МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФГБОУ ВО АЛТАЙСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Физико-технический факультет

Кафедра информатики и вычислительной техники (ИВТ)

Лабораторная работа №04

**Решение задач численными методами с использованием циклов**

Выполнил студент 585 гр.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ В.М. Губченко

Проверил: к.т.н,, доцент каф. ВТиЭ

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

Лабораторная работа защищена

«\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2019 г.

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

Барнаул 2019

1. **Формулировка задачи**

Ввести с клавиатуры x и точность вычисления Eps. Вычислить с заданной точностью сумму ряда

1. **Постановка задачи**

На вход поступает вещественное число, определяющее переменную x и вещественное положительное число Eps, которое на много меньше единицы, определяющую точность вычисления. Программа вычисляет сумму ряда. На выход поступает положительное действительное число, равное результату вычислений.

1. **Математическая модель**

Для построения математической модели рассмотрим пример. Инициализируется переменная sum с значением 0 и переменная k с значением 0. Программа получает на вход число x=2, Eps=0.0001. При k=1 значение элемента ряда будет равно -0.666667, сумма ряда равна -0.666667, при k=2 значение элемента ряда равно 0.166667 сумма ряда равна -0.5, при k=6 численное значение элемента ряда будет равно 9.92064e-005, а сумма ряда равна 0.528472. На данном шаге модуль значения элемента будет меньше численного значения переменной Eps и программа выведет число 0.528472 равное сумме ряда.

1. **Описание алгоритма**

**Начало алгоритма**

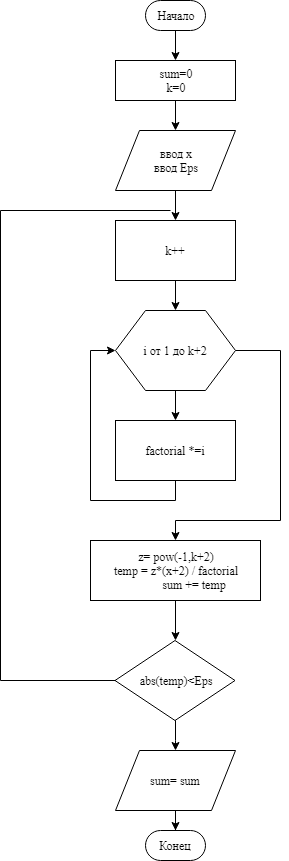
* 1. Присвоение переменной sum значения 0
  2. Присвоение переменной k значения 0
  3. Присвоение переменной factorial значения 1
  4. Ввод числа х
  5. Ввод числа Eps
  6. Присвоение переменной k значения k+1
  7. Присвоение переменной i значения 1
  8. Если условие i<= k+2 не выполняется, то переход к 4.11
  9. Присвоение переменной i значения i+1
  10. Присвоение переменной factorial значения factorial\*i переход к пункту 4.8
  11. Присвоение переменной z значения -1k+2
  12. Присвоение переменной temp значения z\*(x+2) / factorial
  13. Присвоение переменной sum значения sum+temp
  14. Если |temp|>Eps переход к пункту 4.6
  15. Вывод значения переменной sum

**Конец алгоритма**

1. **Проект программы с нахождением суммы числового ряда с заданной точностью**

Проект программы с определением замкнутых программных единиц и необходимых структур данных. В данном случае программа должна состоять из десяти частей: 2 блока ввода, цикла с параметром (содержит блок определения переменной i, 1 блок проверки условия, два блока арифметических действий ), 3 блока арифметических вычислений и 1 блок вывода и цикл с постусловием. В качестве структуры данных могут быть введены переменные для хранения исходного числа, переменной x, результата.

