Практическая работа №2

по дисциплине «Вычислительная математика»

**«Нахождение корней нелинейного уравнения»**

студента группы M3207 Бойцова Виталия

## 

Разделим отрезок [a,b] на оси абсцисс на отрезки длиной h. Узнаем, равна ли нулю функция на краях отрезков. Если знаки противоположны применим гибридный метод Ньютона + половинного деления.

Step1: a = ai ; b = bi   
Step2: x = (a + b) / 2  
Step3: если f(x) = 0, то return x,  
если f(x) \* f(a) < 0, то b = x, иначе a = x  
Step4: если f’(x) = 0, то goto Step2   
Step5: x = x – f(x)/f’(x)  
Step6: если x не принадлежит [a, b] goto Step2  
Step7: если |f(x)/f’(x)| < eps то return x, иначе goto Step3

## main.cpp

//Boytsov programming company

//All right reserved

#include <iostream>

#include <vector>

#include <string>

#include <iomanip>

#include <cmath>

#include <algorithm>

using namespace *std*;

struct Func {

public:

Func(*vector*<double> v) : coef(v) {}

Func(*initializer\_list*<double> l) : coef(l) {}

//возвращает производную

Func derivative() {

*vector*<double> v;

for (int i = 1; i < coef.*size*(); i++)

v.*push\_back*(coef[i] \* i);

return Func(v);

}

//возвращает значение функции в точке x

double gorner(double x) {

double ans = 0;

for (int i = (int)coef.*size*() - 1; i >= 0; i--)

ans = coef[i] + ans \* x;

return ans;

}

//возвращает список корней на промежутке [a,b] с шагом разбиения h и точностью eps

*vector* <double> find\_roots(double a, double b, double h, double eps) {

*vector*<double> ans;

for (double l = a; l < b; l += h) {

if (*abs*(gorner(l)) < eps\_y) {

ans.*push\_back*(l);

continue;

}

double r = *min*(l + h, b);

if (gorner(l) \* gorner(r) < 0)

ans.*push\_back*(find\_root(l, r, eps));

}

if (*abs*(gorner(b)) < eps\_y) {

ans.*push\_back*(b);

}

return ans;

}

private:

//коэффициенты при степенях

*vector* <double> coef;

const double eps\_y = 1e-6;

//находит корень на отрезке гибридным методом Ньютона-половинного деления

double find\_root(double a, double b, double eps\_x) {

while (true) {

//step 2

double x = (a + b) / 2;

while (true) {

//step 3

if (*abs*(gorner(x)) < eps\_y)

return x;

if (gorner(x) \* gorner(a) < 0)

b = x;

else

a = x;

//step 4

if (*abs*(derivative().gorner(x)) < eps\_y)

break; //to step 2

//step 5

double delta = gorner(x) / derivative().gorner(x);

x -= delta;

//step 6

if (x < a || b < x)

break; //to step 2

//step 7

if (*abs*(delta) <= eps\_x)

return x;

else

continue; //to step 3

}

}

}

};

int main() {

Func f{ -14.766, 5.606, 2.84, 1};

for (auto root : f.find\_roots(0, 2, 0.5, 0.0002))

*cout* << root << '\n';

}