**Московский государственный технический**

**университет им. Н.Э. Баумана**

Факультет «Информатика и системы управления»

Кафедра ИУ5 «Системы обработки информации и управления»

Курс «Основы информатики»

Отчет по лабораторной работе №5

«Сортировки»

|  |  |  |
| --- | --- | --- |
| Выполнил: |  | Проверил: |
| студент группы ИУ5-13Б |  | преподаватель каф. ИУ5 |
| Лачина Екатерина |  | Аксенова М.В. |
| Подпись и дата: |  | Подпись и дата: |

Москва, 2023 г.

2) Разработка алгоритма

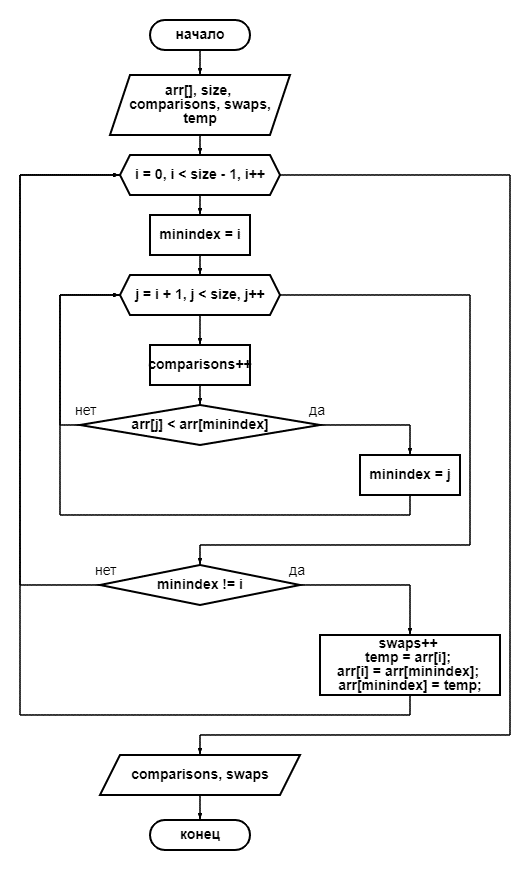
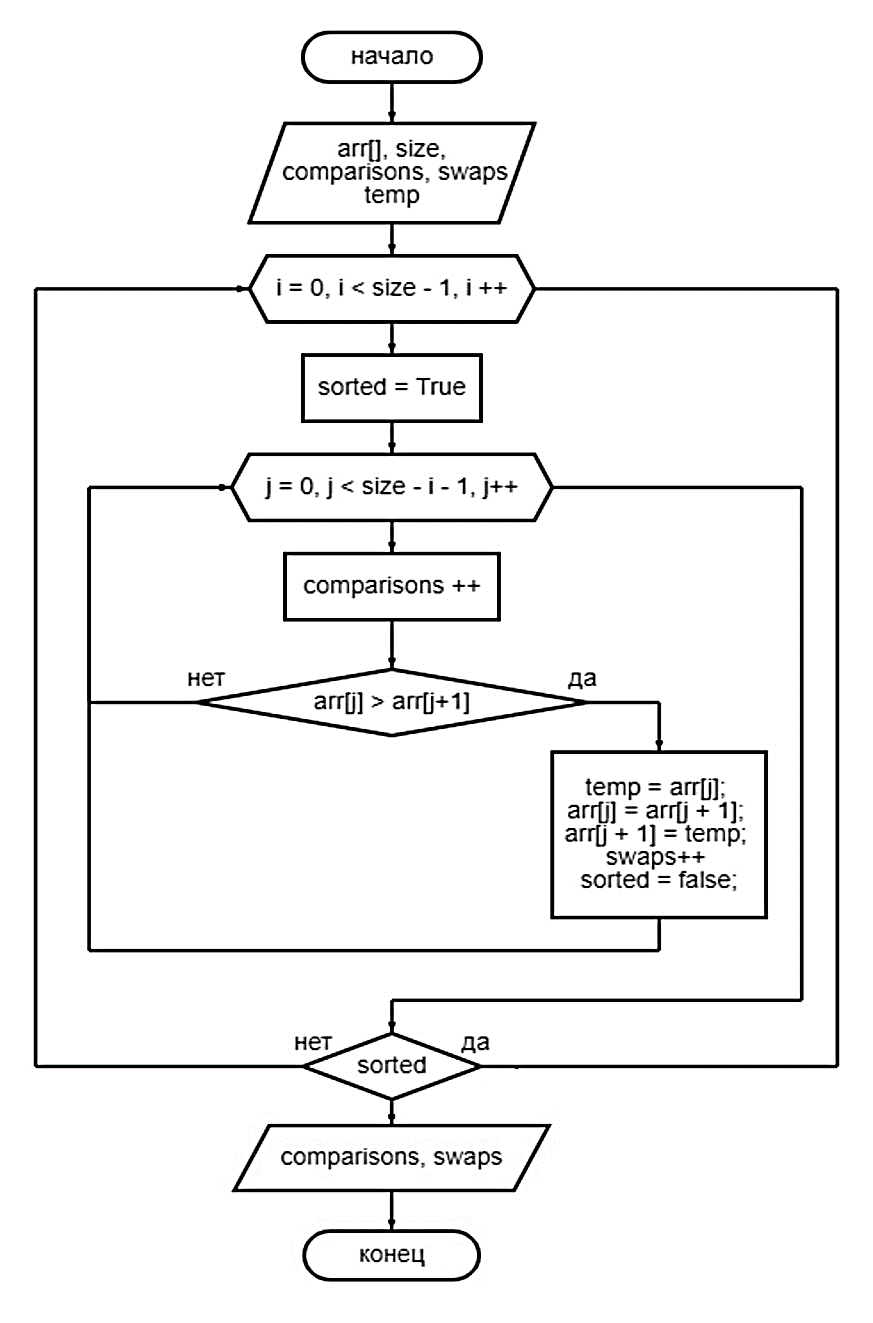
Функции, файл special.cpp

