МИНИСТЕРСТВО науки и высшего ОБРАЗОВАНИЯ РОссИЙСКОЙ ФЕДЕРАЦИИФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ

УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«МОСКОВСКИЙ АВИАЦИОННЫЙ ИНСТИТУТ

(национальный исследовательский университет)»

Институт №3 «Системы управления, информатика и электроэнергетика»

Кафедра № 304 «Вычислительные машины, системы и сети»

Информатика

Отчет по лабораторной работе №3

Выполнила бригада группы М3О-211Б-21

Багиров Э. Р.

Нуриев Н. Н.

Оглавление

[Задание 3](#_Toc118718958)

[Структурные схемы алгоритмов 4](#_Toc118718959)

[Код программы 7](#_Toc118718960)

[Сравнительные таблицы 11](#_Toc118718961)

[Вывод 13](#_Toc118718962)

# Задание

Для массива из ***n*** элементов выполнить сортировку с помощью двух указанных методов для:

1. заданной произвольным образом последовательности чисел,
2. уже отсортированных последовательностей в возрастающем и убывающем порядке (лучший и худший случаи для выполнения сортировки).

Этапы выполнения ЛР:

1. Промежуточные результаты сортировки представить по каждой итерации для массива размерностью ***n*** = 15 (необходимо выводить на печать весь массив на каждом промежуточном этапе сортировки). Сравнить число **необходимых** **сравнений и число пересылок**.
2. Выполнить сортировку массивов размерности ***n*** = 1000 ( 10000, 100000, 500000 ). Сравнить **время выполнения алгоритмов, число необходимых сравнений и число пересылок**.

