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»

Программирование

Отчет по лабораторной работе №2

Формирование массивов экспериментальных данных

Выполнили студенты группы M3О-211Б-21

Плоцкий Б.А.

Раужев Ю.М.

Проверила Дмитриева Е.А.

Москва 2022 г.

Оглавление

[Задание 3](#_Toc115035236)

[Общая блок-схема программы 4](#_Toc115035237)

[Структурные схемы алгоритмов 5](#_Toc115035238)

[Функция first\_task() 5](#_Toc115035239)

[Функция second\_task() 8](#_Toc115035240)

[Функция f1 () 11](#_Toc115035241)

[Код программы 12](#_Toc115035242)

[Графики функций 23](#_Toc115035243)

[Результаты по временным интервалам с различными размерами массивов 25](#_Toc115035244)

[Тестирование программы 26](#_Toc115035245)

[Тестирование некорректных режимов 26](#_Toc115035246)

[Тест 1 26](#_Toc115035247)

[Тестирование корректных режимов 27](#_Toc115035248)

[Тест 1 27](#_Toc115035249)

[Вывод 29](#_Toc115035250)

# Задание

Для алгоритмов **BLS** и **SLS** в качестве входного массива использовать одну и ту же последовательность значений (функция **rand**( )).

Для алгоритмов **Т** и **В** – значения массива должны быть отсортированы по не убыванию, одна и та же последовательность чисел (можно использовать соответствующую функцию из первой лабораторной работы).

Оценить длительность поиска для различных значений размеров последовательностей (начиная с 10000 до 200000 элементов массива, провести измерения не менее, чем для 10 разных размерностей).

Для каждой размерности рассматриваются случаи нахождения ключа поиска в начале, в середине и в конце массива.

Для алгоритмов **BLS** и **SLS** кроме подсчета **времени,** необходимого для поиска, требуется определить сколько раз выполняются операции **сравнения** (сравнение ключа с элементом массива, а также в одном из этих двух алгоритмов добавляется подсчет сравнений при анализе индекса элемента массива в цикле… ).

Все результаты оформить в виде таблиц и графиков. На графиках **- только временные характеристики** поиска.

По результатам сделать выводы об эффективности того или иного алгоритма поиска.

# Общая блок-схема программы



# Структурные схемы алгоритмов

## Функция first\_task()

1. Назначение:

Разработать функции, формирующие последовательности чисел:

* + упорядоченные по возрастанию,
  + упорядоченные по убыванию,
  + случайную последовательность,
  + частично упорядоченные: «пилообразные», «синусоидальные», «ступенчатые».

1. Прототип функции:

void first\_task();

1. Обращение:

first\_task();

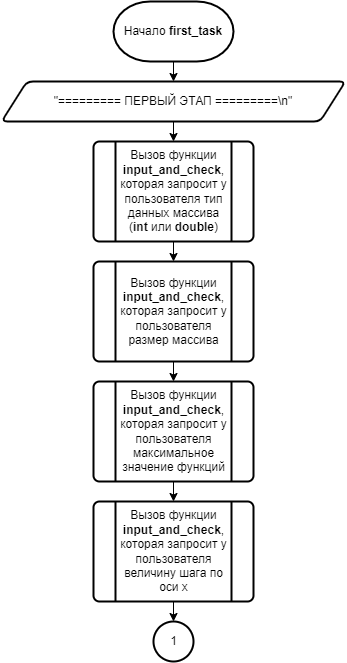
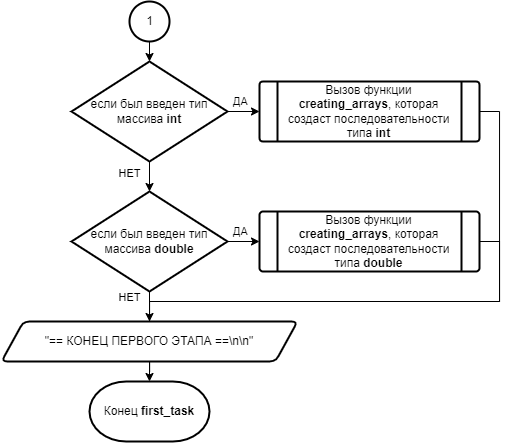
 

Рисунок 1. Структурная схема алгоритма функции first\_task().

