Листинг

```
#include <conio.h> // getch
#include <iostream> // ввод вывод
#include <cmath> // математика

using namespace std; // чтобы не писать каждый раз std


class PosException { // класс исключения на знаки функций на концах
    public:
        void what() {
            cout << "f(a)*f(b) > 0\n";
        }
};

class CorrectDataExc { // исключение на некорректные данные
public:
    void what() {
        cout << "Data isn't correct\n";
    }
};

class NonlinearEquation { // родительский класс со всеми полями методами и тд
    double a; // коэффициент перед X
    double intervalStart; // начало интервала
    double intervalEnd; // конец интервала
    double epsilon; // точность

public:
    NonlinearEquation(double coefficient, double start, double end, double eps) //
конструктор с параметрами
        : a(coefficient), intervalStart(start), intervalEnd(end), epsilon(eps) {
    }

    ~NonlinearEquation() // деструктор без параметров
    {

    }

    double getA() const { // вызываем и получаем коэффициент
        return a;
    }

    double getIntervalStart() const { // вызываем и получаем начало отрезка
        return intervalStart;
    }

    double getIntervalEnd() const { //вызываем и получаем конец отрезка
        return intervalEnd;
    }

    double getEpsilon() const { // вызываем и получаем точность
        return epsilon;
    }

    double calculateEquationValue(double x) { // вызываем и получаем функцию
        return getA() * x - cos(x);
```

```
    }

    double calculateEquationDerivative(double x) { // вызываем и получаем первую производную
        return getA() + sin(x);
    }

    double solveByBisectionMethod() { // метод деления отрезка пополам
        double start = getIntervalStart(); // обозначаем старт
        double end = getIntervalEnd(); // обозначаем конец отрезка
        double root = 0.0; // наш корень

        if (fabs(getA()) >= 20 || fabs(getIntervalStart()) >= 20 || fabs(getIntervalEnd())
>= 20 || (getEpsilon() <= 0 && getEpsilon() >= 1)) throw CorrectDataExc(); // проверка, что
данные корректны иначе исключение
        else if (calculateEquationValue(start) * calculateEquationValue(end) > 0) throw
PosException(); // проверка на знак функции на концах отрезка иначе исключение
        else // иначе)
        {
            while ((end - start) > getEpsilon()) { // пока границы отрезка больше точности
крутим цикл
                double mid = (start + end) / 2.0; // находим середину отрезка

                if (calculateEquationValue(mid) < getEpsilon()) { // если значение функции
середины меньше точности, то говорим, что это корень и выходим из цикла
                    root = mid;
                    break;
                }
                else if (calculateEquationValue(mid) * calculateEquationValue(start) < 0) {
// если значения функции на концах разного знака, то сдвигаем одну часть отрезка
                    end = mid;
                }
                else { // иначе двигаем другую часть отрезка
                    start = mid;
                }
            }

            if (root != 0.0) { // если корень хоть раз изменился, то возвращаем его
                return root;
            }
            else {
                throw std::runtime_error("Equation does not have a root in the given
interval"); // иначе кидаем исключение
            }
        }
    }
};

class NewtonMethodSolver : public NonlinearEquation { // дочерний класс
public:
    NewtonMethodSolver(double coefficient, double start, double end, double eps) //
конструктор с параметрами
        : NonlinearEquation(coefficient, start, end, eps) {
    }

    ~NewtonMethodSolver() // деструктор без параметров
    {

    }

    double solveByNewtonMethod() { // решение методом ньютона
```

```
        if (fabs(getA()) >= 20 || fabs(getIntervalStart()) >= 20 || fabs(getIntervalEnd())
>= 20 || (getEpsilon() <= 0 && getEpsilon() >= 1)) // все как и в прошлом случае
        {
            throw CorrectDataExc();
        }
        double root = (getIntervalStart() + getIntervalEnd()) / 2.0;  // опять находим центр
        if (calculateEquationValue(getIntervalStart()) *
calculateEquationValue(getIntervalEnd()) > 0) // проверка на знак
            throw PosException();
        else
        {
            while (std::abs(calculateEquationValue(root)) > getEpsilon()) { // пока функция
больше точности
                root = root - calculateEquationValue(root) /
calculateEquationDerivative(root); // кайфуем с математики
            }
            return root; // возвращаем ответ
        }

    }
};

int main() { // главная функция

    double coefficient; // я не буду это комментировать...
    double Start;
    double End;
    double eps;

    int n=1; // переменная для case

    while (n != 0)
    {
        cout << "a*x - cos(x) = 0\n";
        cout << "Choose your fighter:\n";
        cout << "1. Bisection method\n";
        cout << "2. Newton method\n";
        cout << "0. Exit!\n";
        cin >> n;
        system("cls"); // чистим экран
        switch (n)
        {
        case 1:

        {
            cout << "a*x - cos(x) = 0\n";
            cout << "Bisection method\n";
            cout << "Enter the coef, limits of interval and epsilon:\n";
            cin >> coefficient >> Start >> End >> eps; // вводим параметры

            NonlinearEquation equation(coefficient, Start, End, eps); // оформляем
конструктор
            try {
                double root = equation.solveByBisectionMethod(); // получаем корень
                system("cls");
                cout << "Bisection method\n";
                cout << equation.getA() << " * x - cos(x) = 0 \ninterval: [" <<
equation.getIntervalStart() << ";" << equation.getIntervalEnd() << "] \nepsilon = " <<
equation.getEpsilon();
                cout << "\nAnswer: x = " << root << "\n";
            }
```

```
            catch (const std::runtime_error& e) { // если корень не найден
                cout << e.what();
            }
            catch (const PosException& e) { // если проблемы со знаком
                PosException exc;
                exc.what();
            }
            catch (const CorrectDataExc& e) { // если данные не правильны
                CorrectDataExc exc;
                exc.what();
            }
        }
            _getch();
            break;
        case 2:
        {
            cout << "a*x - cos(x) = 0\n";
            cout << "Newton method\n";
            cout << "Enter the coef, limits of interval and epsilon:\n";
            cin >> coefficient >> Start >> End >> eps; // заполняем данные

            NewtonMethodSolver eq(coefficient, Start, End, eps); // конструктор
            try {
                double root = eq.solveByNewtonMethod(); // получаем корень
                system("cls");
                cout << "Newton method\n";
                cout << eq.getA() << " * x - cos(x) = 0 \ninterval: [" <<
eq.getIntervalStart() << ";" << eq.getIntervalEnd() << "] \nepsilon = " << eq.getEpsilon();
                cout << "\nAnswer: x = " << root << "\n";
            }
            catch (const PosException& e) { // исключение раз
                PosException exc;
                exc.what();
            }
            catch (const CorrectDataExc& e) { // исключение два
                CorrectDataExc exc;
                exc.what();
            }
        }
            _getch();
            break;
        case 0:
            cout << "Press F to pay respect\n"; // Это на выход
            _getch();
            break;
        default:
            cout << "Wow, sorry\n"; // это если неправильно ввел цифру
            _getch();
            break;
        }
        system("cls"); // чистим экран
    }




    return 0; // конец)
}
```