Пермский национальный исследовательский политехнический университет

Кафедра информационные технологии и автоматизированные системы

Лабораторная работа №1

**Методы решения нелинейных уравнений**

Вариант: 9

|  |  |
| --- | --- |
|  | Выполнил:  студент группы: ИВТ-24-2б  Ваулин Артём Сергеевич  Проверил:  доцент кафедры ИТАС  О.А. Полякова |

Пермь 2024

# Постановка задачи

Решить уравнение: 3 методами: методом Ньютона, итерации и методом половинного деления (бисекции). C заданной точностью на отрезке

# Геометрическая интерпретация

График функции для метода Ньютона

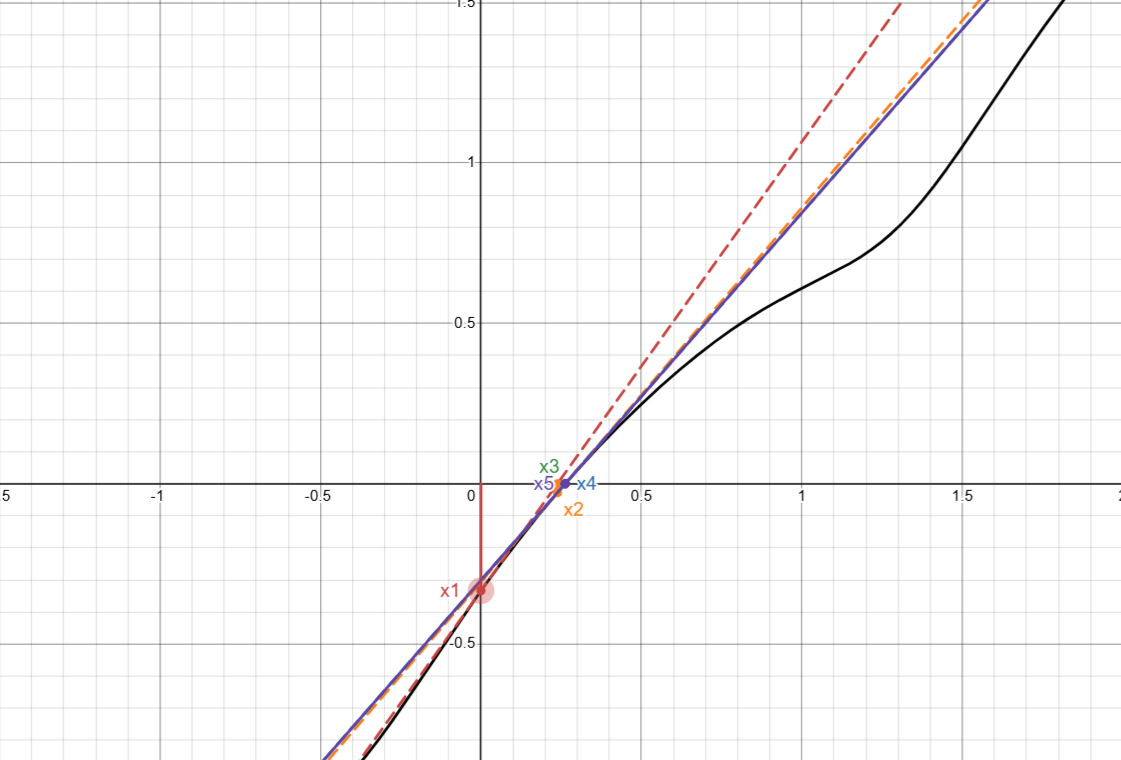


График функции для метода Бисекции

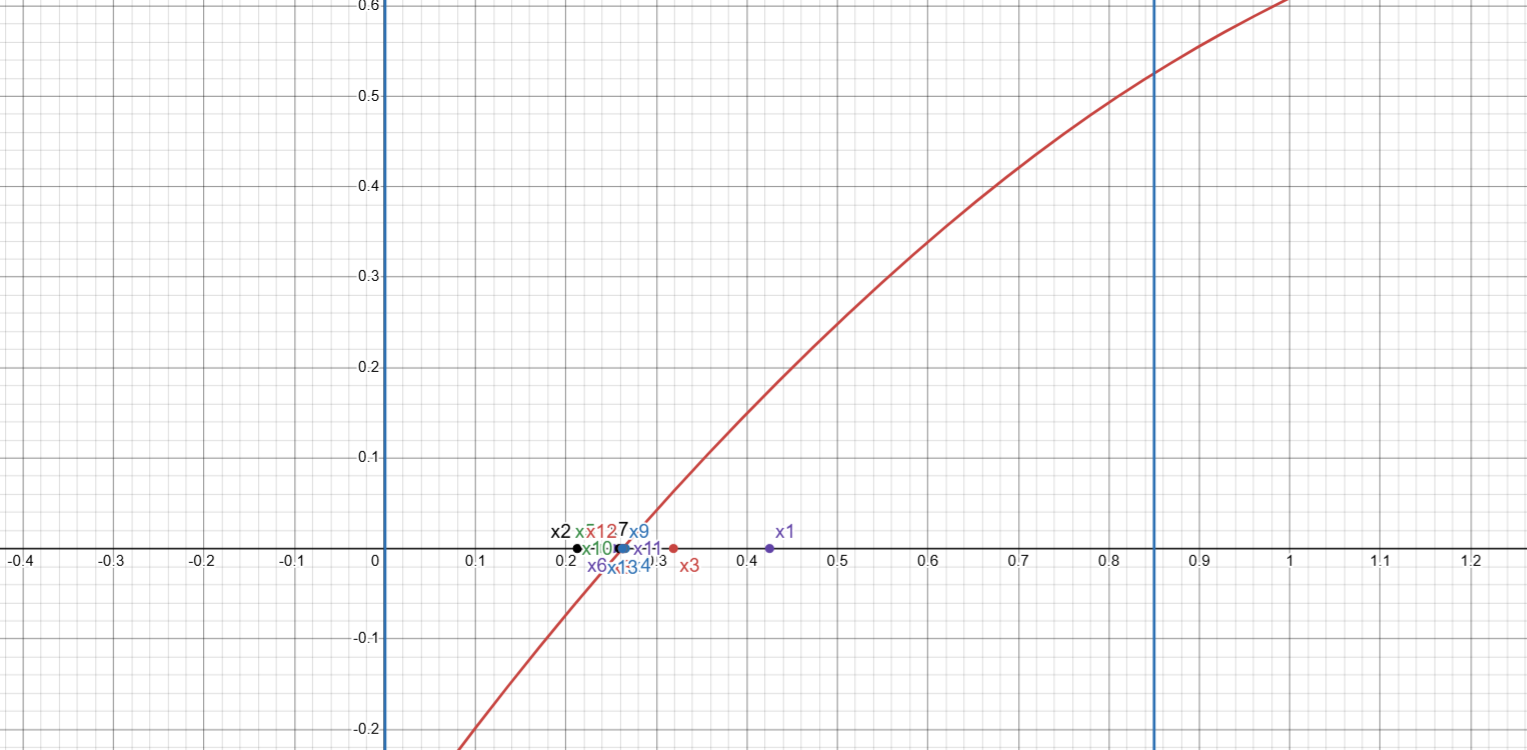
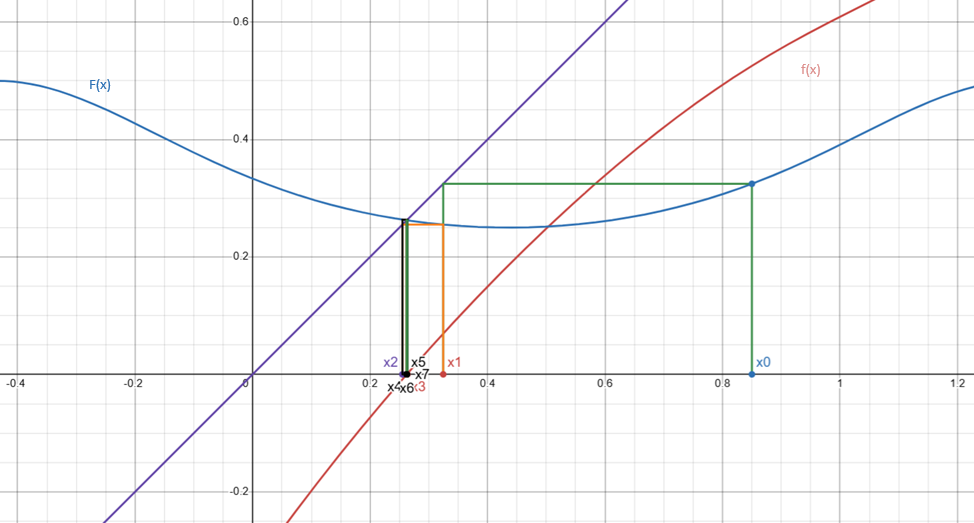


