Московский Авиационный Институт

(Национальный Исследовательский Университет)

Кафедра вычислительной математики и программирования

Курсовая работа

По курсу «Фундаментальная информатика»

I семестр

Задание №3 «Вещественный тип. Приближенные вычисления»

Выполнил студент

1-го курса, 105-ой группы

Махмудов О. С.

____

(подпись)

Научный руководитель

Доцент кафедры 806

Сластушенский Ю. В.

____

(подпись)

Работа защищена

«__»___________ 2019

Оценка___________

ПОСТАНОВКА ЗАДАЧИ

Составить программу на языке Си, которая печатает таблицу значений 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 производятся с точностью ε*10k, где ε - машинное эпсилон аппаратно реализованного вещественного типа для данной ЭВМ, а k – коэффициент точности, вводимый с клавиатуры. Число итераций для вычислений по формуле Тейлора ограничивается 100. Программа должна сама определять машинное ε и обеспечивать корректные размеры генерируемой таблиц значений.

13 вариант

13	$x - \frac{x^3}{3!} + \ldots + (-1)^n \frac{x^{2n+1}}{(2n+1)!}$	0.0	1.0	$\sin x$

ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

Формула Тейлора

$$f(x) = f(a) + \frac{f'(a)}{1!}(x-a) + \frac{f''(a)}{2!}(x-a)^2 + \ldots + \frac{f^{(n)}(a)}{n!}(x-a)^n + R_n(x)$$

где Rn(x) называется остаточным членом.

Формула Тейлора показывает поведение функции в окрестности некоторой точки. Она часто используется при доказательстве теорем в дифференциальном исчислении.

Формула Тейлора позволяет вычислить приближенное значение функции в данной точке с помощью бесконечного сложения элементов ряда Тейлора, вычисляемых по формуле $\frac{f^{(n)}(a)}{n!}(x-a)^n$.

Машинное эпсилон

Машинное эпсилон — числовое значение, меньше которого невозможно задавать относительную точность для любого алгоритма, возвращающего вещественные числа. Абсолютное значение «машинного эпсилон» зависит от разрядности сетки применяемой ЭВМ, типа (разрядности) используемых при расчетах чисел, и от принятой в конкретном трансляторе структуры представления вещественных чисел (количества бит, отводимых на мантиссу и на порядок). Формально машинное эпсилон обычно определяют как минимальное из чисел eps, для которого 1+eps>1 при машинных расчетах с числами данного типа[3]. Альтернативное определение — максимальное положительное eps, для которого справедливо равенство 1+eps=1.

Практическая важность машинного эпсилон связана с тем, что два (отличных от нуля) числа являются одинаковыми с точки зрения машинной арифметики, если их относительная разность по модулю меньше (при определении первого типа) или не превосходит (при определении второго типа) машинного эпсилон.

ОБОРУДОВАНИЕ И СП

Используется ноутбук с операционной системой семейства Windows 10-ой версии (64 bit), с процессором Intel Core i3 8130U и оперативной памятью в 4 ГБ. Для компиляции и отладки программы используется подсистема Linux для Windows, после чего программа компилируется с помощью GNU CC.

ОПИСАНИЕ ФУНКЦИЙ И ПЕРЕМЕННЫХ

Функции

double eps – вычисляет машинное эпсилон. К сожалению, невозможно создать переменную, равную значению этой функции в глобальной области, поэтому приходится пользоваться этой функцией в каждом месте, где необходим машинный эпсилон. Также в языке Си существует константа, равная машинному эпсилон, но мы не будем ее использовать, так как значение эпсилон зависит от аппаратной части.

double anti_fac – так как факториал может быть слишком велик, то в данной конкретной ситуации имеет смысл вычислять число, обратное факториалу, что и делает данная функция.

double formula_taylora - вычисляет сумму n членов формулы Тейлора, пока n+1 член не окажется по значению меньше машинного эпсилона. Выдает результат работы функции и запоминает количество итераций.

sin(x) – встроенная функция языка, вычисляющая значение синуса в x.

Переменные функции main

int n – число подотрезков, на которое разбивается отрезок, данный в условии, вводится пользователем с помощьюи scanf

int *i – для подсчета количества итераций в функции **double formula_taylora** и вывода ее в таблицу

int i1 – обнуление переменной **i** для того, чтобы она работала корректно в программе и после подачи переменной в функцию **double formula_taylora**

double k – коэффициент точности, вводится пользователем с помощью scanf

double a – левая граница отрезка, данного в условии

double b – правая граница отрезка, данного в условии

double part – разница между значениями точек после деления отрезка [a,b] на n равных частей

double x – значение каждой точки на отрезке [a,b] поделенной на n равных частей

ВХОДНЫЕ ДАННЫЕ И ВЫХОДНЫЕ ДАННЫЕ

Программа считывает из стандартного входного потока два числа: n и k.

Число n определяет количество подотрезков, на которое разбивается заданный в условии отрезок. В результате разбиения, получаем n+1 точек, в которых необходимо высчитать значение функции.

Число k определяет точность вычислений по формуле Тейлора. Вычисления производятся, пока модуль текущего элемента ряда Тейлора больше, чем значение выражения eps*pow(10,k).

Теперь о формате вывода. Сначала программа выводит машинное эпсилон для типа double. После этого программа выводит таблицу, состоящую из n+1 строк. В каждой строке выводится текущий x, значение функции, вычисленное с помощью формулы Тейлора, значение функции, вычисленное с помощью языковых средств, и количество итераций, понадобившееся, чтобы вычислить значение функции с приемлемой точностью, используя формулу Тейлора.

