Министерство образования и науки РФ

Санкт-Петербургский политехнический университет Петра Великого

Институт компьютерных наук и кибербезопасности

Высшая школа «Компьютерных технологий и информационных систем»

ОТЧЕТ

по дисциплине ««Теория и технология программирования»

**Лабораторная работа № 9**

**Выполнил:**

Cтудент гр. 5130902/40002 Г.Ю. Рюмин

**Проверил**

Ст. преподаватель А.М. Журавская

Санкт-Петербург

2025 г.

**1 Цель работы**

Цель задания – Ознакомиться с возможностью и методами использования библиотек динамической компоновки dll (dynamic linked library).

**2. Задание**

Написать программу, в которой для вычисления значений функции, используется динамически подключенная библиотека dll. Созданная программа должна быть совместима со всеми библиотеками одногрупников (должна быть реализована возможность «подмены» библиотеки без потери функционала работы программы).

В **основной программе** должны приниматься и проверяться принимаемые от пользователя параметры, создана callback функция, реализована отрисовка таблицы.

В качестве принимаемых от пользователя данных, интерфейс должен запрашивать следующие параметры:

* начальная, конечная границы вычисляемого промежутка, шаг;
* значение точности вычислений;
* дополнительный параметр (применяется в варианте функции ax).

В программе необходимо реализовать обработку входных данных и передачу их в функцию вычисления (расположенную в библиотеке). Помимо данных внутрь библиотеки необходимо передавать указатель на callback функцию. Callback функция должна быть одна, она должна принимать переменное число параметров и одним из принимаемых параметров должен быть тип информационного сообщения (вывод 1 таблицы, вывод 2 таблицы или вывод ошибки). Callback функцию необходимо вызывать каждый раз при успешном расчете очередного значения из промежутка.

Проверку возможности вычисления значений математических функций необходимо проводить в библиотеке. Вывод сообщения о невозможности вычисления необходимо реализовывать также посредством той же callback функции.

Формат вывода таблиц вычисленных значений функций указан в пункте 2.2. Цвет фона и текста таблиц необходимо выбирать в соответствии с указаниями оракула [«Гадалкин дом»](https://gadalkindom.ru/numerologiya/tsvet-imeni.html) Цвет букв – имя, цвет фона – отчество.

В **библиотеке необходимо**:

* Создать внешние (extern) функции для вывода имени функции и получения входных параметров;
* Создать две внутренние функции для вычисления рядного значения и математического значения;
* вычислить невязку (дельту) формула (1) значений рядной функции и стандартной библиотеки «math.h»;

|  |  |  |
| --- | --- | --- |
|  |  | (1) |

где – рядная функция, – библиотечная функция.

Вычисления ряда проводить до условия минимизации значения разности двух соседних членов ряда меньше заданного формула (2). Ряд рассчитывать через **рекуррентную форму расчета ряда**, пример вывода рекуррентной формы приведен **приложении А и Б** в конце документа.

|  |  |  |
| --- | --- | --- |
|  |  | (2) |

Результаты вычислений оформить виде таблиц! Пример таблицы находится под вариантами заданий в пункте 2.2.

**Варианты библиотечной и рядной функции:**

1. вычислить:

## **Код программы**

**Код из MathLibrary.dll:**

#include "pch.h"

#include <utility>

#include "MathLibrary.h"

#include <cmath>

double ryumin\_func(double x, double e) {

double sum = 0.0;

for (double i = 1; i < abs(round(x)); i++) {

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

}

return sum;

}

extern "C" \_\_declspec(dllexport) double myf\_math(double x, double a) {

return (1 + x) \* exp(x);

}

extern "C" \_\_declspec(dllexport) double myf\_2(double x, double a, double e) {

return ryumin\_func(x, e);

}

extern "C" \_\_declspec(dllexport) double myf\_delta(double fr, double fl) {

return sqrt(fabs(pow(fr, 2) - pow(fl, 2)));

}

extern "C" \_\_declspec(dllexport) void show(int code, std::vector<Set>\* sets, callbackShow func) {

if (code == cc::table) {

func(code, sets);

}

else {

if (code == cc::row) {

func(code, sets[0]);

}

}

}

extern "C" \_\_declspec(dllexport) std::string FName() {

return "Exponential";

