|  |  |  |  |
| --- | --- | --- | --- |
| Толкованов  Николай  Романович | Лаб. Раб. №1 | Выполнение: |  |
| Нахождение корня методом дихотомии | Зачет: |  |

**Задача:** получить приближённый корень уравнения с помощью метода дихотомии.

**Теория:**

Считаем, что отделение корней уравнения проведено и на отрезке [a, b] расположен один корень, который необходимо уточнить с погрешностью ε.

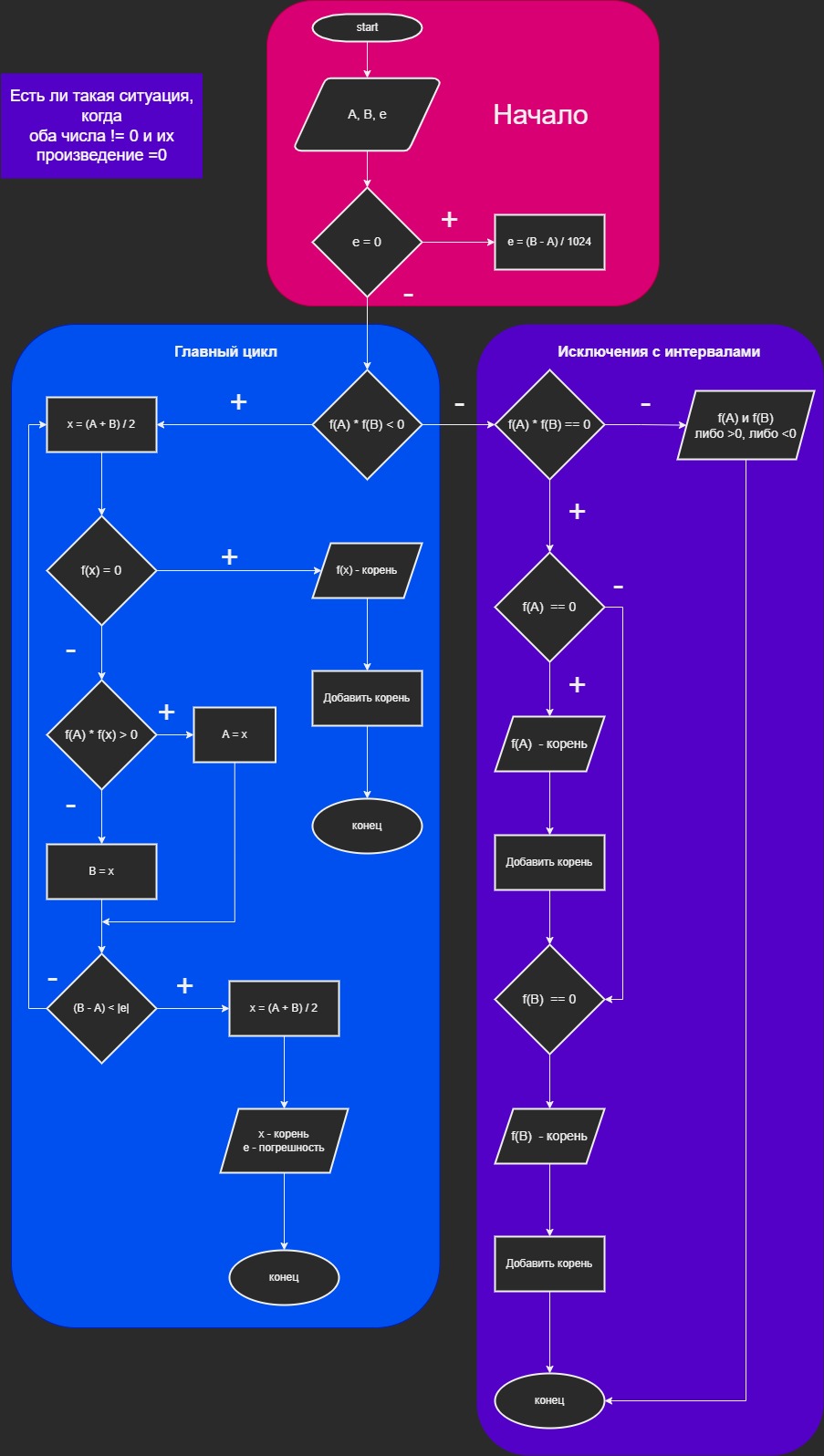
Метод дихотомии, или половинного деления заключается в следующем. Определяем середину отрезка [a, b] и вычисляем функцию f (x) . Далее делаем выбор, какую из двух частей отрезка взять для дальнейшего уточнения корня. Если левая часть уравнения f (x) есть непрерывная функция аргумента x , то корень будет находиться в той половине отрезка, на концах которой f (x) имеет разные знаки. Для очередного шага уточнения точку a перемещаем в середину отрезка и продолжаем процесс деления как с первоначальным отрезком [a, b].

Итерационный (повторяющийся) процесс будем продолжать до тех пор, пока интервал [a, b] не станет меньше заданной погрешности ε .

**Таблица:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **ε** | **k(теор)** | **n(реал)** | **x** | **f(x)** | **tсек** |
| 0.01 | 4 | 3 | 1.1375 | 0.0649513 | 0.001 |
| 0.001 | 7 | 6 | 1.12969 | 0.00619292 | 0.002 |
| 0.0001 | 10 | 9 | 1.12871 | -0.00116571 | 0.002 |

**Блок-схема:**

****

**Код программы:**

#include <iostream>

#include <cmath>

#include <ctime>

using namespace std;

*//Функция для полинома:*

double myFunction(double x)

{

*// return sin(x);*

    return 4 \* (1 + pow(x, 0.5)) \* log(x) - 1;

}

*//Функция поиска корня:*

double findRoot(double (\*f)(double), double a, double b, double eps)

{

    double x, seconds;

    int steps = 0;

    while ((b - a) / 2 > eps)

    {

        steps++;

        x = (a + b) / 2;

        if ((f(a) \* f(x)) > 0)

            a = x;

        else

            b = x;

    }

    cout << "x: " << x << " steps: " << steps << endl;

    steps = 0;

    return x;

}

double teoreticSteps(double a, double b, double eps) {

    double k = (log(b - a) - log(eps)) / log(2) + 1;

    return floor(k);

}

int main()

{

*//Интервал, погрешность и корень:*

    double a, b, eps = 0.000001;

    double x\_array[100], y\_array[100], intervals;

    double counter = 0.1;

    int start\_time, end\_time;

*// Заполняем массивы:*

    for (int i = 0; i < 100; i++) {

        x\_array[i] = counter;

        y\_array[i] = myFunction(counter);

        counter += 0.1;

    }

*// Ищем корни в интервалах:*

    for (int j = 1; j < sizeof(y\_array) / sizeof(y\_array[0]); j++) {

        if (y\_array[j - 1] \* y\_array[j] < 0) {

            a = x\_array[j - 1];

            b = x\_array[j];

            start\_time = clock();

            findRoot(myFunction, a, b, eps);

            end\_time = clock() - start\_time;

            cout << "time: " << (float)end\_time / CLOCKS\_PER\_SEC << endl;

            cout << "teoretic steps: " << teoreticSteps(a, b, eps) << "\n\n" << endl;

        }

    }

}

**Заключение:** мы получили приближённый корень уравнения с помощью метода дихотомии.