Министерство науки и образования РФ

Федеральное государственное автономное образовательное

учреждение высшего профессионального образования

«Санкт-Петербургский государственный электротехнический

университет «ЛЭТИ» им. В. И. Ульянова (Ленина)»

(СПбГЭТУ «ЛЭТИ»)

Факультет компьютерных технологий и информатики

Кафедра вычислительной техники

Отчет по лабораторной работе №3

на тему:

“Вычисление суммы ряда с заданной точностью. Числа с плавающей точкой ”

по дисциплине “Программирование”

Вариант 5.

Выполнил: Быков И. В.

Группа 6383

Принял: Самойленко В. П.

Санкт-Петербург  
2016 г.

**Цель**

Изучение представления и особенностей использования чисел с плавающей точкой.

**Краткие теоретические сведения**

Формат чисел с плавающей точкой является приближением вещественных чисел (бесконечное множество) на ограниченном участке памяти (конечное подмножество вещественных чисел) и определяется стандартом **IEEE 754**.

Число представляется в памяти в виде последовательности бит , где количество бит для хранения порядка и мантиссы зависят от формата. Старший бит мантиссы всегда равен 1, поэтому он не хранится в памяти. Порядок представлен в виде кода со сдвигом.

*Таблица 1. Форматы IEEE 754 чисел.*

|  |  |  |  |
| --- | --- | --- | --- |
| Тип по стандарту IEEE 754 | Всего бит | Порядок, w бит | Мантисса, t бит |
| Binary16 | 16 | 5 | 10 |
| Binary32 | 32 | 8 | 23 |
| Binary64 | 64 | 11 | 52 |
| Binary128 | 128 | 15 | 112 |
| BinaryK {k >=128} | k | Round(4log2(k)) - 13 | k – w – 1 |

Типы хранения данных в С++: float, double, long double.

sizeof(float) <= sizeof(double) <= sizeof(long double)

Особенности арифметики чисел с плавающей точкой.

1. При сложении чисел с большой разницей между порядками меньшее число может не повлиять на сумму, так как бит мантиссы может не хватить для включения меньшего числа. Поэтому в качестве точности представления чисел используют машинное эпсилон Значение *eps* для типа *double* хранится в константе DBL\_EPSILON в файле *cfloat.*
2. Нельзя напрямую сравнивать 2 вещественных числа из-за погрешности в представлении. Для сравнения вещественных чисел на равенство рекомендуется сравнивать модуль разности чисел с некоторой малой величиной, например, *abs(a-b)<eps.*

**Задание**

Вычислить частичную сумму сходящегося бесконечного ряда изложения функции

**Требования к лабораторной работе.**

1. Значение аргумента *х* считывать с клавиатуры, значение eps вычислять в программе. Вычислить абсолютную и относительную погрешность вычислений.
2. Использовать рекуррентное соотношение для вычисления значения очередного слагаемого.

**Рекуррентное соотношение.**

Найдем рекуррентное соотношение для вычисления очередного элемента ряда.

Первый элемент последовательности (при х = 0 первый элемент последовательности не определен.)

**Таблица экспериментальных данных.**

eps = 1.084202e-019

|  |  |  |  |
| --- | --- | --- | --- |
| Тестовые данные , x | Ожидаемые результаты u(i) , s(i) , s, f(x) | | |
| Ааа | Введены нечисловые данные. Программа попросит ввести данные еще раз. | | |
| 0 | Т.к. при х=0 нулевой член не определен, то решения не будет. Программа попросит ввести данные еще раз. | | |
| 1 | i | U(i) | S(i) |
| 0 | 1.0000000000e+000 | 1.0000000000e+000 |
| 1 | 1.6666666667e-001 | 1.1666666667e+000 |
| 2 | 1.3888888889e-003 | 1.1680555556e+000 |
| 3 | 2.7557319224e-006 | 1.1680583113e+000 |
| 4 | 2.0876756988e-009 | 1.1680583134e+000 |
| 5 | 7.6471637318e-013 | 1.1680583134e+000 |
| 6 | 1.5619206969e-016 | 1.1680583134e+000 |
| F(x) = 1.1680583134e+000  Абс. погр-ть = 0.0000000000e+000  Отн. Погр-ть = 0.0000000000e+000 | | |

**Текст программы с комментариями**

#include <stdio.h>

#include <iostream> //setlocale()

#include <cstdlib> //atof()

#include <string.h> //strcmp()

#include <cmath> //exp(x) etc

double Find\_Eps();

void Find\_U\_S(double , double);

int main()

{

setlocale(LC\_CTYPE , "rus");

char x\_str[256];

double x = 0; //параметр х

double eps = Find\_Eps();

bool flag = false;

printf("Введите x:\n");

while (!flag)

{

scanf("%s", x\_str);

x = atof(x\_str); // преобразует в double

if ((strcmp(x\_str, "0") && (x == 0))) // делаем проверку, являются ли наши //символами числами. При этом нужно помнить, что atof преобразует и //строку, и нуль в нуль

printf("Вы ввели нечисловые данные! Повторите ввод!\n"); //Т.е. нам нужно проверить, вводился ли в аргументы нуль сначала

else if (x == 0)

printf("При х=0 нулевой член последовательности неопределен. Введите другое числовое значение.\n");

else

flag = true;