## Функция second\_task()

1. Назначение:

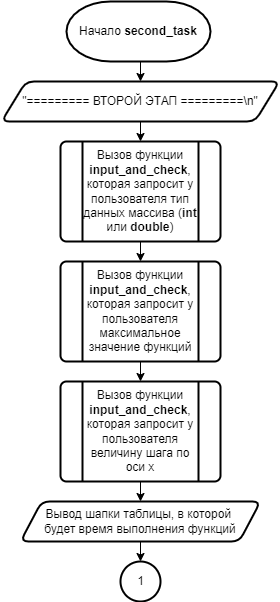
Оценить длительность формирования последовательностей всех типов для различных значений размеров последовательностей (5⋅105, 10⋅105, …, 50⋅105)

1. Прототип функции:

void second\_task();

1. Обращение

second\_task();



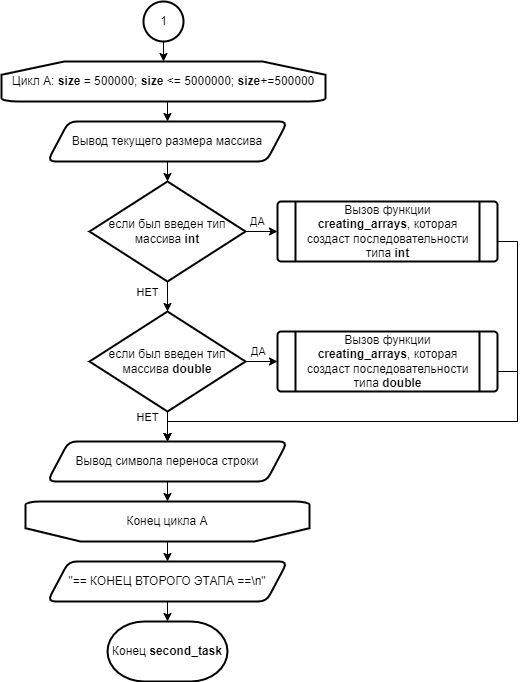


Рисунок 2. Структурная схема алгоритма функции second\_task().

## Функция f1 ()

1. Назначение  
   Формирование последовательности по возрастанию.
2. Прототип функции

template<typename T>

void f1(

T\* arr,

int size,

double max,

double step

);

1. Обращение

f1(arr, size, max, x\_step);

1. Описание параметров

|  |  |  |  |
| --- | --- | --- | --- |
| Идентификаторы | Тип | Назначение | Входной/Выходной |
| arr | double\* или int\* | Массив | Выходной |
| size | int | Размер массива | Входной |
| max | double | Максимальное значение функции | Входной |
| x\_step | double | Шаг по оси х | Входной |

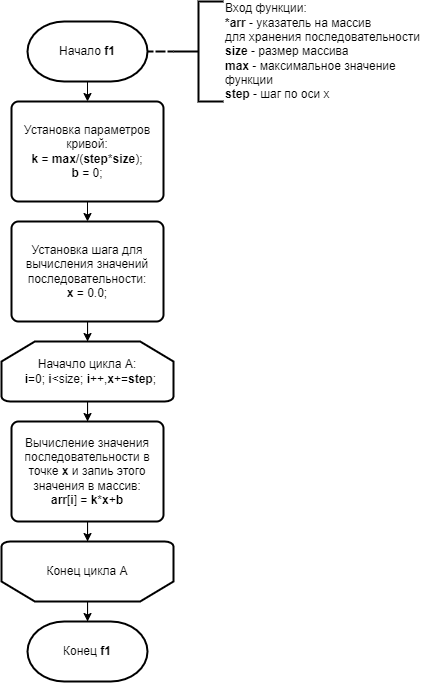


Рисунок 3. Структурная схема алгоритма функции f1()

