**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ  
 РОССИЙСКОЙ ФЕДЕРАЦИИ**

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ**

**НОВОСИБИРСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ**

**Факультет информационных технологий**

**Кафедра параллельных вычислений**

**ОТЧЕТ**

**О ВЫПОЛНЕНИИ ЛАБОРАТОРНОЙ РАБОТЫ**

«название работы»

студента (ки) \_\_\_\_2\_\_\_\_\_ курса, \_\_19211\_\_ группы

**Олимпиева Юрия Юрьевича**

Направление 09.03.01 – «Информатика и вычислительная техника»

Преподаватель:

**Н.А. Беляев**

2020\_год

**СОДЕРЖАНИЕ**

Цель……….………………………………………………………………….3

Задание...….………………………………………………………………….3

Описание работы....………………………………………………………….3

Заключение.………………………………………………………………….4

Приложение 1………………………………………………………………..5

Приложение 2………………………………………………………………..6

Приложение 3………………………………………………………………..7

Приложение 4………………………………………………………………..8

Приложение 5………………………………………………………………..9

**Цели**

1. Изучение методики измерения времени работы подпрограммы.

2. Изучение приемов повышения точности измерения времени работы подпрограммы.

3. Изучение способов измерения времени работы подпрограммы.

4. Измерение времени работы подпрограммы в прикладной программе.

**Задание**

1. Написать программу на языке C или C++, которая реализует **алгоритм вычисления функции e^x с помощью разложения в ряд Маклорена по первым N членам этого ряда**.

2. Проверить правильность работы программы на нескольких тестовых наборах входных данных.

3. Выбрать значение параметра N таким, чтобы время работы программы было порядка 15 секунд.

4. По приведенной методике определить время работы подпрограммы тестовой программы с относительной погрешностью не более 1%.

5. Составить отчет по лабораторной работе.

**Описание работы**

* Реализовал заданный алгоритм на ЯП Си. (приложение 1)
* В Microsoft Visual studio написал UnitTest1 для тестирования результатов работы подпрограммы (проверял абсолютную погрешность вычислений.) (приложение 3)
* Поместил вызов подпрограммы с некоторыми заданными аргументами в цикл в теле функции main(). Функцией clock() из библиотеки time.h

замерил время перед началом цикла и после тела цикла, разность, делённую на константу CLK\_TCK (число тактов/ секунду) вывел в консоль функцией printf(). (приложение 2)

* Эмпирическим путём подобрал необходимое значение переменной для ограничения счетчика цикла так, чтобы результирующее время выполнения было близко к 15 секундам. (приложение 5)
* Разделил полученное время на количество циклов, найденное в п.4.

 Тем самым получил приблизительное время выполнения одной итерации. (приложение 4)

* Составил отчёт о выполненной работе.

**Заключение**

В результате проделанной работы изучил методики измерения времени работы подпрограммы, способы измерения времени работы подпрограммы,

получил практические навыки измерения времени работы подпрограммы.

Приложение 1

double McLorenExp(double x, int N) {

double result = 0.0;

double element = 1.0;

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

result += element;

element = element \* x / (i + 1);

}

return result;

};

Приложение 2

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <stdio.h>

#include <stdlib.h>

#include <time.h>

using namespace std;

int main(void) {

int Start = clock();

for (int i = 0; i < 1600000; i++)

McLorenExp(2.0, 1000);

int Finish = clock();

printf("Result time:\t%f", (float)(Finish - Start)/(float)CLK\_TCK);

return 0;

}

Приложение 3

#include "pch.h"

#include "CppUnitTest.h"

#include <math.h>

#include "../ECM\_LAB\_1/ECM\_LAB\_1.cpp"

using namespace Microsoft::VisualStudio::CppUnitTestFramework;

namespace UnitTest1

{

TEST\_CLASS(UnitTest1)

{

public:

TEST\_METHOD(TestMethod1)

{

double x = 2.0;

int N = 120;

double Expected\_Value = 7.389056;

double Error = abs(Expected\_Value - McLorenExp(x, N));

Assert::IsTrue(Error <= 0.01);

}

TEST\_METHOD(TestMethod2)

{

double x = -0.5;

int N = 100;

double Expected\_Value = 0.606530;

double Error = abs(Expected\_Value - McLorenExp(x, N));

Assert::IsTrue(Error <= 0.01);

}

TEST\_METHOD(TestMethod3)

{

double x = 10.0;

int N = 100;

double Expected\_Value = 22026.465794;

double Error = abs(Expected\_Value - McLorenExp(x, N));

