**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ**

**«САНКТ-ПЕТЕРБУРГСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ ПЕТРА ВЕЛИКОГО»**

Институт компьютерных наук и технологий

Образовательная программа «Информационные системы и технологии»

Дисциплина «Теория и технология программирования»

Отчет к лабораторной работе № 9

«Использование библиотек динамической компоновки»

Выполнил:

студент группы 3530902/20001

Сафонов А. И.

Принял:

Доцент к.т.н Хлопин С. В.

Санкт-Петербург, 2023 год

**Задание лабораторной № 9:**

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.

**Текст программы:**

Файл main.cpp:

#define \_USE\_MATH\_DEFINES

#include <iostream>

#include <Windows.h>

#include <iomanip>

#include "Data.h"

#include "ReadFunc.h"

#include "round.h"

typedef void(\*calcFunc)(SafonovInfoType info, void\*, ...);

typedef const char\* (\*functionName)();

using namespace std;

HANDLE h = GetStdHandle(STD\_OUTPUT\_HANDLE);

ColorSaf MyColor = { 12,3 };

int openDll(HINSTANCE& dll, calcFunc& calculator, functionName& fName) {

if (dll == nullptr) {

cout << "Библиотека не была открыта.\n"s;

system("pause");

return 1;

}

calculator = (calcFunc)GetProcAddress(dll, "calculator");

if (calculator == nullptr) {

cout << "Функция \"calculator\" не обнаружена в библиотеке."s << endl;

system("pause");

return 1;

}

fName = (functionName)GetProcAddress(dll, "FName");

if (fName == nullptr) {

cout << "Функция \"FName\" не обнаружена в библиотеке."s << endl;

system("pause");

return 1;

}

cout << endl;

return 0;

}

void SetColor(ColorSaf color) {

SetConsoleTextAttribute(h, (WORD)((color.bgSaf << 4) | color.textSaf));

}

void SetupDefaultColor(CONSOLE\_SCREEN\_BUFFER\_INFO defaultColor) {

SetConsoleTextAttribute(h, defaultColor.wAttributes);

}

namespace Drow {

void DrowLine(SafonovInfoType infoTable, ColorSaf color, ...) {

SetColor(color);

cout << string(100, '-') << endl;

if (infoTable == Table1Saf) {

va\_list arguments;

va\_start(arguments, color);

int numberLine = va\_arg(arguments, int);

double x = va\_arg(arguments, double);

double f\_row = va\_arg(arguments, double);

double f\_math = va\_arg(arguments, double);

double delta = va\_arg(arguments, double);

va\_end(arguments);

cout << "| " << left << setw(7) << numberLine << " | ";

if (x != 0 && (double\_round(x) == 0 || abs(x) > 99999)) {

cout << left << scientific << setw(21) << x << " | ";

}

else {

cout << left << fixed << setprecision(7) << setw(21) << x << " | ";

}

if (f\_row != 0 && (double\_round(f\_row) == 0 || abs(f\_row) > 99999)) {

cout << left << scientific << setw(19) << f\_row << " | ";

}

else {

cout << left << fixed << setprecision(7)<< setw(19)<< f\_row << " | ";

}

if (f\_math != 0 && (double\_round(f\_math) == 0 || abs(f\_math) > 99999)) {

cout << left << scientific << setw(19) << f\_math << " | ";

}

else {

cout << left << fixed << setprecision(7) << setw(19) << f\_math << " | ";

}

if (delta != 0 && (double\_round(delta) == 0 || delta > 99999)) {

cout << left << scientific << setw(18) << delta << " |\n";

}

else {

cout << left << fixed << setprecision(7)<< setw(18)<< delta << " |\n";

}

}

else if (infoTable == Table2Saf) {

va\_list arguments;

va\_start(arguments, color);

int numberLine = va\_arg(arguments, int);

double tb\_epsilon = va\_arg(arguments, double);

double f\_row = va\_arg(arguments, double);

double f\_math = va\_arg(arguments, double);

double delta = va\_arg(arguments, double);

va\_end(arguments);

SetColor(color);

cout << "| " << setw(7) << left << numberLine << " | "

