

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: 09.04.01- «Программная инженерия»

**Лабораторная работа №2
«Решение нелинейных уравнений»**

Выполнил студент гр. РИС-24-36

Ушаков Арсений Анатольевич

Проверил:

Доц. Каф. ИТАС

Ольга Андреевна Полякова

г. Пермь, 2024

Постановка задачи.

Разработать алгоритм и написать код на языке C++ для решения данного уравнения: $x - \frac{1}{3 + \sin(3.6x)} = 0$.

Вариант 9.

Задано нелинейное уравнение $x - \frac{1}{3 + \sin(3.6x)} = 0$, отрезок $[0; 0.85]$, содержащий корень 0.2624. Точность вычислений $\text{eps} = 10^{-6}$.

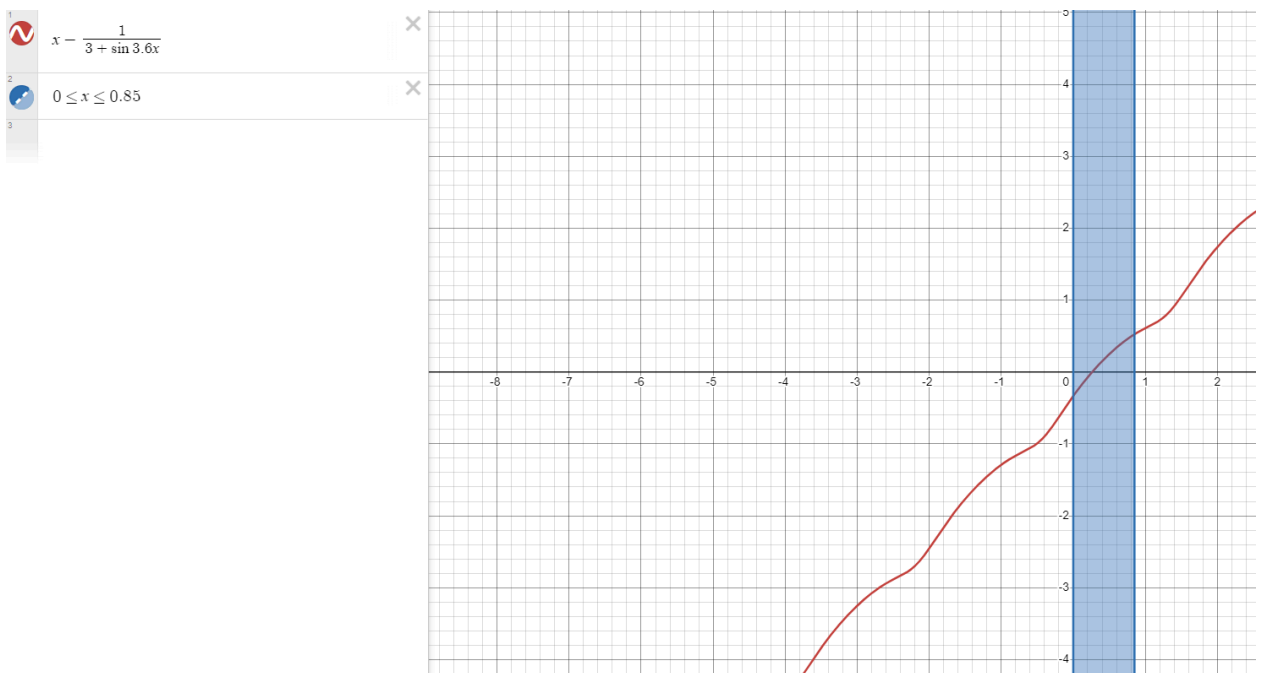


Рис. 1 - График уравнения

Метод Ньютона

1. Обозначим функцию $f(x) = x - \frac{1}{3 + \sin(3.6x)}$
2. Найдем первую производную от функции: $f'(x) = 1 - \frac{18\cos(\frac{18x}{5})}{5(3 + \sin(\frac{18x}{5}))^2}$
3. Найдем вторую производную от функции:

$$f''(x) = \frac{-3240\sin(\frac{18x}{5}) + 1944 - 324\sin(\frac{18x}{5})\cos(\frac{18x}{5})^2}{25(3 - \sin(\frac{18x}{5}))^4}$$

Если для интервала $[a; b]$ выполняется $f(a) * f''(a) > 0$ или $f(b) * f''(b) > 0$, то функция монотонна и непрерывна, и корень на интервале существует, иначе корня на интервале не существует.

Примем $x_0 = b$, через точку $(x_0; f(x_0))$ проведем касательную к графику функции. Приближенным значением корня x_1 будет пересечение касательной с осью Ox . Следующее значение вычисляется по формуле: $x_i = x_{i-1} - \frac{f(x_{i-1})}{f'(x_{i-1})}$

.

Пока $\text{eps} \leq |x_0 - x_i|$, проводим новые касательные и получаем новые приближенные значения корня. Когда это условие выполнено - найдено точное решение уравнения.

