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»

**Лабораторная работа**

**по дисциплине «Основы алгоритмизации и программирования»**

По теме: «Решение нелинейных алгоритмов»

Выполнил студент гр. РИС-24-3б

Монов Андрей Александрович

(Фамилия, Имя, Отчество)

(подпись)

Проверил:

доцент

кафедры ИТАС

Полякова О.А.

(оценка) (подпись)

(дата)

Пермь 2024

**Постановка задачи**

Написать 3 программы, реализующие 3 различных метода решения нелинейных уравнения: метод Ньютона, метод половинного деления, метод итераций.

**Анализ задачи**

Ход выполнения:

1. Ввод интервала и переменных
2. Объявление точности вычислений
3. Вычисление приближений корня уравнения на интервале [0,1]:

**Метод половинного деления**

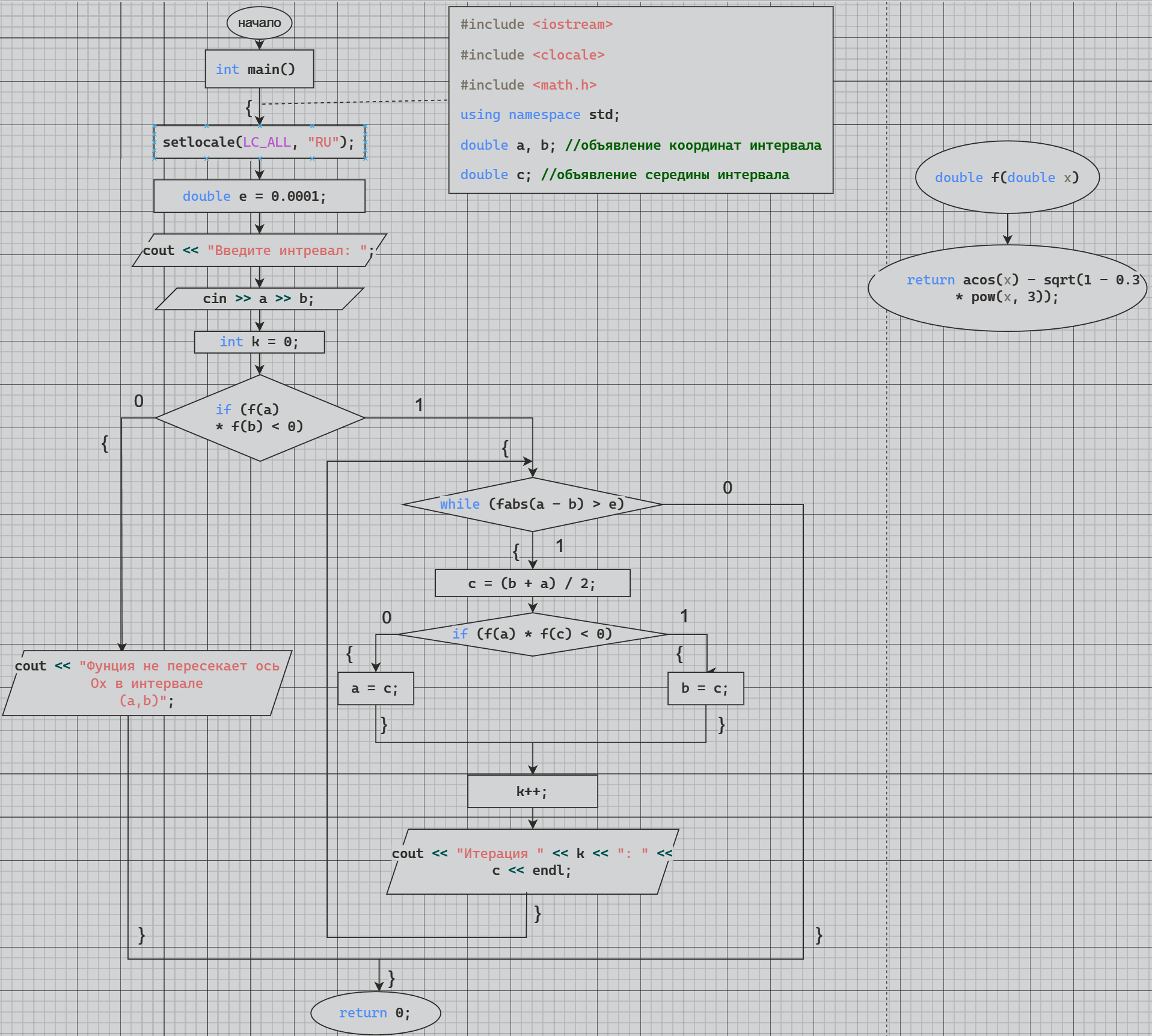
Описание переменных:

1.Переменная вещественного типа е – точность вычислений

2.Переменные вещественного типа а, b – интервал в котором находится корень

3.Переменная вещественного типа с – середина интервала

4.Переменная целочисленного типа k – счётчик итераций  
Блок схема:



Код:

#include <iostream>

#include <clocale>

#include <math.h>

using namespace std;

double a, b; //объявление координат интервала

double с; //объявление середины интервала

double f(double x)

{

return acos(x) - sqrt(1 - 0.3 \* pow(x, 3)); // уравнение

}

int main() {

setlocale(LC\_ALL, "RU");

double e = 0.0001; //точность эпсилона

cout << "Введите интревал: ";

cin >> a >> b;

int k = 0;

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

{

while (fabs(a - b) > e)

{

с = (b + a) / 2;

if (f(a) \* f(с) < 0)

{

b = с;

}

else

{

a = с;

}

k++;

cout << "Итерация " << k << ": " << с << endl;

}

cout << "Итог: " << с;

}

else

{

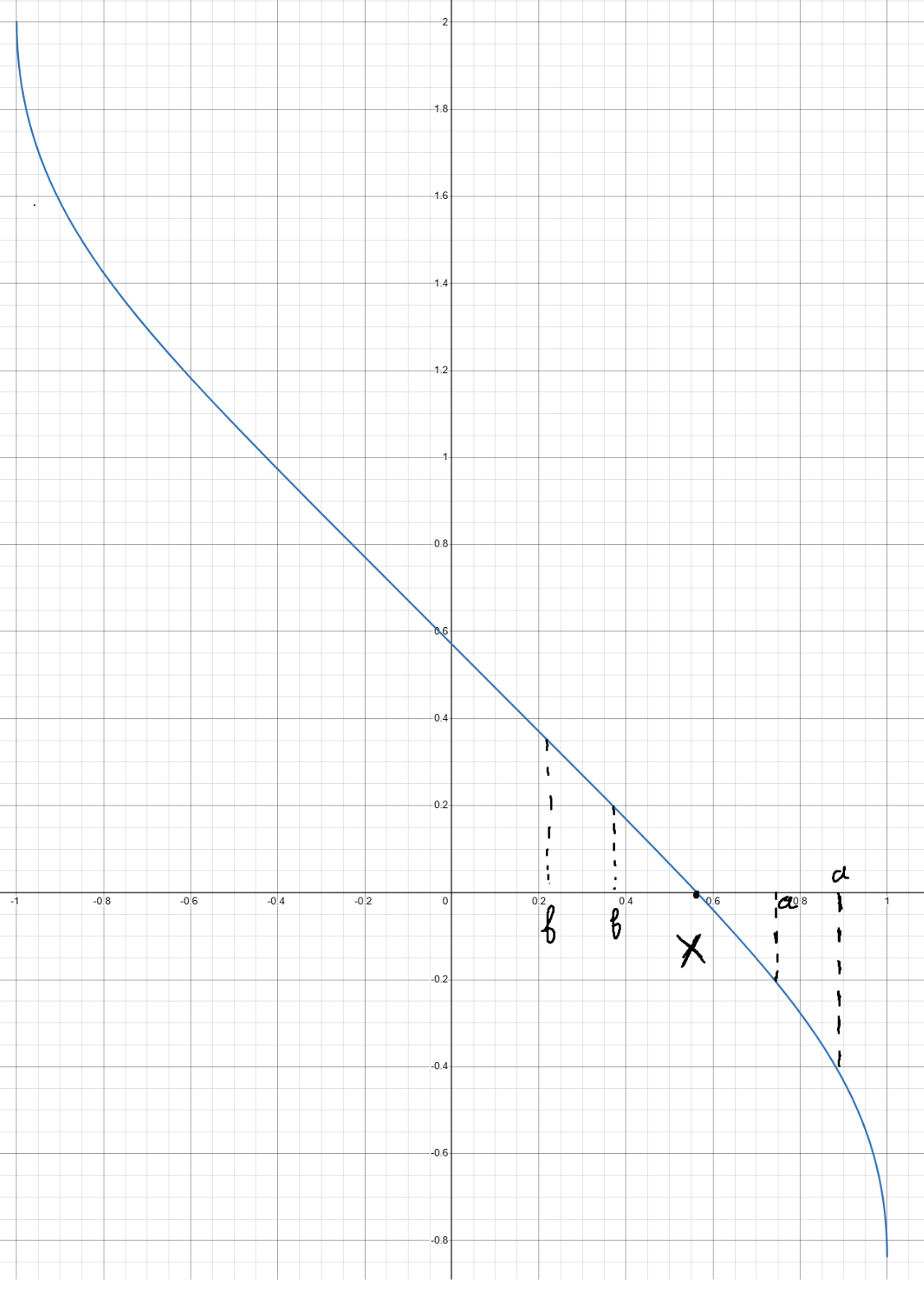
cout << "Фунция не пересекает ось Ox в интервале (a,b)";