<< setw(21) << left << tb\_epsilon << " | ";

if (f\_row != 0 && (double\_round(f\_row) == 0 || f\_row > 99999)) {

cout << left << scientific << setw(19)<< f\_row << " | ";

}

else {

cout << left << fixed << setprecision(6) << setw(19) << f\_row << " | ";

}

if (f\_math != 0 && (double\_round(f\_math) == 0 || f\_math > 99999)) {

cout << left << scientific << setw(19) << f\_math << " | ";

}

else {

cout << left << fixed << setprecision(6) << setw(19) << f\_math << " | ";

}

if (delta != 0 && (double\_round(delta) == 0 || delta > 99999)) {

cout << left << scientific << setw(18) << delta << " |\n";

}

else {

cout << left << fixed << setprecision(6) << setw(18) << delta << " |\n";

}

}

else if (infoTable == ErrorSaf) {

va\_list arguments;

va\_start(arguments, color);

int numberLine = va\_arg(arguments, int);

double arg = va\_arg(arguments, double);

va\_end(arguments);

SetColor(color);

cout << "| " << setw(7) << left << numberLine << " | ";

if (arg != 0 && (double\_round(arg) == 0 || arg > 99999)) {

cout << left << scientific << setw(21)<< arg;

}

else {

cout << left << fixed << setprecision(7) << setw(21) << arg;

}

cout << " | ошибка вычисления функции |\n";

}

}

void DrowUpTable(string funcName) {

string myFName = funcName + "\_my(x)";

funcName += "(x)";

cout << "\nТаблица 1.\n";

SetColor(MyColor);

cout << string(100, '-') << endl;

cout << "| № | x | " << left << setw(19) << myFName << " | " << left << setw(19) << funcName << " | del |\n";

}

void DrowUpTableIdeal(string funcName) {

string myFName = funcName + "\_my(x\_ideal)";

cout << endl;

cout << "\nТаблица 2.\n";

funcName += "(x\_ideal)";

SetColor(MyColor);

cout << string(100, '-') << endl;

cout << "| № | eps | " << left << setw(19) << myFName << " | " << left << setw(19) << funcName << " | del |\n";

}

void DrowDownTable(CONSOLE\_SCREEN\_BUFFER\_INFO startColor) {

SetColor(MyColor);

cout << string(100, '-');

SetupDefaultColor(startColor);

cout << endl << endl;

}

}

int main() {

setlocale(LC\_CTYPE, "Russian");

HANDLE h = GetStdHandle(STD\_OUTPUT\_HANDLE);

CONSOLE\_SCREEN\_BUFFER\_INFO startColor;

GetConsoleScreenBufferInfo(h, &startColor);

HINSTANCE dll = LoadLibraryA("Safonov.dll");

calcFunc calculator;

functionName fName;

if (openDll(dll, calculator, fName)) {

return 1;

}

double x\_start, x\_end, x\_del, tb\_epsilon, a, x\_ideal;

Read::ReadlineInterval(tb\_epsilon, x\_start, x\_end, x\_del, a);

string funcName = fName();

Drow::DrowUpTable(funcName);

calculator(Table1Saf, Drow::DrowLine, x\_start, x\_end, x\_del, tb\_epsilon, a);

Drow::DrowDownTable(startColor);

x\_ideal = Read::GetNum("Введите x\_ideal:");

Drow::DrowUpTableIdeal(funcName);

calculator(Table2Saf, Drow::DrowLine, x\_ideal, a);

Drow::DrowDownTable(startColor);

system("pause");

return 0;

}

Файл round.cpp:

#include "round.h"

double double\_round(double x)

{

return (double)round(10000000 \* x) / 10000000;

}

Файл round.h:

#include "round.h"

double double\_round(double x)

{

return (double)round(10000000 \* x) / 10000000;

}

Файл ReadFunc.h:

#pragma once

#include <string>

#include <vector>

#include <iostream>

using namespace std::literals;

class Read {

public:

static std::string ReadLine()

{

std::string s;

std::getline(std::cin, s);

return s;

}

static double GetNum(std::string message, std::string error\_message = "Неккоректное значение"s)