# Структурные схемы алгоритмов

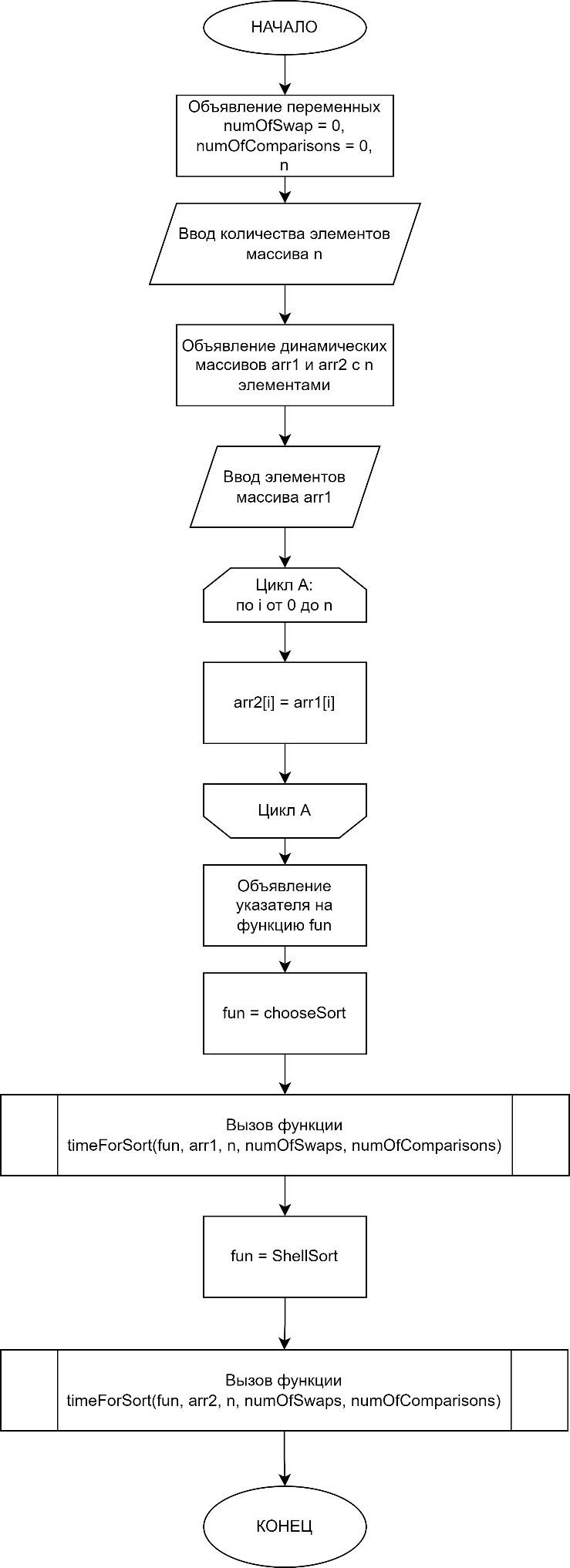


Рисунок 1. Структурная схема главной (main) функции

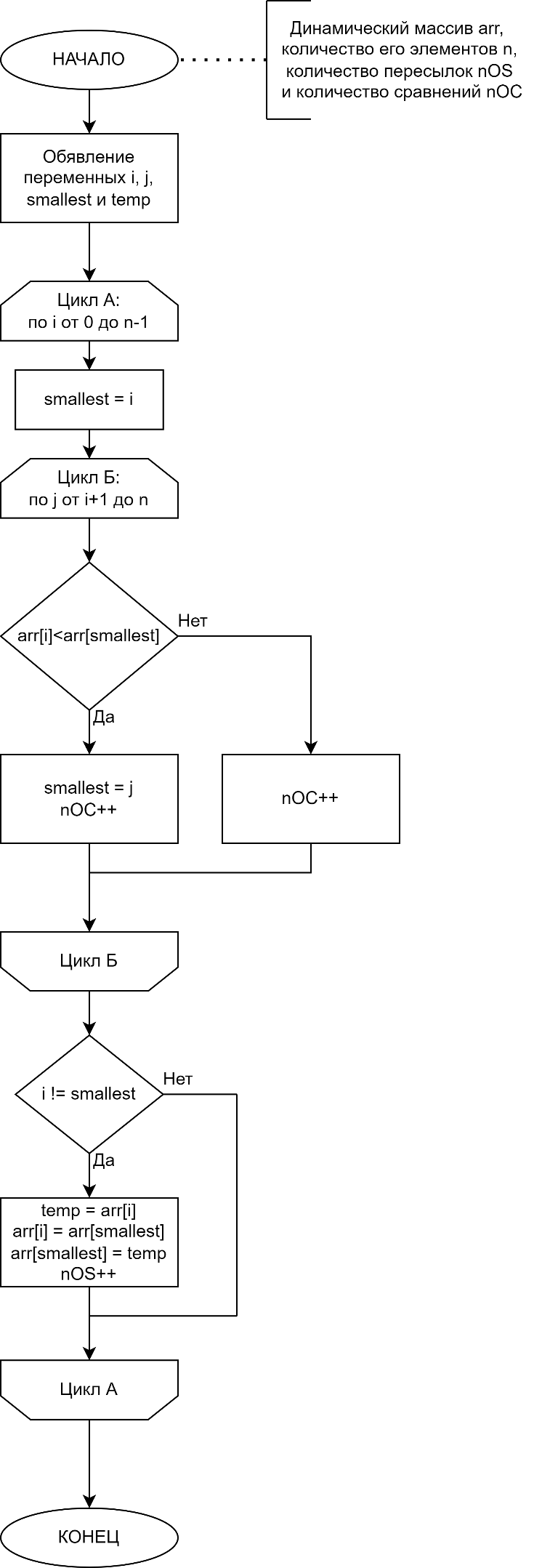


Рисунок 2. Структурная схема алгоритма сортировки выбором (chooseSort)

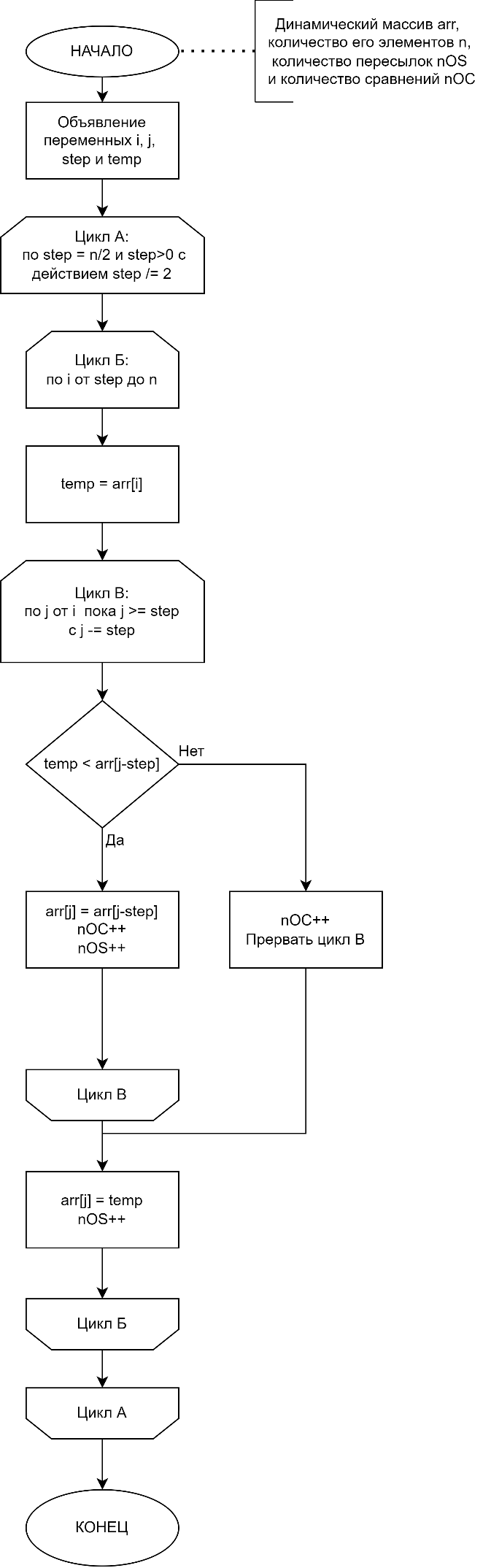


Рисунок 3. Структурная схема алгоритма сортировки Шелла (ShellSort)

