Министерство науки и высшего образования Российской Федерации

Федеральное государственное автономное учреждение высшего образования

«Пермский национальный исследовательский политехнический университет»

ПНИПУ

**Лабораторная работа**

**«Решение нелинейных уравнений»**

Выполнил:

студент группы РИС-23-2б

Ившин Максим Сергеевич

Проверила:

доцент кафедры ИТАС

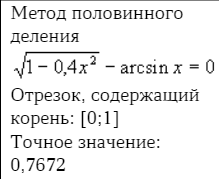
О.А. Полякова

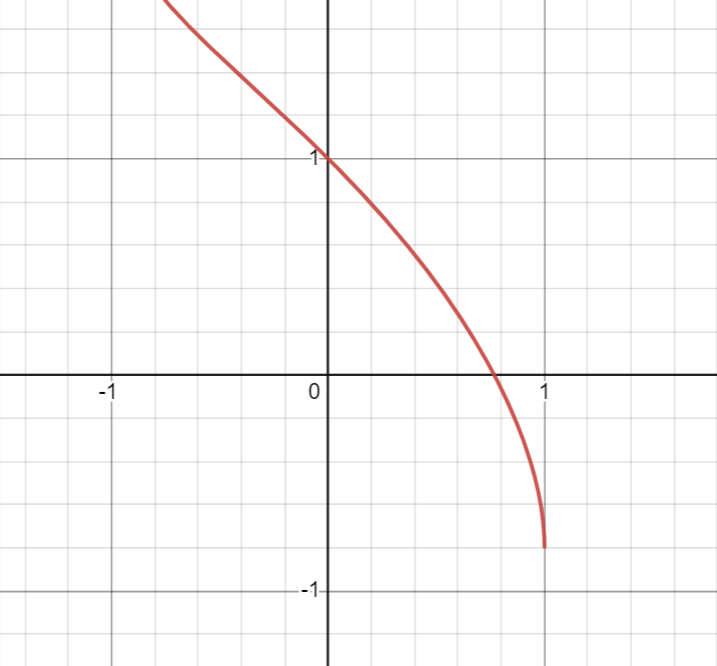
2023 г.

***Поставленная задача:***

Создать алгоритм решения нелинейных уравнений методом половинного деления (бисекции), методом Ньютона (методом касательных), методом итераций.

Дано:





**Метод половинного деления (метод бисекции)**

***Анализ метода:***

Метод предполагает нахождение приближенного значения исходного корня на данном отрезке, основанное на теореме Больцана-Коши о промежуточном значении:

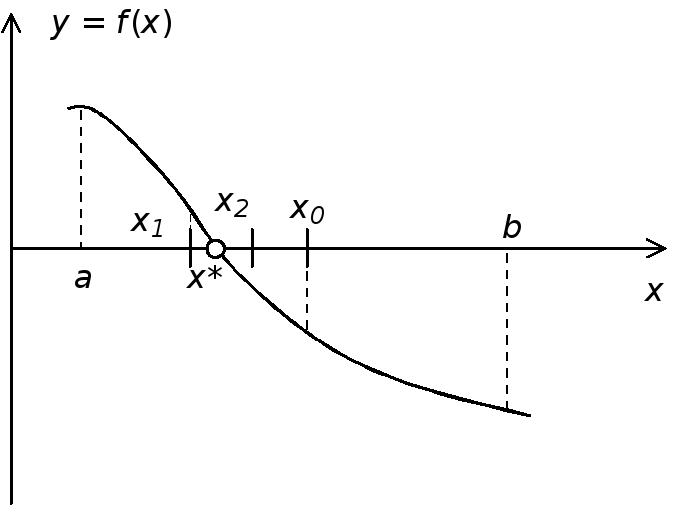
«Если функция, непрерывная на отрезке, принимает на его концах значения разных знаков, то на отрезке есть точка, в которой функция обращается в нуль».

Решение сводится к нахождению на каждом шаге приблизительного корня уравнения, сравнение его с найденным на предыдущем шаге значением, и если разность этих значений по модулю меньше или равно ε (заданной точности, обычно оно равняется 10-6, такую и возьмем в этой задаче), то последнее найденное значение и будет считаться исходным корнем.

