МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение высшего образования

«Национальный исследовательский Нижегородский государственный университет им. Н.И. Лобачевского» (ННГУ)

Институт информационных технологий, математики и механики

Направление подготовки: «Фундаментальная информатика и информационные технологии»

Программа бакалавриата: «Системное программирование»

Отчёт

на тему

«Расчет значений для некоторого набора функций в заданной точке с заданной погрешностью за счёт разложения этих функций в ряд Тейлора»

Выполнил:

студент группы 3825Б1ФИсп1

Чернов А.А.

Нижний Новгород

2025

Содержание

1. Введение

Разложение в ряд Тейлора — это способ представить функцию в виде бесконечной суммы степенных функций (многочлена), используя её значения и значения её производных в определённой точке.

2. Постановка задачи

Задача: Разработать программу, позволяющую выполнить расчёт значений для набора функций (sin(x), cos(x), exp(x), sh(x)) в заданной пользователем точке с заданной пользователем погрешностью за счёт разложения функций в ряд Тейлора.

Во время работы в режиме 1 пользователь выбирает функцию; точку, в которой программа будет искать значение; количество слагаемых в разложении в ряд Тейлора; точность вычисления (от 0.000001 и больше). На выходе пользователь получает вычисленное значение функции в заданной точке, эталонное значение функции в этой точке (благодаря встроенным функциям), разницу между эталонным значением и вычислениям с помощью наших функций и количество использованных для этого вычисления слагаемых в разложении в ряд Тейлора.

Во время работы в режиме 2 пользователь выбирает функцию; точку, в которой программа будет искать значение; Nmax - количество экспериментов (количество слагаемых в разложении в ряд Тейлора).
На выходе пользователь получает эталонное значение функции в этой точке; заданное им количество строк со значениями, полученными в результате использования Nmax количества слагаемых в разложении в ряд Тейлора; для каждой строчки выводится разница между вычисленным и эталонным значениями.

Для решения было выполнено следующее:

1. Изучены теоретические сведения о разложении функции в ряд Тейлора
2. Разработаны функции taylor_sin, taylor_cos, taylor_exp, taylor_sh которые высчитывают значение функции в заданной точке у sin(x), cos(x), exp(x) и sh(x) соответственно.
3. Разработана функция serial_experiment для работы программы в режиме 2.
4. Добавлены элементы кода, проверяющие введенные пользователем символы на правильность
5. Были проведены тесты для проверки работоспособности программы.

3. Программная реализация

3.1. Стандартные библиотеки.

```
#include <math.h>
```

Math.h используется для вычислений эталонных значений функций в точках, для приведения аргумента в sin(x), cos(x).

```
#include <stdio.h>
```

Stdio.h используется для добавления функция ввода и вывода (printf и scanf_s)

3.2. Функции для расчета значения выбранной пользователем функции в точке.

3.2.1. Функция для sin(x)

```
long double taylor_sin(long double x, int N, long double calculation_accuracy, int* used_count)
{
	long double TWO_PI = PI * 2.0L;
	x = fmodl(x, TWO_PI);
	if (x > PI)
	{
		x = x - TWO_PI;
	}
	else if (x < -PI)
	{
		x = x + TWO_PI;
	}

	long double result = x;
	long double term = x;
	*used_count = 1;

	for (int i = 1; i < N; i++)
	{
		term = -term * x * x / ((2.0L * i) * (2.0L * i + 1.0L));
		result = result + term;
		(*used_count)++;
		if (fabsl(term) < calculation_accuracy)
		{
			break;
		}
	}
	return result;

}
```

Функция принимает заданную пользователем точку X в которой программа ищет значение, N – количество слагаемых в разложении в ряд Тейлора, calculation_accuracy – точность вычисления, *used_count для вычисления количества использованных слагаемых в разложении в ряд Тейлора.