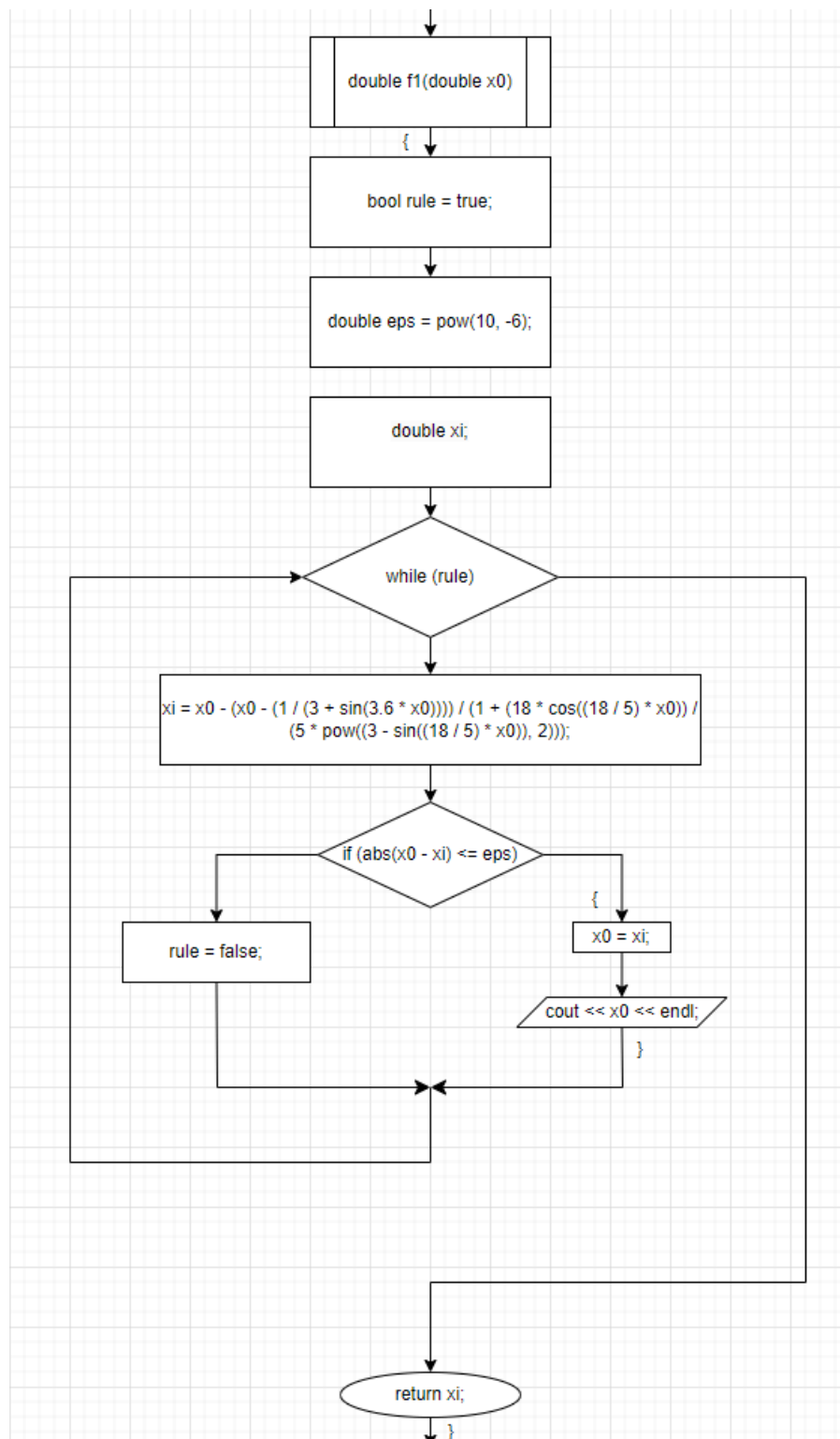


Рис. 2 - Блок-схема дополнительной функции. Метод Ньютона

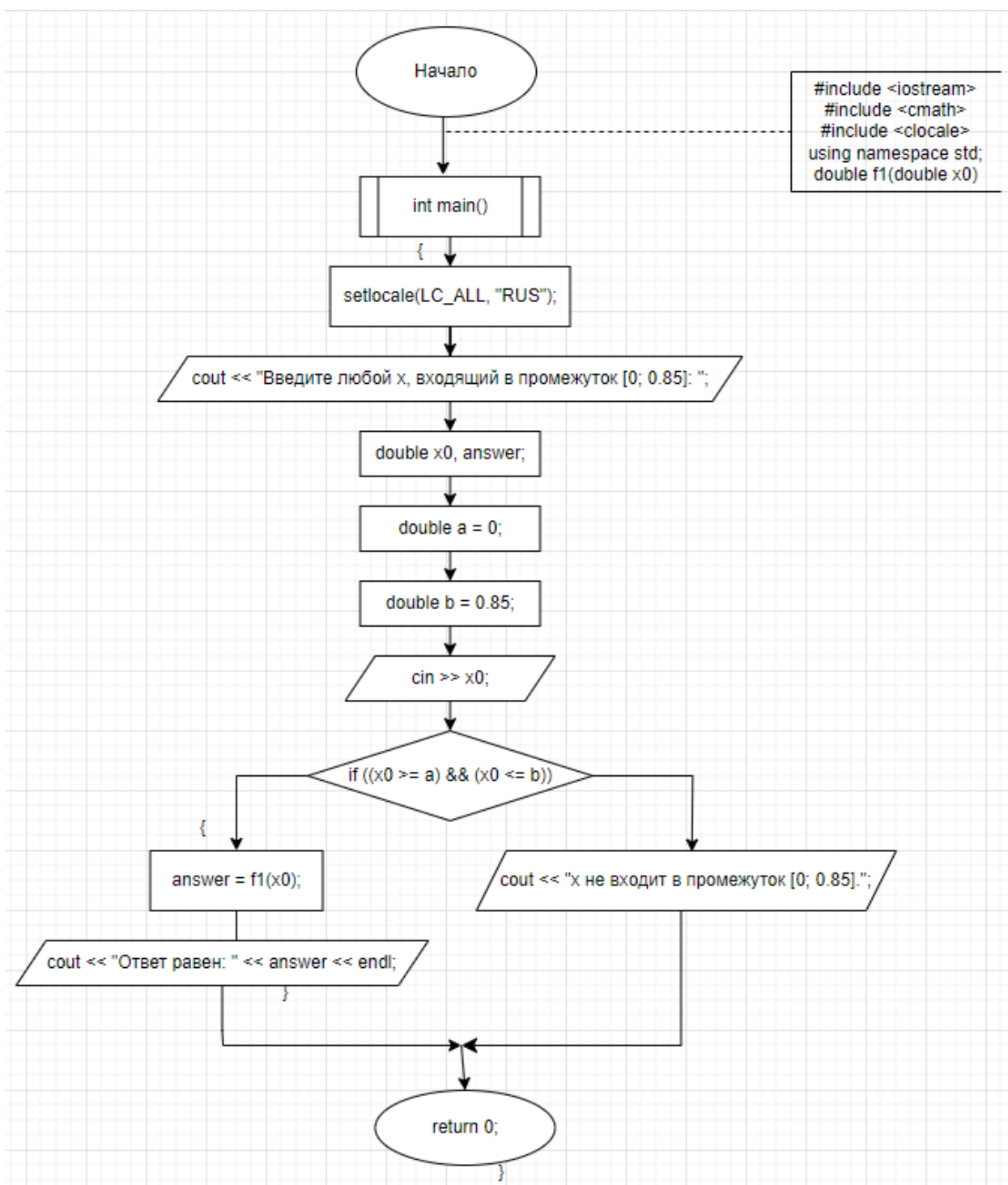


Рис. 3 - блок-схема функции main. Метод Ньютона

```

1 #include <iostream>
2 #include <cmath>
3 #include <locale>
4 using namespace std;
5 double f1(double x0)
6 {
7     bool rule = true;
8     double eps = pow(10, -6); //Точность вычисления
9     double xi;
10    while (rule) {
11        xi = x0 - (x0 - (1 / (3 + sin(3.6 * x0)))) / (1 + (18 * cos((18 / 5) * x0)) / (5 * pow((3 - sin((18 / 5) * x0)), 2))); //Касательная к графику
12        if (abs(x0 - xi) <= eps) //Проверка промежуточного корня
13        {
14            rule = false; //Цикл завершается
15        }
16        else {
17            x0 = xi; //Присваиваем новое значение, чтобы найти следующее приближенное значение корня
18            cout << "Промежуточное значение = " << x0 << endl; //Вывод промежуточного значения
19        }
20    }
21    return xi;
22 }
23 int main()
24 {
25     setlocale(LC_ALL, "RUS");
26     cout << "Введите любой x, входящий в промежуток [0; 0.85]: ";
27     double x0, answer;
28     double a = 0;
29     double b = 0.85;
30     cin >> x0;
31     if ((x0 >= a) && (x0 <= b)) { //Проверяем входит ли x в промежуток
32         answer = f1(x0);
33         cout << "Ответ равен: " << answer << endl;
34     }
35     else {
36         cout << "x не входит в промежуток [0; 0.85].";
37     }
38     return 0;
39 }

```

Рис. 4 - Программная реализация метода Ньютона

```

C:\> Консоль отладки Microsoft Visual Studio
Введите любой x, входящий в промежуток [0; 0.85]: 0.5
Промежуточное значение = 0.266475
Промежуточное значение = 0.263362
Промежуточное значение = 0.262651
Промежуточное значение = 0.262489
Промежуточное значение = 0.262452
Промежуточное значение = 0.262444
Промежуточное значение = 0.262442
Ответ равен: 0.262442

```

Рис. 5 - Результат выполнения программы

Найденное значение корня равно 0.2624.

Метод половинчатого деления.

Если выполняется условие $f(a) * f(b) < 0$, то график функции пересекает ось Ох в интервале $[a; b]$. Делим интервал пополам, полученная на половине точка x_0 считается приближенным значением корня.