# Код программы

*/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\  
\*                    кафедра № 304 2 курс 3 семестр информатика         \*  
\*-----------------------------------------------------------------------\*  
\*   Project type : solution                                             \*  
\*   Project name : LW1                                                  \*  
\*   File name    : main.cpp                                             \*  
\*   Language     : c/c++                                                \*  
\*   Programmers  : Плоцкий Б.А. Раужев Ю. М.                            \*  
\*   Created      :  12/09/22                                            \*  
\*   Last revision:  25/09/22                                            \*  
\*   Comment(s)   :                                                      \*  
\*                                                                       \*  
\*   1 этап работы                                                       \*  
\*   Разработать функции, формирующие последовательности чисел:          \*  
\*   - упорядоченные по возрастанию,                                     \*  
\*   - упорядоченные по убыванию,                                        \*  
\*   - случайную последовательность,                                     \*  
\*   - частично упорядоченные: «пилообразные», «синусоидальные»,         \*  
\*       «ступенчатые».                                                  \*  
\*                                                                       \*  
\*   > для целых чисел и чисел с плавающей запятой                        \*  
\*   > (должно быть два комплекта функций и соответственно результатов).  \*  
\*                                                                       \*  
\*   2 этап работы                                                       \*  
\*   Оценить длительность формирования последовательностей всех типов    \*  
\*   для различных значений размеров последовательностей,                \*  
\*   и на основе полученных значений построить графики зависимостей      \*  
\*   длительностей формирования массивов от их размера (лучше в виде     \*  
\*   столбчатых диаграмм, можно в Microsoft Excel).                      \*  
\*                                                                       \*  
\*   Составить отчет, в котором привести структурные схемы алгоритмов    \*  
\*   главной функции и какой-то одной из функций (например,              \*  
\*   реализующей частично упорядоченную последовательность),             \*  
\*   текст программы, результаты по временным интервалам для различных   \*  
\*   размерностей массивов, графики зависимостей (времени от             \*  
\*   размерностей) и выводы по полученным результатам.                   \*  
\*                                                                       \*  
\\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  
   
#define \_USE\_MATH\_DEFINES   // для PI  
   
#include <iostream>  
#include <fstream>  
#include <ctime>  
#include <cmath>  
#include <iomanip>  
#include <chrono>  
#include "gui/GraphsDrawer.h"   // для отрисовки графиков****using******namespace****std;  
   
/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
\*                      К О Н С Т А Н Т Ы                        \*  
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  
   
// имена файлов****const******char****name\_of\_file[6][13] =  
{  
    "files/f1.txt",  
    "files/f2.txt",  
    "files/f3.txt",  
    "files/f4.txt",  
    "files/f5.txt",  
    "files/f6.txt"  
};  
   
/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
\*              П Р О Т О Т И П Ы   Ф У Н К Ц И Й                \*  
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  
   
// генерация рандомного вещественного числа  
// в интервале [min, max]****double****rand\_double(****double****min,****double****max);  
   
// упорядочная ф-ия по возрастанию  
template<typename T>****void****f1(T\* arr,****int****size,****double****max,****double****step);  
   
// упорядочная ф-ия по убыванию  
template<typename T>****void****f2(T\* arr,****int****size,****double****max,****double****step);  
   
// случайная ф-ия  
template<typename T>****void****f3(T\* arr,****int****size,****double****max,****double****step);  
   
// синусоидальная функция  
template<typename T>****void****f4(T\* arr,****int****size,****double****max,****double****step);  
   
// пилообразная функция  
template<typename T>****void****f5(T\* arr,****int****size,****double****max,****double****step);  
   
// ступенчатая функция  
template<typename T>****void****f6(T\* arr,****int****size,****double****max,****double****step);  
   
