**Лабораторная работа №1. Вспомогательные функции**

**Цель работы:** приобретение навыков составления и отладки программ с использованием пользовательских функций для замера продолжительности процесса вычисления.

**Ход работы**

1. **Написание кода программы.**

Основной код программы приведен в листинге 1.1. В данном листинге приведена реализация всех вспомогательных функций: генерация случайных чисел, получение случайного числа из промежутка, функция для чисел Фибоначчи.

Листинг 1.1

#include "stdafx.h"

#define CYCLE 1000000

int main(int argc, char\* argv[])

{

double av1 = 0, av2 = 0;clock\_t t1 = 0, t2 = 0;

setlocale(LC\_ALL, "rus");auxil::start();t1 = clock();

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

av1 += (double)auxil::iget(-100, 100);

av2 += auxil::dget(-100, 100);}

t2 = clock();

std::cout << std::endl << "количество циклов: " << CYCLE;

std::cout << std::endl << "среднее значение (int): " << av1 / CYCLE;

std::cout << std::endl << "среднее значение (double): " << av2 / CYCLE;

std::cout << std::endl << "продолжительность (у.е): " << (t2 - t1);

std::cout << std::endl << " (сек): "

<< ((double)(t2 - t1)) / ((double)CLOCKS\_PER\_SEC);

std::cout << std::endl;

system("pause");

std::cout << "\n-------Функция чисел фибоначи-------\n";

clock\_t t3 = 0, t4 = 0;int n;

std::cout << "Введите число n: ";std::cin >> n;

t3 = clock();

long double result = fibonachi(n);

std::cout << std::endl << n << " число Фиббоначи = " << result;

t4 = clock();

std::cout << std::endl << "продолжительность (у.е): " << (t4 - t3);

std::cout << std::endl << " (сек): "

<< ((double)(t4 - t3)) / ((double)CLOCKS\_PER\_SEC);

std::cout << std::endl;

system("pause");

return 0;

}

В листинге 1.2 представлена реализация функций генераций случайных чисел.

Листинг 1.2

#include "stdafx.h"

namespace auxil

{

//старт генератора чисел

void start()

{

srand((unsigned)time(NULL));

};

//получить случайное число

double dget(double rmin, double rmax)

{

return ((double)rand() / (double)RAND\_MAX) \* (rmax - rmin) + rmin;

};

//получить случайное число

int iget(int rmin, int rmax)

{

return (int)dget((double)rmin, (double)rmax);

};

}

#pragma once

#include <cstdlib>

namespace auxil {

void start(); //старт генератора сл. чисел

double dget(double rmin, double rmax); // получить случайное число

int iget(int rmin, int rmax); //получить случайное число

};

Реализация для функции числе Фибоначчи представлена в листинге 1.3.

Листинг 1.3

#include "Fibonachi.h"

long double fibonachi(int n)

{

if (n == 0)

return 0;

if (n == 1)

return 1;

return fibonachi(n - 1) + fibonachi(n - 2);

}

#pragma once

long double fibonachi(int n);

Все подключенные библиотеки и заголовочные файлы, используемые в программе описаны в файле “stdafx.h”. Содержимое данного файла представлено в листинге 1.4

Листинг 1.4

#pragma once

#include "Auxil.h"

#include "Fibonachi.h"

#include <iostream>

#include <locale>

#include <ctime>

Результат работы данной программы представлен на рисунке 1.5

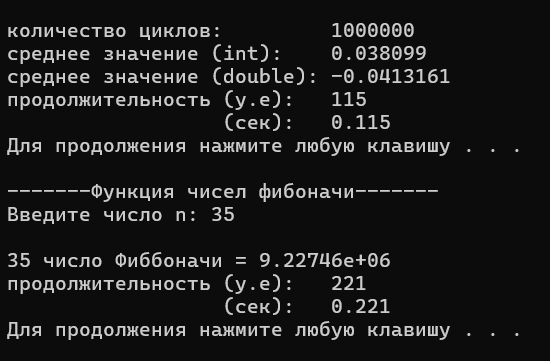


Рисунок 1.5 – результат работы программы

1. **Графики зависимости**

График зависимости (Excel) продолжительности процесса вычисления от количества циклов в первом примере с генерацией случайного числа и получением случайного числа из диапазона чисел приведен на рисунке 2.1.

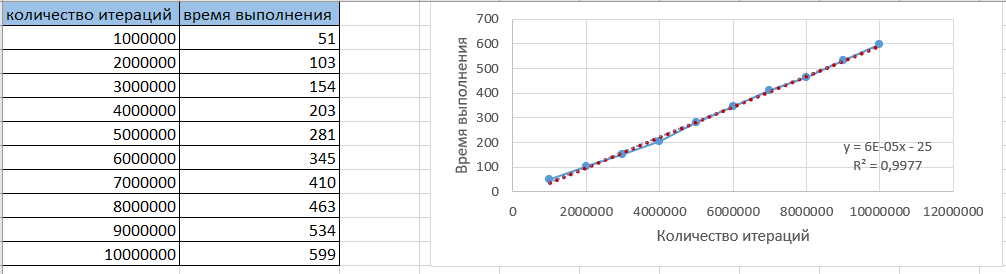


Рисунок 2.1 – график зависимости для первого примера

Из этого можем сделать вывод, что данный график близок к графику линейной зависимости, так как корреляционный момент почти равен единице.

График зависимости (Excel) продолжительности процесса вычисления от количества циклов во втором примере с числами Фибоначчи приведен на рисунке 2.2.

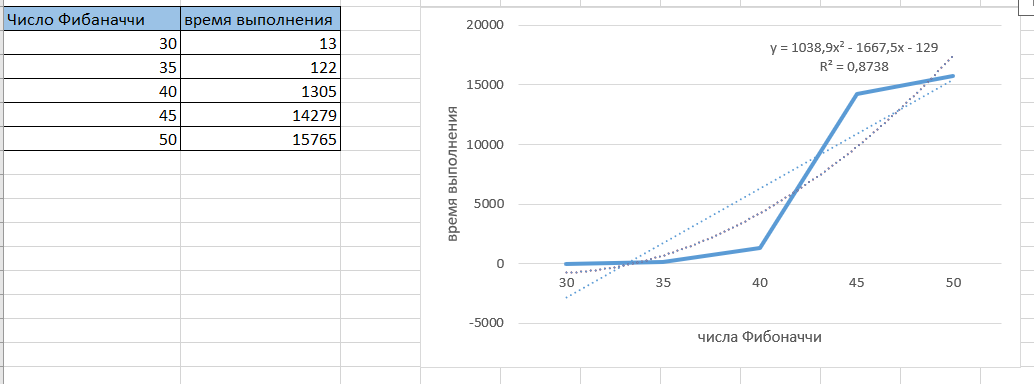


Рисунок 2.2 – график зависимости для второго примера

Из этого можем сделать вывод, что данный график близок к графику полиномиальной зависимости, так как корреляционный момент приближен к единице.