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 №1

Выполнила бригада группы М3О-211Б-21

Багиров Э. Р.

Нуриев Н. Н.

Оглавление

[Структурные схемы алгоритмов 3](#_Toc117536789)

[Главная (основная – main) функция 3](#_Toc117536790)

[Функция создания синусоидного массива 4](#_Toc117536791)

[Код программы 5](#_Toc117536792)

[Графики полученных массивов (для 100 элементов) 10](#_Toc117536793)

[Графики зависимости времени (в мс) от размерности 13](#_Toc117536794)

[Для массивов из float-элементов 13](#_Toc117536795)

[Для массивов с int-элементами 15](#_Toc117536796)

[Вывод 18](#_Toc117536797)

# Структурные схемы алгоритмов

## Главная (основная – main) функция

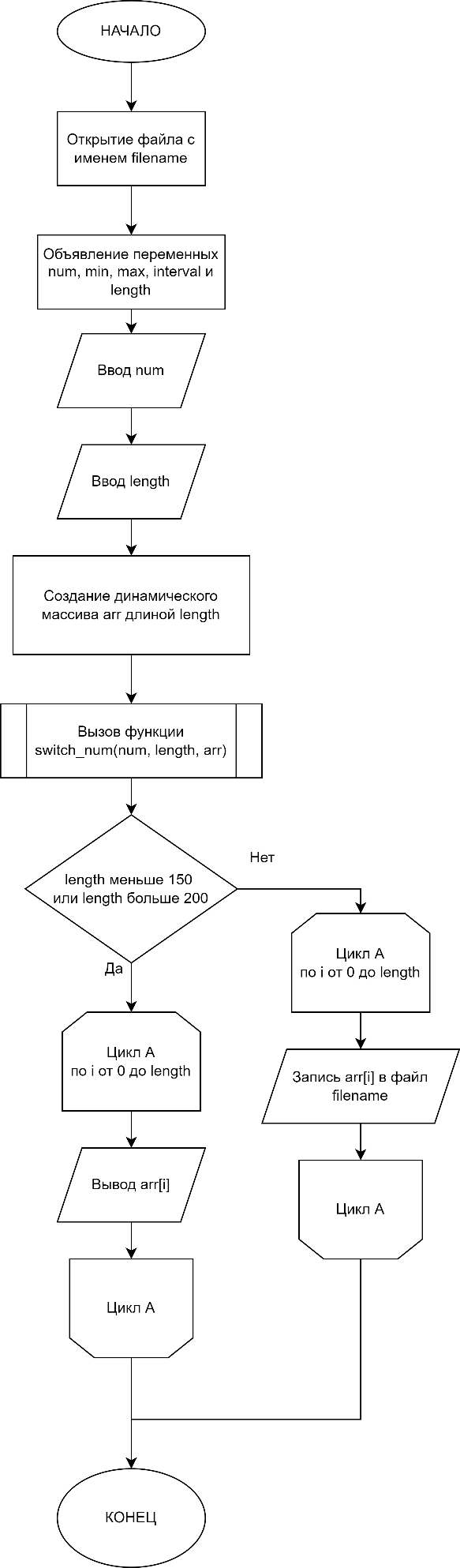


Рисунок 1. Структурная схема алгоритма функции main()

