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 №4

на тему:

“Функции. Передача аргументов по ссылке и по значению ”

по дисциплине “Программирование”

Вариант 5.

Выполнил: Быков И. В.

Группа 6383

Принял: Самойленко В. П.

Санкт-Петербург  
2016 г.

**Цель**

Ознакомление с парадигмой процедурного программирования. Получение навыков реализации функций на языке программирования С++. Изучение различных способов передачи аргумента в функцию.

**Задание**

Вычислить частичную сумму сходящегося бесконечного ряда изложения функции

**Требования к лабораторной работе.**

1) Оформить в виде функций фрагменты программы, обеспечивающие ввод и проверку исходной информации, ее преобразование, вывод результатов

(промежуточных и окончательных).

2) При реализации функций обязательно обеспечить передачу данных с помощью параметров;3) Для всех определяемых функций указать спецификацию

**Теоретические сведения.**

**Объявление и определение функции.**

Объявление функции (описание прототипа функции) - описание имени функции, списка формальных параметров и типа возвращаемого значения:

<Тип возвращаемого значения> <Имя функции> (список формальных параметров)

Определение функции имеет следующую структуру:

<Тип возвращаемого значения> <Имя функции> (список формальных параметров)

{

операторы, среди которых есть хотя бы один оператор return,

аргументом которого является результат выполнения функции

}

**Переменные**.Переменные, объявленные внутри функции, называются локальными, остальные – глобальными.

Локальные переменные (без модификатора static) имеют автоматическую продолжительность хранения.

Глобальные переменные имеют статическую продолжительность хранения.

**Передача параметров в функцию.**

Параметры, которые указываются при объявлении/определении функции, называются формальными, а параметры, которые передаются непосредственно при вызове функции – фактическими. Параметры функции имеют локальную область видимости.

Рассмотрим три способа передачи аргумента в функцию:

1. Объявление функции с параметрами, передаваемыми по значению, имеет вид:

int func(int i, float f, char c);

1. Заголовок процедуры с параметрами, передаваемыми по указателю, имеет вид:

int func(int\* i, const float\* f, char\* c);

1. Заголовок процедуры с параметрами, передаваемыми по ссылке, может, например, иметь вид:

int func(int& i, const float& f, char& c);

**Рекуррентное соотношение.**

Найдем рекуррентное соотношение для вычисления очередного элемента ряда.

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

**Спецификации функций**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Название | Описание | Предусловие | Постусловие | Что возвращает? |
| double Find\_Eps(); | Нахождение епсилон | Нет | Нет | eps |
| double Find\_U\_S(double x , double eps , double& u , int& i ); | Нахождение частичной суммы S и вывод промежуточного результата | x<>0, eps = abs(MIN) | u – последнее слагаемое, i – порядковый номер последнего слагаемого | S – считает итоговую сумму |
| double Input\_Inspection\_X(const char\* x\_str); | Ввод и проверка на правильность ввода | x\_str - константный указатель на введенный массив символов | нет | x |
| double Find\_F(double x); | Нахождение F | x <> 0 | нет | F |
| void Output\_Text(int i , double u , double x , double s ); | Вывод окончательного результата в файл | i – вычисленный номер последнего слагаемого , u – последнее слагаемое,  s – получившаяся сумма ,  f – посчитанное по формуле выражение | нет | Ничего. |
| void Output\_Screen(int i, double u , double x , double s ); | Вывод окончательного результата на экран | i, u, s, f (аналогичны предыдущей строке) | нет | Ничего. |

**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

// Функция нахождения епсилон

// нет предусловия и постусловия

// возвращает eps =abs(MIN) - значение епсилон

double Find\_Eps();

// функция нахождения S и U

// функция вычисляет и выводит промежуточное решение в текстовый файл. U(i)\*=(x \* x \* x) / ((3 \* i - 2) \* (3 \* i - 1) \* (3 \* i)); S(i) +=u(i).

// предусловие: x <> 0 , eps = abs(MIN).

// постусловие: i - номер последнего слагаемого, u - значение последнего слагаемого

// возвращает значение s

double Find\_U\_S(double x , double eps , double& u , int& i );

// Функция ввода и проверки на корректность

// Организует ввод строки x\_str, переводит ее в double x и проверяетна корректность

// Предусловие: x\_str

// Постусловий нет

// Возвращает значение x - double

double Input\_Inspection\_X(const char\* x\_str);

// Функция нахождения F по формуле

// вычисляет f = ((exp(x)) / 3) + ((2 \* exp(-x \* 0.5)) /3) \* cos(sqrt(3) \* x \* 0.5);

// предусловие: х <> 0

// нет постусловия

// возвращает f;

double Find\_F(double x);

// Функция выводит в текст итоговый результат вычислений

// предусловие: i - номер последнего слагаемого, u - значение последнего слагаемого , x <> 0 , s - итоговая сумма

// без постусловия

// ничего не возвращает

void Output\_Text(int i , double u , double x , double s );

// Функция вывода окончательного результата на экран. Все аналогично предыдущей функции

void Output\_Screen(int i , double u , double x , double s );

int main()

