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 №2.**

**«Решение нелинейных уравнений. Численные методы»**

Выполнил студент гр. РИС-24-3б

Гладков Ярослав Алексеевич

Проверил:

Доцент. каф. ИТАС

Ольга Андреевна Полякова

(оценка) (подпись)

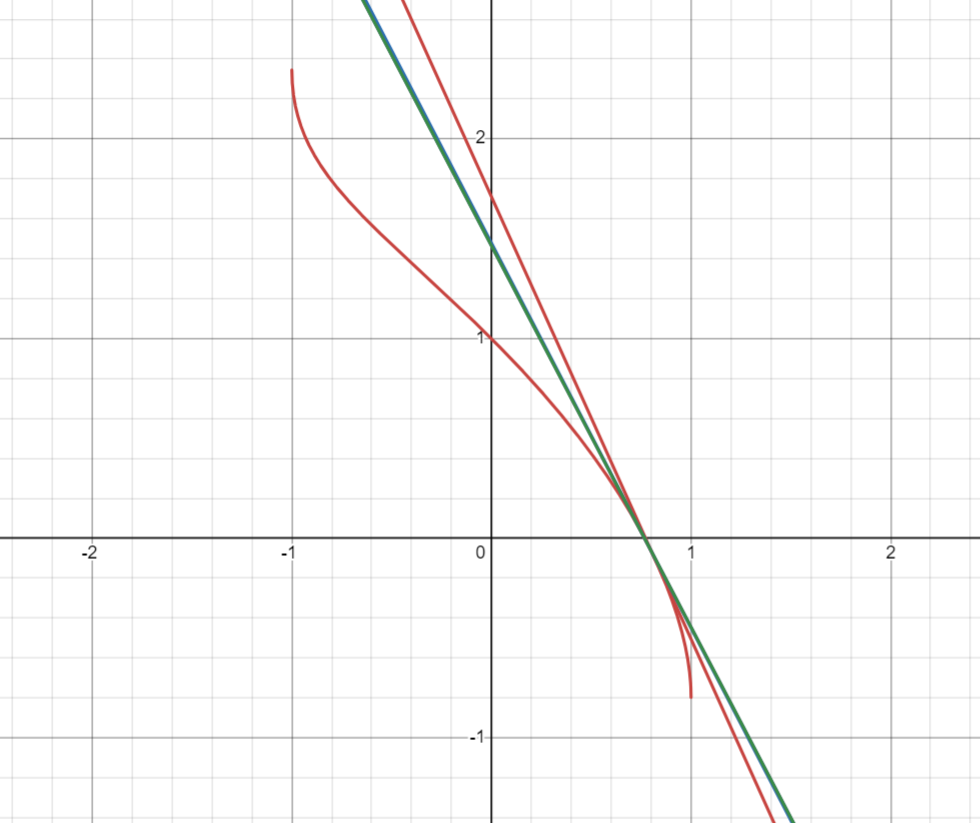
(дата)

г. Пермь, 2024

Написать геометрическую интерпретацию, ф также написать программы, реализующие метод Ньютона, метод дихотомии (половинного деления), метод итераций, решающие уравнение при точности = .

**Метод Ньютона**

Геометрическая интерпретация:



Шаг 1:

Найдем корень уравнения:

– значение производной функции *f* (*x*) в точке

Шаг 2:

Теперь мы имеем данные, необходимые, чтобы вычислить корень уравнения по методу ньютона:

Шаг 3:

Теперь используя приведенную выше формулу и зная , вычислим значение . Если n =1, то формула принимает вид:

Шаг 4:

Подставляем

Шаг 5:

Упрощая, получаем значение для :

= 0.8358017711949657

Шаг 6:

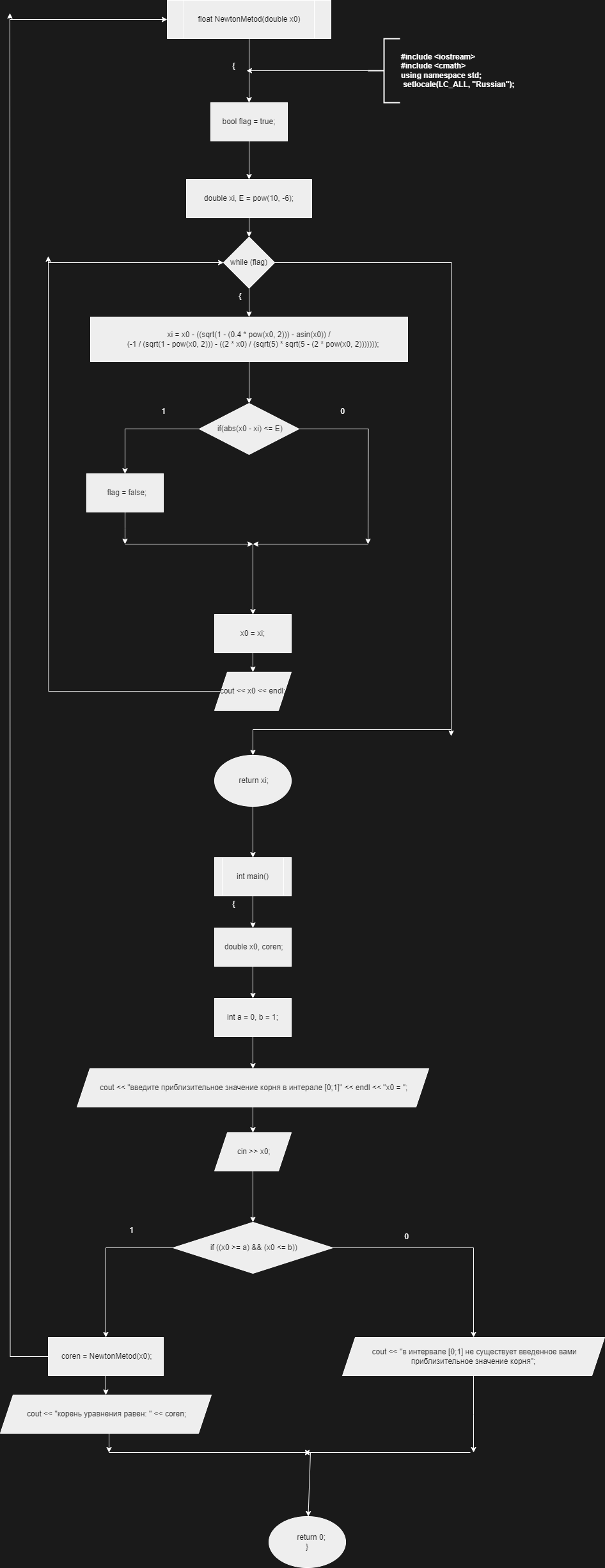
Повторяем, наши действия пока критерий остановки вычисления не сработает, то есть

Шаг 7:

Итак мы получили приблизительный корень уравнения равный

X = 0.7671622423265332

**Блок схема**:



**Код программы:**

#include <iostream>

#include <cmath>

using namespace std;

float NewtonMetod(double x0)

{

bool flag = true;

double xi, E = pow(10, -6);

while (flag)

{

xi = x0 - ((sqrt(1 - (0.4 \* pow(x0, 2))) - asin(x0)) /

(-1 / (sqrt(1 - pow(x0, 2))) - ((2 \* x0) / (sqrt(5) \* sqrt(5 - (2 \* pow(x0, 2)))))));

if(abs(x0 - xi) <= E)

{

flag = false;

}

x0 = xi;

cout << x0 << endl;

}

return xi;

}

int main()

{

setlocale(LC\_ALL, "Russian");

double x0, coren;

int a = 0, b = 1;

cout << "введите приблизительное значение корня в интерале [0;1]" << endl << "x0 = ";

cin >> x0;

if ((x0 >= a) && (x0 <= b))

{

coren = NewtonMetod(x0);

cout << "корень уравнения равен: " << coren;

