**Московский государственный технический**

**университет им. Н.Э. Баумана**

Факультет «Информатика и системы управления»

Кафедра ИУ5 «Системы обработки информации и управления»

Отчет по лабораторной работе №

«**Сортировка одномерного числового массива**»

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Выполнил:** | |  | **Принял:** | |
| ФИО: | \_Цыпышев Т. А.\_\_\_\_\_ |  | ФИО: | \_**Самохвалов А. Э.**\_\_\_ |
| Группа: | \_ИУ5-11Б\_\_\_\_\_\_\_\_\_\_ |  | Должность: | \_Преподаватель\_\_\_\_ |
| Дата: | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |  | Дата: | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |
| Подпись: | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |  | Подпись: | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |

Москва, 2022 г.

**Постановка задачи**

Отсортировать числовой массив методом выбора максимального (минимального) элемента и методом пузырькового всплытия. По окончании сортировки вывести отсортированный массив и количество сделанных сравнений и перестановок элементов.

Сравнить быстродействие алгоритмов, которое определяется числом сравнений и перестановок, для исходного не отсортированного массива и для исходного массива, отсортированного в прямом и обратном порядке.

Исследовать зависимость быстродействия от размера массива. Возможность изменения длины массива реализуйте с помощью динамического массива, а для его инициализации используйте датчик случайных чисел. Результаты исследования выведите в виде отформатированной таблицы.

**Разработка алгоритма**

Функция int GetRandomNumber(int a, int b);

* Входные данные – границы диапазона
* Выходные данные – случайное число из диапазона [a,b]

Функция int \*CreatingArray(int size);

* Входные данные – size (длина массива)
* Выходные данные – массив длины size, заполненный случайными числами.

Функция void SortingBubble(int \*Array, int size);

Сортировка массива arr длины len методом пузырькового всплытия

* Входные данные – указатель на исходный массив, size (длина массива)

Функция void SortingMin(int \*arr, int size);

Сортировка массива arr длины len методом выбора минимального элемента

* Входные данные – указатель на исходный массив, size (длина массива)

Функция void SortingMax(int \*arr, int size);

Сортировка массива arr длины len методом выбора максимального элемента

* Входные данные – указатель на исходный массив, size (длина массива)

Функция void ReversArray(int \*source, int size);

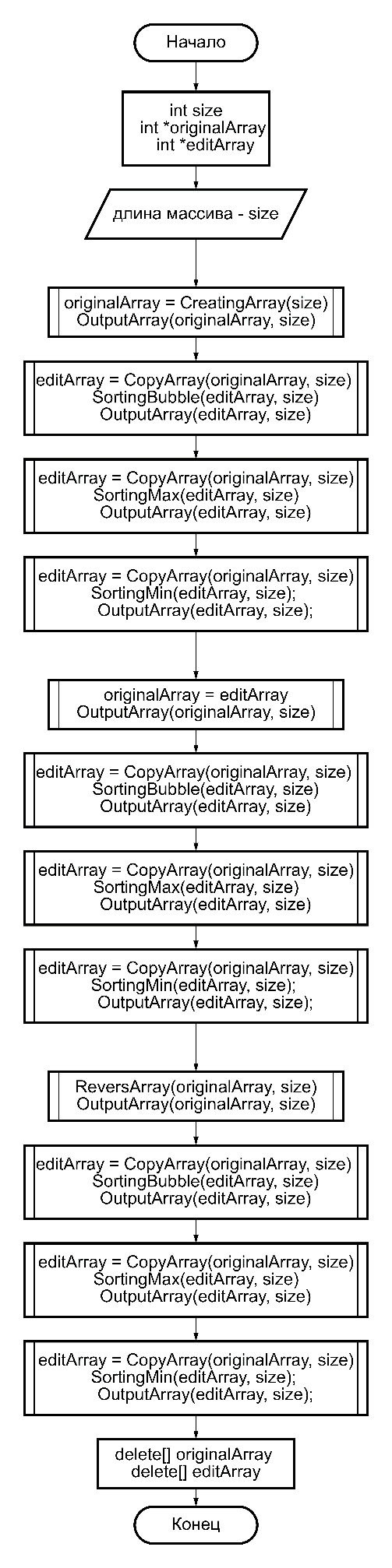
* Входные данные – указатель на исходный массив, size (длина массива)

