МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МО ЭВМ

ОТЧЕТ

по лабораторной работе №3

по дисциплине «Вычислительная математика»

Тема: Решение нелинейных уравнений

Студент гр. 0304 _____ Алексеев Р.В.

Преподаватель _____ Попова Е.В.

Санкт-Петербург

2021

Вариант 1

Цель работы.

Формирование практических навыков нахождения корней алгебраических и трансцедентных уравнений методами бисекции и хорд.

Основные теоретические положения.

Метод бисекции. Если найден отрезок [a, b], такой, что f(a)f(b) < 0, an = ξ, bn = bn − 1, если f(ξ) f(an − 1) > 0, Если требуется найти корень с точностью ε, то деление пополам продолжается до тех пор, пока длина отрезка не станет меньше 2ε. Тогда координата середины отрезка есть значение корня с требуемой точностью ε. Метод бисекции является простым и надежным методом поиска простого корня уравнения f(x)=0 (простым называется крень x=c дифференцируемой функции f(x), если f(c)=0 и f′(c)≠0). Этот метод сходится для любых непрерывных функций f(x), в том числе недифференцируемых. Скорость его сходимости невысока. Для достижения точности ε необходимо совершить N≈log2((b−a)/ε) итераций. Это означает, что для получения каждых трех верных десятичных знаков необходимо совершить около 10 итераций.

Постановка задачи. Используя программы-функции BISECT и Round из файла methods.cpp (файл заголовков metods.h), найти корень уравнения методом бисекции с заданной точностью Eps, исследовать зависимость числа итераций от точности Eps при изменении Eps от 0.1 до 0.000001, исследовать обусловленность метода (чувствительность к ошибкам в исходных данных).

1. Графически или аналитически отделить корень уравнения f(x)=0, т.е. найти отрезки [a,b], на которых функция удовлетворяет условиям теоремы Больцано-Коши.

2. Составить подпрограмму вычисления функции f(x).

3. Составить головную программу, содержащую обращение к подпрограмме F, BISECT, Round и индикацию результатов.

4. Провести вычисления по программе. Построить график зависимости числа итераций от Eps, сопоставить его с графиком по формуле выше.

5. Исследовать чувствительность метода к ошибкам в исходных данных. Ошибки в исходных данных моделировать с использованием программы Round, округляющей значения функции с заданной точностью Delta.

Метод хорд. Пусть найден отрезок [a,b], на котором функция меняет знак. Для определенности положим f(a) > 0, f(b) < 0. В методе хорд процесс итераций состоит в том, что в качестве приближений к корню уравнения f(x)=0 принимаются значения c0, c1,... точек пересечения хорды с осью абсцисс. Сначала находится уравнение хорды AB: $y-f(a)\,f(b)-f(a)=x-a\,b-a$ Для точки пересечения ее с осью абсцисс (x = c0, y = 0) получается уравнение: $c0=a-b-a\,f(b)-f(a)\cdot f(a)$ Далее сравниваются знаки величин f(a) и f(c0) и для рассматриваемого случая оказывается, что корень находится в интервале (a, c0), так как f(a)f(c0) < 0. Отрезок [c0, b] отбрасывается. Следующая итерации состоит в определении нового приближения c1 как точки пересечения хорды AB1 с осью абсцисс и т. д. Итерационный процесс продолжается до тех пор, пока значение f(cn) не станет по модулю меньше заданного числа ε. Алгоритмы методов бисекции и хорд похожи, однако метод хорд в ряде случаев дает более быструю сходимость итерационного процесса, причем успех его применения, как и метода бисекции, гарантирован.

Постановка задачи. Используя функции HORDA и Round, найти корень уравнения f(x) = 0 заданной точностью eps методом хорд, исследовать скорость сходимости и обусловленность метода. Порядок выполнения работы. 1. Графически или аналитически отделить корень уравнения f(x)=0, т.

е. найти отрезки [a,b], на которых функция f(x) удовлетворяет условиям применимости метода.

