**«Пермский национальный исследовательский политехнический университет»**

Электротехнический факультет

Кафедра «Информационные технологии и автоматизированные системы»

**Лабораторная работа №2.**

**«Решение нелинейных уравнений численными методами»**

Вариант 14

Выполнил студент гр. РИС-24-1б

Погадаев Данил Владимирович

Проверил:

Доц. каф. ИТАС

Ольга Андреевна Полякова

(оценка) (подпись)

(дата)

г. Пермь, 2024

**Отчёт**

**Метод Ньютона**

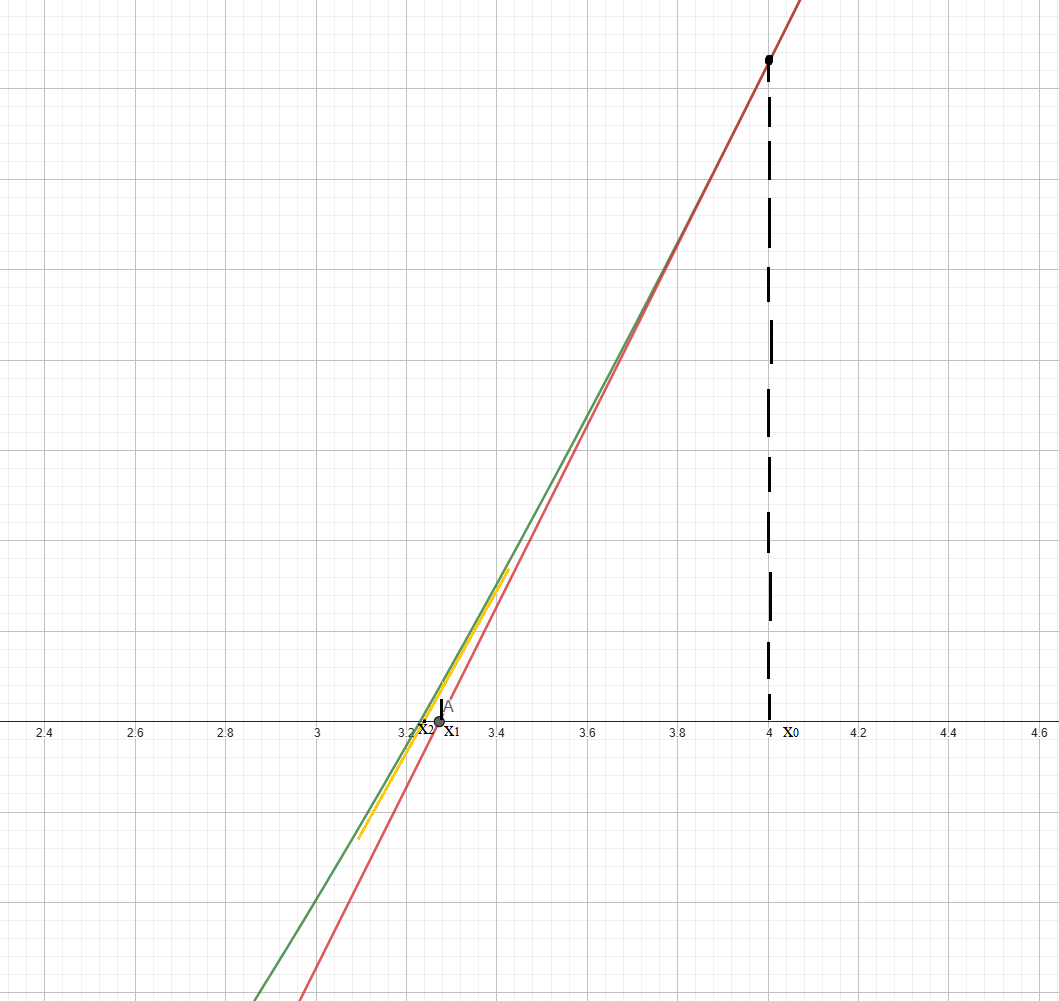
**1. Постановка задачи**

Рассмотрим уравнение:  
3x – 4lnx - 5 = 0  
Необходимо найти корень данного уравнения методом Ньютона на заданном отрезке: [2;4]

Точное значение: 3,2300

**2. Геометрическая интерпретация метода**

* На каждом шаге итерации проводится касательная к графику функции f(x) в точке xn​.
* Точка пересечения этой касательной с осью Ox становится следующим приближением.

