**Московский государственный технический**

**университет им. Н.Э. Баумана.**

Факультет «Информатика и управление»

Кафедра «Системы обработки информации и управления»

Курс «Основы информатики»

Отчет по лабораторной работе №2

# «Численные методы решения нелинейных уравнений»

|  |  |  |
| --- | --- | --- |
| Выполнил: |  | Проверил: |
| студент группы ИУ5-13 |  | преподаватель каф. ИУ5 |
| Коновалов Максим |  | Папшев И.С |
| Подпись и дата: |  | Подпись и дата: |

Москва, 2020 г.

## Содержание отчета

Отчет по лабораторной работе состоит из 4-х разделов, отражающих основные этапы разработки программы:

- Постановка задачи;

- Разработка алгоритма;

- Кодирование (соответствующий раздел отчета называется «Текст программы»);

- Тестирование (соответствующий раздел отчета называется «Анализ результатов»).

# Постановка задачи

Разработайте программу для вычисления корней уравнения

x – k\*cos(x) = 0

при k=1 простой итерацией, половинным делением и методом Ньютона с погрешностью eps < 0.000001 и eps < 0.00000001. Для каждого из трех методов определить (и вывести на экран) количество шагов алгоритма, использованных для получения результата.

2. Выполнить п.1 для k=5 и объяснить результаты.   
Программа должна обеспечивать возможность многократного выполнения пунктов 1 и 2 для различных значений параметров:

- начального приближения *x0*,

- коэффициента *k,*

- требуемой точности вычислений *eps*.  
Для реализации этого требования используйте функцию «Меню».

3. Разработайте «универсальную» функцию для вычисления корней уравнений вида *y(x)=0* методом половинного деления, используя в качестве ***параметра указатель на функцию***.

# Разработка алгоритма

* краткое словесное описание алгоритма (для 3-х методов);

***Простая итерация***

Хj+1 = Xj - f(Xj).

***Метод Ньютона (метод касательной)***

Хj+1 = Xj - f(Xj)/ f``(Xj).

***Метод половинного деления***

Суть метода заключается в следующем. Выбирают Х на середине интервала [xl, xr] и определяют f(X). Если |f(X)| < eps, то середина интервала считается корнем уравнения, иначе корень ищется на том интервале из двух полученных, для которого значения функции на концах имеют разные знаки. Действия повторяются до тех пор, пока интервал >eps. Перед входом в цикл необходимо проверить, не являются ли границы интервала корнями уравнения f(X)=0.

* описание входных, выходных и вспомогательных данных с указанием их идентификаторов и типов;

Global

int n , eps = 0.00000001;

const int Nmax = 10000;

Local

double Xn, Xk, k;