{

setlocale(LC\_CTYPE , "rus"); //функция перекодирует программу на требуемый язык

char x\_str[256];

bool flag;

double u;

int i;

double x = Input\_Inspection\_X(x\_str); //параметр х

double eps = Find\_Eps();

double s = Find\_U\_S(x , eps , u , i ); //результат s

Output\_Text(i , u , x , s); //вывод в файл

Output\_Screen(i , u , x , s); // вывод на экран

return 0;

}

double Input\_Inspection\_X(const char\* x\_str)

{

double x\_arg = 0;

bool flag = false;

printf("Введите x:\n");

while (!flag)

{

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

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

if ((strcmp(x\_str, "0") && (x\_arg == 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\_arg == 0)

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

else

flag = true;

}

return x\_arg;

}

double Find\_Eps() // нахождение машинного эпсилон

{

double eps = 1.0;

while (1 + eps > 1)

eps /= 2;

eps \*= 2;

return eps;

}

double Find\_F(double x) // находим f

{

double f1 = (exp(x)) / 3;

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

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

double f = f2 \* f3 + f1;

return f;

}

double Find\_U\_S(double x , double eps , double& u , int& i ) // нахождение частичной суммы ряда

{

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

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

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

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

double arg\_s = 1; // сумма ряда

u = 1; // слагаемое

i = 0; // шаг

while (fabs(u) > eps)

{

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

i++;

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

arg\_s+=u;

}

return arg\_s;

}

void Output\_Text(int i , double u , double x , double s )

{

FILE \*file;

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

fprintf( file , "\nпри i = %d и u = %.10e\ns(x) = %.16e\nf(x) = %.16e\n", i , u , s, Find\_F(x));

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

return;

}

void Output\_Screen(int i , double u , double x , double s )

{

printf( "\nпри i = %d и u = %.10e\ns(x) = %.16e\nf(x) = %.16e\n", i , u , s, Find\_F(x));

printf( "Абсолютная погрешность: %.10e\nОтносительная погрешность: %.10e\n", (fabs(s - Find\_F(x))) , (fabs((s - Find\_F(x)) / Find\_F(x))));

return;

}

**Результат работы программы.**

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

Промежуточный и окончательный результат программа выведет в файл fprintf.txt, так же окончательный результат программа выведет на экран:

Введите x:

abc

Вы ввели нечисловые данные! Повторите ввод!

0

При х=0 нулевой член последовательности не определен. Введите другое числовое значение.

1

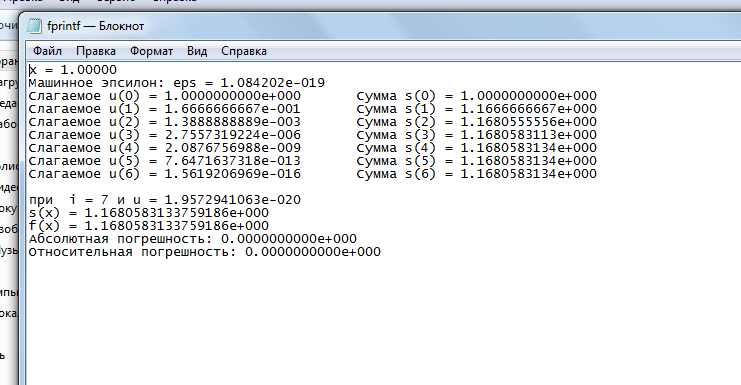
при i = 7 и u = 1.9572941063e-020

s(x) = 1.1680583133759186e+000

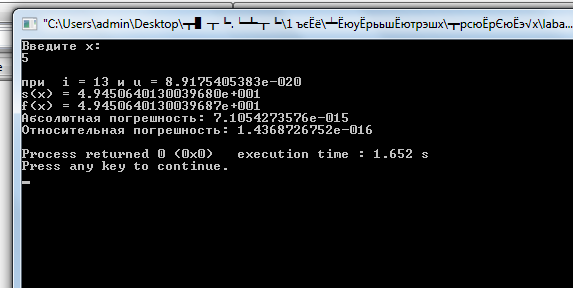
f(x) = 1.1680583133759186e+000

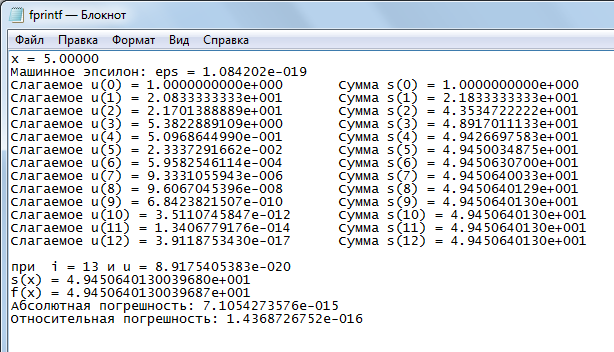
Абсолютная погрешность: 0.0000000000e+000

Относительная погрешность: 0.0000000000e+000



1. Введем х = 5. Результат работы программы:





**Вывод**

Я ознакомился с парадигмой процедурного программирования. Получил навыки реализации функций на языке программирования С++. Изученил различные способы передачи аргумента в функцию.