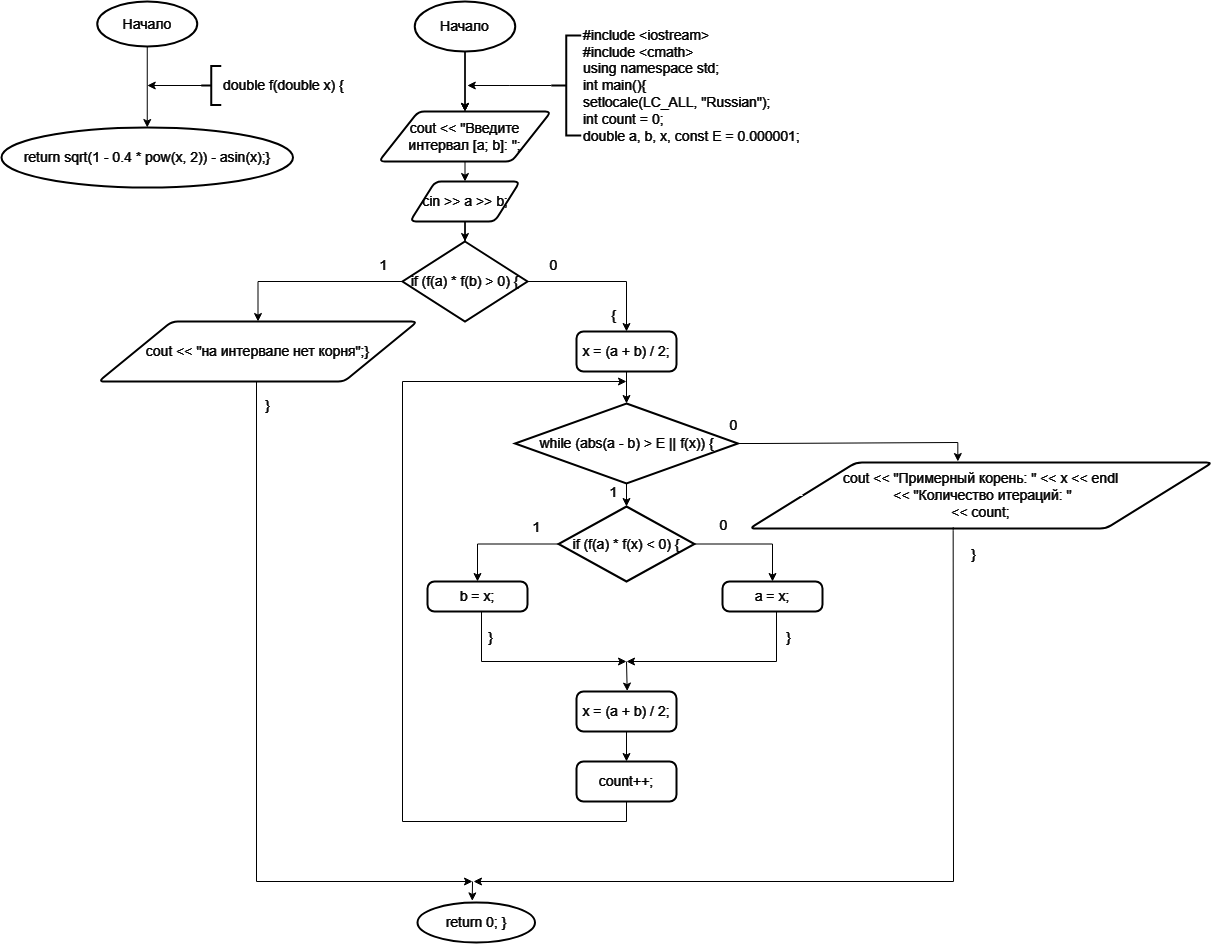
Метод половинного деления предусматривает поиск корня в интервале [a; b] путем деления этого интервала пополам и повторном применении теоремы для нахождения нового исходного промежутка, в котором имеется корень уравнения:

Середина интервала находится по формуле: x = (a+b)/2, проверяем истинность выражения |x - x0| <= ε, если оно верно, то x – последний найденный корень – и будет исходным корнем уравнения, иначе из промежутков [a; x0] и [x0; b] отбрасываем тот, что не имеет корня и продолжаем деления промежутка.