Функция void OutputArray(int \*source, int size);

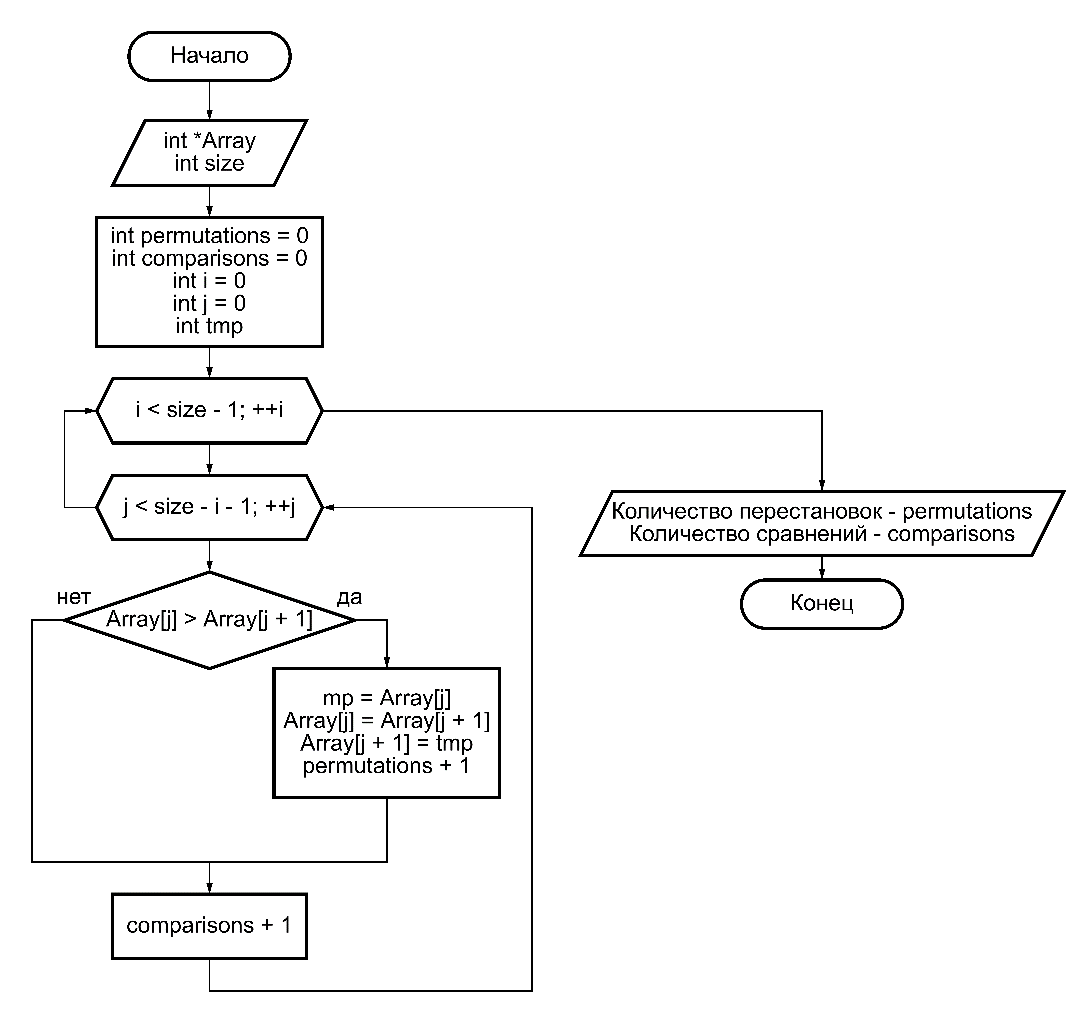
* Входные данные – указатель на исходный массив, size (длина массива)