****

**3. Обоснование стороны подхода**

Для определения стороны подхода к графику нужно воспользоваться неравенствами:

Если f(a) \* f ‘(a)> 0, то x0 = a

Если f(b) \* f ‘(b)> 0, то x0 = b

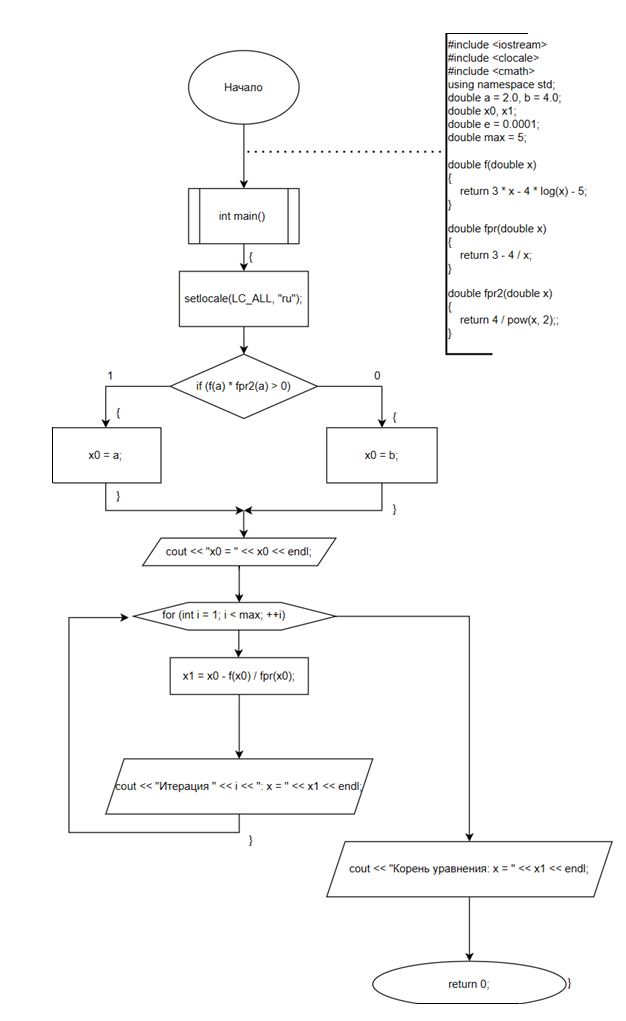
**4. Вывод формулы нахождения корня**

1. Угол касательной к функции f(x) определеятся tgx угла наклона этой касательной через

f′(x0​)=tgα=k.

1. Запишем уравнение прямой , так как касательная прямая в виде - y=k⋅x+b
2. Запишем уравнение касательной в точке
3. Из уравнение выразим b
4. Подставим выражение (4) в пункт (3)
5. Вынесем производную как общий множитель

**5. Блок-схема**

****

**6. Код**

#include <iostream>

#include <clocale>

#include <cmath>

using namespace std;

double f(double x){

return 3 \* x - 4 \* log(x) - 5;

}

double fpr(double x){

return 3 - 4 / x;

}

int main() {

setlocale(LC\_ALL, "ru");

double a = 2.0, b = 4.0;

double x0, x1;

double e = 0.0001;

double max = 5;

double fpr2(double x)

{

return 4 / pow(x, 2);;

}

if (f(a) \* fpr2(a) > 0) {

x0 = a;

