**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 ИУ5-13Б |  | преподаватель каф. ИУ5 |
| Поляков Данила |  | Аксёнова М.В. |
| 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. Результаты исследования выведите в виде отформатированной таблицы.

**Разработка алгоритма**

**Описание алгоритма**

В данной лабораторной работе используются два метода сортировки – сортировка пузырьком и сортировка выбором.

Алгоритм сортировки пузырьком состоит в повторяющихся проходах по сортируемому массиву. На каждой итерации последовательно сравниваются соседние элементы, и, если порядок в паре неверный, то элементы меняют местами. За каждый проход по массиву как минимум один элемент встает на свое место, поэтому необходимо совершить не более **n – 1** проходов, где **n** -размер массива, чтобы отсортировать массив. Так же как некое подобие оптимизации используется проверка на то, была ли совершена хоть одна перестановка после прохода внутреннего цикла. Если нет, то массив уже отсортирован и дальнейшие проходы цикла не совершаются.

Алгоритм сортировки выбором заключается в следующем:

на каждом **i**-м шаге цикла алгоритм находит максимальный/минимальный (в зависимости от направления сортировки) элемент из **(n-i)** элементов и ставит его в начало массива.

**Описание используемых переменных**

int[] arrToSort – массив для сортировки, передающийся в функции

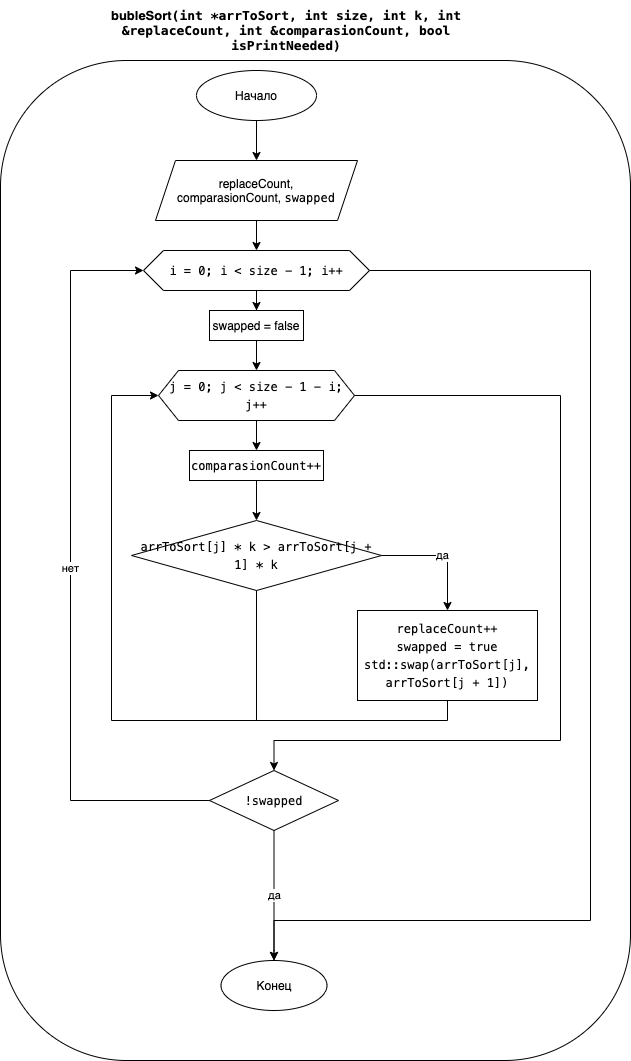
int size – размер массива

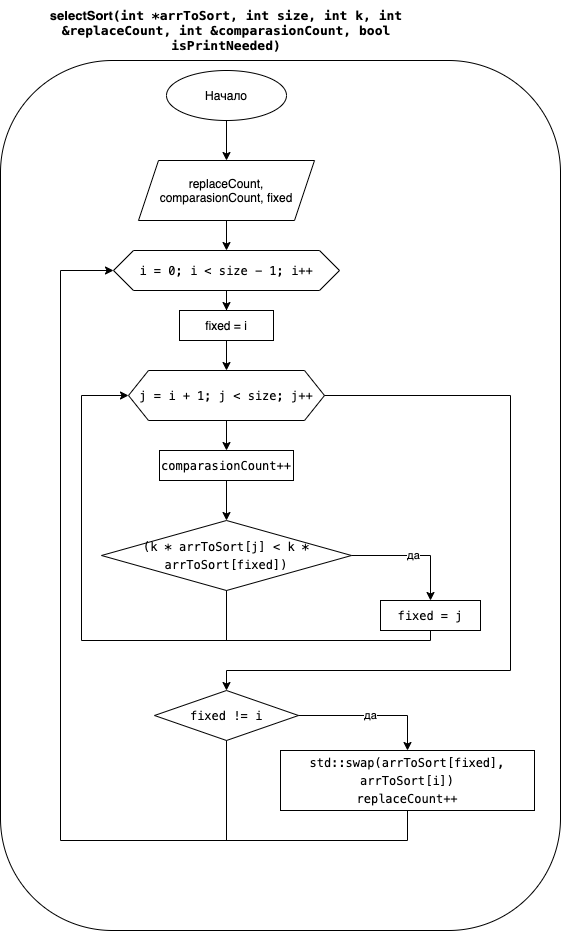
int k – коэффициент для выбора направления сортировки (1/-1)

int replaceCount – количество перестановок

int comparasionCount – количество сравнений

**Схема алгоритма**

****

****

**Текст программы**

**Листинг кода программы:**

**main.cpp**

#include "functions.h"

