Министерство науки и высшего образования Российской Федерации

Федеральное государственное автономное образовательное учреждение высшего образования

**«Пермский национальный исследовательский политехнический университет»**

Электротехнический факультет

Кафедра «Информационные технологии и автоматизированные системы»

направление подготовки: 09.03.04 - «Программная инженерия»

**Лабораторная работа №1**

**«Решения нелинейных уравнений»**

**Вариант №9**

Выполнил студент гр. РИС-24-1б

Иванов Семен Сергеевич

Проверил:

Доц. каф. ИТАС

Ольга Андреевна Полякова

(оценка) (подпись)

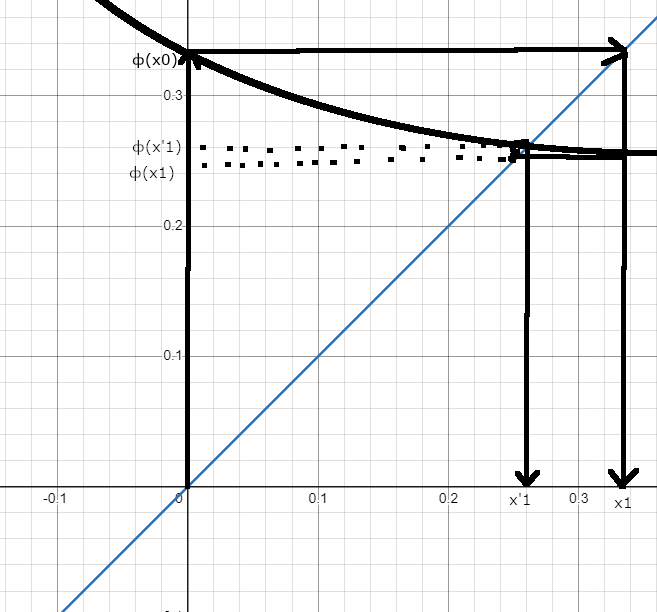
(дата)

# Метод итераций

**Постановка задачи**

При помощи метода итераций найти корень уравнения x-(1/(3+sin(3.6x)))=0 с заданной пользователем точностью вычислений на отрезке [0; 0.85].

**Геометрическая интерпретация метода**

****

**Обоснование стороны подхода к функции**

Пусть F(x) =

Уравнение F(x) = 0 преобразуется в уравнение вида .

Находится

Находится первое приближение к корню по формуле:

Подставляется а:

**Вывод формулы нахождения корня**

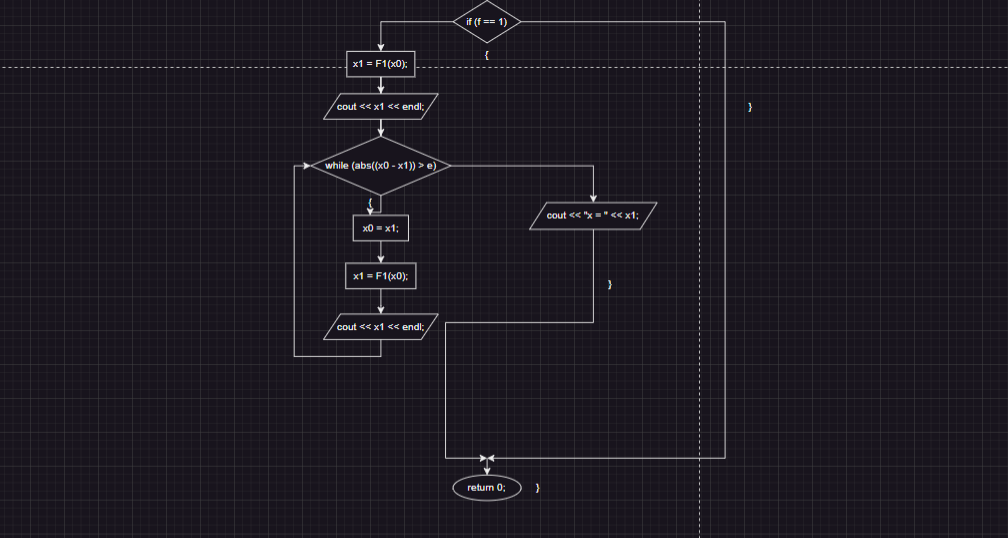
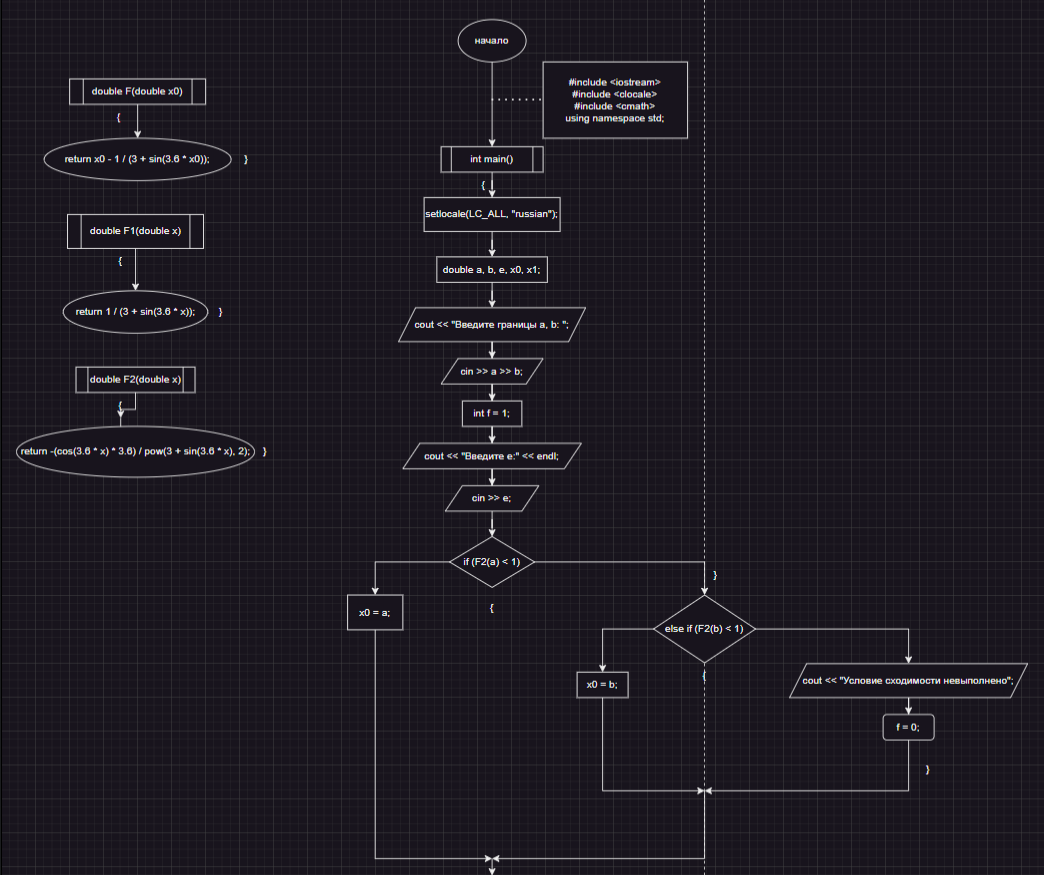
1. Вычисляется следующее приближение к корню по формуле:

2. При получении нужной точности по формуле:  приближение к корню останавливается.

– является найденным корнем уравнения

3. Если нужная точность не достигнута, то продолжается приближение корня по формуле (1)

# Блок-схема



**Код**

#include <iostream>

#include <clocale>

#include <cmath>

using namespace std;

double F(double x0){

return x0 - 1 / (3 + sin(3.6 \* x0));

}

double F1(double x){

return 1 / (3 + sin(3.6 \* x));

}

double F2(double x){

return -(cos(3.6 \* x) \* 3.6) / pow(3 + sin(3.6 \* x), 2);

}

int main(){

setlocale(LC\_ALL, "russian");

double a, b, e, x0, x1;

cout << "Введите границы a, b: ";

cin >> a >> b;

int f = 1;

cout << "Введите e:" << endl;

cin >> e;

if (F2(a) < 1)

{

x0 = a;

}

else if (F2(b) < 1)

{

x0 = b;

}

else

{

cout << "Условие сходимости невыполнено";

f = 0;

}

if (f == 1)

{

x1 = F1(x0);

cout << x1 << endl;

while (abs((x0 - x1)) > e){

x0 = x1;

x1 = F1(x0);

cout << x1 << endl;