}

else

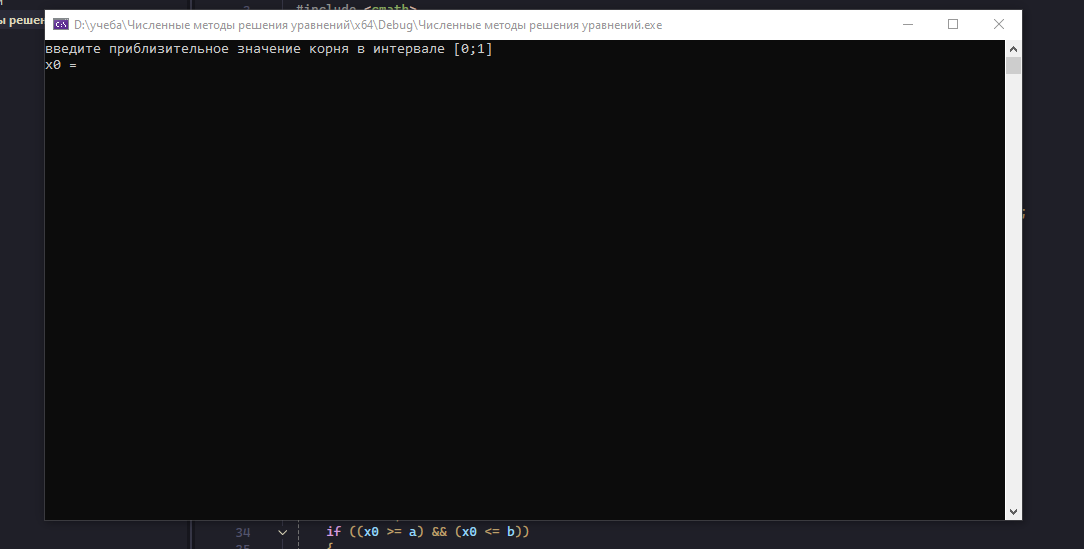
{

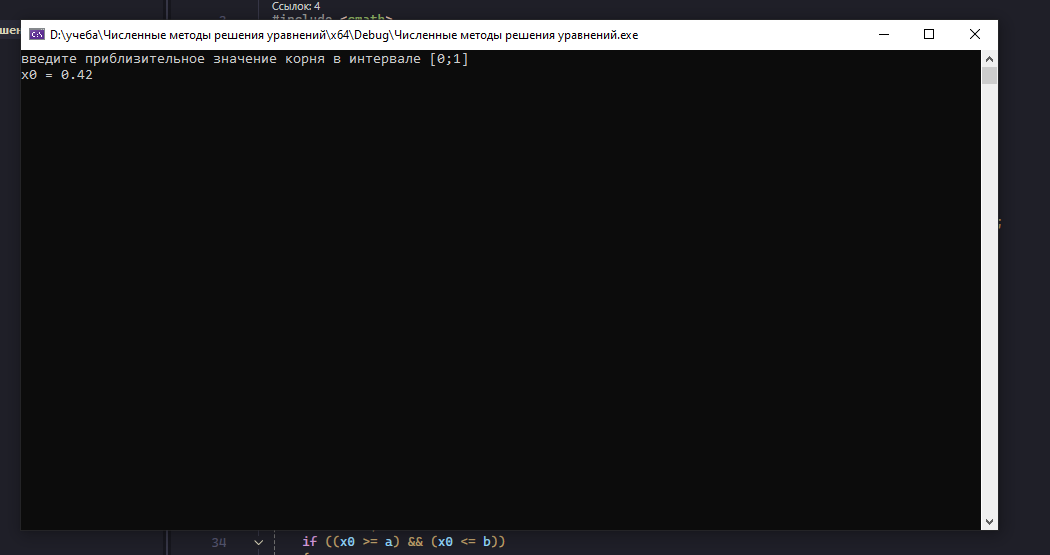
cout << "в интервале [0;1] не существует введенное вами приблизительное значение корня";

