**МИНОБРНАУКИ РОССИИ**

**САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ**

**ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**

**«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)**

**Кафедра МО ЭВМ**

**ОТЧЕТ**

**по лабораторной работе №3/4**

**по дисциплине «Программирование»**

**Тема: Вычисление суммы ряда с заданной**

**точностью. Числа с плавающей точкой и функции. Передача аргументов по ссылке и по значению**

|  |  |  |
| --- | --- | --- |
| Студент гр. 6382 |  | Вайгачев А.О. |
| Преподаватель |  | Шолохова О.М. |

Санкт-Петербург

2016

**Цель работы.**

Изучение представления и особенностей использования чисел с плавающей точкой.

Ознакомление с парадигмой процедурного программирования. Получение навыков реализации функций на языке программирования C. Изучение различных способов передачи аргументов в функцию.

**Постановка задачи.**

Разработать программу для выполнения индивидуального задания с использованием функций языка С++. Первая цифра шифра задания определяет номер одной из предыдущих лабораторных работ, а вторая – номер варианта задачи соответствующей лабораторной работы.

Задание:

Вычислить частичную сумму сходящегося бесконечного ряда разложения функции.

Рассматриваются ряды *f(x) = Σi = 0...∞ ui.* Для каждого индивидуального

задания определены вид элемента ряда *ui*, функция *f(x)*, область сходимости *D*,

если *D ≠ R*.

Вариант *1 : u(i) = (–1)ix2i/ (2i)!; f(x) = cos(x).*

**Основные теоретические положения.**

Найдем рекуррентное соотношение для вычисления очередного элемента

ряда.

Получилось линейное рекуррентное соотношение первого порядка:

, где . Первый элемент последовательности.

Вычисления суммы ряда построим по схеме итерации:

x – переменные *<i, ui, si>*, где *i* - номер итерации, *ui* = элемент ряда и *si =*

*= Σj = 0...iuj*;

x0 – начальные значения *<i, ui, si> = <0,x,x>*;

F(x) = F(*<i, ui, si>*) = *<i+1, ui×p(i), si + ui>*;

B(x) = B(*<i, ui, si>*) = *i < N*.

По условию . Однако ясно, что вычисления не могут продолжаться бесконечно. Так как ряд сходящийся, то . Так как для хранения суммы и текущего элемента ряда будем использовать тип с плавающей точкой, то при некотором *k* разность порядков текущего значения суммы sk и элемента ряда uk будет достаточно велика, чтобы при сложении *ui>k* не влияли на значение суммы. Для определения k будем использовать машинное эпсилон, определяемое по формуле *eps = min {e: (e + 1) > 1}*. Так как ряд знакочередующийся, то условием завершения итераций будет условие

B(x) = B(*<i, ui, si>*) = .

**Спецификация программы.**

*Назначение программы*. Программа предназначена для вычисления синуса угла с помощью вычисления частичной суммы сходящегося бесконечного ряда разложения функции sin(x).

*Описание программы*. (Программа написана на языке …с использованием компилятора …Входные и выходные данные)

Входными данными для программы является значение угла в радианах в области определения D, вводимое пользователем с клавиатуры. Проверка значений, вводимых пользователем присутствует: программа будет ожидать появления числа в строчке ввода. Поведение программы при некорректном вводе: Если число будет больше , чем D, то результат будет содержать большую погрешность или зациклится. Выходными данными являются:

1. Значение косинуса угла, полученное с помощью вычисления частичной сходящегося бесконечного ряда разложения функции (а также все элементы ряда и промежуточные значения суммы ряда);

2. Значение машинного эпсилон, используемого при вычислениях;

3. Значение косинуса угла, полученное с помощью стандартной библиотеки С;

4. Анализ полученных результатов (относительная и абсолютная погрешности).

*Реализация*.

Четыре функции:

1)Epsilon() - Вычисляет значение машинного эпсилон.

Возвращаемое значение: const real;

2)GiveMeX() - Ведёт диалог с поьзователем. Требует ввода значения x в радианах.

Возврашаемое значение : const real;

3)Summator\_cos(real \*x, real \*eps) – Вычисляет через рекуррентное соотношение следующее слагаемое и складывает его.

Real \*x – адрес константы x;

real \*esp – адрес вычисленного машинного эпсилон;

возвращаемое значение : real;