}

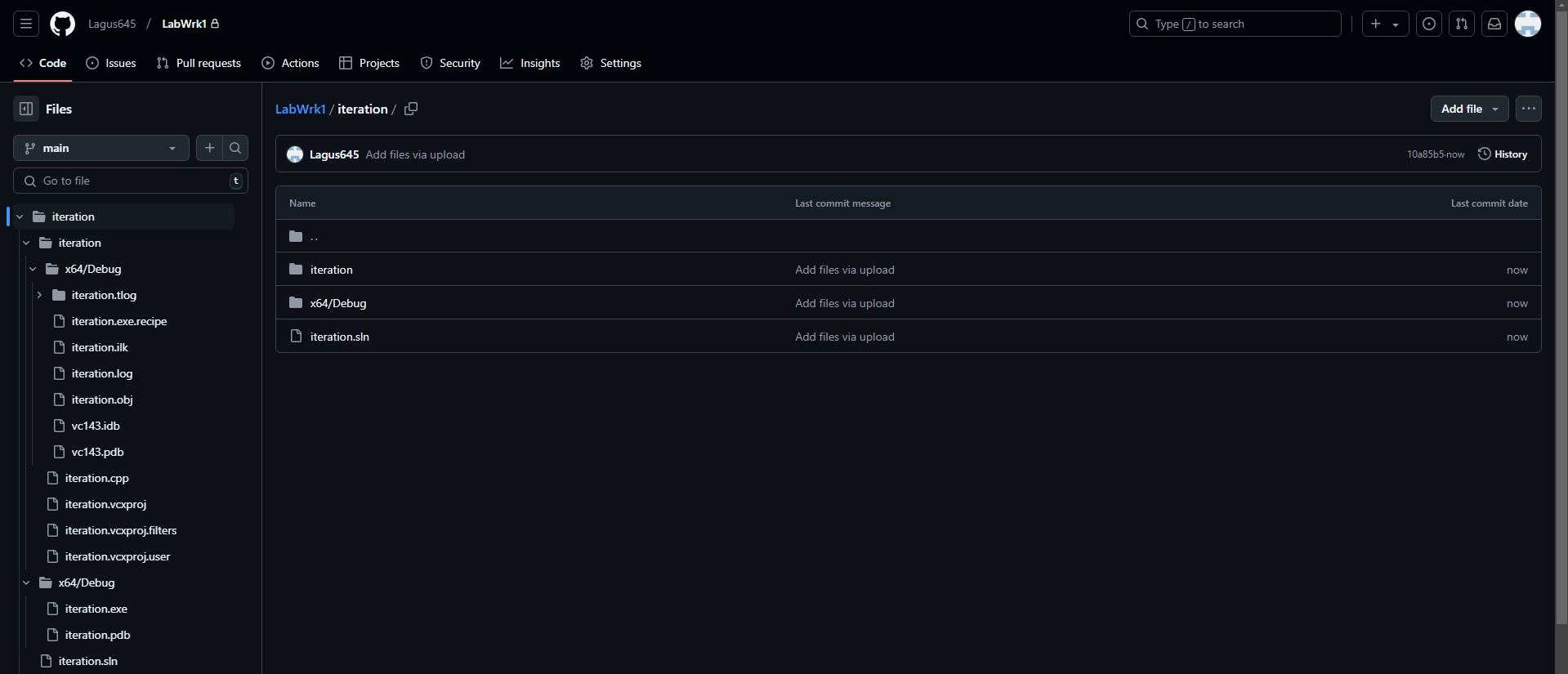
cout << "x = " << x1;

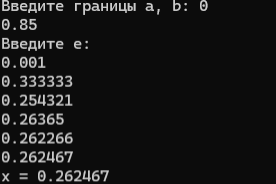
}

return 0;