// печать массива в поток  
template<typename T>****void****print\_arr(T\* arr,****int****size,****double****step, ostream& stream);  
   
// ввод и проверка значений  
template<typename T>  
T input\_and\_check(T min, T max,****const******char****\* welcome\_str,****const******char****\* err\_str);  
   
// создание массивов и вывод их в файлы  
template<typename ARRAY\_TYPE, typename TYPE\_OF\_TIME = chrono::nanoseconds>****void****creating\_arrays(****int****size,****double****max,****double****x\_step,****void****(\*stage) (     // функция, которая выводит данные  
        ARRAY\_TYPE\* arr,            // массив****int****size,                   // размер массива****double****x\_step,              // шаг по оси х****int****func\_num,               // номер функции  
        TYPE\_OF\_TIME elapsed\_time   // тип времени (нс или мкС)  
        )  
);  
   
// функция обработки массивов для первого этапа работы  
// она нажна для корректоного вывода данных  
template<typename ARRAY\_TYPE, typename TYPE\_OF\_TIME = chrono::nanoseconds>****void****first\_stage(  
    ARRAY\_TYPE\* arr,            // массив****int****size,                   // размер массива****double****x\_step,              // шаг по оси х****int****func\_num,               // номер функции  
    TYPE\_OF\_TIME elapsed\_time   // тип времени (нс или мкС)  
);  
   
// функция обработки массивов для второго этапа работы  
// она нажна для корректоного вывода данных  
template<typename ARRAY\_TYPE, typename TYPE\_OF\_TIME = chrono::microseconds>****void****second\_stage(  
    ARRAY\_TYPE\* arr,            // массив****int****size,                   // размер массива****double****x\_step,              // шаг по оси х****int****func\_num,               // номер функции  
    TYPE\_OF\_TIME elapsed\_time   // тип времени (нс или мкС)  
);  
   
// решение первой задачи  
// - массивы размером от 150 до 200  
// - отрисовка графиков функций  
// - есть вывод в файл данных массива****void****first\_task();  
   
// вторая задача  
// - массивы размером 5\*10^5,10\*10^5,...,50\*10^5  
// - отрисовки нет  
// - вывода в файл нет  
// - в консоли видно время формирования последовательностей****void****second\_task();  
   
   
/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
\*                Г Л А В Н А Я   Ф У Н К Ц И Я                  \*  
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/****int****main()  
{  
    // установка русского языка в консоли  
    setlocale(LC\_ALL, "ru");  
   
    // инициализация генератора случайных чисел  
    srand(time(NULL));  
   
    // выполнение первого этапа  
    first\_task();  
   
    // выполнение второго этапа  
    second\_task();****return****0;  
}  
   
/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
\*             Р Е А Л И З А Ц И Я   Ф У Н К Ц И Й               \*  
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  
   
// генерация рандомного вещественного числа  
// в интервале [min, max]****double****rand\_double(****double****min,****double****max)  
{****double****f = (****double****)rand() / RAND\_MAX;****return****min + f \* (max - min);  
}  
   
// упорядочная функция по возрастанию  
template<typename T>****void****f1(T\* arr,****int****size,****double****max,****double****step)  
{  
    //k, b - коэффициенты прямой  
    T k = max / (step \* size);  
    T b = 0;****double****x = 0.0; // координата x  
   
    // заполнение массива****for****(****int****i = 0; i < size; i++, x += step)  
    {  
        arr[i] = k \* x + b;  
    }  
}  
   
// упорядочная функция по убыванию  
template<typename T>****void****f2(T\* arr,****int****size,****double****max,****double****step)  
{  
    //k, b - коэффициенты прямой  
    T k = max / (step \* size);  
    T b = 0;****double****x = 0.0; // координата x  
   
    // заполнение массива****for****(****int****i = 0; i < size; i++, x += step)  
    {  
        arr[i] = -k \* x + b;  
    }  
}  
   
