МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФГБОУ ВО АЛТАЙСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Физико-технический факультет

Кафедра вычислительной техники и электроники (ВТиЭ)

Лабораторная работа № 04

**Программирование с использованием циклов**

Выполнил студент 585 гр.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Г.Е. Хлыстов

Проверил: к.т.н,, доцент каф. ВТиЭ

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

Лабораторная работа защищена

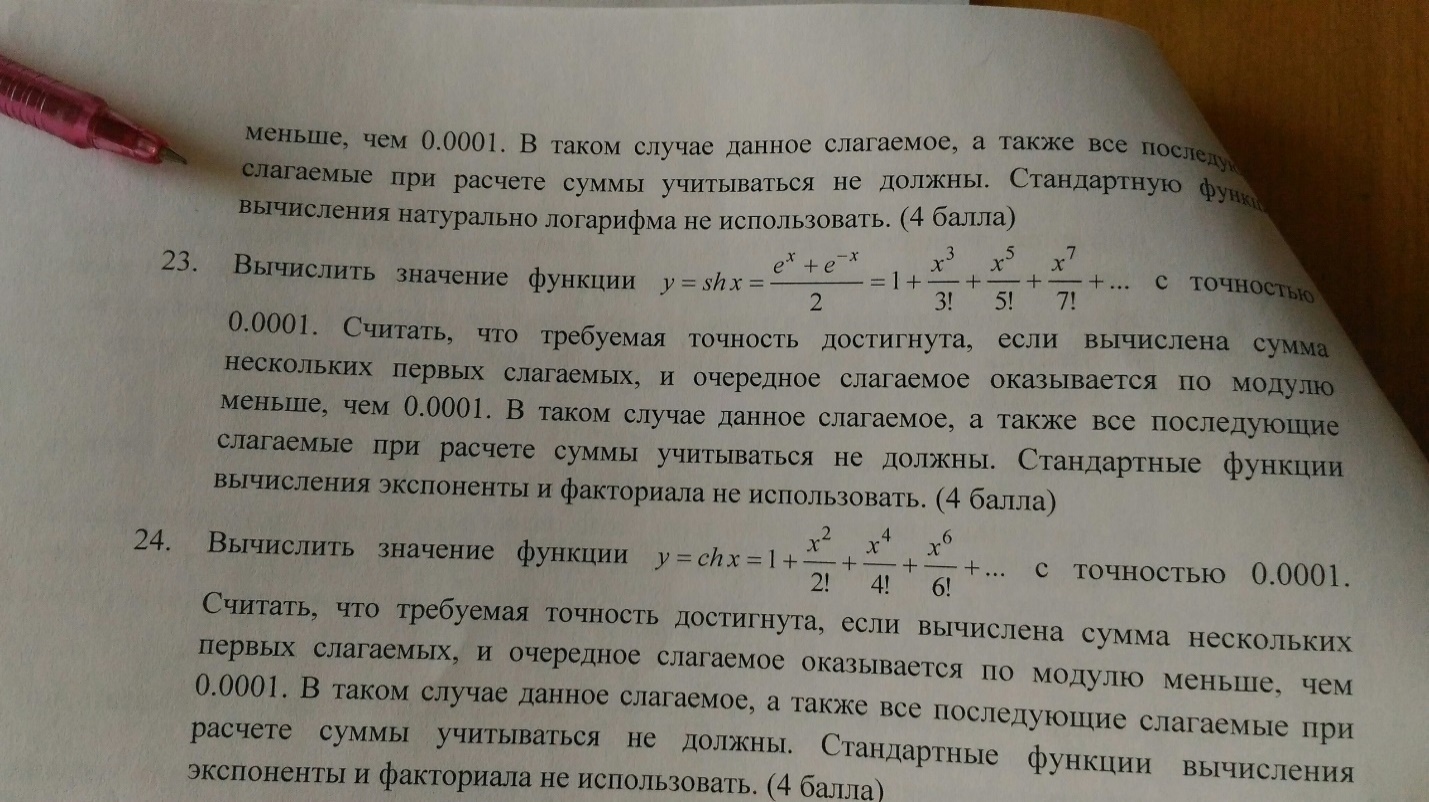
«\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2019 г.