|  |  |  |
| --- | --- | --- |
| функция | Значение | Переменные в функции |
| int random(int a, int b); | генерация рандомного числа в заданном диапозоне | a – начало диапозона  b - конец |
| void output\_list(int arr[], int size); | вывод массива | arr[] – массив для вывода  size – его размер |
| void selectionsort(int arr[], int size, int comparisons, int swaps); | функция сортировки методом выбора по возрастанию | arr[] – массив для сортировки  size – его размер  comparisons – кол-во сравнений  swaps – кол-во перестановок  temp – временная переменная  minindex – минимальный индекс |
| void selectionsort\_reverse(int arr[], int size, int comparisons, int swaps); | функция сортировки методом выбора по убыванию | arr[] – массив для сортировки  size – его размер  comparisons – кол-во сравнений  swaps – кол-во перестановок  temp – временная переменная  maxindex – максимальный индекс |
| void bubblesort(int arr[], int size, int comparisons, int swaps); | функция сортировки методом пузырька по возрастанию | arr[] – массив для сортировки  size – его размер  comparisons – кол-во сравнений  swaps – кол-во перестановок  temp – временная переменная  bool sorted – флаг для проверки отсортирован ли массив |
| void bubblesort\_reverse(int arr[], int size, int comparisons, int swaps); | функция сортировки методом пузырька по убыванию | arr[] – массив для сортировки  size – его размер  comparisons – кол-во сравнений  swaps – кол-во перестановок  temp – временная переменная  bool sorted – флаг для проверки отсортирован ли массив |

Переменные для файла reslab5.cpp

|  |  |
| --- | --- |
| Переменная типа **int** | Значение |
| size\_analyse | размер массива для примера |
| arr\_analyse | динамический массив для примера |
| arrCopy\_analyse | копия массива для сортировки пузырьком для примера |
| sizes | размеры массивов для исследования |
| size | размер массива с которым я работаю |
| arr | динамический массив |
| arrCopy | копия массива для сортировки пузырьком |

Selectionsort Bubblesort



3) Листинг

special.cpp

#include <iostream>

#include <stdlib.h> // srand, rand

using namespace std;

**//генерация рандомного числа в заданном диапозоне**

int random(int a, int b) {

return a + (b - a + 1) \* rand() / RAND\_MAX;

}

void output\_list(int arr[], int size) {

for (int i = 0; i < size; i++) {

cout << arr[i] << " ";

}

cout << endl;}

**// функция сортировки методом выбора ПО ВОЗВРАСТАНИЮ**

void selectionsort(int arr[], int size, int comparisons, int swaps) {

int temp = 0;

for (int i = 0; i < size - 1; i++) {

int minindex = i;

for (int j = i + 1; j < size; j++) {

comparisons++; // увеличиваю счетчик сравнений

if (arr[j] < arr[minindex]) {

minindex = j;

}

}

if (minindex != i) {

swaps++; // увеличиваю счетчик перестановок

temp = arr[i];

arr[i] = arr[minindex];

arr[minindex] = temp;

