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 № 1

Формирование массивов экспериментальных данных

Выполнил студент группы М3О-210Б-20

Кильмишкин Никита Владимирович

Проверила доцент, к.т.н., Дмитриева Е.А.

Москва 2021 г.

Оглавление

[**ТЗ** 3](#_Toc83461197)

[**Структурные схемы алгоритма** 4](#_Toc83461198)

[Модуль main() 4](#_Toc83461199)

[Модуль FSin() 5](#_Toc83461200)

[**Листинг программы** 6](#_Toc83461201)

[**Графики** 10](#_Toc83461202)

[Графики вещественных функций 10](#_Toc83461203)

[Вещественный массив 13](#_Toc83461204)

[Целочисленный массив 14](#_Toc83461205)

[**Вывод** 15](#_Toc83461206)

**ТЗ**

**1 этап работы:**

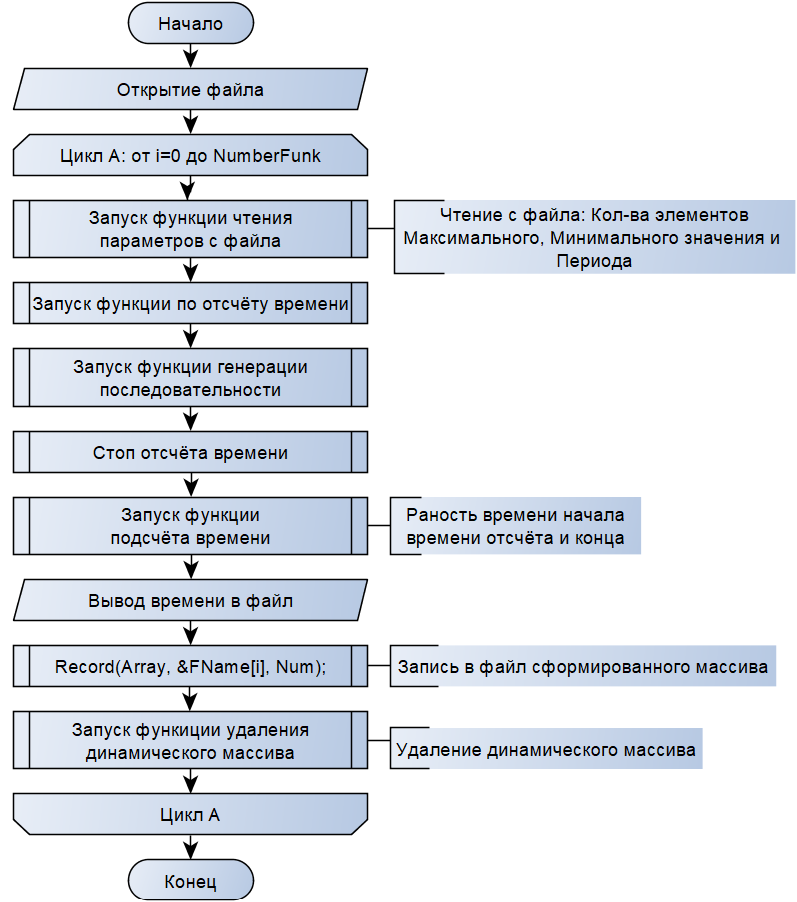
Разработать функции, формирующие упорядоченные (по возрастанию и убыванию), случайные последовательности и частично упорядоченные («пилообразные», «синусоидальные», «ступенчатые», «квази-упорядоченные» - из этого списка любые две на выбор) **для целых чисел и чисел с плавающей запятой.**

**2 этап работы:**

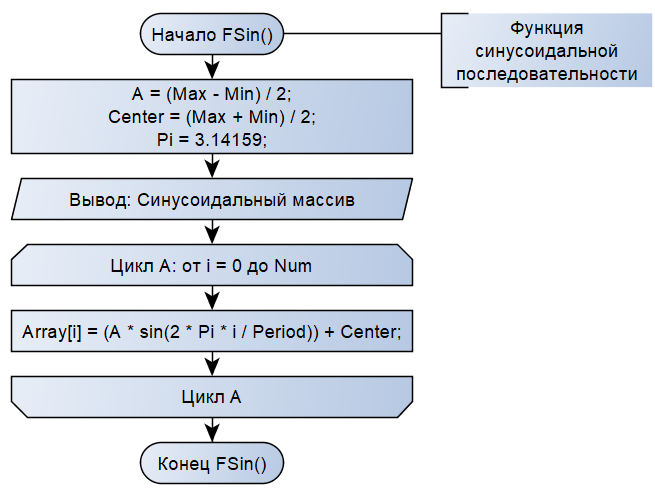
1. Оценить длительность формирования последовательностей всех типов для различных значений размеров последовательностей (5⋅105, 10⋅105, …, 50⋅105), и на основе полученных значений построить графики зависимостей длительностей формирования от размера последовательностей (лучше в виде столбчатых диаграмм, можно в Microsoft Exel).
2. Составить отчет, в котором привести структурные схемы алгоритмов главной функции и какой-то одной из функций (например, реализующей частично упорядоченную последовательность), текст программы, полученные графики зависимостей и выводы по полученным результатам.

