

Московский Авиационный Институт
(Национальный Исследовательский Университет)

Факультет прикладной математики и информатики
Кафедра вычислительной математики и программирования

**Отчёт по 3 курсовому проекту
По дисциплине
«Фундаментальная**

информатика»

1 семестр

На тему:

**«Вещественный тип. Приближённые вычисления.
Табулирование функций»**

| | |
|-----------------------|---------------|
| Студент: | Серякова А.А. |
| Группа: | М8О-109Б-22 |
| Преподаватель: | Сысоев М.А. |
| Подпись: | |
| Оценка: | |
| Дата: | 28.12.2022 |

1. Цель работы

Составить программу на Си, которая печатает таблицу значений элементарной функции, вычисленной двумя способами: по формуле Тейлора и с помощью встроенных функций языка программирования. В качестве аргументов таблицы взять точки разбиения отрезка $[a, b]$ на n равных частей ($n + 1$ точка включая концы отрезка), находящихся в рекомендованной области хорошей точности формулы Тейлора. Вычисления по формуле Тейлора проводить по экономной в сложностном смысле схеме с точностью $E \cdot k$, где E - машинное эpsilon аппаратно реализованного вещественного типа для данной ЭВМ, а k - экспериментально подбираемый коэффициент, обеспечивающий приемлемую сходимость. Число итераций должно ограничиваться сверху числом порядка 100.

2. Задание

| | | | | |
|---|--|-----|-----|-------------------------------|
| 9 | $-(1 + \frac{2}{3}) - (1 + \frac{2}{3^2})x - \dots - (1 + \frac{2}{3^{n+1}})x^n$ | 0.0 | 0.5 | $\frac{3x - 5}{x^2 - 4x + 3}$ |
|---|--|-----|-----|-------------------------------|

Входные данные

Единственная строка содержит два целых числа N ($0 \leq N \leq 100$) – число разбиений отрезка на равные части.

Выходные данные

Программа должна вывести значение эpsilon, а затем $N+1$ строку. В каждой строке должно быть значение x , для которого вычисляется функция, число $A1$ — значение, вычисленное с помощью формулы Тейлора, $A2$ – значение, вычисленное с помощью встроенных функций языка, i – количество итераций. $A1$, $A2$ должны быть выведены с приемлемой точностью.

3. Выполнение

Теоретическая часть:

Формула Тейлора — формула разложения функции в бесконечную сумму степенных функций. Формула широко используется в приближённых вычислениях, так как позволяет приводить трансцендентных функций к более простым. Сама она является следствием теоремы Лагранжа о среднем значении

дифференцируемой функции. В случае $a=0$ формула называется рядом Маклорена.

$$\sum_{n=0}^{\infty} \frac{f^{(n)}(a)}{n!} (x-a)^n = f(a) + f'(a)(x-a) + \frac{f''(a)}{2!}(x-a)^2 + \dots + \frac{f^{(k)}(a)}{k!}(x-a)^k + \dots$$

Func – функция для вычисления значения стандартной функции.

FuncTaylor – функция для вычисления ряда Тейлора.

```
double Func(double x){
    return ((3*x)-5)/(pow(x,2)-(4*x)+3);
}

double FuncTaylor(double x, unsigned p){
    return (-(1.0+(2.0/pow(3.0,p+1)))*pow(x,p));
}
```

calculateTaylor – это функция, которая принимает на вход границы отрезка, количество итераций и две функции, то есть заданную нами и ряд Тейлора.

Условие выполняется до тех пор, пока мы не достигнем конца отрезка.

```
void calculateTaylor(double a, double b, unsigned n, double (*Ftalon)(double), double (*Taylor)(double, unsigned)){
    double x = a, i = 0, step, epsilon = 1e-7;
    step = (b - a) / n;
    printf("%.14f", epsilon);
    printf("\n=====\\n");
    printf("\\ step\\t x\\t FuncTaylor\\t Function\\t i\\t \\n");
    printf("\\-----\\n");
    n = 0;
    while (x <= 0.5){
        int i = 0;
        double currentValueFunc = Ftalon(x);
        double currentValueFuncTaylor = Taylor(x, i);
        double currentValue = 0, difference = 0;
        do{
            currentValue += Taylor(x, i);
            i++;
            difference = currentValueFunc - currentValue;
        } while ((difference > epsilon || difference < -epsilon) && i < 100);
        printf("\\ %d\\t %.2f\\t %.17f\\t %.17f\\t %d\\t \\n", n, x, currentValue, currentValueFunc, i);
        x += step;
        n++;
    }
    printf("\\=====\\n");
}
```

Это основная функция, где мы задаём начало и конец отрезка, считываем количество итераций, передаём значения и функции в calculateTeylor.

```
int main(){
    float a = 0.0, b = 0.5;
    int n = 0;
    printf("Enter the number of steps: ");
    scanf("%d", &n);
    calculateTeylor(a, b, n, &Func, &FuncTeylor);
    return 0;
}
```

Весь код:

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

double Func(double x){
    return ((3*x)-5)/(pow(x,2)-(4*x)+3);
}

double FuncTeylor(double x, unsigned p){
    return -(1.0+(2.0/pow(3.0,p+1)))*pow(x,p);
}

void calculateTeylor(double a,double b, unsigned n, double
(*Etalon)(double), double (*Taylor)(double, unsigned)){
    double x = a, i = 0, step, epsilon = 1e-7;
    step = (b - a) / n;
    printf("%.14f", epsilon);

    printf("\n=====
=====\\n");
    printf("|| step\\t x\\t FuncTeylor          Function
i      ||\\n");
    printf("||-----
-----||\\n");
    n = 0;
    while (x <= 0.5){
        int i = 0;
        double currentValueFunc = Etalon(x);
        double currentValueFuncTeylor = Taylor(x,i);
        double currentValue = 0, difference = 0;
        do{
            currentValue += Taylor(x, i);
            i++;
            difference = currentValueFunc - currentValue;
        } while ((difference > epsilon || difference < -epsilon) && i <
100);
```

```

        printf("||  %d\t %.2f\t %.17f\t %.17f\t %d\t||\n", n, x,
currentValue, currentValueFunc, i);
        x += step;
        n++;
    }

printf("=====\n");
}

int main(){
    float a = 0.0, b = 0.5;
    int n = 0;
    printf("Enter the number of steps: ");
    scanf("%d", &n);
    calculateTeylor(a, b, n, &Func, &FuncTaylor);
    return 0;
}

```

4. Вывод

В данной работе я научилась вызывать функции внутри других функций. Данный опыт очень ценен и обязательно пригодится мне в будущих проектах.