## Функция создания синусоидного массива

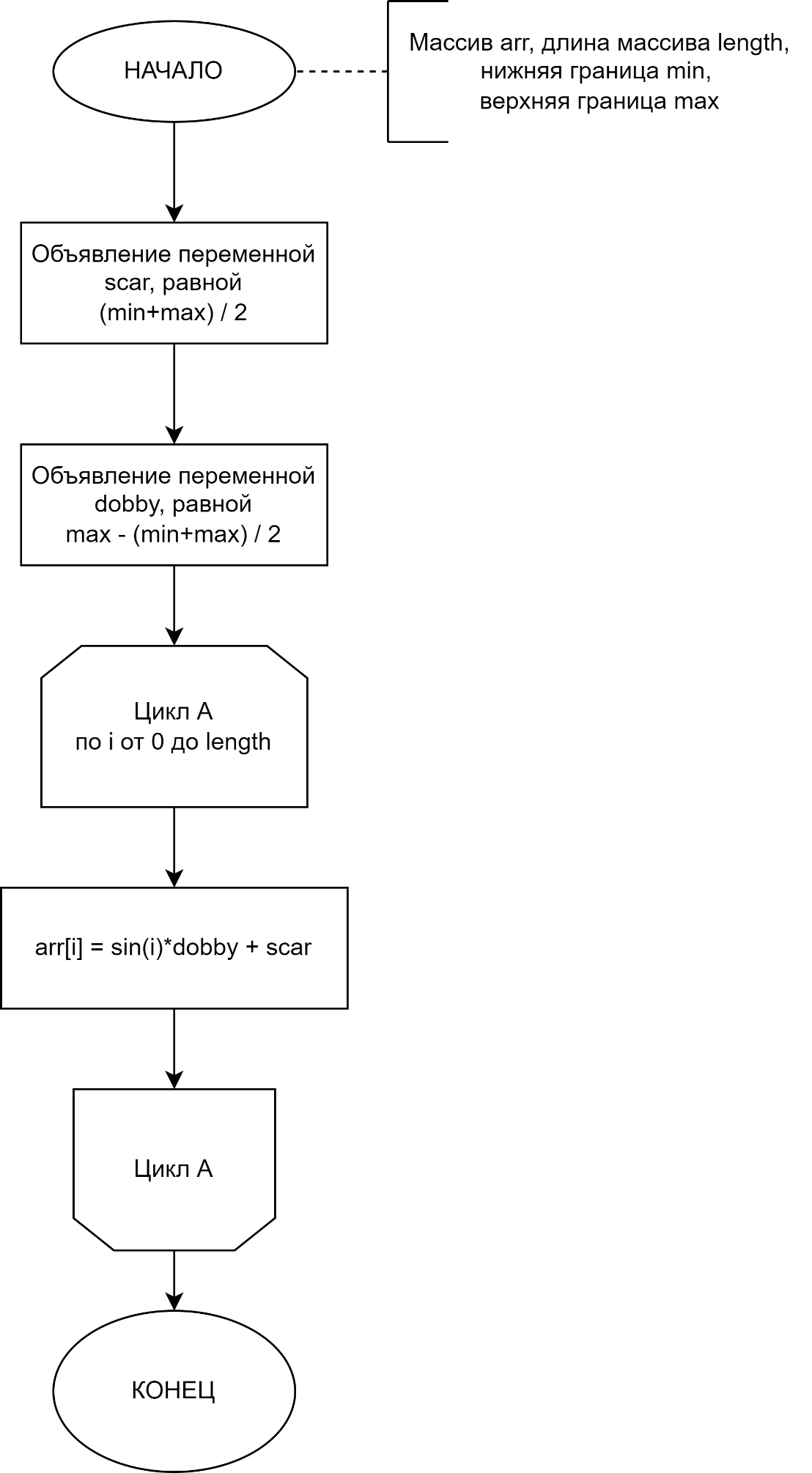


Рисунок 2. Структурная схема алгоритма функции generate\_sinuous()

# Код программы

#include <iostream>

#include <ctime>

#include <fstream>

using namespace std;

void generate\_ascending(float\* arr, unsigned int length);

void generate\_descending(float\* arr, unsigned int length);

void generate\_random(float\* arr, unsigned int length);

void generate\_sawtooth(float\* arr, unsigned int length, float min, float max, float interval);

void generate\_sinuous(float\* arr, unsigned int length, float min, float max);

void generate\_stepped(float\* arr, unsigned int length, float min, float max, float interval);

int random(int min, int max);

float random\_fl();

int random\_sign();

int random(int min, int max) {

return min + (max - min) \* random\_fl();

}

float random\_fl() {

return float(rand() / RAND\_MAX);

}

int random\_sign() {

int q = rand();

if (q > RAND\_MAX / 2)

return 1;

else

return -1;

}

void generate\_ascending(float\* arr, unsigned int length) {

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

\*(arr + i) = random(0, 1000 \* random(0, 3) \* random\_fl());

if (i > 0) {

\*(arr + i) += \*(arr + i - 1);

}

}

}

void generate\_ascending\_int(float\* arr, unsigned int length) {

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

\*(arr + i) = random(0, 1000 \* random(0, 3));

if (i > 0) {

\*(arr + i) += \*(arr + i - 1);

