Министерство науки и высшего образования Российской Федерации

Федеральное государственное автономное образовательное
учреждение высшего образования
Национальный исследовательский Нижегородский государственный
университет им. Н.И. Лобачевского

Институт информационных технологий, математики и механики

Отчет по лабораторной работе

«Поиск значений функций при помощи рядов Тейлора»

Выполнил:
студент группы 3824Б1ПМ-1
Распопов И.С.

Проверила:

Бусько П.В.

Нижний Новгород
2025

Содержание

Введение

Для производства некоторых вычислений, требуется использовать точные значения функций в точках. Для определения точных значений мы используем уже готовые функция, которые встроенны в язык программирования. Для того, чтобы проверить точность этих значений, воспользуемся рядами Тейлора и проведем анализ.

Постановка задачи

Проверить, совпадают ли базовые значения из встроенных функций sinx, cosx, e**x, ln(x+1), с разложением их же через ряды Тейлора.

Руководство пользователя

Пользователь должен вводить только данные, которые предлагает ему программа!

Для корректной работы программы пользователь должен запустить файл main.c (при наличии компилятора для языка c/c++:

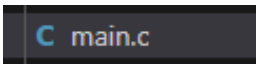

Рисунок.1 (стартовый файл main.cpp.)

При отсутствии компилятора для языка c/c++ main.exe:

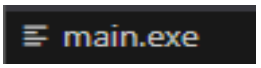

Рисунок.2(стартовый файл main.exe.)

Далее, пользователь сможет выбрать функция для проверки:

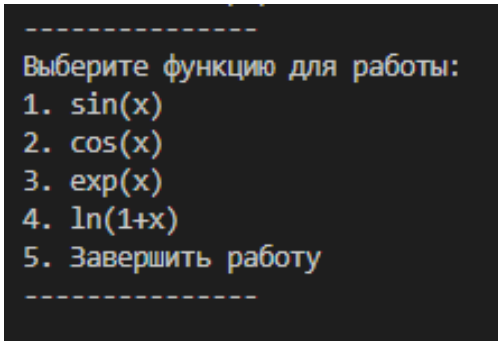

Рисунок.3(стартовое меню программы.)

Нажмите нужную цифру, указанную в столбце выбора, для выбора функция, с которой будет производится проверка.

При выборе цифры пользователь может ввести число x, которое будет использоваться:

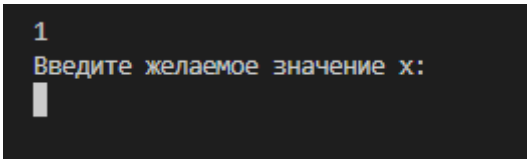

Рисунок.4(пример работы программы при выборе 1.)

После выбора числа, пользователь получает значения при разных количествах строк в разложении Тейлора:

```
Введите желаемое значение x:
1
Сравниваем значение: 0,841471, при количество строк: 2
Значение синуса из библиотеки: 0,841471, полученное значение: 0,833333
Сравниваем значение: 0,841471, при количество строк: 3
Значение синуса из библиотеки: 0,841471, полученное значение: 0,841667
Сравниваем значение: 0,841471, при количество строк: 4
Значение синуса из библиотеки: 0,841471, полученное значение: 0,841468
Сравниваем значение: 0,841471, при количество строк: 5
Значение синуса из библиотеки: 0,841471, полученное значение: 0,841471
Сравниваем значение: 0,841471, при количество строк: 6
Значение синуса из библиотеки: 0,841471, полученное значение: 0,841471
```

Рисунок.5 (меню программы после выбора 1 для функции sin(x).)

После завершения расчетов пользователь получает стартовое меню

При нажатии на 5 - программа будет завершена.

Описание программной реализации

Проект состоит из:

1. 3 файлов .cpp: main, print, function.
2. 3 файла .h: main, print, function.

Файл **main.cpp** представляет из себя главный файл, в котором содержатся функции:

1. **int main():**

Функция, в которой создаются:

1. Два объекта структуры: test , p_tes

2. Неизменяемый массив с длинами: int count_string_teilor_test[10]

3. Переменная для выбора функции: int choice_func

4. Переменная итогового значения вычислений: double rez_val

В функции реализовано основное меню и взаимодействие пользователя и программы.

2. **struct Values**

Структура значений точки X и длины строчек в ряде Тейлора.

Файл **function.cpp** представляет из себя файл, в котором содержатся функции:

1. int factorial(int n) - функция возведения числа в факториал.

a. Функция получает:

1. число, факториал которого нужен - int n;

b. Использует дополнительные библиотеки: math.h s и stdio.h.

c. Возвращает целочисленное значение факториала.

