Московский Авиационный Институт
(Национально Исследовательский Университет)
Факультет информационных технологий и прикладной математики
Кафедра вычислительной математики и программирования

Курсовая работа по курсу
«Вычислительные системы»
Первый семестр

Задание 3

«Вещественный тип. Приближенные вычисления. Табулирование функций»

Студент:	Сыроежкин К.Г
Группа:	М8О-104Б-18
Преподаватель:	Никулин С.П.
Оценка:	
Дата:	

Москва, 2017 г.

Содержание

1. Задание……………………………………………………………………3

2. Решение…………………………………………………………………...3

3. Описание переменных………………………………………………...5

4. Листинг программного кода…………………………………………6

5. Результат работы программы………………………………………...7

6. Вывод…..……………………………………………………………………8

Задание

Составить программу на языке 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, находящихся в рекомендованной области хорошей точности формулы Тейлора. Вычисления по формуле Тейлора проводить по экономной в сложностном смысле схеме с точностью $\varepsilon * k$, 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 8

Функция: $\dfrac{1}{2x-5}$

Отрезок:

0.0	2.0

Ряд: $-\dfrac{1}{5}-\dfrac{2x}{5^2}-\dfrac{4x^2}{5^3}-\ldots-\dfrac{2^{n-1}x^{n-1}}{5^n}$

Решение

Всё решение сводится к тому, чтобы записать на языке Си две функции и вывести их значения на заданном отрезке. Функция, реализующая вычисление с помощью ряда Тейлора, представляет собой итерационный процесс, в ходе которого последовательно вычисляется сумма членов ряда. Ряд Тейлора для функции $f(x)$ в окрестности точки a выглядит следующим образом:

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

Основной вопрос заключается в точности вычислений. Дело в том, что точность каких-либо алгоритмов в ЭВМ ограничена. Отсюда и возникает понятие «машинного эпсилон». От машинного эпсилон зависит, насколько точно можно посчитать значение функции по ряду Тейлора.

Машинное эпсилон — это максимальная относительная погрешность для конкретной процедуры округления, это такое числовое значение, меньше которого невозможно задавать относительную точность для любого алгоритма, возвращающего вещественные числа. Его можно представить как $1.0 + \varepsilon \neq 1.0$. Чем меньше его значение, тем выше точность вычисления. Практическая важность машинного эпсилон связана с тем, что два числа являются одинаковыми с точки зрения машинной арифметики, если их относительная разность по модулю меньше эпсилон. Необходимо сказать об округлении чисел: правило округления в стандарте IEEE754 говорит о том, что результат любой арифметической операции должен быть таким, как если бы он был выполнен над точными значениями и округлен до ближайшего числа, представимого в этом формате. Округление до ближайшего в стандарте сделано не так как мы привыкли. Математически показано, что если 0,5 округлять до 1 (в большую сторону), то существует набор операций, при которых ошибка округления будет возрастать до бесконечности. Поэтому в IEEE754 применяется правило округления до четного.

Способы вычисления машинного эпсилон:

1. Подключить библиотеку limits.h и использовать константу DBL_EPSILON
2. Делим 1.0 пополам пока не получится так, что мы не можем отличить одно от другого. Если так случилось, значит, разница на предыдущем шаге и есть машинное эпсилон.

   ```
   double eps = 1.0;
   while (1.0 + (eps / 2.0) > 1.0) {
   ```

eps /= 2.0;

}

3. Нужно найти число, у которого мантисса «сдвинута» на единицу левее другого числа. Эти числа 7/3 и 4/3:

   7/3 = 10.010101010101…

   4/3 = 1.0101010101010…

   Запишем числа в формате IEEE-734:

   7/3 = 1.0010101010101010101010101010101010101010101010101011 $* 2^1$

   4/3 = 1.0101010101010101010101010101010101010101010101010101 $* 2^0$

   При вычитании все биты, кроме последнего, зануляются:

   7/3 – 4/3 = 1.0000000000000000000000000000000000000000000000000001 $* 2^0$

   Получается, 7/3 – 4/3 = 1 + ε

   ε = 7/3 – 4/3 – 1

   double eps = 7.0 / 3.0 – 4.0 / 3.0 – 1.0;

Описание переменных

divider	Значение разделения длины отрезка
A,B	Постоянные интервала
interval	Шаг аргументов
eps	Машинное эпсилон
n	Число итераций
Sum_Telor	Значение ряда Тейлора
member	Значение Тейлора на данном шаге
arg1	Аргумент функции

Листинг программного кода

```
#include <stdio.h>
#include <math.h>
#include <float.h>
double func(double argument)
{
    return 1/(2*argument-5);
}
double epsil(void)
{
    double e = 1.0;
    while((1+(e/2.0))>1) e/=2.0;
    return e;
}
int main()
{
    #define A 0.0
    #define B 2.0
    int divider;
    printf("Введите делитель области: ");
    scanf("%d",&divider);
    double eps, interval = (B - A)/divider;
    int n = 0;
    eps = epsil();
    printf("Машинное эпсилон для типа double E =%.30lf \n", eps);
    printf("Таблица значений функции 1/(2*X-5)\n");
    printf("----------------------------------------------------------\n");
    printf("| X  |  Функция     | Тейлор        | Итерации    |\n");
    printf("----------------------------------------------------------\n");
    for(double arg1=A;arg1<=B;arg1+=interval)
    {
        double member,sum_Telor =0;
        for(int i=1;i<=100;i++)
        {
            member = -(pow(2, i-1)*pow(arg1, i-1))/pow(5, i);
            if (epsil()>= fabs (member)) break;
            else sum_Telor+=member;
            n++;
        }
        printf( "|%.2lf|%.15lf|%.15lf|\t%d\t|\n",arg1,func(arg1),sum_Telor,n);
        n = 0;
    }
    printf("----------------------------------------------------------\n");
    return 0;
}
```

