МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

* + 1. федеральное государственное автономное образовательное учреждение высшего образования

«САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ АЭРОКОСМИЧЕСКОГО ПРИБОРОСТРОЕНИЯ»

(ГУАП)

КАФЕДРА ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМ И СЕТЕЙ

Преподаватель

канд. техн. наук, доцент Л.Н. Бариков

Отчёт

по лабораторной работе №5

по дисциплине ОСНОВЫ ПРОГРАММИРОВАНИЯ

на тему: «Суммирование рядов»

Работу выполнила

студентка гр. 4141 А. В. Мазанова

Санкт-Петербург

2022

**Вариант 23**

**Цель лабораторной работы:** изучение концепций и освоение технологии процедурного программирования, приобретение навыков программирования на языке C/C++ циклических вычислений при решении задач суммирования рядов.

**Задание на программирование:** используя технологию процедурного программирования разработать программу вычисления суммы ряда с заданной точностью в заданном интервале допустимых значений аргумента.

**Ход выполнение работы:**

1. **Получила индивидуальное задание:**

****

1. **Разработала математическую модель:**

Поскольку нумерация слагаемых по заданию начинается с n = 1, вычисление необходимой суммы осуществляется по формуле:

*sum = x + sl1 + sl2 + sl3 + … +* *sln*, где *sln –* значение слагаемого с номером n (n = 1, 2, 3 …).

Процесс накопления суммы будет осуществляться следующим образом:

За исходное значение суммы принимаем значение *x* (*sum = x*)

Для определения величины слагаемого с номером *n = 1*, подставляем это значение *n* в формулу и получаем *sl1 =*

Сравниваем абсолютную величину этого слагаемого с заданной точностью вычисления. Если , то добавляем это значение к текущему значению суммы, увеличиваем номер *n* очередного слагаемого на 1 и определяем его величину

В формуле для вычисления величины очередного слагаемого присутствуют не реализованные в языке программирования операции возведения в степень и вычисления значения факториала, поэтому для получения величин, слагаемых с последующими номерами, выведем рекуррентную формулу. Найдем величину отношения величины слагаемого с текущим номером к величине слагаемого с предыдущим номером:

Отсюда следует, что величина очередного слагаемого определяется по формуле, в которой присутствуют только реализованные в языке программирования операции (*вычисление x2 заменяем на x×x):*

Итак, для накопления суммы увеличиваем номер *n* очередного слагаемого, вычисляем его величину, умножая на *k* величину предыдущего слагаемого, сравниваем его абсолютную величину с заданной точностью вычисления и, если точность не достигнута, опять увеличиваем номер очередного слагаемого.

Эти действия повторяем до достижения заданной точности вычисления.

Все перечисленные действия оформляем в виде отдельной функции, аргументами которой будут значения *x* и точности *ɛ*. Функция будет возвращать вычисленное значение суммы. Число слагаемых, попавших в сумму, будет передаваться в функцию *main*() через указатель.

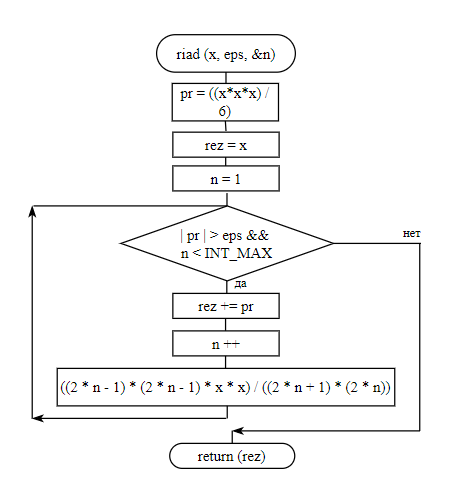
По заданию необходимо получить **таблицу значений** суммы ряда в зависимости от значения аргумента *x*, вычисленных с заданной точностью *ɛ*. Поэтому в качестве исходных данных, помимо значения точности вычисления результата, вводятся начальное и конечное значения аргумента (*xn, xk*), а также шаг изменения значения аргумента от начального значения до конечного (*h*). По условию решаемой задачи допустимые значения аргумента заданы неравенством |x| < 1, а значение точности вычисления *ɛ* должно принадлежать интервалу (0, 0.1]. Поэтому при вводе исходных данных проверяем указанные соотношения.

После этого за текущее значение аргумента *x* принимаем значение *xn*, вычисляем для этого значения величину суммы ряда и выводим одну строку искомой таблицы (текущее значение аргумента *x*, значение вычисленной суммы, количество слагаемых *n*, попавших в сумму и контрольное значение, полученное с использованием стандартных функций). Контрольное значение в рассматриваемом примере – это величина *arcsin(x)*. Вычисленное значение суммы ряда для текущего значения аргумента и контрольное значение не должны различаться более чем на величину точности вычисления.

Затем увеличиваем значение аргумента *x* на величину шага изменения *h*, проверяем не превышение аргументом *x* величины *xk* и вновь вычисляем для этого значения величину суммы ряда и выводим очередную строку искомой таблицы.

Эти действия повторяются до момента, когда текущее значение *x* превысит значение *xk*. Построение таблицы закончено. Задача решена.