# Код программы

#include <iostream>

#include <chrono>

using namespace std;

char red[] = { 0x1b,'[','0',';','3','1','m',0 };

char normal[] = { 0x1b,'[','0',';','3','9','m',0 };

void generateIncreasing(int\* arr, int n) {

int step = rand();

arr[0] = step;

for (int i = 1; i < n; i++) arr[i] = arr[i - 1] + step;

}

void generateDecreasing(int\* arr, int n) {

int step = rand();

arr[0] = step;

for (int i = 1; i < n; i++) arr[i] = arr[i - 1] - step;

}

void generateRandom(int\* arr, int n){

for (int i = 0; i < n; i++) arr[i] = rand();

}

void printArrayWithColor(int\* arr, int n, int k, int j) {

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

if ((i == k || i == j) && k != j) {

cout << red << arr[i] << "\t";

}

else {

cout << normal << arr[i] << "\t";

}

}

cout << endl;

}

void chooseSort(int\* arr, int n, int numOfSwaps, int numOfComparisons) {

int i, j, smallest, temp;

int choose;

numOfSwaps = 0;

numOfComparisons = 0;

cout << "Выводить сортировку построчно?(по умолчанию Нет)\n1.Да\n2.Нет\nВаш выбор: ";

cin >> choose;

switch (choose) {

case 1:

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

smallest = i;

for (j = i + 1; j < n; j++)

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

smallest = j;

numOfComparisons++;

}

else numOfComparisons++;

if (i != smallest) {

temp = arr[i];

arr[i] = arr[smallest];

arr[smallest] = temp;

numOfSwaps++;

}

printArrayWithColor(arr, n, smallest, i);

}

break;

default:

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

smallest = i;

for (j = i + 1; j < n; j++)

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

smallest = j;

numOfComparisons++;

}

else numOfComparisons++;

if (i != smallest) {

temp = arr[i];

arr[i] = arr[smallest];

arr[smallest] = temp;

numOfSwaps++;

}

}

break;

}

cout << normal << "\nКоличество сравнений: " << numOfComparisons

<< "\nКоличество пересылок: " << numOfSwaps << "\n";

}

void ShellSort(int\* arr, int n, int numOfSwaps, int numOfComparisons) {

int i, j, step, temp;

int choose;

numOfComparisons = 0;

numOfSwaps = 0;

cout << "Выводить сортировку построчно?(по умолчанию Нет)\n1.Да\n2.Нет\nВаш выбор: ";

cin >> choose;

switch (choose) {

case 1:

for (step = n / 2; step > 0; step /= 2)

for (i = step; i < n; i++) {

temp = arr[i];

for (j = i; j >= step; j -= step) {

if (temp < arr[j - step]) {

arr[j] = arr[j - step];

numOfComparisons++;

}

else {

numOfComparisons++;

break;

}

}

arr[j] = temp;

numOfSwaps++;

printArrayWithColor(arr, n, i, j);

}

break;

default:

for (step = n / 2; step > 0; step /= 2)

for (i = step; i < n; i++) {

temp = arr[i];

for (j = i; j >= step; j -= step) {

if (temp < arr[j - step]) {

arr[j] = arr[j - step];

numOfComparisons++;

}

else {

numOfComparisons++;

break;

}

}

arr[j] = temp;

numOfSwaps++;

}

break;

}

cout << normal << "\nКоличество сравнений: " << numOfComparisons

<< "\nКоличество пересылок: " << numOfSwaps << "\n";

}

void timeForSort(void (\*function)(int\* arr, int n, int numOfSwaps, int numOfComparisons), int\* arr, int n, int numOfSwaps, int numOfComparisons) {

auto startTime = chrono::steady\_clock::now();

function(arr, n, numOfSwaps, numOfComparisons);

auto endTime = chrono::steady\_clock::now();

auto time = chrono::duration\_cast<chrono::microseconds>(endTime - startTime);

cout << "\nВремя на сортировку (в мск): " << time.count() << "\n\n";

cout << "Вывести отсортированный массив?(по умолчанию Нет)\n1.Да\n2.Нет\nВаш выбор: ";

int choose4;

cin >> choose4;

switch (choose4) {

case 1:

cout << normal << "\nОтсортированный массив:\n";

for (int i = 0; i < n; i++) cout << arr[i] << "\t";

break;

default:

break;