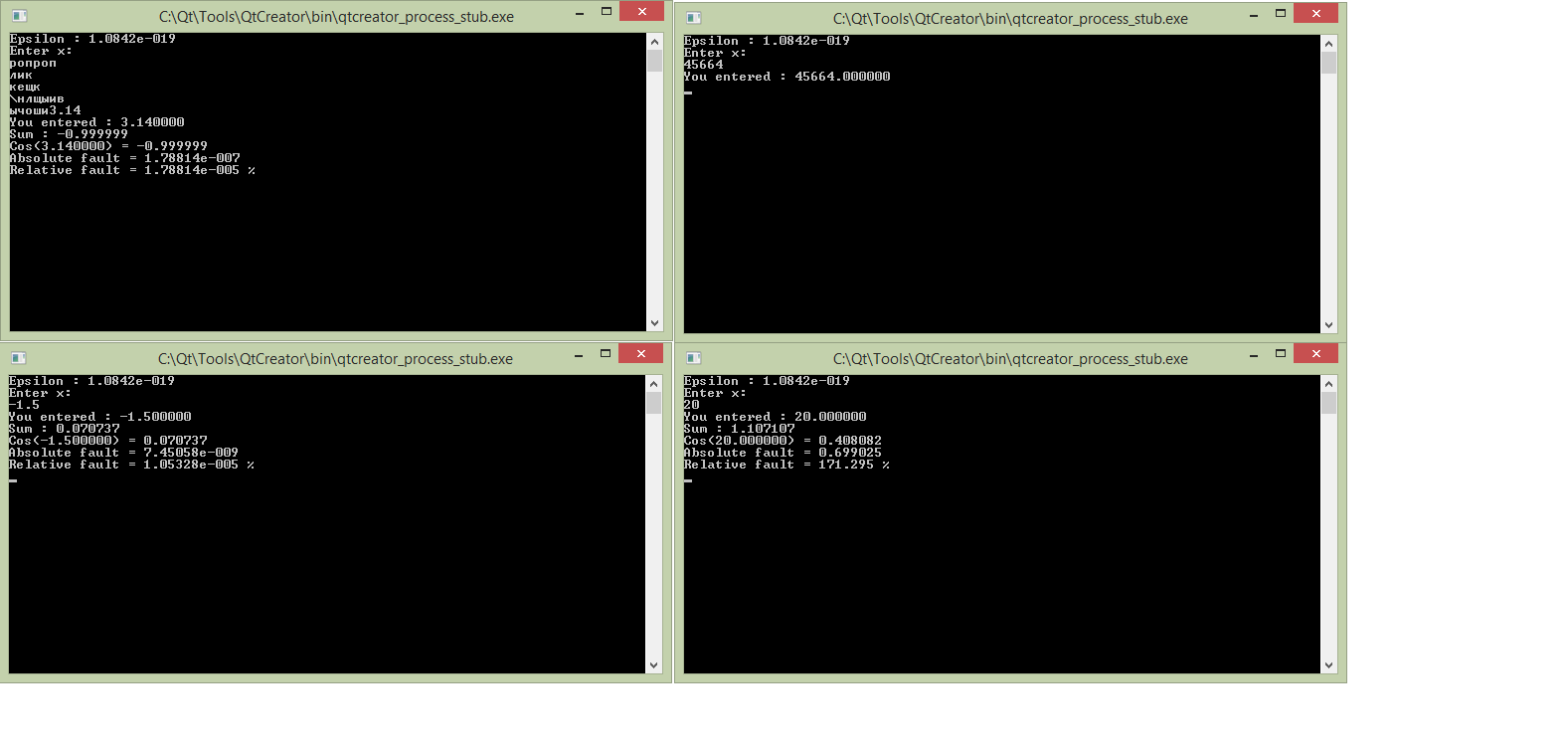
4)fault\_cos(real \*sum, real \*x) — считает и выводит абсолютную и относительную погрешность значений cos(x), полученных через стандартную библиотеку <math.h> и через разложение Тэйлора.

real \*sum — адрес, где хронится сумма сходящегося ряда;

real \*x — адрес, где хранится x;

- void;

*Пример диалога с пользователем:*



**Тестирование.**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Входные данные  x | Тип данных, используемый при вычислениях | Значение с cos(x), полученный суммированием ряда | Значение cos(x), полученный с помощью стандартной библиотеки C++ | Абсолютная погрешность | Относительная погрешность |
| 1.0 | float | 0,540302 | 0,540302 | 0 | 0% |
| 1.0 | double | 0,540302 | 0,540302 | 1,11022e-016 | 2,05482e-014 % |
| 1.0 | Long double | 0,000000 | 0,000000 | 0,0000000 | 0,0(похоже , что не работает) |
| 30,0 | double | 0,154254 | 0,154251 | 2,090574e-006 | 1,355303e-003 |
| 0 | double | 1,000 | 1,000 | 000 | 000 |

**Выводы.**

В ходе выполнения лабораторной работы были получены навыки реализации функций на языке программирования C и изучены различные способы передачи аргументов в функцию. И узнал, про сходящиеся ряды разных функций.

**Приложение А. Исходный код.**

/\*

\*Лабораторная работа №3-4

\*lab3-4.cpp

\*Вайгачев Андрей Олегович

\* 12.10.2016

\*

\* программа для вычисления косинуса

\*

\*/

#include <stdio.h>

#include <math.h>

#define real double

const real Epsilon () {

real eps = 1.0;

while (1 + eps > 1.0){

eps \*= 0.5;

}

eps \*= 2;

printf("Epsilon : %Le\n",eps); // %f - в случае если real - float, %lf - в случае если не так

return eps;

}

const real GiveMeX(){

real x = 0;

printf("Enter x: \n");

while (!scanf("%Lf", &x)) // <---

{

getchar();

}

printf("You entered : %Lf\n",x); // <---

return x;

}

real Summator\_cos(real \*x,real \*eps){

//const real eps = Epsilon();

real s = 0;

real u = 1;

int i = 0;

while(fabs(u) > \*eps){

s += u;

u \*= -(\*x)\*(\*x)/(2\*i+1)/(2\*i+2);

i++;

}

printf("Sum : %Lf\n",s); // <---

return s;

}

void fault\_cos(real \*sum,real \*x){

real \_cos = cos(\*x);

printf("Cos(%f) = %Lf\n",\*x,\_cos); // <---

printf("Absolute fault = %le\n",fabs(\_cos-\*sum)); // <---

printf("Relative fault = %le %%\n",fabs((fabs(\_cos-\*sum)/\_cos)\*100)); // <---

}

int main(){

const real eps = Epsilon();

const real x = GiveMeX();

const real sum = Summator\_cos(&x,&eps);

fault\_cos(&sum,&x);

return 0;

}