2. Составить подпрограмму-функцию вычисления функции f(x), предусмотрев округление значений функции с заданной точностью delta с использованием программы Round.

3. Составить головную программу, вычисляющую корень уравнения f(x)=0 и содержащую обращение к подпрограмме F, HORDA, Round и индикацию результатов.

4. Провести вычисления по программе. Теоретически и экспериментально исследовать скорость сходимости и обусловленность метода.

Выполнение работы.

Функция $f(x) = \frac{4e^{2x}+6}{e^{4x}+2e^{2x}+1} - 1$

1. Посторим график и локализуем корень (см. рис. 1)

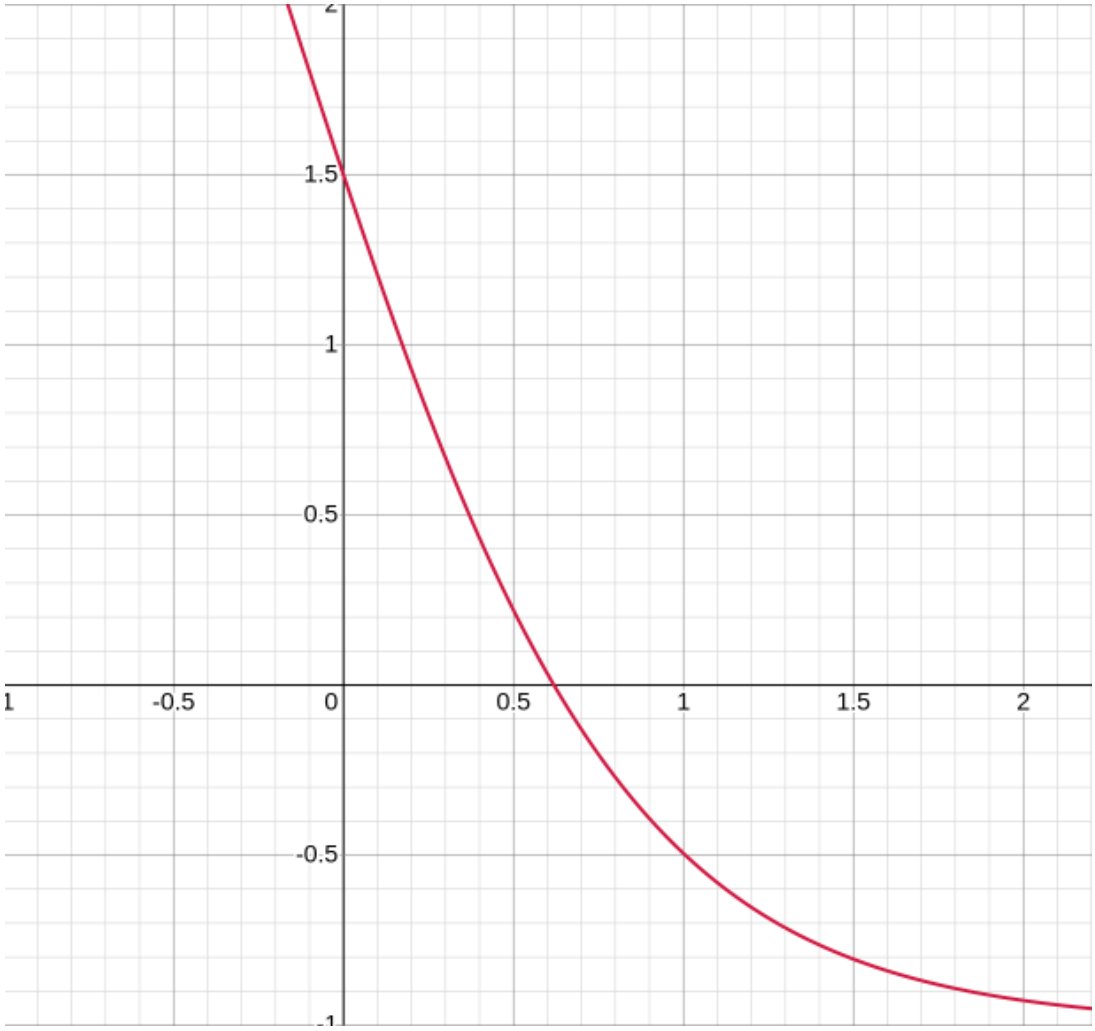

Рисунок 1 — График функции $f(x) = \frac{4e^{2x}+6}{e^{4x}+2e^{2x}+1} - 1$

По графику видно, что корень уравнения x лежит в промежутку [0;1].

$f(0) > 0,\ f(1) < 0$.

Найдем первую производную функции, которая нужна для поиска коэффициента обусловленности:

$$f'(x) = \frac{(4e^{2x}+6)'(e^{4x}+2e^{2x}+1) - (4e^{2x}+6)(e^{4x}+2e^{2x}+1)'}{(e^{4x}+2e^{2x}+1)^2} =$$

$$= \frac{4e^{2x}(2x)'(e^{4x}+2e^{2x}+1) - (4e^{2x}+6)(e^{4x}(4x)'+2e^{2x}(2x)')}{(e^{4x}+2e^{2x}+1)^2} =$$

$$= \frac{8e^{2x}(e^{4x}+2e^{2x}+1)-(4e^{2x}+6)(4e^{4x}+4e^{2x})}{(e^{4x}+2e^{2x}+1)^2} = \frac{-8e^{4x}-16e^{2x}}{e^{6x}+3e^{4x}+3e^{2x}+1}$$

Найдем вторую производную функции, которая нужна для нахождения неподвижной точки в методе хорд:

$$f''(x)= \frac{(-8e^{4x}-16e^{2x})'(e^{6x}+3e^{4x}+3e^{2x}+1)-(e^{6x}+3e^{4x}+3e^{2x}+1)'(-8e^{4x}-16e^{2x})}{(e^{6x}+3e^{4x}+3e^{2x}+1)^2} =$$

$$= \frac{(-32e^{4x}(x)'-32e^{2x}(x)')(e^{6x}+3e^{4x}+3e^{2x}+1)}{(e^{6x}+3e^{4x}+3e^{2x}+1)^2} -$$

$$- \frac{(6e^{6x}(x)'+12e^{4x}(x)'+6e^{2x}(x)')(-8e^{4x}-16e^{2x})}{(e^{6x}+3e^{4x}+3e^{2x}+1)^2} =$$

$$= \frac{(-32e^{4x}-32e^{2x})(e^{6x}+3e^{4x}+3e^{2x}+1)-(6e^{6x}+12e^{4x}+6e^{2x})(-8e^{4x}-16e^{2x})}{(e^{6x}+3e^{4x}+3e^{2x}+1)^2} =$$

$$= \frac{16e^{6x}+32e^{4x}-32e^{2x}}{e^{8x}+4e^{6x}+6e^{4x}+4e^{2x}+1}$$

Метод бисекции

2. eps = 0,001, delta варьируется от 0,1 до 0,000001. Промежуток, содержащий корень — [0;1]. Результат работы программы представлен на рис. 2.

eps	delta	x	nu	nu_max	Вывод
0.001000	0.100000	0.600000	0.570587	0.010000	Плохо
0.001000	0.010000	0.620000	0.586494	0.100000	Плохо
0.001000	0.001000	0.619000	0.585682	1.000000	Хорошо
0.001000	0.000100	0.619100	0.585763	10.000000	Хорошо
0.001000	0.000010	0.619140	0.585796	100.000000	Хорошо
0.001000	0.000001	0.619141	0.585796	1000.000000	Хорошо

Рисунок 2 — Первая таблица для метода бисекции.

По рисунку видно, что с уменьшением delta увеличивается величина максимальной абсолютной обусловеленности, и, при достежении параметром delta значения 0,001, задача становится хорошо обусловленной. Значение

обусловленности находится по формуле $nu = \frac{1}{|f'(x)|}$, где x — корень, найденный методом бисекции.