}

elsе {

x0 = b;

}

cout << "x0 = " << x0 << endl;

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

x1 = x0 - f(x0) / fpr(x0);

cout << "Итерация " << i << ": x = " << x1 << endl;

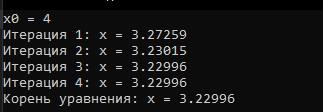
x0 = x1;

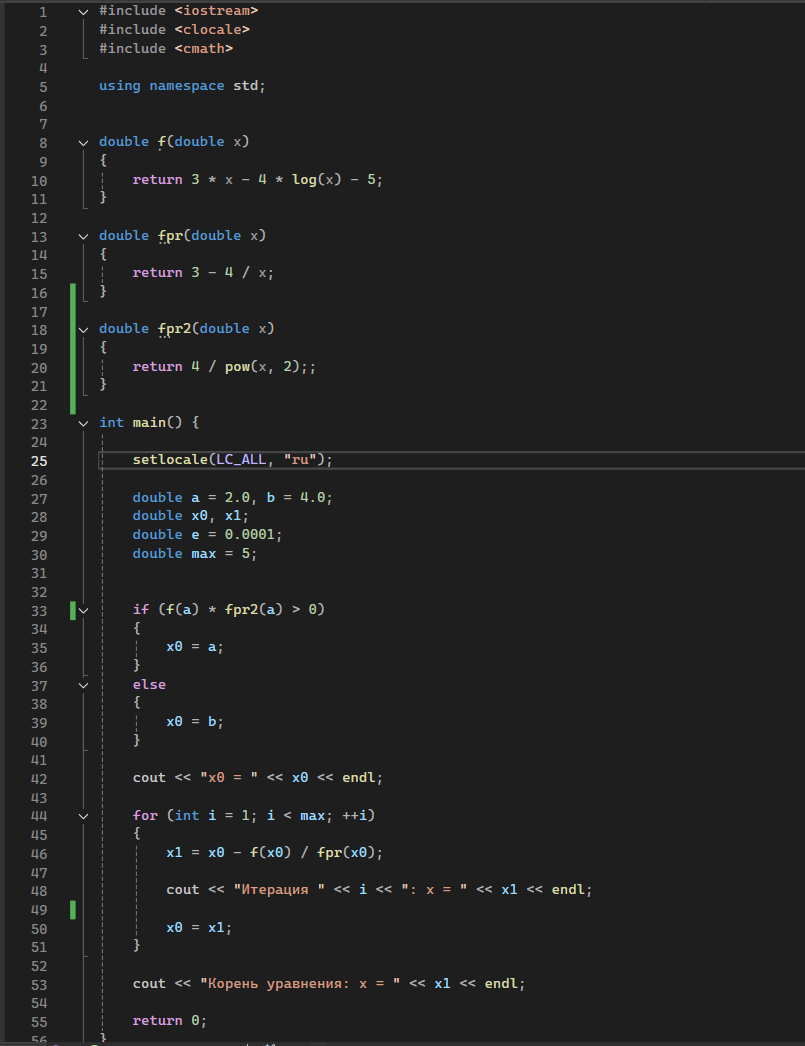
}

cout <<«Корень уравнения: x = " <<x1 <<endl;

