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  Сурнин Кирилл Александрович |
|  | Проверил:  Доц. каф. ИТАС  Ольга Андреевна Полякова  \_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_  *(оценка) (подпись)*  \_\_\_\_\_\_\_\_\_\_\_\_\_  *(дата)* |

Пермь, 2024

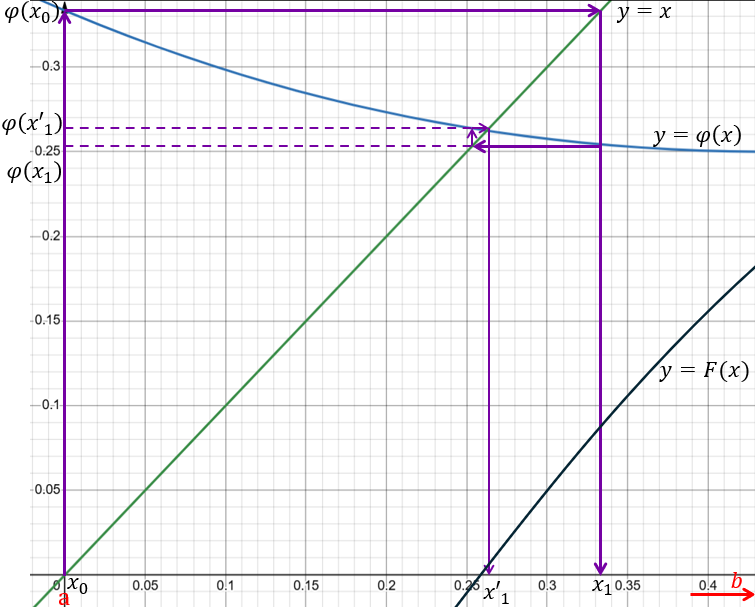
1. **Постановка задачи**

Найти корень уравнения на отрезке [0; 0,85] с заданной пользователем точностью вычислений (ɛ).

Решить уравнение с помощью трёх методов: метод простых итераций, метод Ньютона и метод половинного деления + рекурсивный способ.

**Метод простых итераций**

1. **Геометрическая интерпретация метода**

****

1. **Обоснование стороны подхода к функции**
   1. Пусть F(x) =

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

Находится

* 1. Находится первое приближение к корню по формуле:

Подставляется а:

Первое приближение к корню:

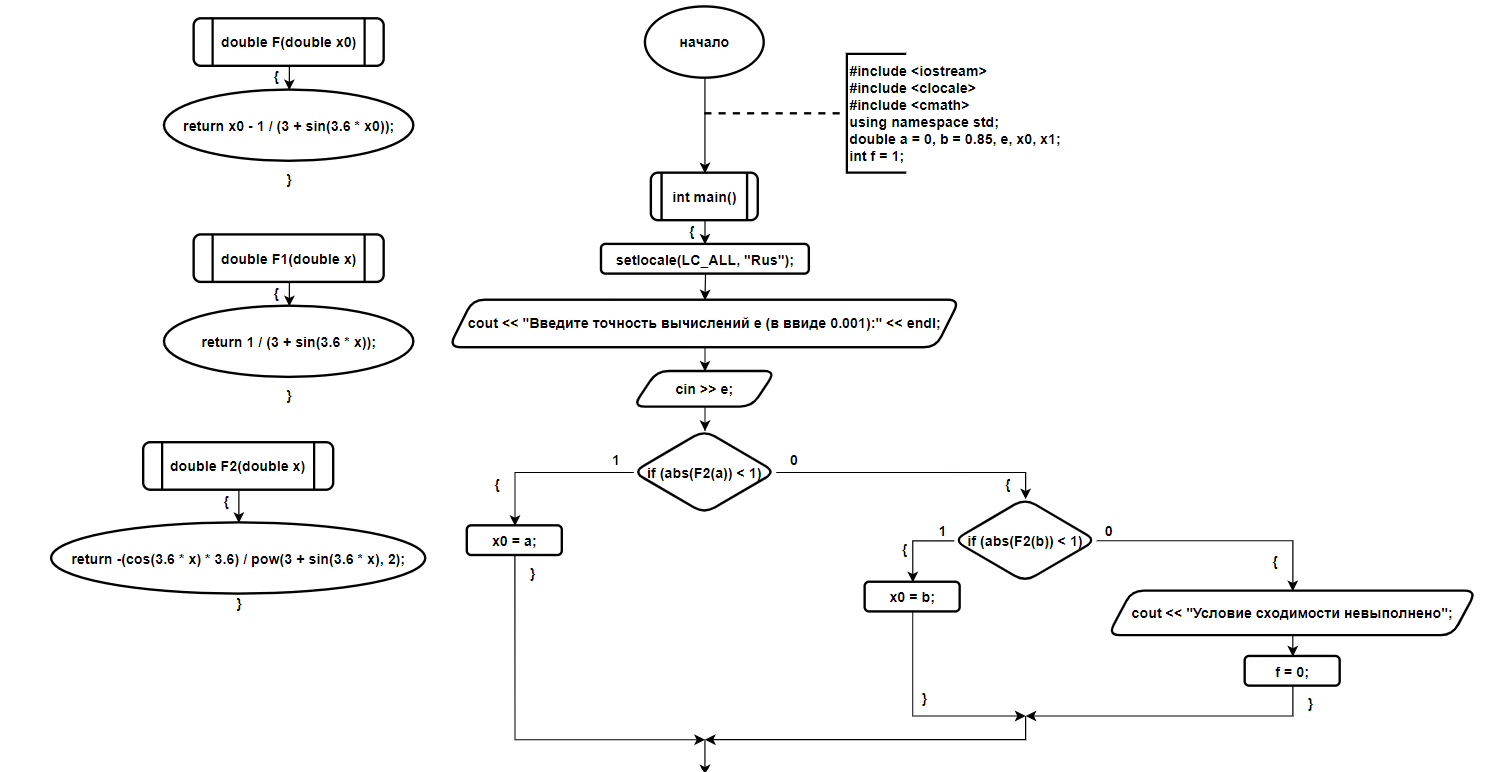
1. **Вывод формулы нахождения корня**
   1. Вычисляется следующее приближение к корню по формуле:

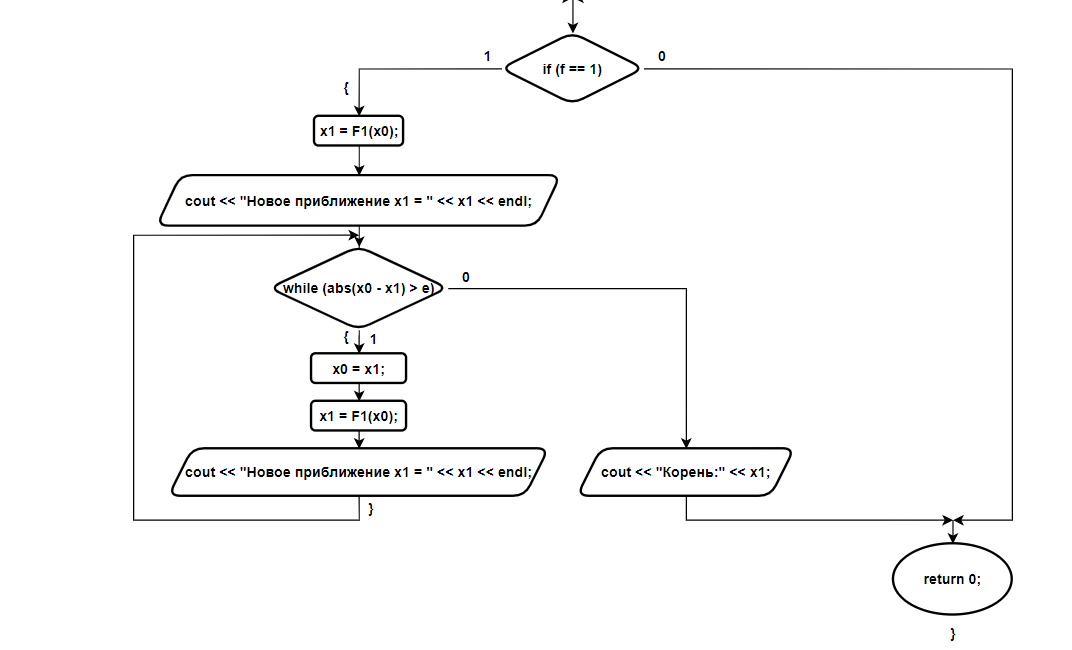
* 1. При получении нужной точности по формуле:  приближение к корню останавливается.

