  
}  
  
void task1(double arr[], int n) {  
 quickSort(arr3, 0, n - 1);  
 cout << "Sorted array: \n";  
 for (int i = 0; i < n; i++) {  
 cout << arr3[i] << " ";  
 }  
 cout << endl;  
}  
  
void task2(double arr[], int n) {  
 bubbleSort(arr, n);  
 cout << "Sorted array: \n";  
 printArray(arr, n);  
}  
  
int main() {  
 task1(arr1, n);  
 task2(arr3, n);  
 return 0;  
}

**Вывод:** в данной работе мы изучили алгоритмы сортировки массивов и освоили их на практике, проверили работу алгоритмов на различных наборах данных.