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

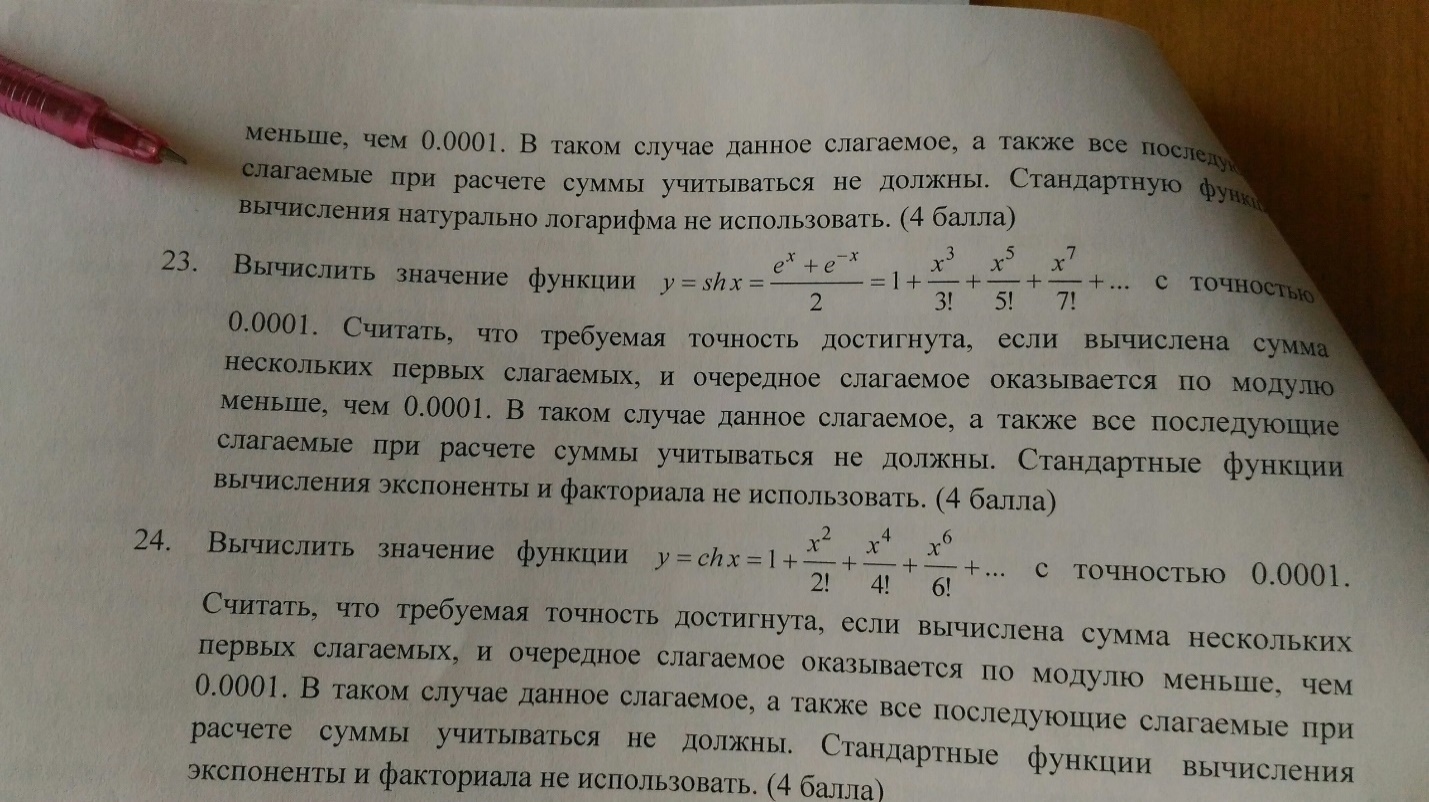
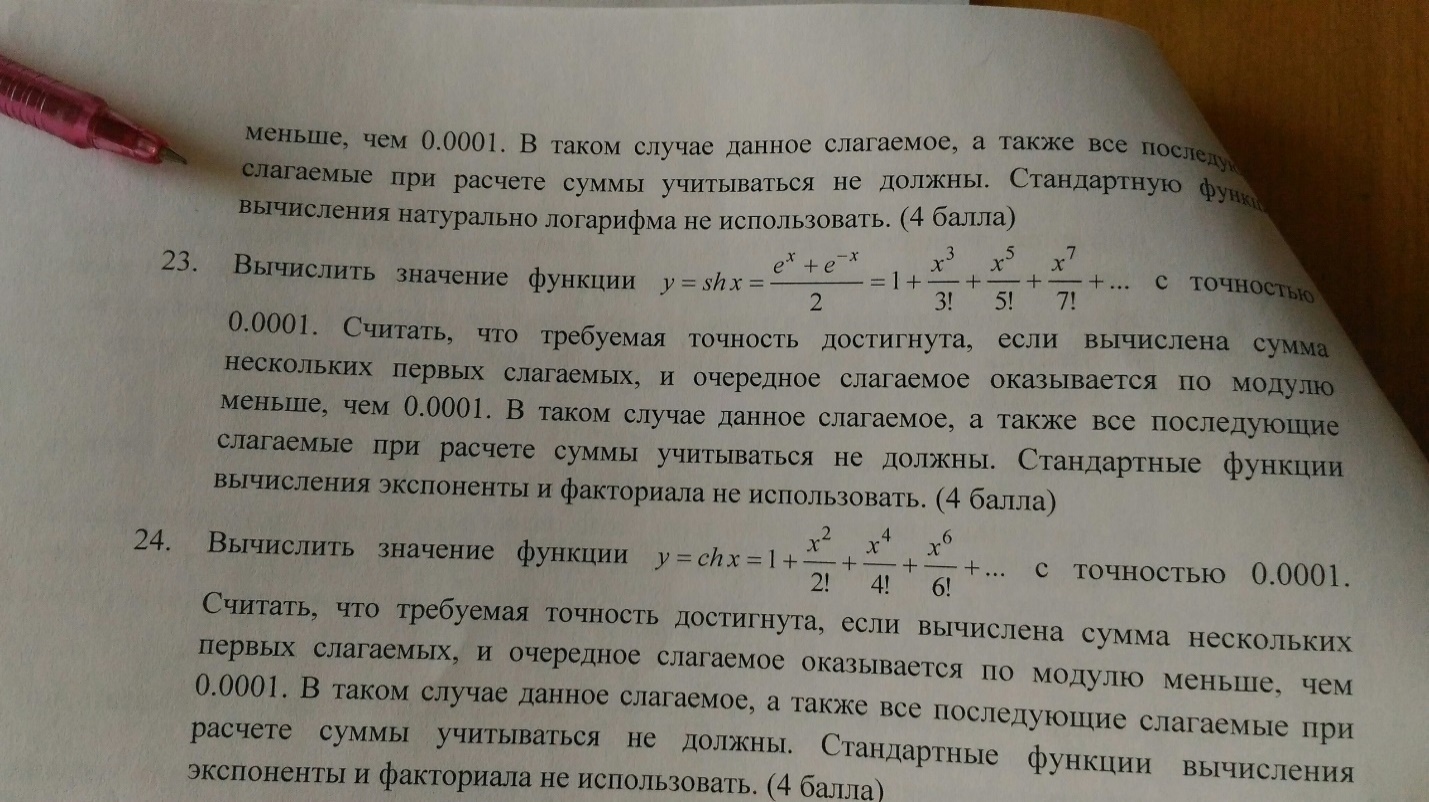
Барнаул 2018

**Задача №1**

1. **Формулировка задачи**

Вычислить значение функции  с точностью 0.0001. Считать, что требуемая точность достигнута, если вычислена сумма нескольких первых слагаемых, и очередное слагаемое оказывается по модулю меньше, чем 0.0001. В таком случае данное слагаемое, а также все последующие слагаемые при расчете суммы учитываться не должны.