График функции для метода Итерации



# Анализ задачи

Метод Ньютона

1. **Определение функции:**
2. **Вычисление производных:**

* Первая производная:
* Вторая производная:

1. **Инициализация:** задаем начальное приближение и проверяем условие
2. **Итерации:**

* Обновляем значение по формуле:
* Продолжаем итерации, пока значение функции не станет меньше заданной точности

Метод Бисекции

1. **Определение функции:**
2. **Инициализация**: устанавливаем начальные значения интервала и *b*, где функция меняет знак.
3. **Итерации:**

* Находим среднюю точку интервала *x*:
* Если , то корень найден
* Если , обновляем интервал:
* Иначе:
* Продолжаем итерации, пока длина интервала не станет меньше заданной точности

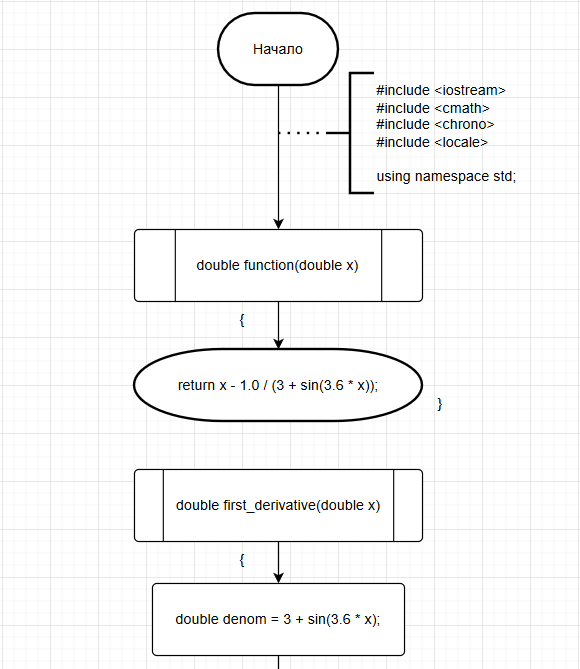
Метод Итерации

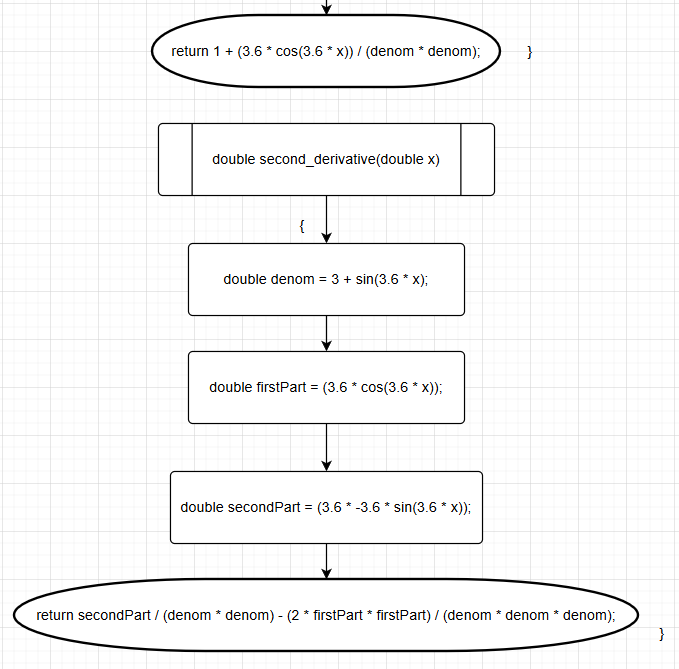
1. **Определение функции:**
2. **Переписывание функции для итерационного метода**:
3. **Инициализация**: устанавливаем начальное значение
4. **Итерации**:

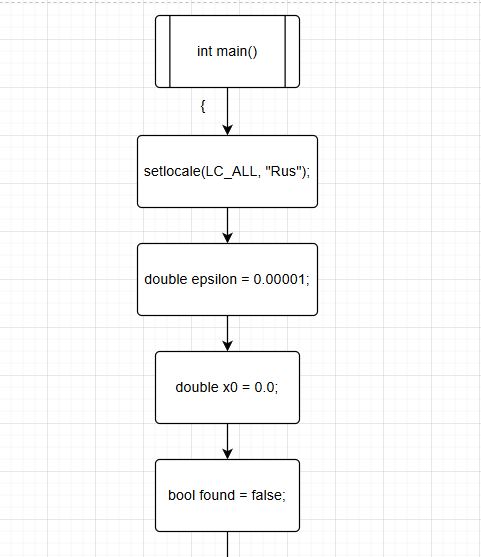
* Находим новое значение *x*:
* Продолжаем итерации, пока разница между значениями не станет меньше заданной точности

