**Лабораторная работа 1. Вспомогательные функции**

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

**ВЫПОЛНЕНИЕ РАБОТЫ:** составить и реализовать программы.

***Задание 1. Разработка кода.***

#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

};

Рисунок 1.1 Содержание файла Auxil.h

#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);

};

}

Рисунок 1.2 Содержание файла Auxil.cpp

#include "stdafx.h"

#include "Auxil.h" // вспомогательные функции

#include <iostream>

#include <ctime>

#include <locale>

#define CYCLE 10000000

int fibonacci(int i)

{

int value = 0;

if (i < 1) {

return 0;

}

if (i == 1) {

return 1;

}

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

}

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

{

double av1 = 0, av2 = 0;

clock\_t startTime = 0, endTime = 0;

setlocale(LC\_ALL, "rus");

auxil::start(); // старт генерации

startTime = 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

}

endTime = 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): " << (endTime - startTime);

std::cout << std::endl << " (сек): "

<< ((double)(endTime - startTime)) / ((double)CLOCKS\_PER\_SEC);

std::cout << std::endl;

for (int n = 1; n <= 50; n++)

{

startTime = clock();

int num = fibonacci(n);

endTime = clock();

std::cout << std:: endl << n << "-е число Фибоначчи: " << num << ". Время на нахождение числа: " /\*<< ((double)(endTime - startTime)) / ((double)CLOCKS\_PER\_SEC);\*/ << (endTime - startTime);

}

std:: cout << std::endl;

system("pause");

return 0;}

Рисунок 1.3 Содержание файла main.cpp

***Задание 2. Проверки работоспособности разработанных функций и приобретения навыков замера продолжительности процесса***

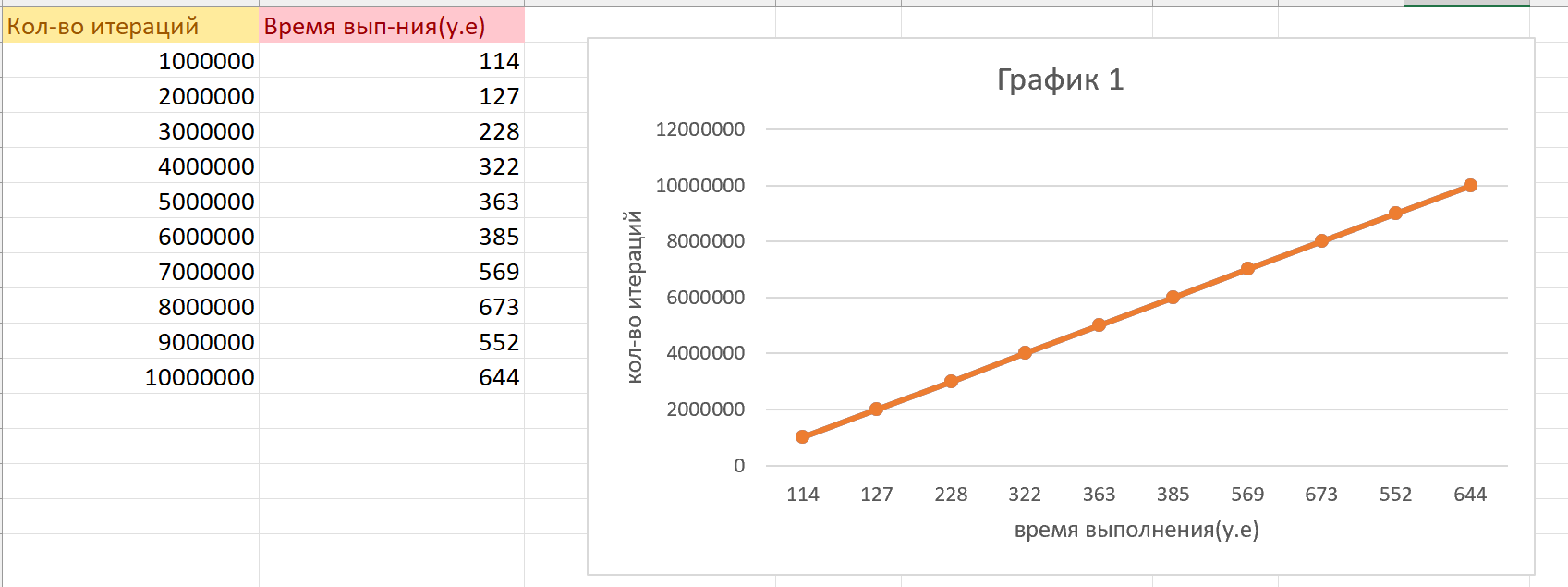


Рисунок 2.1 Результаты выполнения программы и график

Вывод: время выполнения программы линейно зависит от количества итераций в цикле.