Для sin(x) стало необходимым приведение аргумента X к промежутку [-pi; pi] для облегчения вычислений значения (иначе вылет программы при x >300)

Функция использует не исходную формулу Тейлора, а преобразованную в рекуррентную для облегчения вычисления значения. Функция не с самого первого члена вычисляет каждый последующий, а использует предыдущий. (иначе вылет программы при x>25)

3.2.2. Функция для cos(x)

```
long double taylor_cos(long double x, int N, long double calculation_accuracy, int* used_count)
{
	long double TWO_PI = PI * 2.0L;
	x = fmodl(x, TWO_PI);
	if (x > PI)
	{
		x = x - TWO_PI;
	}
	else if (x < -PI)
	{
		x = x + TWO_PI;
	}


	long double result = 1.0L;
	long double term = 1.0L;
	*used_count = 1;

	for (int i = 1; i < N; i++)
	{
		term = -term * x * x / ((2.0L * i) * (2.0L * i - 1.0L));
		result = result + term;
		(*used_count)++;
		if (fabsl(term) < calculation_accuracy)
		{
			break;
		}
	}
	return result;
}
```

Функция принимает заданную пользователем точку X в которой программа ищет значение, N – количество слагаемых в разложении в ряд Тейлора, calculation_accuracy – точность вычисления, *used_count для вычисления количества использованных слагаемых в разложении в ряд Тейлора.

Для cos(x) стало необходимым приведение аргумента X к промежутку [-pi; pi] для облегчения вычислений значения (иначе вылет программы при x >300)

Функция использует не исходную формулу Тейлора, а преобразованную в рекуррентную для облегчения вычисления значения. Функция не с самого первого члена вычисляет каждый последующий, а использует предыдущий. (иначе вылет программы при x>25)

3.2.3. Функция для exp(x)

```
long double taylor_exp(long double x, int N, long double calculation_accuracy, int* used_count)
{
	long double result = 1.0L;
	long double term = 1.0L;
	*used_count = 1;

	for (int i = 1; i < N; i++)
	{
		term = term * (x / i);
		result = result + term;
		(*used_count)++;
		if (fabsl(term) < calculation_accuracy)
		{
			break;
		}
	}
	return result;
}
```

Функция принимает заданную пользователем точку X в которой программа ищет значение, N – количество слагаемых в разложении в ряд Тейлора, calculation_accuracy – точность вычисления, *used_count для вычисления количества использованных слагаемых в разложении в ряд Тейлора.

Функция использует не исходную формулу Тейлора, а преобразованную в рекуррентную для облегчения вычисления значения. Функция не с самого первого члена вычисляет каждый последующий, а использует предыдущий. (иначе вылет программы при x>25)

3.2.4. Функция для sh(x)

```
long double taylor_sh(long double x, int N, long double calculation_accuracy, int* used_count)
{
	long double result = x;
	long double term = x;
```

```
        *used_count = 1;

        for (int i = 1; i < N; i++) {
                term = term * x * x / ((2.0L * i) * (2.0L * i + 1.0L));
                result = result + term;
                (*used_count)++;
                if (fabsl(term) < calculation_accuracy)
                {
                        break;
                }
        }
        return result;
}
```

Функция принимает заданную пользователем точку X в которой программа ищет значение, N – количество слагаемых в разложении в ряд Тейлора, calculation_accuracy – точность вычисления, *used_count для вычисления количества использованных слагаемых в разложении в ряд Тейлора.

Функция использует не исходную формулу Тейлора, а преобразованную в рекуррентную для облегчения вычисления значения. Функция не с самого первого члена вычисляет каждый последующий, а использует предыдущий. (иначе вылет программы при x>25)

