МИНИСТЕРСТВО науки и высшего ОБРАЗОВАНИЯ РОссИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ 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»

Структуры и алгоритмы обработки данных

Отчет по лабораторной работе № 3

«Алгоритмы поиска»

Выполнили студенты группы М3О-210Б-20  
Гудынин Д. Д.

Артёмов И. Е.

Проверили Дмитриева Е.А.  
Офицеров В.П.

**Оглавление.**

[**Цель работы.** 3](#_Toc87086090)

[**Блок-схемы функций алгоритма поиска** 4](#_Toc87086091)

[**Листинг программы.** 7](#_Toc87086092)

[**Работа программы.** 10](#_Toc87086093)

[**1 этап работы** 11](#_Toc87086094)

[**2 этап работы** 12](#_Toc87086095)

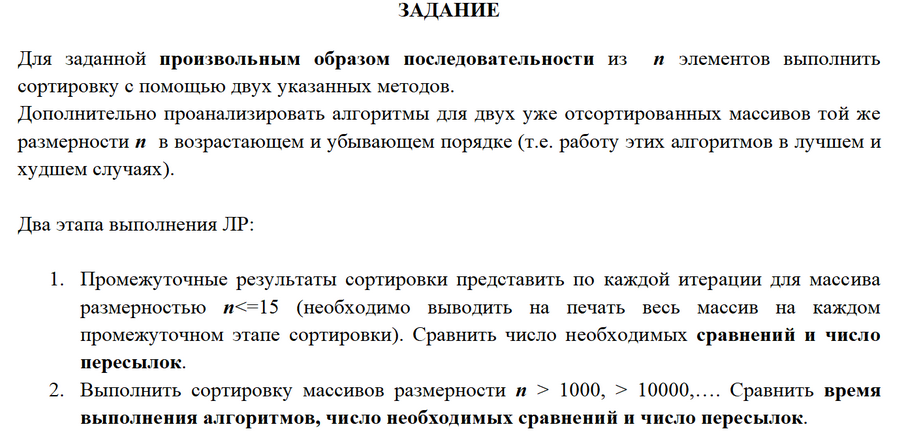
[**Вывод.** 13](#_Toc87086096)

# **Цель работы.**

Исследование и сравнение алгоритмов сортировки:

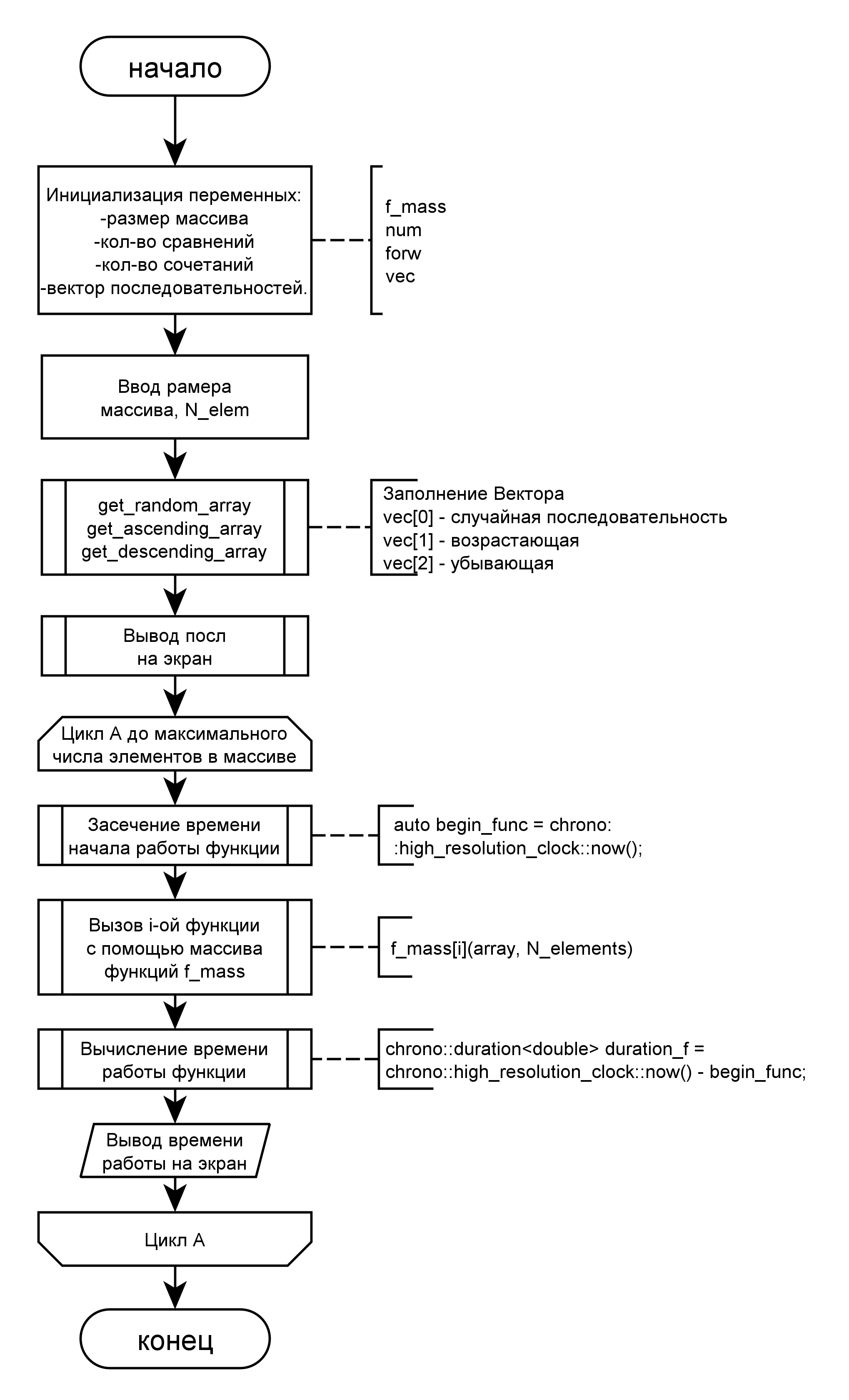
· шейкерная сортировка,

· сортировка Шелла,

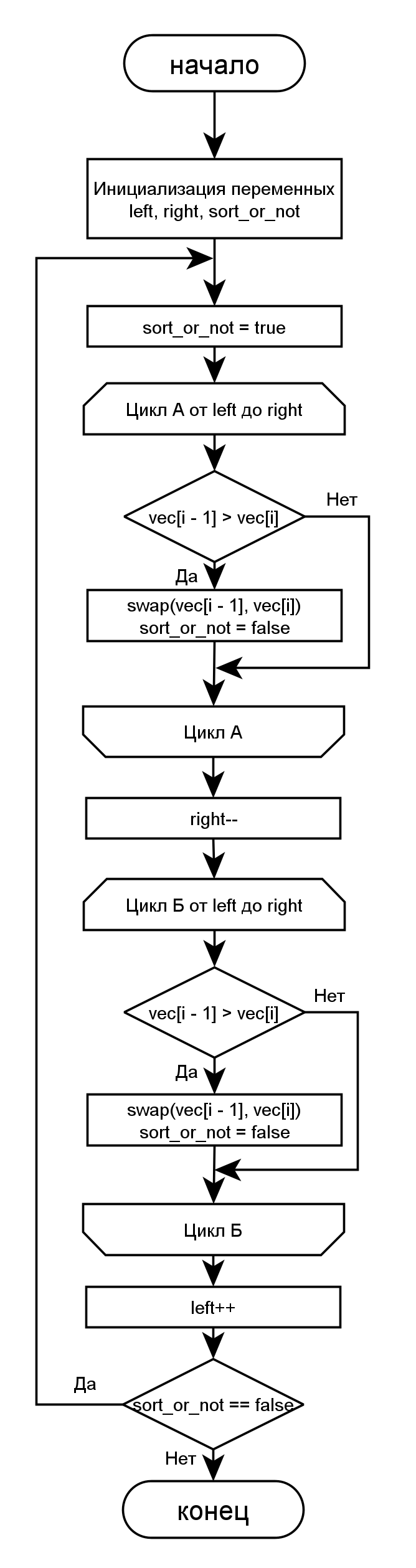


# **Блок-схемы функций алгоритма поиска**

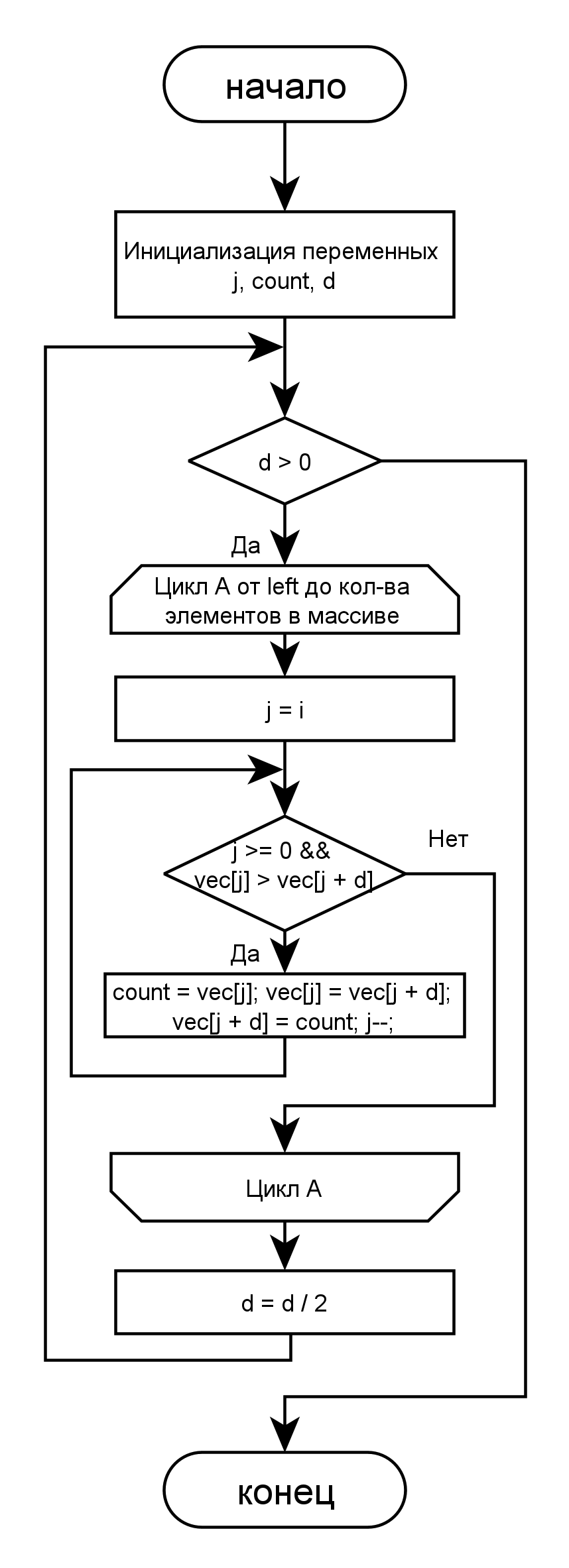
1. Блок-схема main ()



1. Блок-схема Shake\_sort ()



1. Блок-схема Shell\_sort ()

****

# **Листинг программы.**

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Курс ПРОГРАММИРОВАНИЕ \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*Project type :Win64 Console Application \*

\*Project name :2k-3c-ЛР2-Алгоритмы сортировки \*

\*File name :2k-3c-ЛР2-Алгоритмы сортировки \*

\*Language :CPP, MSVS 2022 \*

\*Programmers :Артемов Игорь Евгеньевич, Гудынин Данила Денисович М3О-210Б-20 \*

\*Modified By : \*

\*Created :23.09.2021 \*

\*Last revision:25.09.2021 \*

\*Comment :Алгоритмы сортировки \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#include <iostream>

#include <string>

#include <vector>

#include <chrono>

#include <fstream>

using namespace std;

void print\_vec(vector<vector<int>> vec) {

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

for (int j = 0; j < vec[i].size(); j++) {

cout << vec[i][j] << "\t";

}

cout << "\n";

}

}

void write\_txt(vector<int> vec, string FNAME, bool clear\_file) {

ofstream my\_file;

if (clear\_file == true) {

my\_file.open(FNAME, ios\_base::trunc);

}

else {

my\_file.open(FNAME, ios\_base::app);

}

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

my\_file << vec[i] << "\t";

}

my\_file << "\n";

my\_file.close();

}

//Генерация рандомных значений от MIN до MAX

vector<int> get\_random\_array(int size) {

vector<int> vec(size);

srand(time(NULL));

cout << \_\_func\_\_ << endl;