1. **Постановка задачи**

На вход поступает вещественное число x. Программа будет рассчитывать сумму числового ряда, заданного функцией  , до тех пор, пока одно из слагаемых не будет меньше точности равной 0.0001. На выход будет поступать вещественное число, равное сумме числового ряда функции .

1. **Математическая модель**

Для расчета функции необходимо рассчитывать слагаемые, сравнивая их с точностью. Пока слагаемые больше 0.0001 слагаемые будут суммироваться. Например, на вход поступает x=1. Дальше происходит сложение, изначально сумма равна 1, к ней высчитывается слагаемое равное = , 1/6 > 0,0001 поэтому мы прибавляем слагаемое к сумме и продолжаем высчитывать и прибавлять слагаемые, до слагаемого 1/9!, так как оно будет меньше 0,0001, поэтому сложение закончиться и на выход пойдет значение суммы числового ряда равная 1,1752.

1. **Описание алгоритма**

**Начало алгоритма основной программы**

1. На вход поступает x
2. x\_f=1
3. n=3
4. Проверка условия: если выполняется (pow (x, n))/fact(n)>eps), программа продолжает работу, иначе переходит к пункту 7
5. x\_f=x\_f+(pow (x, n))/fact(n)
6. n=n+2, переход к пункту 4
7. Вывод x\_f

**Конец алгоритма основной программы**

**Начало алгоритма fact**

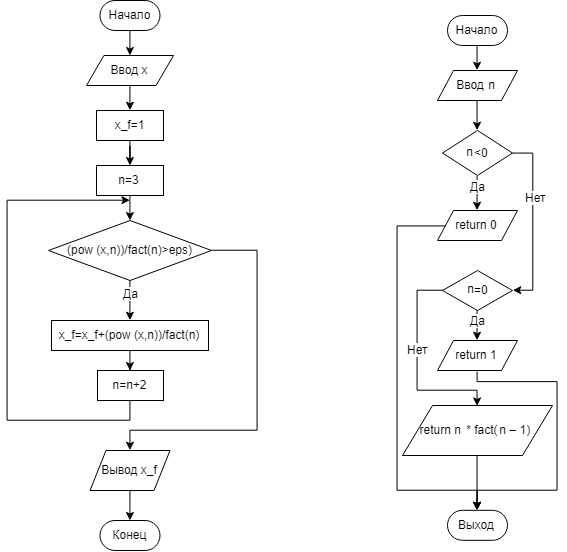
1. На вход поступает n
2. Проверка условия n<0. Если выполняется, возвращаем 0, производится выход из подпрограммы
3. Проверка условия n=0. Если выполняется, возвращаем 1, производится выход из подпрограммы, иначе возвращаем значение функции, производится выход из подпрограммы

**Конец алгоритма fact**

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

В данном случае программа состоит из трех частей – блок ввода, блок сравнения и блока вывода. А также в ней присутствует подпрограмма, состоящая из шести частей – блок ввода, два блока сравнения и три блока вывода.

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



1. **Текст программы:**

**на языке С++**

#include <iostream>

#include <math.h>

#include <conio.h>

using namespace std;

long double fact(int N)

{

if(N < 0) // если пользователь ввел отрицательное число

return 0; // возвращаем ноль

if (N == 0) // если пользователь ввел ноль,

return 1; // возвращаем факториал от нуля

else // Во всех остальных случаях

return N \* fact(N - 1); // делаем рекурсию.