}

**Код из MathLibrary.h:**

#pragma once

#ifdef MATHLIBRARY\_EXPORTS

#define MATHLIBRARY\_API \_\_declspec(dllexport)

#else

#define MATHLIBRARY\_API \_\_declspec(dllimport)

#endif

#include <limits>

#include <string>

#include <vector>

typedef void (\*callbackShow)(int, ...);

namespace cc {

const int len\_val{ 14 };

const std::string chars\_value("+-.0123456789");

const std::string chars\_step(".0123456789");

const int dp{ 12 };

const double err{ std::numeric\_limits<double>::infinity() };

const int table{ 1 };

const int row{ 2 };

}

struct Set {

std::string e, x, fr, fl, d;

Set() = default;

Set(std::string e\_, std::string x\_, std::string fr\_, std::string fl\_, std::string d\_)

: e(e\_), x(x\_), fr(fr\_), fl(fl\_), d(d\_) {

}

};

// Переименованная функция

extern "C" MATHLIBRARY\_API double myf\_2(double x, double a, double e);

extern "C" MATHLIBRARY\_API double myf\_math(double x, double a);

extern "C" MATHLIBRARY\_API double myf\_delta(double fr, double fl);

extern "C" MATHLIBRARY\_API void show(int code, std::vector<Set>\* sets, callbackShow func);

extern "C" MATHLIBRARY\_API std::string FName();

**Код из main.cpp:**

#define NOMINMAX

#include <iostream>

#include <string>

#include <vector>

#include <windows.h>

#include <locale>

#include <clocale>

#include <sstream>

#include <iomanip>

#include <cmath>

#include <limits>

using namespace std;

#include "MathLibrary.h"

void \_\_stdcall ShowTable(int code, ...) {

va\_list args;

va\_start(args, code);

const string inf\_str = to\_string(numeric\_limits<double>::infinity());

HANDLE hConsole;

hConsole = GetStdHandle(STD\_OUTPUT\_HANDLE);

SetConsoleTextAttribute(hConsole, FOREGROUND\_RED | FOREGROUND\_BLUE | BACKGROUND\_RED | BACKGROUND\_INTENSITY);

if (code == cc::table) {

vector<Set>\* sets = va\_arg(args, vector<Set>\*);

cout << string(120, '-') << endl;

cout << "|" << setw(22) << "X"

<< "|" << setw(31) << "f(X)"

<< "|" << setw(31) << "F(X)"

<< "|" << setw(31) << "delta"

<< "|" << endl;

cout << string(120, '-') << endl;

for (const auto& set : \*sets) {

bool fr\_inf = (set.fr == inf\_str);

bool fl\_inf = (set.fl == inf\_str);

cout << "|" << setw(22) << set.x

<< "|" << setw(31) << (fr\_inf ? "error" : set.fr)

<< "|" << setw(31) << (fl\_inf ? "error" : set.fl)

<< "|" << setw(31) << (fr\_inf || fl\_inf ? "error" : set.d)

<< "|" << endl;

}

cout << string(120, '-') << endl;

}

else if (code == cc::row) {

Set\* set = va\_arg(args, Set\*);

bool fr\_inf = (set->fr == inf\_str);

bool fl\_inf = (set->fl == inf\_str);

if (set->fr.c\_str() == "inf" && set->fl.c\_str() == "inf")

cout << "|" << setw(22) << set->x

<< "|" << setw(31) << (fr\_inf ? "error" : set->fr)

<< "|" << setw(31) << (fl\_inf ? "error" : set->fl)

<< "|" << setw(31) << (fr\_inf || fl\_inf ? "error" : set->d)

<< "|" << endl;

}

SetConsoleTextAttribute(hConsole,

FOREGROUND\_RED | FOREGROUND\_GREEN | FOREGROUND\_BLUE

);

va\_end(args);

}

template <typename T>

T safe\_input(const string& prompt) {

T value;

while (true) {

cout << prompt;

string input;

getline(cin, input);

if (input.empty()) {

cerr << "Ошибка: Пустой ввод!" << endl;

continue;

}

bool is\_valid = true;

for (char c : input) {

if (!isdigit(c) && c != '.' && c != '-' && c != '+') {

is\_valid = false;

break;