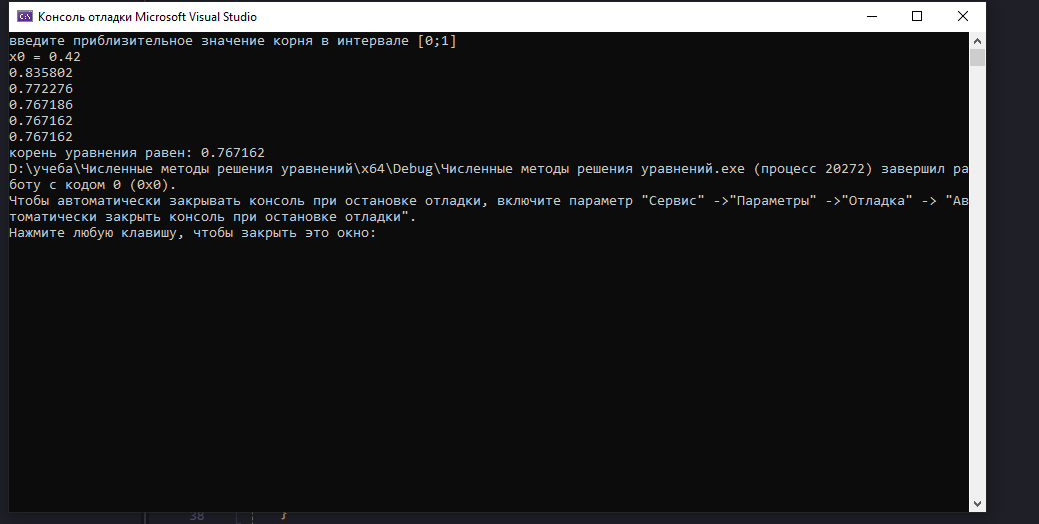
}

return 0;

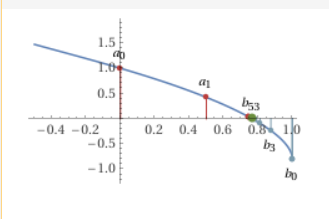
}

****

****

****

**Метод дихотомии**



Решение:

Поскольку F(0.75)\*F(1)<0 (т.е. значения функции на его концах имеют противоположные знаки), то корень лежит в пределах [0.75;1].

Повторяем, наши действия пока критерий остановки вычисления не сработает, то есть

Таким образом, в качестве корня можно принять:

**Блок схема и код программы:**

**#include <iostream>**

**#include <cmath>**

**using namespace std;**

**float f(double x0)**

**{**

**return sqrt(1- 0.4\*pow(x0,2))-asin(x0);**

**}**

**double MetodDichotomy(double a, double b)**

**{**

**double x0, xi = 0, E = pow(10,-6);**

**bool flag = true;**

**while (flag)**

**{**

**x0 = (a + b) / 2;**

**if (abs(x0 - xi) <= E)**

**{**

**flag = false;**

**}**

**if (f(x0)\*f(a)<0)**

**{**

**b = x0;**

**xi = x0;**

**}**

**else**

**{**

**a = x0;**

**xi = x0;**

**}**

**cout<< xi << endl;**

**}**

**return x0;**

**}**

**int main()**

**{**

**setlocale(LC\_ALL, "rus");**

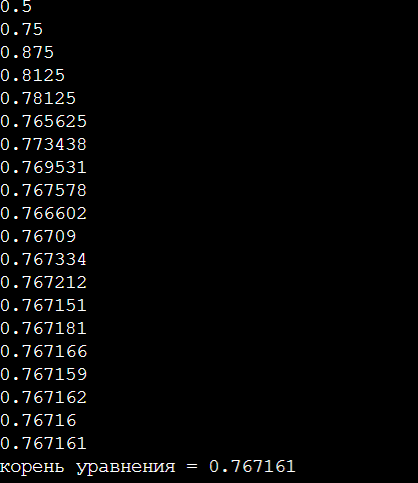
**double coren, a = 0, b = 1;**

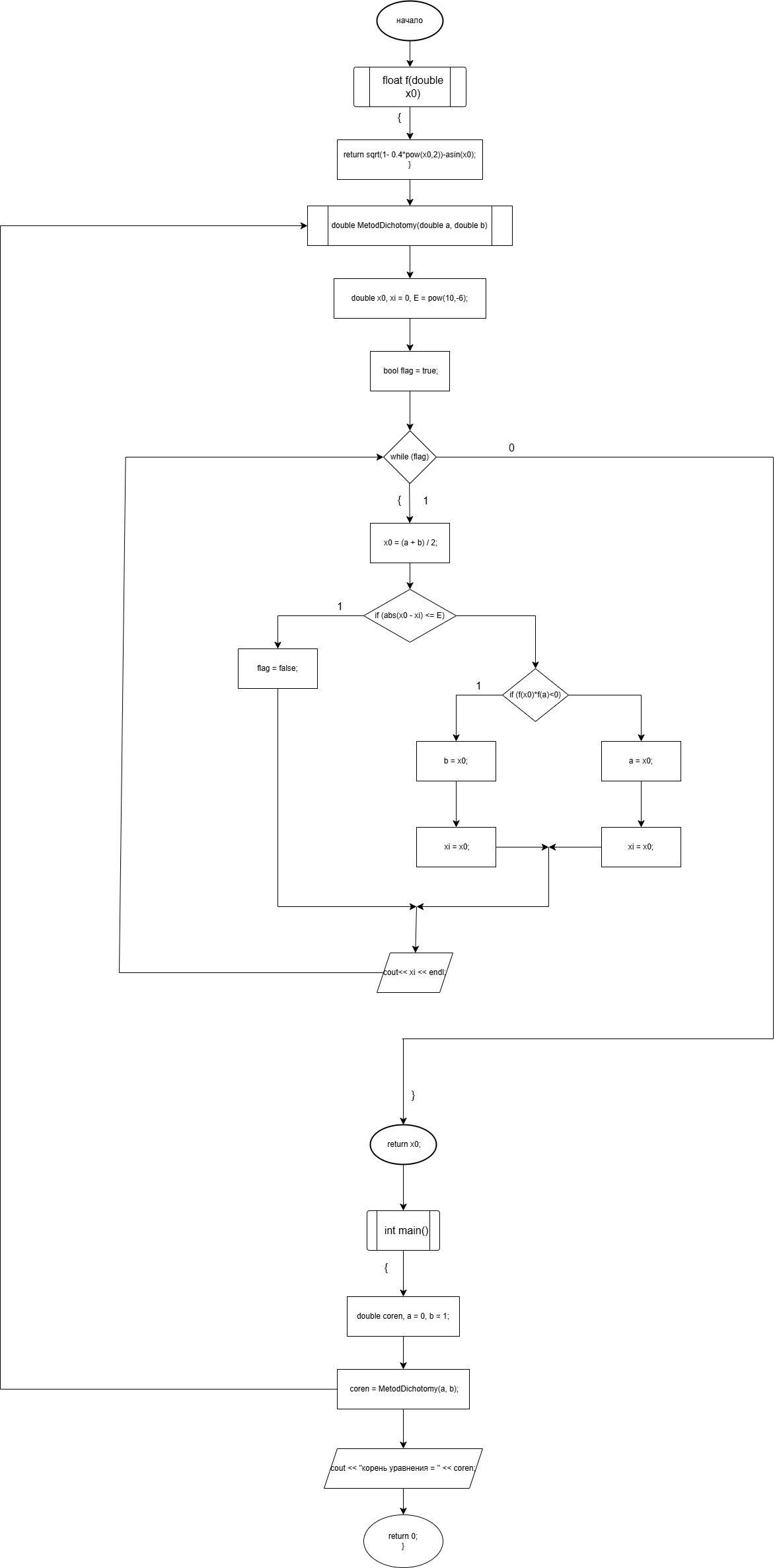
**coren = MetodDichotomy(a, b);**

**cout << "корень уравнения = " << coren;**

**return 0;**

**}:**

****



**Метод итераций**

Решение:

Шаг 1:

Выбираем начальное приближение

Шаг 2:

Проверяем сходимость по формуле

Шаг 3:

Применяем формулу итераций до тех пор, пока не будет достигнута желаемая точность.