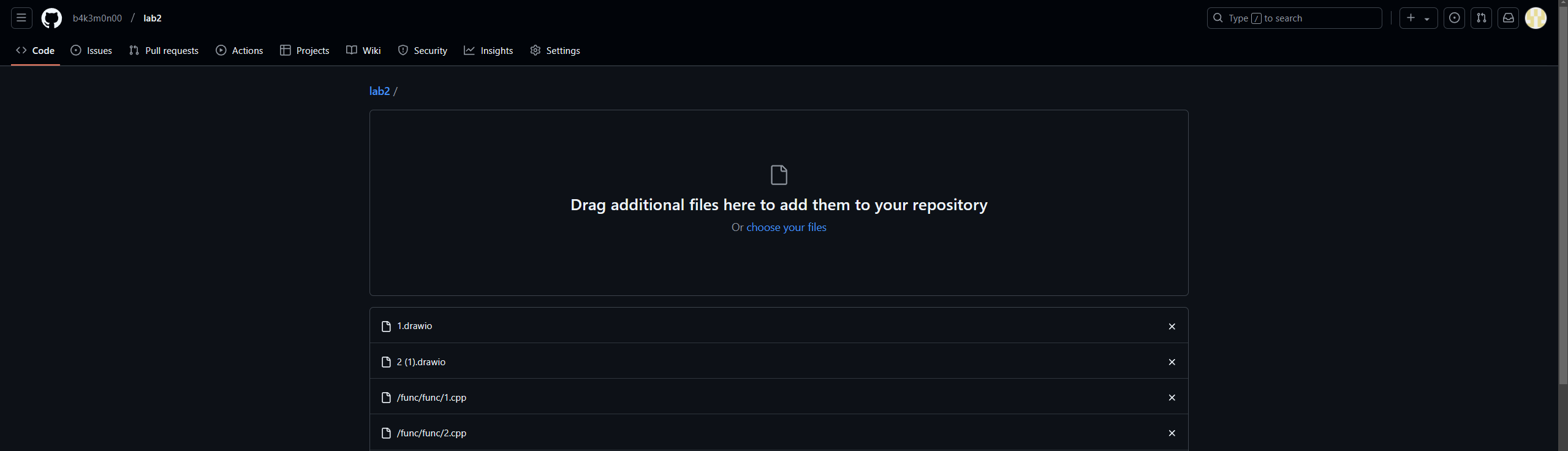
return 0;

}





**7. Скриншоты с git hub**



**Метод итераций**

**1. Постановка задачи**

Рассмотрим уравнение:  
3x – 4lnx - 5 = 0  
Необходимо найти корень данного уравнения методом итераций на заданном отрезке: [2;4]