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



1. **Текст программы:**

**на языке С++**

#include <iostream>

#include <cstdlib>

#include <math.h>

using namespace std;

int main (int argc, char const \*argv[]) {

float sum=0, Eps, x, temp, factorial=1;

int k=0, z;

cout << "x=";

cin >> x;

cout << "Eps=";

cin >> Eps;

do

{

k++;

for (int i=1; i <= k+2; i++) {

factorial \*=i;

};

z=pow(-1,k+2);

temp = z\*(x+2) / factorial;

sum += temp;

factorial = 1;

}

while (abs(temp)>Eps);

cout << "sum=" << sum << '\n';

return 0;

}

**на языке Pascal**

Program lab\_04;

var sum, Eps, x, temp, factorial: real;

var z, i, k, t: integer;

Begin

sum:=0;

k:=0;

Write('x=');

ReadLn(x);

Write('Eps=');

ReadLn(Eps);

REPEAT

BEGIN

k:=k+1;

for i := 1 to k+2 do begin

factorial:= factorial \* i;

end;

t:= (k+2) mod 2;

if (t=0) then

begin

z:= 1;

end

else

begin

z:= -1;

end;

temp:= z \* (x+2) / factorial;

sum:= sum + temp;

END;

UNTIL (abs(temp)<Eps);

Write('sum=', sum);

end.

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

Для проверки работоспособности программы, переменной x присвоим значение 2, а переменной Eps значение 0.01. Программа выведет значение переменной sum равное -0.666667.

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

Ввиду простоты программы она одинаково эффективно выполняется для обоих языков программирования.

1. **Формулировка задачи**

По формуле Симпсона и методом прямоугольников вычислить приближенное значение интеграла . Точность не превышает 0.001. Сравнить методы вычисления по количеству итераций.

1. **Постановка задачи**

На вход поступает положительное вещественное число eps, определяющее точность вычисления, которое не превышает 0.001. После этого программа вычисляет интеграл по формуле Симпсона и методом прямоугольников. На выход поступают 2 действительных числа определяющих значения вычисленного интеграла разными способами и ещё два целых, положительных числа определяющих количество итераций в каждом методе вычислений.

1. **Математическая модель**

Для построения математической модели нам нужно рассмотреть алгоритм вычисления значения интеграла двумя способами. Программа получает на вход число, определяющее точность вычисления, eps=0.001. После этого программа вычисляет значение интеграла по формуле Симпсона в первом приближении, которое равно 0.0721471 и значение интеграла во втором приближении равное 0.236739. После этого мы находим модуль разности этих чисел и получаем число 0.164592. Т.к. это число больше степени точности, то программа находит значения интеграла в следующем приближении и находит модуль разности полученного значения с предыдущим. И так продолжается до тех пор, пока модуль не станет меньше степени точности. В итоге программа выведет значение интеграла, равное 0.283561. После этого программа вычисляет значение интеграла методом прямоугольников. Сначала она находит значение интеграла с использованием 2 прямоугольников равное -0.0659744. Потом количество прямоугольников увеличивается в два раза и находится значение интеграла равное 0.148694. Находится модуль разности между вторым значением и предыдущим. Численное значение которого равно 0.214669. Пока разность не будет меньше точности вычисления, количество прямоугольников будет увеличиваться в два раза. В итоге программа выведет число 0.283678 равное значению интеграла.