***Задание 3 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.***

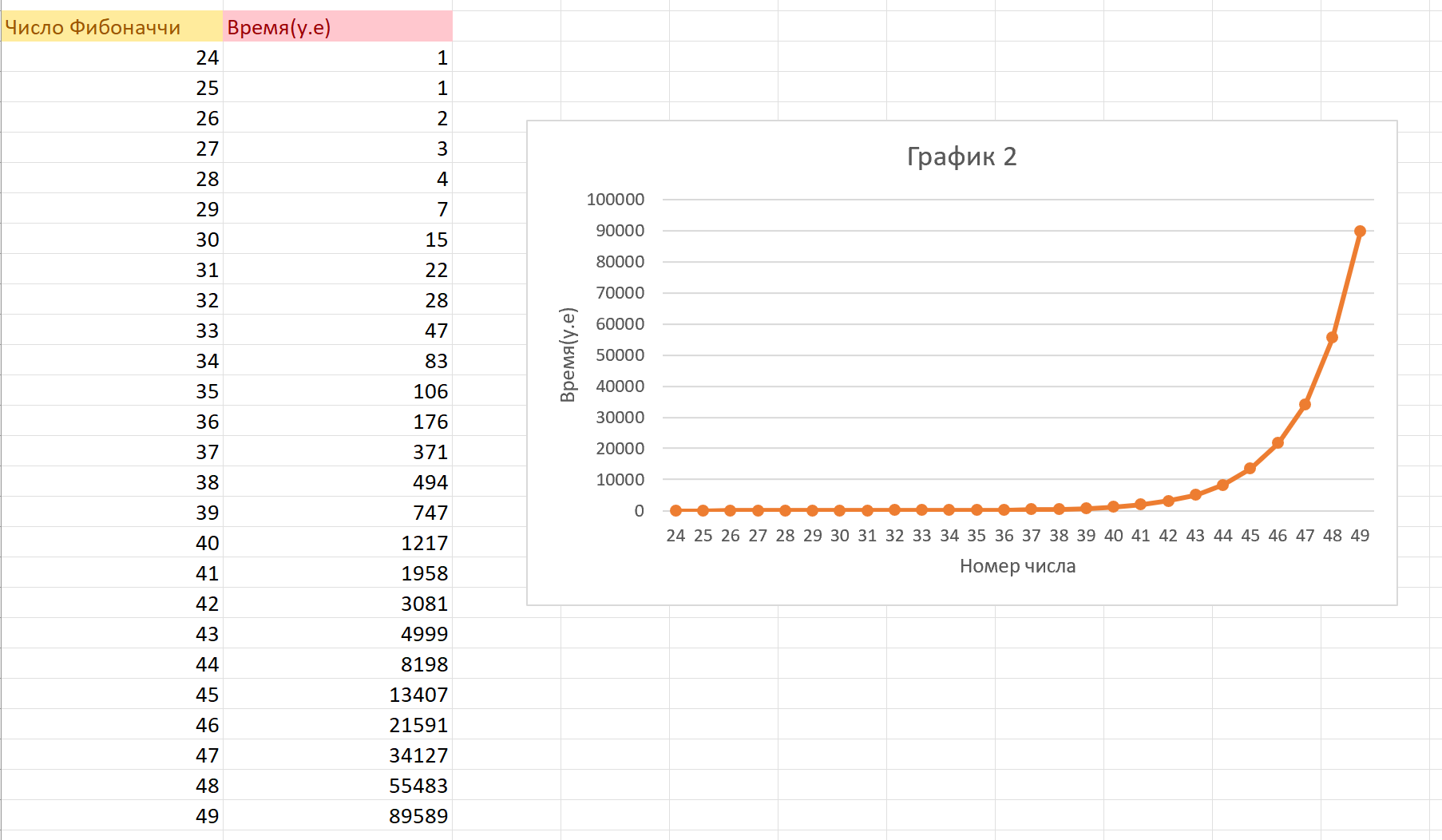
******

Рисунок 3.1 Результаты выполнения программы и график

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