Бюджетное учреждение высшего образования Ханты-Мансийского автономного округа – Югры «СУРГУТСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Политехнический институт

Кафедра автоматики и компьютерных систем

Отчет

по лабораторной работе № 1

по дисциплине «Алгоритмы и структуры данных»

Проверила: старший преподаватель кафедры АиКС,

Карамышев. Э. Р

Выполнил: студент группы 605-31,

Хайитов Ш. Д.

Сургут 2025

СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 3](#_Toc194868567)

[ОСНОВНАЯ ЧАСТЬ 4](#_Toc194868568)

[ЗАКЛЮЧЕНИЕ 10](#_Toc194868569)

# **ВВЕДЕНИЕ**

**Цель работы:** Освоить принципы формирования монотонных (упорядоченных и упорядоченных в обратном порядке), частично упорядоченных и случайных последовательностей данных; изучить функции, позволяющие производить оценку длительности времени выполнения алгоритмов.

**Задания:**

1. Разработать функции, формирующие упорядоченные (в том числе и в обратном порядке), частично упорядоченные и случайные последовательности целых чисел и чисел с плавающей запятой. При реализации функций считать, что выделение памяти под последовательности происходит вне этих функций, а в качестве формальных параметров функции получают указатель на массив, его размер и, возможно, диапазон изменения величин и длину интервалов (для частично упорядоченных последовательностей). Рекомендуется продумать единый прототип для функций, формирующих последовательности, определить его и использовать массив указателей на функции для автоматизации сбора статической информации.
2. Оценить длительность формирования последовательностей всех типов для нескольких значений размеров последовательностей (5\*10^5, 10\*10^5, …, 50\*10^5), и на основе полученных значений построить графики зависимостей длительностей формирования от размера последовательностей.

# **ОСНОВНАЯ ЧАСТЬ**

Блок схема функции, формирующей упорядоченную последовательность представлена на рисунке 1.

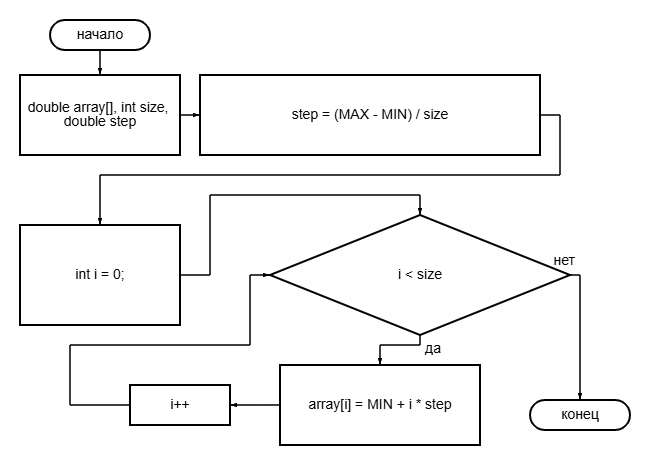


Рисунок 1 — Блок схема упорядоченной последовательности

Блок схема функции main представлена на рисунке 2.

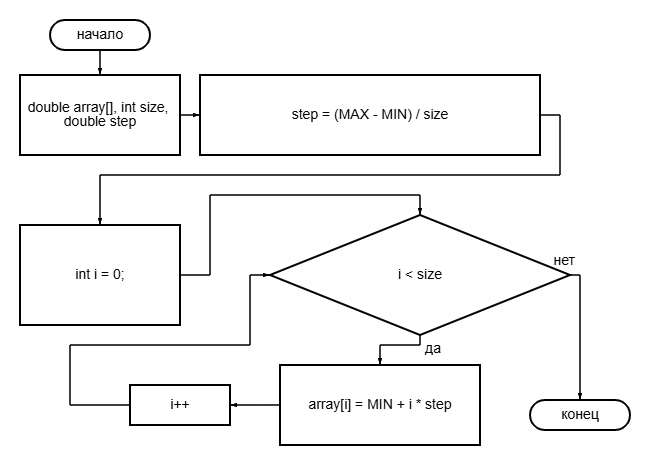


Рисунок 2 — Блок схема главной функции

**Листинг программы:**

#include <iostream>

#include <fstream>

#include <cmath>

#include <cstdlib>

#include <ctime>

#define MIN -50

#define MAX 50

#define d 5

**using** **namespace** std**;**

// Функция для вывода массива

void output**(**double array**[],** int size**)** **{**

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

cout **<<** array**[**i**];**

**if** **(**i **+** 1 **!=** size**)** **{**

cout **<<** ", "**;**

**}**

**}**

cout **<<** endl**;**

**}**

void ascending\_order**(**double array**[],** int size**)** **{**

double step **=** **static\_cast<**double**>(**MAX **-** MIN**)** **/** size**;**

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

array**[**i**]** **=** MIN **+** i **\*** step**;**

**}**

**}**

void descending\_order**(**double array**[],** int size**)** **{**

double step **=** **static\_cast<**double**>(**MAX **-** MIN**)** **/** size**;**

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

array**[**i**]** **=** MAX **-** i **\*** step**;**

**}**

**}**

void random\_order**(**double array**[],** int size**)** **{**

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

array**[**i**]** **=** MIN **+** rand**()** **%** **(**MAX **-** MIN **+** 1**);**