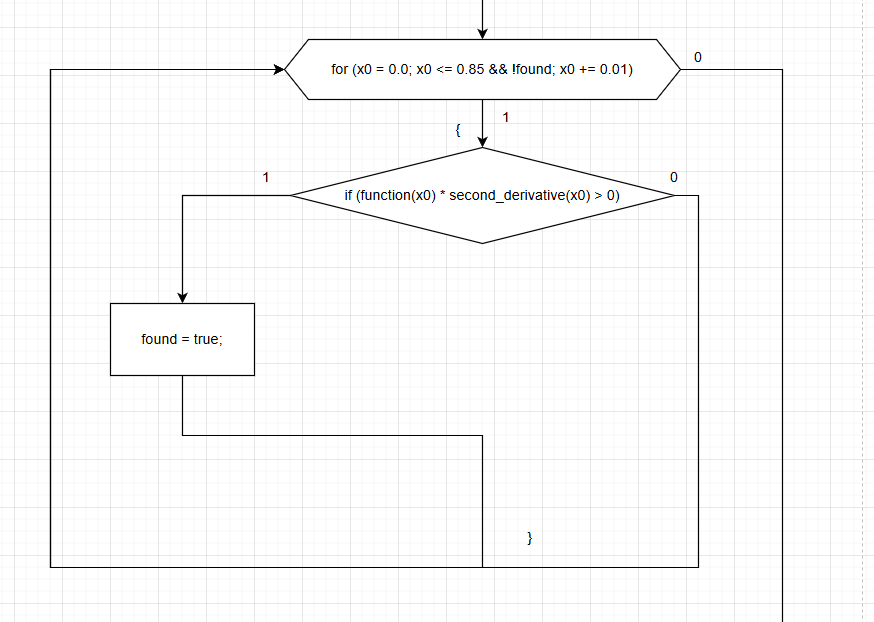
# Блок-схемы

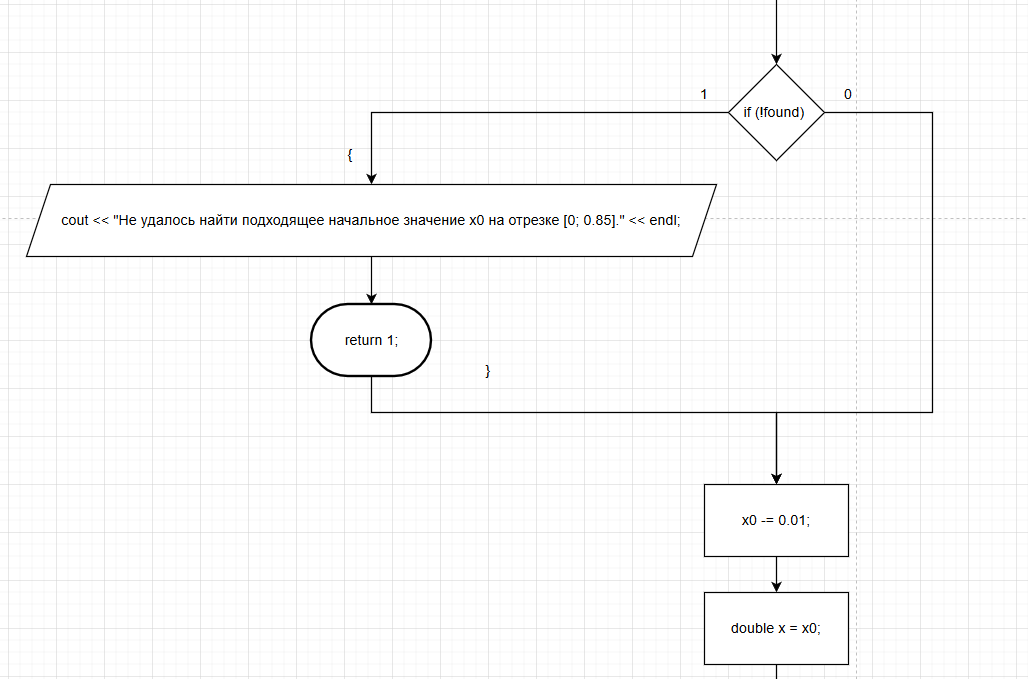
Метод Ньютона

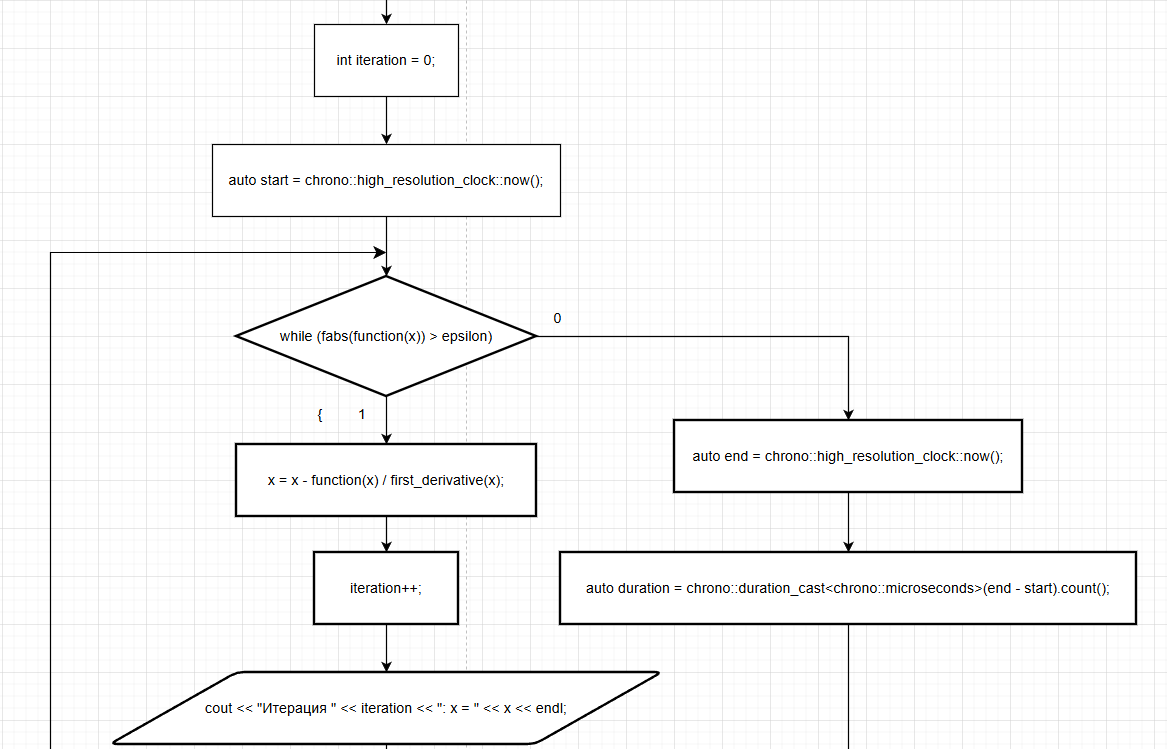


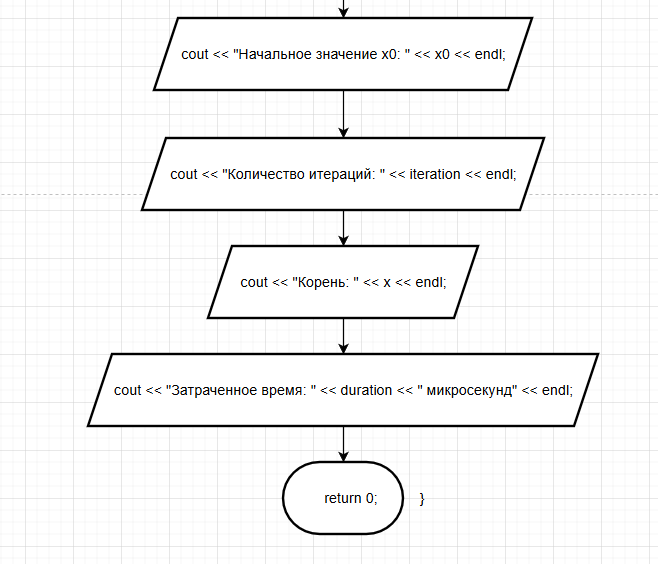




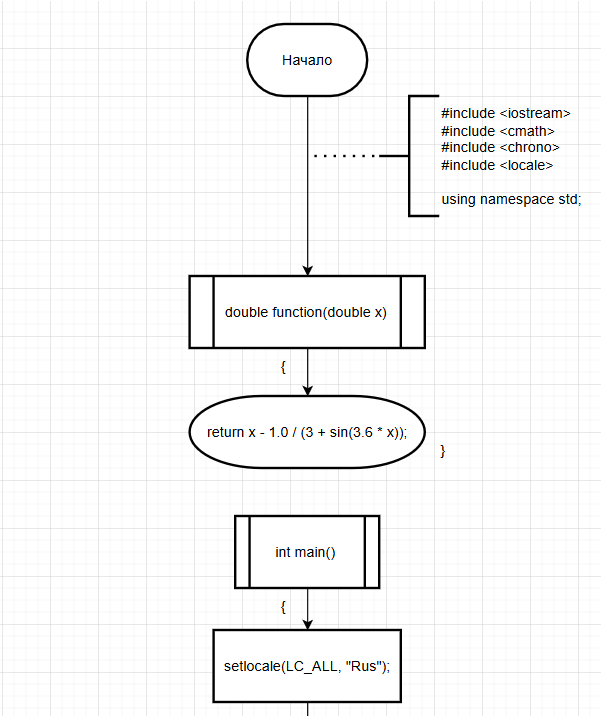


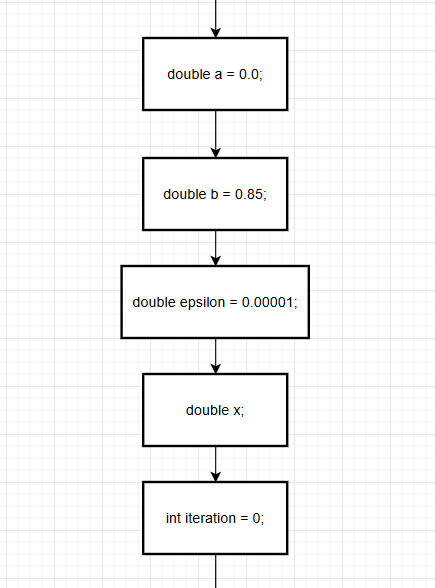


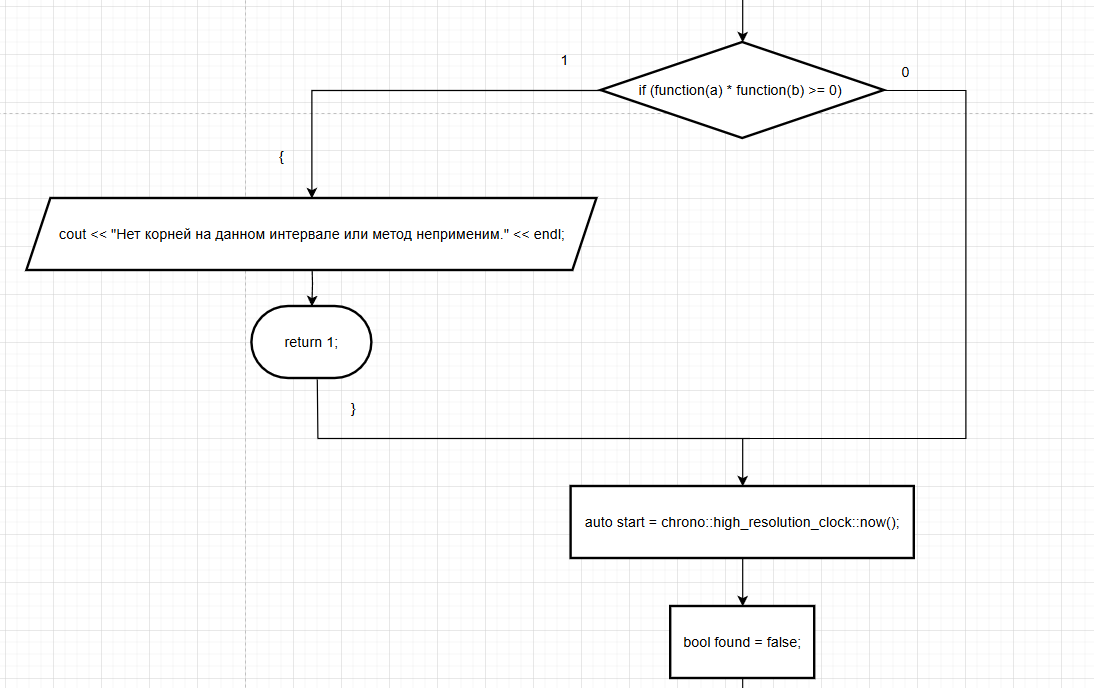


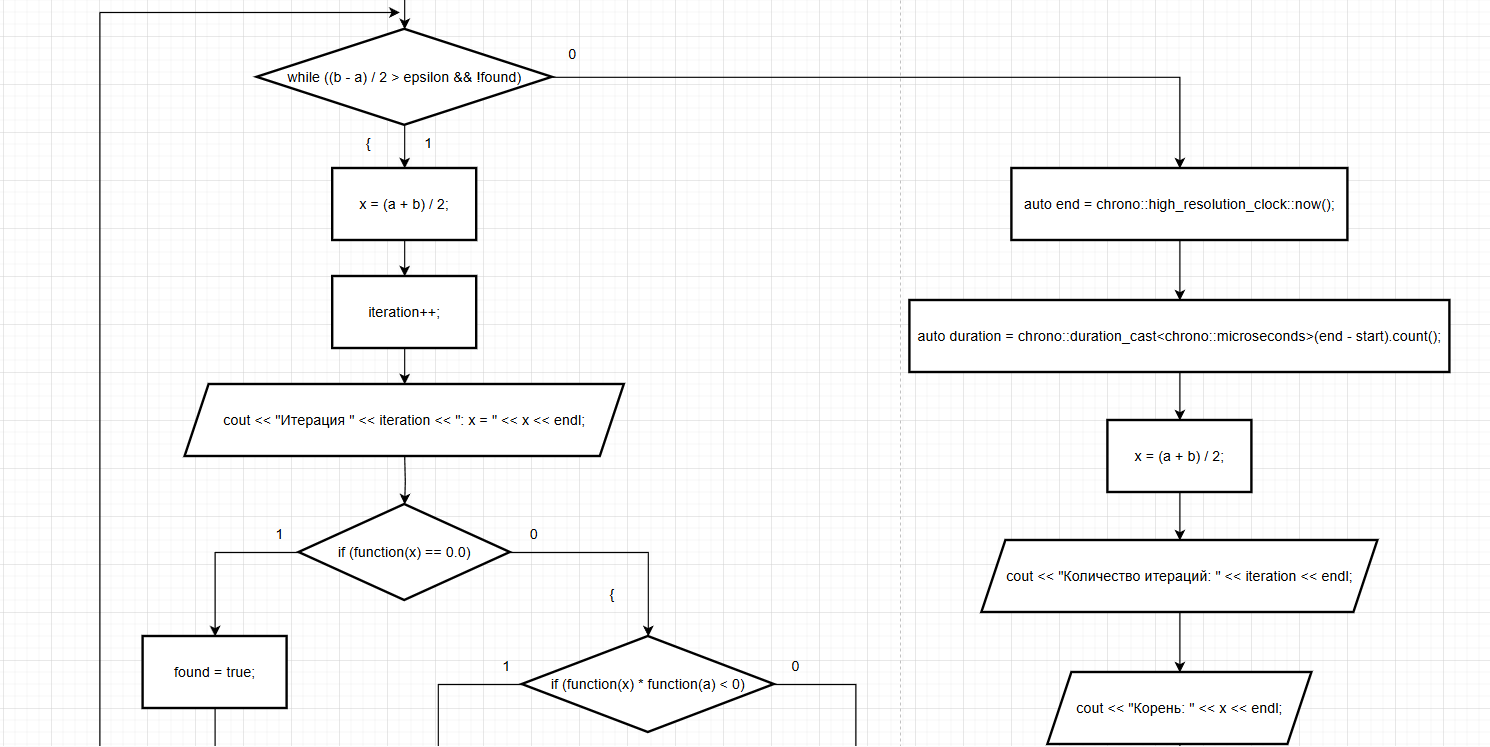


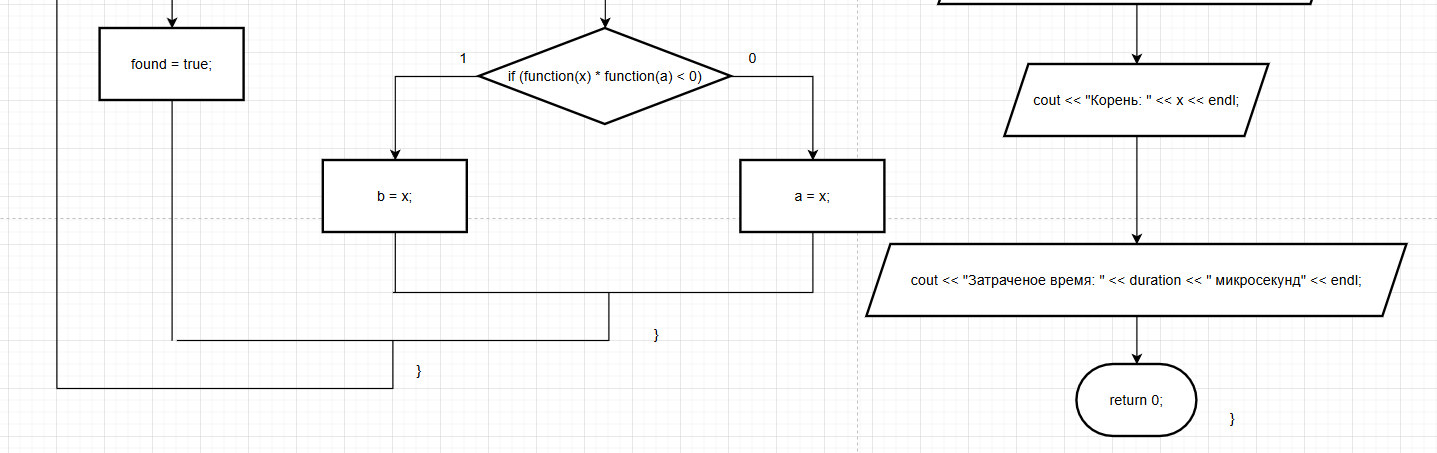
Метод Бисекции



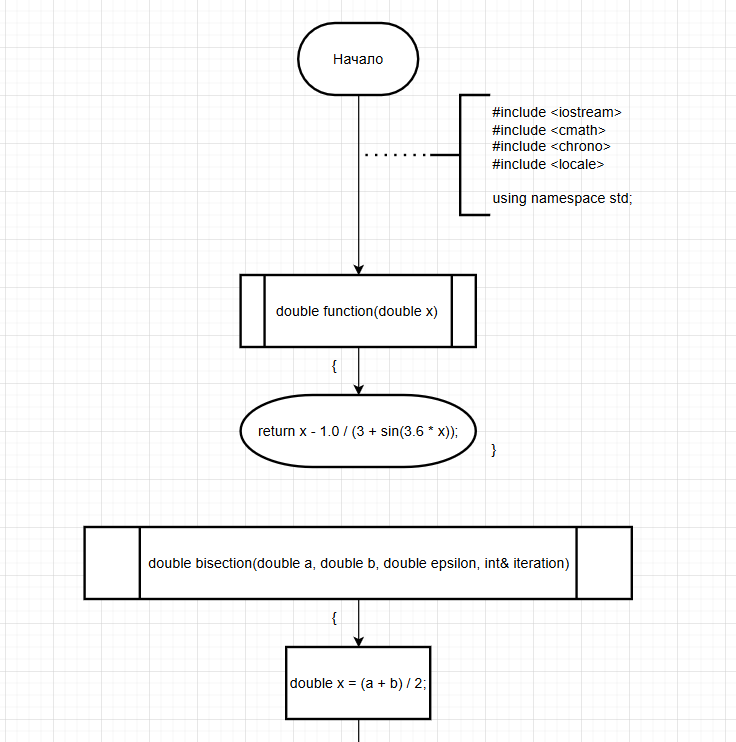


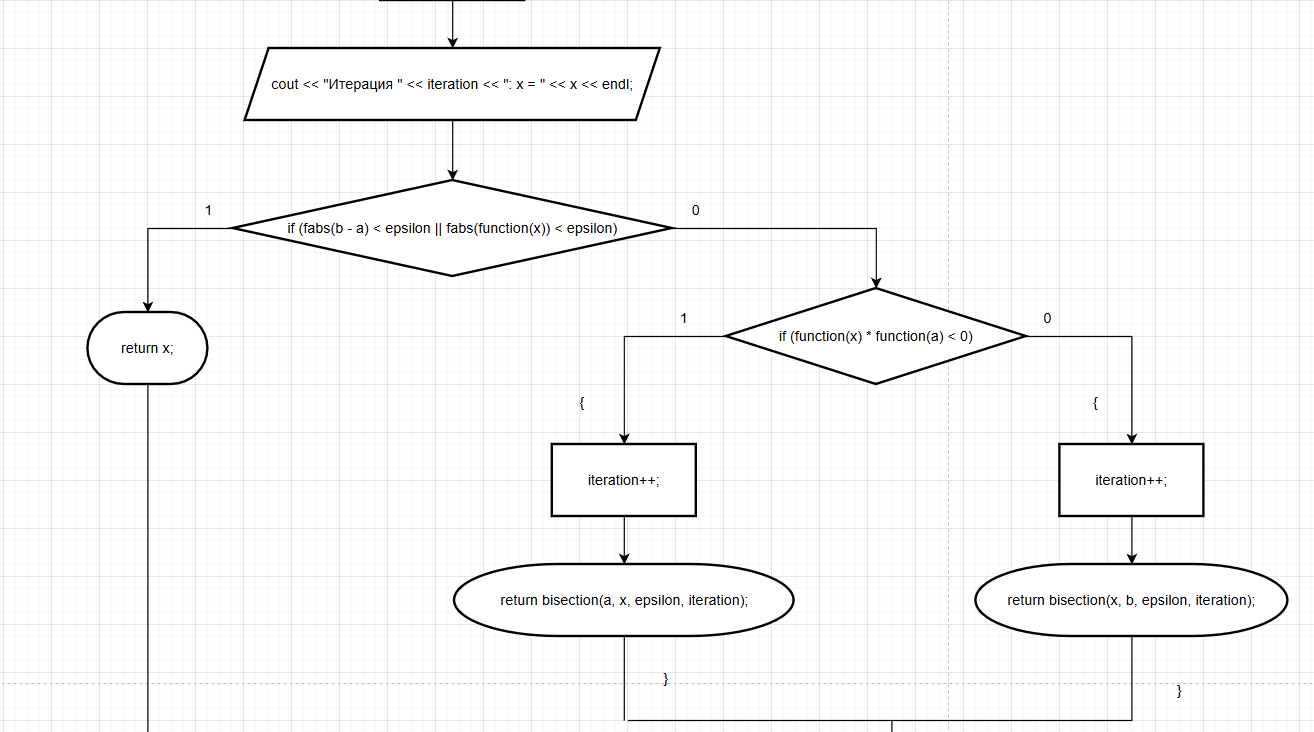


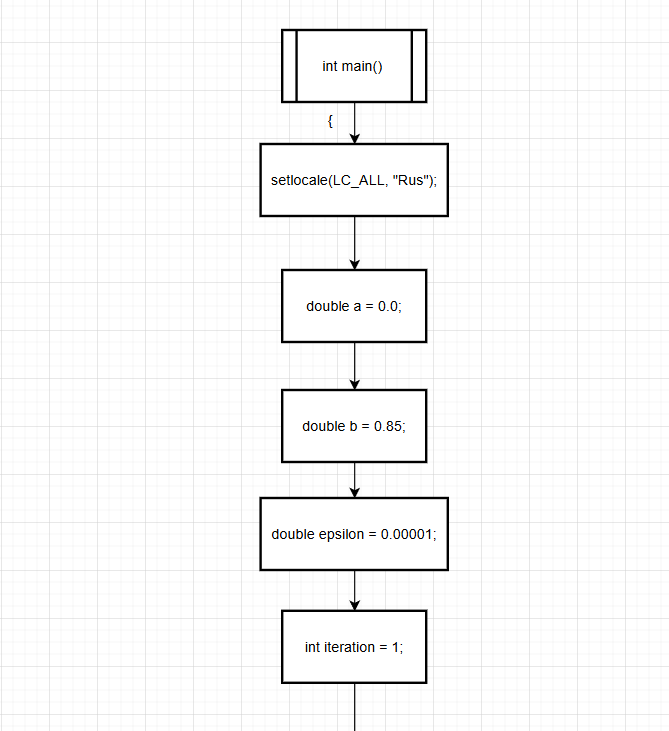


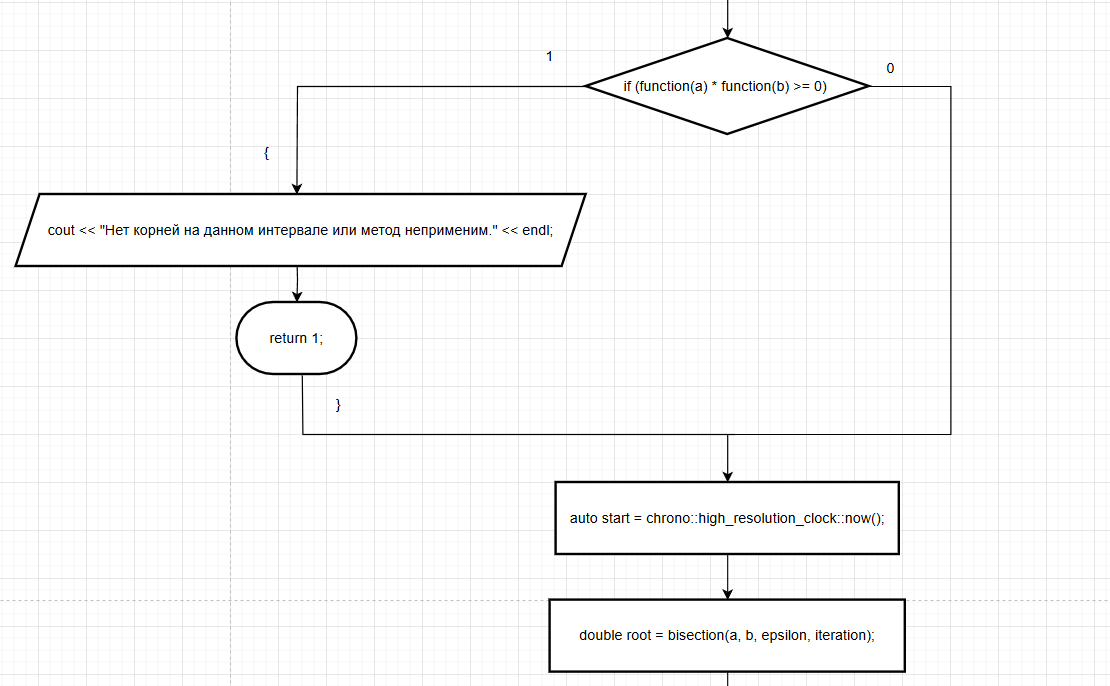


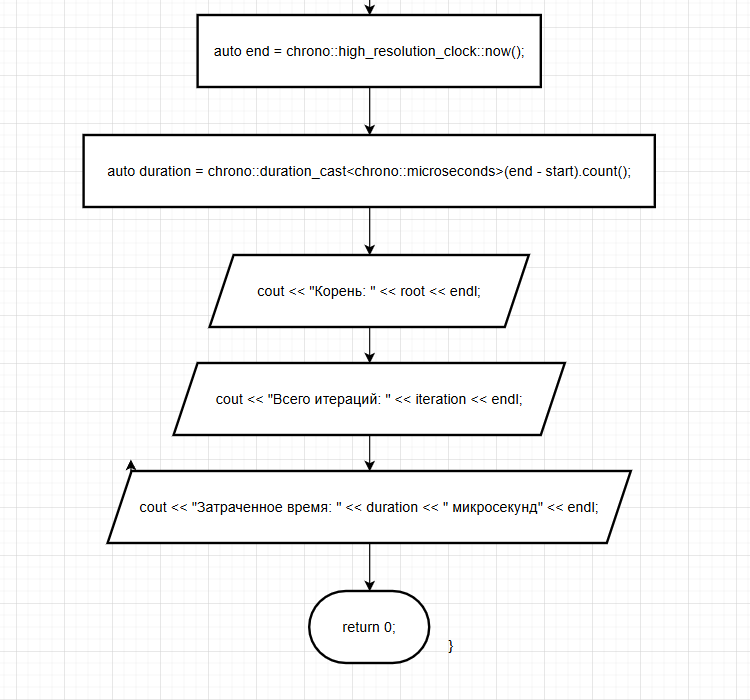
Метод Бисекции с рекурсивной функцией



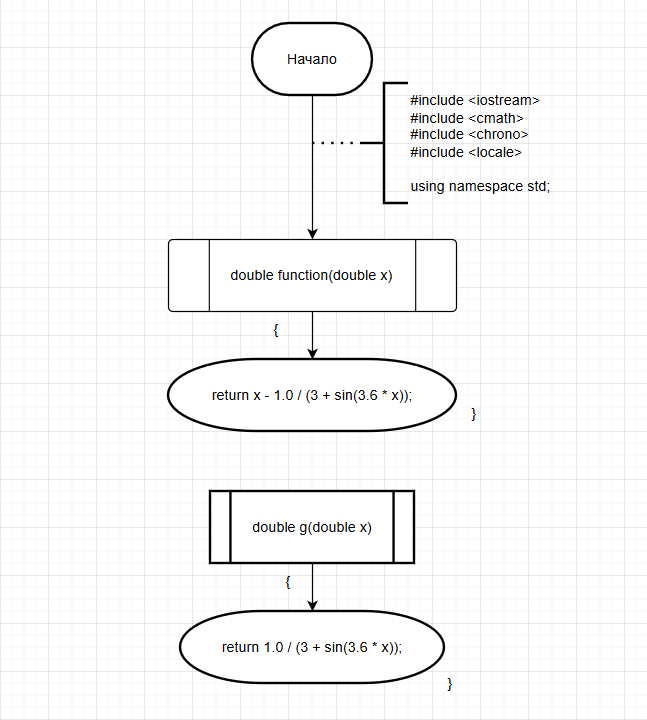


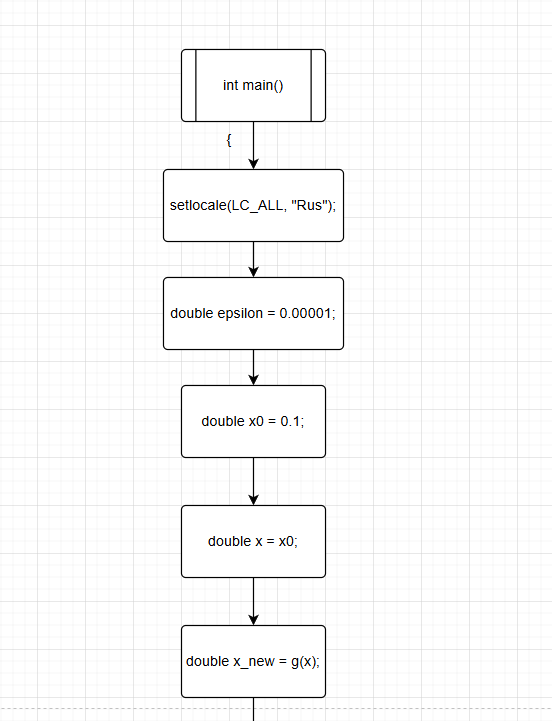


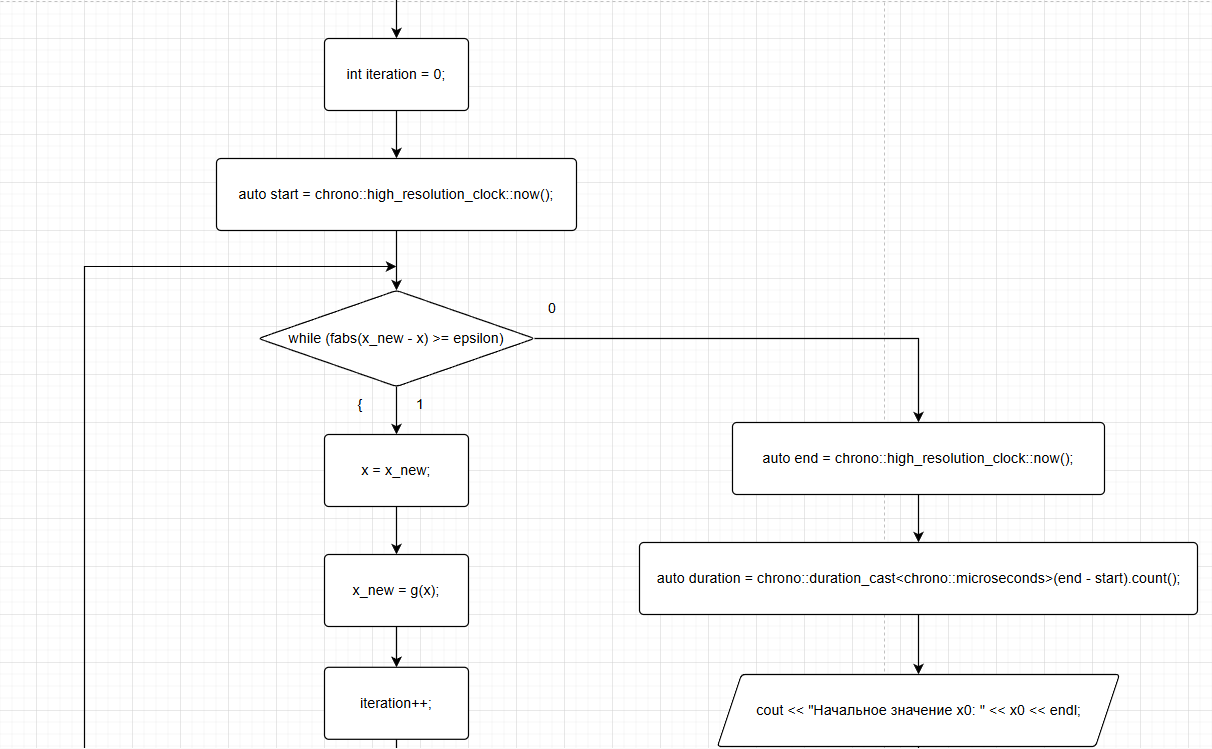


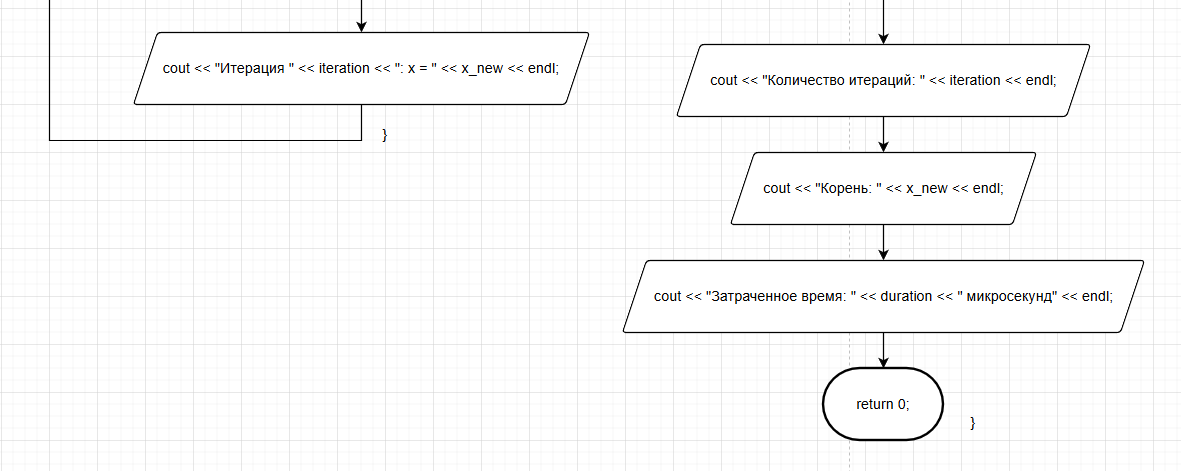


Метод Итераций









# Код программы

Метод Ньютона

#include <iostream>

#include <cmath>

#include <chrono>

#include <locale>

using namespace std;

double function(double x)

{

return x - 1.0 / (3 + sin(3.6 \* x));

}

double first\_derivative(double x)

{

double denom = 3 + sin(3.6 \* x);

return 1 + (3.6 \* cos(3.6 \* x)) / (denom \* denom);

