Белорусский государственный технологический университет

Факультет информационных технологий

Кафедра программной инженерии

 Лабораторная работа № 1

По дисциплине «Математическое программирование»

На тему «Вспомогательные функции»

Выполнила:

Студентка 2 курса 10 группы

Рублевская Маргарита Владимировна

Преподаватель: асс. Ромыш А.С.

2025, Минск

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

**Ход работ**

1. **Написание кода программы (выполнение заданий № 1-2)**

Код файла main.cpp приведен в листинге 1.1

#include "stdafx.h"

#include "Auxil.h" // вспомогательные функции

#include <iostream>

#include <ctime>

#include <locale>

#define CYCLE 1000000 // количество циклов

int Fib(int i)

{

if (i < 1)

return 0;

if (i == 1)

return 1;

return Fib(i - 1) + Fib(i - 2);

}

int \_tmain(int argc, \_TCHAR\* argv[])

{

double av1 = 0, av2 = 0;

clock\_t t1 = 0, t2 = 0;

setlocale(LC\_ALL, "rus");

auxil::start(); // старт генерации

t1 = clock(); // фиксация времени

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

{

av1 += (double)auxil::iget(-100, 100); // сумма случайных чисел

av2 += auxil::dget(-100, 100); // сумма случайных чисел

}

t2 = clock(); // фиксация времени

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;

for (int n = 22; n < 43; n++)

{

t1 = clock();

int num = Fib(n);

t2 = clock();

std::cout << std::endl << n << "-е число Фибоначчи " << (t2 - t1) << " у.е.";

}

std::cout << std::endl;

system("pause");

return 0;

}

Листинг 1.1. — Содержание файла main.cpp

Код файла Auxil.h приведен в листинге 1.2.

#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

};

Листинг 1.2. — Содержание файла main.cpp

Код файла Auxil.cpp приведен в листинге 1.3.

#include "stdafx.h"

#include "Auxil.h"

#include <ctime>

namespace auxil

{

void start() // старт генератора сл. чисел

{

srand((unsigned)time(NULL));

};

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

{

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

};

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

{

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

};

}

Листинг 1.3. — Содержание файла main.cpp

Результат работы программы представлен на рисунке 1.1.

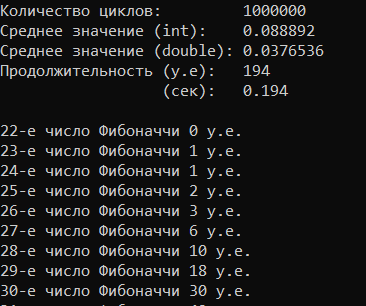
****

Рисунок 1.1 — Результаты работы программы

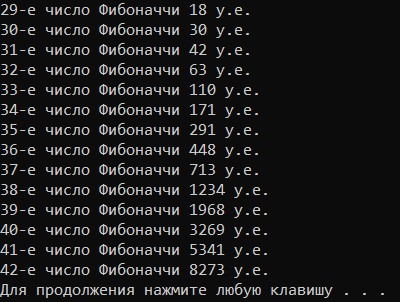
****

Рисунок 1.1 (продолжение) — Результаты работы программы

1. **Выполнение задания № 3**
2. Проведите необходимые эксперименты и постройте график зависимости (Excel) продолжительности процесса вычисления от количества циклов в примере 2.

Сначала найдем зависимость продолжительности вычислений от циклов. Результаты измерений и соответствующий график, представлены на рисунке 1.2.