}

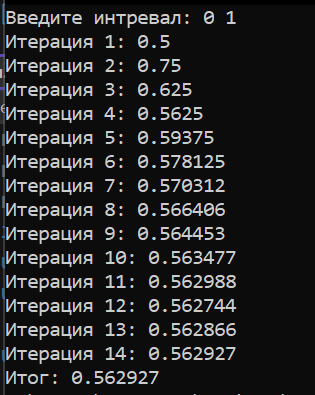
return 0;

}

График функции:



Результаты:



**Метод Ньютона**

Описание переменных:

1. Переменная вещественного типа е – точность вычислений

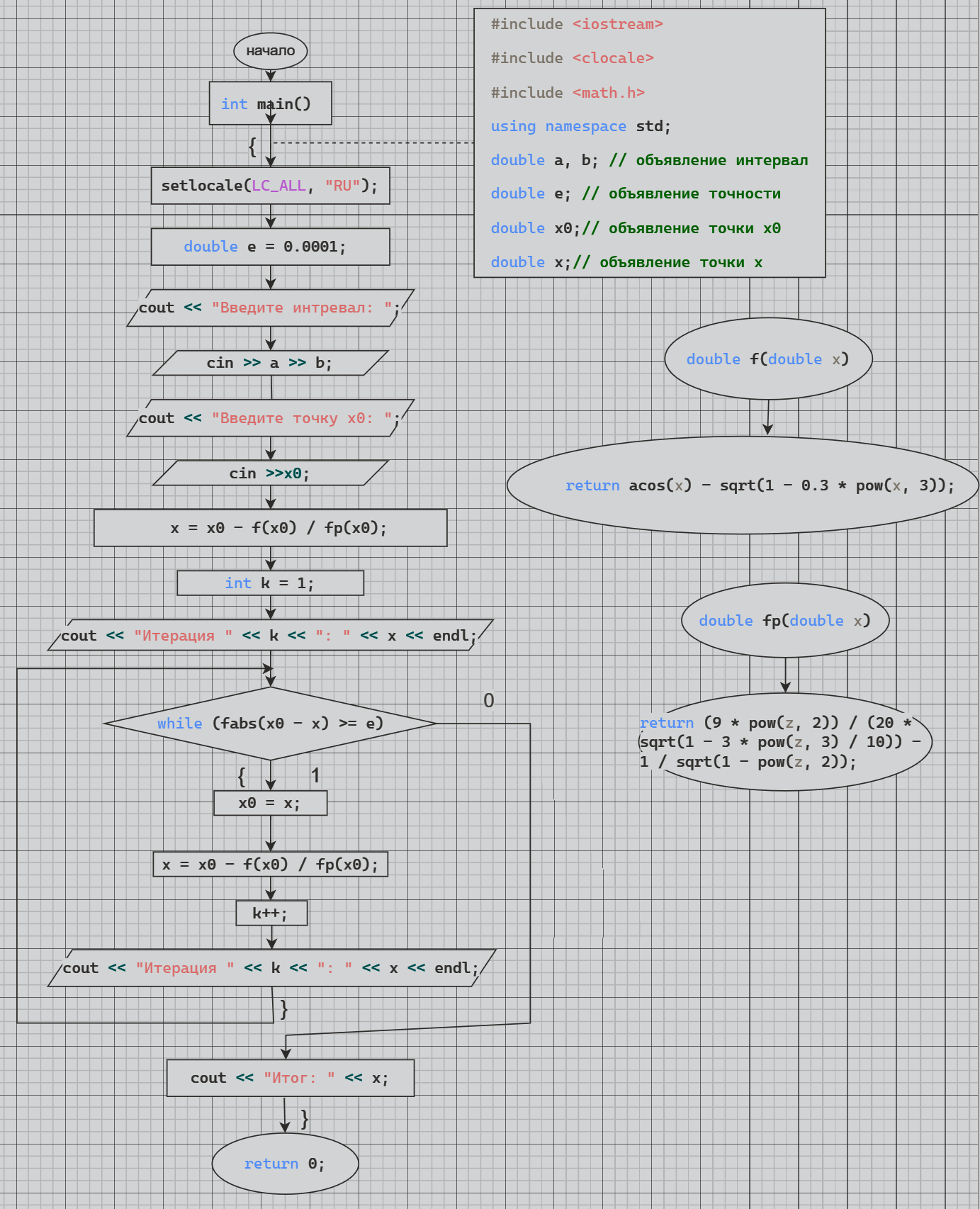
2. Переменные вещественного типа а, b – интервал в котором находится корень

