**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

«Сортировка одномерного числового массива»

|  |  |  |
| --- | --- | --- |
| Выполнил: |  | Проверил: |
| студент группы ИУ5-11 |  | преподаватель каф. ИУ5 |
| Михалёв Ярослав |  | Козлов А.Д. |
| Подпись и дата: |  | Подпись и дата: |
|  |  |  |

Москва, 2021 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 (см. Приложение 1). Результаты исследования выведите в виде отформатированной таблицы.

Разработка алгоритма

Входные данные

* int\* arr = CreateArray(size) – массив случайных чисел
* int compCount - число сравнений
* int swapCount - число перестановок
* int SSminComp, SSmaxComp, BSComp – число сравнений
* int SSminSwap, SSmaxSwap, BSSwap – число перестановок
* int\* aCopy – копия массива

Метод выбора максимального(минимального) элемента

Входные данные

* int arr[] – указатель на неотсортированный массив
* int size – размер массива
* int & compCount – количество сравнений
* int & swapCount – количество перестановок
* bool bymax – флаг, сортировать ли массив по возрастанию (по умолчанию true)

Выходные данные:

* Отсортированный массив

Метод пузырька

Входные данные

* int arr[] – указатель на неотсортированный массив
* int size – размер массива
* int & compCount – количество сравнений
* int & swapCount – количество перестановок
* bool bymax – флаг, сортировать ли массив по возрастанию (по умолчанию true)

Выходные данные:

* Отсортированный массив

CreateArray

Создаёт массив, наполняет его случайными числами

Входные данные

* int size – размер массива

Выходные данные

* Массив случайных чисел

CopyArray

Копирует массив по ссылке

Входные данные

* int\* source – сслыка на массив, который нужно скопировать
* int size – размер массива, который нужно скопировать