**Схема алгоритма**

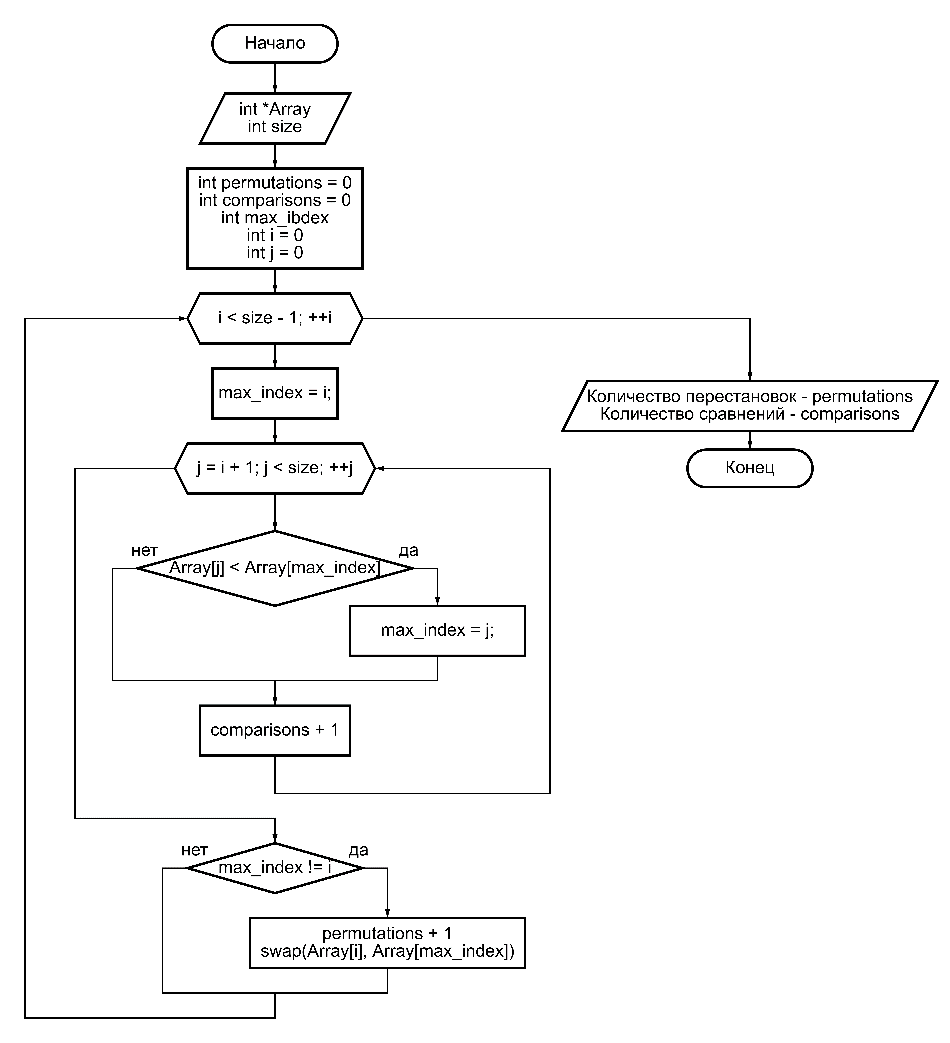
Main.cpp