{

while (true) {

std::cout << message;

std::string str = ReadLine();

std::vector<std::string> words = SplitIntoWords(str);

if (words.size() == 1 && CheckIsFloat(words[0]))

{

return std::stod(str);

}

std::cout << error\_message << std::endl;

}

}

static double GetPositivNum(std::string message, std::string error\_message = "Неккоректное значение"s)

{

while (true) {

double epsilon = GetNum(message, error\_message);

if (epsilon > 0) {

return epsilon;

}

std::cout << error\_message << std::endl;

}

}

static void ReadlineInterval(double& tb\_epsilon, double& x\_start, double& x\_end, double& x\_del, double& a) {

tb\_epsilon = Read::GetPositivNum("Введите точность (для корректной работы программы ее значение должно быть больше 0):"s);

do {

x\_start = Read::GetNum("Введите x\_start:"s);

x\_end = Read::GetNum("Введите x\_end : "s);

do {

x\_del = Read::GetNum("Введите шаг интервала (для корректной работы программы оно не должно быть равно нулю):"s);

} while (x\_del == 0);

if (!(x\_start < x\_end && x\_del < 0) || (x\_start > x\_del && x\_del > 0)) {

break;

}

std::cout << "Интервал введен неверно, проход невозможен\n"s;

} while (true);

a = Read::GetNum("Введите значение параметра a:"s);

}

private:

static std::vector<std::string> SplitIntoWords(const std::string& textSaf)

{

std::vector<std::string> words;

std::string word;

for (const char c : textSaf) {

if (c == ' ') {

if (!word.empty()) {

words.push\_back(word);

word.clear();

}

}

else {

word += c;

}

}

if (!word.empty()) {

words.push\_back(word);

}

return words;

}

static bool CheckIsNum(std::string str)

{

if (str[0] == '+' || str[0] == '-')

str = str.substr(1, str.length());

bool have\_dots = false;

for (char ch : str) {

if (ch == '.' || ch == ',') {

have\_dots = true;

}

else if (have\_dots && ch != '0' || ch > '9' || ch < '0') {

return false;

}

}

return true;

}

static bool CheckIsFloat(std::string str)

{

if (str[0] == '+' || str[0] == '-')

str = str.substr(1, str.length());

bool have\_dots = false;

for (char ch : str) {

if (ch == '.' || ch == ',') {

if (have\_dots) {

return false;

}

have\_dots = true;

}

else if (ch > '9' || ch < '0') {

return false;

}

}

return true;

}

};

Файл ReadFunc.h:

#pragma once

enum SafonovInfoType {

Table1Saf,

Table2Saf,

ErrorSaf

};

struct ColorSaf {

int textSaf;

int bgSaf;

};

**Текст dll-библиотеки:**

файл dllmain.h:

#pragma once

#include "pch.h"

enum SafonovInfoType {

Table1Saf,

Table2Saf,

ErrorSaf

};

struct ColorSaf {

int textSaf;

int bgSaf;

};

ColorSaf color = { 12,3 };

typedef void(\*FCallback)(SafonovInfoType type, ColorSaf c, ...);

extern "C" \_\_declspec(dllexport) const char\* FName();

extern "C" \_\_declspec(dllexport) void calculator(SafonovInfoType, void\*(SafonovInfoType infoTable, ColorSaf My\_color, ...), ...);

void table1\_processing(SafonovInfoType, void\*(SafonovInfoType infoTable, ColorSaf My\_color, ...), ...);

void table2\_processing(SafonovInfoType, void\*(SafonovInfoType infoTable, ColorSaf My\_color, ...), ...);

double Safonov\_arcth(double, double);

double myf\_3530902\_20001(double, double, double);

double myf\_math(double, double);

bool IsEqual(double, double);

double double\_round(double);

файл dllmain.cpp:

// dllmain.cpp : Определяет точку входа для приложения DLL.