– является найденным корнем уравнения

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

1. **Блок-схема**





1. **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, "Rus");

double a = 0, b = 0.85, e, x0, x1;

int f = 1;

cout << "Введите точность вычислений e (в ввиде 0.001):" << endl;

cin >> e;

if (abs(F2(a)) < 1) х0 = a; // проверка условия сходимости и нахождение первого приближения к корню

else if (abs(F2(b)) < 1) x0 = b;

else

{

cout << "Условие сходимости невыполнено";

f = 0;

}

if (f == 1)

{

x1 = F1(x0); // следующее приближение к корню

cout << "Новое приближение х1 = " << x1 << endl;

while (abs(x0 - x1) > e) // приближение к корню до достижения нужной точности

{

x0 = x1;

x1 = F1(x0);

cout << "Новое приближение х1 = " << x1 << endl;

}

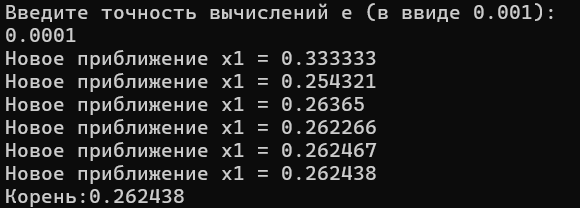
cout << "Корень:" << x1;

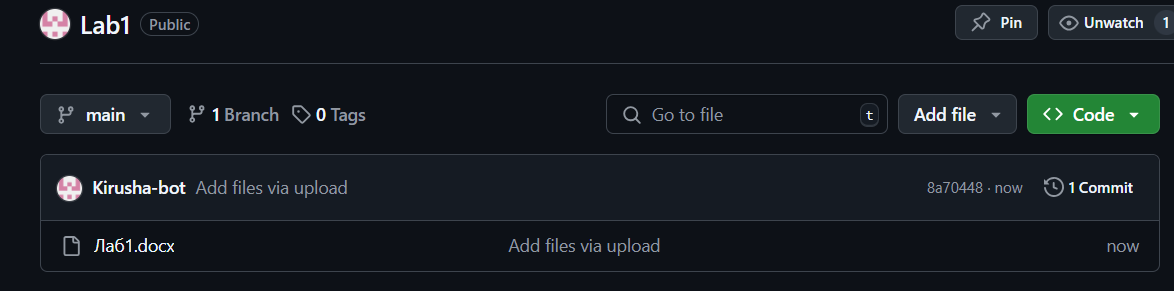
}

return 0;