}

double second\_derivative(double x)

{

double denom = 3 + sin(3.6 \* x);

double firstPart = (3.6 \* cos(3.6 \* x));

double secondPart = (3.6 \* -3.6 \* sin(3.6 \* x));

return secondPart / (denom \* denom) - (2 \* firstPart \* firstPart) / (denom \* denom \* denom);

}

int main()

{

setlocale(LC\_ALL, "Rus");

double epsilon = 0.00001;

double x0 = 0.0;

bool found = false;

// Поиск начального значения x0

for (x0 = 0.0; x0 <= 0.85 && !found; x0 += 0.01)

{

if (function(x0) \* second\_derivative(x0) > 0)

{

found = true;

}

}

if (!found)

{

cout << "Не удалось найти подходящее начальное значение x0 на отрезке [0; 0.85]." << endl;

return 1;

}

x0 -= 0.01; // Коррекция x0 после завершения цикла

double x = x0;

int iteration = 0;

auto start = chrono::high\_resolution\_clock::now();

while (fabs(function(x)) > epsilon)

{

x = x - function(x) / first\_derivative(x);

iteration++;

cout << "Итерация " << iteration << ": x = " << x << endl;

}

auto end = chrono::high\_resolution\_clock::now();

auto duration = chrono::duration\_cast<chrono::microseconds>(end - start).count();

cout << "Начальное значение x0: " << x0 << endl;

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