}

Find\_U\_S(x , eps);

return 0;

}

double Find\_Eps() // нахождение машинного эпсилон

{

double eps = 1.0;

while (1 + eps > 1)

eps /= 2;

eps \*= 2;

return eps;

}

void Find\_U\_S(double arg\_x , double arg\_eps) // нахождение частичной суммы //ряда

{

FILE \*file; // организуем вывод данных в файл.

file = fopen("fprintf.txt", "w");

fprintf(file,"x = %.5f\n" , arg\_x);

fprintf(file,"Машинное эпсилон: eps = %e\n" , arg\_eps);

double s = 1; // сумма ряда

double u = 1; // слагаемое

int i = 0; // шаг

double f1 = (exp(arg\_x)) / 3;

double f2 = (2 \* exp(-arg\_x \* 0.5)) /3;

double f3 = cos(sqrt(3) \* arg\_x \* 0.5);

double f = f2 \* f3 + f1; // значение f(x)

while (fabs(u) > arg\_eps)

{

fprintf(file,"Слагаемое u(%d) = %.10e \t Сумма s(%d) = %.10e\n" , i , u , i , s);

i++;

u \*= (arg\_x \* arg\_x \* arg\_x) / (3 \* i - 2) / (3 \* i - 1) / (3 \* i);

s+=u;

}

fprintf(file,"s(x) = %.10e\nf(x) = %.10e\n", s , f);

fprintf(file,"Абсолютная погрешность: %.10e\nОтносительная погрешность: %.10e\n", fabs(s - f) , fabs((s - f) / f));

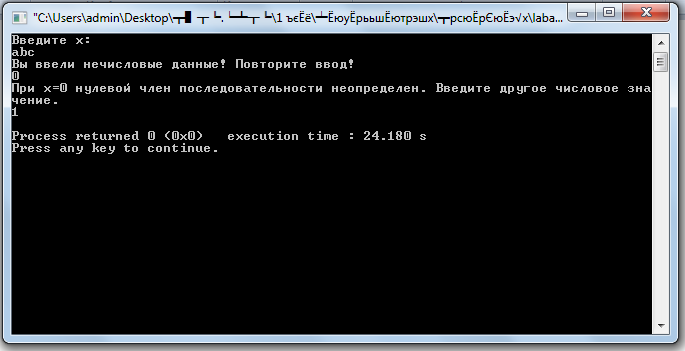
return;

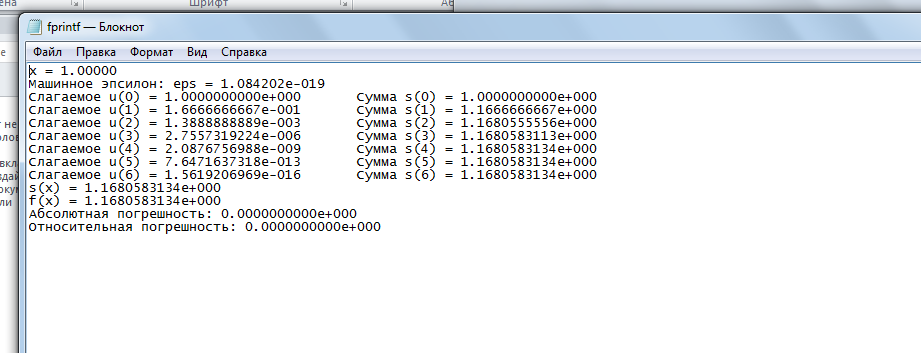
}

**Результат работы программы.**

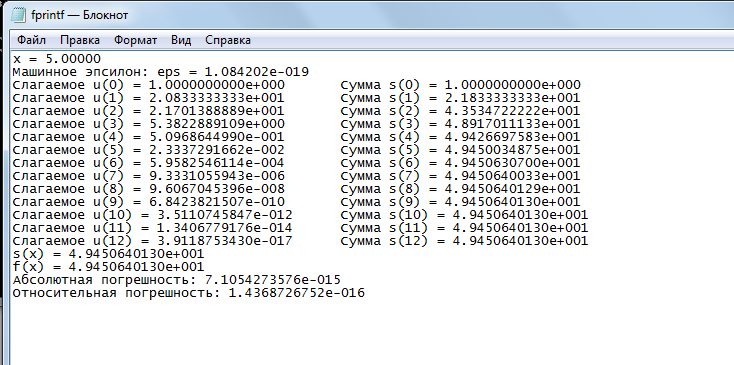
1. Введем сначала abc, затем 0, затем 1.

Результат программа выведет в файл fprintf.txt:





1. Введем х = 5. Результат работы программы:



**Вывод**

При выполнении лабораторной работы были получены навыки использования операторов циклов с неизвестным количеством повторений. Также я ознакомился с особенностями вычислений с целыми числами. Получил дополнительный опыт в получении рекуррентных соотношений.