}

}

if (!is\_valid) {

cerr << "Ошибка: Недопустимые символы!" << endl;

continue;

}

istringstream iss(input);

if (iss >> value) {

return value;

}

else {

cerr << "Ошибка: Некорректный формат!" << endl;

}

}

}

int input\_epsilon() {

int epsilon\_power;

while (true) {

epsilon\_power = safe\_input<int>("Введите степень точности E (1-12): ");

if (epsilon\_power >= 1 && epsilon\_power <= 12) {

return epsilon\_power;

}

else {

cerr << "Ошибка: Степень должна быть от 1 до 12!" << endl;

}

}

}

int main() {

setlocale(LC\_ALL, "Russian");

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

HINSTANCE hDLL = LoadLibrary(TEXT("MathLibrary.dll"));

if (!hDLL) {

cerr << "Ошибка загрузки DLL!" << endl;

return 1;

}

// Получение функций с проверкой

auto myf\_series = reinterpret\_cast<double(\*)(double, double, double)>(GetProcAddress(hDLL, "myf\_2"));

auto myf\_math = reinterpret\_cast<double(\*)(double, double)>(GetProcAddress(hDLL, "myf\_math"));

auto delta = reinterpret\_cast<double(\*)(double, double)>(GetProcAddress(hDLL, "myf\_delta"));

auto show = reinterpret\_cast<void(\*)(int, const vector<Set>&, callbackShow)>(GetProcAddress(hDLL, "show"));

auto FName = reinterpret\_cast<std::string(\*)()>(GetProcAddress(hDLL, "FName"));

if (!myf\_series || !myf\_math || !delta || !show) {

cerr << "Ошибка загрузки функций из DLL!" << endl;

FreeLibrary(hDLL);

return 1;

}

double x\_start = safe\_input<double>("Введите начальную границу x\_start: ");

while (x\_start > pow(10, 12) || -pow(10, 12) > x\_start) {

cout << "Ошибка: введите значение из диапазона (-10^12;10^12)!" << endl;

x\_start = safe\_input<double>("Введите начальную границу x\_start: ");

}

double x\_end = safe\_input<double>("Введите конечную границу x\_end: ");

while (x\_end > pow(10, 12) || -pow(10, 12) > x\_end || x\_end < x\_start) {

cout << "Ошибка: введите значение из диапазона (-10^12;10^12), меньшее, чем x\_start!" << endl;

x\_end = safe\_input<double>("Введите начальную границу x\_end: ");

}

double step = safe\_input<double>("Введите шаг step: ");

while (step > pow(10, 12) || step < 0) {

cout << "Ошибка: введите значение из диапазона (0;10^12)!" << endl;

step = safe\_input<double>("Введите шаг step: ");

}

vector<Set> table1;

for (double x = x\_start; x <= x\_end; x += step) {

double fr = myf\_series(x, 0, 7);

double fl = myf\_math(x, 0);

table1.push\_back({

to\_string(7),

to\_string(x),

to\_string(fr),

to\_string(fl),

to\_string(delta(fr, fl))

});

}

cout << endl << FName() << endl;

show(cc::table, table1, ShowTable);

int epsilon = input\_epsilon();

double x\_ideal = safe\_input<double>("Введите x\_ideal: ");

while (x\_ideal > pow(10, 12) || -pow(10, 12) > x\_ideal) {

cout << "Ошибка: введите значение из диапазона (-10^12;10^12)!" << endl;

x\_ideal = safe\_input<double>("Введите начальную границу x\_ideal: ");

}

vector<Set> table2;

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

double e = pow(10,-i);

double fr = myf\_series(x\_ideal, 0, e) \* e;

double fl = myf\_math(x\_ideal, 0) \* e;

table2.push\_back({

to\_string(e),

to\_string(x\_ideal),

to\_string(fr),

to\_string(fl),

to\_string(delta(fr, fl))

});

}

show(cc::table, table2, ShowTable);

FreeLibrary(hDLL);

return 0;