1. **Описание алгоритма**

**Начало алгоритма**

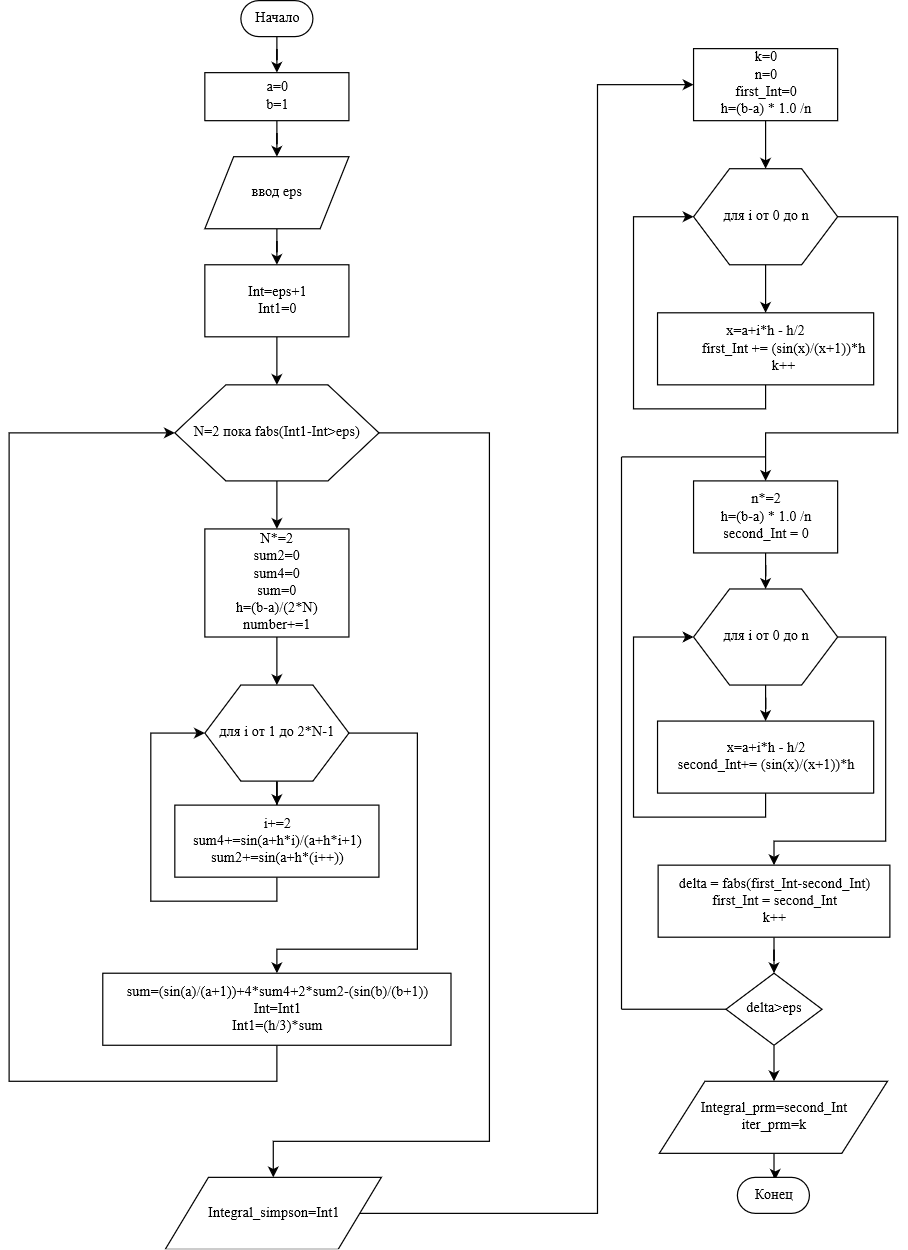
* 1. Присвоение переменной a значения 0
  2. Присвоение переменной b значения 1
  3. Ввод переменной eps
  4. Присвоение переменной Int значения eps+1
  5. Присвоение переменной Int1 значения 0
  6. Присвоение переменной N значения 2
  7. Если условие |Int1-Int|>eps не выполняется переход к пункту 4.20
  8. Присвоение переменной sum2 значения 0
  9. Присвоение переменной sum4 значения 0
  10. Присвоение переменной sum значения 0
  11. Присвоение переменной h значения (b-a)/(2\*N)
  12. Присвоение переменной I значения 1
  13. Если условие i<=2\*N-1 не выполняется переход к пункту 4.17
  14. Присвоение переменной sum4 значения sum4 + sin(a+h\*i)/(a+h\*i+1)
  15. Присвоение переменной sum2 значения sum2 + sin(a+h\*(i++))
  16. Присвоение переменной i значения i + 2
  17. Присвоение переменной sum значения (sin(a)/(a+1))+4\*sum4+2\*sum2-(sin(b)/(b+1))
  18. Присвоение переменной Int значения Int1
  19. Присвоение переменной Int1 значения (h/2.4)\*sum
  20. Присвоение переменной number значения number+1
  21. Вывод Integral\_simpson= Int1
  22. Вывод number
  23. Присвоение переменной k значения 0
  24. Присвоение переменной n значения 2
  25. Присвоение переменной first\_Int значения 0
  26. Присвоение переменной h значения (b-a) \* 1.0 /n
  27. Присвоение переменной i значения 0
  28. Если условие i < n не выполняется 4.31
  29. Присвоение переменной x значения a+i\*h - h/2
  30. Присвоение переменной first\_Int значения (sin(x)/(x+1))\*h
  31. Присвоение переменной k значения k+1
  32. Присвоение переменной n значения n\*2
  33. Присвоение переменной h значения (b-a) \* 1.0 /n
  34. Присвоение переменной second\_Int значения 0
  35. Присвоение переменной i значения 0
  36. Если условие i<n не выполняется переход 4.38
  37. Присвоение переменной x значения a+i\*h - h/2
  38. Присвоение переменной second\_Int значения (sin(x)/(x+1))\*h
  39. Присвоение переменной delta значения fabs(first\_Int-second\_Int)
  40. Присвоение переменной first\_Int значения second\_Int
  41. Присвоение переменной k значения k+1
  42. Если delta>eps переход к пункту 4.31
  43. Вывод Integral\_prm= second\_Int
  44. Вывод iter\_prm= k