}

}

}

void generate\_descending(float\* arr, unsigned int length){

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

\*(arr + i) = random(-1000 \* random(0,3) \* random\_fl(), 0);

if (i > 0) {

\*(arr + i) += \*(arr + i - 1);

}

}

}

void generate\_descending\_int(float\* arr, unsigned int length) {

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

\*(arr + i) = random(-1000 \* random(0, 3), 0);

if (i > 0) {

\*(arr + i) += \*(arr + i - 1);

}

}

}

void generate\_random(float\* arr, unsigned int length) {

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

\*(arr + i) = random(-10000 \* random(0, 3) \* random\_fl(), 10000 \* random(0, 3) \* random\_fl());

}

}

void generate\_random\_int(float\* arr, unsigned int length) {

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

\*(arr + i) = random(-10000 \* random(0, 3), 10000 \* random(0, 3));

}

}

void generate\_sawtooth(float\* arr, unsigned int length, float min, float max, float interval) {

float step = (max - min) / interval;

int num = 1;

\*(arr) = min;

for (int i = 1; i < length; i++) {

\*(arr + i) = \*(arr + i - 1) + step;

num++;

if (num == interval) {

i++;

\*(arr + i) = min;

num = 1;

}

}

}

void generate\_sawtooth\_int(float\* arr, unsigned int length, int min, int max, float interval) {

int step = (max - min) / interval;

int num = 1;

\*(arr) = min;

for (int i = 1; i < length; i++) {

\*(arr + i) = \*(arr + i - 1) + step;

num++;

if (num == interval) {

i++;

\*(arr + i) = min;

num = 1;

}

}

}

void generate\_sinuous(float\* arr, unsigned int length, float min, float max) {

float scar = (min + max) / 2;

float dobby = max - scar;

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

\*(arr + i) = sin(i) \* dobby + scar;

}

}

void generate\_sinuous\_int(float\* arr, unsigned int length, int min, int max) {

int scar = (min + max) / 2;

int dobby = max - scar;

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

\*(arr + i) = int(sin(i)) \* dobby + scar;

}

}

void generate\_stepped(float\* arr, unsigned int length, float min, float max, int interval) {

float step = (max - min) / (length/interval-1);

\*(arr) = min;

int k = 0;

int num = 1;

for (int i = 1; i < length; i++) {

float q = (rand() / 10000) \* random\_sign();

\*(arr + i) = \*(arr + k) + q;

num++;

if (num == interval) {

num = 0;

k = i + 1;

\*(arr + k) = \*(arr + i) + step;

}

}

}

void generate\_stepped\_int(float\* arr, unsigned int length, int min, int max, int interval) {

int step = (max - min) / (length / interval - 1);

\*(arr) = min;

int k = 0;

int num = 1;

for (int i = 1; i < length; i++) {

int q = (rand() / 10000) \* random\_sign();

\*(arr + i) = \*(arr + k) + q;

num++;

if (num == interval) {

num = 0;

k = i + 1;

\*(arr + k) = \*(arr + i) + step;

}

}

}

void switch\_num(int num, int length, float\* arr) {

int min;

int max;

int interval;

unsigned int start\_time;

unsigned int end\_time;

unsigned int search\_time;

switch (num) {

case 1:

start\_time = clock();

generate\_ascending(arr, length);

search\_time = clock() - start\_time;

cout << "ВРЕМЯ РАБОТЫ ВОЗРАСТАЮЩЕЙ С ПЛАВАЮЩЕЙ ТОЧКОЙ: " << search\_time << endl;

start\_time = clock();

generate\_ascending\_int(arr, length);

search\_time = clock() - start\_time;

cout << "ВРЕМЯ РАБОТЫ ВОЗРАСТАЮЩЕЙ ЦЕЛОЧИСЛЕННОЙ: " << search\_time << endl;

break;

case 2:

start\_time = clock();

generate\_descending(arr, length);

search\_time = clock() - start\_time;

cout << "ВРЕМЯ РАБОТЫ УБЫВАЮЩЕЙ C ПЛАВАЮЩЕЙ ТОЧКОЙ: " << search\_time << endl;

start\_time = clock();

generate\_descending\_int(arr, length);

search\_time = clock() - start\_time;

cout << "ВРЕМЯ РАБОТЫ УБЫВАЮЩЕЙ ЦЕЛОЧИСЛЕННОЙ: " << search\_time << endl;

break;