2. double teilor_sin(double x, int count_string) - функция, которая раскладывает синус, используя ряд Тейлора

a. Функция получает:

1. значения x: double x.

2. длину строки, до которой нужно раскладывать:int count_string

b. Использует дополнительные библиотеки: math.h s и stdio.h.

c. Возвращает значение типа double.

3. double teilor_cos(double x, int count_string) - функция, которая раскладывает косинус, используя ряд Тейлора

a. Функция получает:

1. значения x: double x.

2. длину строки, до которой нужно раскладывать:int count_string

b. Использует дополнительные библиотеки: math.h s и stdio.h.

c. Возвращает значение типа double.

4. double teilor_exp (double x, int count_string) - функция, которая раскладывает e**x, используя ряд Тейлора

a. Функция получает:

1. значения x: double x.

2. длину строки, до которой нужно раскладывать:int count_string

b. Использует дополнительные библиотеки: math.h s и stdio.h.

c. Возвращает значение типа double.

5. double teilor_ln (double x, int count_string) - функция, которая раскладывает ln(x+1), используя ряд Тейлора

a. Функция получает:

1. значения x: double x.

2. длину строки, до которой нужно раскладывать:int count_string

b. Использует дополнительные библиотеки: math.h s и stdio.h.

c. Возвращает значение типа double.

Файл **print.cpp** содержит функции:

1. void print_menu_func() - функция, которая выводит основное меню программы в консоль.

a. Функция ничего не получает.

b. Использует дополнительные библиотеки: locale.h и stdio.h.

c. Ничего не возвращает.

2. void print_required_val() - функция, которая вводит строку пользователю, с просьбой ввести значение для X.

a. Функция ничего не получает.

b. Использует дополнительные библиотеки: locale.h и stdio.h.

c. Ничего не возвращает.

3. void print_test_teilor(int count, double value) - функция, которая выводит сравниваемое значение с количеством строк для сравнения

a. Функция получается:

1. Количество строк - int count.

2. Нужное значение функции - double value.

b. Использует дополнительные библиотеки: locale.h и stdio.h.

c. Ничего не возвращает.

4. void print_rez_val_sin(double value_sin, double rez_val) - функция, которая выводит в консоль значение синуса и результат вычисления его, используя ряды Тейлора.

a. Функция получается:

1. Нужное значение синуса в точке - double value_sin.

2. Значения, полученное вычислением через ряды - double rez_val.

b. Использует дополнительные библиотеки: locale.h и stdio.h.

c. Ничего не возвращает.

5. void print_rez_val_cos(double value_cos, double rez_val) - функция, которая выводит в консоль значение косинуса и результат вычисления его, используя ряды Тейлора.

a. Функция получается:

1. Нужное значение косинуса в точке - double value_cos.

2. Значения, полученное вычислением через ряды - double rez_val.

b. Использует дополнительные библиотеки: locale.h и stdio.h.

c. Ничего не возвращает.

6. void print_rez_val_exp(double value_exp, double rez_val) - функция, которая выводит в консоль значение e**x и результат вычисления его, используя ряды Тейлора.

a. Функция получается:

1. Нужное значение косинуса в точке - double value_exp.

2. Значения, полученное вычислением через ряды - double rez_val.

b. Использует дополнительные библиотеки: locale.h и stdio.h.

c. Ничего не возвращает.

7. void print_rez_val_ln(double value_ln, double rez_val) - функция, которая выводит в консоль значение ln(x+1) и результат вычисления его, используя ряды Тейлора.

a. Функция получается:

1. Нужное значение косинуса в точке - double value_ln.

2. Значения, полученное вычислением через ряды - double rez_val.

b. Использует дополнительные библиотеки: locale.h и stdio.h.

c. Ничего не возвращает.

Результаты экспериментов

Данные эксперимента при запуске программы и вводе числа x = 1.

Для Sin:

Требуемое значение - 0,841471.

Результат при 2 строчках разложения: 0,833333.

Результат при 3 строчках разложения: 0,841667.

Результат при 4 строчках разложения: 0,841468.

Результат при 5 строчках разложения: 0,841471.

Результат при 6 строчках разложения: 0,841471.

Имеем, что для синуса при увеличении количества строк разложения, значение становится ближе к требуемому.

Для Cos:

Требуемое значение - 0,540302.

Результат при 2 строчках разложения: 0,5.

Результат при 3 строчках разложения: 0,541667.

Результат при 4 строчках разложения: 0,540278.