3. Возьмем delta = 0,000001, eps варьируется между 0,1 и 0,000001. Промежуток, содержащий корень — [0;1]. Результаты работы программы на рис. 3.

eps	delta	x	iter
0.000001	0.000001	0.619112	19
0.000010	0.000001	0.619125	16
0.000100	0.000001	0.619019	13
0.001000	0.000001	0.619141	9
0.010000	0.000001	0.609375	6
0.100000	0.000001	0.625000	3

Рисунок 3 — Вторая таблица для метода бисекции.

По рисунку видно, что чем больше eps, тем меньше количество итераций. График зависимости количества итераций N от eps представлен на рис. 4.

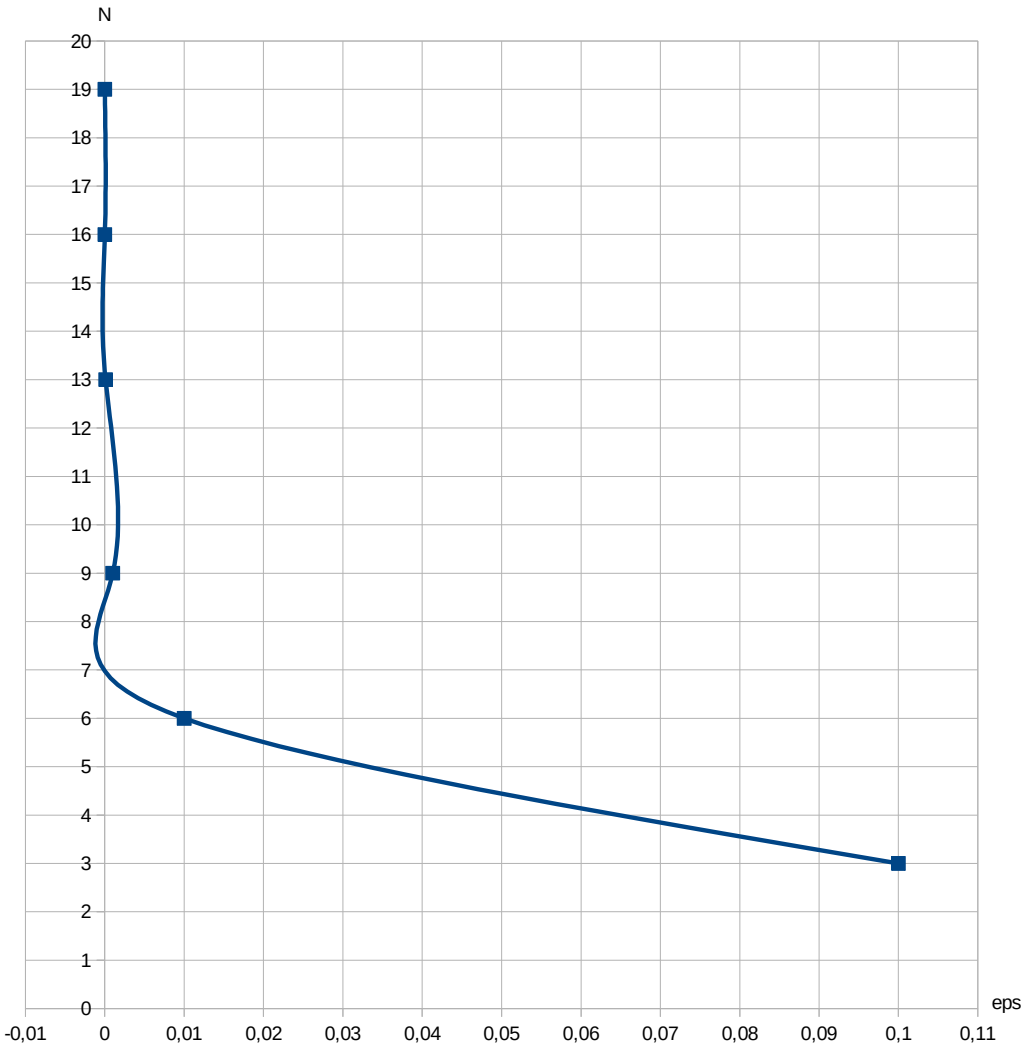

Рисунок 4 — График зависимости количества итераций от eps.