#define \_USE\_MATH\_DEFINES

#include "pch.h"

#include "dllmain.h"

#include <math.h>

#include <iostream>

#include <cstdarg>

using namespace std;

extern "C" \_\_declspec(dllexport) const char\* FName()

{

return "f24";

}

double Safonov\_arcth(double x, double tb\_epsilon) {

int n = 1;

long double result = x, last = result, curr = result \* (-9) \* x \* x / 2;

do

{

result += curr;

n++;

last = curr;

curr \*= -9 \* x \* x / ((2\*n) \* (2\*n-1));

} while ((fabs(fabs(curr) - fabs(last))) >= tb\_epsilon);

return result;

}

double myf\_3530902\_20001(double x, double tb\_epsilon, double a) {

return Safonov\_arcth(x, tb\_epsilon);

}

double myf\_math(double x, double a) {

return x \* cos(3\*x);

}

extern "C" \_\_declspec(dllexport) void calculator(SafonovInfoType infoTable, void\* output\_line\_Safonov(SafonovInfoType infoTable, ColorSaf My\_color, ...), ...) {

if (infoTable == Table1Saf) {

va\_list argumentsForFunction;

va\_start(argumentsForFunction, output\_line\_Safonov);

double x\_start = va\_arg(argumentsForFunction, double);

double x\_end = va\_arg(argumentsForFunction, double);

double x\_del = va\_arg(argumentsForFunction, double);

double tb\_epsilon = va\_arg(argumentsForFunction, double);

double a = va\_arg(argumentsForFunction, double);

va\_end(argumentsForFunction);

table1\_processing(infoTable, output\_line\_Safonov, x\_start, x\_end, x\_del, tb\_epsilon, a);

}

else if (infoTable == Table2Saf) {

va\_list arguments;

va\_start(arguments, output\_line\_Safonov);

double x\_ideal = va\_arg(arguments, double);

double a = va\_arg(arguments, double);

va\_end(arguments);

table2\_processing(infoTable, output\_line\_Safonov, x\_ideal, a);

}

}

void table1\_processing(SafonovInfoType infoTable, void\* output\_line\_Safonov(SafonovInfoType infoTable, ColorSaf My\_color, ...), ...) {

va\_list argumentsForFunction;

va\_start(argumentsForFunction, output\_line\_Safonov);

double x\_start = va\_arg(argumentsForFunction, double);

double x\_end = va\_arg(argumentsForFunction, double);

double x\_del = va\_arg(argumentsForFunction, double);

double tb\_epsilon = va\_arg(argumentsForFunction, double);

double a = va\_arg(argumentsForFunction, double);

va\_end(argumentsForFunction);

double f\_row, f\_math, delta;

int LineCount = 1;

for (double x = x\_start; (x\_start < x\_end ? x < x\_end : x > x\_end) && !IsEqual(x\_start, x\_end); x += x\_del) {

if (fabs(x - x\_end) >= numeric\_limits<double>::epsilon()) {

x = double\_round(x);

f\_row = myf\_3530902\_20001(x, tb\_epsilon, a);

f\_math = myf\_math(x, a);

delta = sqrt(abs(f\_row \* f\_row - f\_math \* f\_math));

if (isnan(delta) || isinf(delta) || (abs(x) > 10)) {

output\_line\_Safonov(ErrorSaf, color, ++LineCount, x);

}

else {

output\_line\_Safonov(infoTable, color, ++LineCount, x, f\_row, f\_math, delta);

}

}

}

f\_row = myf\_3530902\_20001(x\_end, tb\_epsilon, a);

f\_math = myf\_math(x\_end, a);

delta = sqrt(abs(f\_row \* f\_row - f\_math \* f\_math));

if (isnan(delta) || isinf(delta) || (abs(x\_end) > 10)) {

output\_line\_Safonov(ErrorSaf, color, LineCount++, x\_end);

}

else {

output\_line\_Safonov(infoTable, color, LineCount++, x\_end, f\_row, f\_math, delta);