3. Переменная целочисленного типа k – счётчик итераций

4. Переменные вещественного типа x0 – значение выбранной точки

5. Переменные вещественного типа x – значение точки после преобразований

Блок схема:



Код:

#include <iostream>

#include <clocale>

#include <math.h>

using namespace std;

double a, b; // объявление интервал

double e; // объявление точности

double x0;// объявление точки x0

double x;// объявление точки x

double f(double z) // значение функции в точке касания

{

return acos(z) - sqrt(1 - 0.3 \* pow(z, 3));

}

double fp(double z) // значение производной функции в точке касания

{

return (9 \* pow(z, 2)) / (20 \* sqrt(1 - 3 \* pow(z, 3) / 10)) - 1 / sqrt(1 - pow(z, 2));

}

int main()

{

setlocale(LC\_ALL, "RU");

e = 0.0001; // задаём точность

cout << "Введите интревал: ";

cin >> a >> b;

cout << "Введите точку x0: ";

cin >> x0;

x = x0 - f(x0) / fp(x0);

int k = 1;

cout << "Итерация " << k << ": " << x << endl;

while (fabs(x0 - x) >= e)

{

x0 = x;

x = x0 - f(x0) / fp(x0);

k++;

cout << "Итерация " << k << ": " << x << endl;

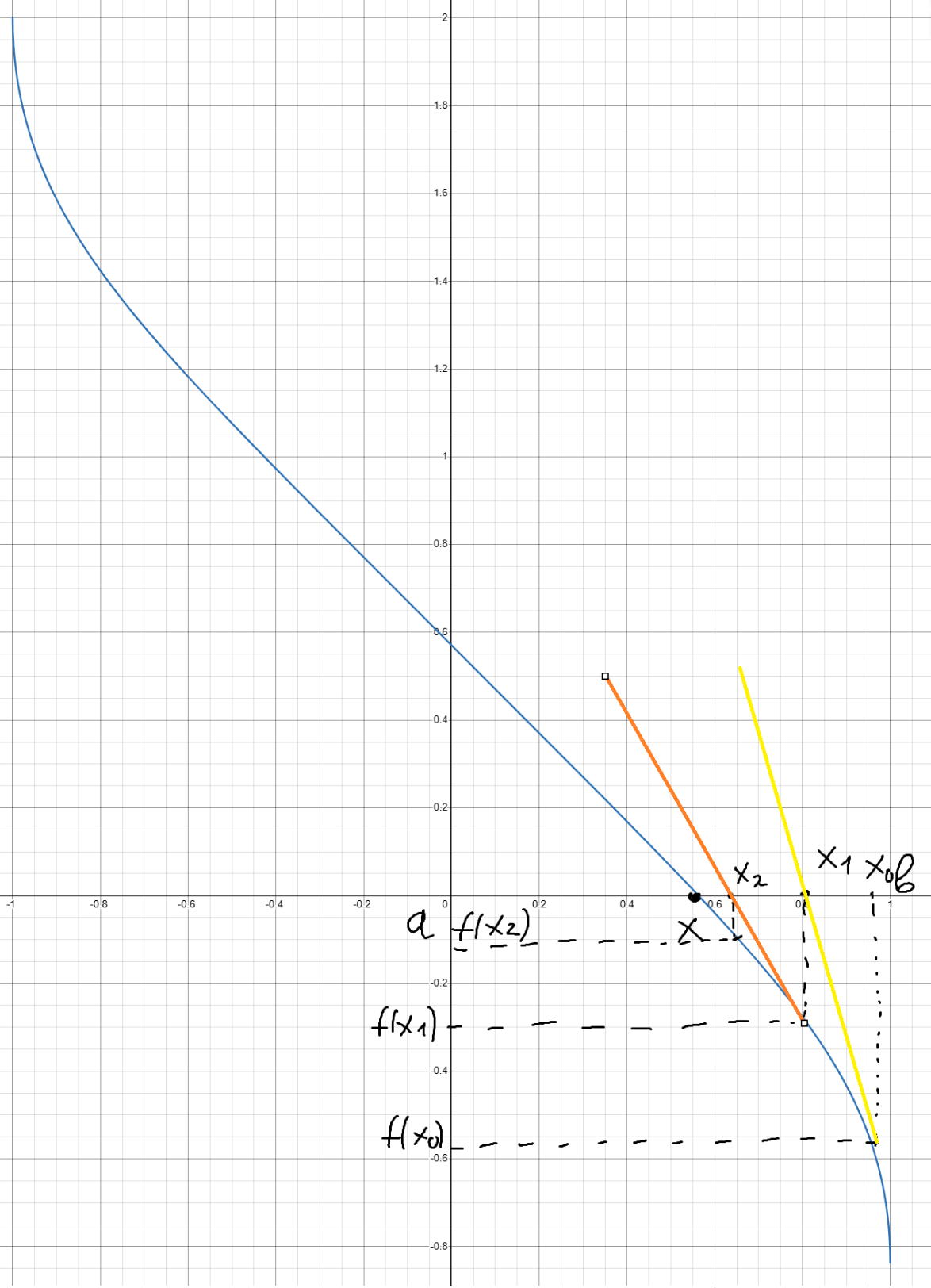
}

cout << "Итог: " << x;

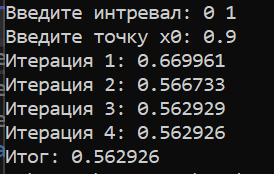
return 0;

}

График функции:



Результат:



**Метод Итераций**

Описание переменных:

1. Переменная вещественного типа е – точность вычислений

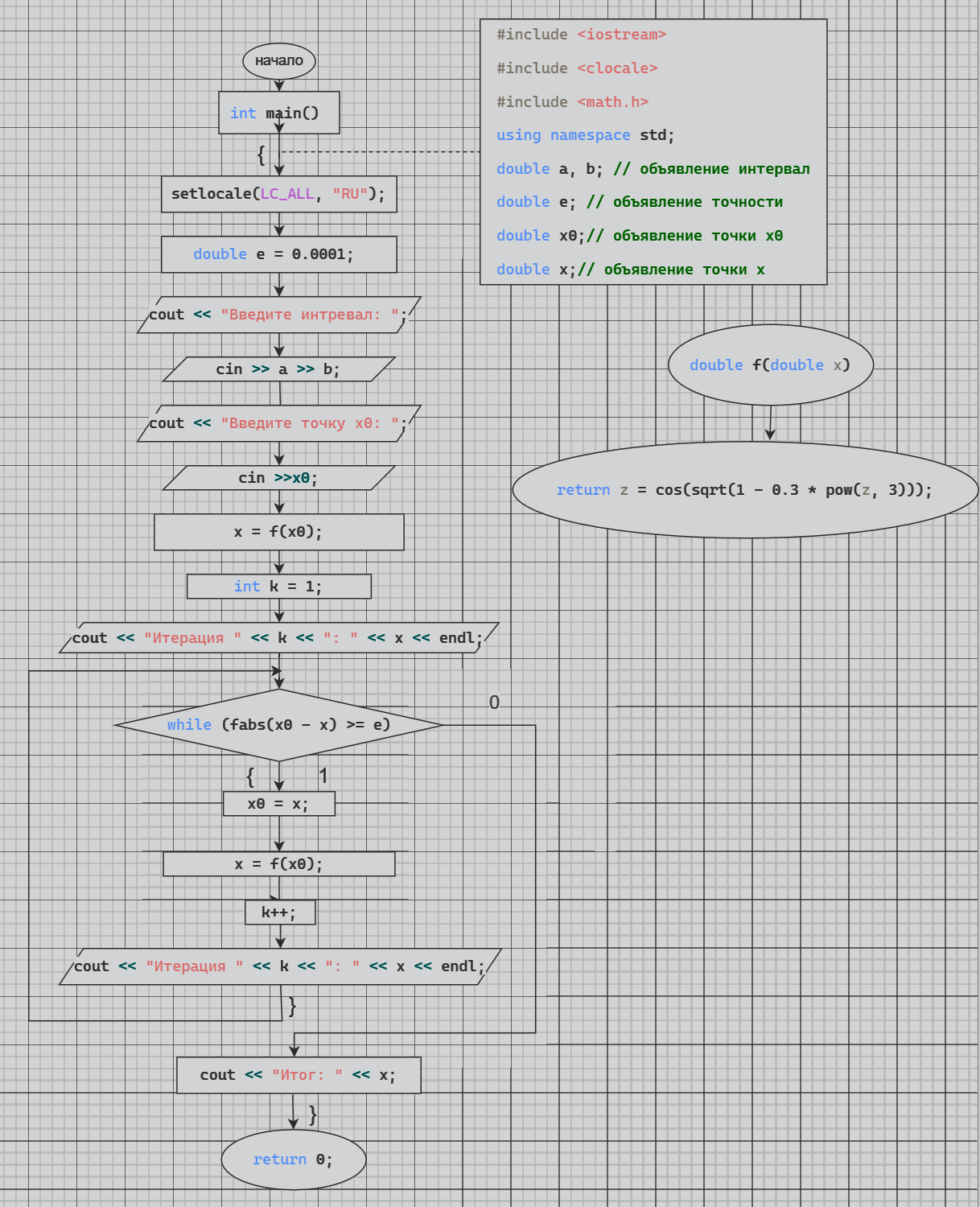
2. Переменные вещественного типа а, b – интервал в котором находится корень

3. Переменная целочисленного типа k – счётчик итераций

4. Переменные вещественного типа x0 – значение выбранной точки

5. Переменные вещественного типа x – значение точки после преобразований

Блок схема:



Код:

#include <iostream>

#include <clocale>

#include <math.h>

using namespace std;

double a, b; // объявление интервал

double e; // объявление точности

double x0;// объявление точки x0

double x;// объявление точки x

double f(double z) // значение функции fi(x)

{

return z = cos(sqrt(1 - 0.3 \* pow(z, 3)));

}

int main()

{

setlocale(LC\_ALL, "RU");

e = 0.0001; // задаём точность

cout << "Введите интревал: ";

cin >> a >> b;

cout << "Введите точку x0: ";

cin >> x0;

x = f(x0);

int k = 1;

cout << "Итерация " << k << ": " << x << endl;

while (fabs(x0 - x) >= e)