case 3:

start\_time = clock();

generate\_random(arr, length);

search\_time = clock() - start\_time;

cout << "ВРЕМЯ РАБОТЫ СЛУЧАЙНОЙ С ПЛАВАЮЩЕЙ ТОЧКОЙ: " << search\_time << endl;

start\_time = clock();

generate\_random\_int(arr, length);

search\_time = clock() - start\_time;

cout << "ВРЕМЯ РАБОТЫ СЛУЧАЙНОЙ ЦЕЛОЧИСЛЕННОЙ: " << search\_time << endl;

break;

case 4:

cout << "ВВВЕДИТЕ НИЖНЮЮ ГРАНИЦУ: ";

cin >> min;

cout << "ВВЕДИТЕ ВЕРХНЮЮ ГРАНИЦУ: ";

cin >> max;

cout << "ВВЕДИТЕ ИНТЕРВАЛ: ";

cin >> interval;

start\_time = clock();

generate\_sawtooth(arr, length, min, max, interval);

search\_time = clock() - start\_time;

cout << "ВРЕМЯ РАБОТЫ ПИЛООБРАЗНОЙ С ПЛАВАЮЩЕЙ ТОЧКОЙ: " << search\_time << endl;

start\_time = clock();

generate\_sawtooth\_int(arr, length, min, max, interval);

search\_time = clock() - start\_time;

cout << "ВРЕМЯ РАБОТЫ ПИЛООБРАЗНОЙ ЦЕЛОЧИСЛЕННОЙ: " << search\_time << endl;

break;

case 5:

cout << "ВВВЕДИТЕ НИЖНЮЮ ГРАНИЦУ: ";

cin >> min;

cout << "ВВЕДИТЕ ВЕРХНЮЮ ГРАНИЦУ: ";

cin >> max;

start\_time = clock();

generate\_sinuous(arr, length, min, max);

search\_time = clock() - start\_time;

cout << "ВРЕМЯ РАБОТЫ ДЛЯ СИНУСОИДНОЙ С ПЛАВАЮЩЕЙ ТОЧКОЙ: " << search\_time << endl;

start\_time = clock();

generate\_sinuous\_int(arr, length, min, max);

search\_time = clock() - start\_time;

cout << "ВРЕМЯ РАБОТЫ ДЛЯ СИНУСОИДНОЙ ЦЕЛОЧИСЛЕННОЙ: " << search\_time << endl;

break;

case 6:

cout << "ВВВЕДИТЕ НИЖНЮЮ ГРАНИЦУ: ";

cin >> min;

cout << "ВВЕДИТЕ ВЕРХНЮЮ ГРАНИЦУ: ";

cin >> max;

cout << "ВВЕДИТЕ ИНТЕРВАЛ: ";

cin >> interval;

start\_time = clock();

generate\_stepped(arr, length, min, max, interval);

search\_time = clock() - start\_time;

cout << "ВРЕМЯ РАБОТЫ ДЛЯ СТУПЕНЧАТОЙ С ПЛАВАЮЩЕЙ ТОЧКОЙ " << search\_time << endl;

start\_time = clock();

generate\_stepped\_int(arr, length, min, max, interval);

search\_time = clock() - start\_time;

cout << "ВРЕМЯ РАБОТЫ ДЛЯ СТУПЕНЧАТОЙ ЦЕЛОЧИСЛЕННОЙ " << search\_time << endl;

break;

}

}

const string filename = "file.txt";

int main()

{

srand(time(NULL));

setlocale(LC\_ALL, "rus");

ofstream fout(filename);

cout << "КАКОЙ МАССИВ ХОТИТЕ СОЗДАТЬ?\n1.ВОЗРАСТАЮЩИЙ\n2.УБЫВАЮЩИЙ\n3.СЛУЧАЙНЫЙ\n4.ПИЛООБРАЗНЫЙ\n5.СИНУСОИДНЫЙ\n6.СТУПЕНЧАТЫЙ\nВаш выбор: ";

int num;

cin >> num;

cout << endl;

int min;

int max;

int interval;

int length;

system("cls");

cout << "ВВЕДИТЕ ДЛИНУ МАССИВА: ";

cin >> length;

float\* arr = new float[length];

switch\_num(num, length, arr);

if (length < 150 || length > 200) {

int var;

cout << "ВЫВЕСТИ МАТРИЦУ:\n1.ДА\n2.НЕТ\nОТВЕТ: ";

cin >> var;

cout << "\n";

if (var == 1)