}

int main()

{

setlocale (LC\_ALL, "RUSSIAN");

double eps=0.0001, x\_f;

int n, x;

cout << "vvedite x " << '\n';

cin >> x;

x\_f=1;

n=3;

while ((pow (x, n))/fact(n)>eps){

x\_f=x\_f+(pow (x, n))/fact(n);

n=n+2;

}

cout << x\_f;

\_getch();

return 0;

}

**на языке Pascal**

program project1;

var eps, x\_f:double;

n, x: integer;

function fact(n:integer):longint;

begin

if n=1 then fact:=1

else fact:=fact(n-1)\*n;

end;

begin

write ('vvedite x ');

read (x);

eps:=0.0001;

x\_f:=1;

n:=3;

while ((exp(ln(x) \* n)/fact(n))>eps) do

begin

x\_f:=x\_f+(exp(ln(x) \* n))/fact(n);

n:=n+2;

end;

writeln (x\_f);

read(x);

end.

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

На вход поступает x=1. Дальше происходит сложение, изначально сумма равна 1, к ней высчитывается слагаемое равное = , 1/6 > 0,0001 поэтому мы прибавляем слагаемое к сумме и продолжаем высчитывать и прибавлять слагаемые, до слагаемого 1/9!, так как оно будет меньше 0,0001, поэтому сложение закончиться и на выход пойдет значение суммы числового ряда равная 1,1752.

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

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

**Задача №2**

1. **Формулировка задачи**

По формуле Симпсона и методом прямоугольников вычислить приближенное значение интеграла . Точность не превышает 0.0001. Сравнить методы вычисления по количеству итераций.

1. **Постановка задачи**

Программа рассчитывает интеграл двумя способами ведя подсчет итераций отдельно в каждом способе. На выход идут два вещественных числа, являющиеся решением интеграла, а также два положительных действительных числа, являющиеся количеством итераций.

1. **Математическая модель**

Для построения математической модели нужно построить алгоритм вычисления значения интеграла двумя способами. Программа имеет значение точности вычисления, равное 0.0001. Сперва программа высчитывает интеграл по методу прямоугольников, с использованием 2 прямоугольников равное 1.40254. Потом количество прямоугольников увеличивается в два раза и получаем значение интеграла равное 1.40495. Находиться модуль разности между вторым и предыдущим значением, оно будет равно 0,00241. Пока разность не будет меньше точности, количество прямоугольников будет расти. В итоге программа выведет число 1.4058, равное значению интеграла. После этого программа вычисляет значение интеграла по формуле Симпсона, в первом приближении равное 1.41395, а во втором приближении равное 1.4094. Далее находиться модуль разности первого и второго приближений равное 0,0036. Так как это число больше значения точности, программа продолжит искать значение интеграла в следующих приближениях пока их модуль разности не будет меньше точности вычисления. В итоге программа выведет значение интеграла равное 1.4058.

1. **Описание алгоритма**

**Начало алгоритма основной программы**

1. n=4;
2. sch\_1=0; //счетчик итераций в первом методе
3. sch\_2=0; //счетчик итераций во втором методе
4. a=0.2;
5. b=1.8;
6. eps=0.0001;
7. S\_1=0; //первое приближение значения площади
8. h=(b-a)/n;
9. i=1

//Метод прямоугольников

1. Проверка условия: пока i<=n программа продолжает работу, иначе переходит к пункту 14
2. i++
3. x=a+i\*h-h/2;
4. S\_1=S\_1+f(x)\*h, переход к пункту 10
5. n=2\*n;
6. h=(b-a)/n;
7. S\_2=0;
8. i=1
9. Проверка условия: условия: пока i<=n программа продолжает работу, иначе переходит к пункту 23
10. i++
11. x=a+i\*h-h/2;
12. S\_2=S\_2+f(x)\*h;
13. sch\_1++, переходит к пункту 18
14. delta=fabs(S\_1-S\_2);
15. S\_1=S\_2;
16. Проверка условия: (delta>eps) если выполняется, то переходит к пункту 14, иначе продолжает работу программы
17. Вывод S\_1