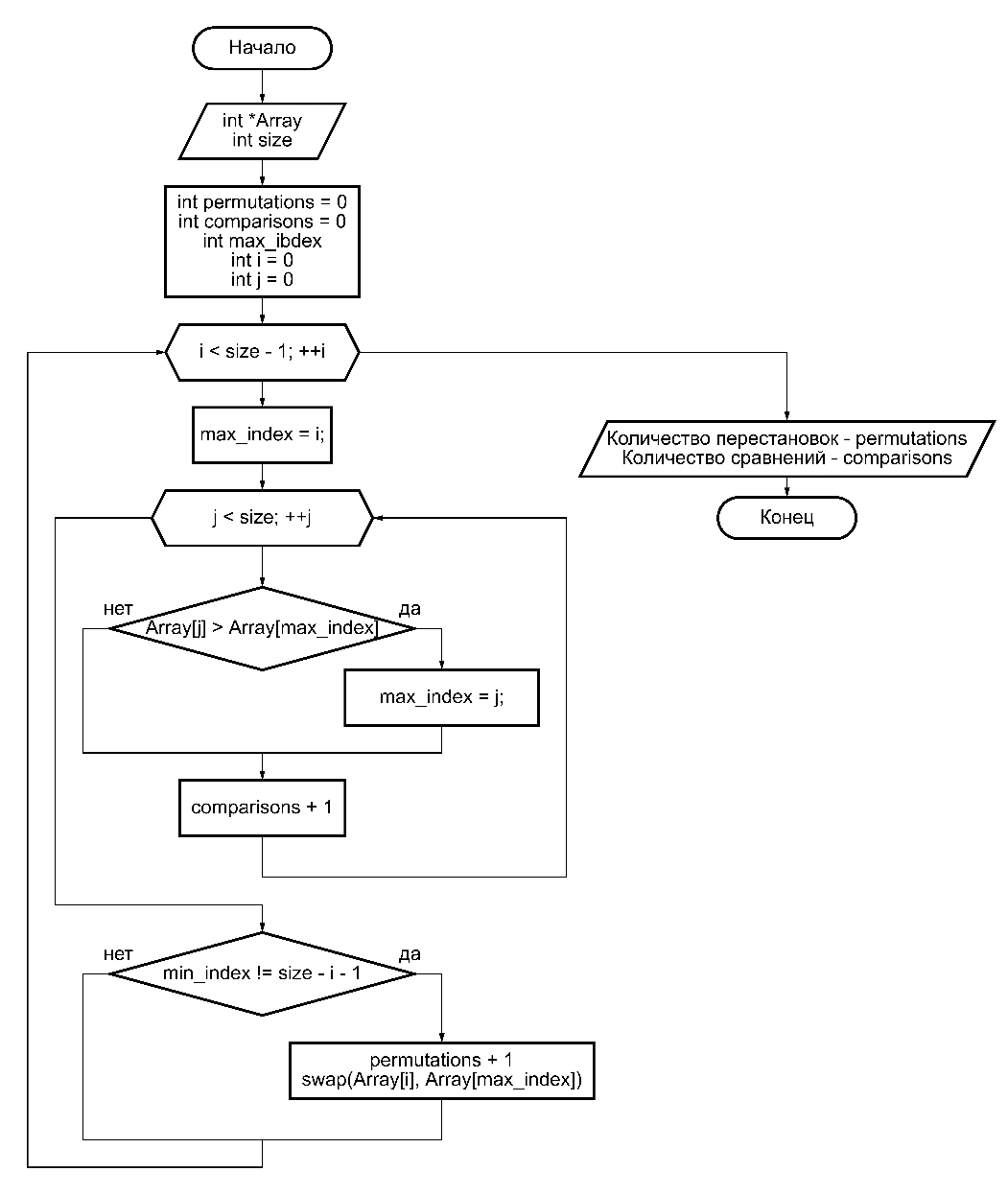
Method\_Bubble.cpp



Method\_MinMax.cpp (SortingMin)

