Учреждение образования

«БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Математическое программирование

Лабораторная работа № 1

Вспомогательные функции

Студент: Жамойдо А. И.

ФИТ 2 курс 10 группа

Преподаватель: Ромыш А. С.

Минск 2025

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

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

|  |
| --- |
| #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 1.1 — Файл Auxil.cpp

|  |
| --- |
| #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 — Файл Auxil.h

***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.

|  |
| --- |
| #include "stdafx.h"  #include "Auxil.h" // Вспомогательные функции  #include "Fibonachi.h"  #include <iostream>  #include <ctime>  #include <locale>  #define CYCLE 1000000 // Количество циклов  using namespace std;  using namespace auxil;  int main(int argc, char\* argv[])  {  double av1 = 0, av2 = 0;  clock\_t t1 = 0, t2 = 0;  setlocale(0, "RU");  start(); // Старт генерации  t1 = clock(); // Фиксация времени  for (int i = 0; i < CYCLE; i++)  {  av1 += (double)iget(-100, 100); // Сумма случайных чисел  av2 += dget(-100, 100); // Сумма случайных чисел  }  t2 = clock(); // Фиксация времени  cout << "\nКоличество циклов: " << CYCLE;  cout << "\nСреднее значение (int): " << av1 / CYCLE;  cout << "\nСреднее значение (double): " << av2 / CYCLE;  cout << "\nПродолжительность (у.е): " << (t2 - t1);  cout << "\n (сек): "  << ((double)(t2 - t1)) / ((double)CLOCKS\_PER\_SEC);  cout << endl;  system("pause");  cout << "\nФункция чисел Фиббоначи";  clock\_t t3 = 0, t4 = 0;  int a;  cout << "\nВведите N-ое число: ";  cin >> a;  t3 = clock();  long double result = fibonachi(a);  cout << endl << a << " число Фиббоначи = " << result;  t4 = clock();  cout << "\nПродолжительность (у.е): " << (t4 - t3);  cout << "\n (сек): "  << ((double)(t4 - t3)) / ((double)CLOCKS\_PER\_SEC);  cout << endl;  system("pause");  return 0;  } |

Листинг 1.3 — Пример 2

***Задание 3***

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



Рисунок 1.1 — График зависимости продолжительности процесса вычисления от количества циклов

***Числа Фибоначчи:***

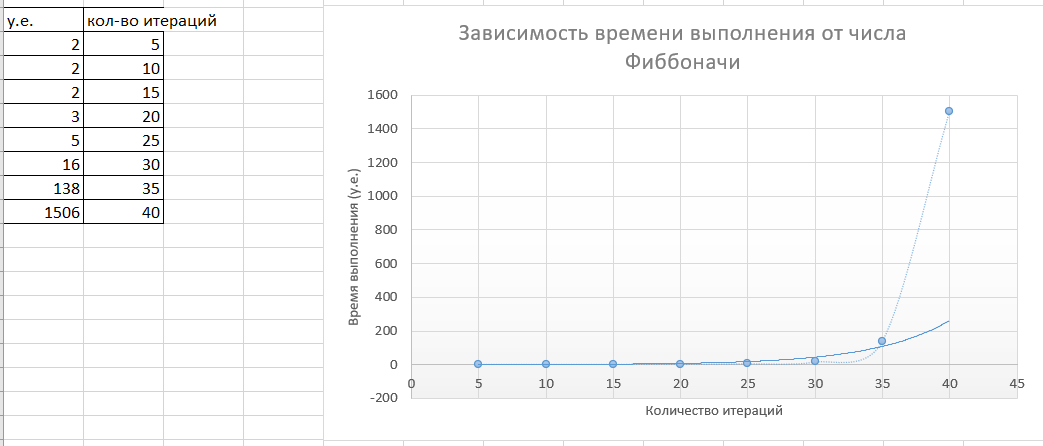
Последовательность чисел Фибоначчи определяется формулой Fn = Fn-1 + Fn-2. То есть, следующее число получается как сумма двух предыдущих.

Первые два числа равны 1, затем 2(1+1), затем 3(1+2), 5(2+3) и так далее: 1, 1, 2, 3, 5, 8, 13, 21....

Реализацию данного алгоритма и график зависимости времени выполнения от заданного числа можно посмотреть в листинге 1.3 и на рисунке 1.4, соответственно.

|  |
| --- |
| #include "Fibonachi.h"  #include <iostream>  long double fibonachi(int n)  {  if (n == 0)  return 0;  if (n == 1)  return 1;  return fibonachi(n - 1) + fibonachi(n - 2);  } |

Листинг 1.3 — Реализация алгоритма чисел Фибоначчи

Рисунок 1.4 — График зависимости продолжительности процесса вычисления от количества циклов

Вывод: скорость выполнения программы линейно зависит от количества итераций цикла. Скорость нахождения N-го числа Фибоначчи имеет вид экспоненциальной зависимости.