cout << "Корень: " << x << endl;

cout << "Затраченное время: " << duration << " микросекунд" << endl;

return 0;

}

Метод Бисекции

#include <iostream>

#include <cmath>

#include <chrono>

#include <locale>

using namespace std;

double function(double x)

{

return x - 1.0 / (3 + sin(3.6 \* x));

}

int main()

{

setlocale(LC\_ALL, "Rus");

double a = 0.0;

double b = 0.85;

double epsilon = 0.00001;

double x;

int iteration = 0;

if (function(a) \* function(b) >= 0)

{

cout << "Нет корней на данном интервале или метод неприменим." << endl;

return 1;

}

auto start = chrono::high\_resolution\_clock::now();

bool found = false;

while ((b - a) / 2 > epsilon && !found)

{

x = (a + b) / 2;

iteration++;

cout << "Итерация " << iteration << ": x = " << x << endl;

if (function(x) == 0.0)

{

found = true;

}

else if (function(x) \* function(a) < 0)

{

b = x;

}

else

{

a = x;

}

}

auto end = chrono::high\_resolution\_clock::now();

auto duration = chrono::duration\_cast<chrono::microseconds>(end - start).count();

x = (a + b) / 2;

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

cout << "Корень: " << x << endl;

cout << "Затраченое время: " << duration << " микросекунд" << endl;