3.3. Функция для многократного расчета значения функции в точке.

```
void serial_experiment(int function_choice, long double x, int Nmax)
{

        printf("%-20s %-30s %-30s\n", "Count of elements", "Our calculation", "Difference");

        for (int n = 1; n <= Nmax; n++)
        {
                int used_count;
                long double intermediate_value = 0.0L;
                long double difference = 0.0L;

                switch (function_choice)
                {
                case 1:
                        intermediate_value = taylor_sin(x, n + 1, 0.0L, &used_count);
                        difference = fabsl(intermediate_value - sinl(x));
                        printf("%-20d %-30.15Lf %-25.20Lf\n", n, intermediate_value, difference);
                        break;
                case 2:
                        intermediate_value = taylor_cos(x, n + 1, 0.0L, &used_count);
                        difference = fabsl(intermediate_value - cosl(x));
                        printf("%-20d %-30.15Lf %-25.20Lf\n", n, intermediate_value, difference);
                        break;
                case 3:
                        intermediate_value = taylor_exp(x, n + 1, 0.0L, &used_count);
                        difference = fabsl(intermediate_value - expl(x));
                        printf("%-20d %-30.15Lf %-25.20Lf\n", n, intermediate_value, difference);
```

```
                        break;
                case 4:
                        intermediate_value = taylor_sh(x, n + 1, 0.0L, &used_count);
                        difference = fabsl(intermediate_value - sinhl(x));
                        printf("%-20d %-30.15Lf %-25.20Lf\n", n, intermediate_value, difference);
                        break;
                }

        }

}
```

Данная функция используется при режиме работы 2. Данная функция принимает function_choise – выбор пользователя функции, X – точка в которой необходимо рассчитать значение функции, Nmax для количества подсчётов значения при использовании от 1 до Nmax слагаемых в разложении в ряд Тейлора.

Также в этой функции используются функции из пункта 3.2. для расчёта значения заданных пользователем функции, но без заданной точности.

3.4. Основная функция main

```
int main()
{
        int work_choose, function_choose;
        printf("Choose mode of programm:\n1. One-time calculation of the function at your point\n2. Serial expirement\n----> "); scanf_s("%d", &work_choose); printf("\n"); if (work_choose != 1 && work_choose != 2) { printf("Choose 1 or 2, please"); }
        switch (work_choose)
        {
        case 1:
        {
                int N, used_count;
                long double x, calculation_accuracy;
                printf("Choose function:\n1. sin(x)\n2. cos(x)\n3. exp(x)\n4. sh(x)\n----> "); scanf_s("%d", &function_choose); printf("\n"); if (function_choose < 1 || function_choose>4) { printf("Choose something from list on your screen, please"); break; }
                printf("Choose point x:\n----> "); scanf_s("%Lf", &x); printf("\n");
                printf("Choose calculation accuracy (0.000001 and more):\n----> "); scanf_s("%Lf", &calculation_accuracy); printf("\n"); if (calculation_accuracy > 0.00001) { printf("Enter a accuracy more than 0.00001, please"); break; }
                printf("Choose count of elements in the Taylor series (1-1000):\n----> "); scanf_s("%d", &N); printf("\n"); if (N < 1 || N>1000) { printf("Select count in the range from 1 to 1000, please"); break; }
                switch (function_choose)
                {
                case 1:
                {
                        printf("Our calculation: %.20Lf\n", taylor_sin(x, N, calculation_accuracy, &used_count));
                        printf("Etalon sin: %.20Lf\n", sinl(x));
                        printf("Difference: %.20Lf\n", fabsl(taylor_sin(x, N, calculation_accuracy, &used_count) - sinl(x)));
                        printf("Count of used elements in the Taylor series: %d", used_count);
                        break;
```

```
                }
                case 2:
                {
                        printf("Our calculation: %.20Lf\n", taylor_cos(x, N, calculation_accuracy, &used_count));
                        printf("Etalon cos: %.20Lf\n", cosl(x));
                        printf("Difference: %.20Lf\n", fabsl(taylor_cos(x, N, calculation_accuracy, &used_count) -
cosl(x)));
                        printf("Count of used elements in the Taylor series: %d", used_count);
                        break;
                }
                case 3:
                {
                        printf("Our calculation: %.20Lf\n", taylor_exp(x, N, calculation_accuracy, &used_count));
                        printf("Etalon exp: %.20Lf\n", expl(x));
                        printf("Difference: %.20Lf\n", fabsl(taylor_exp(x, N, calculation_accuracy, &used_count) -
expl(x)));
                        printf("Count of used elements in the Taylor series: %d", used_count);
                        break;
                }
                case 4:
                {
                        printf("Our calculation: %.20Lf\n", taylor_sh(x, N, calculation_accuracy, &used_count));
                        printf("Etalon sh: %.20Lf\n", sinhl(x));
                        printf("Difference: %.20Lf\n", fabsl(taylor_sh(x, N, calculation_accuracy, &used_count) -
sinhl(x)));
                        printf("Count of used elements in the Taylor series: %d", used_count);
                }
                default: break;
                }
                break;
        }
        case 2:
        {
                int Nmax;
                long double x;
                printf("Choose function:\n1. sin(x)\n2. cos(x)\n3. exp(x)\n4. sh(x)\n----> "); scanf_s("%d",
&function_choose); printf("\n"); if (function_choose < 1 || function_choose>4) { printf("Choose something from list on
your screen, please"); break; }
                printf("Choose point x:\n----> "); scanf_s("%Lf", &x); printf("\n");
                printf("Choose count of experiments (0-25) (count of elements in the Taylor series):\n----> ");
scanf_s("%d", &Nmax); printf("\n"); if (Nmax < 0 || Nmax>25) { printf("Enter count of experiments in the range 0-25,
please"); break; }
                switch (function_choose)
                {
                case 1:
                {
                        printf("Etalon sin: %.20Lf\n", sinl(x));
                        serial_experiment(function_choose, x, Nmax);
                        break;
                }
                case 2:
                {
                        printf("Etalon cos: %.20Lf\n", cosl(x));
                        serial_experiment(function_choose, x, Nmax);
                        break;
                }
                case 3:
                {
                        printf("Etalon exp: %.20Lf\n", expl(x));
                        serial_experiment(function_choose, x, Nmax);
                        break;
                }
                case 4:
                {
                        printf("Etalon sh: %.20Lf\n", sinhl(x));
```

```
                    serial_experiment(function_choose, x, Nmax);
                    break;
                }
                break;
                }
        default: break;
        }
        return 0;
        }
        return 0;
}
```