Отбрасываем половину, в которой не содержится корня. Если выполняется условие $f(a) * f(x_0) < 0$, то правая граница интервала переносится в точку x_0 , иначе левая граница интервала переносится в точку x_0 .

Продолжаем делить интервал и отсекай ненужную половину, пока не выполнится условие $|a - b| < eps$, тогда приближенным значением корня будет являться любая граница интервала.

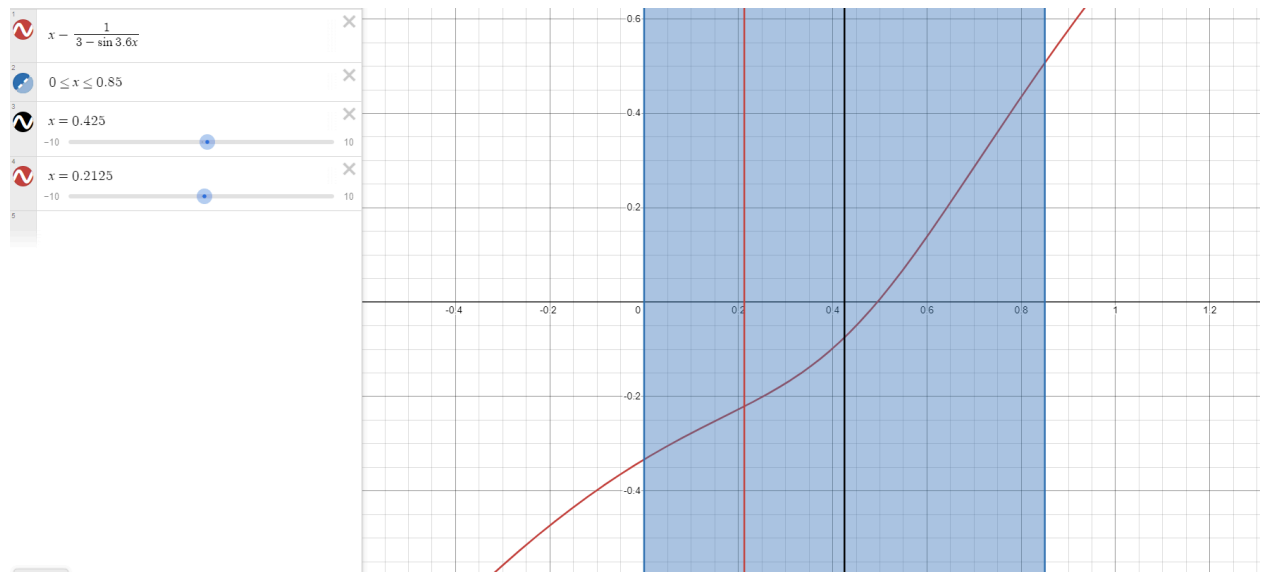


Рис. 6 - Геометрическая интерпретация метода половинчатого деления

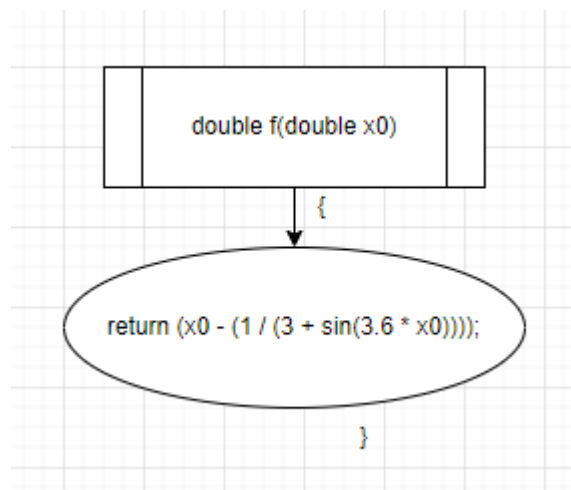


Рис. 7 - Блок-схема первой дополнительной функции.