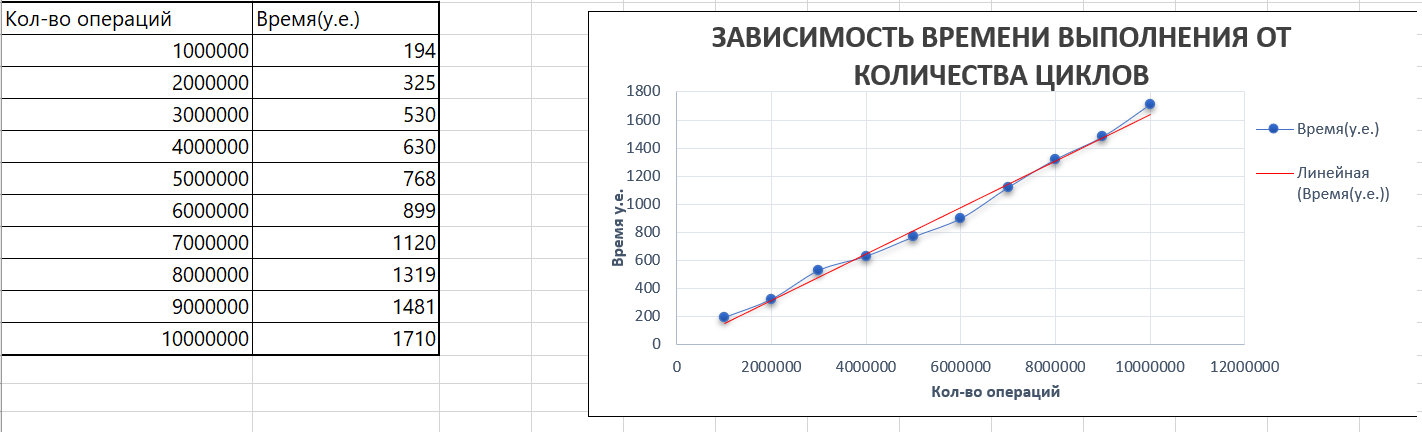
****

Рисунок 1.2 — Результаты измерений

Как мы видим, зависимость у нас линейная, так как кол-во операций прямо пропорционально числу циклов. Вывод: скорость выполнения программы линейно зависит от количества итераций цикла.

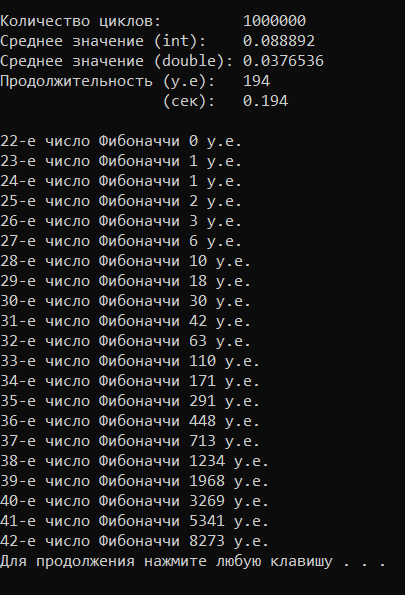
1. Проанализируйте характер зависимости. Проведите исследование любого другого рекурсивного алгоритма, например, вычисления факториала или генератора чисел Фибоначчи (прим. – например вычислите каким будет 100-е, 200-е, 300-е и т.д число), и включите в отчет график.

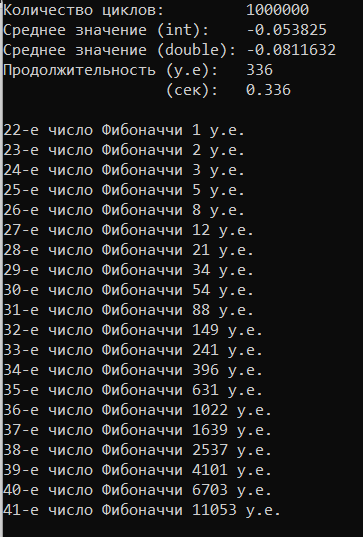
Результаты измерений и соответствующий график, представлены на рисунке 1.3.

****

Рисунок 1.3 — Результаты измерений

Вывод: скорость выполнения программы экспоненциально зависит от порядкового номера числа Фибоначчи.

****

****