}

**Пример работы программы**

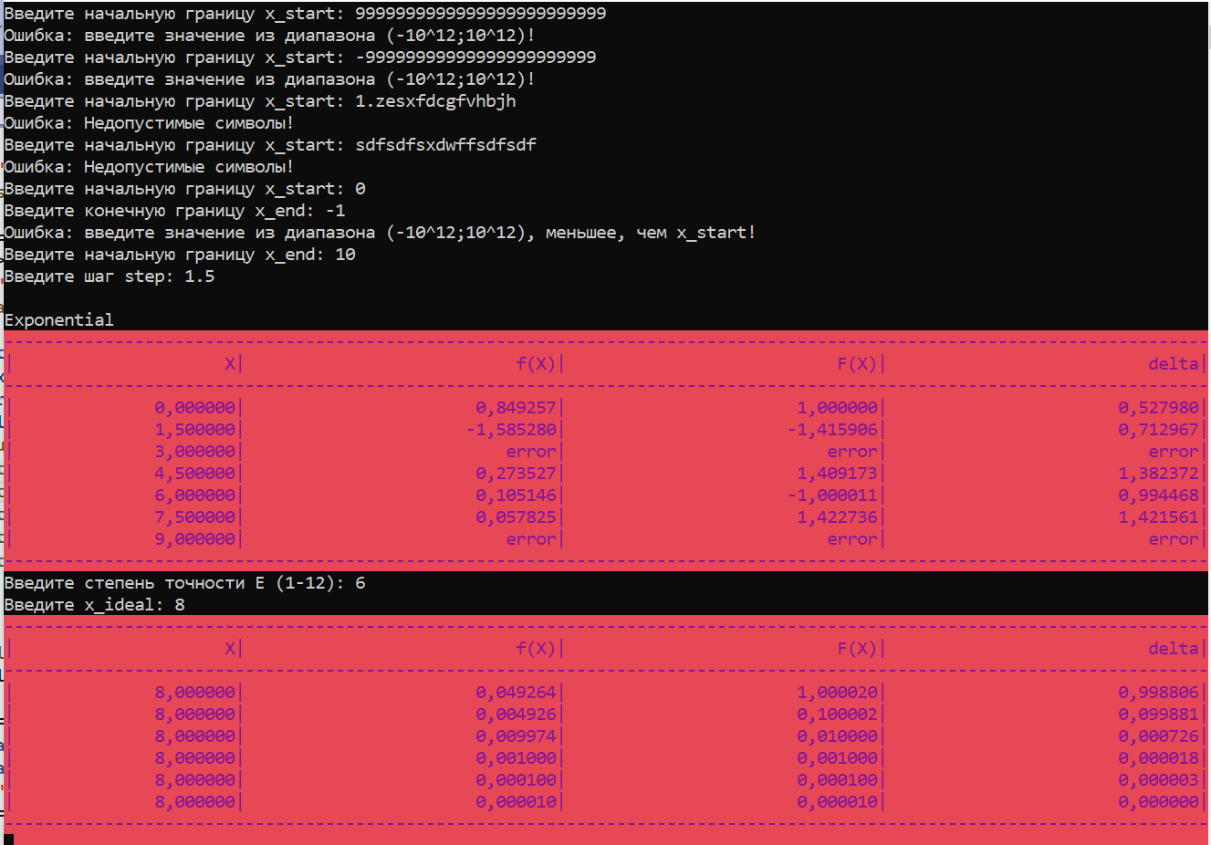


Рисунок 1 – Пример работы программы с некорректным и корректным вводом и с библиотекой одногруппника



Рисунок 2 – Пример работы программы с корректным вводом и с собственной библиотекой

## **Вывод**

В ходе выполнения лабораторной работы была успешно реализована динамически подключаемая библиотека (DLL), обеспечивающая вычисление значений функции *ex*(1+*x*) через рядное разложение и математическую формулу, а также расчёт невязки между ними. Основная программа интегрировала эту библиотеку, предоставив пользователю интуитивно понятный интерфейс для ввода параметров с автоматической проверкой на корректность данных, включая ограничение точности в диапазоне от -1012 до -1012. Результаты вычислений на заданном интервале и для различных уровней точности были оформлены в виде структурированных таблиц. Работа продемонстрировала навыки работы с динамическими библиотеками, обработкой исключений, настройкой консольного интерфейса через Windows API, а также обеспечение совместимости между компонентами программы.