На графике видна зависимость N ~ log($\frac{1}{eps}$), которая обусловлена тем, что алгоритм имеет логарифмическую сложность, а параметр алгоритма обратно зависим от eps.

4. Пусть eps = 0,001 и delta = 0,0001. Значение правой границы отрезка, содержащего корень, варьируется от 6 до 1 с шагом 1, значение левой границы будет варьироваться -0,5 до 0 с шагом 0,1. Результат работы программы представлен на рис. 5.

left	right	x	iter
-0.500000	6.000000	0.619629	12
-0.400000	5.000000	0.618896	12
-0.300000	4.000000	0.619141	11
-0.200000	3.000000	0.619629	11
-0.100000	2.000000	0.619141	10
-0.000000	1.000000	0.619141	9

Рисунок 5 — Третья таблица для метода бисекции.

По результатам работы программы видно, что точность вычисленного значения не зависит от отрезка локализации, но чем больше длина отрезка, тем больше количество итераций.

Метод хорд

5. Возьмем eps = 0,001, delta варьируется от 0,1 до 0,000001. Промежуток, содержащий корень — [0;1]. Результат программы представлен на рис. 6.

eps	delta	x	nu	nu_max	Вывод
0.001000	0.100000	0.600000	0.570587	0.010000	Плохо
0.001000	0.010000	0.620000	0.586494	0.100000	Плохо
0.001000	0.001000	0.619000	0.585682	1.000000	Хорошо
0.001000	0.000100	0.619400	0.586007	10.000000	Хорошо
0.001000	0.000010	0.619440	0.586039	100.000000	Хорошо
0.001000	0.000001	0.619442	0.586041	1000.000000	Хорошо

Рисунок 6 — Первая таблица для метода хорд.

По рисунку видно, что с уменьшением значения delta увеличивается значение абсолютной обсуловленности, при delta = 0,001 задача становится хорошо обусловленной. Значение обусловленности находится по формуле $nu = \frac{1}{|f'(x)|}$, где x — корень, найденный методом хорд.

6. Возьмем delta = 0,0001, eps варьируется от 0,1 до 0,000001. Промежуток, содержащий корень — [0;1]. Результаты работы программы, представлены на рис.7.

eps	delta	x	iter
0.100000	0.000100	0.661000	2
0.010000	0.000100	0.622900	4
0.001000	0.000100	0.619400	6
0.000100	0.000100	0.619100	8
0.000010	0.000100	0.619100	10
0.000001	0.000100	0.619100	12

Рисунок 7 — Вторая таблица для метода хорд.

По рисунку видно, что чем меньше значение eps, тем больше количество итераций. График зависимости количества итераций N от eps представлен на рис. 8.

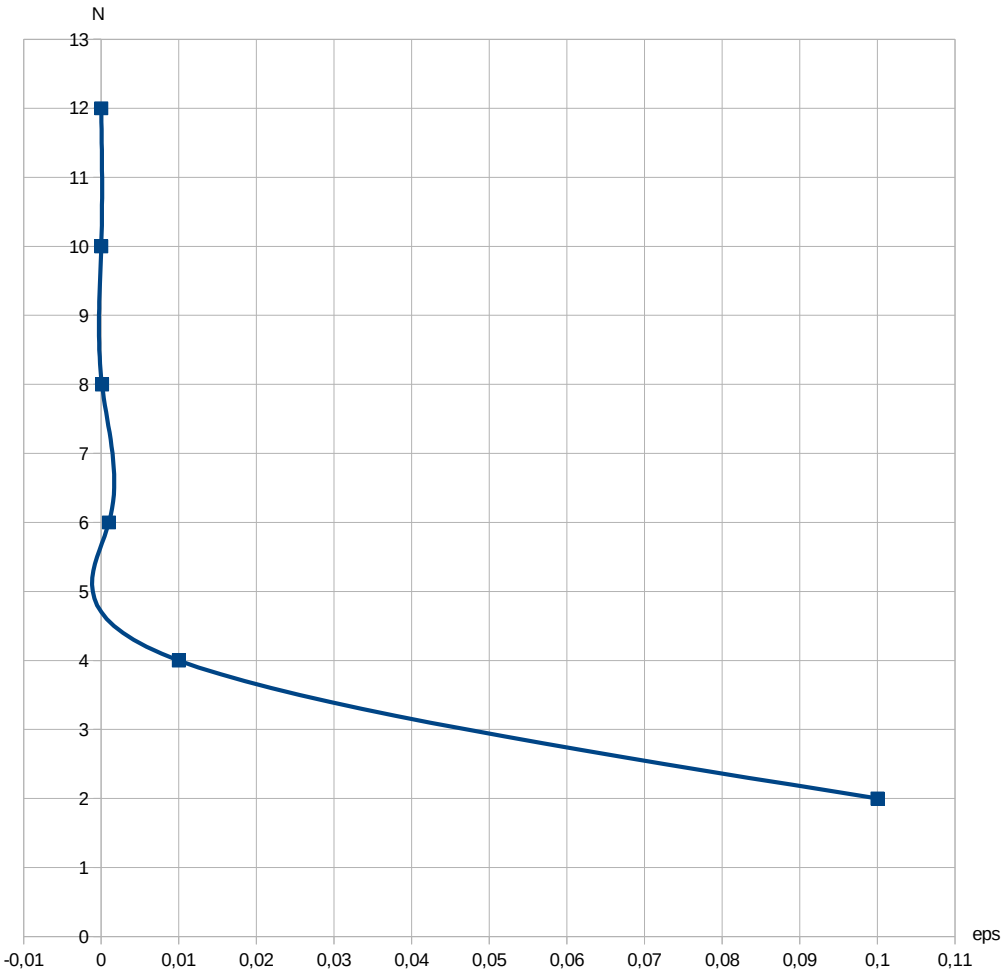

Рисунок 8 — График зависимости количества итераций N от eps.

На графике как и в методе бисекции прослеживается зависимость N ~ log($\frac{1}{eps}$), но в методе хорд количество итераций меньше чем в методе бисекции (12 итераций в методе хорд и 19 в методе бисекции).

7. Возьмем eps = 0,001, delta = 0,0001. Будет варьироваться значение точки входа. Точка входа — одна из границ отрезка, а именно та, знак функции в которой совпадает со знаком второй производной в этой точке. Правая граница варьируется от 1,2 до 0,7 с шагом 0,1. Результаты работы программы представлены на рис. 9.

left	right	x	iter
0.000000	1.200000	0.619681	6
0.000000	1.100000	0.619555	6
0.000000	1.000000	0.619442	6
0.000000	0.900000	0.619340	6
0.000000	0.800000	0.619575	5
0.000000	0.700000	0.619306	5

Рисунок 9 — Третья таблица для метода хорд.

По рисунку видно, что точность вычисленного значения не зависит от точки входа, но чем ближе точка входа к корню, тем меньше количество итераций.

Сравнение методов решения уравнений

8. Возьмем delta = 0,0001, eps варьируется от 0,1 до 0,000001. Промежуток, содержащий корень — [0;1]. Результаты работы программы, представлены на рис. 10.

eps	x_BISECT	x_HORDA	iter_B	iter_H
0.100000	0.625000	0.660969	3	2
0.010000	0.609375	0.622867	6	4
0.001000	0.619141	0.619442	9	6
0.000100	0.619019	0.619142	13	8
0.000010	0.619125	0.619116	16	10
0.000001	0.619112	0.619113	19	12

Рисунок 10 — Сравнительная таблица методов бисекции и хорд.

