**Лабораторная работа №1. Вспомогательные функции**

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

**Ход работы**

## **Написание кода программы**

Код файла Main.cpp представлен на рисунке 1.1.

|  |
| --- |
| #include "pch.h"  #include "Auxil.h"  #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 firstAverage = 0, secondAverage = 0;  clock\_t startTime = 0, endTime = 0;  setlocale(LC\_ALL, "rus");  Auxil::start();  startTime = clock();  for (int i = 0; i < CYCLE; i++)  {  firstAverage += (double)Auxil::iget(-100, 100);  secondAverage += Auxil::dget(-100, 100);  }  endTime = clock();  cout << endl << "\nКоличество циклов: " << CYCLE;  cout << endl << "Среднее значение (int): " << firstAverage / CYCLE;  cout << endl << "Среднее значение (double): " << secondAverage / CYCLE;  cout << endl << "Продолжительность (CLOCKS\_PER\_SEC): " << (endTime - startTime);  cout << endl << "Продолжительность (сек): " << ((double)(endTime - startTime)) / ((double)CLOCKS\_PER\_SEC) << "\n";  for (int n = 1; n < 51; n++)  {  startTime = clock();  int num = Fibonacci(n);  endTime = clock();  cout << endl << n << "-е число Фибоначчи: " << num << ". Время на нахождение числа: " << (endTime - startTime) << " CLOCKS\_PER\_SEC";  }  cout << endl;  system("pause");  return 0;  } |

Рисунок 1.1 Содержание файла Main.cpp

Код файла Auxil.h представлен на рисунке 1.2.

|  |
| --- |
| #pragma once  #include <cstdlib>  namespace Auxil {  void start();  double dget(double rmin, double rmax);  int iget(int rmin, int rmax);  } |

Рисунок 1.2. Содержание файла Auxil.h

Код файла Auxil.cpp представлен на рисунке 1.3.

|  |
| --- |
| #include "pch.h"  #include "Auxil.h"  namespace Auxil  {  void start()  {  srand((unsigned)time(NULL));  };  double dget(double rmin, double rmax)  {  return ((double)rand() / (double)RAND\_MAX) \* (rmax - rmin) + rmin;  };  int iget(int rmin, int rmax)  {  return (int)dget((double)rmin, (double)rmax);  };  } |

Рисунок 1.3. Содержание файла Auxil.cpp

Результат работы программы представлен на рисунке 1.4.