Шаг 3:

x

Блок схема и программа:

#include <iostream>

#include <cmath>

using namespace std;

double f(double x) {

return sqrt(1 - 0.4 \* x \* x) - asin(x);

}

double fprime(double x) {

return (1 / 2) \* pow((1 - 0.4 \* pow(x, 2)), (-1 / 2)) \* (-0.8 \* x) - 1 / (sqrt(1 - pow(x, 2)));

}

int main() {

setlocale(LC\_ALL, "rus");

double x0, x1, pogreshnost;

bool flag = true;

cout << "Введите начальное приближение: ";

cin >> x0;

cout << "Введите погрешность: ";

cin >> pogreshnost;

int iterations = 0;

do {

// Вычисляем следующее приближение

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

iterations++;

// Проверяем условие сходимости

double proizvodnaia = fprime(x0);

if (abs(proizvodnaia) <= 1) {

cout << "Внимание! Метод итераций может не сходиться для данного начального приближения." << endl;

}

// Проверяем достижение погрешности

if (abs(x1 - x0) < pogreshnost) {

flag = false;

}

x0 = x1;

cout << "итерация " << iterations << ": " << x1 << endl;

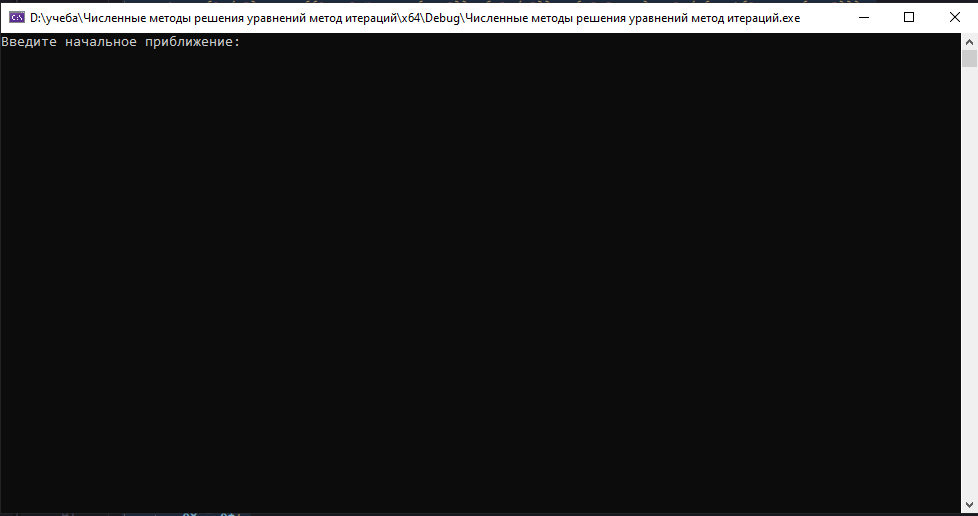
} while (flag);

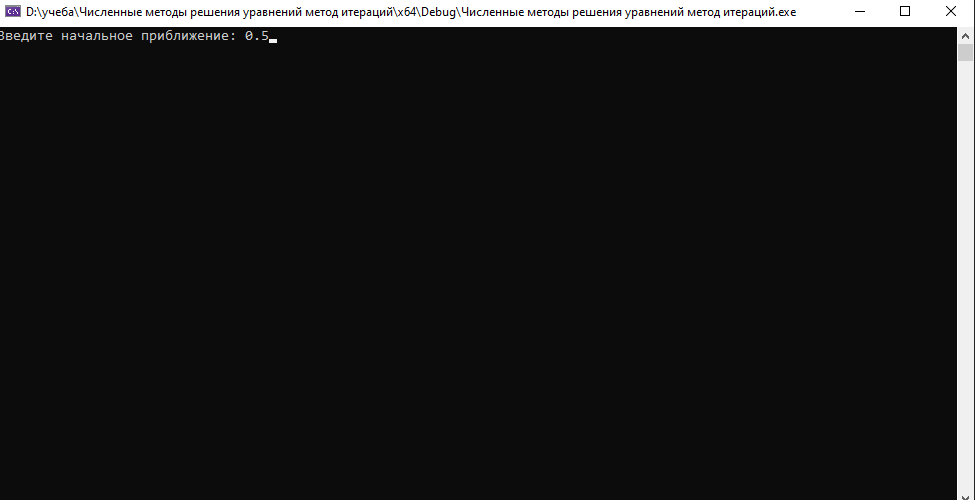
cout << "Корень уравнения: " << x1 << endl;