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

cout << arr[i] << "\n";

else

return 0;

}

else {

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

cout << arr[i] << "\n";

fout << arr[i] << "\n";

}

}

}

# Графики полученных массивов (для 100 элементов)

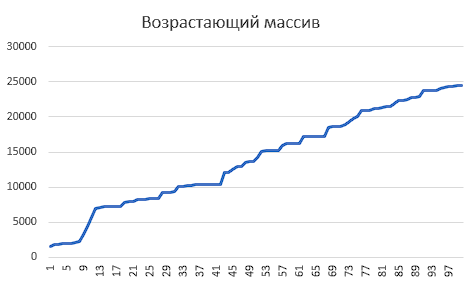


Рисунок 3. График возрастающего массива

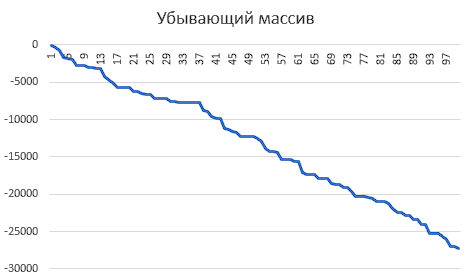


Рисунок 4. График убывающего массива



Рисунок 5. График случайного массива



Рисунок 6. График пилообразного массива



Рисунок 7. График синусоидного массива



Рисунок 8. График ступенчатого массива

# Графики зависимости времени (в мс) от размерности

## Для массивов из float-элементов

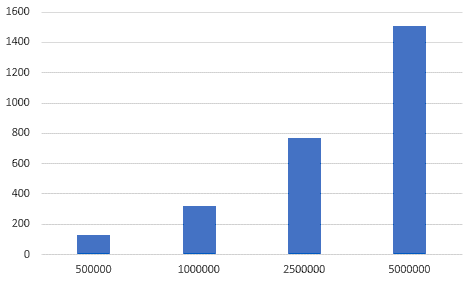


Рисунок 9. График зависимости времени от размерности возрастающего массива

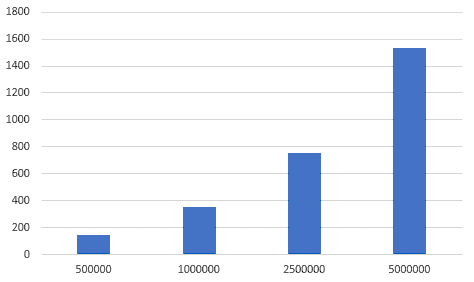


Рисунок 10. График зависимости времени от размерности убывающего массива

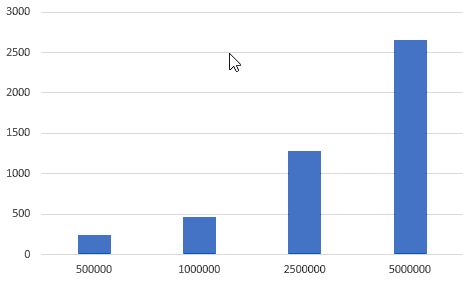


Рисунок 11. График зависимости времени от размерности случайного массива

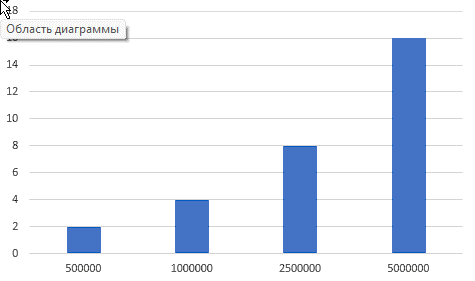


Рисунок 12. График зависимости времени от размерности пилообразного массива

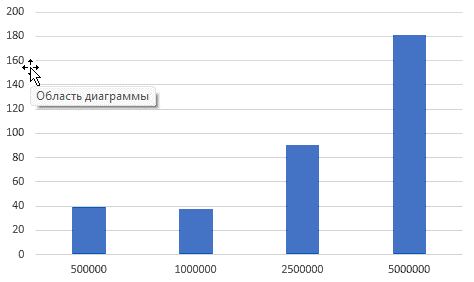


Рисунок 13. График зависимости времени от размерности синусоидного массива

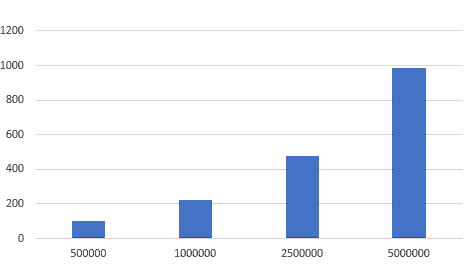