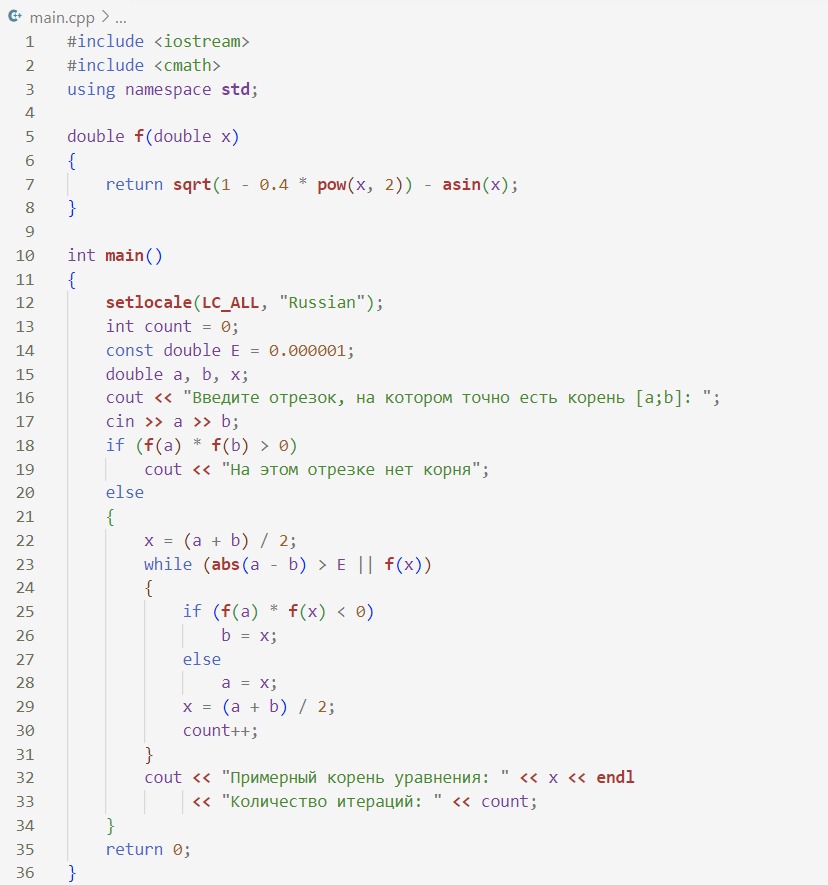
**Геометрическая интерпретация:**



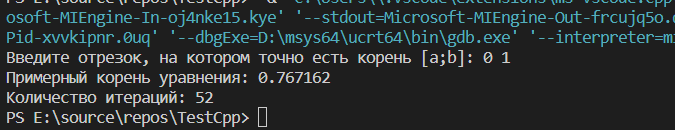
**Блок-схема**

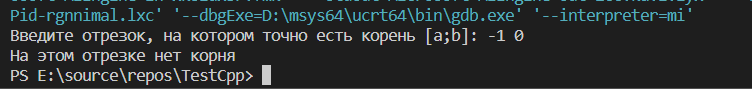


**Код на C++**

****

**Результаты**

****

****

**Метод Ньютона (касательной)**

**Анализ метода:**

Пусть уравнение F(x) = 0 имеет один корень на отрезке [a; b], причем F′(x) и F″(x) определены, непрерывны и сохраняют постоянные знаки на отрезке [a; b].

Выберем на отрезке [a; b] произвольную точку х0 – нулевое приближение. Процесс нахождения корня уравнения сводится к вычислению чисел xn по формуле:



Процесс вычисления продолжается до тех пор, пока не будет выполнено условие:

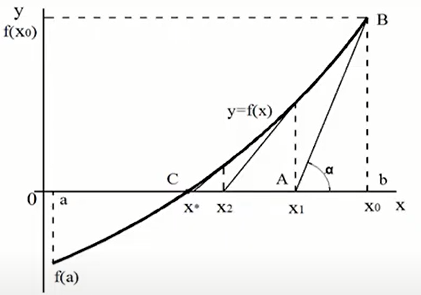


Точку х0 необходимо выбирать так, чтобы выполнялось условие:

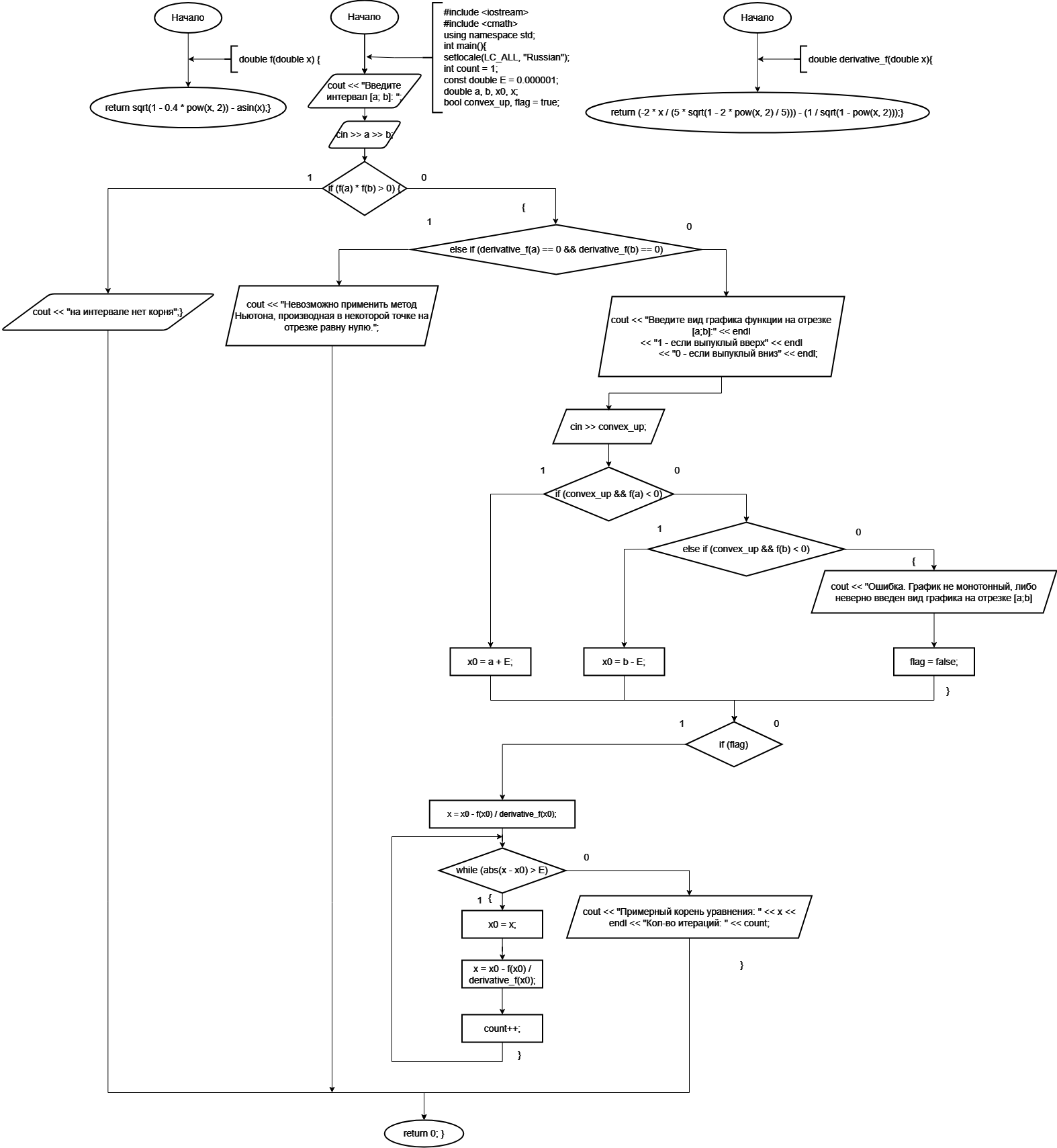


Соответственно должна существовать вторая производная функции на отрезке. Это нужно для того, чтобы новые найденные корни постепенно сближались к исходному корню уравнения.