Выходные данные

* Копия массива

PrintArray

Выводит все элементы массива

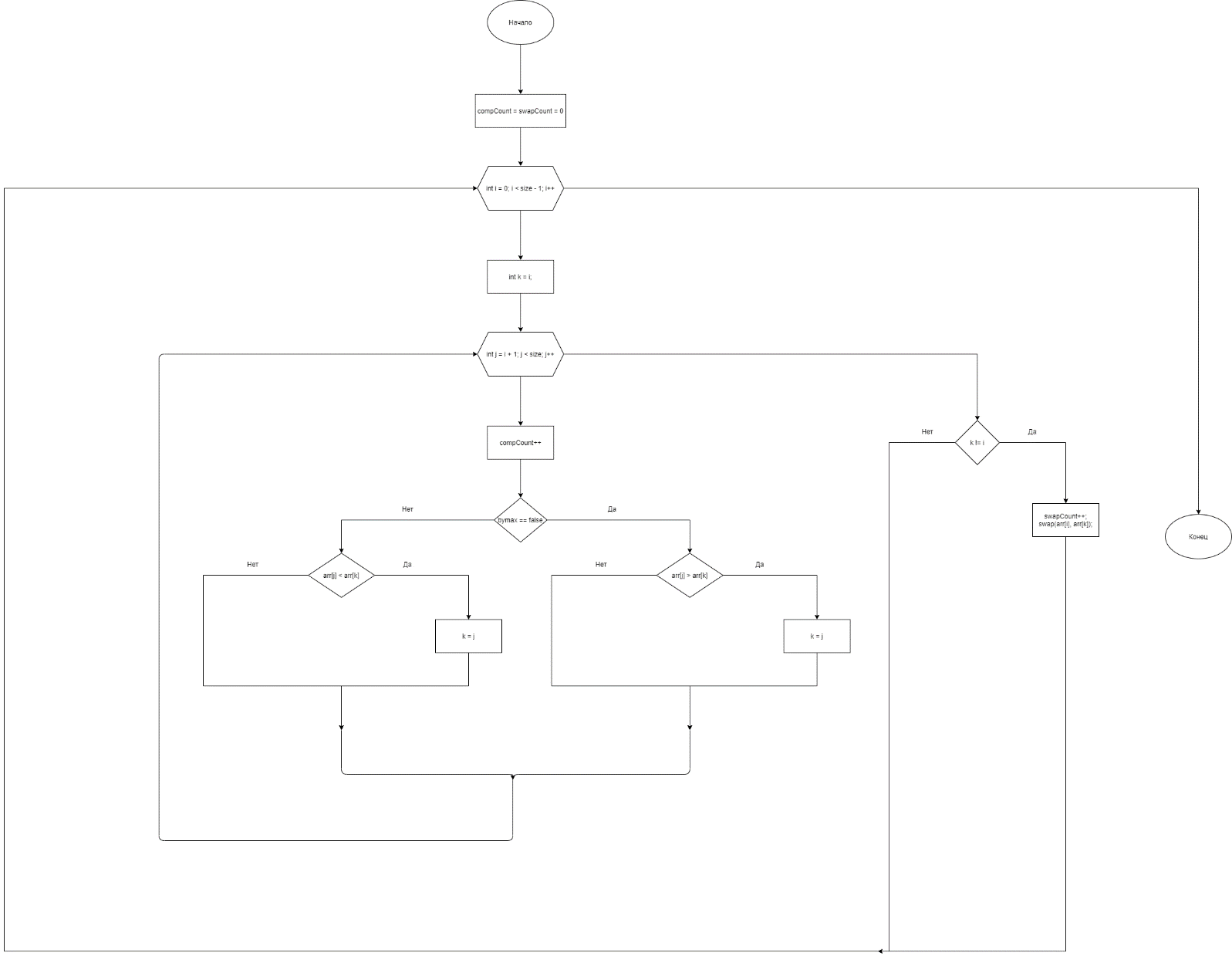
Входные данные

* int arr[] – ссылка на массив
* int size – размер массива

Схема алгоритма

SelectionSort.cpp

SelectionSortMax

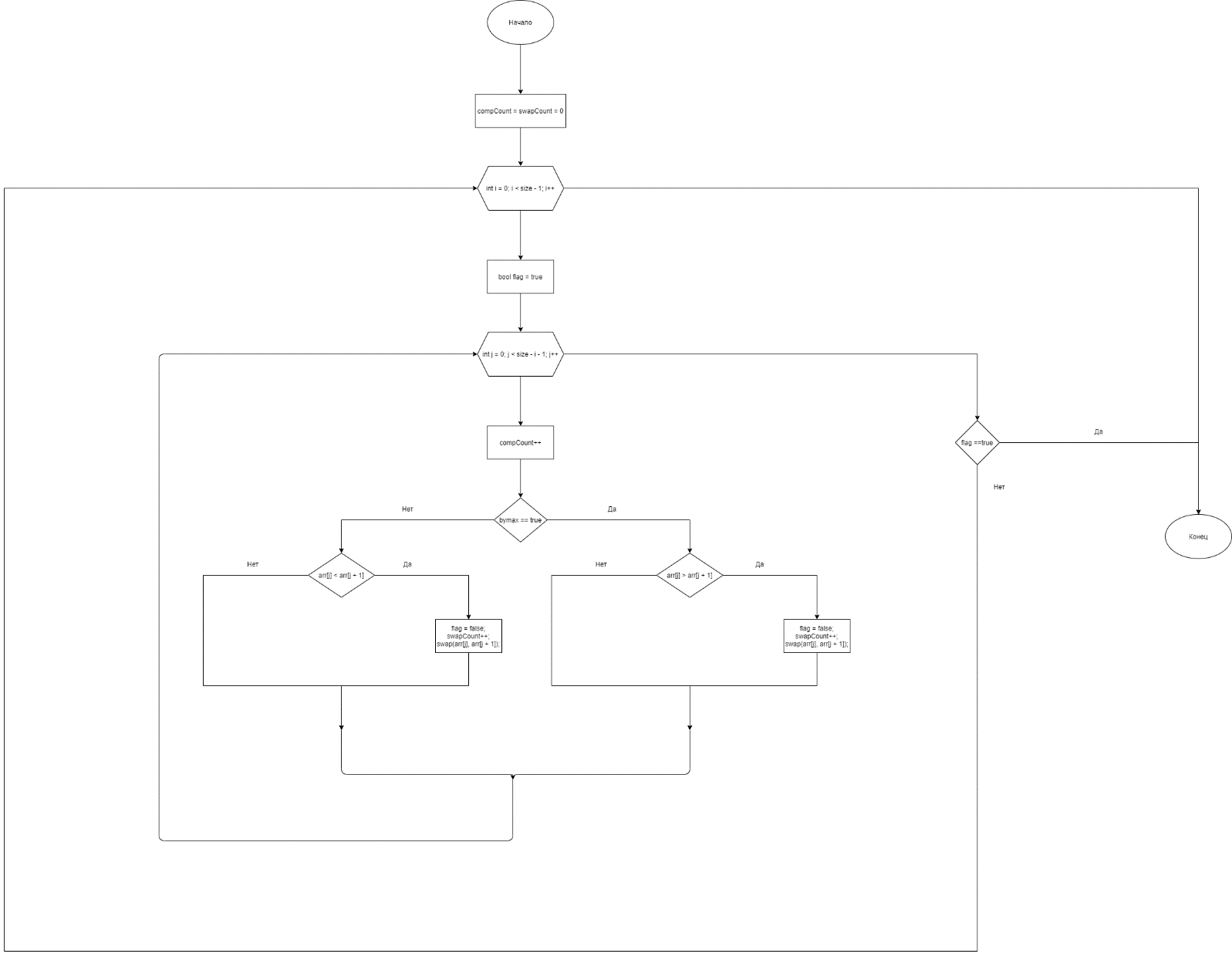


SelectionSortMin



BubbleSort.cpp

BubbleSort



Текст программы

Main.cpp

#include <iostream>

#include "SelectionSort.h"

#include "BubbleSort.h"

#include "Utils.h"

#include "Randomizer.h"

**using** **namespace** std**;**

void test**(**int size**,** bool bymax**)**

**{**

h2**(**"Массив размером "**,** size**);**

int**\*** arr **=** CreateArray**(**size**);**

cout **<<** "Исходный массив - "**;**

printArray**(**arr**,** size**);**

cout **<<** endl**;**

int compCount**;** // число сравнений

int swapCount**;** // число перестановок

int SSminComp**,** SSmaxComp**,** BSComp**;**

int SSminSwap**,** SSmaxSwap**,** BSSwap**;**

int**\*** aCopy1**;**

int**\*** aCopy2**;**

int**\*** aCopy3**;**

aCopy1 **=** CopyArray**(**arr**,** size**);**

SelectionSortMax**(**aCopy1**,** size**,** compCount**,** swapCount**,** bymax**);**

SSminComp **=** compCount**;**

SSminSwap **=** swapCount**;**

aCopy2 **=** CopyArray**(**arr**,** size**);**

SelectionSortMin**(**aCopy2**,** size**,** compCount**,** swapCount**,** bymax**);**

SSmaxComp **=** compCount**;**

SSmaxSwap **=** swapCount**;**

aCopy3 **=** CopyArray**(**arr**,** size**);**

BubbleSort**(**aCopy3**,** size**,** compCount**,** swapCount**,** bymax**);**

BSComp **=** compCount**;**

BSSwap **=** swapCount**;**

printString**(**" "**);**

printString**(**"Сравнений"**);**

printString**(**"Перестановок"**);**

printString**(**"Массив"**);**

cout **<<** "\n"**;**