}

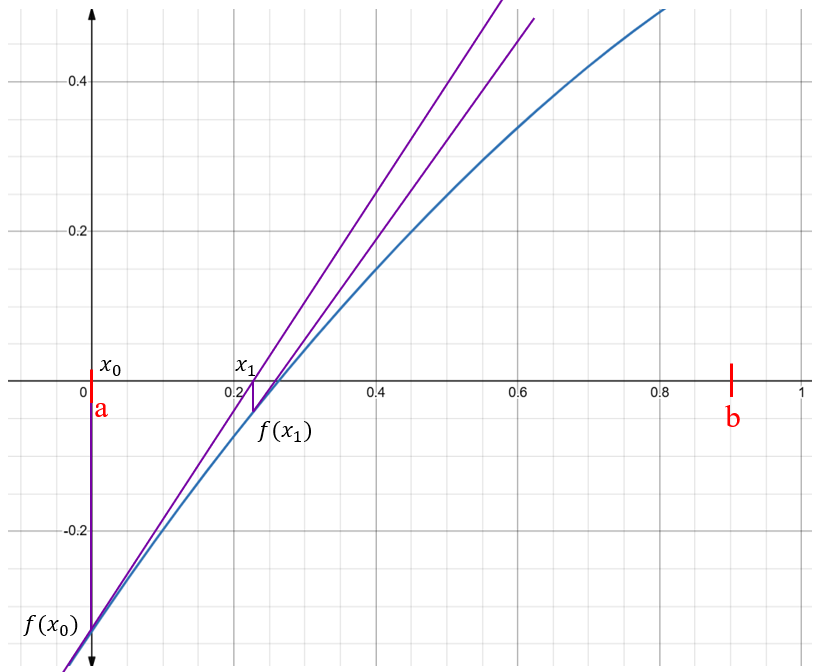
1. **Вывод:**





**Метод Ньютона**

1. **Геометрическая интерпретация метода**

****

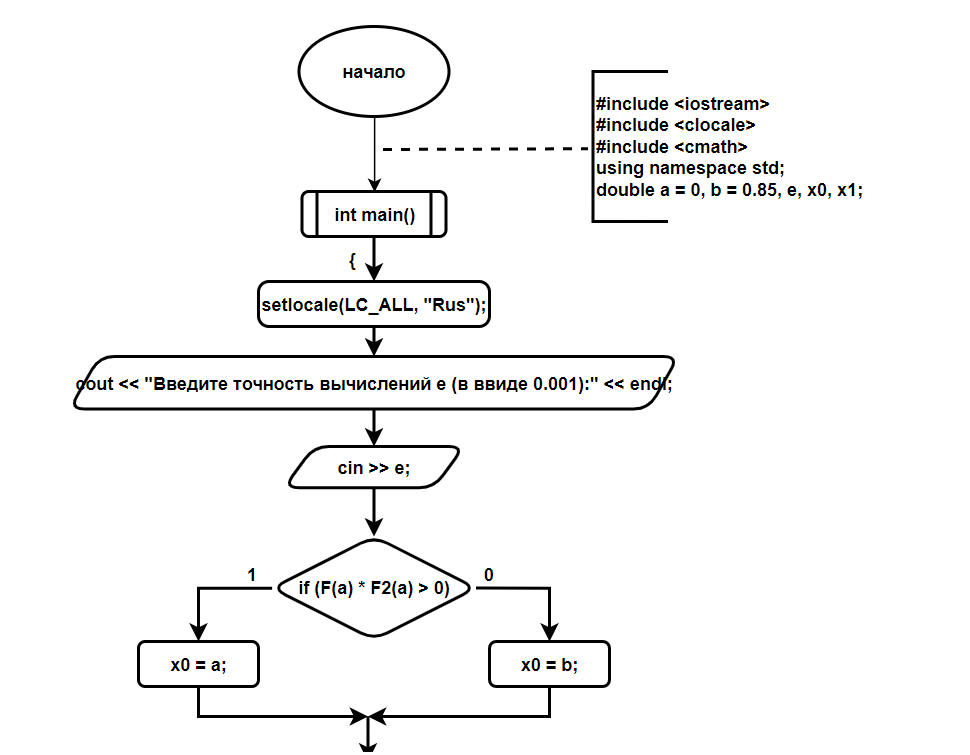
1. **Обоснование стороны подхода к функции**
   1. Найдём .
   2. Найдём c помощью формулы:

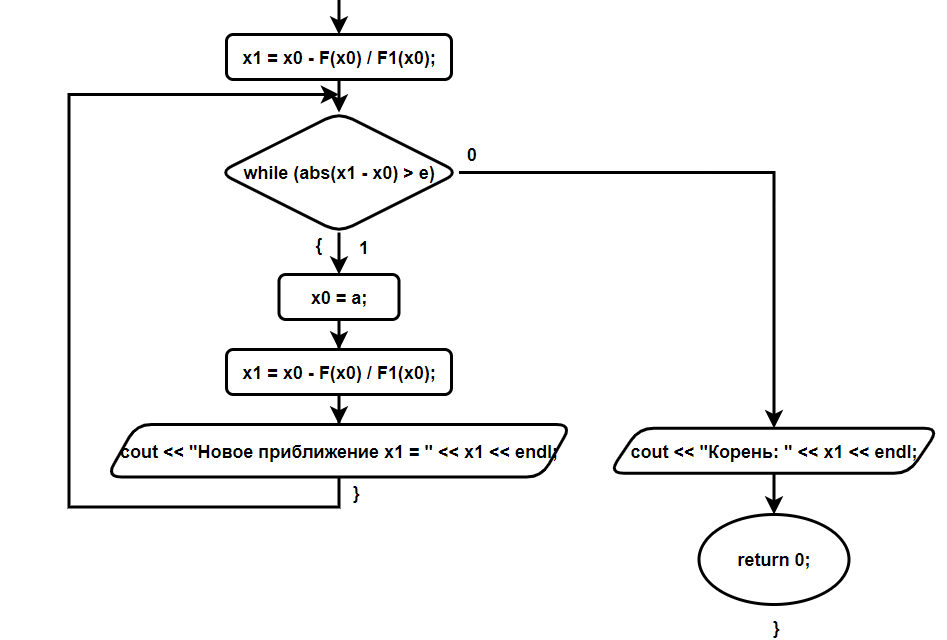
Подставим

1. **Вывод формулы нахождения корня**
   1. Угол касательной к функции определяется тангенсом угла наклона этой касательной к оси Ох.
   2. Уравнение касательной в виде
   3. Уравнение касательной в точке :
   4. Из уравнения 4.3. выразить :
   5. Подставить 4.4. в выражение 4.3.:

* 1. Пересечение с осью :
  2. – формула нахождения следующего приближённого корня
  3. – точность
  4. При достижении нужной точности оба корня будут считаться допустимыми для корня уравнения.