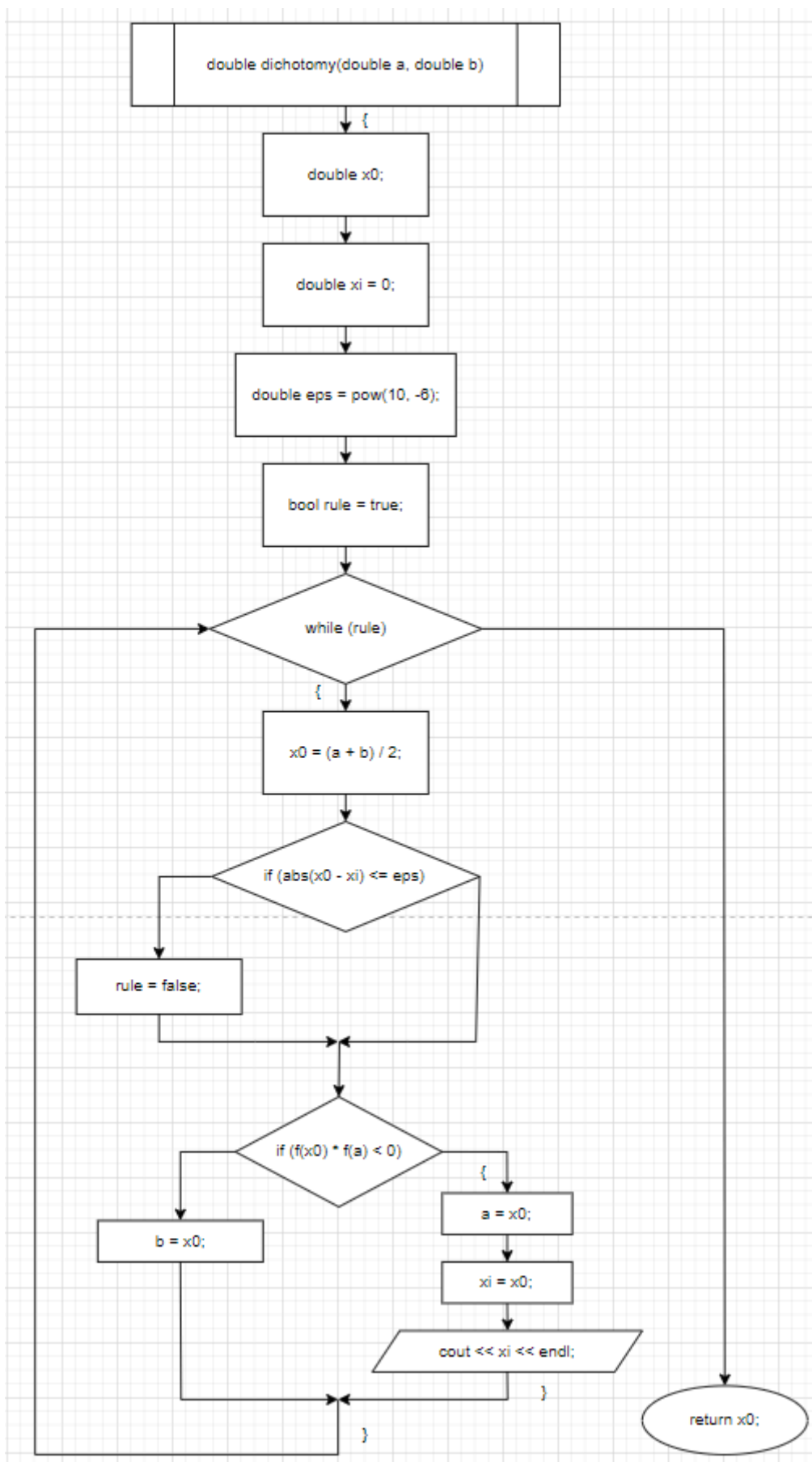


Рис. 8 - Блок-схема второй дополнительной функции.