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

vec[i] = rand();

}

return vec;

}

//Генерация возрастающей последовательности от MIN до MAX

vector<int> get\_ascending\_array(int size) {

cout << \_\_func\_\_ << endl;

vector<int> vec(size);

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

vec[i] = i;

}

return vec;

}

//Генерация убывающей последовательности от MIN до MAX

vector<int> get\_descending\_array(int size) {

cout << \_\_func\_\_ << endl;

vector<int> vec(size);

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

vec[i] = size - 1 - i;

}

return vec;

}

void Shake\_sort(vector<int> vec, int &num, int &forw, int n\_sort) { // Шейкерная сортировка

//write\_txt(vec, "Shake\_sort\_" + to\_string(n\_sort) + ".txt", true);

bool sort\_or\_not = true;

int right = vec.size() - 1; // n - размер массива

int left = 1;

do {

bool sort\_or\_not = true;

for (int i = left; i <= right; i++) {

num++;

if (vec[i - 1] > vec[i]) {

swap(vec[i - 1], vec[i]);

sort\_or\_not = false;

forw++;

}

}

right--;

for (int i = right; i >= left; i--) {

num++;

if (vec[i] < vec[i - 1]) {

swap(vec[i], vec[i - 1]);

sort\_or\_not = false;

forw++;

}

}

left++;

} while (sort\_or\_not == false);

}

void Shell\_sort(vector<int> vec, int& num, int& forw, int n\_sort) { //сортировка Шелла

int j, count;

int d = vec.size() / 2;

while (d > 0)

{

for (int i = 0; i < vec.size() - d; i++)

{

num++;

j = i;

while (j >= 0 && vec[j] > vec[j + d])

{

forw++;

count = vec[j];

vec[j] = vec[j + d];

vec[j + d] = count;

j--;

}

}

d = d / 2;

}

}

int main() {

void (\*f\_mass[2]) (vector<int>, int &, int &, int) = {Shake\_sort, Shell\_sort};

vector<int>(\*array\_generator[3]) (int) = {get\_random\_array, get\_ascending\_array, get\_descending\_array};

vector<string> names\_array = { "случайной", "возрастающей", "убывающей" };

vector<string> f\_names = { "шейкерной cортировки", "сортировки Шелла" };

setlocale(LC\_ALL, "ru");

int N\_elem, num=0, forw=0;

cout << "Введите количество элементов в векторе: ";

cin >> N\_elem;

vector<vector<int>> vec(3, vector<int>(N\_elem));

vec[0] = get\_random\_array(N\_elem);

vec[1] = get\_ascending\_array(N\_elem);

vec[2] = get\_descending\_array(N\_elem);

//cout << "\nИсходные последовательности:\n";

print\_vec(vec);

for (int k = 0; k < names\_array.size(); k++) {

cout << "\n\n";

cout << "------------Сортировка " << names\_array[k] << " последовательности------------";

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

auto begin\_func = chrono::high\_resolution\_clock::now();

f\_mass[i](vec[k], num, forw, k);

chrono::duration<double> duration\_f = chrono::high\_resolution\_clock::now() - begin\_func;

cout << "\n\033[32mФункция " << f\_names[i] << " отработала за " << duration\_f.count() \* 1000 << " МЛсек!\033[0m";

cout << "\n\tКол-во cравнений: " << num;

cout << "\tКол-во пересылок: " << forw;