{

x0 = x;

x = f(x0);

k++;

cout << "Итерация " << k << ": " << x << endl;

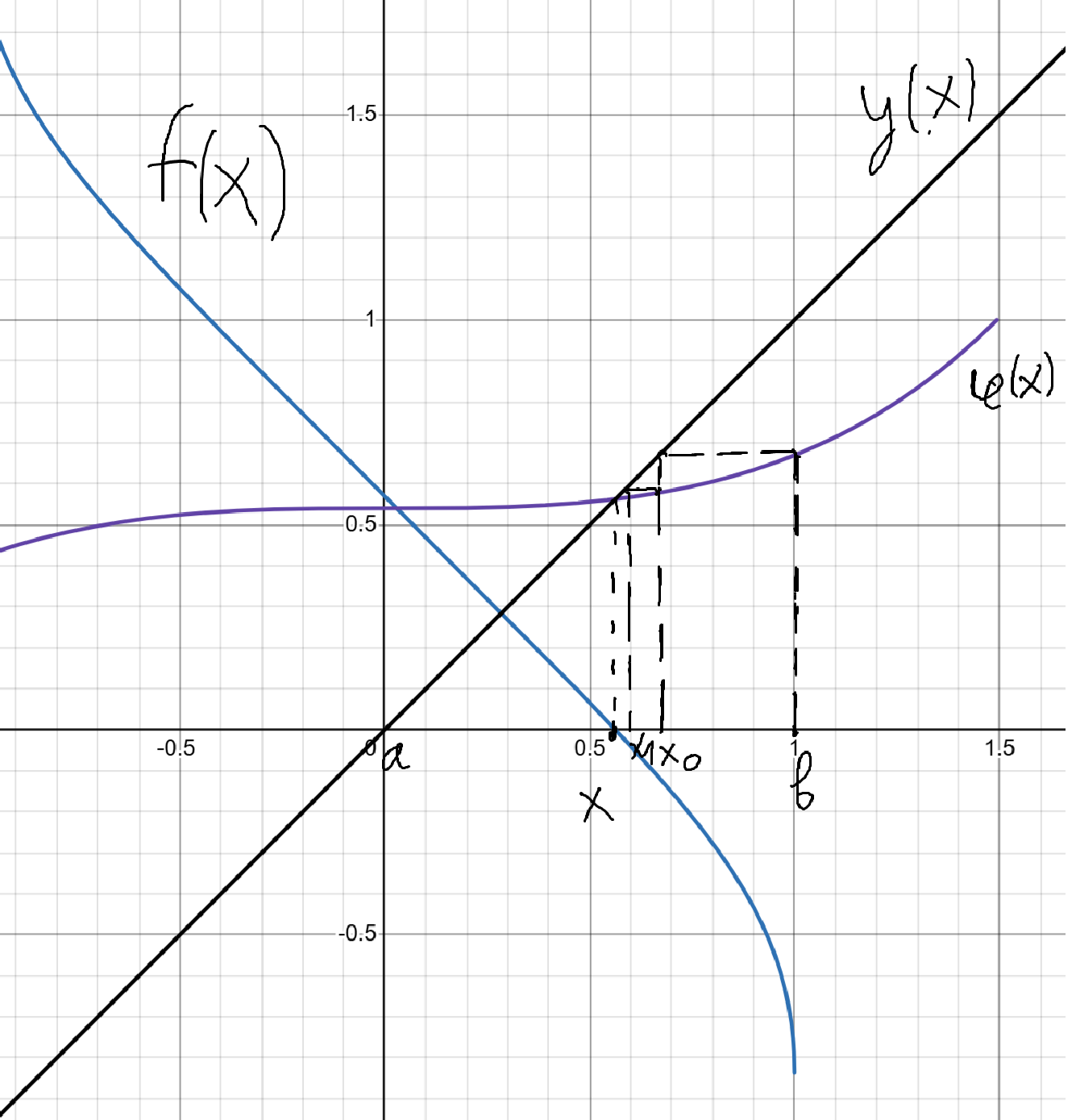
}

cout << "Итог: " << x;

return 0;

}

График функции:



Результат:

