Федеральное государственное автономное

образовательное учреждение

высшего образования

«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт Космических и информационных технологий

институт

Кафедра «Информатика»

кафедра

**ОТЧЕТ О ПРАКТИЧЕСКОЙ РАБОТЕ**

Лабораторная работа №4. Усовершенствованные алгоритмы сортировки массивов

тема

Преподаватель \_\_\_\_\_\_\_\_\_ Р.Ю. Царёв

подпись, дата инициалы, фамилия

Студент КИ18-17/1б 031830504 \_\_\_\_\_\_\_\_\_ Е.В. Железкин

номер группы, зачетной книжки подпись, дата инициалы, фамилия

Красноярск 2020

Содержание

[Содержание 2](#_Toc33628970)

[1 Цель работы 3](#_Toc33628971)

[2 Задача работы 3](#_Toc33628972)

[3 Ход работы 3](#_Toc33628973)

[4 Вывод 8](#_Toc33628974)

# Цель работы

Изучение улучшенных алгоритмов сортировки массива.

# Задача работы

Реализовать в программе два алгоритма (по выбору студента) из указанных ниже:

а) шейкерная сортировка,

б) сортировка Шелла,

в) быстрая сортировка.

Сравнить эффективность реализованных алгоритмов.

# Ход работы

Листинг 1 – содержание файла main.cpp проекта Lab\_4:

#include **<iostream>**#include **"inputFunc.h"**#include **"sortAlgs.h"  
  
using namespace** std;  
  
**int** main() {  
  
 **int** size = -322;  
 string str;  
  
 **while**(size == -322){  
 cout << **"Введите размер массива: \n"**;  
 getline(cin, str);  
 size = Checker(str);  
 }  
  
 **int** temp;  
 **int** \*arrayForShaker = **new int**[size];  
 **int** \*arrayForFast = **new int**[size];  
  
 **for** (**int** i = 0; i < size; i++){  
  
 temp = -322;  
  
 **while**(temp == -322){  
 cout << **"Введите "** << i + 1 << **" элемент массива: "**;  
 getline(cin, str);  
 temp = Checker(str);  
 **if** (temp == -322) cout << **"Некорректный ввод, повторите попытку!\n"**;  
 }  
  
 arrayForShaker[i] = temp;  
 arrayForFast[i] = temp;  
  
 }  
  
 **struct** Returner tempStruct{};  
  
 tempStruct = ShakerSort(arrayForShaker, size);  
  
 cout << **"\nОтсортированный массив:\n"**;  
 **for** (**int** i = 0; i < size; i++){  
 cout << arrayForShaker[i] << **" "**;  
 }  
  
 cout << **"\n\nКоличество перестановок и сравнений алгоритма "  
 "сортировки с помощью прямого обмена(пузырькового) соответственно:"  
 "\n"** << tempStruct.countOfSwaps << **" "** << tempStruct.countOfComparisons << endl << endl;  
  
 tempStruct = FastSort(arrayForFast, 0, size - 1);  
  
 cout << **"\nКоличество перестановок и сравнений алгоритма "  
 "сортировки с помощью прямого включения соответственно:"  
 "\n"** << tempStruct.countOfSwaps << **" "** << tempStruct.countOfComparisons << endl;  
  
 **delete**[] arrayForShaker;  
 **delete**[] arrayForFast;  
  
 **return** 0;  
}

Листинг 2 – содержание файла inputFunc.cpp проекта Lab\_4:

#include **"inputFunc.h"  
  
  
int** Checker(string tempForCheck)  
{  
 **int** number = 0;  
 **int** flag = 0;  
  
 **for** (**unsigned int** i = 0; i < tempForCheck.length(); i++)  
 {  
 **if** (i == 0 && tempForCheck[0] == **'-'**)  
 {  
 flag++;  
 **continue**;  
 }  
  
 **if** (tempForCheck[i] == **'0'** || tempForCheck[i] == **'1'** || tempForCheck[i] == **'2'** || tempForCheck[i] == **'3'** ||  
 tempForCheck[i] == **'4'** || tempForCheck[i] == **'5'** || tempForCheck[i] == **'6'** ||  
 tempForCheck[i] == **'7'** || tempForCheck[i] == **'8'** || tempForCheck[i] == **'9'**) number = number \* **ORDER** +  
 (tempForCheck[i] - **ASCII\_CONST**);  
 **else  
 return NUMBER\_ERROR**;  
 }  
  
 **if** (flag > 0) number \*= **NEGATIVE**;  
  
 **return** number;  
}

Листинг 3 – содержание файла inputFunc.h проекта Lab\_4:

#ifndef **LAB\_3\_INPUTFUNC\_H**#define **LAB\_3\_INPUTFUNC\_H**#include **<iostream>**#define **NUMBER\_ERROR** -322  
  
#define **ORDER** 10  
  
#define **NEGATIVE** -1  
  
#define **ASCII\_CONST** 48  
  
**using namespace** std;  
  
**int** Checker(string tempForCheck);  
  
#endif *//LAB\_3\_INPUTFUNC\_H*

Листинг 4 – содержание файла sortAlgs.cpp проекта Lab\_4:

#include **<algorithm>**#include **<iostream>***//#include <ctime>*#include **"sortAlgs.h"**Returner ShakerSort(**int**\* tempArray, **int** size){  
  
 **int** countOfSwaps = 0;  
 **int** countOfComparisons = 0;  
 **int** left = 0;  
 **int** right = size - 1;  
 **int** flag = 0;  
  