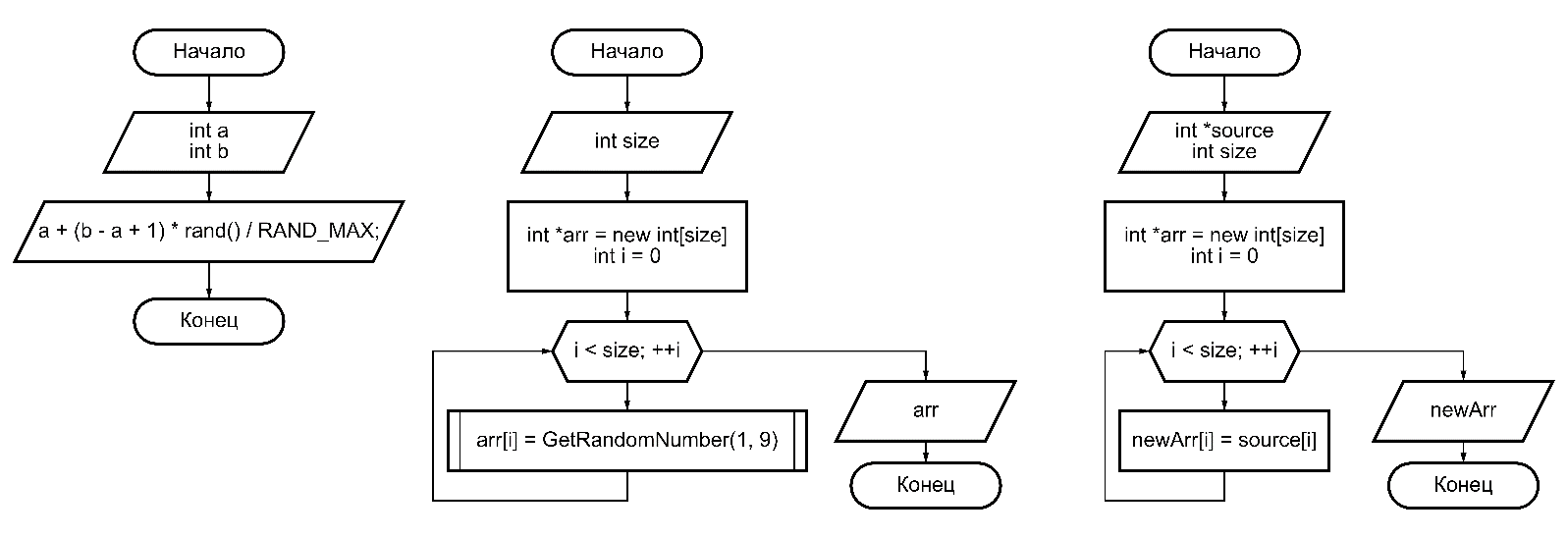


Method\_MinMax.cpp (SortingMax)