}

}

cout << "selectionsort comparisons: " << comparisons;

cout << " selectionsort swaps: " << swaps << endl;}

**// функция сортировки методом выбора ПО УБЫВАНИЮ**

void selectionsort\_reverse(int arr[], int size, int comparisons, int swaps) {

int temp = 0;

for (int i = 0; i < size - 1; i++) {

int maxindex = i;

for (int j = i + 1; j < size; j++) {

comparisons++; // увеличиваю счетчик сравнений

if (arr[j] > arr[maxindex]) {

maxindex = j;

}

}

if ((maxindex != i) && (arr[maxindex]!=arr[i-1])) {

swaps++; // увеличиваю счетчик перестановок

temp = arr[i];

arr[i] = arr[maxindex];

arr[maxindex] = temp;

}

}

cout << "selectionsort comparisons: " << comparisons;

cout << " selectionsort swaps: " << swaps << endl;}

**// функция сортировки методом пузырька ПО ВОЗВРАСТАНИЮ**

void bubblesort(int arr[], int size, int comparisons, int swaps) {

int temp = 0;

for (int i = 0; i < size - 1; i++) {

bool sorted = true;

for (int j = 0; j < size - i - 1; j++) {

comparisons++; // увеличиваю счетчик сравнений

if (arr[j] > arr[j + 1]) {

temp = arr[j];

arr[j] = arr[j + 1];

arr[j + 1] = temp;

swaps++; // увеличиваю счетчик перестановок

sorted = false;

}

}

if (sorted) {

break;

}

}

cout << "bubblesort comparisons: " << comparisons;

cout << " bubblesort swaps: " << swaps << endl;}

**// функция сортировки методом пузырька ПО УБЫВАНИЮ**

void bubblesort\_reverse(int arr[], int size, int comparisons, int swaps) {

int temp = 0;

for (int i = 0; i < size - 1; i++) {

bool sorted = true;

for (int j = 0; j < size - i - 1; j++) {

comparisons++; // увеличиваю счетчик сравнений

if (arr[j] < arr[j + 1]) {

temp = arr[j];

arr[j] = arr[j + 1];

arr[j + 1] = temp;

swaps++; // увеличиваю счетчик перестановок

sorted = false;

}

}

if (sorted) {

break;

}

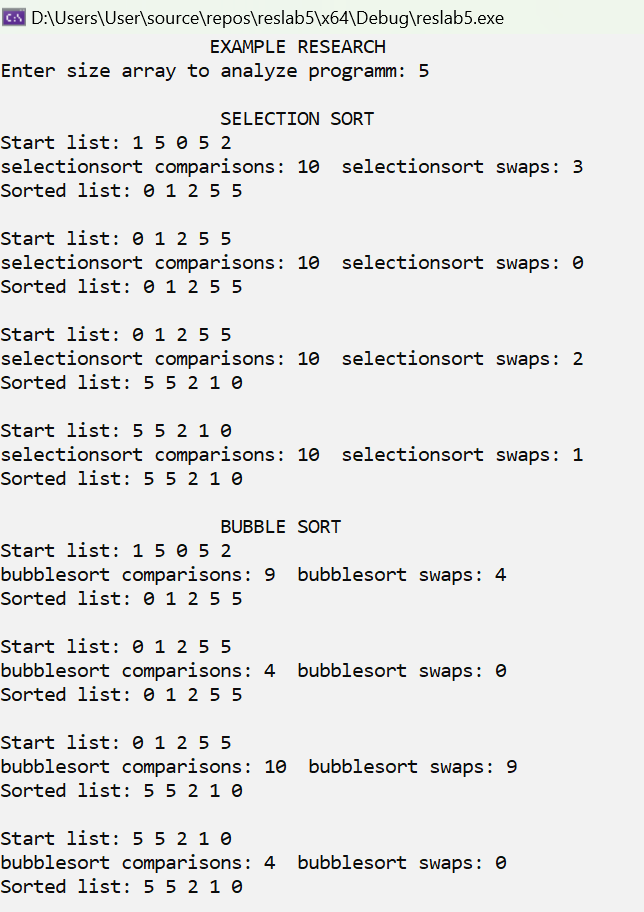
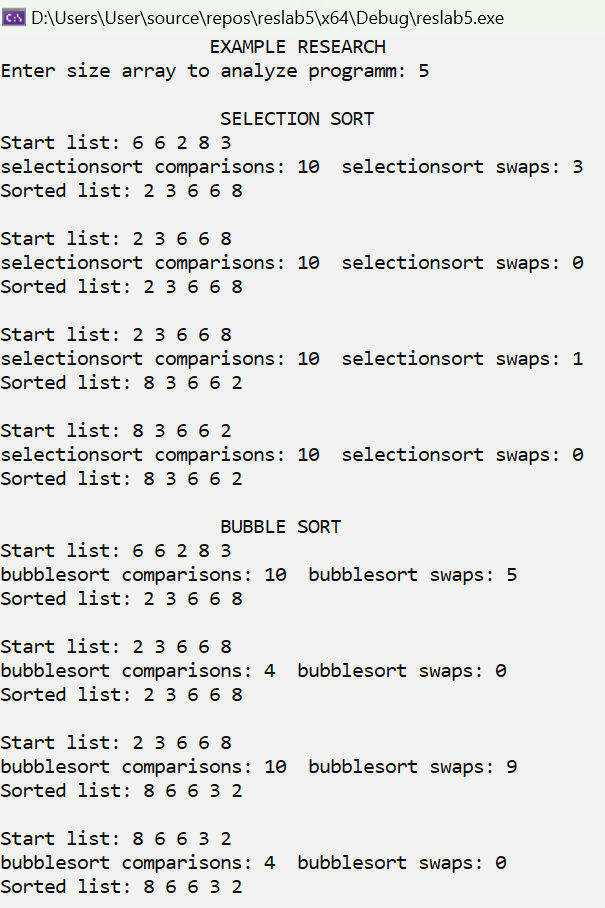
}