}

**Скриншот из GitHub и результаты работы**

****

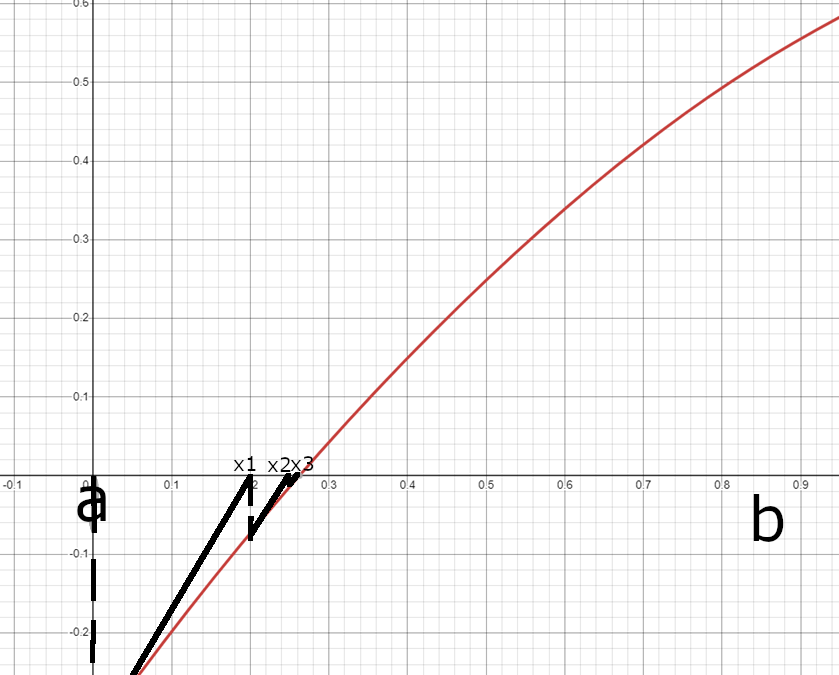
****

# Метод Ньютона

**Постановка задачи**

При помощи метода Ньютона найти корень уравнения x-(1/(3+sin(3.6x)))=0 с заданной пользователем точностью вычислений на отрезке [0; 0.85].

**Графическая интерпретация метода**

****

**Обоснование стороны подхода к функции**

Заметим, что f(a) \* f”(a)>0, а f(b) \* f”(b)<=0, значит x0 = a.

**Вывод формулы нахождения корня**

1. Геометрический смысл первой производной тангенс угла наклона касательной и Ох, то есть f'(x) = tg(а)= k

2. Запишем уравнение прямой с угловым коэффициентом у = k\*x + b.

3. Найдем уравнение касательной в точке х0: f(x) = f'(x0)\*x0+b

4. Выразим b: b = f(x)-f'(x)\*x0

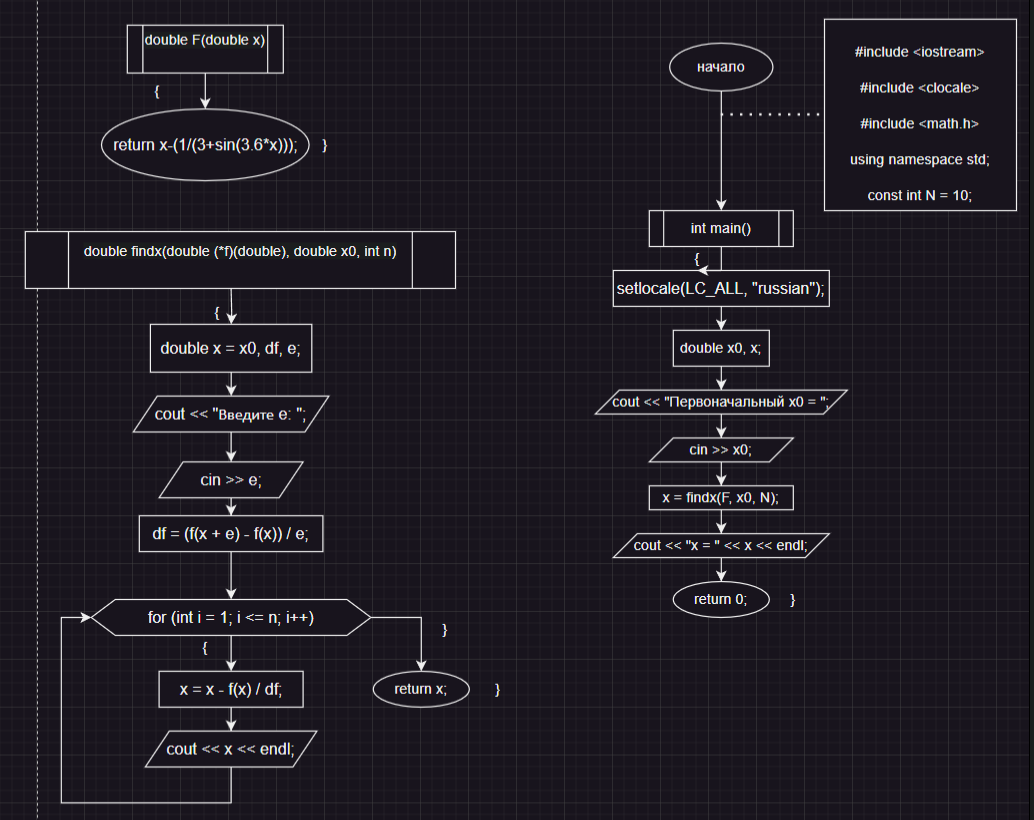
5. Теперь запишем уравнение касательной в новом виде, подставив выражение в, полученное в 4 пункте: y=f’(x0)\*x+f(x0)-f’(x0)\*x0