**Структурные схемы алгоритма**

Модуль main()



Модуль FSin()



**Листинг программы**

#include <iostream>

#include <fstream>

#include <math.h>

#include <Windows.h>

#include <chrono>

#include <cstdlib>

#include <ctime>

using namespace std;

// Упорядочивание чисел по убыванию

void Uporyad321( double\*\* Array, // Вещественный динамический массива

int\*\* ArrayInt, // Целочисленный динамический массив

int Num, // Число элементов массива

int Max, // Максимальное значение

int Min, // Минимальное значение

int Period); // Период функции

// Упорядочивание чисел по возрастанию

void Uporyad123( double\*\* Array, // Вещественный динамический массива

int\*\* ArrayInt, // Целочисленный динамический массив

int Num, // Число элементов массива

int Max, // Максимальное значение

int Min, // Минимальное значение

int Period); // Период функции

// Случайная последовательность

void Rand( double\*\* Array, // Вещественный динамический массива

int\*\* ArrayInt, // Целочисленный динамический массив

int Num, // Число элементов массива

int Max, // Максимальное значение

int Min, // Минимальное значение

int Period); // Период функции

// Пилообразное упорядочивание чисел

void Pila( double\*\* Array, // Вещественный динамический массива

int\*\* ArrayInt, // Целочисленный динамический массив

int Num, // Число элементов массива

int Max, // Максимальное значение

int Min, // Минимальное значение

int Period); // Период функции

// Синусоидальное упорядочивание чисел

void FSin( double\*\* Array, // Вещественный динамический массива

int\*\* ArrayInt, // Целочисленный динамический массив

int Num, // Число элементов массива

int Max, // Максимальное значение

int Min, // Минимальное значение

int Period); // Период функции

// Чтение с файла

void Read( double\*\* Array, // Динамический массив

int\*\* ArrayInt, // Целочисленный динамический массив

int& Num, // Число элементов массива

int& Max, // Максимальное значение

int& Min, // Минимальное значение

int& Period); // Период функции

// Запись на файл

void Record( double\* Array,

int\* ArrayInt,

const char\* FName[],

int Num);

// Удаление динамического массива

void DeleteMas( double \*Array, int \*ArrayInt);

// "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_"

void BotBorder();

const int NumberFunc = 5; // Количество функций

// Массивы с именами файлов

const char\* FName[] = { {"Вывод 123.csv"}, {"Вывод 321.csv"}, {"Вывод Rand.csv"}, {"Вывод Pila.csv"}, {"Вывод Sin.csv"} }; // Exel файлы для вывода результатов каждой фунции

const char FNameIn[] = "Universe.txt";

const char FNameOut[] = "Универсальный вывод.csv"; // Файл для таблицы Exel

ofstream foutUn(FNameOut); // Открытие файла для записи массивов

int main()

{

system("color F0"); // Смена цвета

setlocale(LC\_ALL, "Russian"); // Локализация

ofstream fout("Time.csv"); // Открытие файла для записи

// Массив указателей на функции

void(\*Menu[NumberFunc]) (double\*\* Array, int\*\* ArrayInt, int Num, int Max, int Min, int Period) =

{ Uporyad321, Uporyad123, Rand, Pila, FSin };

double\* Array; // Динамический массив

int\* ArrayInt; // Целочисленный динамический массив

int Num = 0; // Число элементов массива

int Max = 0; // Максимальное значение

int Min = 0; // Минимальное значение

int Period = 0; // Период функции

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

{

Read(&Array, &ArrayInt, Num, Max, Min, Period); // Чтение

auto begin = chrono::steady\_clock::now(); // Запуск отсчёта

Menu[i](&Array, &ArrayInt, Num, Max, Min, Period);

auto end = std::chrono::steady\_clock::now(); // получаем время по окончанию формирования последовательности

auto elapsed\_ms = std::chrono::duration\_cast<std::chrono::microseconds>(end - begin); // получаем время работы в микросекунд

fout << elapsed\_ms.count() << ";" << endl; // Запись времени в файл

foutUn << elapsed\_ms.count() << ";;" << endl; // Запись времени в общий файл

Record(Array, ArrayInt, &FName[i], Num); foutUn << ";" << endl; // Запись функции в файл