//Метод Симпсона

1. h=0;
2. Проверка условия: i < n если выполняется, то продолжает работу иначе переходит к пункту 33
3. h=(b-a)/n;
4. k1 += f(a + i\*h);
5. k2 += f(a + (i+1)\*h);
6. sch\_2++, переход к пункту 28
7. S\_2 = h/3\*(f(a) + 4\*k1 + 2\*k2)
8. S\_1 = S\_2
9. n = 2 \* n
10. Проверка условия: i < n если выполняется, то продолжает работу иначе переходит к пункту 41
11. h=(b-a)/n;
12. k1 += f(a + i\*h);
13. k2 += f(a + (i+1)\*h);
14. sch\_2++, переход к пункту 36
15. S\_2=h/3\*(f(a) + 4\*k1 + 2\*k2)
16. Проверка условия: (fabs(S\_2 - S\_1) > eps) если выполняется, то переходит к пункту 34
17. Вывод S\_2

**Конец алгоритма основной программы**

**Начало алгоритма f**

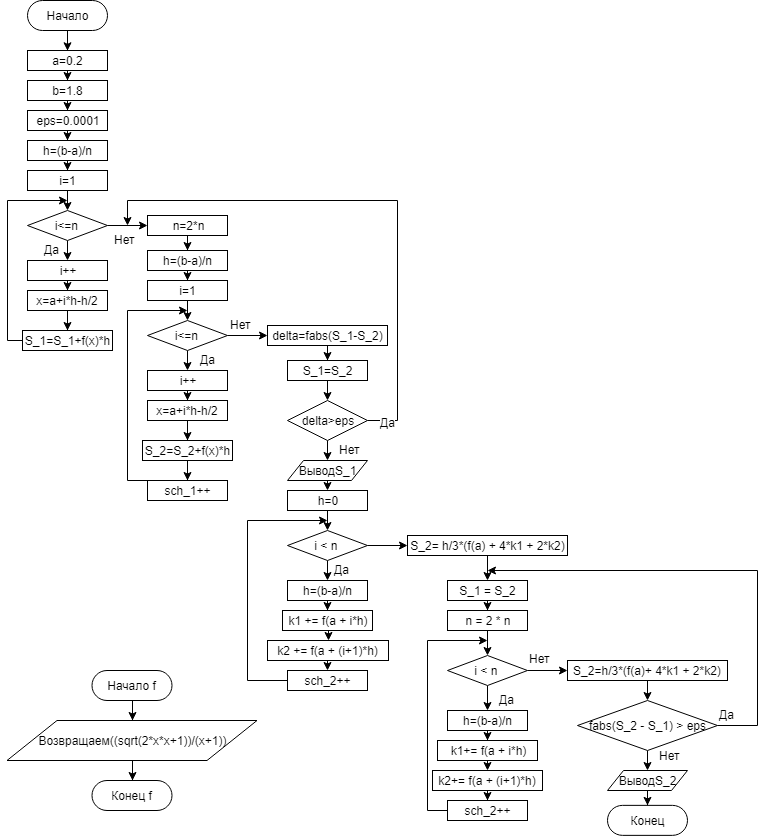
return ((sqrt(2\*x\*x+1))/(x+1))

**Конец алгоритма f**

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

В данном случае программа состоит из 5 циклов, 24 арифметических блоков, из 6 переходов, 2 блоков вывода. Так же в программе существует подпрограмма f состоящая из одного блока вывода.