return 0;

}

Метод Бисекции (с рекурсивной функцией)

#include <iostream>

#include <cmath>

#include <chrono>

#include <locale>

using namespace std;

double function(double x)

{

return x - 1.0 / (3 + sin(3.6 \* x));

}

double bisection(double a, double b, double epsilon, int& iteration)

{

double x = (a + b) / 2;

cout << "Итерация " << iteration << ": x = " << x << endl;

if (fabs(b - a) < epsilon || fabs(function(x)) < epsilon)

{

return x;

}

else if (function(x) \* function(a) < 0)

{

iteration++;

return bisection(a, x, epsilon, iteration);

}

else

{

iteration++;

return bisection(x, b, epsilon, iteration);

}

}

int main()

{

setlocale(LC\_ALL, "Rus");

double a = 0.0;

double b = 0.85;

double epsilon = 0.00001;

int iteration = 1;

if (function(a) \* function(b) >= 0)

{

cout << "Нет корней на данном интервале или метод неприменим." << endl;

return 1;

}

auto start = chrono::high\_resolution\_clock::now();

double root = bisection(a, b, epsilon, iteration);

auto end = chrono::high\_resolution\_clock::now();

auto duration = chrono::duration\_cast<chrono::microseconds>(end - start).count();

cout << "Корень: " << root << endl;

cout << "Вcего итераций: " << iteration << endl;