}

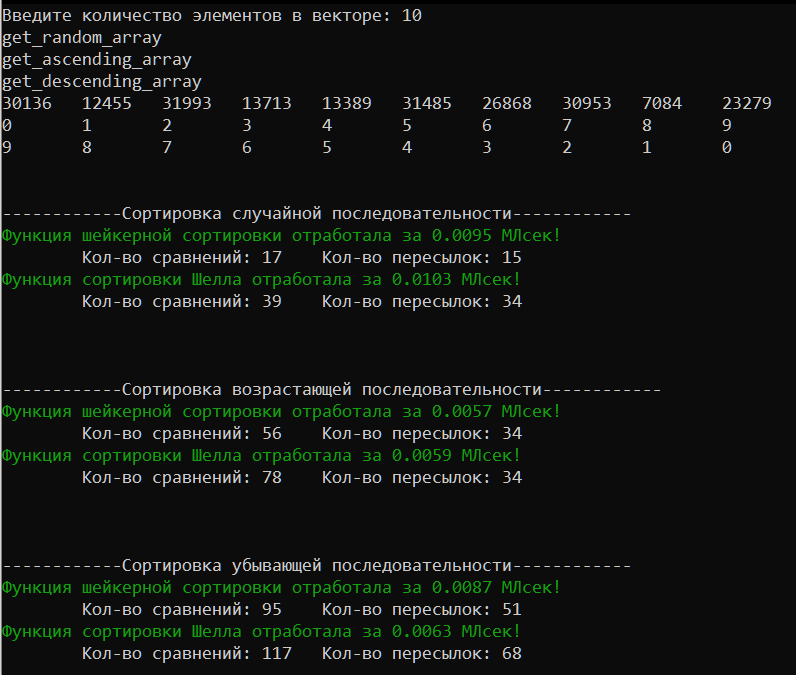
cout << "\n\n";

}

return 0;

}

# **Работа программы.**



# **1 этап работы**

**Количество элементов** - 15

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | Массив со случайными числами | | Отсортированный массив по возрастанию | | Отсортированный массив по  убыванию | |
|  | Сравнения | Пересылки | Сравнения | Пересылки | Сравнения | Пересылки |
| Шейкерная сортировка | 27 | 25 | 88 | 48 | 15 | 75 |
| Сортировка Шелла | 61 | 48 | 122 | 48 | 183 | 108 |

# **2 этап работы**

**Количество элементов** - 1000  
**Сортировка по возрастанию**  
**Время сортировки** (в млС)

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | Массив со случайными числами | | | Отсортированный массив по возрастанию | | | Отсортированный массив по  убыванию | | |
|  | Время | Сравнения | Пересылки | Время | Сравнения | Пересылки | Время | Сравнения | Пересылки |
| Шейкерная сортировка | 0.2402 | 1997 | 1984 | 0.0556 | 12000 | 0 | 0.2408 | 22003 | 47565 |
| Сортировка Шелла | 3.4542 | 10003 | 45568 | 0.2817 | 20006 | 0 | 0.6585 | 30009 | 52697 |

**Количество элементов** - 10000  
**Сортировка по возрастанию**  
**Время сортировки** (в млС)

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | Массив со случайными числами | | | Отсортированный массив по возрастанию | | | Отсортированный массив по  убыванию | | |
|  | Время | Сравнения | Пересылки | Время | Сравнения | Пересылки | Время | Сравнения | Пересылки |
| Шейкерная сортировка | 2.36 | 19997 | 19985 | 0.5166 | 159999 | 0 | 2.348 | 300001 | 3965384 |
| Сортировка Шелла | 293.505 | 140002 | 3945387 | 4.2594 | 280004 | 0 | 9.1631 | 420006 | 4031176 |

**Количество элементов** - 50000  
**Сортировка по возрастанию**  
**Время сортировки** (в млС)

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | Массив со случайными числами | | | Отсортированный массив по возрастанию | | | Отсортированный массив по  убыванию | | |
|  | Время | Сравнения | Пересылки | Время | Сравнения | Пересылки | Время | Сравнения | Пересылки |
| Шейкерная сортировка | 12.4716 | 99997 | 99977 | 2.708 | 900000 | 0 | 12.3407 | 1700003 | 92913468 |
| Сортировка Шелла | 7091.16 | 800003 | 92813471 | 27.0832 | 1600006 | 0 | 55.3672 | 1249975000 | 2400009 |

**Вывод.**

Разработка программы успешно завершена, т.к.:

1. Полученные результаты совпадают с ожидаемыми
2. Изучены основные принципы работы алгоритмов сортировок
3. В 1 этапе работы на малой размерности массива, заданным случайным образом, были протестированы алгоритмы сортировок.   
   Сортировка Шелла имеет худший случай O(n\* n), если неудачны выбраны промежутки разбиения массива.   
   Примерная сложность шейкерной сортировки O().
4. В 2 этапе работы на большом массиве данных были протестированы алгоритмы сортировок.
5. Итак, в конечном счете можно сказать следующее: при небольшой размерности массива эффективно использовать шейкерную сортировку, так как её сложность в худшем случае намного меньше.