// случайная функция  
template<typename T>****void****f3(T\* arr,****int****size,****double****max,****double****step)  
{  
    // заполнение массива  
    // i изменяется в интервале [-max,max]****for****(****int****i = 0; i < size; i++)  
    {  
        arr[i] = rand\_double(-max, max);  
    }  
}  
   
// синусоидальная функция  
template<typename T>****void****f4(T\* arr,****int****size,****double****max,****double****step)  
{  
    T x\_scale = 1;      // масштабирование по x  
    T x\_shift = 1;      // сдвиг по х  
    T y\_scale = max;    // мастабирование по y****double****x = 0.0;     // координата x  
   
    // заполнение массива****for****(****int****i = 0; i < size; i++, x += step)  
    {  
        arr[i] = sin(x\_scale \* x + x\_shift) \* y\_scale;  
    }  
}  
   
// пилообразная функция  
template<typename T>****void****f5(T\* arr,****int****size,****double****max,****double****step)  
{  
    T x\_scale = 1;      // масштабирование по x  
    T x\_shift = 0;      // сдвиг по х  
    T y\_scale = max;    // мастабирование по y****double****x = 0.0;     // координата x  
   
    // заполнение массива****for****(****int****i = 0; i < size; i++, x += step)  
    {  
        arr[i] = asin(sin(x \* x\_scale + x\_shift)) \* y\_scale / M\_PI\_2;  
    }  
}  
   
// ступенчатая функция  
template<typename T>****void****f6(T\* arr,****int****size,****double****max,****double****step)  
{****int****count\_of\_step = 5;                      // количество ступеней****double****step\_height = max / count\_of\_step;   // высота одной ступени****int****step\_length = ceil(size \* step);        // длина одной ступени****double****rand\_value = max / 20.0;             // значение для создания шума****double****x = 0.0;                             // координата x  
   
    // координата x увеличивается на step\_height (высоту шага),  
    // если прошло ровно step\_length значений****for****(****int****i = 0; i < size; i++, x += ((i % step\_length == 0) ? step\_height : 0.0))  
    {  
        // координата y + шум, зависящий от высоты шага  
        arr[i] = x + rand\_double(-rand\_value, rand\_value);  
    }  
}  
   
// печать массива в поток  
template<typename T>****void****print\_arr(T\* arr,****int****size,****double****step, ostream& stream)  
{  
    // вывод шага по x в файл  
    stream << step << endl;  
   
    // вывод координат массива****for****(****int****i = 0; i < size; i++)  
    {  
        stream <<****fixed****<< setprecision(6)  
            << arr[i] << '\n';  
    }  
}  
   
// первая часть****void****first\_task()  
{  
    cout << "========= ПЕРВЫЙ ЭТАП =========\n";  
   
    // тип массива (int или double), вводится через консоль****int****type = input\_and\_check(1, 2,  
        "\nВыберите тип данных массива [1,2]:\n\t1.int\n\t2.double\n",  
        "Должна быть введена 1 или 2\n");  
   
    // размер массива, вводится через консоль****int****size = input\_and\_check(150, 200,  
        "\nВведите размер массива [150,200]:\n",  
        "Должно быть введено значение в интервале [150,200]\n");  
   
    // максимальное значение функции****double****max\_value = input\_and\_check(10, 1000,  
        "\nВведите максимальное значение по оси y [10,1000]:\n",  
        "Должно быть введено значение в интервале [10,1000]\n"  
    );  
   
    // шаг по оси х****double****x\_step = input\_and\_check(0.01, 10.0,  
        "\nВведите величину шага по x [0.01, 10]:\n",  
        "Должно быть введено значение в интервале [0.01, 10] через точку, а не запятую\n"  
    );  
   
    // выбор типа массива****switch****(type)  
    {  
        // если был выбран int****case****1:  
        creating\_arrays<****int****>(size, max\_value, x\_step, first\_stage);****break****;  
   