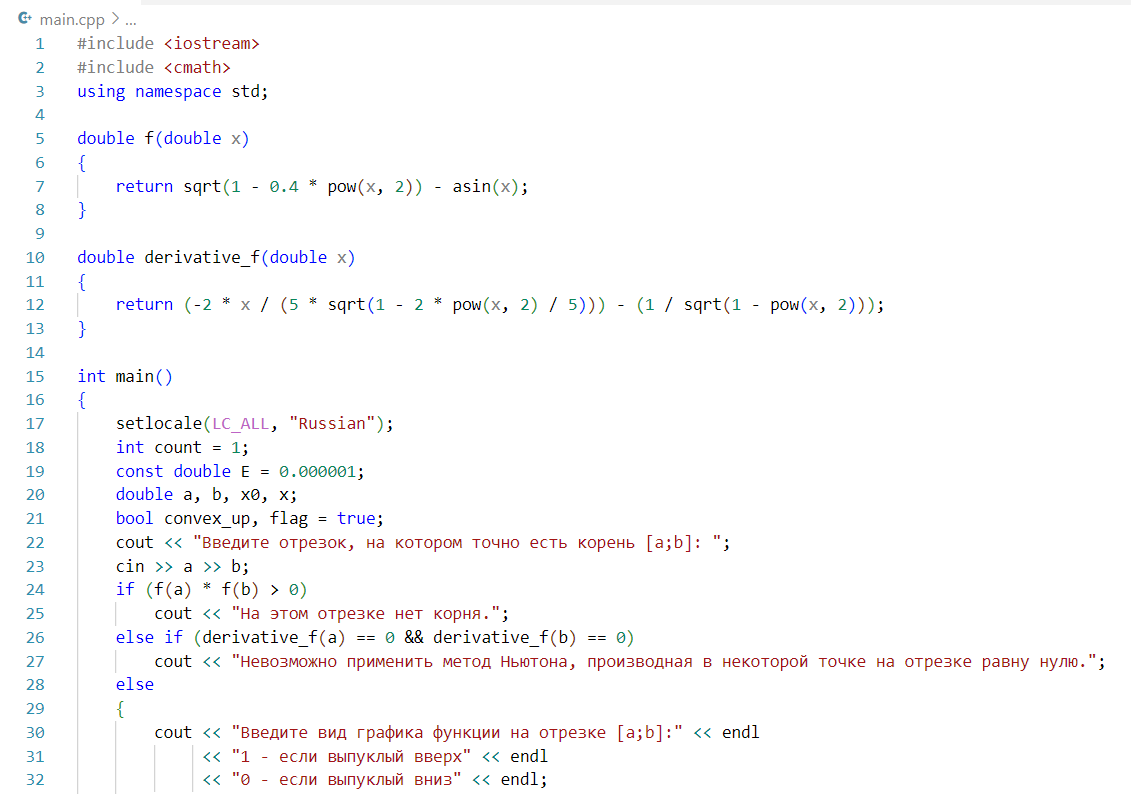
**Геометрическая интерпретация:**

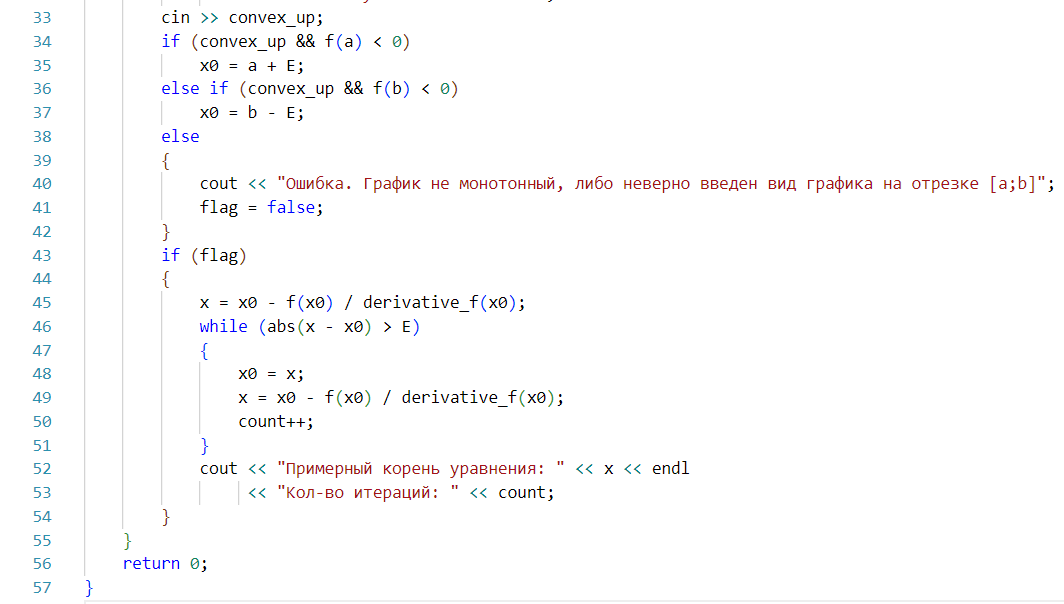


**Блок-схема**

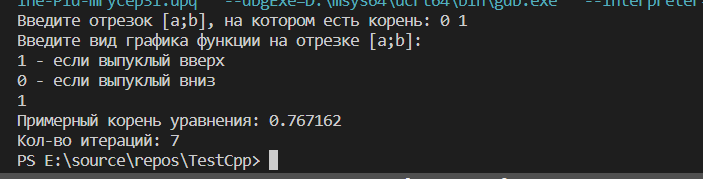


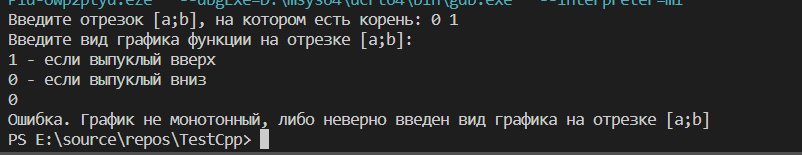
**Код на C++**

****

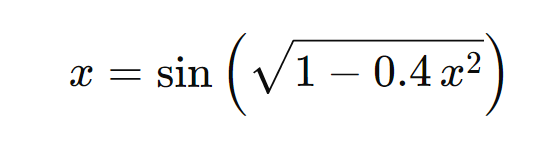
****

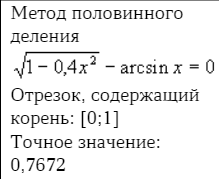
**Результаты**

****

****

**Метод итераций**

 Для использования этого метода представим исходное уравнение f(x) = 0 в