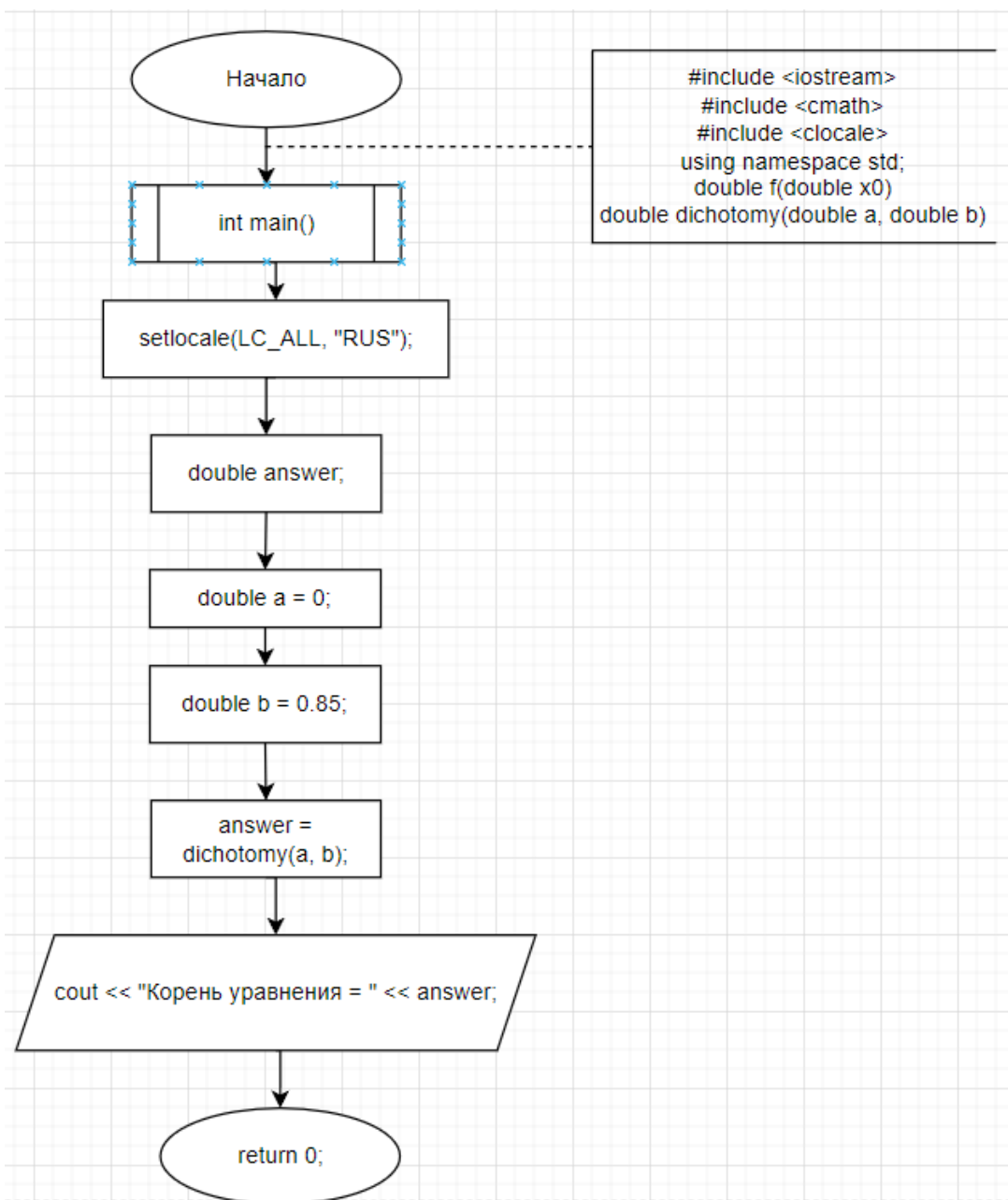


Рис. 9 - Блок-схема функции `main`.

```

1  #include <iostream>
2  #include <cmath>
3  #include <locale>
4  using namespace std;
5
6  double f(double x0) //Функция нелинейного уравнения
7  {
8      return (x0 - (1 / (3 + sin(3.6 * x0))));
9  }
10 double dichotomy(double a, double b)
11 {
12     double x0;
13     double xi = 0;
14     double eps = pow(10, -6); //Точность решения
15     bool rule = true;
16     while (rule)
17     {
18         x0 = (a + b) / 2;
19         if (abs(x0 - xi) <= eps) //Проверка промежуточного значения корня
20         {
21             rule = false; //Завершаем цикл половинчатого деления
22         }
23         if (f(x0) * f(a) < 0) //Проверка на какой половине есть корень
24         {
25             b = x0;
26         }
27         else
28         {
29             a = x0;
30             xi = x0;
31             cout << "Промежуточное значение = " << xi << endl; //Вывод промежуточного значения
32         }
33     }
34     return x0;
35 }
36 int main()
37 {
38     setlocale(LC_ALL, "RUS");
39     double answer;
40     double a = 0;
41     double b = 0.85;
42     answer = dichotomy(a, b);
43     cout << "Корень уравнения = " << answer;
44     return 0;
45 }

```

Рис. 10 - Программная реализация метода половинчатого деления.

```

Консоль отладки Microsoft Visual Studio
Промежуточное значение = 0.2125
Промежуточное значение = 0.239063
Промежуточное значение = 0.252344
Промежуточное значение = 0.258984
Промежуточное значение = 0.262305
Промежуточное значение = 0.262408
Промежуточное значение = 0.262434
Промежуточное значение = 0.262441
Корень уравнения = 0.262442

```

Рис. 11 - Результат выполнения программы.

Найдено значение корня 0.2624.

Метод итераций.

