Белорусский государственный технологический университет

Факультет информационных технологий

Кафедра программной инженерии

Лабораторная работа 1

По дисциплине «Математическое программирование»

На тему «Сбор требований»

Выполнила:

Студентка 2 курса 9 группы

Волосюк Арина Вадимовна

Преподаватель: Ромыш А.С.

2025, Минск

**Лабораторная работа №1. Вспомогательные функции**

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

**Ход работы**

**Задание 1.**Разработайте три функции (start, dget и iget), используя следующие спецификации.

Реализация функций показана в листинге 1, 2.

Листинг 1 – Код файла Auxil.cpp

|  |
| --- |
| #include "pch.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 2 – Прототипы функций Auxil.h

|  |
| --- |
| #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  }; |

**Задание 2.**

1. Реализовать пример 2.
2. Для проверки работоспособности разработанных функций и приобретения навыков замера продолжительности процесса вычисления реализуйте программу, приведенную в примере 2.

Задание под номером 2 показано в листинге 3. Листинг 3 – вызов функции и измерение времени выполнения Lab\_01.cpp

|  |
| --- |
| #include "pch.h"  #include "Auxil.h" // вспомогательные функции  #include <iostream>  #include <ctime>  #include <locale>  #define CYCLE 1000000 // количество циклов  int fibonacci (int i)  {  if (i < 1)  return 0;  if (i == 1)  return 1;  return Fib(i - 1) + Fib(i - 2);  }  int main()  {  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 = fibonacci(n);  t2 = clock();  std::cout << std::endl << n << "-е число Фибоначчи " << (t2 - t1) << " у.е.";  }  std::cout << std::endl;  system("pause");  return 0;  } |

Результат работы программы представлен на рисунке 1.1.

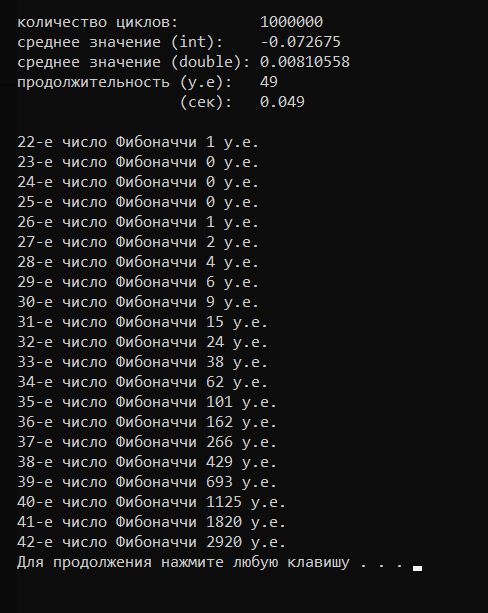


Рисунок 1.1 Результаты работы программы

**Задание 3.**

Проведите необходимые эксперименты и постройте график зависимости (Excel) продолжительности процесса вычисления от количества циклов в примере 2.

Измерение скорости выполнения функции генерации случайных чисел. Результаты измерений и соответствующий график приведены на рисунке 1.2.