Assert::IsTrue(Error <= 0.01);

}

TEST\_METHOD(TestMethod4)

{

double x = 1.0;

int N = 120;

double Expected\_Value = 2.718281;

double Error = abs(Expected\_Value - McLorenExp(x, N));

Assert::IsTrue(Error <= 0.01);

}

TEST\_METHOD(TestMethod5)

{

double x = 0.0;

int N = 120;

double Expected\_Value = 1.0;

double Error = abs(Expected\_Value - McLorenExp(x, N));

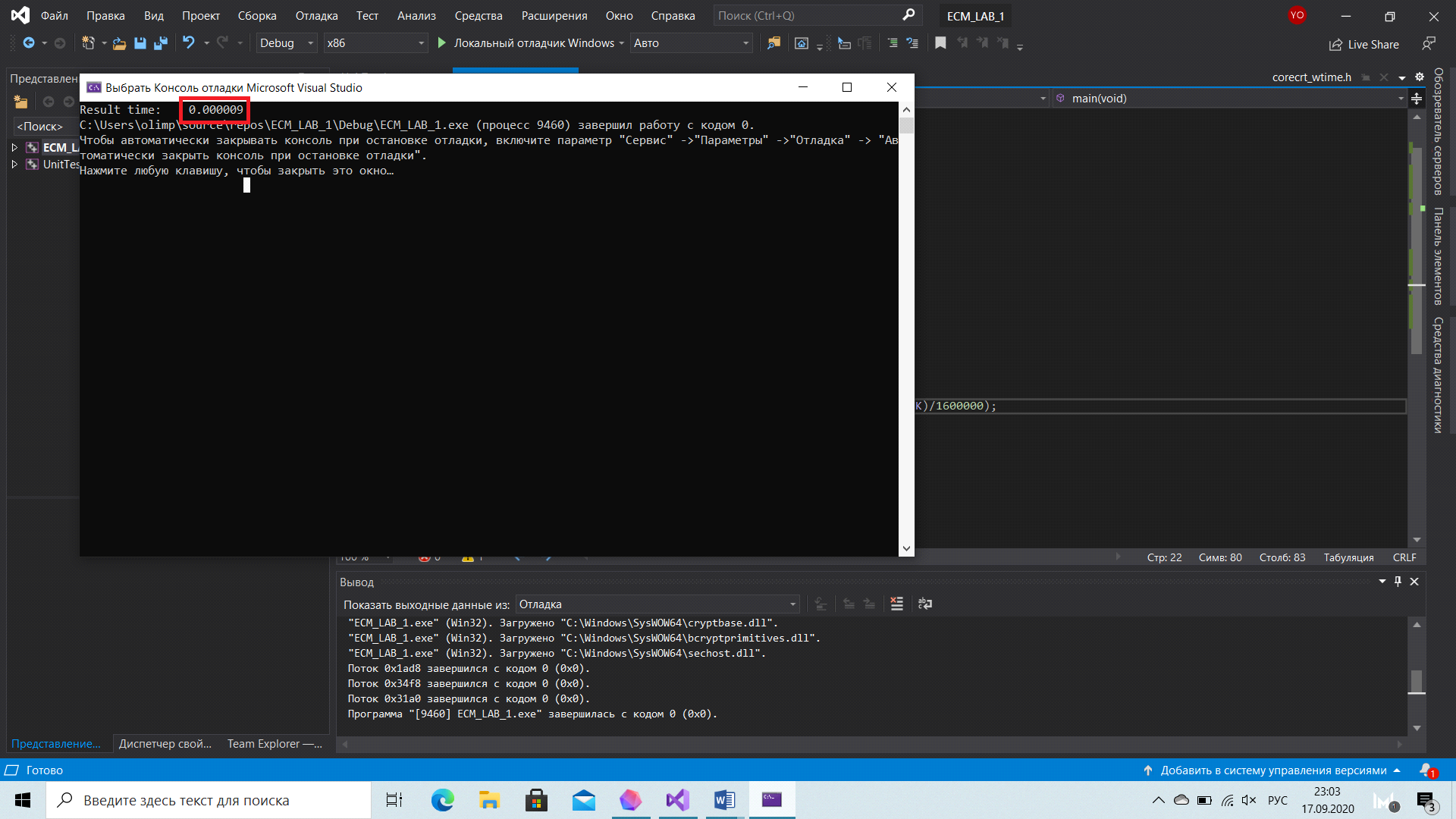
Assert::IsTrue(Error <= 0.01);

}

};

}

Приложение 4



Приложение 5