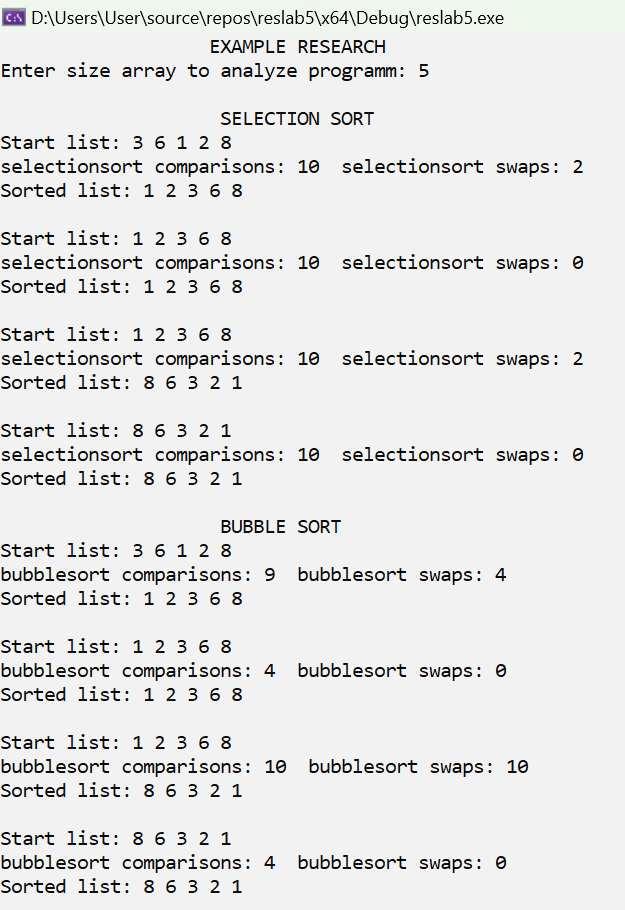
cout << "bubblesort comparisons: " << comparisons;