}

}

void table2\_processing(SafonovInfoType infoTable, void\* output\_line\_Safonov(SafonovInfoType infoTable, ColorSaf My\_color, ...), ...) {

va\_list arguments;

va\_start(arguments, output\_line\_Safonov);

double x\_ideal = va\_arg(arguments, double);

double a = va\_arg(arguments, double);

va\_end(arguments);

double f\_row, delta;

double f\_math = myf\_math(x\_ideal, a);

int LineCount = 1;

for (double tb\_epsilon = 0.1; tb\_epsilon >= 0.0000001; tb\_epsilon /= 10) {

f\_row = myf\_3530902\_20001(x\_ideal, tb\_epsilon, a);

delta = sqrt(abs(f\_row \* f\_row - f\_math \* f\_math));

if (!isnan(delta) && !isinf(delta) && (abs(x\_ideal) < 10)) {

output\_line\_Safonov(infoTable, color, ++LineCount, tb\_epsilon, f\_row, f\_math, delta);

}

else {

output\_line\_Safonov(ErrorSaf, color, ++LineCount, tb\_epsilon);

}

}

}

bool IsEqual(double lhs, double rhs) {

return fabs(lhs - rhs) < numeric\_limits<double>::epsilon();

}

double double\_round(double x) {

return (double)round(10000000 \* x) / 10000000;

}

BOOL APIENTRY DllMain( HMODULE hModule,

DWORD ul\_reason\_for\_call,

LPVOID lpReserved

)

{

switch (ul\_reason\_for\_call)

{

case DLL\_PROCESS\_ATTACH:

case DLL\_THREAD\_ATTACH:

case DLL\_THREAD\_DETACH:

case DLL\_PROCESS\_DETACH:

break;

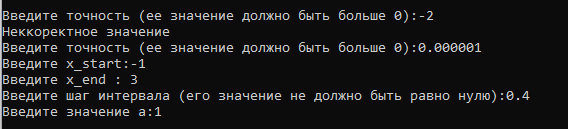
}

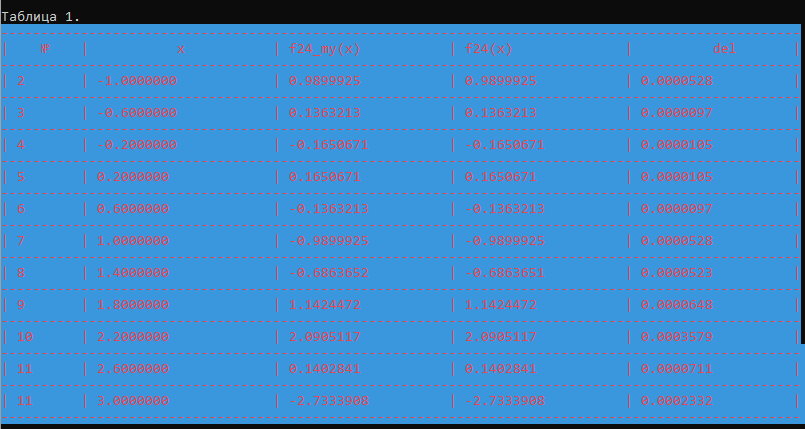
return TRUE;

}

**Пример работы:**

Входные данные:

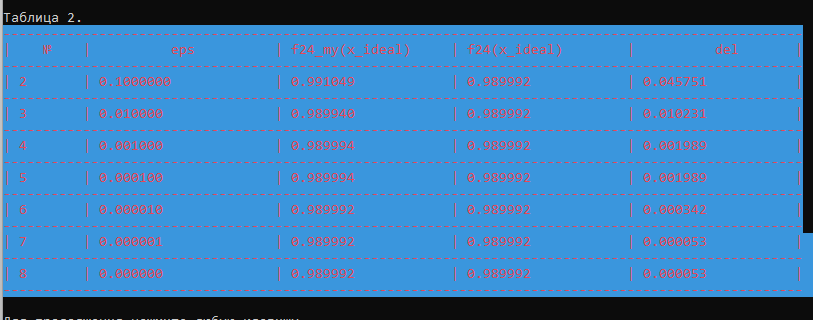


Выходные данные: 

Входные данные:



Выходные данные:

****

**Выводы:** в ходе выполнения лабораторной работы я научился работать с динамически подключаемыми библиотеками. Создав свою библиотеку, мы познакомились с тем как они работают, научились проверять их на правильность открытия. Так же, из-за того что мы написали библиотеку основываясь на общей документации, мы смогли подменять математические функции расчета, вставляя чужие наработки в свою программу.