УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ «БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет информационных технологий

Кафедра программной инженерии

**Отчёт по лабораторной работе №1**

**по дисциплине «Математическое программирование»**

**на тему «Вспомогательные функции»**

Выполнил:

Студент 2 курса 8 группы

Семёнов Даниил Вячеславович

Преподаватель: асс. Ромыш А.С.

Минск

2025

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

Перед выполнением заданий следует подключить предварительно скомпилированный заголовок stdafx.h. Для использования предварительно скомпилированных заголовков их необходимо включить параметром /Y или через страницу свойств проекта **Configuration Properties > C/C++ > Precompiled Headers**. После этого в каждом cpp-файле в самом начале (до любых директив препроцессора или строк кода) нужно добавить #include "stdafx.h".

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

**void start(); -** устанавливает начальные числа для генератора псевдослучайных чисел.

**double dget(**

**double rmin,** //[in]минимальное значение

**double rmax** //[in]максимальное значение

**); -** генерирует действительное псевдослучайное число в заданном диапазоне.

**int iget(**

**int rmin,** //[in]минимальное значение

**int rmax** //[in]максимальное значение

**); -** генерирует целое псевдослучайное число в заданном диапазоне.

Кроме того, по заданию разработанные функции должны располагаться в файле **Auxil.cpp**,а в файле **Auxil.h –** прототипы функций. Код из Auxil.cpp можно увидеть в листинге 1.1.

|  |
| --- |
| #include "stdafx.h"  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

Код из Auxil.h представлен 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 – 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.

Реализация кода представлена в листинге 2.1.

|  |
| --- |
| #include "stdafx.h"  #define CYCLE 1000000 // количество циклов  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;  system("pause");  return 0;  } |

Листинг 2.1 – Файл main.cpp

Результат выполнения программы представлен на рисунке 1.

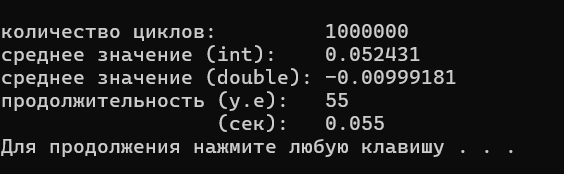


Рисунок 1 – Результат работы программы

**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. Кроме этого продолжительность вычисления следует измерять в условных единицах процессорного времени (функция **clock**).

Перед выполнением следует найти зависимость продолжительности вычислений от количества циклов. Для этого необходимо изменять количество циклов и запоминать время выполнения при этих значениях. Диапазон циклов выбираем от 100000 до 1000000. На рисунке 2 можно увидеть результат вычислений.

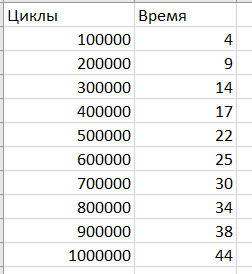


Рисунок 2 – Результаты тестирования №1

На рисунке 3 представлен график зависимости.



Рисунок 3 – График зависимости времени от цикла

Исходя из результатов, мы видим, что зависимость линейная. Теперь проведём исследование с числами Фибоначчи. Перед этим следует реализовать программу.

|  |
| --- |
| #pragma once  int fibonachi(int n); |

Листинг 3.1 – Файл Fibonachi.h

Реализация кода Fibonachi.cpp представлена в листинге 3.2.

|  |
| --- |
| #include "stdafx.h"  using namespace std;  int fibonachi(int n)  {  if (n == 0)  return 0;  if (n == 1)  return 1;  return fibonachi(n - 1) + fibonachi(n - 2);  } |

Листинг 3.1 – Файл Fibonachi.cpp

Обновлённый код main.cpp представлен в листинге 3.3.

|  |
| --- |
| #include "stdafx.h"  #define CYCLE 1000000 // количество циклов  using namespace std;  int main()  {  setlocale(LC\_ALL, "rus");  cout << "\t\tЧисла Фибоначчи" << endl;  int N;  clock\_t t3 = 0, t4 = 0;  cout << "Введите количество чисел N: ";  cin >> N;  t3 = clock();  int result = fibonachi(N);  cout << "Результат функции чисел Фибоначчи: " << result;  t4 = clock();  std::cout << std::endl << "продолжительность (у.е): " << (t4 - t3);  std::cout << std::endl << " (сек): "  << ((double)(t4 - t3)) / ((double)CLOCKS\_PER\_SEC);  std::cout << std::endl;  system("pause");  return 0;  } |

Листинг 3.1 – Файл main.cpp

Результат выполнения программы представлен на рисунке 4.

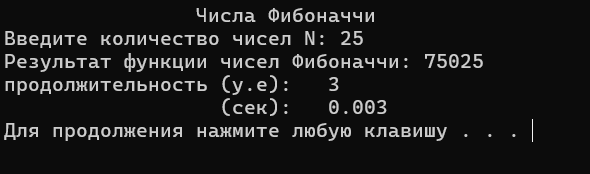


Рисунок 4 – Результат работы программы

Далее определим зависимость продолжительности вычислений от входного параметра N для алгоритма, реализующего числа Фибоначчи.

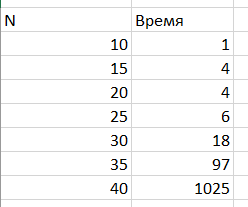


Рисунок 5 – Результаты тестирования №2

Далее по найденным значениям построим график и определим зависимость.

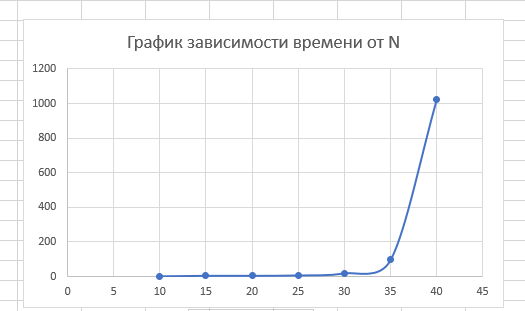


Рисунок 6 – График зависимости времени от N

Исходя из графика можно сделать вывод, что зависимость – экспоненциальная.

Подводя итоги, мы можем утверждать, что зависимость времени выполнения от количества циклов в первом случае линейна, т.к. количество операций прямо пропорционально числу циклов. Во втором эксперименте зависимость выполнения алгоритма Фибоначчи – экспоненциальная, т.к. мы видим резко возрастающий график.