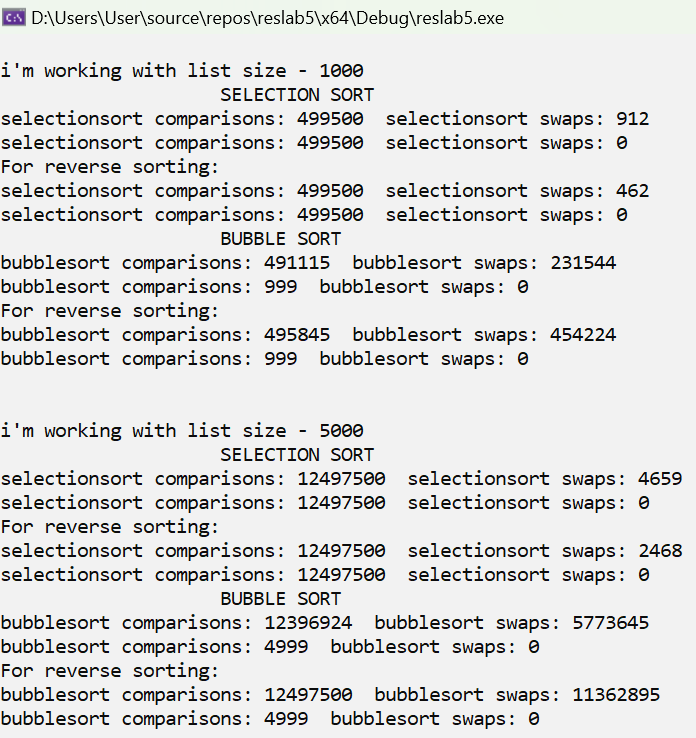
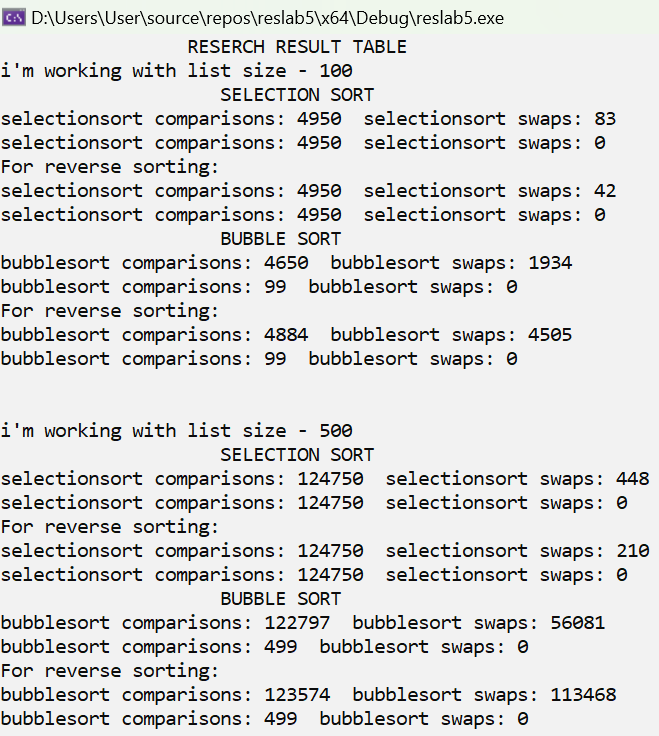
cout << " bubblesort swaps: " << swaps << endl;

}

4) Тестирование





5) Вывод

И метод сортировки пузырьком (bubble sort), и метод сортировки выбором (selection sort) являются простыми и базовыми алгоритмами сортировки, но они обладают разной эффективностью.

Метод сортировки пузырьком имеет временную сложность O(n^2). Он работает путем многократного обмена соседних элементов, если они находятся в неправильном порядке, постепенно перемещая наибольшие элементы в конец массива, пока весь массив не будет отсортирован. Метод сортировки пузырьком неэффективен для больших наборов данных, так как требует большое количество сравнений и обменов, даже если массив уже отсортирован.

Метод сортировки выбором также имеет временную сложность O(n^2). Он работает путем многократного нахождения минимального элемента из неотсортированной части массива и обмена его с первым элементом неотсортированной части. Метод сортировки выбором работает быстрее, чем метод сортировки пузырьком, потому что он сокращает количество необходимых обменов. В отличие от метода сортировки пузырьком, он выполняет только один обмен за проход. Однако он все равно имеет большое количество сравнений и обменов, что делает его неэффективным для больших наборов данных.

В отношении сравнений и обменов метод сортировки пузырьком и метод сортировки выбором имеют похожую производительность в среднем и в худшем случае. Для неотсортированного массива размером n каждый из них требует примерно n^2/2 сравнений и обменов.

Однако в лучшем случае, когда массив уже отсортирован, метод сортировки пузырьком работает лучше. Он избегает необходимости лишних сравнений и обменов, так как на каждом проходе проверяет, были ли сделаны какие-либо обмены. В отличие от него, метод сортировки выбором все равно выполняет то же количество сравнений и обменов, независимо от исходного порядка массива. В худшем случае (в неотсортированном массиве) лучше метод сортировки выбором, так как, тогда количество перестановок будет меньше.

В заключение, ни метод сортировки пузырьком, ни метод сортировки выбором не считаются эффективными для больших наборов данных. Однако, если необходимо выбрать между ними, метод сортировки пузырьком будет чуть более эффективным в лучшем случае.