виде: x = f(x). Это уравнение можно получить путем выделения x из исходного уравнения:

 =>

Далее на заданном отрезке [a;b] выбираем произвольную точку x0, от нее будем находить следующие приближения, например можно брать середину данного отрезка: x0 = (a+b) / 2, и находим следующее значение:

xn = f(xn-1), n = 1, 2, 3…

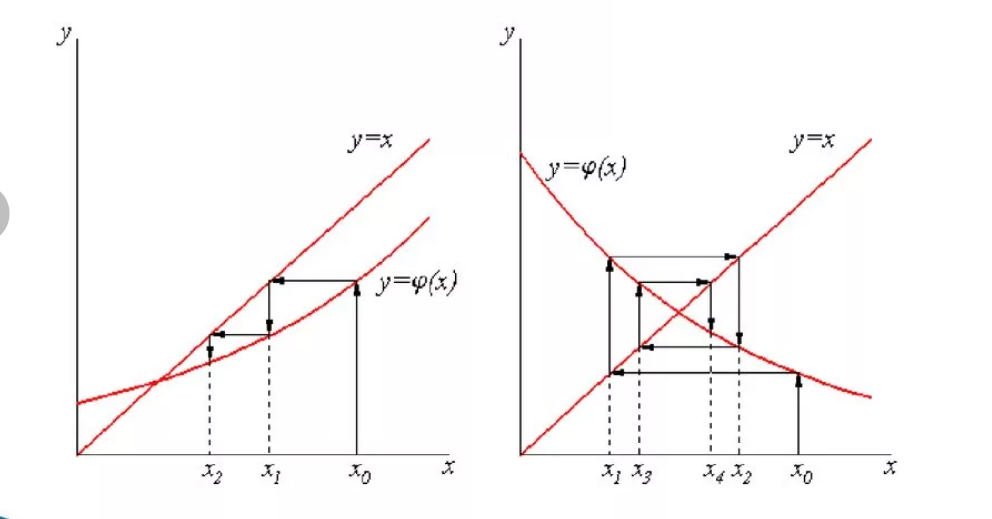
Процесс итераций продолжается до тех пор, пока не будет достигнута заданная точность:

|xn - xn-1|  ε

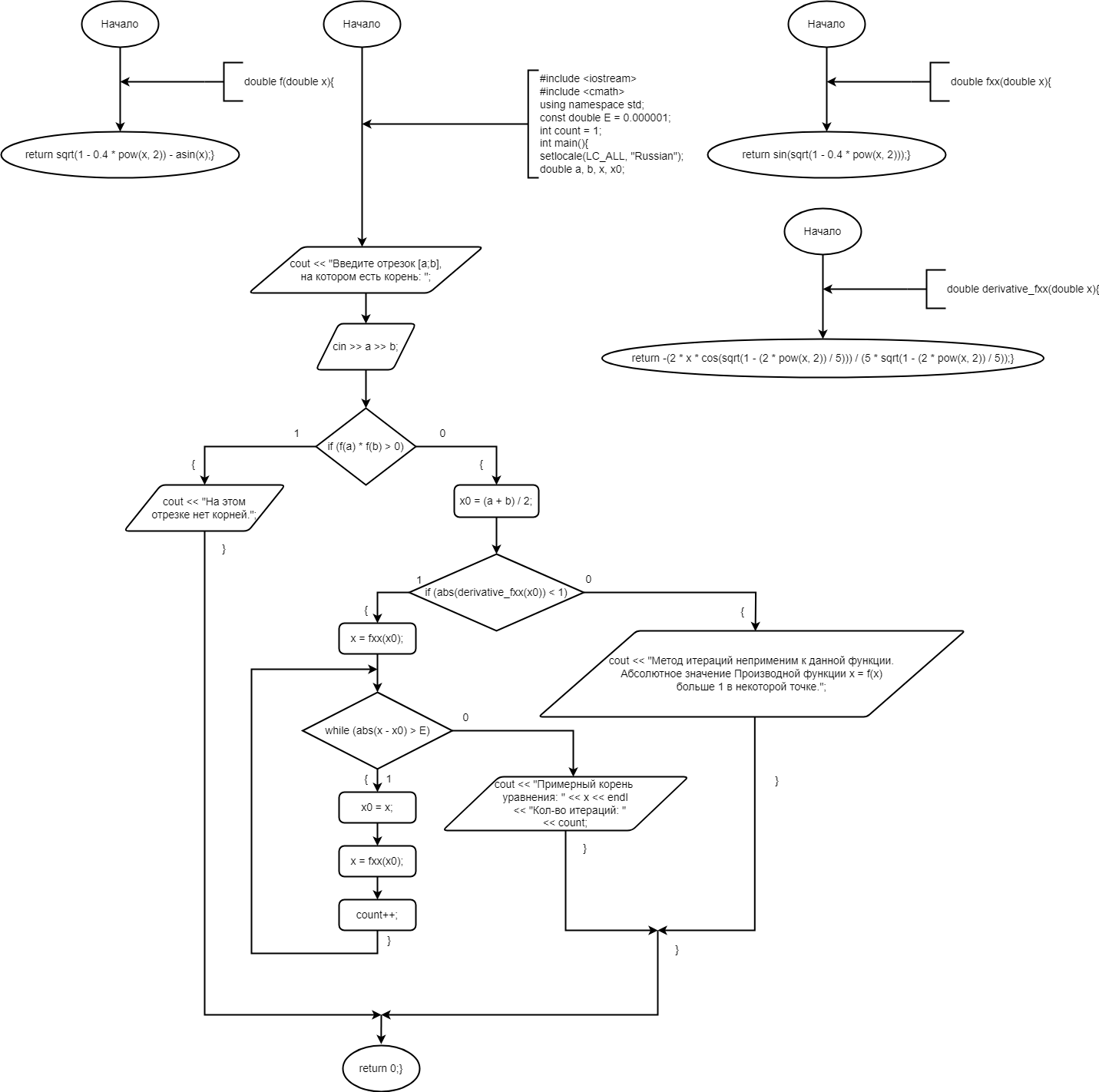
После выполнения этого неравенства исходным корнем считаем xn – последний найденный корень.

