МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФГБОУ ВО АЛТАЙСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Институт цифровых технологий, электроники и физики

Кафедра вычислительной техники и электроники (ВТиЭ)

Лабораторная работа № 4

**Решение задач численными методами с использованием циклов**

Выполнил студент 595 гр.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Д.В. Осипенко

Проверил: к.т.н,, доцент каф. ВТиЭ

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Ю.Г. Скурыдин

Лабораторная работа защищена

«\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2020 г.

Оценка \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Барнаул 2020

1. **Формулировка задачи**

Ввести с клавиатуры значение аргумента x и значение точности вычислений ε. Вычислить с заданной степенью точности сумму элементов бесконечного сходящегося числового ряда (стандартную функцию вычисления факториала не использовать).

, где |x| < 1

1. **Постановка задачи**

На вход программы поступает 2 вещественных чисел в диапазоне – 1 до 1 и +/- 1.7e308, необходимо вычислить сумму членов сходящегося ряда, которые соответствуют заданной точности. На выход программы поступает одно вещественное значение суммы сходящегося ряда.

1. **Математическая модель**

Нужно получить сумму членов ряда с заданной точностью, для этого вычисляем и суммируем каждый элемент, модуль значения которого больше eps, после этого выводим сумму. Например: eps = 0.0001, x = 0.6;k(номер члена ряда)=1,fct(k!)=1\*2 = 2, s\_i(член ряда) = |0.18| > eps, s = 0.18; k=2, fct=2\*3\*4=24, s\_i= |-0.0054| > eps, s =0.1746; k=3, fct=24\*5\*6=720, s\_i = |6.48e-05| < eps, s = 0.1746.

1. **Описание алгоритма**

Начало

* 1. Присвоить s = 0.0, s\_i = 0.0, i = 1, fct = 1, k = 1
  2. Вводим число x
  3. Вводим число eps
  4. Проверка условия: k == 1 or abs(s\_i) > eps

Если условие не выполняется, то переход к пункту 4.12

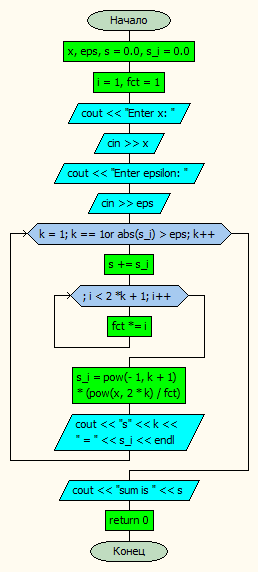
* 1. Присвоить s результат выражения s + s\_i
  2. Проверка условия: i < 2\*k +1

Если условие не выполняется, то переход к пункту 4.9

* 1. Присвоить fct результат выражения fct\*i
  2. Увеличить i на 1, перейти к пункту 4.6
  3. Присвоить s\_i результат выражения pow(-1, k + 1) \* (pow(x, 2 \* k) / fct)
  4. Вывести s\_i
  5. Увеличить k на 1, перейти к пункту 4.4
  6. Вывести s

Конец

1. **Опорный граф (блок-схема) алгоритма**

****

1. **Проект программы с определением замкнутых программных единиц и необходимых структур данных**

Программа состоит из блока двух вложенных циклов, четырех блоков вывода и двух блоков ввода . Структура данных – нет.

1. **Текст программы:**

**на языке С++**

#include <iostream>

#include <cmath>

using namespace std;

int main()

{

double x, eps, s = 0.0, s\_i = 0.0;

int i = 1, fct = 1;

cout << "Enter x: "; cin >> x;

cout << "Enter epsilon: "; cin >> eps;

for (int k = 1; k == 1 or abs(s\_i) > eps; k++) {

s += s\_i;

for (; i < 2 \* k + 1; i++) {

fct \*= i;

}

s\_i = pow(-1, k + 1) \* (pow(x, 2 \* k) / fct);

cout << "s"<< k << " = " << s\_i << endl;

}

cout << "sum is " << s;

return 0;

}

**на языке Pascal**

uses math;

var

x,eps,s,s\_i:real;

i,k,fct: integer;

begin

s:= 0.0; s\_i:=0.0; i:=1; fct:=1; k:=1;

writeln('Enter x'); readln(x);

writeln('Enter eps'); readln(eps);

while((k = 1) or (abs(s\_i)>eps)) do

begin

s := s + s\_i;

while(i < 2\*k+1) do

begin

fct := fct \* i;

i := i+1;

end;

s\_i := Power(-1, k + 1) \* (Power(x, 2 \* k) / fct);

writeln('s',k,' = ',s\_i);

k := k + 1;

end;

writeln('sum is ',s);

end.