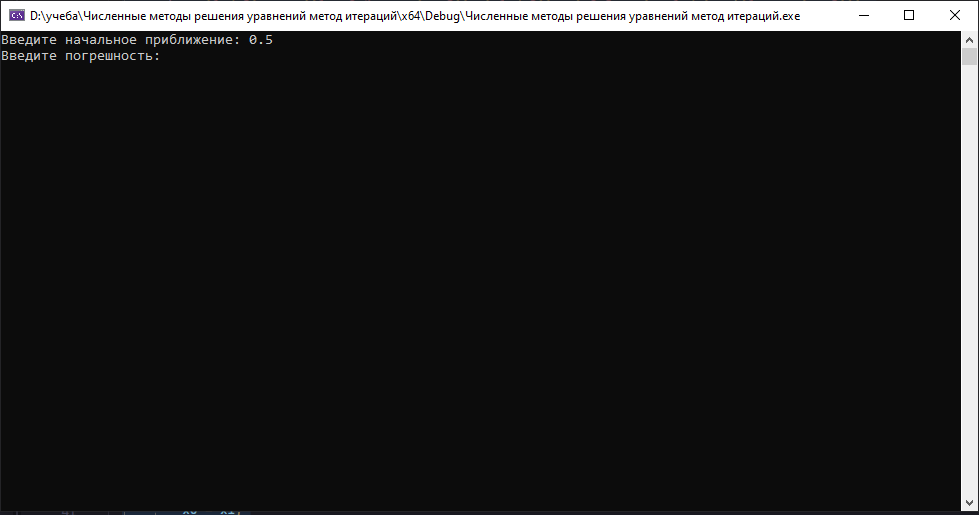
cout << "Количество итераций: " << iterations << endl;

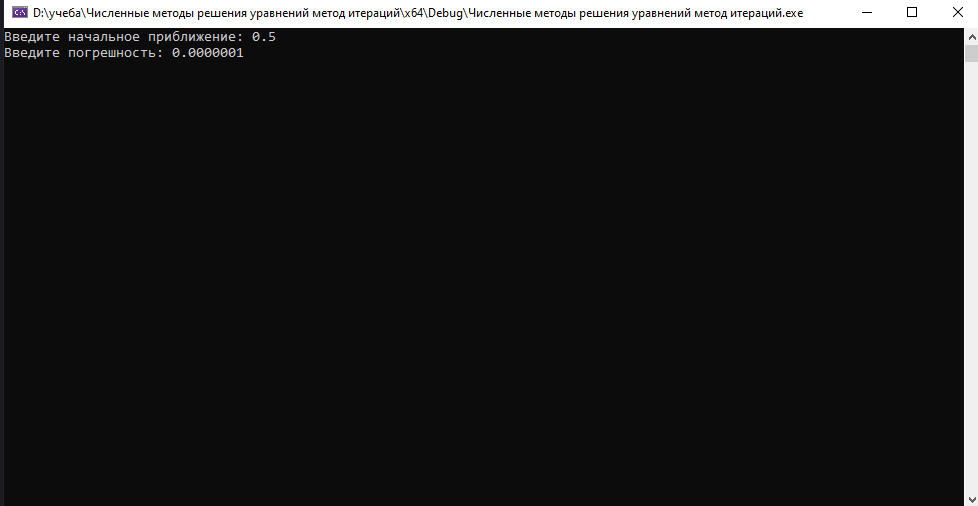
return 0;