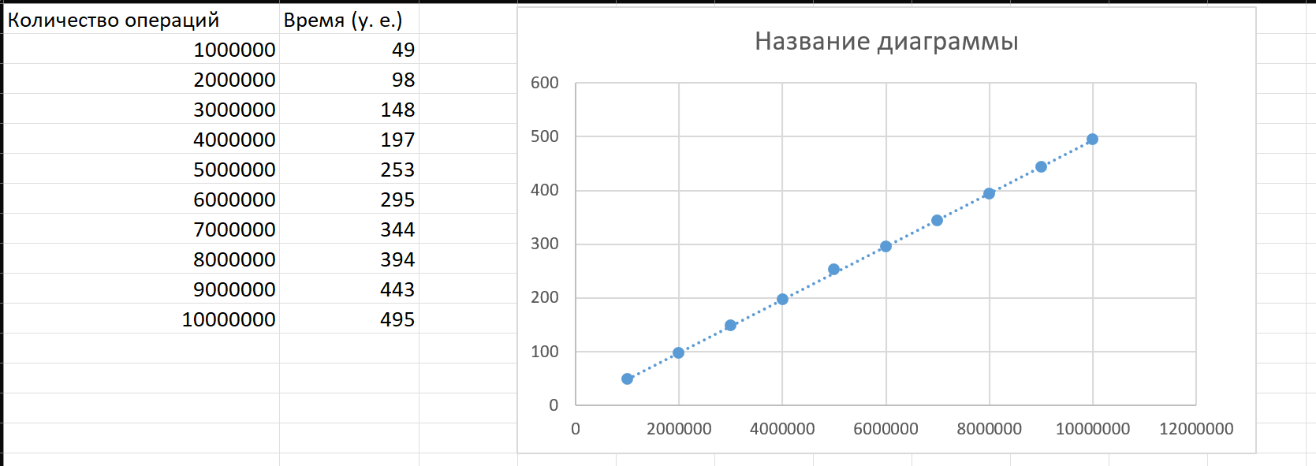


Рисунок 1.2 Результаты измерений и их график

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

Измерение скорости выполнения функции подсчета n-го числа Фибоначчи.

Результаты измерений и соответствующий график приведены на рисунке 1.3.

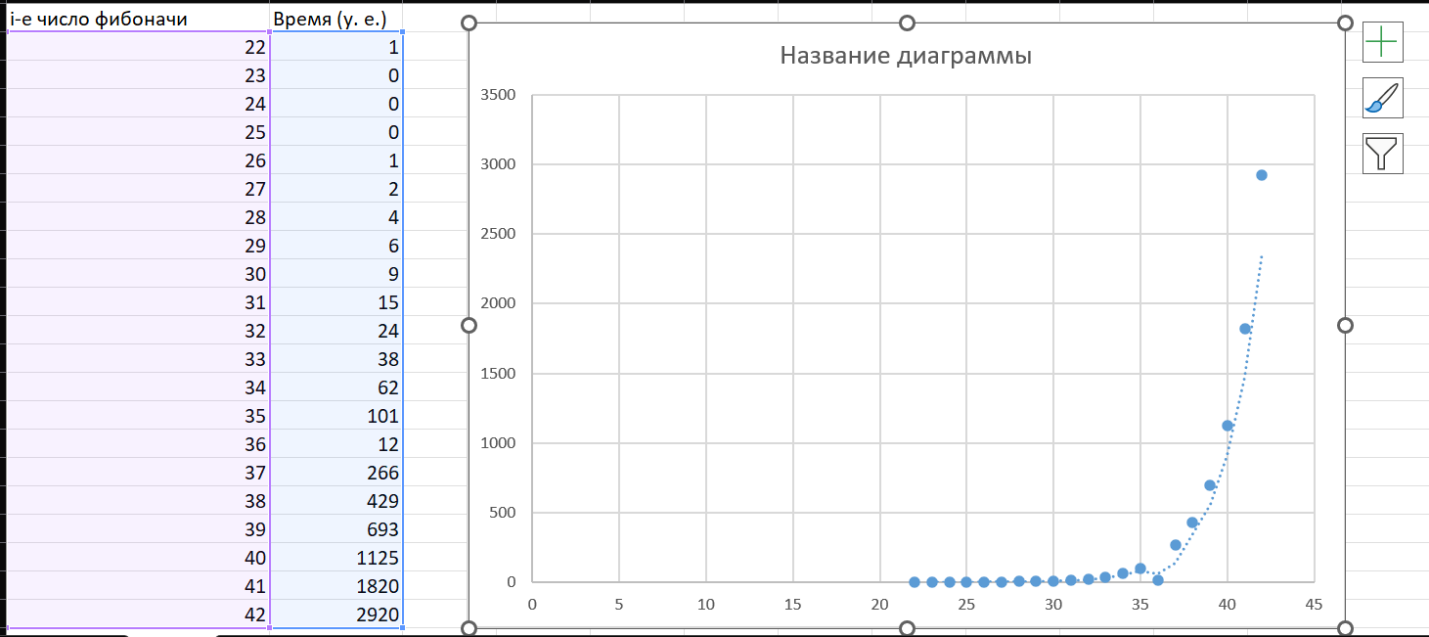


Рисунок 1.3. Результаты измерений и их график

**Вывод: Скорость выполнения программы возрастает экспоненциально с увеличением порядкового номера числа Фибоначчи. Такая зависимость обусловлена особенностями алгоритма вычисления последовательности Фибоначчи.**

**Вывод: В ходе лабораторной работы №1 были освоены навыки разработки и отладки программ с применением пользовательских функций для измерения времени выполнения вычислений. Такой подход позволяет создавать более эффективные и оптимизированные программы, ускоряя вычисления и снижая временные затраты. При разработке функций для замера времени выполнения важно учитывать специфику задачи и требования программы: одни процессы требуют высокой точности измерений из-за их быстрого выполнения, тогда как другие, более длительные, допускают менее точную оценку.**