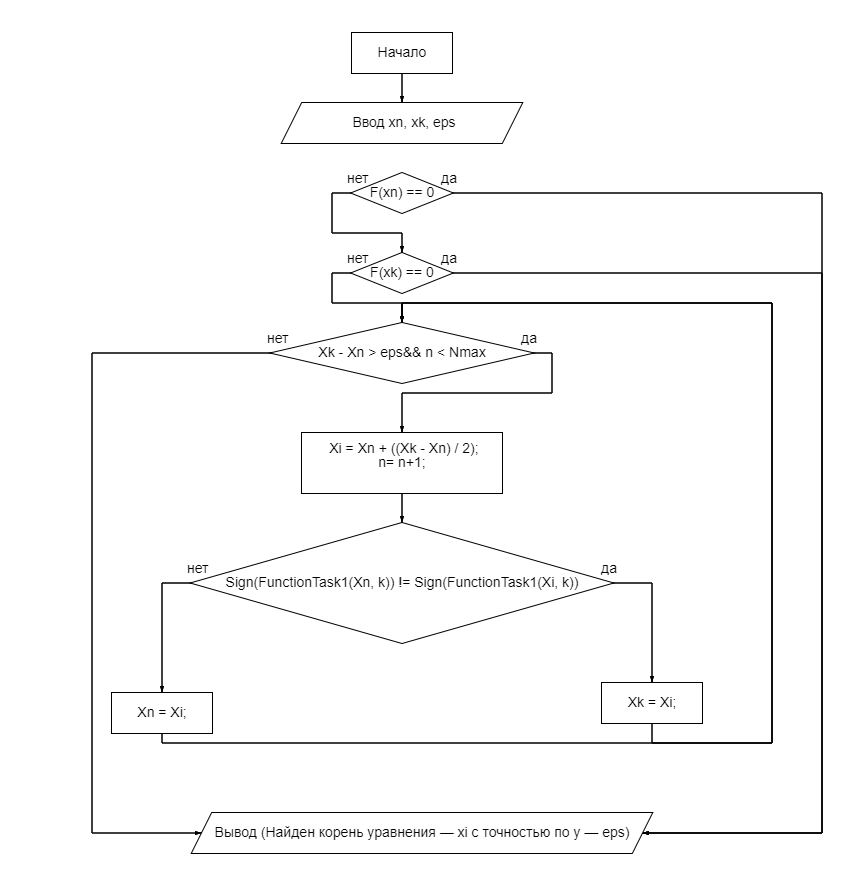
int typeOfTask; // тип задачи

double next = x0;// - след член

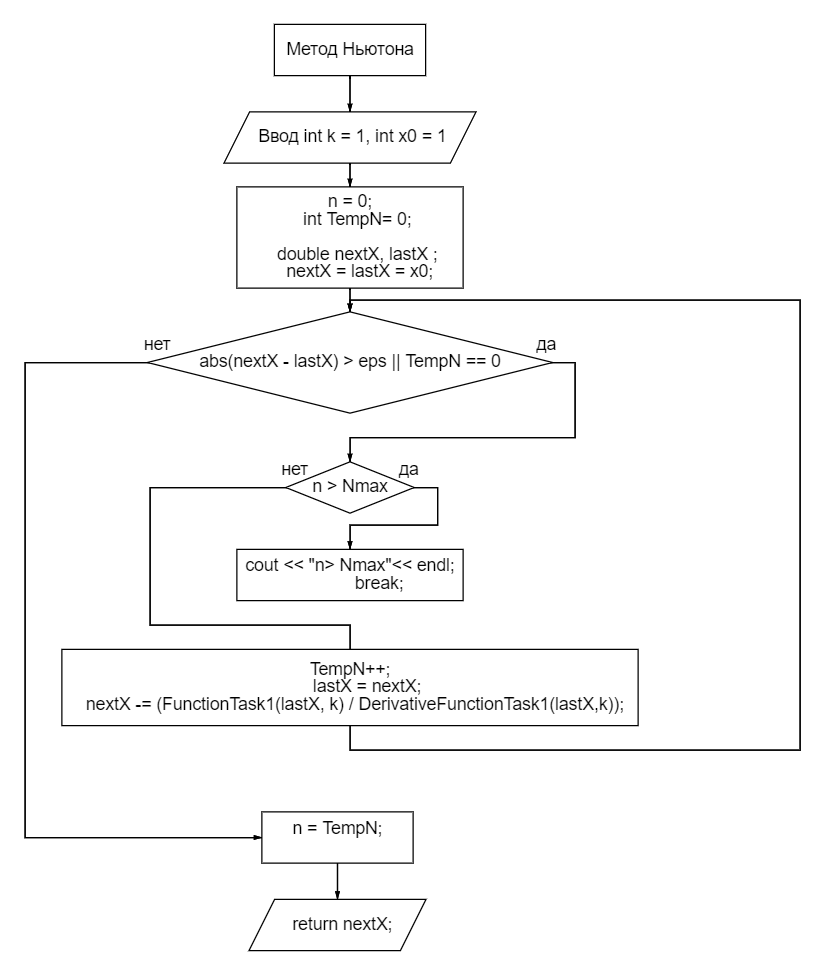
double current = x0;// - текущий член

переменные соответствуют наименования переменных в задании.

* укрупненную схему алгоритма для пункта 3 задания;



* уточненной схемы ***одного из блоков***:



# Текст программы

#include <iostream>

#include <math.h>

#include <tuple>

#include <string>

using namespace std;

int n , eps = 0.00000001;

const int Nmax = 10000;

static int Sign(double a) {

if (a == 0) return 0;

if (a > 0) return 1;

return -1;

}

static int Sign(int a) {

if (a == 0) return 0;

if (a > 0) return 1;

return -1;

}

static int Sign(float a) {

if (a == 0) return 0;

if (a > 0) return 1;

return -1;

}

//TASK 1

double FunctionTask1(long double x, long double k) {

return x - k \* cos(x);

}

double DerivativeFunctionTask1(long double x, long double k) {

return 1 + k \* sin(x);

}

double Neuwton(int k = 1, int x0 = 1) {

n = 0;

int TempN= 0;

double nextX, lastX ;

nextX = lastX = x0;

while (abs(nextX - lastX) > eps || TempN == 0) {

if (n > Nmax) {

cout << "n> Nmax"<< endl;

break;