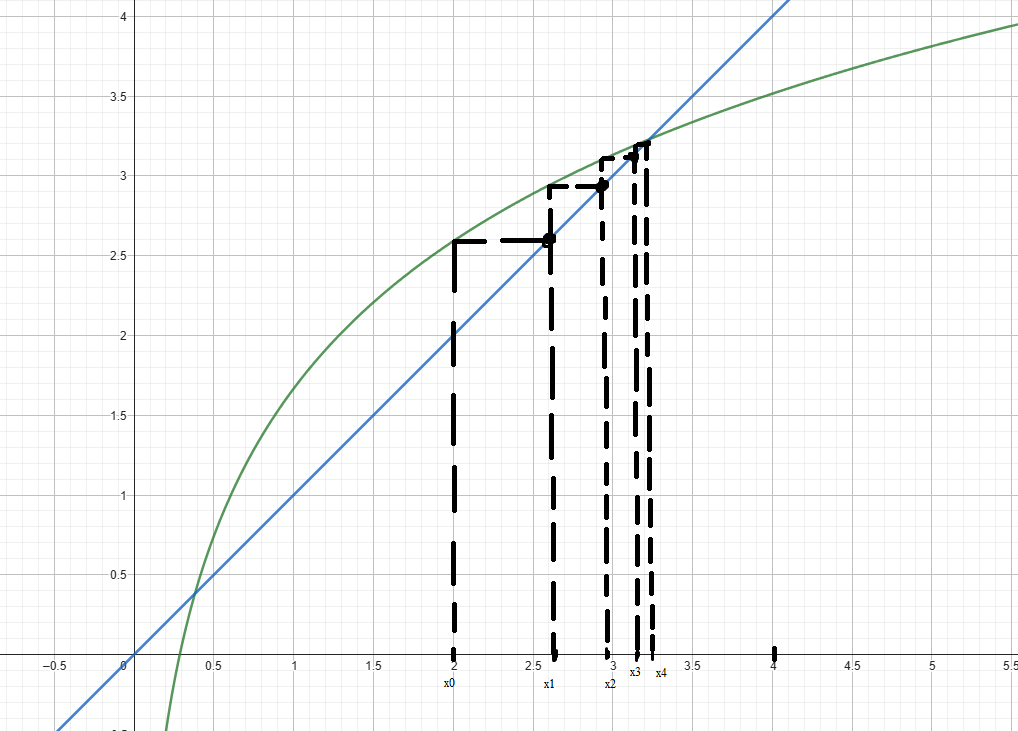
Точное значение: 3,2300

**2. Геометрическая интерпретация метода**

Метод простых итераций заключается в замене исходного уравнения f(x)=0 на итерационное уравнение вида: x=φ(x)

Геометрически метод итераций можно представить следующим образом:

* Строится график функции y=φ(x) и прямая y=x.
* Берётся один из концов отрезка и проецируется на график x=φ(x)
* Построим горизонтальную линию до пересечения с графиков y=x
* Опустим вертикальную линию на ось Ox чтобы получить значение x1
* Следуя этому алгоритму, мы будем получать всё более близкие к корню значения

****

**3. Обоснование стороны подхода**

Для выбора начального приближения на отрезке [a; b], нужно найти производную от φ(x) и воспользоваться неравенствами:

Если φ‘(a) <1, то x0 = a

Если φ‘(b) <1, то x0 = b

**4. Вывод формулы нахождения корня**

Для применения метода итераций исходное уравнение

преобразуем к виду:

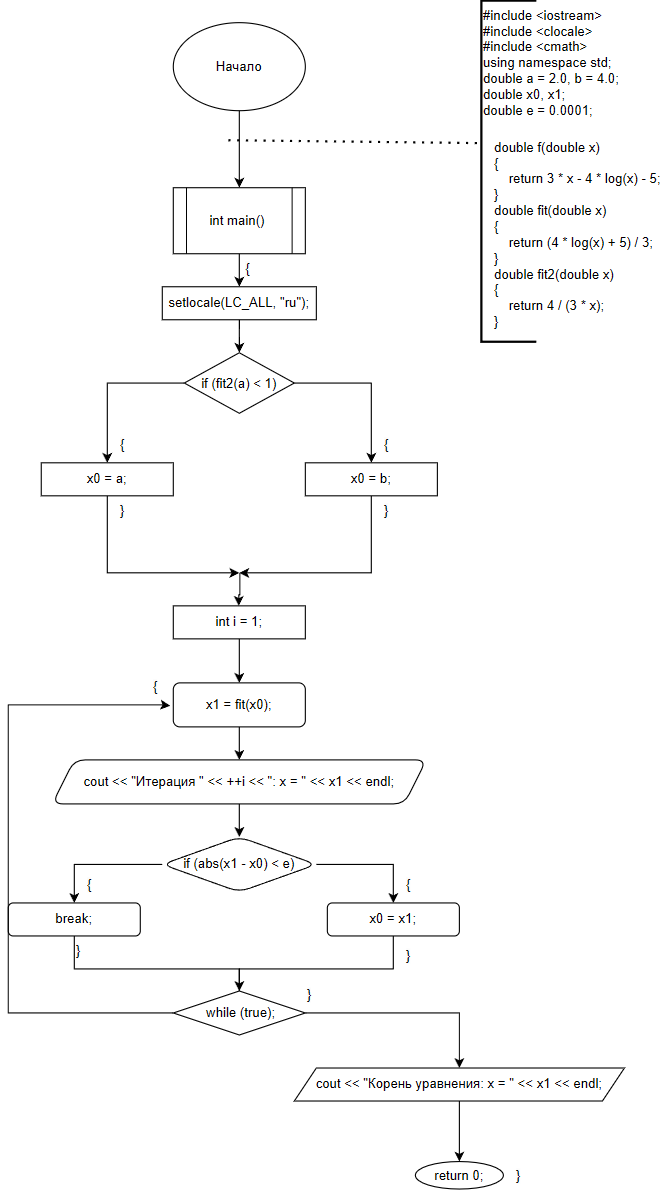
Таким образом, на каждой итерации вычисляется новое приближение по следующей формуле:

*φ(xn)=*

Итерационный процесс продолжается до тех пор, пока разность между текущим и предыдущим значением не станет меньше заданной точности:

∣xn+1​−xn​∣ <ε  
где xn​ — текущее приближение, xn+1​ — следующее приближение.

**5. Блок-схема**



**6. Код**

#include <iostream>

#include <clocale>

#include <cmath>

using namespace std;

double f(double x){

return 3 \* x - 4 \* log(x) - 5;

}

double fit(double x){

return (4 \* log(x) + 5) / 3;

}

double fit2(double x){

return 4 / (3 \* x);

}

int main(){

setlocale(LC\_ALL, "ru");

double a = 2.0, b = 4.0;

double e = 0.0001;

double x0, x1;

if (fit2(a) < 1) {

x0 = a;

}

else{

x0 = b;

}

int i = 1;

do{

x1 = fit(x0);

cout << "Итерация " << ++i << ": x = " << x1 << endl;

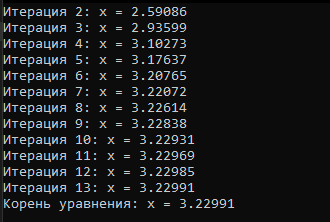
if (abs(x1 - x0) < e)

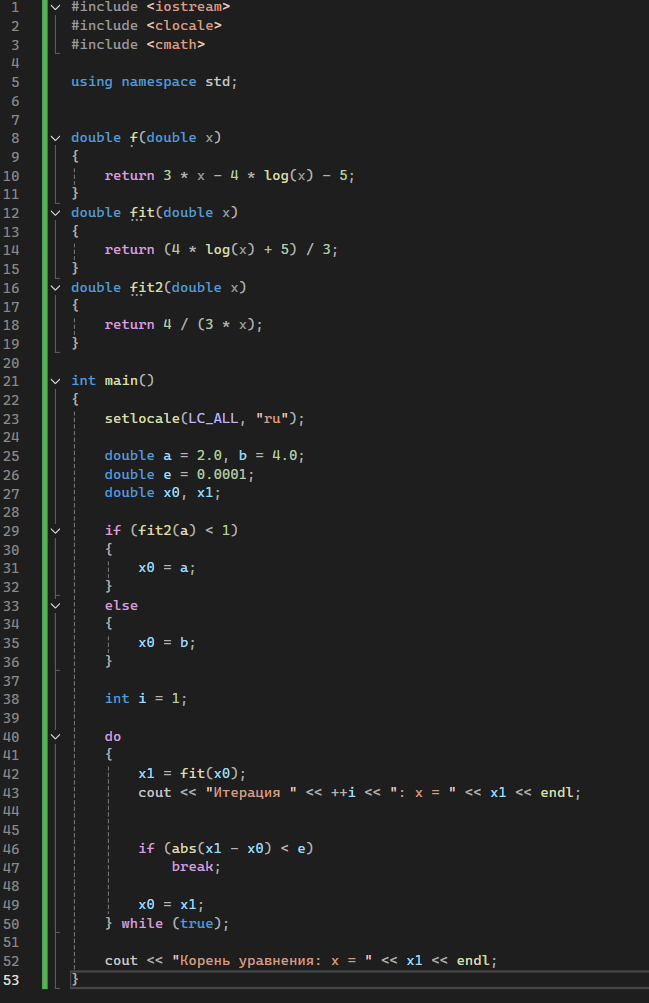
break;

x0 = x1;

} while (true);

cout << "Корень уравнения: x = " << x1 << endl;

} 



**7. Скриншоты с git hub**