printString**(**"Выбор по минимуму"**);**

printNumber**(**SSminComp**);**

printNumber**(**SSminSwap**);**

printArray**(**aCopy1**,** size**);**

cout **<<** "\n"**;**

printString**(**"Выбор по максимуму"**);**

printNumber**(**SSmaxComp**);**

printNumber**(**SSmaxSwap**);**

printArray**(**aCopy2**,** size**);**

cout **<<** "\n"**;**

printString**(**"Метод пузырька"**);**

printNumber**(**BSComp**);**

printNumber**(**BSSwap**);**

printArray**(**aCopy3**,** size**);**

cout **<<** "\n\n"**;**

**delete[]** aCopy1**,** aCopy2**,** aCopy3**;**

**}**

int main**()**

**{**

system**(**"chcp 1251 > nul"**);**

Randomize**();**

h1**(**"Сортировка по возрастанию"**);**

test**(**5**,** **true);**

test**(**100**,** **true);**

test**(**1000**,** **true);**

h1**(**"Сортировка по убыванию"**);**

test**(**5**,** **false);**

test**(**100**,** **false);**

test**(**1000**,** **false);**

**}**

SelectionSort.cpp

#include "SelectionSort.h"

#include <iostream>

**using** **namespace** std**;**

void SelectionSortMax**(**int arr**[],** int size**,** int **&** compCount**,** int **&** swapCount**,** bool bymax**)**

**{**

compCount **=** swapCount **=** 0**;**

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

**{**

int k **=** i**;**

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

**{**

compCount**++;**

**if** **(!**bymax**)**

**{**

**if** **(**arr**[**j**]** **>** arr**[**k**])**

k **=** j**;**

**}**

**else**

**{**

**if** **(**arr**[**j**]** **<** arr**[**k**])**

k **=** j**;**

**}**

**}**

**if** **(**k **!=** i**)** **{**

swapCount**++;**

swap**(**arr**[**i**],** arr**[**k**]);**

**}**

**}**

**}**

void SelectionSortMin**(**int arr**[],** int size**,** int **&** compCount**,** int **&** swapCount**,** bool bymax**)**