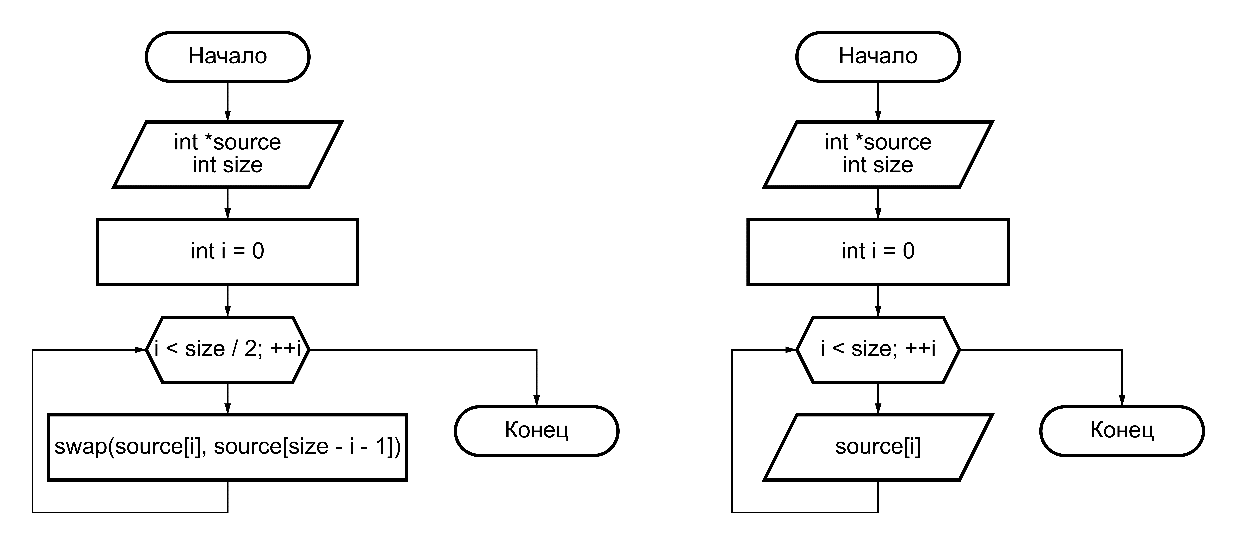
other\_functions.cpp other\_functions.cpp other\_functions.cpp

(GetRandomNumber) (CreatingArray) (CopyArray)



other\_functions.cpp other\_functions.cpp

(ReversArray) (OutputArray)



**Текст программы**

Main.cpp

/// Copyright 2022 ttsypyshev <ttsypyshev01@gmail.com>

#include <iostream>

#include "../include/all\_functios.h"

using namespace std;

int main() {

system("chcp 65001");

int size;

int \*originalArray;

int \*editArray;

cout << "Введите длину массива: ";

cin >> size;

cout << "\n\n\n---Сортировка рандомно заданного массива---\n";

originalArray = CreatingArray(size);

cout << "Исходный массив: ";

OutputArray(originalArray, size);

cout << "\n\nМетод пузырькового всплытия";

editArray = CopyArray(originalArray, size);

SortingBubble(editArray, size);

OutputArray(editArray, size);

cout << "\n\nМетод выбора максимального элемента";

editArray = CopyArray(originalArray, size);

SortingMax(editArray, size);

OutputArray(editArray, size);

cout << "\n\nМетод выбора минимального элемента";

editArray = CopyArray(originalArray, size);

SortingMin(editArray, size);

OutputArray(editArray, size);

cout << "\n\n\n---Сортировка такого же предварительно отсортированного массива---\n";

originalArray = editArray;

cout << "Исходный массив: ";

OutputArray(originalArray, size);

cout << "\n\nМетод пузырькового всплытия";

editArray = CopyArray(originalArray, size);

SortingBubble(editArray, size);

OutputArray(editArray, size);

cout << "\n\nМетод выбора максимального элемента";

editArray = CopyArray(originalArray, size);

SortingMax(editArray, size);

OutputArray(editArray, size);

cout << "\n\nМетод выбора минимального элемента";

editArray = CopyArray(originalArray, size);

SortingMin(editArray, size);

OutputArray(editArray, size);

cout << "\n\n\n---Сортировка такого же предварительно отсортированного в обратном порядке массива---\n";

ReversArray(originalArray, size);

cout << "Исходный массив: ";

OutputArray(originalArray, size);

cout << "\n\nМетод пузырькового всплытия";

editArray = CopyArray(originalArray, size);

SortingBubble(editArray, size);

OutputArray(editArray, size);

cout << "\n\nМетод выбора максимального элемента";

editArray = CopyArray(originalArray, size);

SortingMax(editArray, size);

OutputArray(editArray, size);

cout << "\n\nМетод выбора минимального элемента";

editArray = CopyArray(originalArray, size);

SortingMin(editArray, size);

OutputArray(editArray, size);

delete[] originalArray;

delete[] editArray;

return 0;

