Міністерство освіти і науки України

Національний університет “Львівська політехніка”

Інститут комп’ютерних наук та інформаційних технологій

Кафедра програмного забезпечення



**Звіт**

До лабораторної роботи №1

**На тему:**  «Розв’язування нелінійних рівнянь за методом дихотомії та за методом хорд»

**З дисципліни:** “Чисельні методи”

**Лектор:**

доц. каф. ПЗ

Мельник Н.Б.

**Виконав:**

ст. гр. ПЗ-18

Лук’янов Н.О.

**Прийняв:**

проф. каф. ПЗ

Гавриш В.І.

« … » … 2022 р.

∑ = \_\_\_\_\_\_\_\_\_\_\_\_

Львів – 2022

**Тема**: розв’язування нелінійних рівнянь за методом дихотомії та за методом хорд.

**Мета**: ознайомлення на практиці з методами відокремлення дійсних ізольованих коренів нелінійних рівнянь. Вивчення методів уточнення коренів - методу дихотомії та методу хорд.

**Завдання**

Методами дихотомії та хорд визначити корінь рівняння з точністю . Рівняння: - 3 + 12x - 9 = 0. Код до програми у Додатку.

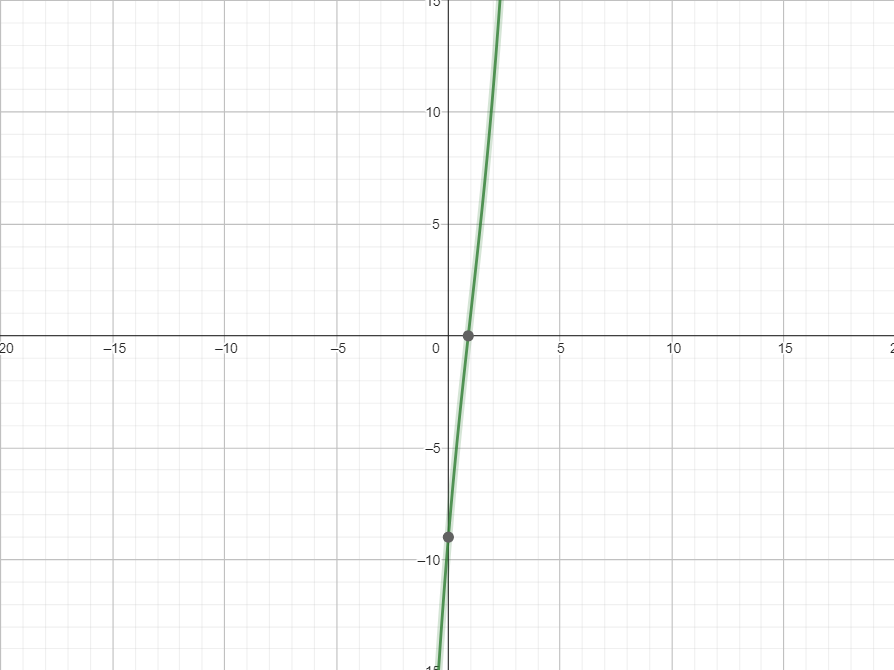


Рис.1. Графік функції f(x) =  - 3 + 12x - 9

**Метод дихотомії**

Наведемо основні етапи методу дихотомії:

1. локалізуємо дійсний корінь на відрізку;
2. якщо на даному відрізку дійсний корінь існує, то визначаємо середину відрізка, а саме точку с;
3. перевіряємо знаки значень функції в точках a та c і c та b. Якщо f(a) \* f(c) > 0 - це означає, що відрізок [a;c] не містить кореня рівняння, а отже він знаходиться на відрізку [c;b], тому повинно виконуватися наступне f(c) \* f(b) < 0;
4. вибираємо відрізок [c;b], який містить корінь;
5. виконуємо наступні кроки наведені в пунктах 2, 3, 4 доти, поки не отримаємо задану точність.