}

}

int main()

{

setlocale(LC\_ALL, "rus");

int numOfSwaps = 0, numOfComparisons = 0;

int n;

cout << "Введите количество элементов массива: ";

cin >> n;

int\* arr1 = new int[n];

int\* arr2 = new int[n];

int choose;

cout << "Задать массив от руки?(по умолчанию Нет)\n1.Да\n2.Нет\nВаш выбор: ";

cin >> choose;

cout << endl;

switch (choose) {

case 1:

for (int i = 0; i < n; i++) cin >> arr1[i];

for (int i = 0; i < n; i++) arr2[i] = arr1[i];

break;

default:

int choose2;

cout << "Какой массив вы хотите сгенерировать?\n1.Возрастающий\n2.Убывающий\n3.Рандомный\nВаш выбор: ";

cin >> choose2;

switch (choose2) {

case 1:

generateIncreasing(arr1, n);

for (int i = 0; i < n; i++) arr2[i] = arr1[i];

break;

case 2:

generateDecreasing(arr1, n);

for (int i = 0; i < n; i++) arr2[i] = arr1[i];

break;

case 3:

generateRandom(arr1, n);

for (int i = 0; i < n; i++) arr2[i] = arr1[i];

break;

default:

exit(-1);

}

break;

}

system("cls");

int choose3;

cout << "Вывести исходный массив?(по умолчанию Нет)\n1.Да\n2.Нет\nВаш выбор: ";

cin >> choose3;

switch (choose3) {

case 1:

cout << "Введенный массив:\n";

for (int i = 0; i < n; i++) cout << arr1[i] << "\t";

break;

default:

break;

}

cout << "\n\n\t\tСортировка выбором:\n\n";

void (\*fun)(int\* arr, int n, int numOfSwaps, int numOfComparisons);

fun = chooseSort;

timeForSort(fun, arr1, n, numOfSwaps, numOfComparisons);

cout << "\n\n\t\tСортировка Шелла:\n\n";

fun = ShellSort;

timeForSort(fun, arr2, n, numOfSwaps, numOfComparisons);

}

# Сравнительные таблицы

|  |  |  |  |
| --- | --- | --- | --- |
| Количество элементов | Время выполнения алгоритма (в мкс) | Число необходимых сравнений | Число пересылок |
| 1000 | 326454 | 499500 | 500 |
| 10000 | 808873 | 49995000 | 5000 |
| 100000 | 9528128 | 704982704 | 50000 |
| 500000 | 241852677 | 445698416 | 250000 |

Таблица 1. Сортировка выбором для убывающего массива

|  |  |  |  |
| --- | --- | --- | --- |
| Количество элементов | Время выполнения алгоритма (в мкс) | Число необходимых сравнений | Число пересылок |
| 1000 | 353411 | 11716 | 8006 |
| 10000 | 915698 | 172578 | 120005 |
| 100000 | 1364974 | 2244585 | 1500006 |
| 500000 | 1192072 | 12428778 | 8500007 |

Таблица 2. Сортировка Шелла для убывающего массива

|  |  |  |  |
| --- | --- | --- | --- |
| Количество элементов | Время выполнения алгоритма (в мкс) | Число необходимых сравнений | Число пересылок |
| 1000 | 496652 | 499500 | 0 |
| 10000 | 393974 | 49995000 | 0 |
| 100000 | 10391322 | 704982704 | 0 |
| 500000 | 234703559 | 445698416 | 0 |

Таблица 3. Сортировка выбором для возрастающего массива

|  |  |  |  |
| --- | --- | --- | --- |
| Количество элементов | Время выполнения алгоритма (в мкс) | Число необходимых сравнений | Число пересылок |
| 1000 | 1327273 | 8006 | 0 |
| 10000 | 728995 | 120005 | 0 |
| 100000 | 1123072 | 1500006 | 0 |
| 500000 | 819025 | 8500007 | 0 |

Таблица 4. Сортировка Шелла для возрастающего массива

# Вывод

Были изучены два алгоритма сортировки: сортировка Шелла и сортировка выбором. Написана программа для осуществления сортировки одного и того же массива целых чисел и составлены сравнительные таблицы по результатам сопоставительного анализа. Было проведено сравнение времени, количества перестановок и количества сравнений для массивов, упорядоченных по возрастанию и по убыванию. По результатам анализа можно сделать вывод, что для малого количества элементов массива лучше использовать сортировку выбором, а для большего количества – сортировку Шелла. Для вычисления времени работы в мкс использовалась библиотека chrono.