DeleteMas(Array, ArrayInt);

BotBorder();

}// for i

}// main

void Read( double \*\*Array, // Вещественный динамический массив

int\*\* ArrayInt, // Целочисленный динамический массив

int& Num, // Число элементов массива

int& Max, // Максимальное значение

int& Min, // Минимальное значение

int& Period) // Период функции

{

ifstream fin(FNameIn);

fin >> Num; cout << " Num = " << Num << endl;

fin >> Max; cout << " Max = " << Max << endl;

fin >> Min; cout << " Min = " << Min << endl;

fin >> Period; cout << " Period = " << Period << endl;

(\*Array) = new double[Num]();

(\*ArrayInt) = new int[Num]();

fin.close();

} //Read()

void Uporyad321( double\*\* Array, // Динамический массив

int\*\* ArrayInt, // Целочисленный динамический массив

int Num, // Число элементов массива

int Max, // Максимальное значение

int Min, // Минимальное значение

int Period) // Период функции

{

double k = double(Max- Min) / double(Num-1); // Угол наклона

cout << " Убывающая последовательность" << endl;

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

{

(\*Array)[i] = (i \* (-k) + Max);

(\*ArrayInt)[i] = int(i \* (-k) + Max);

} //for i

}// Uporad321()

void Uporyad123( double\*\* Array, // Динамический массив

int\*\* ArrayInt, // Целочисленный динамический массив

int Num, // Число элементов массива

int Max, // Максимальное значение

int Min, // Минимальное значение

int Period) // Период функции

{

double k = double(Max - Min) / double(Num - 1); // Угол наклона

cout << " Возрастающая последовательность" << endl;

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

{

(\*Array)[i] = (i \* k + Min);

(\*ArrayInt)[i] = int((i \* k + Min));

}// for i

}// Uporyad123()

void Rand( double\*\* Array, // Динамический массив

int\*\* ArrayInt, // Целочисленный динамический массив

int Num, // Число элементов массива

int Max, // Максимальное значение

int Min, // Минимальное значение

int Period) // Период функции

{

srand(time(0));

cout << " Случайная последовательность:" << endl;

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

{

(\*Array)[i] = (Min + (rand() % (Max - Min)));

(\*ArrayInt)[i] = int(Min + (rand() % (Max - Min)));

} // for i

}// Rand()

void Pila( double\*\* Array, // Динамический массив

int\*\* ArrayInt, // Целочисленный динамический массив

int Num, // Число элементов массива

int Max, // Максимальное значение

int Min, // Минимальное значение

int Period) // Период функции

{

double Height = Max - Min;

cout << " Пилообразный массив" << endl;

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

{

(\*Array)[i] = ((i % Period) \* (Height / Period) + Min);

(\*ArrayInt)[i] = int((i % Period) \* (Height / Period) + Min);

}// for i

}// Pila()

void FSin( double\*\* Array, // Динамический массив

int\*\* ArrayInt, // Целочисленный динамический массив

int Num, // Число элементов массива

int Max, // Максимальное значение

int Min, // Минимальное значение

int Period) // Период функции

{

double A = (Max - Min) / 2;

double Center = (Max + Min) / 2;;

double Pi = 3.14159;

cout << " Синусоидальный массив" << endl;

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

{

(\*Array)[i] = ((A \* sin(2 \* Pi \* i / Period)) + Center);

(\*ArrayInt)[i] = int((A \* sin(2 \* Pi \* i / Period)) + Center);

}// for i

}// FSin()

void DeleteMas(double \*Array, int \*ArrayInt)

{

delete[] Array; delete[] ArrayInt;

} // DeleteMas()

void BotBorder()

{

cout << "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_" << endl << endl;

} // BotBorder()

void Record( double\* Array,

int\* ArrayInt,

const char\* FName[],

int Num)

{

ofstream fout(\*FName);

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

{

cout << " [" << i << "] : " << Array[i] << endl;

fout << Array[i] << ";;";

foutUn << ";" << Array[i] << ";";

fout << ArrayInt[i] << ";" << endl;

foutUn << ";" << ArrayInt[i] << ";" << endl;