ПРОГРАММА

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

double eps() {
	double e = 1;
	while (1 + e != 1) {
		e /= 2;
	}
	return e;
}

double anti_fac(double k) {
	double z = 1, f = 1;
	while (k != 0) {
		f = f * z;
		k--;
		z++;
	}
	f = 1 / f;
```

```
	return f;
}

double formula_taylora(double x, double e, int* i, double k) {
	double m1 = 1, m2 = x, m3 = 1, res = m1 * m2 * m3, t = 0, l;
	*i = 1;
	while (fabs((m1 * m2 * m3)) > (e * pow(10, k))) {
		if (*i == 100) {
			printf("Iteration = 100, value = %lf\n", fabs((sin(x) - res)));
			break;
		}
		t++;
		m1 = pow(-1, t);
		l = 2 * t + 1;
		m2 = pow(x, l);
		m3 = anti_fac(l);
		res += m1 * m2 * m3;
		(*i)++;
	}
	return res;
}

int main() {
	int n, *i, i1 = 0;
	i = &i1;
	double a = 0.0, b = 1.0, x, part, k;
		printf("eps = %e\n", eps());
		scanf("%d\n", &n);
		scanf("%lf", &k);
		part = (b - a) / n;
	printf("---------------------------------------------------------------------------\n");
	printf("   x    |       tayor_func          |        value          | iterations\n");
	for (int j = 0; j <= n; j++) {
		x = a + part * j;
		printf("%.6lf|    %.18f    | %.18f |     %d\n", x, formula_taylora(x,
eps(), i, k), sin(x), *i);
	}
	printf("---------------------------------------------------------------------------\n");
	return 0;
}
```

ПРОТОКОЛ РАБОТЫ

```
Admin@LAPTOP-Q5U6S2UH:/mnt/c/Users/Admin/Desktop/Все для вуза$ gcc kursach3.c -lm
Admin@LAPTOP-Q5U6S2UH:/mnt/c/Users/Admin/Desktop/Все для вуза$ ./a.out
eps = 1.110223e-16
20
1
---------------------------------------------------------------------------------------
    x     |         tayor_func        |           value           | iterations
0.000000|   0.000000000000000000  | 0.000000000000000000 |     0
0.050000|   0.049979169270678338  | 0.049979169270678331 |     1
0.100000|   0.099833416646828169  | 0.099833416646828155 |     5
0.150000|   0.149438132473599272  | 0.149438132473599244 |     6
0.200000|   0.198669330795061244  | 0.198669330795061216 |     6
0.250000|   0.247403959254522937  | 0.247403959254522937 |     6
0.300000|   0.295520206661339602  | 0.295520206661339602 |     7
0.350000|   0.342897807455451342  | 0.342897807455451398 |     7
0.400000|   0.389418342308650467  | 0.389418342308650522 |     7
0.450000|   0.434965534111230179  | 0.434965534111230234 |     7
0.500000|   0.479425538604203005  | 0.479425538604203005 |     8
0.550000|   0.522687228930659220  | 0.522687228930659220 |     8
0.600000|   0.564642473395035482  | 0.564642473395035482 |     8
0.650000|   0.605186405736039656  | 0.605186405736039545 |     8
0.700000|   0.644217687237691128  | 0.644217687237691128 |     9
0.750000|   0.681638760023334123  | 0.681638760023334123 |     9
0.800000|   0.717356090899522902  | 0.717356090899522791 |     9
0.850000|   0.751280405140292706  | 0.751280405140292706 |     9
0.900000|   0.783326909627483414  | 0.783326909627483414 |     9
0.950000|   0.813415504789373855  | 0.813415504789373744 |     9
1.000000|   0.841470984807896505  | 0.841470984807896505 |     9
------------------------------------------------------------------------------------------

Admin@LAPTOP-Q5U6S2UH:/mnt/c/Users/Admin/Desktop/Все для вуза$ ./a.out
eps = 1.110223e-16
10
1
---------------------------------------------------------------------------------------
    x     |         tayor_func        |           value           | iterations
0.000000|   0.000000000000000000  | 0.000000000000000000 |     0
0.100000|   0.099833416646828169  | 0.099833416646828155 |     1
0.200000|   0.198669330795061244  | 0.198669330795061216 |     6
0.300000|   0.295520206661339602  | 0.295520206661339602 |     6
0.400000|   0.389418342308650467  | 0.389418342308650522 |     7
0.500000|   0.479425538604203005  | 0.479425538604203005 |     7
0.600000|   0.564642473395035482  | 0.564642473395035482 |     8
0.700000|   0.644217687237691128  | 0.644217687237691128 |     8
0.800000|   0.717356090899522902  | 0.717356090899522791 |     9
0.900000|   0.783326909627483414  | 0.783326909627483414 |     9
1.000000|   0.841470984807896505  | 0.841470984807896505 |     9
-----------------------------------------------------------------------------------------
```

ВЫВОД

Благодаря выполненному заданию, я получил навыки вычисления и дальнейшего использования машинного эпсилон. На практике применил знания, полученные ранее на лекциях и семинарах по информатике. Научился выполнять организацию вывода результатов выполнения программы в виде таблицы, используя элементы псевдографики, а так же вывод вещественных чисел с заданной точностью. На основе созданной программы можно написать большой количество схожих программ, для вычисления приближенного значения корней. Также, благодаря протоколу программы, можно увидеть, что вычисления в вещественных числах в языке Си не слишком точны, и если есть возможность, стоит использовать целочисленные переменные.