        // если был выбран double****case****2:  
        creating\_arrays<****double****>(size, max\_value, x\_step, first\_stage);****break****;  
    }  
   
    // отрисовка графиков  
    GraphsDrawer gd;  
    gd.run();  
   
    cout << "== КОНЕЦ ПЕРВОГО ЭТАПА ==\n\n";  
}  
   
// втроая часть****void****second\_task()  
{  
    cout << "========= ВТОРОЙ ЭТАП =========\n";  
   
    // тип массива (int или double), вводится через консоль****int****type = input\_and\_check(1, 2,  
        "\nВыберите тип данных массива [1,2]:\n\t1.int\n\t2.double\n",  
        "Должна быть введена 1 или 2\n");  
   
    // максимальное значение функции****double****max\_value = input\_and\_check(10, 1000,  
        "\nВведите максимальное значение по оси y [10,1000]:\n",  
        "Должно быть введено значение в интервале [10,1000]\n"  
    );  
   
    // шаг по оси х****double****x\_step = input\_and\_check(0.01, 10.0,  
        "\nВведите величину шага по x [0.01, 10]:\n",  
        "Должно быть введено значение в интервале [0.01, 10] через точку, а не запятую\n"  
    );  
   
    // вывод шапки таблицы  
    cout << "Длина последовательности |f1 мкС |f2 мкС |f3 мкС |f4 мкС |f5 мкС |f6 мкС |\n";  
   
    // создание массивов разных размеров****for****(****int****size = 500000; size <= 5000000; size += 500000)  
    {  
        // печать текущего размера массива  
        cout <<****fixed****<< setfill(' ') << setw(25) << size << "|";  
   
        // выбор типа данных массива****switch****(type)  
        {  
            // если был выбран int****case****1:  
            creating\_arrays<****int****, chrono::microseconds>(  
                size, max\_value, x\_step,  
                second\_stage);****break****;  
   
            // если был выбран double****case****2:  
            creating\_arrays<****double****, chrono::microseconds>(  
                size, max\_value, x\_step,  
                second\_stage);****break****;  
        }  
        // перенос строки  
        cout << endl;  
    }  
    cout << "== КОНЕЦ ВТОРОГО ЭТАПА ==\n";  
}  
   
// ввод и проверка значений  
template<typename T>  
T input\_and\_check(T min, T max,****const******char****\* welcome\_str,****const******char****\* err\_str)  
{  
    // размер массива  
    T num;  
   
    // вывод сообщения  
    cout << welcome\_str;  
    cin >> num;  
   
    // если было введено не то****if****(num > max || num < min) {  
        // если была введена не цифра****if****(cin.fail())  
        {  
            cin.clear();  
            cin.ignore(INT\_MAX, '\n');  
        }  
   
        // отчистка консоли  
        system("cls");  
        cout << err\_str;  
   
        // рекурсивное обращение  
        num = input\_and\_check(min, max, welcome\_str, err\_str);  
    }****return****num;  
}  
   
// создание массивов и вывод их в файлы  
template<typename ARRAY\_TYPE, typename TYPE\_OF\_TIME = chrono::nanoseconds>****void****creating\_arrays(****int****size,****double****max,****double****x\_step,****void****(\*stage) (     // функция, которая выводит данные  
        ARRAY\_TYPE\* arr,            // массив****int****size,                   // размер массива****double****x\_step,              // шаг по оси х****int****func\_num,               // номер функции  
        TYPE\_OF\_TIME elapsed\_time   // тип времени (нс или мкС)  
        )  
)  
{  
    // массив  
    ARRAY\_TYPE\* arr =****new****ARRAY\_TYPE[size];  
   
    // массив указателей на функции****void****(\*funcs[])(  
        ARRAY\_TYPE \* arr,       // массив****int****size,               // размер массива****double****max\_y,           // максимальное значение по y****double****x\_step           // шаг по x  
        ) = { f1,f2,f3,f4,f5,f6 };  
   