6. Преобразуем уравнение, вынеся общий множитель f'(x0): y = f'(x0)\*(x-x0)+f(x0)

7. Так как нам нужно найти точку пересечения уравнения из 6 пункта с Ох, то приравняем его к нулю: f'(x0)\*(x-x0)+f(x0)=0

8. Выражаем к пересечение уравнения касательной с Ох: x = x0 – (f(x0)/f’(x0))

**Блок-схема**



**Код**

#include <iostream>

#include <clocale>

#include <math.h>

using namespace std;

const int N = 10;

double F(double x) {

return x-(1/(3+sin(3.6\*x)));

}

double findx(double (\*f)(double), double x0, int n) {

double x = x0, df, e;

cout << "Введите e: ";

cin >> e;

df = (f(x + e) - f(x)) / e;

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

x = x - f(x) / df;

cout << x << endl;

}

return x;

}

int main() {

setlocale(LC\_ALL, "russian");

double x0, x;

cout << "Первоначальный x0 = ";

cin >> x0;

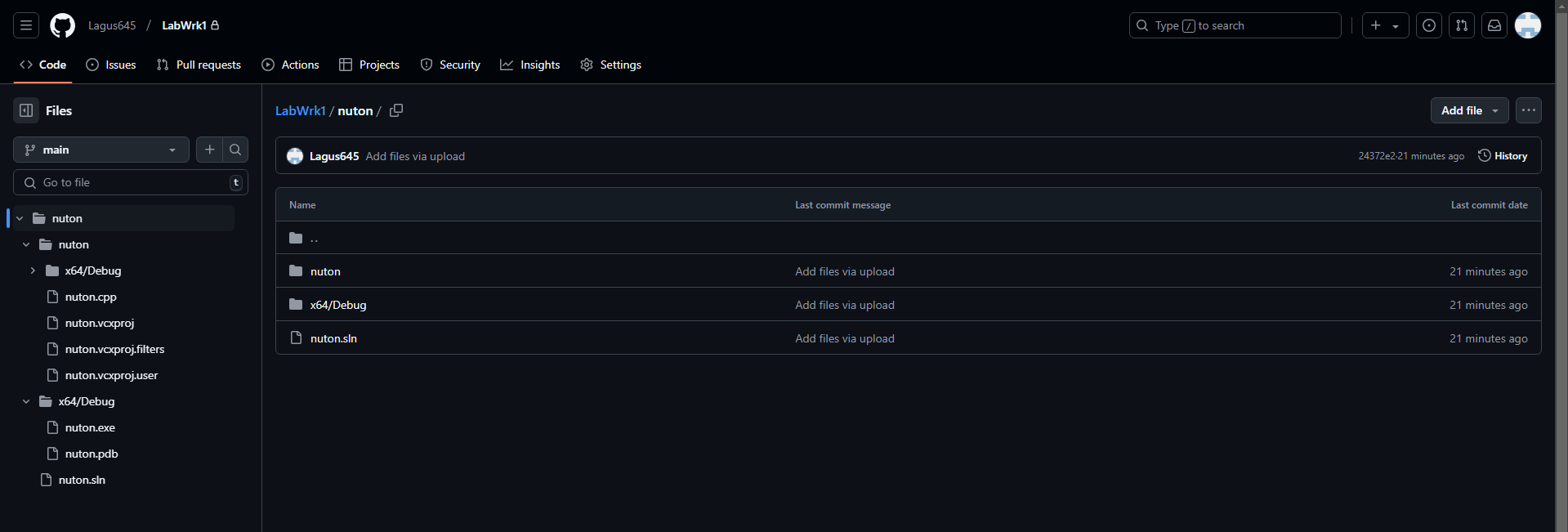
x = findx(F, x0, N);

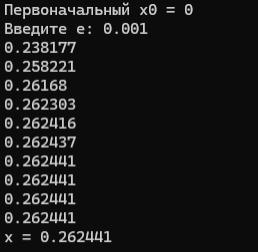
cout << "x = " << x << endl;

return 0;