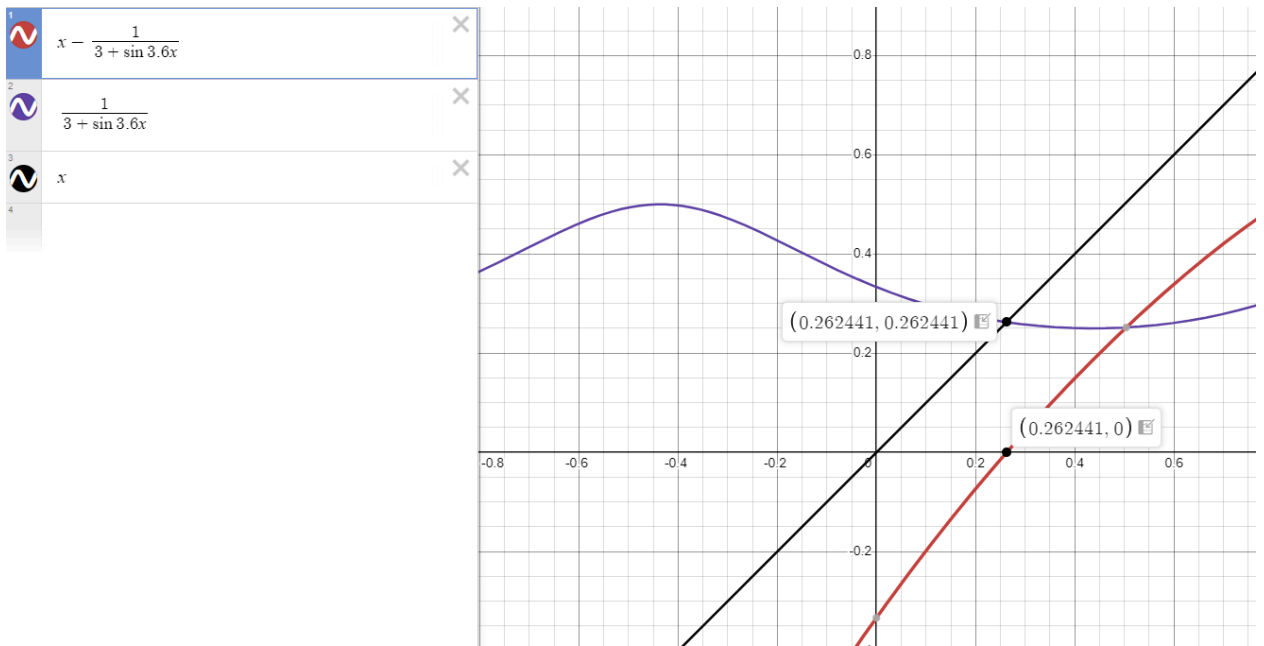


Рис. 12 - Графики метода итераций

Выражаем вспомогательную функцию $x=\varphi(x)$, $x = \frac{1}{3+\sin(3.6x)}$ (график фиолетового цвета на рисунке 12).

Находим производную от вспомогательной функции

$$\varphi'(x) = \frac{-3.6 * \cos(3.6x)}{(3 - \sin(3.6x))^2} \text{ и проверим условие сходимости}$$

$$|\varphi'(0)| < 1. \quad \varphi'(0) = \left| \frac{-3.6 * \cos(3.6 * 0)}{(3 - \sin(3.6 * 0))^2} \right| = \left| \frac{-3.6}{9} \right| < 1$$

Следовательно, эта вспомогательная функция подходит.

Примем за начальное значение x_0 левую границу интервала 0. Следующее значение $x_1 = \varphi(0)$. Вычисляем следующие значения x по формуле $x_i = \varphi(x_{i-1})$ до тех пор, пока модуль разности двух соседних значений x не будет меньше ϵ_{ps} .

