**Белорусский государственный технологический университет**

**Факультет информационных технологий**

**Специальность программная инженерия**

Отчёт по лабораторной работе №1

По дисциплине «Математическое программирование»

На тему «Вспомогательные функции»

Выполнил:

Студент 2 курса 8 группы

Кулешов Артём Алексеевич

Преподаватель: асс. Ромыш А.С.

2025, Минск

**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:

start – функция установки начального числа как текущего значения для генератора.

dget – функция возвращает действительное число.

iget – функция возвращает целое число.

Перед тем, как начать работу, необходимо установить предкомпилированный заголовок. Для этого нужно создать файлы pch.h и pch.cpp. По умолчанию данных файлов проекта нет. В pch.h пишем все используемые в данное проекте библиотеки. В pch.cpp пишем только подключенный предкомпилированный файл. В свойствах проекта предкомпилированный заголовок stdafx.h меняем на pch.h, а в свойствах pch.cpp устанавливаем галочку на Yc (Создать). Далее приступаем непосредственно к работе. В задании 1 были прототипы функций и сама их реализация, в листинге ниже приведен пример 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

};

Листинг 1.1 – Файл Auxil.h

#include "Auxil.h"

#include <ctime>

namespace auxil {

void start() {

srand((unsigned)time(NULL)); // Инициализация генератора случайных чисел

}

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);

}

}

Листинг 1.2 – Файл Auxil.cpp

***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.

В данном примере проверяем работоспособность трех функций, описанных ранее прототипами.

#include "pch.h"

#define CYCLE 10000000 // Количество циклов

int main() {

setlocale(LC\_ALL, "rus");

double av1 = 0, av2 = 0;

clock\_t t1 = 0, t2 = 0;

auxil::start(); // Инициализация генератора случайных чисел

t1 = clock(); // Фиксация времени

for (int i = 0; i < CYCLE; i++) {

av1 += (double)auxil::iget(-100, 100); // Генерация случайных чисел

av2 += auxil::dget(-100, 100);

}

t2 = clock(); // Фиксация времени после выполнения

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)) / CLOCKS\_PER\_SEC;

std::cout << std::endl;

return 0;

}

Листинг 2.1 – Файл main.cpp

Результат работы программы представлен на рисунке 1.

Изображение выглядит как текст, Шрифт, снимок экрана

Контент, сгенерированный ИИ, может содержать ошибки.

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.

Диапазон циклов от 1000000 до 10000000 с шагом 1000000. Результаты тестирования представлены на рисунке 2.

Изображение выглядит как текст, снимок экрана, Шрифт

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 2 – Результаты тестирования

На рисунке 3 видно, что зависимость линейная, так как количество операций прямо пропорционально числу циклов.

Изображение выглядит как линия, текст, График, снимок экрана

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 3 – График зависимости времени от цикла

Далее исследование с числами Фибоначчи. Ниже представлен листинг программы, реализующей этот алгоритм.

Листинг 3.1 – Файл fibonachi.h

#include "pch.h"

#define CYCLE 1000000

int main()

{

setlocale(LC\_ALL, "rus");

cout << "Числа 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;

return 0;

}

#pragma once

#include "pch.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 3.2 – Файл fibonachi.cpp

На рисунке 4 представлен результат работы программы с числами Фибоначчи.

Изображение выглядит как текст, Шрифт, снимок экрана, черный

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 4 – Результат работы программы с числами Фибоначчи

Для того, чтобы построить график и определить зависимость, необходимо проверить за какое время будет выполняться алгоритм чисел Фибоначчи исходя из входного параметра N. Значения N будут меняться вручную в диапазоне от 10 до 45 с шагом 5.

Изображение выглядит как текст, снимок экрана, Шрифт, число

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 5 – Результаты тестирования программы с числами Фибоначчи

Изображение выглядит как текст, линия, График, снимок экрана

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 6 – График зависимости времени от N (числа Фибоначчи)

На графике наблюдается большое отклонение, следовательно данную зависимость можно считать экспоненциальной.

**ИТОГ**

1. В первом эксперименте зависимость времени выполнения от количества циклов линейная.
2. Во втором эксперименте зависимость времени выполнения рекурсивного алгоритма Фибоначчи экспоненциальная.