**Основні етапи обчислювального алгоритму, який був реалізований у програмному продукті на мові C :**

1. введення даних користувачем для ініціалізації змінних(left\_border, right\_border), що відповідають за початковий проміжок знаходження кореня;
2. перевірка на коректність введених даних, а саме перевірка заданого проміжку на коректність за допомогою функції Is\_Correct\_Data() (рис.2). В разі введення проміжка з некоректними межами, функція змінює межі місцями та виводить інформацію про це користувачу;
3. локалізація кореня на відрізку за допомогою функції First\_Locale() (рис.3);
4. якщо корінь знайдено передаємо відрізок з ним у функцію - Localization\_with\_dividing() (рис.4), яка описує алгоритм методу дихотомії(див. вище), інакше виводиться повідомлення про помилку. Інформація про роботу функції виводиться на екран під час кожної її ітерації, допоки не отримаємо задану точність;
5. результат виконання програми (рис.5).

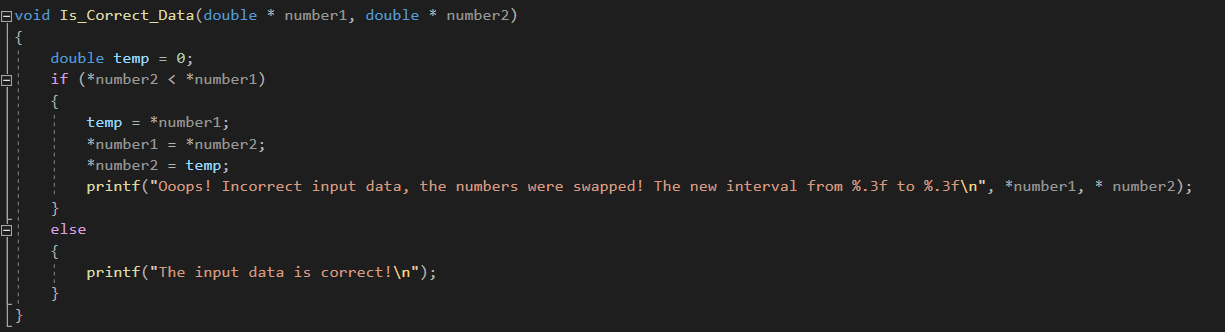


Рис.2. Функція Is\_Correct\_Data

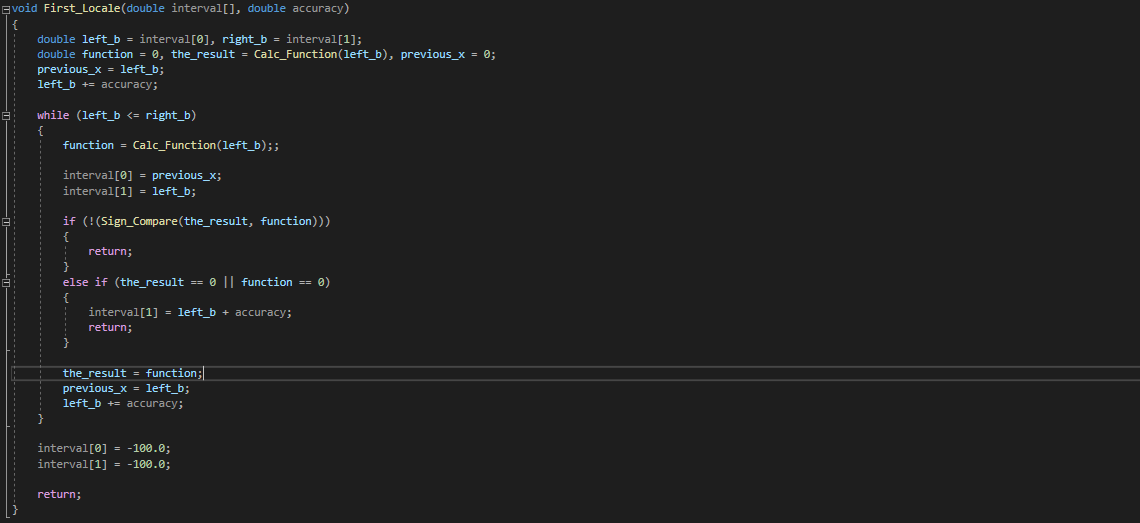


Рис.3. Функція First\_Locale

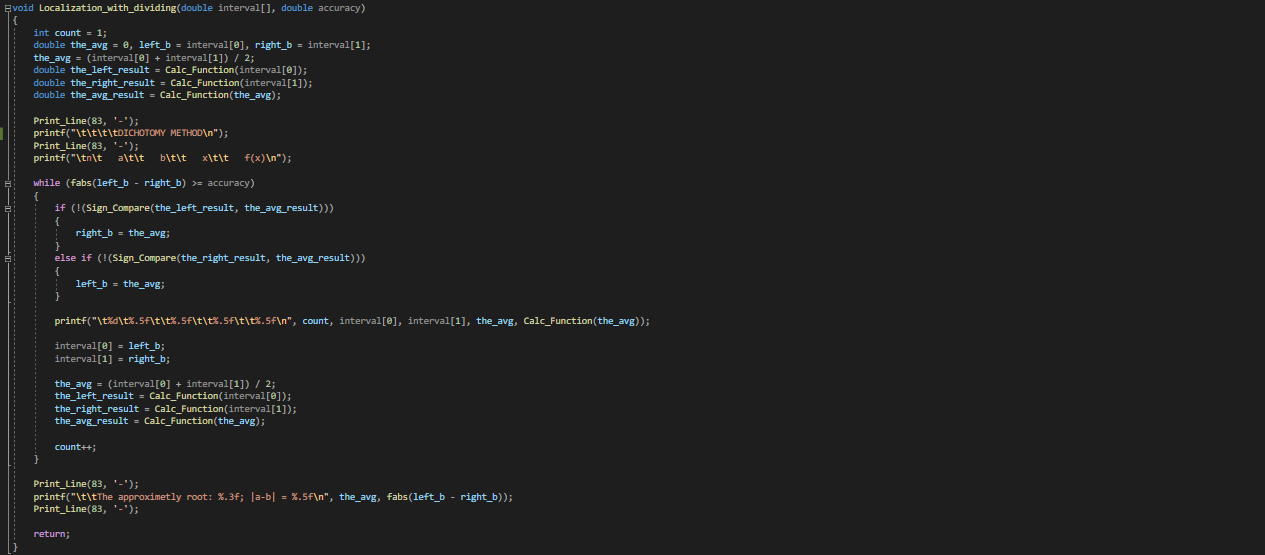


Рис.4. Функція Localization\_with\_dividing

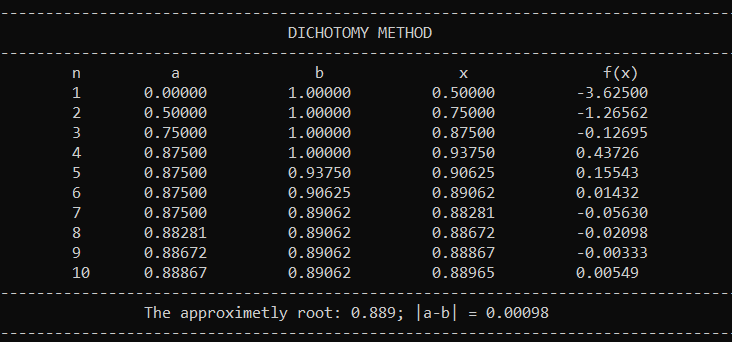


Рис.5. Результат виконання програми

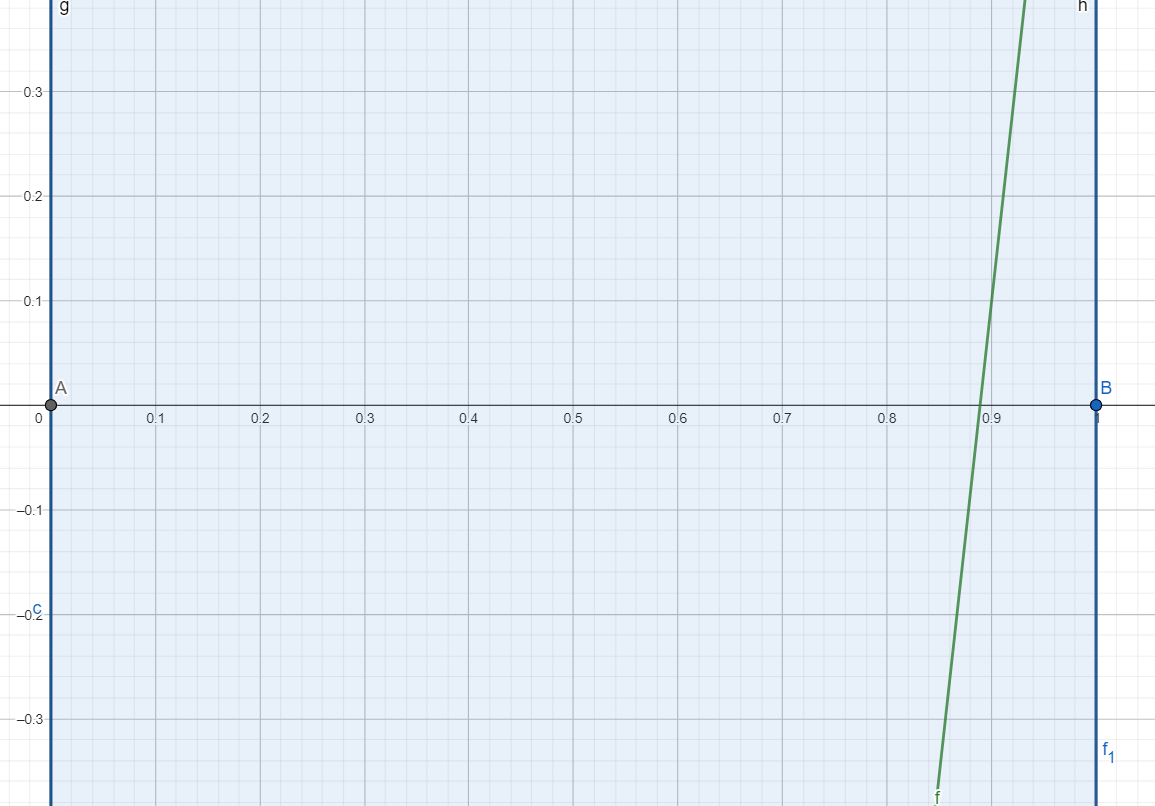


Рис.6. Відрізок AB на якому знаходиться корінь

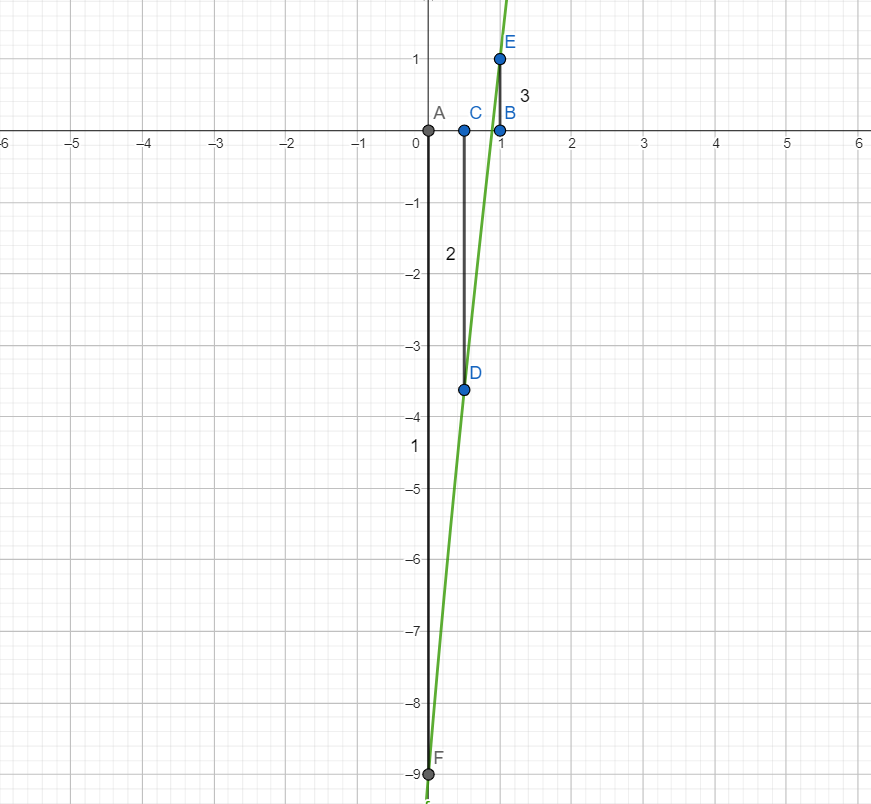


Рис.7. Значення функції в точках A, B, C

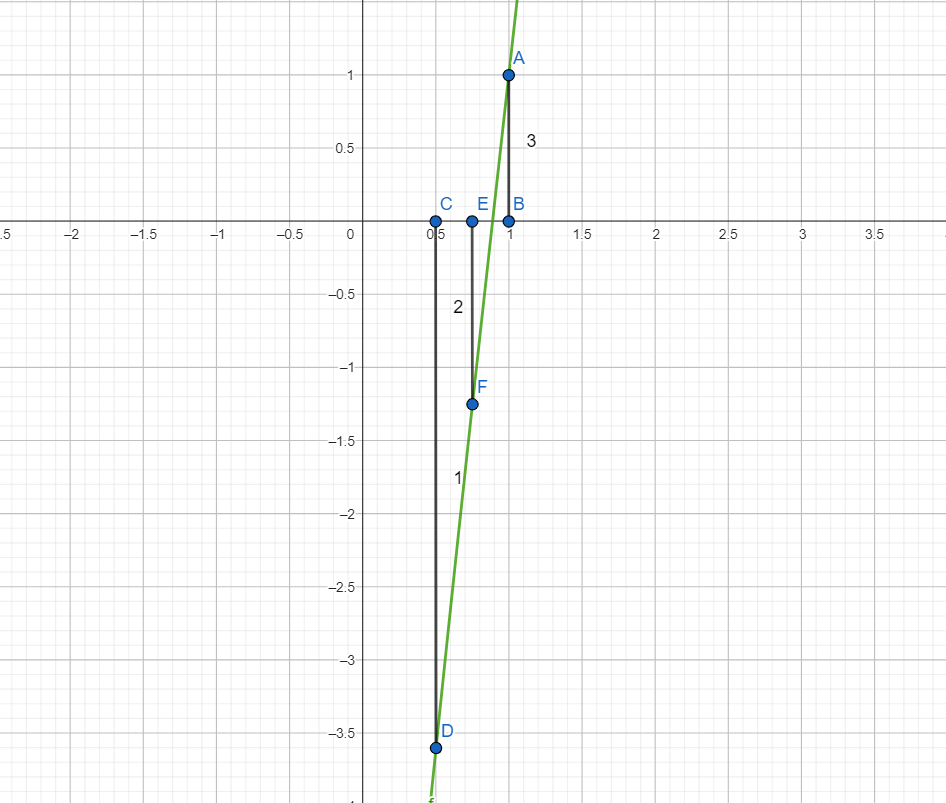


Рис.8. Друга ітерація методу дихотомії

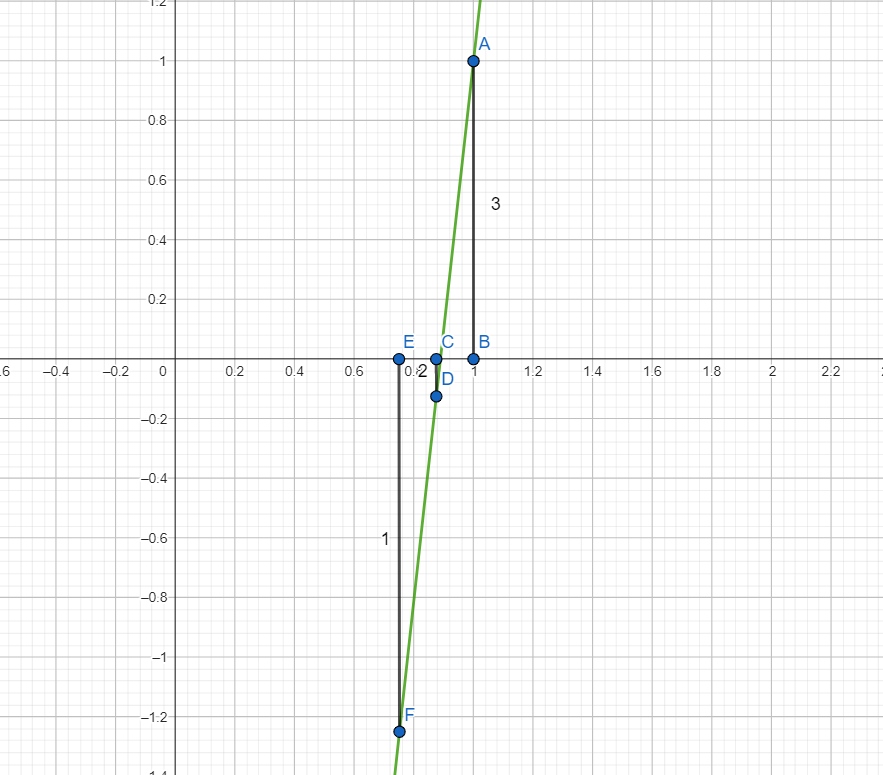


Рис.9. Третя ітерація методу дихотомії