**{**

compCount **=** swapCount **=** 0**;**

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

**{**

int k **=** i**;**

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

**{**

compCount**++;**

**if** **(**bymax**)**

**{**

**if** **(**arr**[**j**]** **<** arr**[**k**])**

k **=** j**;**

**}**

**else**

**{**

**if** **(**arr**[**j**]** **>** arr**[**k**])**

k **=** j**;**

**}**

**}**

**if** **(**k **!=** i**)** **{**

swapCount**++;**

swap**(**arr**[**i**],** arr**[**k**]);**

**}**

**}**

**}**

BubbleSort.cpp

#include "BubbleSort.h"

#include <iostream>

**using** **namespace** std**;**

void BubbleSort**(**int arr**[],** int size**,** int**&** compCount**,** int**&** swapCount**,** bool bymax**)**

**{**

compCount **=** swapCount **=** 0**;**

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

**{**

bool flag **=** **true;**

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

**{**

compCount**++;**

**if** **(**bymax**)**

**{**

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

**{**

flag **=** **false;**

swapCount**++;**

swap**(**arr**[**j**],** arr**[**j **+** 1**]);**

**}**

**}**

**else**

**{**

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

**{**

flag **=** **false;**

swapCount**++;**

swap**(**arr**[**j**],** arr**[**j **+** 1**]);**

**}**

**}**

**}**

**if** **(**flag**)**

**break;**

**}**

**}**

Анализ результатов

Сортировка по возрастанию

Массив размером 5

Исходный массив - 2 9 2 2 4

|  |  |  |  |
| --- | --- | --- | --- |
|  | Сравнений | Перестановок | Отсортированный массив |
| Выбор по минимуму | 10 | 3 | 2 2 2 4 9 |
| Выбор по максимуму | 10 | 3 | 2 2 2 4 9 |
| Метод пузырька | 7 | 3 | 2 2 2 4 9 |

Массив размером 100

Исходный массив - 5 9 2 5 1 3 …

|  |  |  |  |
| --- | --- | --- | --- |
|  | Сравнений | Перестановок | Отсортированный массив |
| Выбор по минимуму | 4950 | 86 | 1 1 1 1 1 1 ... |
| Выбор по максимуму | 4950 | 86 | 1 1 1 1 1 1 ... |
| Метод пузырька | 4872 | 2074 | 1 1 1 1 1 1 ... |

Массив размером 1000

Исходный массив - 1 3 6 5 1 8 ...

|  |  |  |  |
| --- | --- | --- | --- |
|  | Сравнений | Перестановок | Отсортированный массив |
| Выбор по минимуму | 499500 | 886 | 1 1 1 1 1 1 ... |
| Выбор по максимуму | 499500 | 886 | 1 1 1 1 1 1 ... |
| Метод пузырька | 493284 | 221028 | 1 1 1 1 1 1 ... |

Сортировка по убыванию

Массив размером 5

Исходный массив - 3 7 4 4 2 ...

|  |  |  |  |
| --- | --- | --- | --- |
|  | Сравнений | Перестановок | Отсортированный массив |
| Выбор по минимуму | 10 | 3 | 7 4 4 3 2 |
| Выбор по максимуму | 10 | 3 | 7 4 4 3 2 |
| Метод пузырька | 7 | 3 | 7 4 4 3 2 |

Массив размером 100

Исходный массив - 6 6 4 3 4 4 ...

|  |  |  |  |
| --- | --- | --- | --- |
|  | Сравнений | Перестановок | Отсортированный массив |
| Выбор по минимуму | 4950 | 79 | 9 9 9 9 9 9 … |
| Выбор по максимуму | 4950 | 79 | 9 9 9 9 9 9 … |
| Метод пузырька | 4830 | 2356 | 9 9 9 9 9 9 … |

Массив размером 1000

Исходный массив - 7 5 1 7 8 1 ...

|  |  |  |  |
| --- | --- | --- | --- |
|  | Сравнений | Перестановок | Отсортированный массив |
| Выбор по минимуму | 499500 | 860 | 9 9 9 9 9 9 … |
| Выбор по максимуму | 499500 | 860 | 9 9 9 9 9 9 … |
| Метод пузырька | 495495 | 222170 | 9 9 9 9 9 9 … |

Вывод

Проанализировав результаты программы, можно сказать, что при маленьком размере массива эффективность методов выбора и пузырька примерно одинакова, но чем большее размер массива, тем большее преимущества у сортировки методом выбором перед сортировкой методом пузырька