**Конец алгоритма**

1. **Проект программы с нахождением значения интеграла по формуле Симпсона и методом прямоугольников**

Проект программы с определением замкнутых программных единиц и необходимых структур данных. В данном случае программа должна состоять из 23 частей: 1 блок ввода, цикла с параметром (содержит 5 блоков определения переменных, 1 блок проверки условия, 4 блока арифметических действий, 1 цикл с параметром[содержит блок проверки условия, два блока арифметических действий]) и 2 блока вывода, 4 блока арифметических действий, цикл с параметром(содержит блок определения переменной i, 2 блока арифметических действий). В качестве структуры данных могут быть введены переменные для хранения переменной определяющей точность вычислений, промежуточных переменных и результата.

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



1. **Текст программы:**

**на языке С++**

#include<iostream>

#include<cmath>

#include<windows.h>

using namespace std;

int main()

{

double a, number, b, eps;//Нижний и верхний пределы интегрирования (a, b), погрешность (eps).

number=0;

a=0;

b=1;

cout << "eps=";

std::cin >> eps;

double Int, Int1;

Int=eps+1;

Int1=0;

//I-предыдущее вычисленное значение интеграла, I1-новое, с большим N.

for (int N=2; (fabs(Int1-Int)>eps); N\*=2)

{

double h, sum2=0, sum4=0, sum=0;

h=(b-a)/(2\*N);//Шаг интегрирования

for (int i=1; i<=2\*N-1; i+=2)

{

sum4+=sin(a+h\*i)/(a+h\*i+1);//Значения с нечётными индексами, которые нужно умножить на 4.

sum2+=sin(a+h\*(i++));//Значения с чётными индексами, которые нужно умножить на 2.

}

sum=(sin(a)/(a+1))+4\*sum4+2\*sum2-(sin(b)/(b+1));//Отнимаем значение f(b) так как ранее прибавили его дважды.

Int=Int1;

// std::cout << "Int=" << Int << '\n';

Int1=(h/2.4)\*sum;

// std::cout << "Int1=" << Int1 << '\n';

// std::cout << "delta=" << fabs(Int1-Int) << '\n';

number++;

}

cout << "Integral\_simpson=" << Int1 << endl;

cout << "iter\_simpson=" << number << endl;

int k=0;

int n=2;

double delta, h, x, first\_Int, second\_Int;

first\_Int = 0;

int i;

h=(b-a) \* 1.0 /n;

for(i=0;i<n;i++)

{

x=a+i\*h - h/2;

first\_Int += (sin(x)/(x+1))\*h;

//std::cout << "first\_Int= " << first\_Int << '\n';

k++;

}

do

{

n\*=2;

h=(b-a) \* 1.0 /n;

second\_Int = 0;

for (int i = 0; i < n; i++)

{

x=a+i\*h - h/2;

second\_Int+= (sin(x)/(x+1))\*h;

}

//std::cout << "second\_Int= " << second\_Int << '\n';

delta = fabs(first\_Int-second\_Int);

//std::cout << "delta= " << delta << '\n';

first\_Int = second\_Int;

k++;

} while (delta>eps);

std::cout << "Integral\_prm=" << second\_Int << '\n';

cout << "iter\_prm=" << k << endl;

return 0;}

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

Для проверки работоспособности программы, переменной eps присвоим значение 0.001. Тогда программа выведет четыре значения:

* + - 1. Integral\_simpson= 0.283678
      2. iter\_simpson=681
      3. Integral\_prm= 0.283678
      4. iter\_prm=1521

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

Ввиду простоты программы она одинаково эффективно выполняется для обоих языков программирования.