**}**

**}**

void sawtooth**(**double array**[],** int size**)** **{**

int interval **=** size **/** d**;**

double step **=** **static\_cast<**double**>(**MAX **-** MIN**)** **/** interval**;**

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

array**[**i**]** **=** MIN **+** step **\*** **(**i **%** interval**);**

**}**

**}**

void stepwise**(**double array**[],** int size**)** **{**

int interval **=** size **/** d**;**

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

array**[**i**]** **=** MIN **+** **(**MAX **-** MIN**)** **\*** **(**i **/** interval**)** **/** d**;**

**}**

**}**

void sinus**(**double array**[],** int size**)** **{**

double amplitude **=** **static\_cast<**double**>(**MAX **-** MIN**)** **/** 2**;**

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

double x **=** **(**2 **\*** M\_PI **\*** i**)** **/** size**;**

array**[**i**]** **=** MIN **+** amplitude **\*** **(**sin**(**x**)** **+** 1**);**

**}**

**}**

void mix**(**double array**[],** int size**)** **{**

double step **=** **static\_cast<**double**>(**MAX **-** MIN**)** **/** **(**size **/** d**);**

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

array**[**i**]** **=** MIN **+** step **\*** **(**i **/** d**)** **+** sin**(**i **\*** **(**M\_PI **\*** 2 **/** d**))** **\*** step **/** 2**;**

**}**

**}**

double measure\_time**(**void **(\***func**)(**double**[],** int**),** double **\***arr**,** int size**)** **{**

clock\_t start **=** clock**();**

func**(**arr**,** size**);**

clock\_t end **=** clock**();**

**return** **static\_cast<**double**>(**end **-** start**)** **/** CLOCKS\_PER\_SEC**;**

**}**

void runAll**(**int size**)** **{**

double array**[**size**]** **=** **{**0**};**

cout **<<** "Упорядоченная:" **<<** endl**;**

ascending\_order**(**array**,** size**);**

output**(**array**,** size**);** // Вывод массива

cout **<<** "Упорядоченная в обратном порядке:" **<<** endl**;**

descending\_order**(**array**,** size**);**

output**(**array**,** size**);**

cout **<<** "Рандом:" **<<** endl**;**

random\_order**(**array**,** size**);**

output**(**array**,** size**);**

cout **<<** "Пилообразная:" **<<** endl**;**

sawtooth**(**array**,** size**);**

output**(**array**,** size**);**

cout **<<** "Ступенчатая:" **<<** endl**;**

stepwise**(**array**,** size**);**

output(array, size);

cout << "Синусоидальная:" << endl;

sinus(array, size);

output(array, size);

cout << "Квази:" << endl;

mix(array, size);

output(array, size);

}

void test\_time(double array[], int size) {

double ascending\_time = measure\_time(ascending\_order, array, size);

double descending\_time = measure\_time(descending\_order, array, size);

double random\_time = measure\_time(random\_order, array, size);

double sawtooth\_time = measure\_time(sawtooth, array, size);

double stepwise\_time = measure\_time(stepwise, array, size);

double sinus\_time = measure\_time(sinus, array, size);

double mix\_time = measure\_time(mix, array, size);

cout << endl << "Время выполнения:" << endl;

cout << " Упорядоченная: " << ascending\_time << " сек" << endl;

cout << " Упорядоченная в обратном порядке: " << descending\_time << " сек" << endl;

cout << " Рандом: " << random\_time << " сек" << endl;

cout << " Пилообразная: " << sawtooth\_time << " сек" << endl;

cout << " Ступенчатая: " << stepwise\_time << " сек" << endl;

cout << " Синусоидальная: " << sinus\_time << " сек" << endl;

cout << " Квази: " << mix\_time << " сек" << endl;

}

int main() {

system("chcp 65001 > nul");

int size = 10 \* pow(10, 5);

double \*array = new double[size];

// runAll(size);

test\_time(array, size);

delete[] array;

return 0;

}

Графики работы функций последовательности представлены на рисунке 3.

**A screenshot of a graph

AI-generated content may be incorrect.**

Рисунок 3 — Графики функций

График зависимости длительности формирования последовательности от размера последовательности представлен на рисунке 4.

A graph with different colored lines

AI-generated content may be incorrect.

Рисунок 4 — График зависимости длительности

Время генерации значений для всех функций, создающих последовательности, показало линейную зависимость от размера массива.

Функция, отвечающая за генерацию синусоидальной последовательности, заняла больше времени из-за высокой вычислительной сложности функции sin() из библиотеки cmath. В то же время, функция, генерирующая квази-упорядоченные последовательности, показала более длительное время выполнения по сравнению с синусоидальной последовательностью. Это связано с тем, что в реализации квази-упорядоченной последовательности была использована основа функции ступенчатой последовательности в сочетании с функциями sin() и rand(). В результате общее время выполнения этих функций оказалось выше, чем у синусоидальной последовательности.

# **ЗАКЛЮЧЕНИЕ**

В ходе выполнения лабораторной работы была проведена оценка формирования упорядоченных, частично упорядоченных и случайных последовательностей. Также было рассмотрено, как время, необходимое для создания этих последовательностей, зависит от их размера.