По рисунку видно, что значение корня в методах бисекции и хорд примерно одинаково, но метод бисекции требует большего количества итераций.

Разработанный программный код см. в приложении А.

Выводы.

Были получены навыки нахождения корней нелинейного уравнения методами бисекции и хорд на примере функции $f(x) = \frac{4e^{2x}+6}{e^{4x}+2e^{2x}+1} - 1$. Подтверждено, что метод хорд в среднем работает быстрее метода бисекции, при прмиерно одинаковой точности.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ НА ЯЗЫКЕ С++

```
#include <iostream>
#include <cmath>

using namespace std;

double c;

double F(double x){
  return (((4*exp(2*x)+6)/(exp(4*x)+2*exp(2*x)+1))-1);
}

double nu(double x){
  return(fabs((exp(6*x)+3*exp(4*x)+3*exp(2*x)+1)/(-8*exp(4*x)-
16*exp(2*x))));
}

double HORDA(double Left, double Right, double Eps, int &N){
  double FLeft = F(Left);
  double FRight = F(Right);
  double X, Y;
  if(FLeft * FRight > 0.0){
      puts("Неверное задание интервала\n");
      exit(1);
  }
  if(Eps <= 0.0){
      puts("Неверное задание точности\n");
      exit(1);
  }
  N = 0;
  if(FLeft == 0.0){
      return Left;
  }
  if(FRight == 0.0){
      return Right;
  }
  do{
      X = Left - (Right - Left) * FLeft / (FRight - FLeft);
      Y = F(X);
      if(Y == 0.0){
          return X;
      }
      if(Y * FLeft < 0.0){
          Right = X;
          FRight = Y;
      }
      else{
          Left = X;
          FLeft = Y;
      }
      N++;
  }
  while (fabs(Y) >= Eps);
```

```
  return X;
}

double BISECT(double Left, double Right, double Eps, int &N){
  double E = fabs(Eps)*2.0;
  double FLeft = F(Left);
  double FRight = F(Right);
  double X = (Left + Right) / 2.0;
  double Y;

  if (FLeft*FRight>0.0){
      puts("neverno zadan interval\n");
      exit(1);
  }

  if (Eps <= 0.0){
      puts("neverno zadana tochnost\n"); exit(1);
  }

  N = 0;

  if (FLeft == 0.0)
      return Left;

  if (FRight == 0.0)
      return Right;

  while ((Right - Left) >= E){
      X = 0.5*(Right + Left);
      Y = F(X);

      if (Y == 0.0)
          return (X);

      if (Y*FLeft < 0.0)
          Right = X;
      else{
          Left = X;
          FLeft = Y;
      }
      N++;
  };

  return(X);
}

double Round(double X, double Delta){
  if (Delta <= 1E-9){
      puts("Неверно задана точность округления\n");
      exit(1);
  }

  if (X>0.0)
      return (Delta*(long((X / Delta) + 0.5)));
  else
      return (Delta*(long((X / Delta) - 0.5)));
```

```
}

int main(){
  double eps = 0.001;
  double delta = 0.01;
  int n = 0;


puts("-----------------------------------------------------------------
-------------------------------");
  printf("%6s\t%14s\t%12s\t%14s\t%16s\t%6s\t\n",  "eps",  "delta",
"x", "nu", "nu_max", "Вывод");

puts("-----------------------------------------------------------------
-------------------------------");
  for(delta = 0.1; delta >= 0.000001; delta *= 0.1){
     double x = Round(BISECT(0, 1, eps, n), delta);
     printf("%6f\t%6f\t%6f\t%6f\t%6f\t",  eps,  delta,  x,  nu(x),
eps/delta);
     if(nu(x) < eps/delta)
        printf("%6s\n", "Хорошо");
     else
        printf("%6s\n", "Плохо");
  }

puts("-----------------------------------------------------------------
-------------------------------");

  delta = 0.000001;


puts("-----------------------------------------------------------------
-------------------------------");
  printf("%6s\t%14s\t%13s\t%11s\n", "eps", "delta", "x", "iter");

puts("-----------------------------------------------------------------
-------------------------------");
  for(eps = 0.000001; eps <= 0.1; eps *= 10){