**Визначення кореня за методом дихотомії**

1. локалізуємо корінь рівняння: - 3 + 12x - 9 = 0, в випадку з даним рівнянням, корінь знаходиться на відрізку [A;B], довжина якого дорівнює 1 (рис. 6);
2. знаходимо центр цього відрізка, в даному випадку це т. C (x = 0.5). Знаходимо значення в точці C, після чого знаходимо значення функції на кінцях відрізка(рис.7);
3. на рисунку 7 видно, що значення функції в точках A та C мають однакові знаки тобто f(A) \* f(C) > 0, отже корінь знаходиться на відрізку [C;B], тобто f(C) \* f(B) < 0;
4. виконуємо наступні кроки наведені в пунктах 2 та 3 доти, поки не досягнемо заданої точності;
5. після десятої ітерації корінь був визначений з заданою точністю.

**Метод хорд**

Перед тим, як розпочати роботу розберемо алгоритм методу хорд:

1. локалізуємо дійсний корінь на відрізку;
2. якщо відрізок з коренем існує, то складаємо рівняння прямої через дві точки, а саме через кінці відрізка на якому знаходиться корінь;
3. знаходимо перетин цієї прямої з віссю абсцис, G = a + , де G - це x координата точки перетину з віссю абсцис, a - лівий край відрізку, b - правий край відрізку;
4. вибираємо точку через яку проводити наступну пряму, для цього перевіряємо знаки значень функції в точках a та c і c та b, де c це точка перетину прямої через кінці з віссю абсцис. Якщо f(a) \* f(c) > 0 - це означає, що корінь рівняння не знаходиться на відрізку [a;c] і відповідно буде знаходитися на відрізку [c;b], тому виконується наступне f(c) \* f(b) < 0;
5. проводимо пряму через точки, визначені в кроці 4;
6. виконуємо наступні кроки наведені в пунктах 3, 4 та 5 доти, поки отримаємо задану точність.

**Хід виконання роботи програми:**

1. введення даних користувачем для ініціалізації змінних(left\_border, right\_border), що відповідають за початковий проміжок знаходження кореня;
2. перевірка на коректність введених даних, а саме перевірка заданого проміжку на коректність за допомогою функції Is\_Correct\_Data() (рис.2). В разі введення проміжка з некоректними межами, функція змінює межі місцями та виводить інформацію про це користувачу;
3. локалізація кореня на відрізку за допомогою функції First\_Locale() (рис.3);
4. якщо корінь знайдено передаємо відрізок з ним у функцію - Localization\_with\_chord() (рис.10), яка описує алгоритм методу хорд (див. вище), інакше виводиться повідомлення про помилку. Інформація про роботу функції виводиться на екран під час кожної її ітерації, допоки отримаємо задану точність;
5. результат виконання програми (рис.11).