1. **Построила схему алгоритма вычисления значения суммы ряда (*функция raid)***

****

1. **Составила программу на языке *C/C++***

#include<iostream>

#include<math.h>

#include<limits.h>

#include<locale.h>

using namespace std;

double riad(double, double, int& n);

//main\_beg

int main()

{

int vid,

n; //число слагаемых, попавших в сумму

double x, xn, xk, //текущее, начальное и конечное значения аргумента

h; //шаг изменения значения аргумента

double eps, //точность вычисления суммы

rez; //результат вычисления суммы ряда

setlocale(LC\_ALL, "Russian");

for (;;)

{//Ввод исходных данных

cout << "\n Вид действия:" << endl;

cout << " 1 - получение таблицы значений" << endl;

cout << " 2 - завершение работы программы" << endl;

cout << " Выберите вид действия -> ";

cin >> vid;

if (vid == 1)

{

cout << " Введите начальное значение аргумента ";

cin >> xn;

if (fabs(xn) >= 1)

{

cout << " Ошибка! Значение д.б. 0<xn<1";

continue;

}

cout << " Введите конечное значение аргумента ";

cin >> xk;

if ((fabs(xk) >= 1) || (xk < xn))

{

cout << " Ошибка! Значение д.б. 0<xk<1 и xk>=xn";

continue;

}

cout << " Введите шаг изменения аргумента h ";

cin >> h;

if (h <= 0)

{

cout << " Ошибка! Значение h д.б. >0";

continue;

}

cout << " Введите точность вычисления eps ";

cin >> eps;

if ((eps <= 0) || (eps > 0.1))

{

cout << " Ошибка! Значение eps д.б. >0 и <=0.1";

continue;

}

//Вывод заголовка

cout << "\n Аргумент| Сумма |Кол.слаг.|Контрольное значение" << endl;

x = xn;

do {

n = 1;

rez = riad(x, eps, n);

cout.width(7); cout.precision(2); cout << x;

if (n < INT\_MAX)

{

cout.width(11); cout.precision(4); cout << rez;

cout.width(6); cout << n;

cout.width(15); cout.precision(4); cout << asin(x) << endl;

}

else cout << " Точность не достигнута" << endl;

x += h;

} while (x <= xk + eps);

}

else break;

}

return 1;

}

//main\_end

//riad\_beg

//Вычисление суммы

double riad(double x, double eps, int& n)

{

double rez = x; //результат вычисления

double pr = ((x \* x \* x) / 6); //величина текущего слагаемого

n = 1; //номер текущего слагаемого

while (fabs(pr) > eps && n < INT\_MAX)

{

rez += pr;

n++;

pr \*= ((2 \* n - 1) \* (2 \* n - 1) \* x \* x) / ((2 \* n + 1) \* (2 \* n));

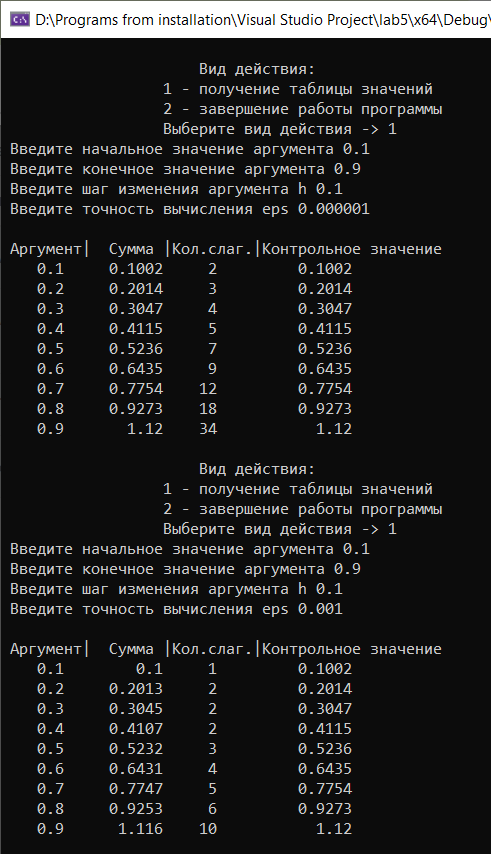
}

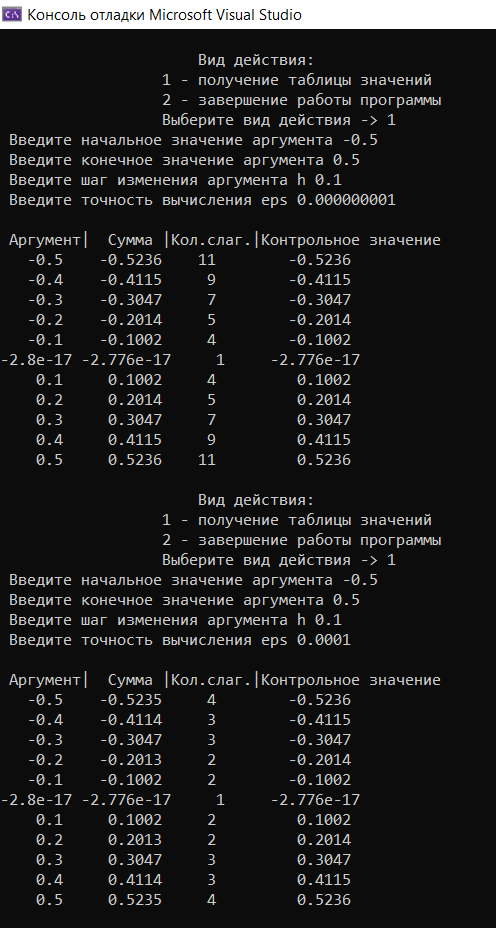
pr += x;

return rez;

}

//riad\_end

1. **Продемонстрировать результат выполнения программы**
2. ****

****

1. **Вывод:** я изучила концепции и освоила технологии процедурного программирования, приобрела навыки программирования на языке C/C++ циклических вычислений при решении задач суммирования рядов.