1. **Блок-схема**





1. **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 + (3.6 \* cos(3.6 \* x)) / pow(3 + sin(3.6 \* x), 2);

}

double F2(double x)

{

return -(pow(3.6, 2) \* sin(3.6 \* x) + (3 + sin(3.6 \* x)) \* 2 \* pow(cos(3.6 \* x), 2) \* pow(3.6, 2)) / pow(3 + sin(3.6 \* x), 4);

}

int main()

{

setlocale(LC\_ALL, "Rus");

double a = 0, b = 0.85, e, x0, x1;

cout << "Введите точность вычислений e (в ввиде 0.001):" << endl;

cin >> e;

if (F(a) \* F2(a) > 0) x0 = a; // определение значения x0

else x0 = b;

x1 = x0 - F(x0) / F1(x0); // первое приближение к корню

while (abs(x1 - x0) > e) // приближение к корню до достижения нужной точности

{

x0 = x1;

x1 = x0 - F(x0) / F1(x0);

cout << "Новое приближение х1 = " << x1 << endl;

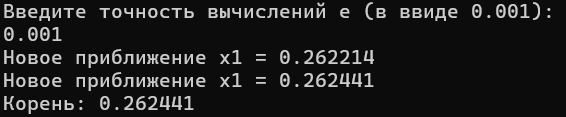
}

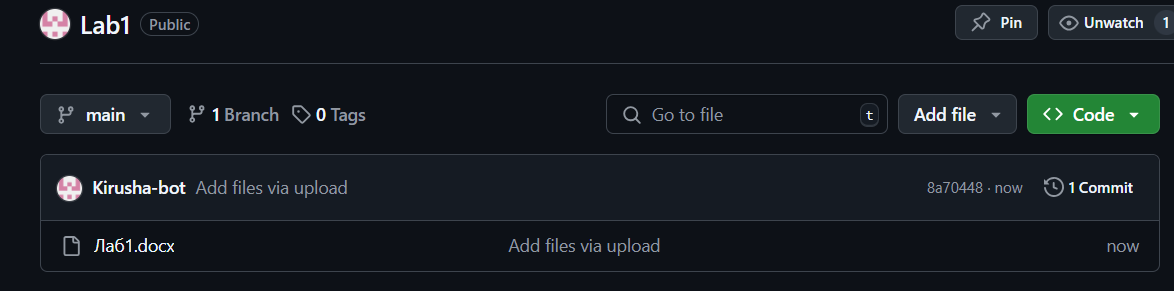
cout << "Корень: " << x1 << endl;

return 0;

}

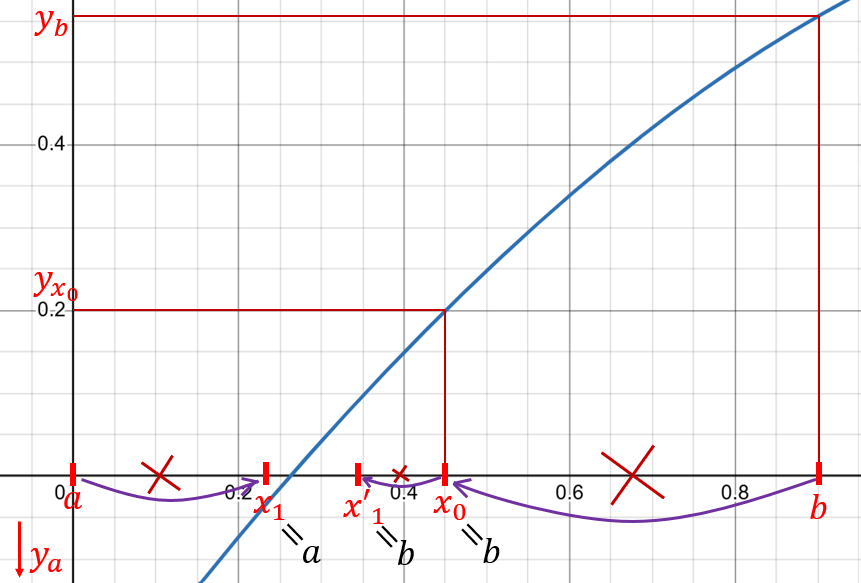
1. **Вывод**

****

****

**Метод половинного деления**

**2. Геометрическая интерпретация метода**



**3. Обоснование стороны подхода к функции**

Начальное приближение к корню считается по формуле . То есть делим отрезок [*a*; *b*] пополам.