     double x = Round(BISECT(0, 1, eps, n), delta);
     printf("%6f\t%6f\t%6f\t%2d\n", eps, delta, x, n);

  }

puts("-----------------------------------------------------------------
-------------------------------");

  eps = 0.001;
  delta = 0.000001;


puts("-----------------------------------------------------------------
-------------------------------");
  printf("%6s\t%14s\t%13s\t%11s\n",    "left",    "right",    "x",
"iter");
```

```
puts("-------------------------------------------------------------------
------------------------------");
       for(double i = 6, e = -0.5; i >= 1.0, e <= 0.0; i -= 1, e +=
0.1){
           double x = Round(BISECT(0, i, eps, n), delta);
           printf("%6f\t%6f\t%6f\t%2d\n", e, i, x, n);

       }

puts("-------------------------------------------------------------------
------------------------------");

     //Метод хорд


puts("-------------------------------------------------------------------
------------------------------");
       printf("%6s\t%14s\t%12s\t%14s\t%16s\t%6s\t\n",  "eps",  "delta",
"x", "nu", "nu_max", "Вывод");

puts("-------------------------------------------------------------------
------------------------------");
       for(delta = 0.1; delta >= 0.000001; delta *= 0.1){
           double x = Round(HORDA(0, 1, eps, n), delta);
           printf("%6f\t%6f\t%6f\t%6f\t%6f\t",  eps,  delta,  x,  nu(x),
eps/delta);
           if(nu(x) < eps/delta)
               printf("%6s\n", "Хорошо");
           else
               printf("%6s\n", "Плохо");
       }

puts("-------------------------------------------------------------------
------------------------------");

       delta = 0.0001;


puts("-------------------------------------------------------------------
------------------------------");
       printf("%6s\t%14s\t%13s\t%11s\n", "eps", "delta", "x", "iter");

puts("-------------------------------------------------------------------
------------------------------");
       for(eps = 0.1; eps >= 0.000001; eps *= 0.1){
           double x = Round(HORDA(0, 1, eps, n), delta);
           printf("%6f\t%6f\t%6f\t%2d\n", eps, delta, x, n);

       }

puts("-------------------------------------------------------------------
------------------------------");

       eps = 0.001;
       delta = 0.000001;
```

```
puts("-------------------------------------------------------------------
-------------------------------");
       printf("%6s\t%14s\t%13s\t%11s\n",    "left",    "right",    "x",
"iter");

puts("-------------------------------------------------------------------
-------------------------------");
       for(double i = 1.2; i >= 0.7; i -= 0.1){
           double x = Round(HORDA(0, i, eps, n), delta);
           printf("%6f\t%6f\t%6f\t%2d\n", 0.0, i, x, n);

       }

puts("-------------------------------------------------------------------
-------------------------------");

     //Сравнение

       delta = 0.000001;

puts("-------------------------------------------------------------------
-------------------------------");
       printf("%6s\t%16s\t%8s\t%2s\t%s\t\n",     "eps",     "x_BISECT",
"x_HORDA", "iter_B", "iter_H");

puts("-------------------------------------------------------------------
-------------------------------");
       for(eps = 0.1; eps >= 0.000001; eps *= 0.1){
           int m;
           double x_b = Round(BISECT(0, 1, eps, n), delta);
           double x_h = Round(HORDA(0, 1, eps, m), delta);
           printf("%6f\t%6f\t%6f\t%3d\t%3d\n", eps, x_b, x_h, n, m);
       }

puts("-------------------------------------------------------------------
-------------------------------");


       return 0;
     }
```