    // запуск всех функций****for****(****int****i = 0; i < 6; i++)  
    {  
        // начало отсчета времени  
        auto begin = std::chrono::steady\_clock::now();  
   
        // вызов формирующей функции  
        funcs[i](arr, size, max, x\_step);  
   
        // конец отсчета времени  
        auto end = std::chrono::steady\_clock::now();  
   
        // вывод в консоль времени работы программы  
        auto elapsed\_ms = std::chrono::duration\_cast<TYPE\_OF\_TIME>(end - begin);  
   
        // вызов функции  
        stage(arr, size, x\_step, i, elapsed\_ms);  
    }  
   
    // очистка памяти, занимаемой массивом  
    delete[] arr;  
}  
   
// функция обработки массивов для первого этапа работы  
template<typename ARRAY\_TYPE, typename TYPE\_OF\_TIME = chrono::nanoseconds>****void****first\_stage(       // функция, которая выводит данные  
    ARRAY\_TYPE\* arr,             // массив****int****size,                    // размер массива****double****x\_step,               // шаг по оси х****int****func\_num,                // номер функции  
    TYPE\_OF\_TIME elapsed\_time    // тип времени (нс или мкС)  
)  
{  
    // печать времени формирования последовательности  
    cout << "\tВремя выполнения " << func\_num + 1 << " функции = " <<****fixed****<< setw(5) << elapsed\_time.count() << " (нс)\n";  
   
    // создание определенного файлового потока  
    ofstream****out****(name\_of\_file[func\_num]);  
   
    // печать массива в файл  
    print\_arr(arr, size, x\_step,****out****);  
   
    // закрытие файлового потока****out****.close();  
   
}  
   
// функция обработки массивов для второго этапа работы  
template<typename ARRAY\_TYPE, typename TYPE\_OF\_TIME = chrono::microseconds>****void****second\_stage(      // функция, которая выводит данные  
    ARRAY\_TYPE\* arr,               // массив****int****size,                      // размер массива****double****x\_step,                 // шаг по оси х****int****func\_num,                  // номер функции  
    TYPE\_OF\_TIME elapsed\_time      // тип времени (нс или мкС)  
)  
{  
    // печать времени формирования последовательности  
    cout <<****fixed****<< setfill(' ') << setw(6) << elapsed\_time.count() << " |";  
}  
/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* End Of main.cpp File \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/*

# Графики функций

# Результаты по временным интервалам с различными размерами массивов

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Количество элементов | Возрастание (мкС) | Убывание(мкС) | Случайная(мкС) | Синусоидальная(мкС) | Пилообразная(мкС) | Ступенчатая(мкС) |
| 500000 | 1150 | 1160 | 20790 | 3799 | 13818 | 20964 |
| 1000000 | 2287 | 2431 | 42542 | 7646 | 26009 | 41599 |
| 1500000 | 3877 | 3616 | 62616 | 12197 | 39852 | 63488 |
| 2000000 | 4664 | 5449 | 86555 | 15446 | 52938 | 86882 |
| 2500000 | 5914 | 5865 | 101866 | 19930 | 71615 | 105312 |
| 3000000 | 7082 | 7059 | 128830 | 23222 | 80213 | 124977 |
| 3500000 | 8118 | 8740 | 147303 | 26994 | 92225 | 146000 |
| 4000000 | 9969 | 9750 | 166039 | 31904 | 106449 | 167247 |
| 4500000 | 10607 | 10924 | 191574 | 35080 | 118704 | 189237 |
| 5000000 | 11893 | 11817 | 206258 | 38313 | 134314 | 211274 |

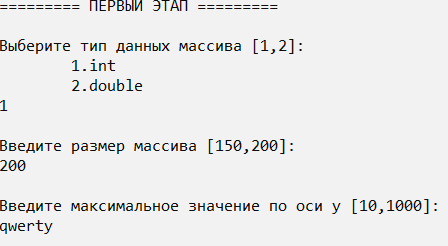
# Тестирование программы

## Тестирование некорректных режимов

### Тест 1

**Цель:** Проверить работу программы при введении некорректных данных

**Исходные данные:** 1 200 qwerty



**Ожидаемый Результат:** Вывод сообщения об ошибке:

Должно быть введено значение в интервале [10,1000].