}// for i

fout.close();

} //Record()

**Графики**

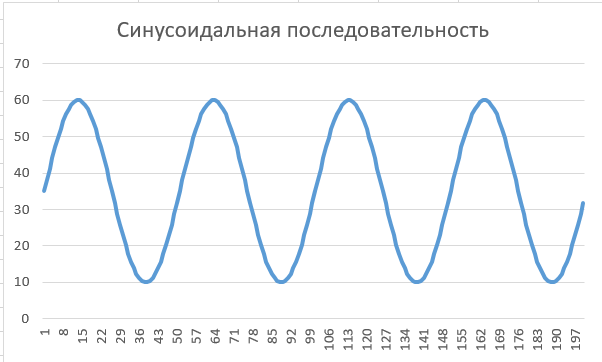
Графики вещественных функций



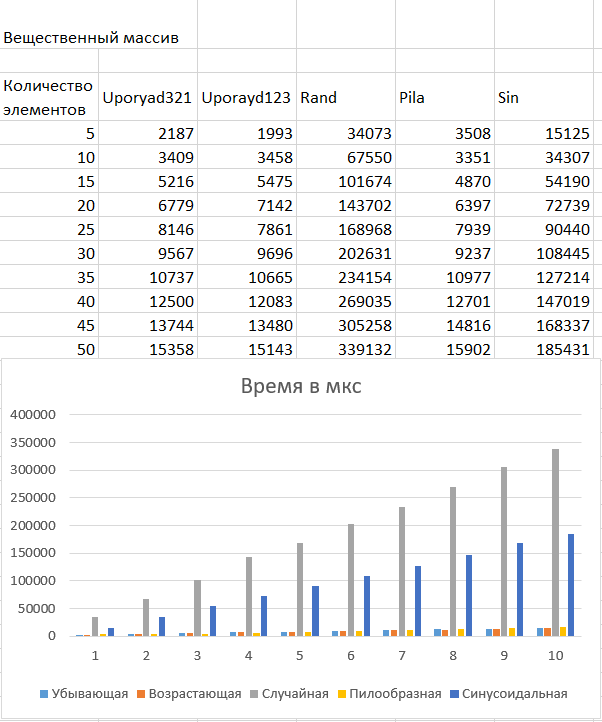




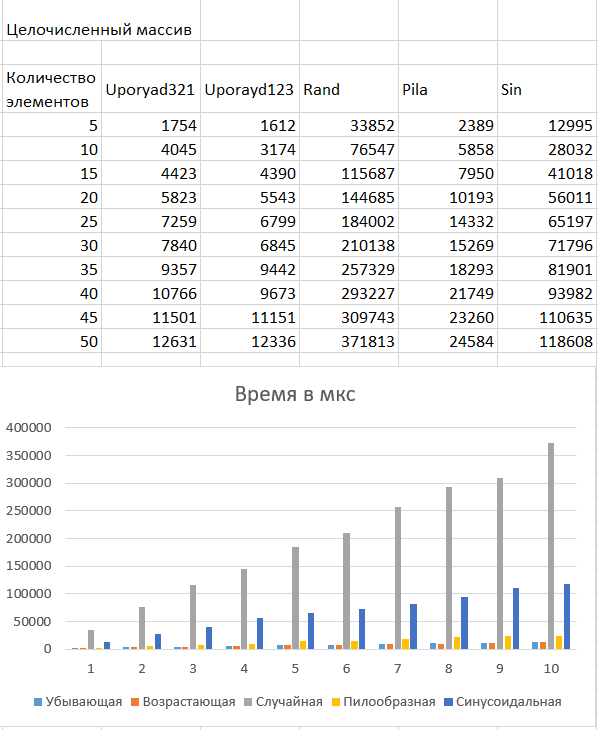




Вещественный массив



Целочисленный массив



**Вывод**

В лабораторной работе №1 было выявлено:

* Освоены принципы формирования монотонных (упорядоченных и упорядоченных в обратном порядке), частично упорядоченных и случайных последовательностей данных;
* Изучены функции, позволяющие производить оценку длительности времени выполнения алгоритмов;
* С увеличением количества элементов время работы программы увеличивается;
* Самая медленная функция – генерация случайной последовательности;
* Время генерации функций возрастания, убывания и пилообразной последовательности одинаковое;
* Целочисленный массив обрабатывается быстрее, чем вещественный.