|  |  |  |  |
| --- | --- | --- | --- |
| Толкованов  Николай  Романович | Лаб. Раб. №3 | Выполнение: |  |
| Нахождение корня методом ньютона | Зачет: |  |

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

**Теория:**

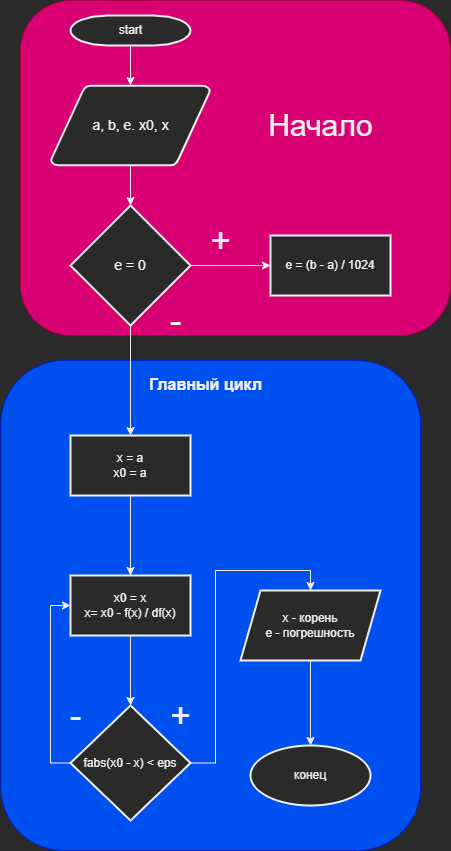
Предположим, что графическим методом определено начальное приближение 0x к   
корню. В точке 0x вычислим левую часть решаемого уравнения , а также   
производную в этой точке. Следующее приближение к корню найдем в точке 1x, где касательная. к функции, проведенная из точки x0, f0 пересекает ось абсцисс.

Затем считаем точку x1 в качестве начальной и продолжаем итерационный процесс. Процесс уточнения корня закончим, когда выполнится условие|xk+1 - xk| < ε,   
где ε - допустимая погрешность определения корня.

**Таблица:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **ε** | **k(теор)** | **n(реал)** | **x** | **f(x)** | **tсек** |
| 0.01 | 4 | 2 | 1.2 | 0.528179 | 0 |
| 0.001 | 7 | 2 | 1.2 | 0.528179 | 0.003 |
| 0.0001 | 10 | 3 | 1.2 | 0.528179 | 0.002 |

**Блок-схема:**



**Код программы:**

#include <iostream>

#include <cmath>

#include <ctime>

using namespace std;

double myFunction(double x)

{

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

}

double myPrFunction(double x) {

    return 2 \* (log(x) / pow(x, 0.5)) + ((4 \* pow(x, 0.5) + 4) / x);

}

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

{

    double x = a, x0 = a;

    int steps = 0;

    do

    {

        steps++;

        x0 = x;

        x = x0 - f(x) / df(x);

    }

    while (fabs(x0 - x) > eps);

    cout << "x: " << b << endl;

    cout << "f(x): " << f(b) << endl;

    cout << "eps: " << eps << endl;

    cout << "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()

{

    double a, b, eps = 0.0001;

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

    double counter = 0.1;

    int start\_time, end\_time;

    for (int i = 0; i < 100; i++)

    {

        x\_array[i] = counter;

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

        counter += 0.1;

    }

    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];

*// cout << "INTERVALS: A: " << a << " B: " << b << endl;*

            start\_time = clock();

            findRoot(myFunction, myPrFunction, 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 ньютона.