cout << "Затраченное время: " << duration << " микросекунд" << endl;

return 0;

}

Метод Итерации

#include <iostream>

#include <cmath>

#include <chrono>

#include <locale>

using namespace std;

// Определение функции

double function(double x)

{

return x - 1.0 / (3 + sin(3.6 \* x));

}

// Переписываем функцию для метода итераций

double g(double x)

{

return 1.0 / (3 + sin(3.6 \* x));

}

int main()

{

setlocale(LC\_ALL, "Rus");

double x0 = 0.1; // Начальное значение

double epsilon = 0.00001; // Точность

double x = x0;

double x\_new = g(x);

int iteration = 0;

auto start = chrono::high\_resolution\_clock::now();

while (fabs(x\_new - x) >= epsilon)

{

x = x\_new;

x\_new = g(x);

iteration++;

cout << "Итерация " << iteration << ": x = " << x\_new << endl;

}

auto end = chrono::high\_resolution\_clock::now();

auto duration = chrono::duration\_cast<chrono::microseconds>(end - start).count();

cout << "Начальное значение x0: " << x0 << endl;

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

cout << "Корень: " << x\_new << endl;

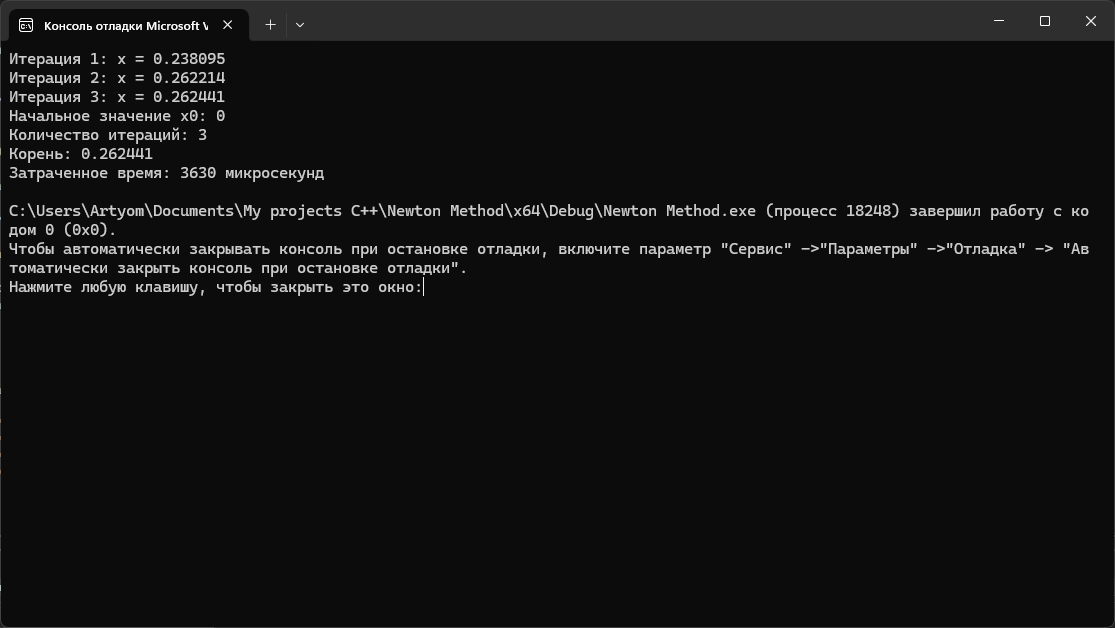
cout << "Затраченное время: " << duration << " микросекунд" << endl;

return 0;