Программа спрашивает у пользователя какой режим работы он хочет получить от программы: 1 или 2.

При работе в режиме 1:
Пользователь выбирает функцию, задаёт точку в которой необходимо вычислить значение, задает точность вычисления, задает количество слагаемых в разложении в ряд Тейлора.

На выходе пользователь получает вычисленное значение функции в заданной точке, эталонное значение функции в этой точке (благодаря встроенным функциям), разницу между эталонным значением и вычислениям с помощью наших функций и количество использованных для этого вычисления слагаемых в разложении в ряд Тейлора.

При работе в режиме 2:
Пользователь выбирает функцию; точку, в которой программа будет искать значение; Nmax - количество экспериментов (количество слагаемых в разложении в ряд Тейлора).

На выходе пользователь получает эталонное значение функции в этой точке; заданное им количество строк со значениями, полученными в результате использования Nmax количества слагаемых в разложении в ряд Тейлора; для каждой строчки выводится разница между вычисленным и эталонным значениями.

4. Результаты экспериментов

4.1. Работа программы в режиме 1 для sin(x) и exp(x)

Для Sin(x):

```
Choose mode of programm:
1. One-time calculation of the function at your point
2. Serial expirement
----> 1

Choose function:
1. sin(x)
2. cos(x)
3. exp(x)
4. sh(x)
----> 1

Choose point x:
----> 56

Choose calculation accuracy (0.000001 and more):
----> 0.00000001

Choose count of elements in the Taylor series (1-1000):
----> 555

Our calculation: -0.52155100208684457286
Etalon sin: -0.52155100208691185237
Difference: 0.00000000000006727952
Count of used elements in the Taylor series: 6
```

Для exp(x):

```
Choose mode of programm:
1. One-time calculation of the function at your point
2. Serial expirement
----> 1

Choose function:
1. sin(x)
2. cos(x)
3. exp(x)
4. sh(x)
----> 3

Choose point x:
----> 55

Choose calculation accuracy (0.000001 and more):
----> 0.000000000001

Choose count of elements in the Taylor series (1-1000):
----> 234

Our calculation: 769478526514201618284544.000000000000000000000
Etalon exp: 769478526514201752502272.000000000000000000000
Difference: 134217728.000000000000000000000
Count of used elements in the Taylor series: 174
```

4.2. Работа программы в режиме 2 для cos(x) и sh(x)

4.2.1. Для cos(x):

```
Choose mode of programm:
1. One-time calculation of the function at your point
2. Serial expirement
----> 2

Choose function:
1. sin(x)
2. cos(x)
3. exp(x)
4. sh(x)
----> 2

Choose point x:
----> 28

Choose count of experiments (0-25) (count of elements in the Taylor series):
----> 25

Etalon cos: -0.96260586631356659382
Count of elements    Our calculation               Difference
1                    -3.110586430745793            2.14798056443222673551
2                    -0.294432963307220            0.66817290300634635258
3                    -1.066169111983934            0.10356324567036789830
4                    -0.952873106952987            0.00973275936057926838
5                    -0.963222285196052            0.00061641888248553656
6                    -0.962577721686123            0.00002814462744338275
7                    -0.962606837444559            0.00000097113099251089
8                    -0.962605840087546            0.00000002622602035540
9                    -0.962605866883116            0.00000000056954962968
10                   -0.962605866303403            0.00000000001016353668
11                   -0.962605866313719            0.00000000000015232260
12                   -0.962605866313565            0.00000000000000133227
13                   -0.962605866313567            0.00000000000000066613
14                   -0.962605866313567            0.00000000000000066613
15                   -0.962605866313567            0.00000000000000066613
16                   -0.962605866313567            0.00000000000000066613
17                   -0.962605866313567            0.00000000000000066613
18                   -0.962605866313567            0.00000000000000066613
19                   -0.962605866313567            0.00000000000000066613
20                   -0.962605866313567            0.00000000000000066613
21                   -0.962605866313567            0.00000000000000066613
22                   -0.962605866313567            0.00000000000000066613
23                   -0.962605866313567            0.00000000000000066613
24                   -0.962605866313567            0.00000000000000066613
25                   -0.962605866313567            0.00000000000000066613
```

4.2.2. Для sh(x):

```
Choose mode of programm:
1. One-time calculation of the function at your point
2. Serial expirement
----> 2

Choose function:
1. sin(x)
2. cos(x)
3. exp(x)
4. sh(x)
----> 4

Choose point x:
----> 34

Choose count of experiments (0-25) (count of elements in the Taylor series):
----> 15

Etalon sh: 291730871263727.43750000000000000000
Count of elements    Our calculation                Difference
1                    6584.666666666666970           291730871257142.75000000000000000000
2                    385213.200000000011642         291730870878514.25000000000000000000
3                    10806512.831746030598879       291730860457214.62500000000000000000
4                    178126268.030335098505020      291730693137459.43750000000000000000
5                    1936504786.299143791198730     291728934758941.12500000000000000000
6                    14966540472.957752227783203    291715904723254.50000000000000000000
7                    86693784538.564178466796875    291644177479188.87500000000000000000
8                    391534571817.391479492187500   291339336691910.06250000000000000000
9                    1421932086713.076660156250000  290308939177014.37500000000000000000
10                   4257978580092.629394531250000  287472892683634.81250000000000000000
11                   10737167802121.804687500000000 280993703461605.62500000000000000000
12                   23220405703231.351562500000000 268510465560496.09375000000000000000
13                   43776848742665.296875000000000 247954022521062.12500000000000000000
14                   73041932675652.546875000000000 218688938588074.87500000000000000000
15                   109418746682677.562500000000000 182312124581049.87500000000000000000
```

5. Заключение

Во время выполнения задачи была разработана программа, позволяющая выполнять расчёт значений для набора функций (sin(x), cos(x), exp(x), sh(x)) в заданной пользователем точке с заданной пользователем погрешностью за счёт разложения функций в ряд Тейлора.

6. Приложение А: код программы

```
#include <math.h>
#include <stdio.h>
#define PI 3.14159265358979323846L

long double taylor_sin(long double x, int N, long double calculation_accuracy, int* used_count)
{
	long double TWO_PI = PI * 2.0L;
	x = fmodl(x, TWO_PI);
	if (x > PI)
	{
		x = x - TWO_PI;
	}
	else if (x < -PI)
	{
		x = x + TWO_PI;
	}

	long double result = x;
	long double term = x;
	*used_count = 1;

	for (int i = 1; i < N; i++)
	{
		term = -term * x * x / ((2.0L * i) * (2.0L * i + 1.0L));
		result = result + term;
		(*used_count)++;
		if (fabsl(term) < calculation_accuracy)
		{
			break;
		}
	}
	return result;

}
long double taylor_cos(long double x, int N, long double calculation_accuracy, int* used_count)
{
	long double TWO_PI = PI * 2.0L;
	x = fmodl(x, TWO_PI);
	if (x > PI)
	{
		x = x - TWO_PI;
	}
	else if (x < -PI)
	{
		x = x + TWO_PI;
	}


	long double result = 1.0L;
	long double term = 1.0L;
	*used_count = 1;

	for (int i = 1; i < N; i++)
	{
		term = -term * x * x / ((2.0L * i) * (2.0L * i - 1.0L));
		result = result + term;
		(*used_count)++;
		if (fabsl(term) < calculation_accuracy)
		{
			break;
		}
	}
	return result;
}
long double taylor_exp(long double x, int N, long double calculation_accuracy, int* used_count)
```

```
{
	long double result = 1.0L;
	long double term = 1.0L;
	*used_count = 1;

	for (int i = 1; i < N; i++)
	{
		term = term * (x / i);
		result = result + term;
		(*used_count)++;
		if (fabsl(term) < calculation_accuracy)
		{
			break;
		}
	}
	return result;
}
long double taylor_sh(long double x, int N, long double calculation_accuracy, int* used_count)
{
	long double result = x;
	long double term = x;
	*used_count = 1;

	for (int i = 1; i < N; i++) {
		term = term * x * x / ((2.0L * i) * (2.0L * i + 1.0L));
		result = result + term;
		(*used_count)++;
		if (fabsl(term) < calculation_accuracy)
		{
			break;
		}
	}
	return result;
}

void serial_experiment(int function_choice, long double x, int Nmax)
{

	printf("%-20s %-30s %-30s\n", "Count of elements", "Our calculation", "Difference");

	for (int n = 1; n <= Nmax; n++)
	{
		int used_count;
		long double intermediate_value = 0.0L;
		long double difference = 0.0L;

		switch (function_choice)
		{
		case 1:
			intermediate_value = taylor_sin(x, n + 1, 0.0L, &used_count);
			difference = fabsl(intermediate_value - sinl(x));
			printf("%-20d %-30.15Lf %-25.20Lf\n", n, intermediate_value, difference);
			break;
		case 2:
			intermediate_value = taylor_cos(x, n + 1, 0.0L, &used_count);
			difference = fabsl(intermediate_value - cosl(x));
			printf("%-20d %-30.15Lf %-25.20Lf\n", n, intermediate_value, difference);
			break;
		case 3:
			intermediate_value = taylor_exp(x, n + 1, 0.0L, &used_count);
			difference = fabsl(intermediate_value - expl(x));
			printf("%-20d %-30.15Lf %-25.20Lf\n", n, intermediate_value, difference);
			break;
		case 4:
			intermediate_value = taylor_sh(x, n + 1, 0.0L, &used_count);
```

```
                        difference = fabsl(intermediate_value - sinhl(x));
                        printf("%-20d %-30.15Lf %-25.20Lf\n", n, intermediate_value, difference);
                        break;
                }

        }

}

int main()
{
        int work_choose, function_choose;
        printf("Choose mode of programm:\n1. One-time calculation of the function at your point\n2. Serial
expirement\n----> "); scanf_s("%d", &work_choose); printf("\n"); if (work_choose != 1 && work_choose != 2) {
printf("Choose 1 or 2, please"); }
        switch (work_choose)
        {
        case 1:
        {
                int N, used_count;
                long double x, calculation_accuracy;
                printf("Choose function:\n1. sin(x)\n2. cos(x)\n3. exp(x)\n4. sh(x)\n----> "); scanf_s("%d",
&function_choose); printf("\n"); if (function_choose < 1 || function_choose>4) { printf("Choose something from list on
your screen, please"); break; }
                printf("Choose point x:\n----> "); scanf_s("%Lf", &x); printf("\n");
                printf("Choose calculation accuracy (0.000001 and more):\n----> "); scanf_s("%Lf",
&calculation_accuracy); printf("\n"); if (calculation_accuracy > 0.00001) { printf("Enter a accuracy more than 0.00001,
please"); break; }
                printf("Choose count of elements in the Taylor series (1-1000):\n----> "); scanf_s("%d", &N);
printf("\n"); if (N < 1 || N>1000) { printf("Select count in the range from 1 to 1000, please"); break; }
                switch (function_choose)
                {
                case 1:
                {
                        printf("Our calculation: %.20Lf\n", taylor_sin(x, N, calculation_accuracy, &used_count));
                        printf("Etalon sin: %.20Lf\n", sinl(x));
                        printf("Difference: %.20Lf\n", fabsl(taylor_sin(x, N, calculation_accuracy, &used_count) -
sinl(x)));
                        printf("Count of used elements in the Taylor series: %d", used_count);
                        break;

                }
                case 2:
                {
                        printf("Our calculation: %.20Lf\n", taylor_cos(x, N, calculation_accuracy, &used_count));
                        printf("Etalon cos: %.20Lf\n", cosl(x));
                        printf("Difference: %.20Lf\n", fabsl(taylor_cos(x, N, calculation_accuracy, &used_count) -
cosl(x)));
                        printf("Count of used elements in the Taylor series: %d", used_count);
                        break;
                }
                case 3:
                {
                        printf("Our calculation: %.20Lf\n", taylor_exp(x, N, calculation_accuracy, &used_count));
                        printf("Etalon exp: %.20Lf\n", expl(x));
                        printf("Difference: %.20Lf\n", fabsl(taylor_exp(x, N, calculation_accuracy, &used_count) -
expl(x)));
                        printf("Count of used elements in the Taylor series: %d", used_count);
                        break;
                }
                case 4:
                {
                        printf("Our calculation: %.20Lf\n", taylor_sh(x, N, calculation_accuracy, &used_count));
                        printf("Etalon sh: %.20Lf\n", sinhl(x));
```

```
                        printf("Difference: %.20Lf\n", fabsl(taylor_sh(x, N, calculation_accuracy, &used_count) -
sinhl(x)));
                        printf("Count of used elements in the Taylor series: %d", used_count);
                }
                default: break;
                }
                break;
        }
        case 2:
        {
                int Nmax;
                long double x;
                printf("Choose function:\n1. sin(x)\n2. cos(x)\n3. exp(x)\n4. sh(x)\n----> "); scanf_s("%d",
&function_choose); printf("\n"); if (function_choose < 1 || function_choose>4) { printf("Choose something from list on
your screen, please"); break; }
                printf("Choose point x:\n----> "); scanf_s("%Lf", &x); printf("\n");
                printf("Choose count of experiments (0-25) (count of elements in the Taylor series):\n----> ");
scanf_s("%d", &Nmax); printf("\n"); if (Nmax < 0 || Nmax>25) { printf("Enter count of experiments in the range 0-25,
please"); break; }
                switch (function_choose)
                {
                case 1:
                {
                        printf("Etalon sin: %.20Lf\n", sinl(x));
                        serial_experiment(function_choose, x, Nmax);
                        break;
                }
                case 2:
                {
                        printf("Etalon cos: %.20Lf\n", cosl(x));
                        serial_experiment(function_choose, x, Nmax);
                        break;
                }
                case 3:
                {
                        printf("Etalon exp: %.20Lf\n", expl(x));
                        serial_experiment(function_choose, x, Nmax);
                        break;
                }
                case 4:
                {
                        printf("Etalon sh: %.20Lf\n", sinhl(x));
                        serial_experiment(function_choose, x, Nmax);
                        break;
                }
                break;
                }
        default: break;
        }
        return 0;
        }
        return 0;
}
```