Результат работы программы

```
Введите делитель области: 10
Машинное эпсилон для типа double E =0.000000000000000222044604925031
Таблица значений функции 1/(2*X-5)

-----------------------------------------------------------
| X  |  Функция            | Тейлор              | Итерации    |
-----------------------------------------------------------
|0.00|-0.200000000000000|-0.200000000000000|     1       |
|0.20|-0.217391304347826|-0.217391304347826|     14      |
|0.40|-0.238095238095238|-0.238095238095238|     19      |
|0.60|-0.263157894736842|-0.263157894736842|     25      |
|0.80|-0.294117647058824|-0.294117647058823|     31      |
|1.00|-0.333333333333333|-0.333333333333333|     38      |
|1.20|-0.384615384615385|-0.384615384615384|     47      |
|1.40|-0.454545454545455|-0.454545454545454|     60      |
|1.60|-0.555555555555555|-0.555555555555555|     78      |
|1.80|-0.714285714285714|-0.714285714285710|     100     |
|2.00|-1.000000000000000|-0.999999999796295|     100     |
-----------------------------------------------------------
```

```
Введите делитель области: 13
Машинное эпсилон для типа double E =0.000000000000000222044604925031
Таблица значений функции 1/(2*X-5)
-----------------------------------------------------------
| X  |  Функция            | Тейлор              | Итерации    |
-----------------------------------------------------------
|0.00|-0.200000000000000|-0.200000000000000|     1       |
|0.15|-0.213114754098361|-0.213114754098361|     13      |
|0.31|-0.228070175438596|-0.228070175438596|     17      |
|0.46|-0.245283018867925|-0.245283018867924|     21      |
|0.62|-0.265306122448980|-0.265306122448980|     25      |
|0.77|-0.288888888888889|-0.288888888888889|     30      |
|0.92|-0.317073170731707|-0.317073170731707|     35      |
|1.08|-0.351351351351351|-0.351351351351351|     41      |
|1.23|-0.393939393939394|-0.393939393939394|     49      |
|1.38|-0.448275862068966|-0.448275862068965|     59      |
|1.54|-0.520000000000000|-0.519999999999999|     71      |
|1.69|-0.619047619047619|-0.619047619047619|     89      |
|1.85|-0.764705882352941|-0.764705882352889|     100     |
|2.00|-0.999999999999999|-0.999999999796295|     100     |
-----------------------------------------------------------
```

```
Введите делитель области: 19
Машинное эпсилон для типа double E =0.000000000000000222044604925031
Таблица значений функции 1/(2*X-5)
-----------------------------------------------------------
| X  |  Функция            | Тейлор              | Итерации    |
-----------------------------------------------------------
|0.00|-0.200000000000000|-0.200000000000000|     1       |
|0.11|-0.208791208791209|-0.208791208791209|     11      |
|0.21|-0.218390804597701|-0.218390804597701|     14      |
|0.32|-0.228915662650602|-0.228915662650602|     17      |
|0.42|-0.240506329113924|-0.240506329113924|     20      |
|0.53|-0.253333333333333|-0.253333333333333|     23      |
|0.63|-0.267605633802817|-0.267605633802817|     26      |
|0.74|-0.283582089552239|-0.283582089552239|     29      |
|0.84|-0.301587301587302|-0.301587301587301|     32      |
|0.95|-0.322033898305085|-0.322033898305084|     36      |
|1.05|-0.345454545454545|-0.345454545454545|     40      |
|1.16|-0.372549019607843|-0.372549019607843|     45      |
|1.26|-0.404255319148936|-0.404255319148936|     51      |
|1.37|-0.441860465116279|-0.441860465116279|     58      |
|1.47|-0.487179487179487|-0.487179487179487|     66      |
|1.58|-0.542857142857143|-0.542857142857142|     75      |
|1.68|-0.612903225806451|-0.612903225806451|     88      |
|1.79|-0.703703703703703|-0.703703703703701|     100     |
|1.89|-0.826086956521738|-0.826086956520984|     100     |
|2.00|-0.999999999999998|-0.999999999796294|     100     |
-----------------------------------------------------------
```

Вывод

После генерации таблицы значений заданной функции можно увидеть, что значения совпадают до 10-15 знака после запятой. Из-за того, что существует понятие ограниченности разрядной сетки, вещественные числа имеют диапазон представления в памяти компьютера, что неизбежно приводит к тому, что в вычислениях в окрестности границ этого диапазона возникают погрешности.

На базе этой программы можно создать много аналогичных программ, выполняющих те же действия с другими функциями, но они не имеют прикладного применения, так как вычисление значения функции по ряду Тейлора требует много процессорного времени, что неэффективно в перспективе глобального применения.