Рисунок 14. График зависимости времени от размерности ступенчатого массива

## Для массивов с int-элементами

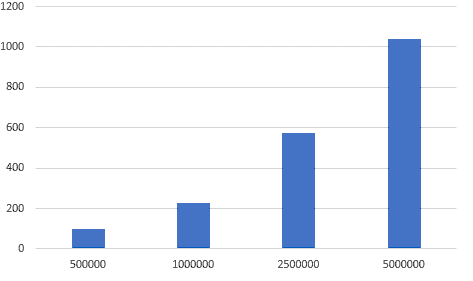


Рисунок 15. График зависимости времени от размерности возрастающего целочисленного массива

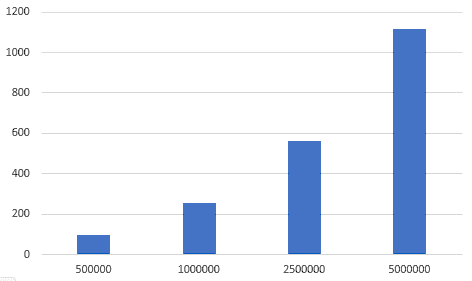


Рисунок 16. График зависимости времени от размерности убывающего целочисленного массива

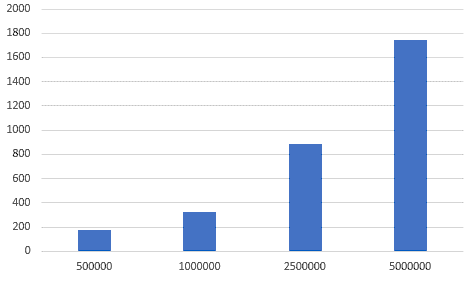


Рисунок 17. График зависимости времени от размерности случайного целочисленного массива

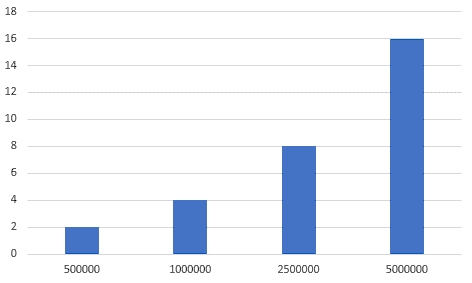


Рисунок 18. График зависимости времени от размерности пилообразного целочисленного массива

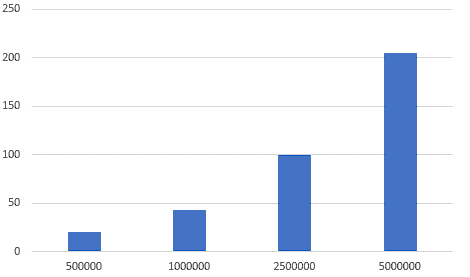


Рисунок 19. График зависимости времени от размерности синусоидного целочисленного массива

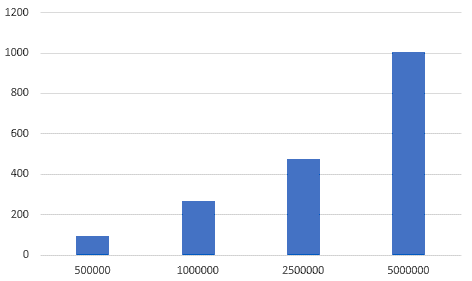


Рисунок 20. График зависимости времени от размерности ступенчатого целочисленного массива

# Вывод

Программа работает корректно на основе проделанных тестов. Были освоены принципы формирования монотонных возрастающих и убывающих массивов, а также пилообразных, ступенчатых и синусоидальных массивов. Также были изучены функции, позволяющие производить оценку длительности времени выполнения алгоритмов, а именно функция clock() из библиотеки ctime. Были изучены и использованы правила оформления функций с использованием указателей на функции в качестве формальных параметров.