1. **Вывод формулы нахождения корня**

4.1 Определяем какой из интервалов-половинок нужно отбросить по следующему условию:

4.2 Проверим *b*:

Если [*a*; *b*] => из вышестоящей проверки

4.3 Таким образом, получаем интервал, который нужно сохранить и который нужно отбросить

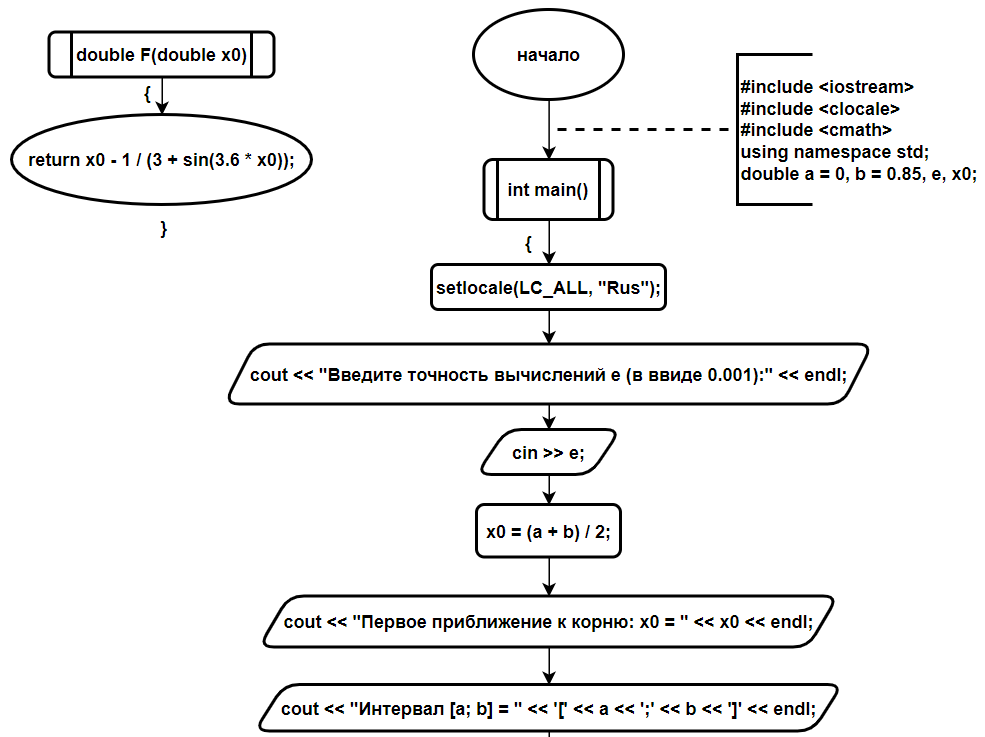
Сдвигаем границу интервала, который нужно отбросить. Получаем снова [*a*; *b*].

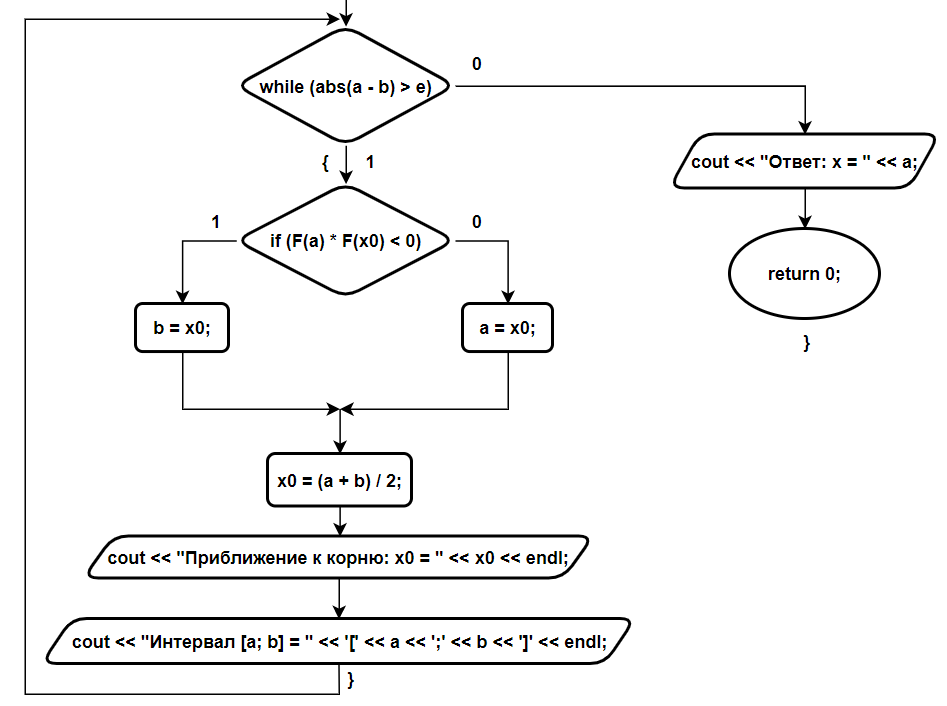
4.4 Делим новый [*a*; *b*] пополам, получая новое значение корня.