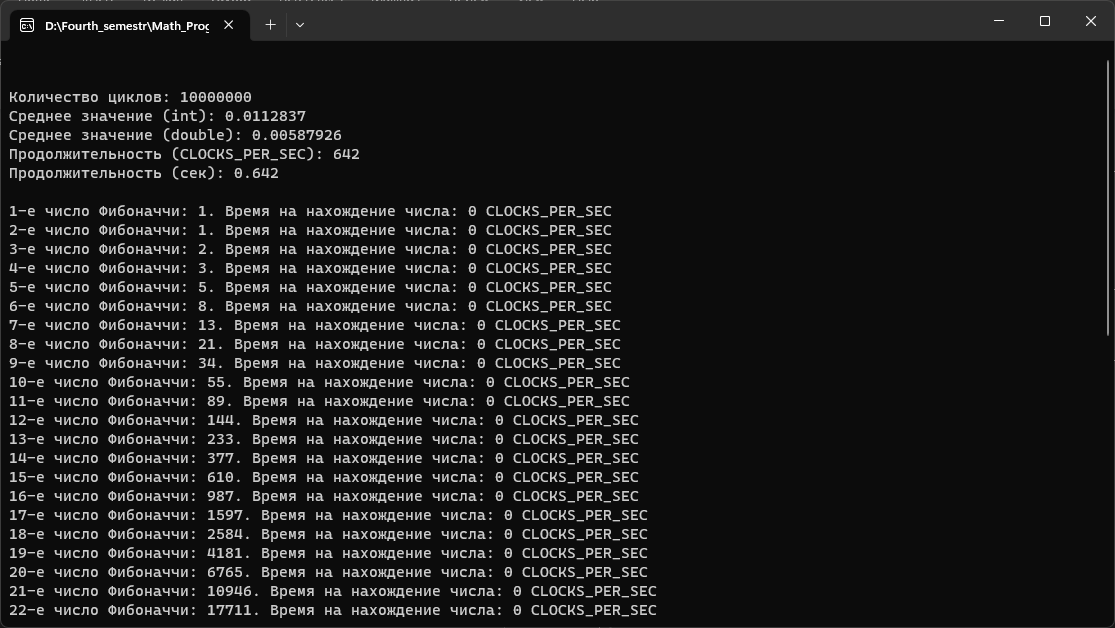


Рисунок 1.4. Результат работы программы

## **2. Измерение скорости выполнения функции генерации случайных чисел.**

Результаты измерений и соотвествующий график представлен на рисунке 2.1.

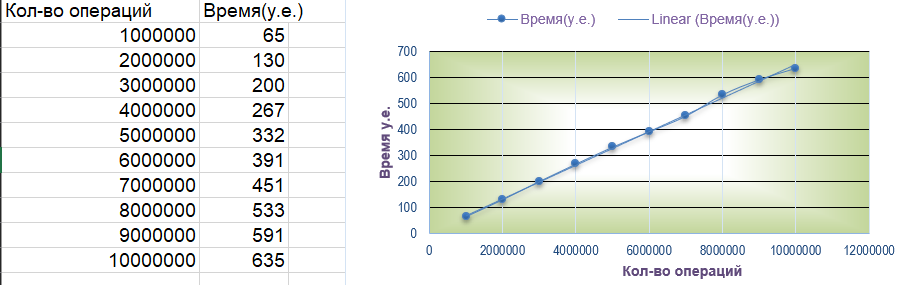


Рисунок 2.1. Результаты измерений и их график

Вывод: скорость выполнения программы линейно зависит от количества итераций цикла.

## **3. Измерение скорости выполнения функции подсчёта n-ого числа Фибоначчи.**

Результаты измерений и соответствующий график приведены на рисунке 3.1.

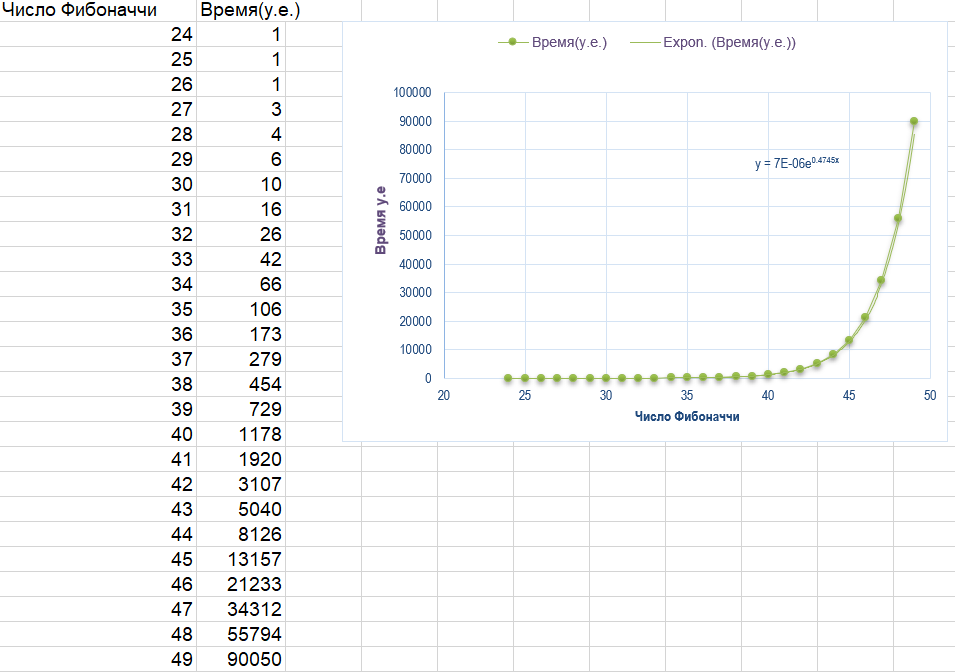


Рисунок 3.1. Результаты измерений и их график

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