}

TempN++;

lastX = nextX;

nextX -= (FunctionTask1(lastX, k) / DerivativeFunctionTask1(lastX,k));

}

n = TempN;

return nextX;

}

//END OF TASK 1-2

//TASK 3

long double BisectionMethod(double Xn, double Xk, long double k) {

if (FunctionTask1(Xn, k) == 0) return Xn;

if (FunctionTask1(Xk, k) == 0) return Xk;

double Xi = 0;

n = 0;

while (Xk - Xn > eps && n < Nmax){

Xi = Xn + ((Xk - Xn) / 2);

if (Sign(FunctionTask1(Xn, k)) != Sign(FunctionTask1(Xi, k))) {

Xk = Xi;

}

else Xn = Xi;

n++;

}

return Xi;

}

//END OF TASK 3

//Task 4

double SimpleIteration(double x0) {

double next = x0;

double current = x0;

n = 0;

while ((abs(current - next) > eps) || (n == 0))

{

n++;

current = next;

next = current - FunctionTask1(current,1);

}

return next;

}

//END OF TASK 4

int main()

{

setlocale(LC\_ALL, "Russian");

cout << "Введите соответсвующую задачу Лабораторной работы #4 >> " ;

int typeOfTask;

cin >> typeOfTask;

switch (typeOfTask)

{

case 1:

cout << "Полученный результат >> " << Neuwton() << endl;

cout << "Результат получен шагом номер >> " << n << endl;

break;

case 2:

cout << "Введите входные данные в формате k x0 eps >> ";

int a, b, c;

cin >> a >> b >> c;

eps = c;

cout << "Полученный результат >> " << Neuwton(a,b) << endl;

cout << "Результат получен шагом номер >> " << n << endl;

break;

case 3:

cout << "Метод половинного деления\n";

double Xn, Xk, k;

Xn = Xk = k = 0;

cout << "Введите Xn >> ";

cin >> Xn;

cout << "Введите Xk >> ";

cin >> Xk;

cout << "Введите k >> ";

cin >> k;

cout << "Полученный результат >> " << BisectionMethod(Xn,Xk,k) << endl;

cout << "Результат получен шагом номер >> " << n << endl;

break;

case 4:

cout << "Метод простых иттераций\n";

cout << "Введите X0 >> ";

int x0;

cin >> x0;

cout << "Результат >> " << SimpleIteration(x0) << endl;

cout << "Результат получен шагом номер >> " << n << endl;

break;

default:

string input;

cout << "Введеты недопустимые значения. Попробовать снова?(Если нет - оставьте поле пустым)\n";

cin >> input;

if (input[0]) main();

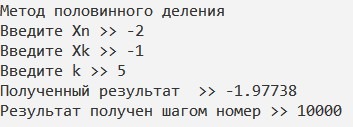
break;

}

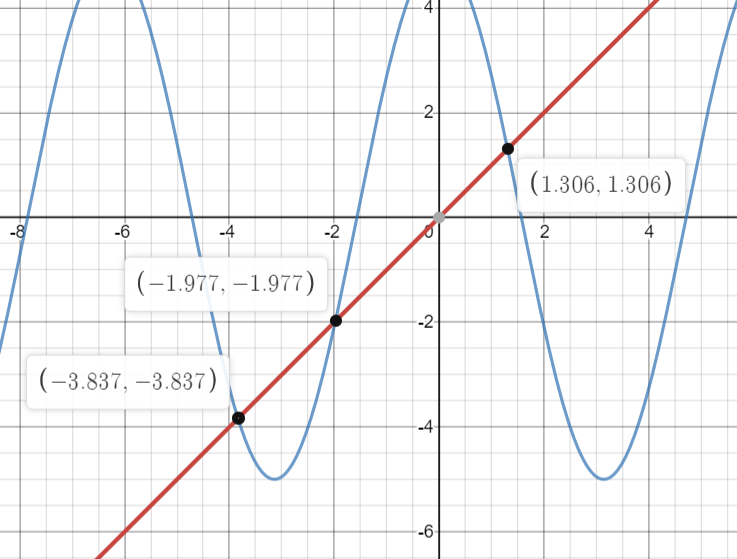
return 0;

}

# Анализ результатов



Численному решению уравнения должно предшествовать хотя бы грубое исследование вопросов существования и положения корней. Так найдя некоторую окрестность содержащую корень мы находим непосредственно сам корень.



Представим функцию y = x – 5 cos(x) в виде системы функций g(x) = x и u(x) = 5 cos(x)

Тогда точки пересечения нашего графика – есть корни.