}

**Скриншот из GitHub и результаты работы**

****

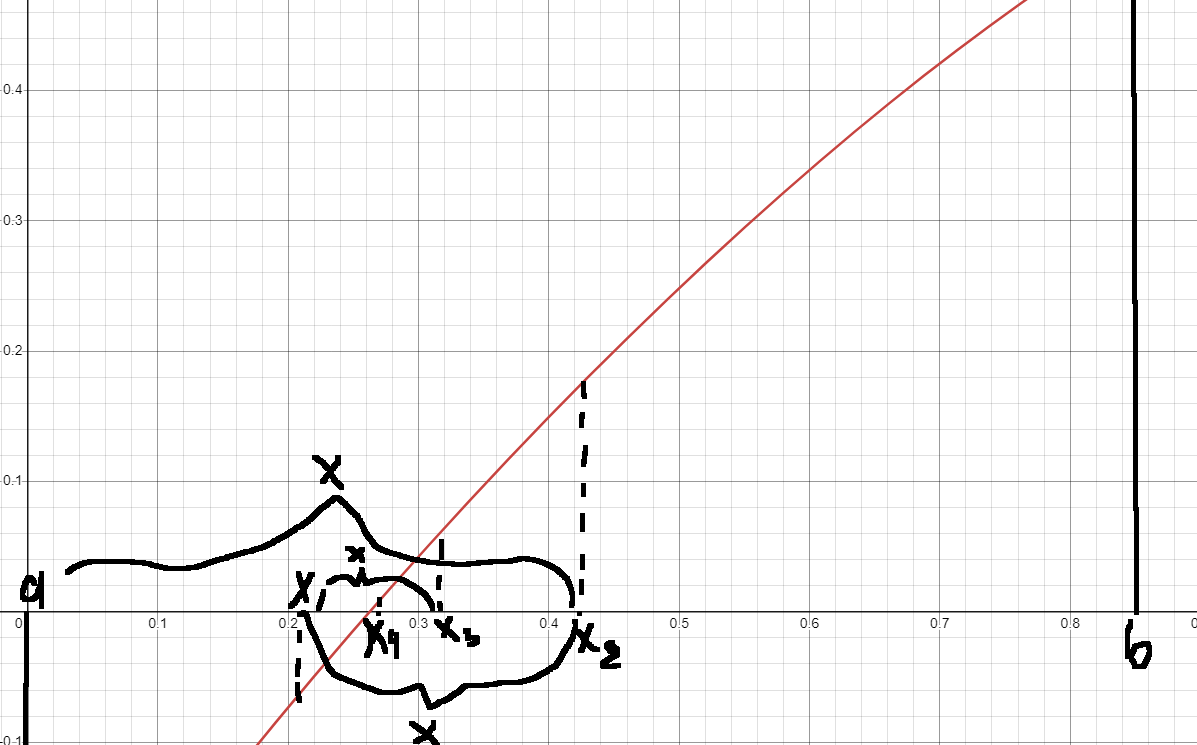
****

**Метод половинного деления**

**Постановка задачи**

При помощи метода половинного деления найти корень уравнения x-(1/(3+sin(3.6x)))=0 с заданной пользователем точностью вычислений на отрезке [0; 0.85].

**Графическая интерпретация метода**



**Метод решения**

Метод применим, если:

1. Известен интервал, на котором функция монотонна и непрерывна.