 **for** (**int** i = 0; i < size; i++) {  
  
 flag = 0;  
  
 countOfComparisons++;  
 **if** (i % 2 == 0){  
 **for** (**int** j = right; j > left; j--) {  
 **if** (tempArray[j] < tempArray[j - 1]) {  
 std::swap(tempArray[j], tempArray[j - 1]);  
 flag++;  
 countOfSwaps++;  
 }  
 countOfComparisons++;  
 }  
 left++;  
 } **else**{  
 **for** (**int** j = left; j < right; j++) {  
 **if** (tempArray[j] > tempArray[j + 1]) {  
 std::swap(tempArray[j], tempArray[j + 1]);  
 flag++;  
 countOfSwaps++;  
 }  
 countOfComparisons++;  
 }  
 right--;  
 }  
  
 countOfComparisons++;  
 **if** (flag == 0) **break**;  
 }  
  
 **struct** Returner ret{countOfComparisons, countOfSwaps};  
  
 ret.countOfComparisons = countOfComparisons;  
 ret.countOfSwaps = countOfSwaps;  
  
 **return** ret;  
}  
  
Returner FastSort(**int**\* tempArray, **int** left, **int** right){  
  
 **if** (left < right) {  
  
 **struct** Returner tempRet = partition(tempArray, left, right);  
 std::cout << left << **" "** << right << **": "**;  
 **for** (**int** k = left; k <= right; k++){  
 std::cout << tempArray[k] << **" "**;  
 }  
 std::cout << std::endl;  
 **int** p = tempRet.pos;  
  
 **struct** Returner ret = tempRet;  
  
 tempRet = FastSort(tempArray, left, p);  
 ret.countOfSwaps += tempRet.countOfSwaps;  
 ret.countOfComparisons += tempRet.countOfComparisons;  
  
 tempRet = FastSort(tempArray, p + 1, right);  
 ret.countOfSwaps += tempRet.countOfSwaps;  
 ret.countOfComparisons += tempRet.countOfComparisons;  
  
 **return** ret;  
 }  
  
 **struct** Returner ret{0,0,0};  
 **return** ret;  
  
}  
  
**struct** Returner partition(**int**\* tempArray, **int** left, **int** right){  
  
 **int** countOfSwaps = 0;  
 **int** countOfComparisons = 0;  
  
 srand(time(0));  
 **int** temp = tempArray[(left + right) / 2];  
  
 **int** i = left;  
 **int** j = right;  
  
 **while** (i <= j) {  
  
 countOfComparisons++;  
 **while** (tempArray[i] < temp) {  
 i++;  
 countOfComparisons++;  
 }  
 countOfComparisons++;  
 **while** (tempArray[j] > temp) {  
 j--;  
 countOfComparisons++;  
 }  
 countOfComparisons++;  
 **if** (i >= j) {  
 **break**;  
 }  
 std::swap(tempArray[i++], tempArray[j--]);  
 countOfSwaps++;  
 countOfComparisons++;  
 }  
  
 **struct** Returner ret{countOfSwaps, countOfComparisons, j};  
  
 **return** ret;  
  
}

Листинг 5 – содержание файла sortAlgs.h проекта Lab\_4:

#ifndef **LAB\_3\_SORTALGS\_H**#define **LAB\_3\_SORTALGS\_H  
  
struct** Returner{  
 **int** countOfSwaps;  
 **int** countOfComparisons;  
 **int** pos;  
};  
  
Returner ShakerSort(**int**\*, **int**);  
Returner FastSort(**int**\*, **int**, **int**);  
**struct** Returner partition(**int**\*, **int**, **int**);  
  
#endif *//LAB\_3\_SORTALGS\_H*

1. Примеры работы:

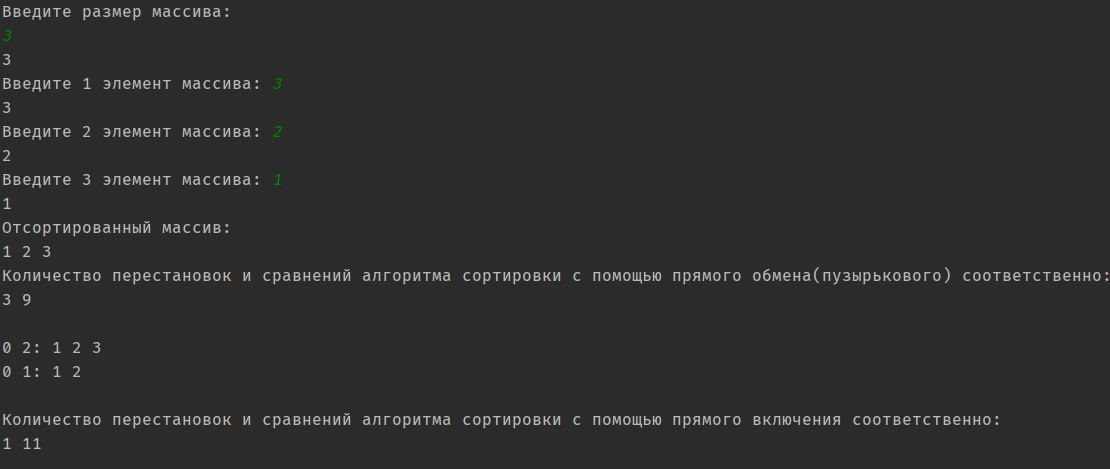


Рисунок 1 – результаты сортировок массива

# Вывод

В ходе данной лабораторной работы были реализованы улучшенные алгоритмы сортировки значений массива, описанные в задании и лекции.