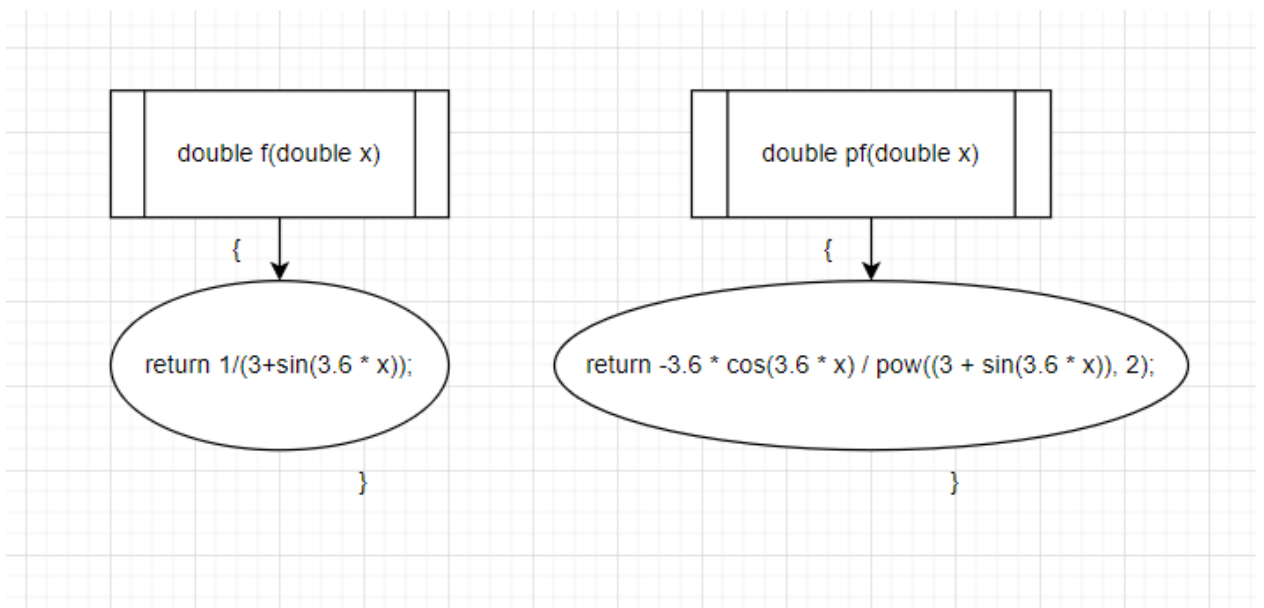


Рис. 13 - Блок-схемы дополнительных функций

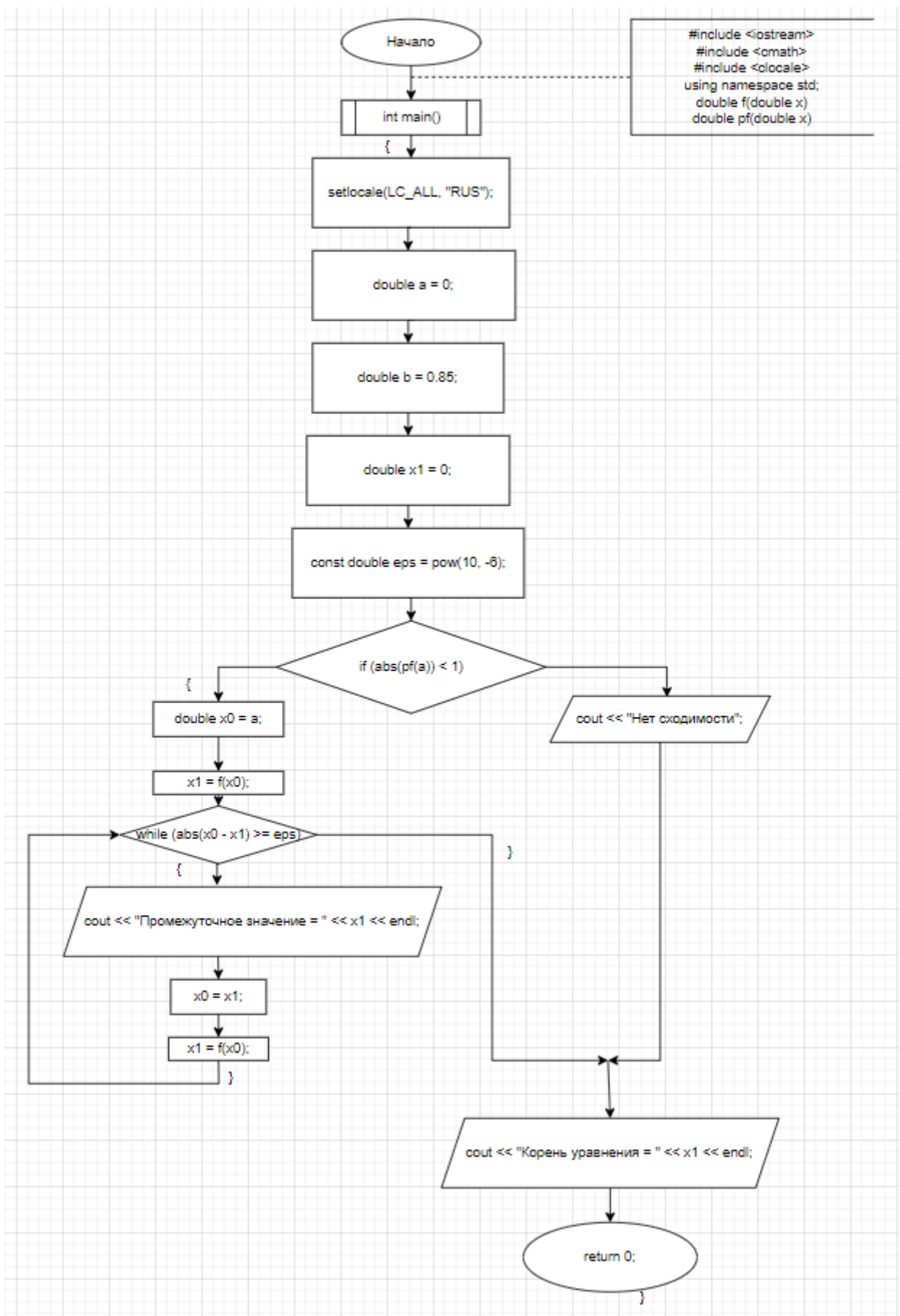


Рис. 14 - Блок-схема функции `main`

```
iteracii (Глобальная область)
1  #include <iostream>
2  #include <cmath>
3  #include <locale>
4  using namespace std;
5  double f(double x)
6  {
7      return 1 / (3 + sin(3.6 * x)); //Вспомогательная функция фи(x)=x
8  }
9  double pf(double x)
10 {
11     return -3.6 * cos(3.6 * x) / pow((3 + sin(3.6 * x)), 2); //Производная вспомогательной функции
12 }
13 int main()
14 {
15     setlocale(LC_ALL, "RUS");
16     double a = 0;
17     double b = 0.85;
18     double x1 = 0;
19     const double eps = pow(10, -6);
20     if (abs(pf(a)) < 1) //Проверка сходимости
21     {
22         double x0 = a;
23         x1 = f(x0);
24         while (abs(x0 - x1) >= eps) //Проверка на точность
25         {
26             cout << "Промежуточное значение = " << x1 << endl;
27             x0 = x1;
28             x1 = f(x0);
29         }
30     }
31     else {
32         cout << "Нет сходимости";
33     }
34     cout << "Корень уравнения = " << x1 << endl;
35     return 0;
36 }
```

Рис. 15 - Программная реализация метода итераций

```
Консоль отладки Microsoft Visual Studio
Промежуточное значение = 0.333333
Промежуточное значение = 0.254321
Промежуточное значение = 0.26365
Промежуточное значение = 0.262266
Промежуточное значение = 0.262467
Промежуточное значение = 0.262438
Промежуточное значение = 0.262442
Корень уравнения = 0.262441
```

Рис. 16 - Результат выполнения программы

Ссылка на github: <https://github.com/ArseniyUshakov/laboratories>