Этот метод имеет жесткое ограничение: производная функции x = f(x) на отрезке [a;b] должна быть строго меньше 1.

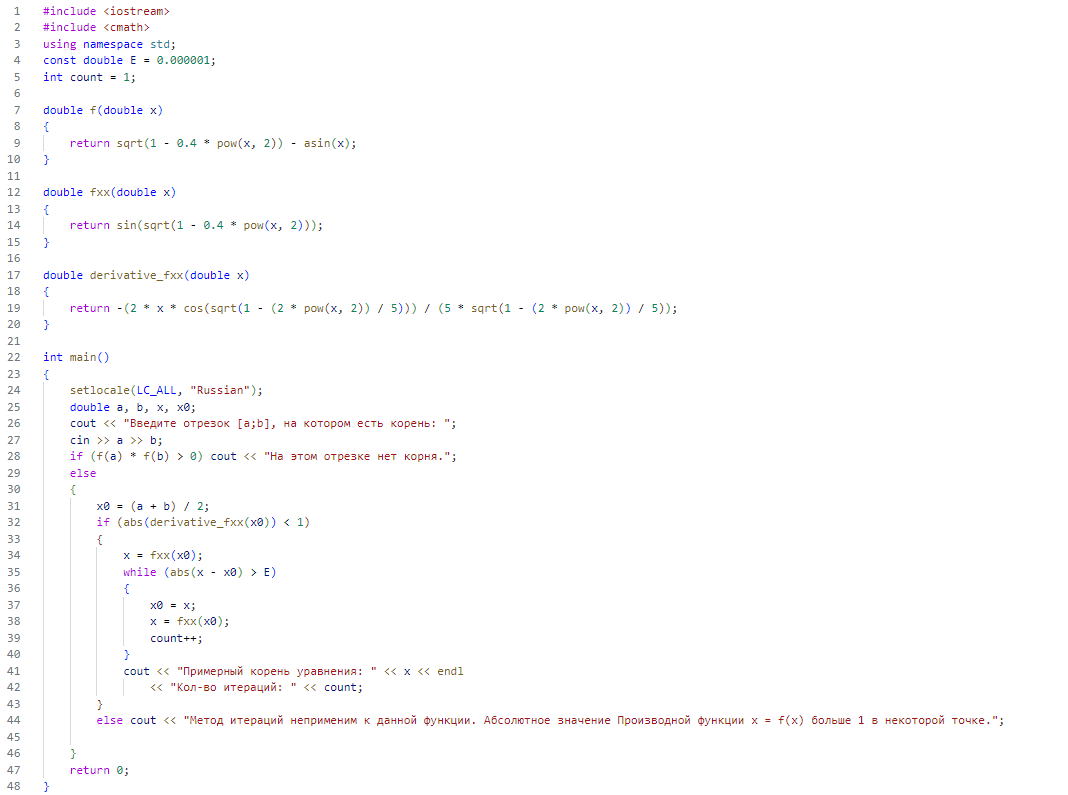
**Геометрическая интерпретация:**



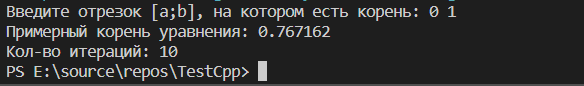
**Блок-схема**

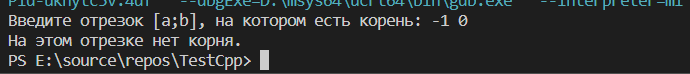
****

**Код на C++**



**Результаты**





**Вывод**

На примере одного уравнения были проведены сравнения трёх методов. На основании результатов можно выявить, что самым быстрым методом является метод касательных (Ньютона), поскольку имеет наименьшее количество итераций за время работы программы.