Результат при 5 строчках разложения: 0,540303.

Результат при 6 строчках разложения: 0,540302.

Имеем, что для косинуса при увеличении количества строк разложения, значение становится ближе к требуемому.

Для e^x :

Требуемое значение - 2,71828

Результат при 2 строчках разложения: 2.

Результат при 3 строчках разложения: 2,5.

Результат при 4 строчках разложения: 2,66667.

Результат при 5 строчках разложения: 2,70833.

Результат при 10 строчках разложения: 2,71828.

Имеем, что для экспоненты в степени при увеличении количества строк разложения, значение становится ближе к требуемому.

Для ln(x+1):

Требуемое значение - 0,693147

Результат при 2 строчках разложения: 0,5

Результат при 3 строчках разложения: 0,833333.

Результат при 10 строчках разложения: 0,645635.

Результат при 1.000.000 строчках разложения: 0, 693147.

Имеем, что для логарифма при увеличении количества строк разложения, значение становится ближе к требуемому.

По данным эксперимента можно выделить, что для ln(x+1) и e^x требуется больше разложения, чем для оставшихся функций.

Заключение

Исходя из проведенного эксперимента и созданной программы, мы можем сделать вывод, что при увеличении количества строк разложения, искомое значение становится более точным.

Литература

1. Керниган Б., Ритчи Д., Фьюэр А. Язык программирования СИ //М.: Финансы и статистика. – 1992.
2. Кнут Д. Э. Искусство программирования: Сортировка и поиск. – Издательский дом Вильямс, 2000. – Т. 3.

Приложение

```
#include "stdio.h"
#include "math.h"
#include "print.cpp"
#include "function.cpp"

struct Values
{
   double val_x_in_t;
   int val_count;
};


int main()
{
   struct Values  test;
   struct Values *p_test = &test;

   const int count_string_teilor_test[10] = {2,3,4,5,6,7,8,9,10,1000000};



   int choise_func;
   double rez_val;

   print_menu_func();
```

```
    scanf("%i", &choise_func);
    while (choise_func != 5)
    {


        switch (choise_func)
        {
        case (1): //sin(x)
            print_required_val();
            scanf("%lg",&test.val_x_in_t);


            p_test -> val_count = count_string_teilor_test[0];
            print_test_teilor(test.val_count, sin(test.val_x_in_t));
            rez_val = teilor_sin(test.val_x_in_t,test.val_count); // значение при 2
строчках
            print_rez_val_sin(sin(test.val_x_in_t),rez_val);


            p_test -> val_count = count_string_teilor_test[1];
            print_test_teilor(test.val_count, sin(test.val_x_in_t));
            rez_val = teilor_sin(test.val_x_in_t,test.val_count);   //  значение при 3
строчках
            print_rez_val_sin(sin(test.val_x_in_t),rez_val);


            p_test -> val_count = count_string_teilor_test[2];
            print_test_teilor(test.val_count, sin(test.val_x_in_t));
            rez_val = teilor_sin(test.val_x_in_t,test.val_count);    // значение при 4
строчках
```

```
            print_rez_val_sin(sin(test.val_x_in_t),rez_val);

            p_test -> val_count = count_string_teilor_test[3];
            print_test_teilor(test.val_count, sin(test.val_x_in_t));
            rez_val = teilor_sin(test.val_x_in_t,test.val_count);    // значение при 5
строчках
            print_rez_val_sin(sin(test.val_x_in_t),rez_val);

            p_test -> val_count = count_string_teilor_test[4];
            print_test_teilor(test.val_count, sin(test.val_x_in_t));
            rez_val = teilor_sin(test.val_x_in_t,test.val_count);    // значение при 6
строчках
            print_rez_val_sin(sin(test.val_x_in_t),rez_val);

            break;

        case(2): // cos(x)
            print_required_val();
            scanf("%lg", &test.val_x_in_t);

            p_test -> val_count = count_string_teilor_test[0];
            print_test_teilor(test.val_count, cos(test.val_x_in_t));
            rez_val = teilor_cos(test.val_x_in_t,test.val_count);
            print_rez_val_cos(cos(test.val_x_in_t),rez_val); // значение при 2 строчках

            p_test -> val_count = count_string_teilor_test[1];
            print_test_teilor(test.val_count, cos(test.val_x_in_t));
```

```
    rez_val = teilor_cos(test.val_x_in_t,test.val_count);
    print_rez_val_cos(cos(test.val_x_in_t),rez_val); // Значение при 3 строчках

    p_test -> val_count = count_string_teilor_test[2];
    print_test_teilor(test.val_count, cos(test.val_x_in_t));
    rez_val = teilor_cos(test.val_x_in_t,test.val_count);
    print_rez_val_cos(cos(test.val_x_in_t),rez_val); // значение при 4 строчках

    p_test -> val_count = count_string_teilor_test[3];
    print_test_teilor(test.val_count, cos(test.val_x_in_t));
    rez_val = teilor_cos(test.val_x_in_t,test.val_count);
    print_rez_val_cos(cos(test.val_x_in_t),rez_val); // значение при 5 строчках

    p_test -> val_count = count_string_teilor_test[4];
    print_test_teilor(test.val_count, cos(test.val_x_in_t));
    rez_val = teilor_cos(test.val_x_in_t,test.val_count);
    print_rez_val_cos(cos(test.val_x_in_t),rez_val); // значение при 6 строчках

    p_test -> val_count = count_string_teilor_test[5];
    print_test_teilor(test.val_count, cos(test.val_x_in_t));
    rez_val = teilor_cos(test.val_x_in_t,test.val_count);
    print_rez_val_cos(cos(test.val_x_in_t),rez_val); // значение при 7 строчках
    break;

case(3): // exp(x)
    print_required_val();
```

```
scanf("%lg", &test.val_x_in_t);

p_test -> val_count = count_string_teilor_test[0];
print_test_teilor(test.val_count, exp(test.val_x_in_t));
rez_val = teilor_exp(test.val_x_in_t,test.val_count);
print_rez_val_exp(exp(test.val_x_in_t),rez_val); //значение при 2 строчках

p_test -> val_count = count_string_teilor_test[1];
print_test_teilor(test.val_count, exp(test.val_x_in_t));
rez_val = teilor_exp(test.val_x_in_t,test.val_count);
print_rez_val_exp(exp(test.val_x_in_t),rez_val); //значение при 3 строчках

p_test -> val_count = count_string_teilor_test[2];
print_test_teilor(test.val_count, exp(test.val_x_in_t));
rez_val = teilor_exp(test.val_x_in_t,test.val_count);
print_rez_val_exp(exp(test.val_x_in_t),rez_val); //значение при 4 строчках

p_test -> val_count = count_string_teilor_test[3];
print_test_teilor(test.val_count, exp(test.val_x_in_t));
rez_val = teilor_exp(test.val_x_in_t,test.val_count);
print_rez_val_exp(exp(test.val_x_in_t),rez_val); //значение при 5 строчках

p_test -> val_count = count_string_teilor_test[4];
print_test_teilor(test.val_count, exp(test.val_x_in_t));
rez_val = teilor_exp(test.val_x_in_t,test.val_count);
print_rez_val_exp(exp(test.val_x_in_t),rez_val); //значение при 6 строчках
```

```
        p_test -> val_count = count_string_teilor_test[8];
        print_test_teilor(test.val_count, exp(test.val_x_in_t));
        rez_val = teilor_exp(test.val_x_in_t,test.val_count);
        print_rez_val_exp(exp(test.val_x_in_t),rez_val); // значение при 10 строчках

        break;
    case(4): // ln(1+x)
        print_required_val();
        scanf("%lg", &test.val_x_in_t);

        p_test -> val_count = count_string_teilor_test[0];
        print_test_teilor(test.val_count, log(test.val_x_in_t+1));
        rez_val = teilor_ln(test.val_x_in_t,test.val_count);
        print_rez_val_ln(log(test.val_x_in_t+1),rez_val); // значение при 2 строчках

        p_test -> val_count = count_string_teilor_test[1];
        print_test_teilor(test.val_count, log(test.val_x_in_t+1));
        rez_val = teilor_ln(test.val_x_in_t,test.val_count);
        print_rez_val_ln(log(test.val_x_in_t+1),rez_val); // значение при 3 строчках

        p_test -> val_count = count_string_teilor_test[8];
        print_test_teilor(test.val_count, log(test.val_x_in_t+1));
        rez_val = teilor_ln(test.val_x_in_t,test.val_count);
        print_rez_val_ln(log(test.val_x_in_t+1),rez_val); // значение при 10
строчках
```

```
            p_test -> val_count = count_string_teilor_test[9];
            print_test_teilor(test.val_count, log(test.val_x_in_t+1));
            rez_val = teilor_ln(test.val_x_in_t,test.val_count);
            print_rez_val_ln(log(test.val_x_in_t+1),rez_val); // значение при 1kk
строчках



            break;
        }
        print_menu_func();
        scanf("%i", &choise_func);
    }
}
```