}

Method\_Bubble.cpp

/// Copyright 2022 ttsypyshev <ttsypyshev01@gmail.com>

#include <iostream>

using namespace std;

void SortingBubble(int \*Array, int size) {

int permutations = 0, comparisons = 0, i, j;

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

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

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

int tmp = Array[j];

Array[j] = Array[j + 1];

Array[j + 1] = tmp;

++permutations;

}

++comparisons;

}

}

cout << "\nКоличество перестановок - " << permutations << ", количество сравнений - " << comparisons << "\n";

}

Method\_MinMax.cpp

/// Copyright 2022 ttsypyshev <ttsypyshev01@gmail.com>

#include <iostream>

using namespace std;

void SortingMin(int \*Array, int size) {

int permutations = 0, comparisons = 0, max\_index, i, j;

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

max\_index = i;

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

if (Array[j] < Array[max\_index])

max\_index = j;

++comparisons;

}

if (max\_index != i) {

++permutations;

swap(Array[i], Array[max\_index]);

}

}

cout << "\nКоличество перестановок - " << permutations << ", количество сравнений - " << comparisons << "\n";

}

void SortingMax(int \*Array, int size) {

int permutations = 0, comparisons = 0, min\_index, i, j;

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

min\_index = 0;

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

if (Array[j] > Array[min\_index])

min\_index = j;

++comparisons;

}

if (min\_index != size - i - 1) {

++permutations;

swap(Array[size - i - 1], Array[min\_index]);

}

}

cout << "\nКоличество перестановок - " << permutations << ", количество сравнений - " << comparisons << "\n";

}

other\_functions.cpp

/// Copyright 2022 ttsypyshev <ttsypyshev01@gmail.com>

#include <iostream>

#include <stdlib.h>

#include <time.h>

using namespace std;

int GetRandomNumber(int a, int b) {

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

}

int \*CreatingArray(int size) {

int \*arr = new int[size];

srand((unsigned int) time(NULL));

rand( );

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

arr[i] = GetRandomNumber(1, 9);;

return arr;

}

int \*CopyArray(int \*source, int size) {

int \*newArr = new int[size];

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

newArr[i] = source[i];

return newArr;

}

void ReversArray(int \*source, int size) {

for (int i = 0; i < size / 2; ++i) {

swap(source[i], source[size - i - 1]);

}

}

void OutputArray(int \*source, int size) {

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

cout << source[i] << ' ';

}

}

all\_functios.h

/// Copyright 2022 ttsypyshev <ttsypyshev01@gmail.com>

#ifndef INF\_LAB\_05\_ALL\_FUNCTIOS\_H

#define INF\_LAB\_05\_ALL\_FUNCTIOS\_H

//method bubble

void SortingBubble(int \*Array, int size);

//method min/max

void SortingMax(int \*Array, int size);

void SortingMin(int \*Array, int size);

//other functions (creating, random, revers)

int \*CreatingArray(int size);

int \*CopyArray(int \*source, int size);

void ReversArray(int \*source, int size);

void OutputArray(int \*source, int size);

#endif //INF\_LAB\_05\_ALL\_FUNCTIOS\_H

**Анализ результатов**

Сортировка случайного массива

Массив размером 5

Исходный массив - 9 7 5 7 5

|  |  |  |  |
| --- | --- | --- | --- |
|  | Сравнений | Перестановок | Отсортированный массив |
| Выбор по минимуму | 10 | 4 | 5 5 7 7 9 |
| Выбор по максимуму | 14 | 4 | 5 5 7 7 9 |
| Метод пузырька | 10 | 7 | 5 5 7 7 9 |

Массив размером 100

Исходный массив – 9 9 7 7 6 3 2…

|  |  |  |  |
| --- | --- | --- | --- |
|  | Сравнений | Перестановок | Отсортированный массив |
| Выбор по минимуму | 4950 | 88 | 1 1 1 1 1 1 ... |
| Выбор по максимуму | 5049 | 97 | 1 1 1 1 1 1 ... |
| Метод пузырька | 4950 | 2096 | 1 1 1 1 1 1 ... |