4.5 Операция повторяется до достижения нужной точности :

4.6 При достижении нужной точности обе границы интервала будут являться допустимыми корнями уравнения.

1. **Блок-схема**

****

****

1. **Код**

#include <iostream>

#include <clocale>

#include <cmath>

using namespace std;

double F(double x0)

{

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

}

int main()

{

setlocale(LC\_ALL, "Rus");

double a = 0, b = 0.85, e, x0;

cout << "Введите точность вычислений e (в виде 0.001):" << endl;

cin >> e;

x0 = (a + b) / 2;

cout << "Первое приближение к корню: x0 = " << x0 << endl;

cout << "Интервал [a; b] = " << '[' << a << ';' << b << ']' << endl;

while (abs(a - b) > e)

{

if (F(a) \* F(x0) < 0) b = x0;

else a = x0;

x0 = (a + b) / 2;

cout << "Приближение к корню: x0 = " << x0 << endl;

cout << "Интервал [a; b] = " << '[' << a << ';' << b << ']' << endl;

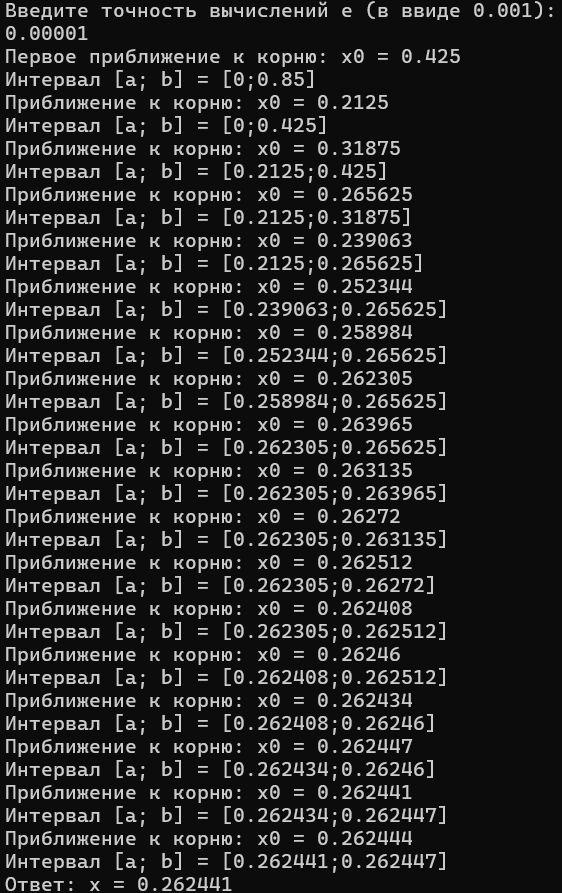
}

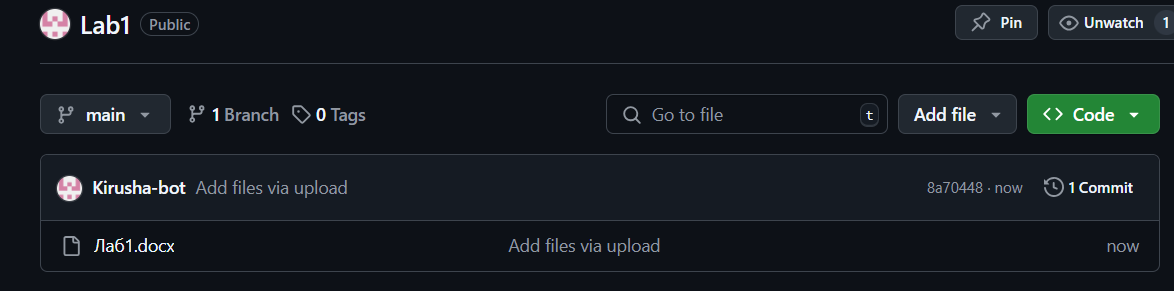
cout << "Ответ: х = " << a;

return 0;