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



1. **Текст программы:**

**на языке С++**

#include <iostream>

#include <math.h>

#include <conio.h>

typedef double(\*pointFunc)(double);

double f(double x)

{

return ((sqrt(2\*x\*x+1))/(x+1));

}

int main() {

long i, n=4;

double a, b, x, y, s, h, S\_1, S\_2, eps, delta, k1 = 0, k2 = 0;

int sch\_1, sch\_2;

sch\_1=0;

a=0.2;

b=1.8;

eps=0.0001;

S\_1=0;

h=(b-a)/n;

for (i=1; i<=n; i++)

{

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

S\_1=S\_1+f(x)\*h;

}

do

{

n=2\*n;

h=(b-a)/n;

S\_2=0;

for (i=1; i<=n; i++)

{

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

S\_2=S\_2+f(x)\*h;

sch\_1++;

}

delta=fabs(S\_1-S\_2);

S\_1=S\_2;

} while (delta>eps);

std::cout<<"S="<<S\_1<<std::endl;

std::cout<<"sch="<<sch\_1<<std::endl;

S\_1=0;

S\_2=0;

h=0;

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

h=0;

h=(b-a)/n;

k1 += f(a + i\*h);

k2 += f(a + (i+1)\*h);

sch\_2++;

}

S\_2 = h/3\*(f(a) + 4\*k1 + 2\*k2); //первое приближение для интеграла

do {

S\_1 = S\_2; //второе приближение

n = 2 \* n; //увеличение числа шагов в два раза,

for(int i = 1; i < n; i += 4) {

h=(b-a)/n;

k1 += f(a + i\*h);

k2 += f(a + (i+1)\*h);

sch\_2++;

// std::cout << "S=" << S\_2 << std::endl;

}

S\_2=h/3\*(f(a) + 4\*k1 + 2\*k2);

}

while (fabs(S\_2 - S\_1) > eps); //сравнение приближений с заданной точностью

std::cout << "S=" << S\_2 << std::endl;

std::cout<<"sch="<<sch\_2<<std::endl;

\_getch();

return 0;

} **на языке Pascal**

program Method\_Prm;

const

e = 0.0001;

var

i, n: word;

a, b, S\_1, S\_2, h, s, x, sch\_1, sch\_2: real;

function F(x:real):real;

begin

F:=((sqrt(2\*x\*x+1))/(x+1));

end;

begin

a:=0.2;

b:=1.8;

n:= 2;

S\_1:= 0;

sch\_1:=0;

sch\_2:=0;

repeat

S\_2:= S\_1;

h:= (b - a)/n;

x:= a; s:= 0;

for i:= 1 to n do begin

s:= s + F(x);

x:= x + h;

sch\_1:=sch\_1+1;

end;

S\_1:= s\*h;

n:= n\*2

until abs(S\_1 - S\_2)<e;

writeln('S=', S\_1:0:6);

writeln('n1=', sch\_1:0:6);

h:=0;

n:=2;

S\_1:=0;//итоговый интеграл

h:=(b-a)/n;//шаг

S\_2:=h\*F(a);//текущий интеграл

repeat

S\_1:=S\_2;//запомним текущий

h:=(b-a)/n;

x:=a;//левый край

for i:=1 to n-1 do

begin

x:=x+h;//наращиваем

if i mod 2 = 0 then S\_2:=S\_2+2\*F(x)//считаем новый интеграл

else S\_2:=S\_2+4\*f(x);

sch\_2:=sch\_2+1;

end;

S\_2:=(S\_2+F(a)+F(b))\*h/3;//конечное значение

n:=n\*2;//если точность не достигнута удваиваем количество разбиений

until abs(S\_1-S\_2)<=e;

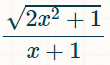
writeln('S=',S\_2:0:6);

writeln('n2=', sch\_2:0:6);

readln;

end.

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

Вычисляем первообразную функции ****. ПолучаетсяC:\Users\Gleb\Downloads\joxi_screenshot_1559373967122.png

Далее считаем определенный интеграл, и в приближении число равно 1.4058.

В итоге программа, сделанная способом прямоугольников, считает интеграл с меньшим количеством итераций.

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

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