**Полученный результат:**



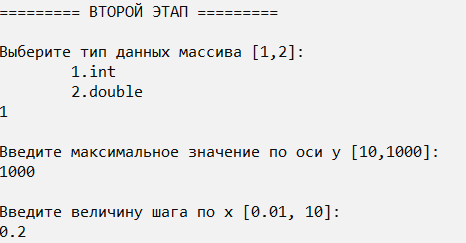
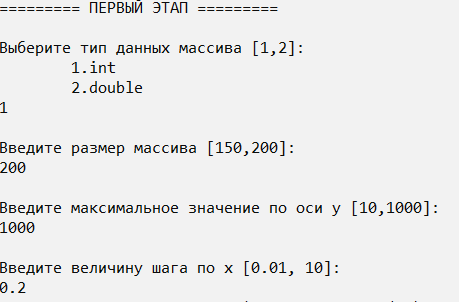
**Вывод по тесту:** Ожидаемый результат совпал с полученным результатом. Тест ошибок не обнаружил.

## Тестирование корректных режимов

### Тест 1

**Цель:** Проверить работу программы при выполнении всех условий задачи.

**Исходные данные:** 1 200 1000 0.2



**Ожидаемый Результат:**

Время выполнения 1 функции = 3000 (нс)

Время выполнения 2 функции = 600 (нс)

Время выполнения 3 функции = 8600 (нс)

Время выполнения 4 функции = 15600 (нс)

Время выполнения 5 функции = 14400 (нс)

Время выполнения 6 функции = 13400 (нс)

Длина последовательности |f1 мкС |f2 мкС |f3 мкС |f4 мкС |f5 мкС |f6 мкС |

500000| 1084 | 1139 | 20470 | 6346 | 13380 | 20500 |

1000000| 2164 | 2264 | 42607 | 12734 | 26839 | 41261 |

1500000| 3223 | 3563 | 62023 | 18923 | 41698 | 63468 |

2000000| 4603 | 4773 | 81696 | 25520 | 54821 | 82164 |

2500000| 5439 | 5696 |106791 | 31774 | 68604 |104894 |

3000000| 6526 | 7123 |128728 | 38289 | 87759 |128972 |

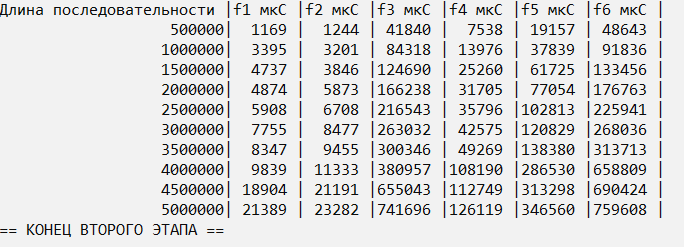
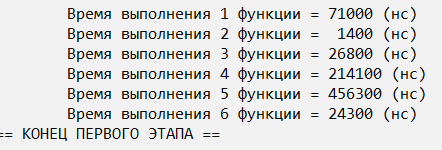
3500000| 7951 | 7924 |148033 | 44747 | 96866 |146140 |

4000000| 8813 | 9271 |163997 | 49858 |108219 |167511 |

4500000| 9957 | 10903 |186861 | 57289 |121730 |185694 |

5000000| 10926 | 11407 |205685 | 63566 |134516 |208095 |

**Полученный результат:**



**Вывод по тесту:** Ожидаемый результат совпал с полученным результатом. Тест ошибок не обнаружил.

# Вывод

Работа программы завершена на основании:

1) Полученные результаты совпали с ожидаемыми;

2) Считаем набор тестов полным.