МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Московский Авиационный Институт»
(Национальный Исследовательский Университет)

Институт: №8 «Информационные технологии и прикладная
математика»
Кафедра: 806 «Вычислительная математика и программирование»

Курсовая работа
по курсу «Фундаментальная информатика»
I семестр
Задание 3
«Вещественный тип. Приближенные вычисления. Табулирование
функций»

Группа	М8О-109Б-22
Студент	Ефименко К.И.
Преподаватель	Сысоев М.А.
Оценка	
Дата	

Москва, 2022

Постановка задачи

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 $\varepsilon * 10^k$, где ε - машинное эпсилон аппаратно реализованного вещественного типа для данной ЭВМ, а k – экспериментально подбираемый коэффициент, обеспечивающий приемлемую сходимость. Число итераций должно ограничиваться сверху числом порядка 100. Программа должна сама определять машинное ε и обеспечивать корректные размеры генерируемой таблицы.

Вариант 24:

Ряд Тэйлора:

$$\left| 1+\frac{x^2}{1}+\frac{x^4}{2}+\ldots+\frac{x^{2n}}{n!}\right.$$

Функция:

$$e^{x^2}$$

Значения a и b: 0.0 и 1.0

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.

$$\sum_{n=0}^{k} \frac{f^{(n)}(a)}{n!}(x-a)^n = f(a)+f^{(1)}(a)(x-a)+\frac{f^{(2)}(a)}{2!}(x-a)^2+\ldots+\frac{f^{(k)}(a)}{k!}(x-a)^k$$

Машинное эпсилон — числовое значение, меньше которого невозможно задавать относительную точность для любого алгоритма, возвращающего вещественные числа. Абсолютное значение для машинного эпсилон зависит от разрядности сетки применяемой ЭВМ и от разрядности используемых при расчёте чисел. Формально это машинное эпсилон определяют как число, удовлетворяющее равенству $1 + \varepsilon = 1$. Фактически, два отличных от нуля числа являются равными с точки зрения машинной арифметики, если их модуль разности меньше или не превосходит машинное эпсилон.

В языке Си машинные эпсилон определено для следующих типов: float – $1.19 * 10^{-7}$, double – $2.20 * 10^{-16}$, long double – $1.08 * 10^{-19}$.

Описание алгоритма

Рассмотрим алгоритм решения. Сперва нужно найти машинное эпсилон, на котором будет основываться точность вычисления. Это можно сделать просто деля 1 на 2.

Для каждой N+1 строки нужно просуммировать i членов формулы Тейлора, пока $|A_1-A_2|> \varepsilon$. Для этого просто ищем каждый новый член из формулы Тэйлора и суммируем с результатом

Использованные в программе переменные

Название переменной	Тип переменной	Смысл переменной
n	int	То самое число N, на которое нужно разбить отрезок
k	int	То самое число K, используемое для вычисления точности.
FLT_EPSILON	float	То самое машинное эпсилон. 1.192092896e-07F
step	double	Формально разница между предыдущим значением из отрезка и следующим, если отрезок разбит на n равных частей.
currentX	double	Переменная, для которой будем производить вычисления
Taylor (currentX, i)	double	То самое значение A_1, вычисленное с помощью формулы Тейлора
func(currentX)	double	То самое значение A_2, вычисленное с помощью встроенных функций языка
i	double§	Счётчик члена формулы Тейлора + кол-во итераций

Исходный код программы:

```c
#include <stdio.h>
#include <float.h>
#include <math.h>

int fact(int n) {
    int res = 1;

    for (int i = 1; i <= n; i++)
        res *= i;

    return res;
}

double func(double x) {
    return exp(x * x);
}

double taylor(double x, int n) {
    double res = 0;
    for (int i = 0; i <= n; i++)
        res += (((double) (pow(x, 2*i))) / ((double) fact( n: i)));

    return res;
}

int main() {
    double sum = 0.0;
    double a = 0.0;
    double b = 1.0;

    int n;
    scanf("%d", &n);
    printf("Machine epsilon: %Lg\n", LDBL_EPSILON);

```

```c
    int n;
    scanf("%d", &n);
    printf("Machine epsilon: %Lg\n", LDBL_EPSILON);

    printf("   x             sum               function         iteration \n");

    double currentX;
    double step = (b - a) / (double) n;

    for (int i = 1; i <= n; ++i) {
        currentX = a + step * (double) i;
        sum = taylor( x: currentX,  n: i);

        printf(" %.3f    %.16f   %.16f        %d       \n", currentX, sum, func( x: currentX), i);

        if (fabsl(func( x: currentX) - sum) < LDBL_EPSILON) break;
    }

    return 0;
}
```

Входные данные

Единственная строка содержит одно целое число N (0≤N≤100) – число разбиений отрезка на равные части

Выходные данные

Программа должна вывести значение машинного эпсилон, а затем N+1 строку.

В каждой строке должно быть значение x, для которого вычисляется

функция, число A_1 — значение, вычисленное с помощью формулы Тейлора, A_2 – значение, вычисленное с помощью встроенных функций языка, i – количество итерация, требуемых для вычисления, и Δ – разница значений A_1 и A_2 по модулю. A_1, A_2 и Δ должны быть выведены с точностью 16 знаков после запятой.

Протокол исполнения и тесты

Тест №1

Ввод:

5

Вывод:

```
Machine epsilon: 1.0842e-19
   x             sum                  function         iteration
 0.200    1.0400000000000000   1.0408107741923882         1
 0.400    1.1728000000000001   1.1735108709918103         2
 0.600    1.4325760000000001   1.4333294145603404         3
 0.800    1.8954811733333337   1.8964808793049517         4
 1.000    2.7166666666666663   2.7182818284590451         5

Process finished with exit code 0
```

Тест №2

Ввод:

15

Вывод
:

```
Machine epsilon: 1.0842e-19
   x             sum                  function         iteration
 0.007    1.0000444444444445   1.0000444454321133         1
 0.013    1.0001777935802469   1.0001777935811833         2
 0.020    1.0004000800106665   1.0004000800106678         3
 0.027    1.0007113640105603   1.0007113640105603         4

Process finished with exit code 0
```

Тест №3

Ввод:

500

Вывод:

```
Machine epsilon: 1.0842e-19
   x              sum                 function          iteration
 0.002    1.0000039999999999   1.0000040000079999          1
 0.004    1.0000160001280001   1.0000160001280007          2
 0.006    1.0000360006480078   1.0000360006480078          3

Process finished with exit code 0
```

Вывод

В работе описано определение машинного эпсилон, приведены его значения для разных переменных языка Си, описана формула Тейлора и составлен алгоритм реализации вычисления значения функции с заданной точностью для заданного числа точек на отрезке. На основе алгоритма составлена программа на языке Си, проведено её тестирование, составлен протокол исполнения программы. Работа не понравилась, так как, по моему мнению, непонятно сформулировано условие. Также непонятно практическое применение данной работы.

Список литературы

1. Машинный ноль – URL: https://ru.wikipedia.org/wiki/Машинный_ноль
2. Ряд Тейлора – URL: https://ru.wikipedia.org/wiki/Ряд_Тейлора