2. f(a) \* f(b) < 0

Суть метода заключается в делении интервала пополам

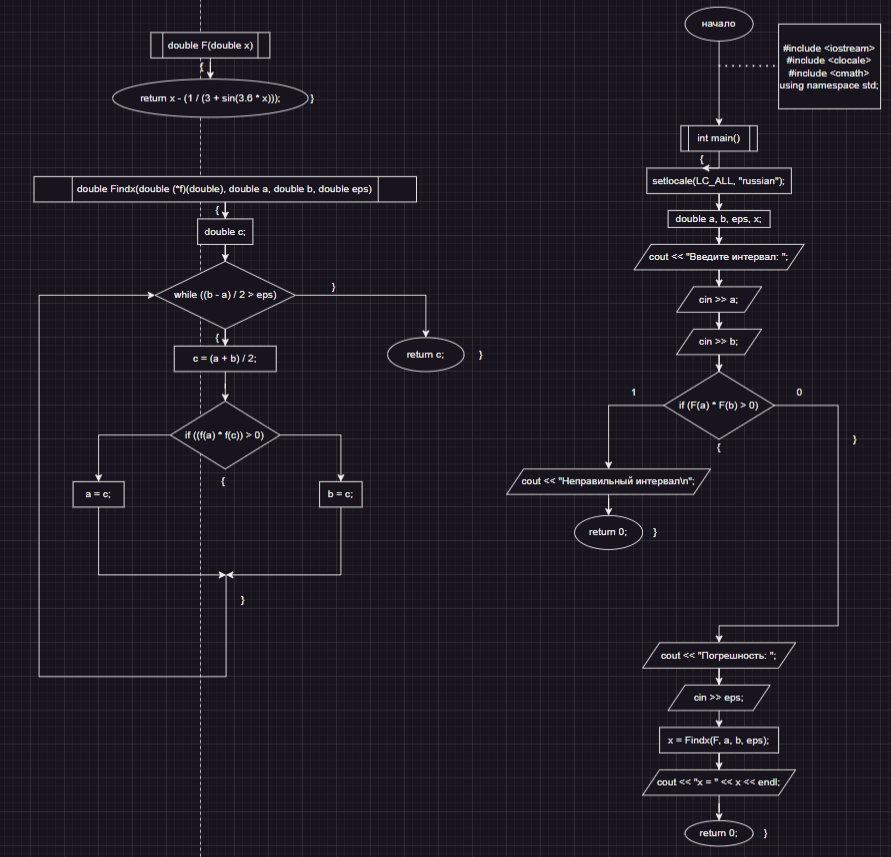
C= (a+b)/2

И отбрасывании той части интервала, в которой нет корня, те условие (2) не выполняется.

Оставшаяся часть корня является новым отрезком, и итерации будут продолжаться, пока расстояние между а и b не будет меньше или равно выбранной погрешности.

**Решение через if**

**Блок-схема**

****

**Код**

#include <iostream>

#include <clocale>

#include <cmath>

using namespace std;

double F(double x) {

return x - (1 / (3 + sin(3.6 \* x)));

}

double Findx(double (\*f)(double), double a, double b, double eps) {

double c;

while ((b - a) / 2 > eps) {

c = (a + b) / 2;

if ((f(a) \* f(c)) > 0) a = c;

else b = c;

}

return c;

}

int main() {

setlocale(LC\_ALL, "russian");

double a, b, eps, x;

cout << "Введите интервал: ";

cin >> a;

cin >> b;

if (F(a) \* F(b) > 0) {

cout << "Неправильный интервал\n";

return 0;

}

cout << "Погрешность: ";

cin >> eps;

x = Findx(F, a, b, eps);

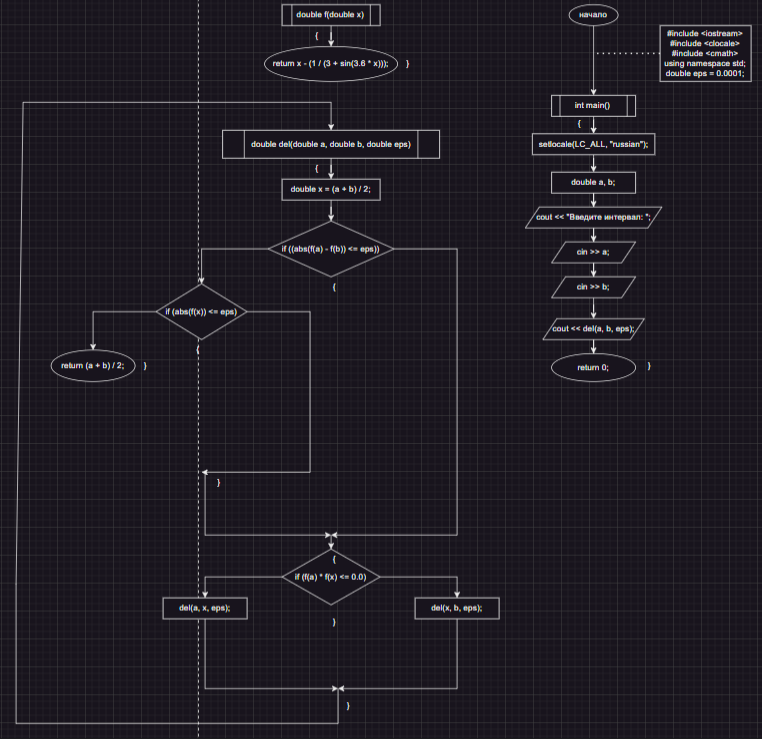
cout << "x = " << x << endl;

return 0;

}

**Решение через рекурсию**

**Блок-схема**

****

**Код**

#include <iostream>

#include <clocale>

#include <cmath>

using namespace std;

double eps = 0.0001;

double f(double x){

return x - (1 / (3 + sin(3.6 \* x)));

}

double del(double a, double b, double eps){

double x = (a + b) / 2;

if ((abs(f(a) - f(b)) <= eps)){

if (abs(f(x)) <= eps){

return (a + b) / 2;

}

}

if (f(a) \* f(x) <= 0.0){

del(a, x, eps);

}

else{

del(x, b, eps);

}

}

int main(){

setlocale(LC\_ALL, "russian");

double a, b;

cout << "Введите интервал: ";

cin >> a;

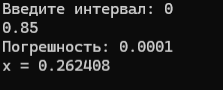
cin >> b;

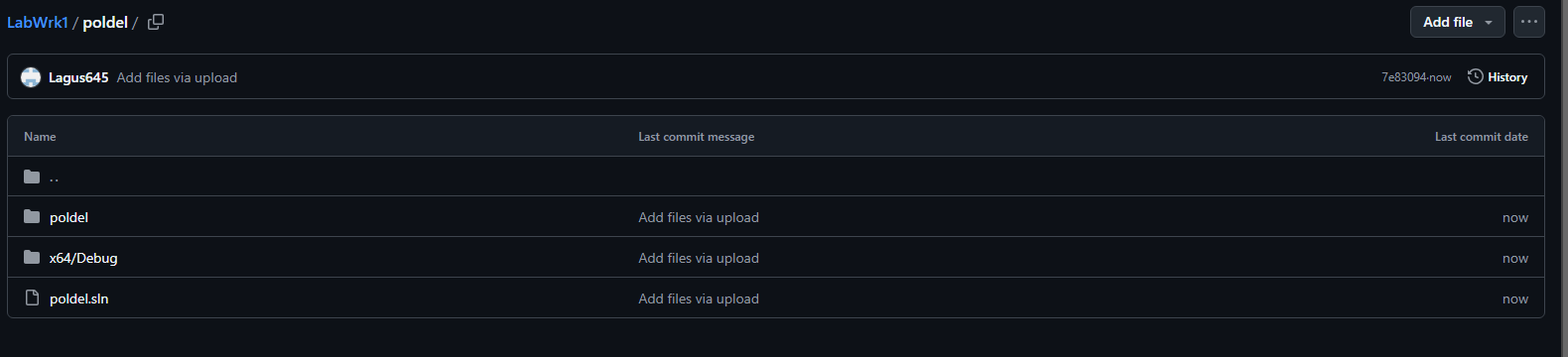
cout << del(a, b, eps);

}

**Скриншот из GitHub и результаты работы**

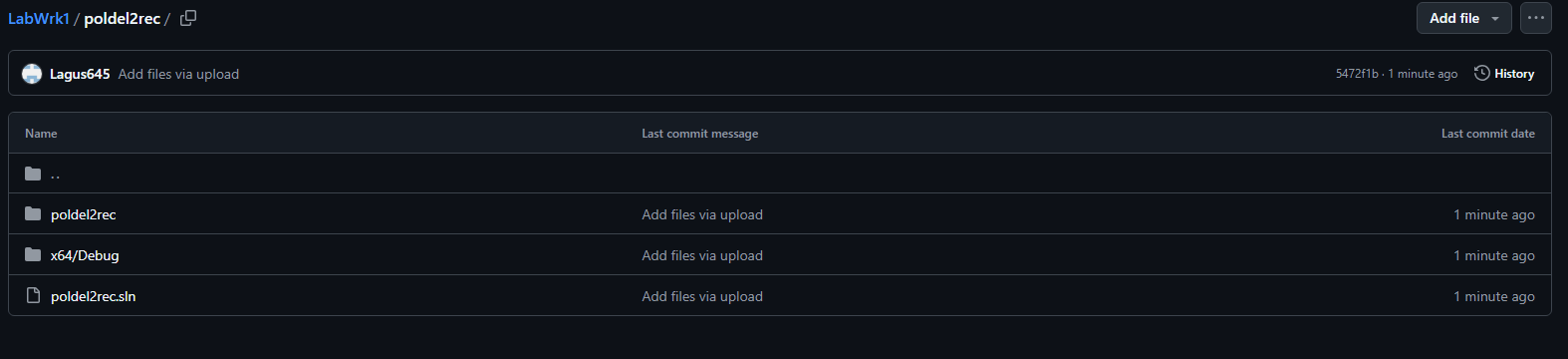
**if**

****

****

**Рекурсия**

****

****