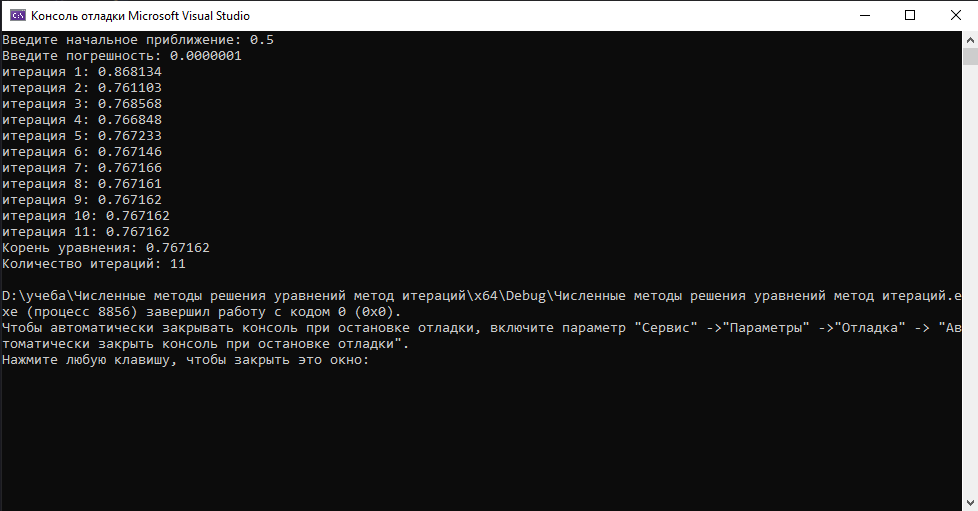
}

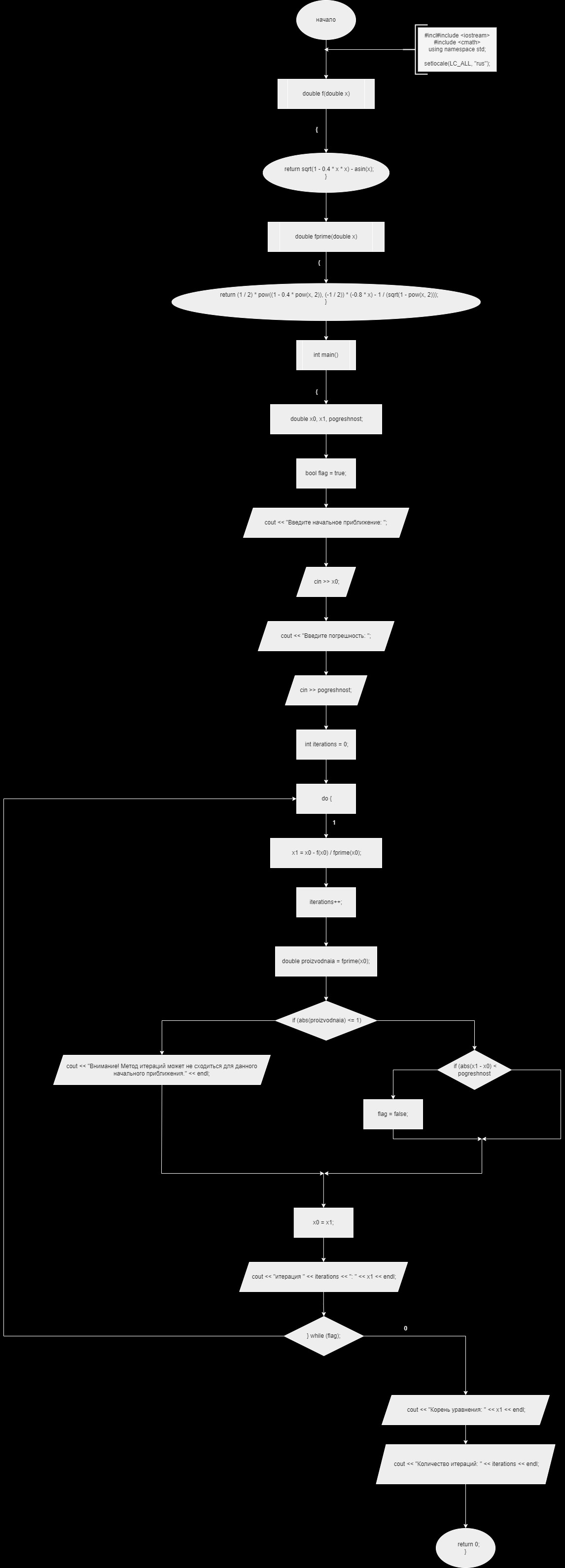












Github ссылка:

https://github.com/YaroslavRi/Chislen-metod-.git