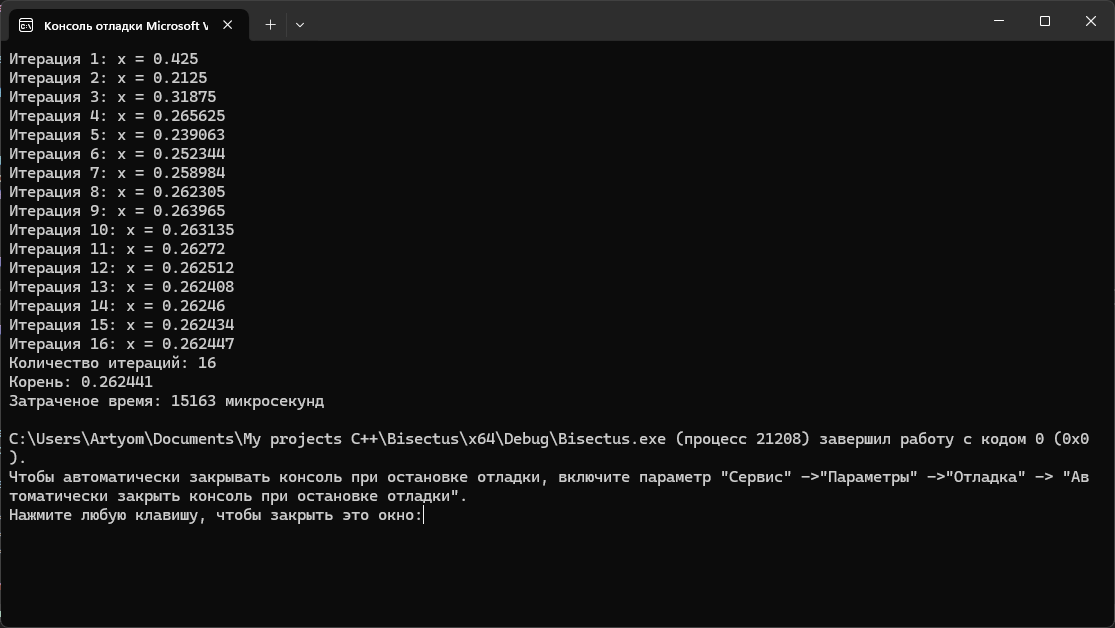
}

# Результаты

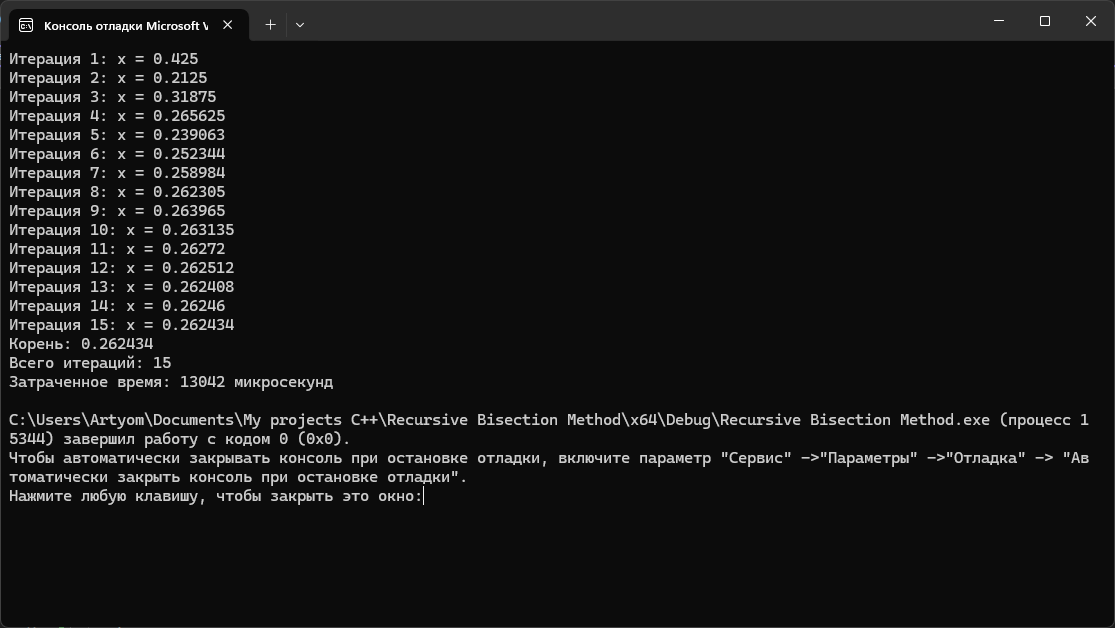
Метод Ньютона



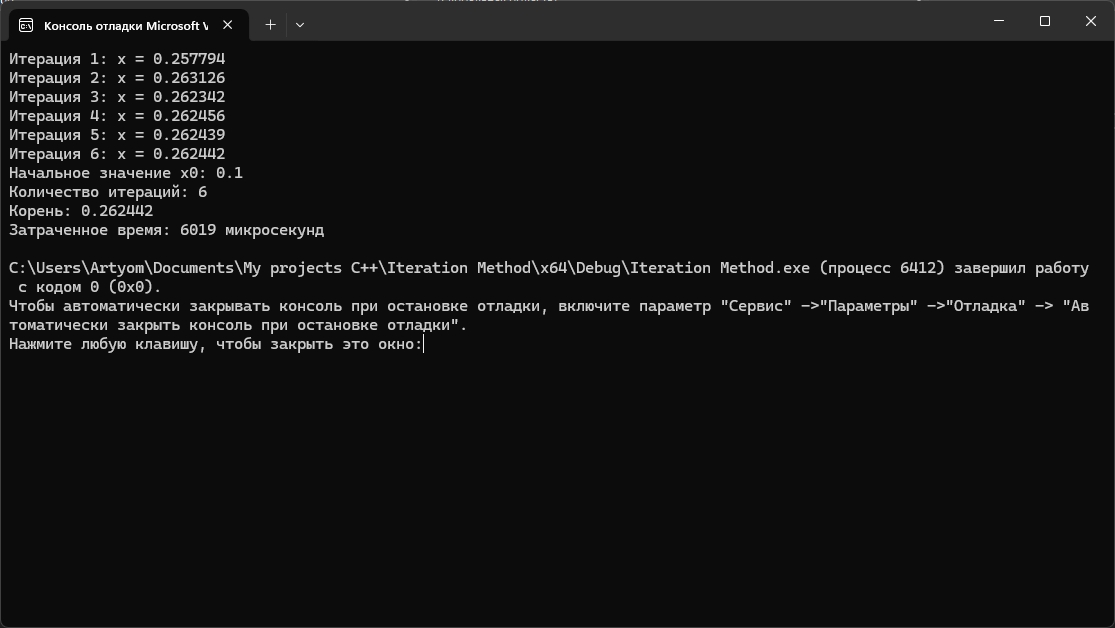
Метод Бисекции



Метод Бисекции с рекурсией



Метод Итерации



Ссылка на GitHub: https://github.com/Artivaa