}

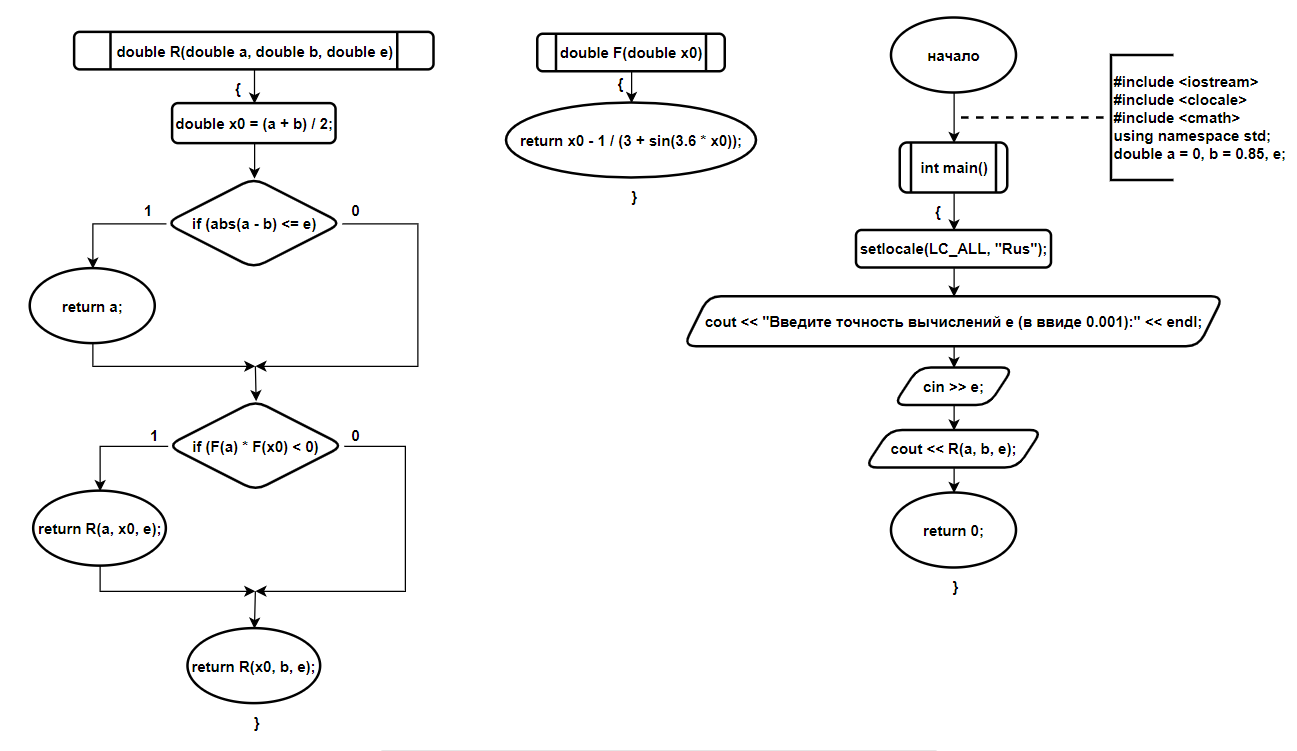
1. **Вывод**

****

****

**Метод половинного деления (рекурсивный способ)**

**5. Блок-схема**

****

**6. Код**

#include <iostream>

#include <clocale>

#include <cmath>

using namespace std;

double F(double x0)

{

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

}

double R(double a, double b, double e)

{

double x0 = (a + b) / 2;

if (abs(a - b) <= e) return a;

if (F(a) \* F(x0) < 0) return R(a, x0, e);

return R(x0, b, e);

}

int main()

{

setlocale(LC\_ALL, "Rus");

double a = 0, b = 0.85, e;

cout << "Введите точность вычислений e (в ввиде 0.001):" << endl;

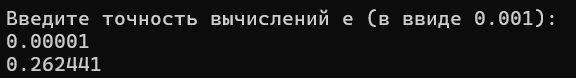
cin >> e;

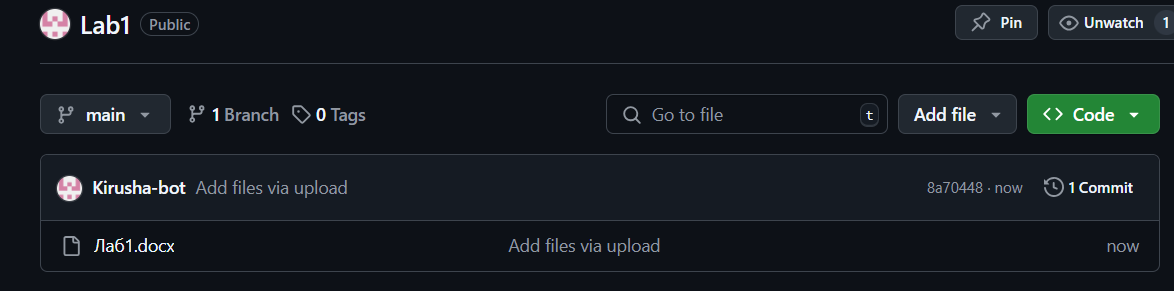
cout << R(a, b, e);

return 0;

}

**7. Вывод**



****