Массив размером 1000

Исходный массив – 9 4 6 1 6 2 4...

|  |  |  |  |
| --- | --- | --- | --- |
|  | Сравнений | Перестановок | Отсортированный массив |
| Выбор по минимуму | 499500 | 883 | 1 1 1 1 1 1 ... |
| Выбор по максимуму | 500499 | 998 | 1 1 1 1 1 1 ... |
| Метод пузырька | 499500 | 227835 | 1 1 1 1 1 1 ... |

Сортировка предварительно отсортированного массива

Массив размером 5

Исходный массив - 9 2 8 1 3

|  |  |  |  |
| --- | --- | --- | --- |
|  | Сравнений | Перестановок | Отсортированный массив |
| Выбор по минимуму | 10 | 0 | 1 2 3 8 9 |
| Выбор по максимуму | 18 | 0 | 1 2 3 8 9 |
| Метод пузырька | 10 | 0 | 1 2 3 8 9 |

Массив размером 100

Исходный массив – 9 4 7 2 3 7 5…

|  |  |  |  |
| --- | --- | --- | --- |
|  | Сравнений | Перестановок | Отсортированный массив |
| Выбор по минимуму | 4950 | 0 | 1 1 1 1 1 1 ... |
| Выбор по максимуму | 5049 | 0 | 1 1 1 1 1 1 ... |
| Метод пузырька | 4950 | 0 | 1 1 1 1 1 1 ... |

Массив размером 1000

Исходный массив – 8 1 4 6 4 2 3 3…

|  |  |  |  |
| --- | --- | --- | --- |
|  | Сравнений | Перестановок | Отсортированный массив |
| Выбор по минимуму | 499500 | 0 | 1 1 1 1 1 1 ... |
| Выбор по максимуму | 500499 | 0 | 1 1 1 1 1 1 ... |
| Метод пузырька | 499500 | 0 | 1 1 1 1 1 1 ... |

Сортировка предварительно отсортированного в обратном порядке массива

Массив размером 5

Исходный массив – 9 7 7 7 5

|  |  |  |  |
| --- | --- | --- | --- |
|  | Сравнений | Перестановок | Отсортированный массив |
| Выбор по минимуму | 10 | 7 | 5 7 7 7 9 |
| Выбор по максимуму | 14 | 3 | 5 7 7 7 9 |
| Метод пузырька | 10 | 1 | 5 7 7 7 9 |

Массив размером 100

Исходный массив – 9 4 2 4 5 5 9 4 3…

|  |  |  |  |
| --- | --- | --- | --- |
|  | Сравнений | Перестановок | Отсортированный массив |
| Выбор по минимуму | 4950 | 69 | 1 1 1 1 1 1 ... |
| Выбор по максимуму | 5049 | 95 | 1 1 1 1 1 1 ... |
| Метод пузырька | 4950 | 4420 | 1 1 1 1 1 1 ... |

Массив размером 1000

Исходный массив – 9 2 9 8 6 6 8 8...

|  |  |  |  |
| --- | --- | --- | --- |
|  | Сравнений | Перестановок | Отсортированный массив |
| Выбор по минимуму | 499500 | 679 | 1 1 1 1 1 1 ... |
| Выбор по максимуму | 500499 | 995 | 1 1 1 1 1 1 ... |
| Метод пузырька | 499500 | 444094 | 1 1 1 1 1 1 ... |

**Вывод**

Проанализировав результаты выполнения программы при разных входных значениях, можно сделать вывод, что при малой размерности массива эффективность методов выбора и пузырькового всплытия примерно одинакова, но по мере увеличения длины эффективность методов выбора по отношению к методу пузырька растет экспоненциально.