**Московский государственный технический**

**университет им. Н.Э. Баумана**

Факультет «Информатика и управление»

Кафедра ИУ5. Курс «Основы информатики»

Отчет по лабораторной работе №4

# «Нахождение корней нелинейного уравнения»

|  |  |  |
| --- | --- | --- |
| Выполнил: |  | Проверил: |
| студент группы ИУ5-11 |  | преподаватель каф. ИУ5 |
| Анцифров Никита |  | Аксёнова М.В. |
|  |  |  |
| Подпись и дата: |  | Подпись и дата: |

Москва, 2019 г.

**Постановка задачи**

1. Найти корень уравнения **x - cos(x) = 0** тремя методами: методом Ньютона, половинного деления и простой итерации с погрешностью **eps < 10^(-6)** и для каждого из трех методов определить количество шагов алгоритма
2. Повторить выполнение п.1 для **eps < 10^(-8)**
3. Выполнить п.1 для уравнения **x - 10\*cos(x) = 0**
4. Объяснить результаты

**Описание входных, выходных и вспомогательных данных**

double **eps** - переменной присваивается значение погрешности

int **i** – переменной присваивается значение счётчика внутри цикла

int **k** – переменной присваивается значение счётчика внутри цикла

double **x** – переменной присваивается значение начальной точки, которое заменяется в ходе программы на значение найденного корня уравнения

double **a** – переменной присваивается значение начала заданного интервала

double **b** – переменной присваивается значение конца заданного интервала

double **fa** – переменной присваивается значение функции в точке **a**

double **fx** – переменной присваивается значение функции в точке **x**

**Алгоритм**

Функция **main**



Функция **f\_newton**



Функция **f\_dividing**



Функция **f\_iter**



**Текст программы**

#include <iostream>

#include <iomanip>

#include "func\_proto.h"

using namespace std;

int main()

{

setlocale(LC\_ALL, "Russian");

double eps=pow(10,-6);

for (int i=1; i<=2; i++)

{

cout << "Заданная точность: " << eps << endl;

cout << "Заданное уравнение: x-cos(x)=0" << endl;

f\_newton (eps, 1);

f\_dividing (eps, 1);

f\_iter (eps, 1);

eps/=100;

}

eps\*=10000;

cout << "Заданная точность: " << eps << endl;

cout << "Заданное уравнение: x-10\*cos(x)=0" << endl;

f\_newton (eps, 10);

f\_dividing (eps, 10);

f\_iter (eps, 10);

system("pause");

return 0;

}

void f\_newton (double e, int m)

{

double x=2;

int k=0;

while (abs (x-m\*cos(x)) > e)

{

k++;

x=x-((x-m\*cos(x))/(1+m\*sin(x)));

}

cout << "I. МЕТОД НЬЮТОНА" << endl;

cout << " Корень уравнения: " << setprecision (9) << x << endl;

cout << " Количество шагов: " << k << endl;

}

void f\_dividing (double e, int m)

{

double a=0, b=2, x=0, fa=0, fx=0, k=0;

do

{

k++;

x=(a+b)/2;

fa=a-m\*cos(a);

fx=x-m\*cos(x);

if (fa\*fx>0)

{

a=x;

}

else

{

b=x;

}

}

while (abs(fx)>=e);

cout << "II. МЕТОД ПОЛОВИННОГО ДЕЛЕНИЯ" << endl;

cout << " Корень уравнения: " << setprecision (9) << x << endl;

cout << " Количество шагов: " << k << endl;

}

void f\_iter (double e, int m)

{

double x=2, k=0;

while ((abs(x-m\*cos(x))>e) && (k<1000))

{

k++;

x=x-(x-m\*cos(x));

}

cout << "III. МЕТОД ПРОСТОЙ ИТЕРАЦИИ" << endl;

cout << " Корень уравнения: " << setprecision (9) << x << endl;

cout << " Количество шагов: " << k << endl << endl;

}

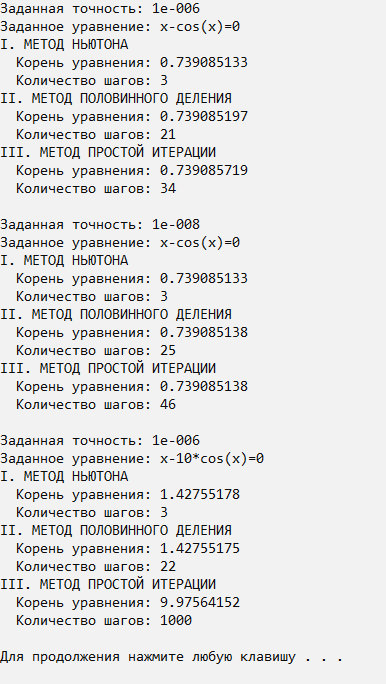
Прототипы в **f\_proto.h**

void f\_newton (double, int);

void f\_dividing (double, int);

void f\_iter (double, int);

**Анализ результатов**

****

Самый эффективный (минимальное количество шагов) метод – метод Ньютона

Самый неэффективный (наибольшее количество шагов) метод – метод итерации