1. **Проверка работоспособности (тестирование) программы**

Введем :0.6 и 0.0001, результат 0.1746.

1. **Сравнительный анализ и оценка эффективности работы программ на разных языках программирования**

С++: 0.64 сек, pascal: 1.3 сек. Данная программа на C++ выполняется в среднем 2 раз быстрее чем на pascal, также в C++ используется цикл for, в pascal цикл while.

1. **Формулировка задачи**

Методом деления отрезка пополам и методом итераций найти приближенное значение корня уравнения  на интервале [0.5, 1]. Абсолютная погрешность не превышает 0.0015. Сравнить методы вычисления по количеству итераций

1. **Постановка задачи**

На вход программы ничего не поступает, необходимо сравнить методы поиска приближенных корней уравнения посчитав кол-во итераций каждой. На выход программы поступает два вещественных значения x и два целочисленных значения count.

1. **Математическая модель**

Нужно посчитать кол-во итераций двух методов, ищущих корни уравнения:

1) Суть метода деления отрезка пополам состоит в разбиении отрезка **[a,b]** (при условии **f(a)f(b)<0**) на два отрезка, определении знака функции **f(x)** в середине отрезка **(a+b)/2** и выборе отрезка, на котором функция меняет знак и содержит решение. Деление отрезка продолжается до достижения необходимой точности решения **ε**. Сначала находим отрезок **[a,b]** такой, что функция **f(x)** непрерывна и меняет знак на отрезке, то есть **f(a)·f(b)<0**.

2) Суть метода итераций состоит в расчётах новой точки **x** (итерациях) по формуле **x = φ(x)**, которая выводится из уравнения **f(x)=0**. Итерации продолжаются до достижения необходимой точности решения **ε**. Метод итераций применим, если уравнение вида **f(x) = 0** сводится к уравнению вида **x = φ(x)** такому, что функция **φ(x)** непрерывна и дифференцируема на отрезке **[ a ; b ]**

Пример:

1) a = 0.5, b = 1.0, eps = 0.0015, 1 итерация – x = 0.75, f(a)f(x) < 0 => b = x = 0.75, 2 итерация – x = 0.625, f(x)f(b) < 0 => a = x, … в итоге получаем x = 0.7217, кол-во итераций(count) – 9.

2) a = 0.5, b = 1.0, eps = 0.0015, x0 = 0.75, 1 итерация – x = **φ(x0)=** 0.705, x0=x = 0.705, 2 итерация – x = **φ(x0)=** 0.732, x0 = x = 0.732 … в итоге получаем x = 0.722, кол-во итераций(count) – 8

Из примеров можно сделать вывод, что у первого метода точность выше, но затрачиваемое кол-во итераций больше, чем у второго.

1. **Описание алгоритма**

Начало

* 1. Присвоить a = 0.5, b = 1.0, eps = 0.0015, count = 1
  2. Проверка условия: |b-a| > 2\*eps:

Если условие не выполняется, то переход к пункту 4.12

* 1. Присвоить x = (a+b)/2
  2. Проверка условия f(a)\*f(x) < 0:

Если условие не выполняется, то переход к пункту 4.9

* 1. Присвоить b = x, перейти к пункту 4.8
  2. Проверка условия f(x)\*f(b)< 0:

Если условие не выполняется, то переход к пункту 4.8

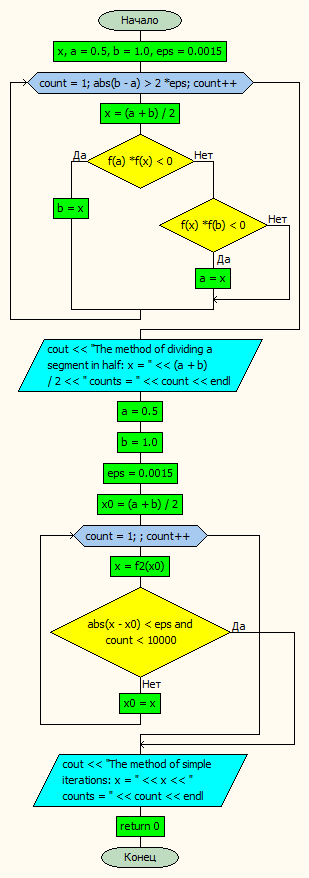
* 1. Присвоить a = x,
  2. Увеличить count на 1, перейти к пункту 4.2
  3. Вывести (a+b)/2, count
  4. Присвоить a = 0.5; b = 1.0; eps = 0.0015; count = 1; x0 = (a + b) / 2
  5. Присвоить x = f2(x0)
  6. Проверка условия abs(x - x0) < eps:

Если условие не выполняется, то присвоить x0 = x, перейти к пункту 4.11

* 1. Вывести x, count

Конец

1. **Опорный граф (блок-схема) алгоритма**

****

1. **Проект программы с определением замкнутых программных единиц и необходимых структур данных**

Программа состоит из двух блоков циклов, содержащих 3 блока условий, двух блоков вывода . Структура данных – нет.

1. **Текст программы:**

**на языке С++**

#include <iostream>

#include <cmath>

#define f(x) (5\*x+8\*log(x)-1)

#define f2(x) (x-0.1\*(5\*x+8\*log(x)-1))

using namespace std;

int main() {

double x, a = 0.5, b = 1.0, eps = 0.0015;

int count;

for (count = 1; abs(b - a) > 2 \* eps; count++) {

x = (a + b) / 2;

if (f(a) \* f(x) < 0)

b = x;

else if (f(x) \* f(b) < 0)

a = x;

}

cout << "The method of dividing a segment in half: x = " << (a + b) / 2 << " counts = " << count << endl;

a = 0.5; b = 1.0; eps = 0.0015;

double x0 = (a + b) / 2;

for (count = 1;; count++) {

x = f2(x0);

if (abs(x - x0) < eps and count < 10000) break;

x0 = x;

}

cout << "The method of simple iterations: x = " << x << " counts = " << count << endl;

return 0;

}

**на языке Pascal**

uses math;

var

x,eps,x0,a,b:real;

count:integer;

begin

a := 0.5; b := 1.0; eps := 0.0015;count := 1;

while((abs(b - a) > 2 \* eps) and (count < 10)) do

begin

x := (a + b) / 2;

if (((5\*a+8\*ln(a)-1)) \* ((5\*x+8\*ln(x)-1))) < 0 then

begin

b := x

end

else if (((5\*b+8\*ln(b)-1)) \* ((5\*x+8\*ln(x)-1))) < 0 then

begin

a := x;

end;

count := count + 1;

end;

writeln('The method of dividing a segment in half: x = ', (a + b) / 2 ,' counts = ' , count);

a := 0.5; b := 1.0; eps := 0.0015;

x0 := (a + b) / 2;

for count:=1 to 10000 do

begin

x := (x0-0.1\*(5\*x0+8\*ln(x0)-1));

if (abs(x - x0) < eps) then break;

x0 := x;

end;

writeln('The method of simple iterations: x = ', x ,' counts = ' , count);

end.

1. **Проверка работоспособности (тестирование) программы**

Результат x =  7.21679 counts = 9, x =  7.2226 counts = 8.

1. **Сравнительный анализ и оценка эффективности работы программ на разных языках программирования**

На удивление программы выполняются с равной скорость, но pascal использует в два раза меньше памяти, чем С++. В место первого цикла for в С++, в паскаль используется цикл while. Также ввиду отсутствия нормальных макросов в pascal, код уравнения был вставлен в ручную в выражения.