#include "template.h"

#define minTaskNumber 1

#define maxTaskNumber 2

using namespace std;

int main()

{

bool continueq = true;

int selectedTask = -1;

while (continueq)

{

enterTaskNumber(selectedTask, minTaskNumber, maxTaskNumber);

switch (selectedTask)

{

case 1:

task1(5);

break;

case 2:

task2();

break;

case -1:

{

exit(continueq);

break;

}

default:

break;

}

}

return 0;

}

**functions.h**

int iRandom(int a, int b);

void printArray(int \*arrToPrint, int size);

void bubbleSort(int \*arrToSort, int size, int k, int &replaceCount, int &comparasionCount, bool isPrintNeeded);

void selectSort(int \*arrToSort, int size, int k, int &replaceCount, int &comparasionCount, bool isPrintNeeded);

void generateArrays(int \*arrayToFill1, int \*arrayToFill2, int size);

void task1(int n);

void task2();

**functions.cpp**

#pragma once

#include <time.h> // time

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

#include <iostream>

#include <iomanip>

#include "functions.h"

int iRandom(int a, int b)

{

return a + rand() % (b - a);

}

void printArray(int \*arrToPrint, int size)

{

std::cout << "\n\t\t \_\_\_\_\_\_\_\_\_\n\t\t|";

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

{

std::cout << arrToPrint[i] << '|';

}

std::cout << "\b|\n\t\t ‾‾‾‾‾‾‾‾‾\n";

}

void bubbleSort(int \*arrToSort, int size, int k, int &replaceCount, int &comparasionCount, bool isPrintNeeded)

{

//1 - increase, -1 - decrease

replaceCount = 0;

comparasionCount = 0;

bool swapped;

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

{

swapped = false;

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

{

comparasionCount++;

if (arrToSort[j] \* k > arrToSort[j + 1] \* k)

{

replaceCount++;

swapped = true;

std::swap(arrToSort[j], arrToSort[j + 1]);

}

}

if (!swapped)

{

break;

}

}

if (isPrintNeeded)

{

std::cout << "sorted(" << size << ") with BubbleSort by ";

k == 1 ? std::cout << "increace\n" : std::cout << "decreace\n";

std::cout << "replaceCount = " << replaceCount << " comparsionCount = " << comparasionCount;

printArray(arrToSort, size);

}

}

void selectSort(int \*arrToSort, int size, int k, int &replaceCount, int &comparasionCount, bool isPrintNeeded)

{

replaceCount = 0;

comparasionCount = 0;

int fixed;

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

{

fixed = i;

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

{

comparasionCount++;

if (k \* arrToSort[j] < k \* arrToSort[fixed])

{

fixed = j;

}

}

if (fixed != i)

{

std::swap(arrToSort[fixed], arrToSort[i]);

replaceCount++;

}

}

if (isPrintNeeded)

{

std::cout << "sorted(" << size << ") with SelectSort by ";

k == 1 ? std::cout << "increace\n" : std::cout << "decreace\n";

std::cout << "replaceCount = " << replaceCount << " comparsionCount = " << comparasionCount ;

printArray(arrToSort, size);

}

}

void generateArrays(int \*arrayToFill1, int \*arrayToFill2, int size)

{

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

rand();

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

{

arrayToFill1[i] = iRandom(1, 9);

arrayToFill2[i] = arrayToFill1[i];

}

}

void task1(int n)

{

int \*a1 = new int[n];

int \*a2 = new int[n];

int replaceCount, comparsionCount;

int &rc = replaceCount;

int &cc = comparsionCount;

generateArrays(a1, a2, n);

std::cout << "Array before sort:\n";

printArray(a1, n);

std::cout << "\n\t\tSORTING DEFAULT RANDOM ARRAY:\n";

bubbleSort(a1, n, -1, rc, cc, 1);

selectSort(a2, n, -1, rc, cc, 1);

std::cout << "\n\t\tSORTING INCREASED ARRAY:\n";

bubbleSort(a1, n, -1, rc, cc, 1);

selectSort(a2, n, -1, rc, cc, 1);

std::cout << "\n\t\tSORTING DECREASED ARRAY:\n";

bubbleSort(a1, n, 1, rc, cc, 1);

selectSort(a2, n, 1, rc, cc, 1);

delete[] a1;

delete[] a2;

}

void task2()

{

int qwe;

std::cout << "\t\t\treplaces\tcomparions\n";

int replaceCount, comparsionCount;

int &rc = replaceCount;

int &cc = comparsionCount;

for (int i = 5; i < 501; i \*= 10)

{

int \*a1 = new int[i];

int \*a2 = new int[i];

generateArrays(a1, a2, i);

bubbleSort(a1, i, 1, rc, cc, 0);

std::cout << "buble sort: " << std::setw(5) << i << std::setw(12) << replaceCount << std::setw(16) << comparsionCount << std::endl;

selectSort(a2, i, 1, rc, cc, 0);

std::cout << "select sort: " << std::setw(5) << i << std::setw(12) << replaceCount << std::setw(16) << comparsionCount << std::endl;

delete[] a1;

delete[] a2;

std::cout << "------------------------------------------------------------\n";

}

}

**Анализ результатов**