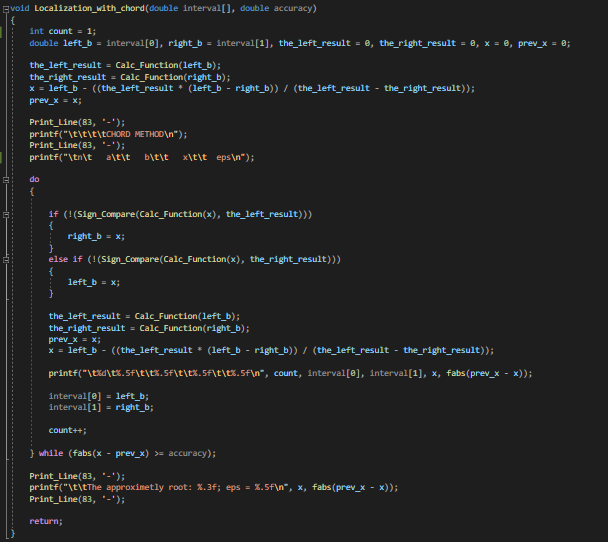


Рис.10. Опис функції Localization\_with\_chord

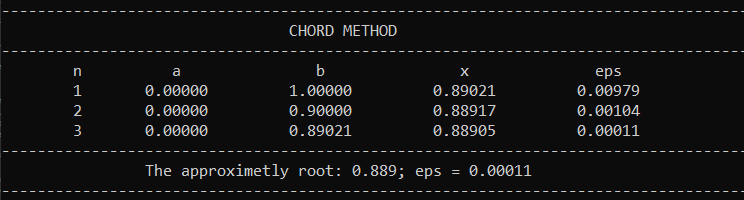


Рис.11. Результат виконання програми

**Визначення кореня за методом хорд**

1. локалізуємо корінь рівняння: - 3 + 12x - 9 = 0, в випадку з даним рівнянням, корінь знаходиться на відрізку [A;B] (рис. 6);
2. через кінці відрізка [A;B], на якому знаходиться корінь проводимо пряму та шукаємо перетин прямої з віссю абсцис, точка C (рис.12);
3. знаходимо x координату точки перетину;
4. знаходимо значення функції в цій точці (рис.13);
5. перевіряємо знаки значень функції в точках A та C і C та B та визначаємо рухомий кінець;
6. проводимо нову пряму через точки, які визначили в 5 пункті та шукаємо її перетин з віссю абсцис (точка E)(рис.14);
7. знаходимо значення функції в точці E та повторюємо дії, які описані в пунктах 5, 6 з відповідними точками доти, поки не отримаємо задану точність;
8. після третьої ітерації корінь був визначений з заданою точністю.

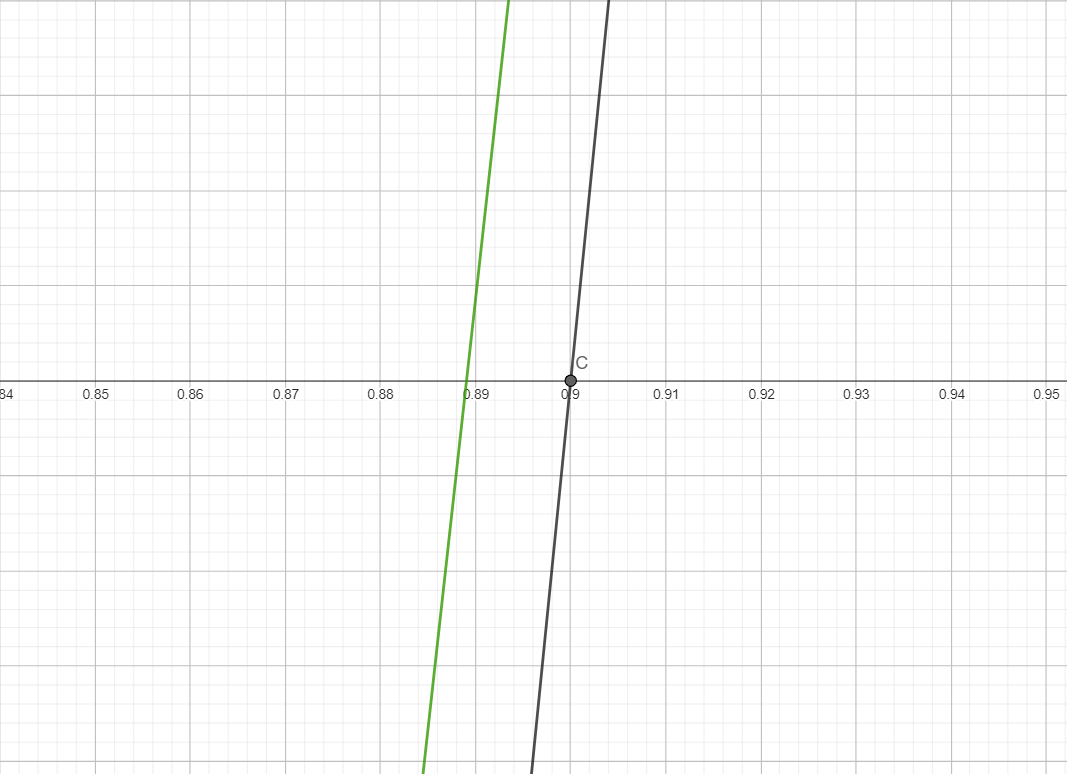


Рис.12. Перетин прямої з віссю абсцис в точці C

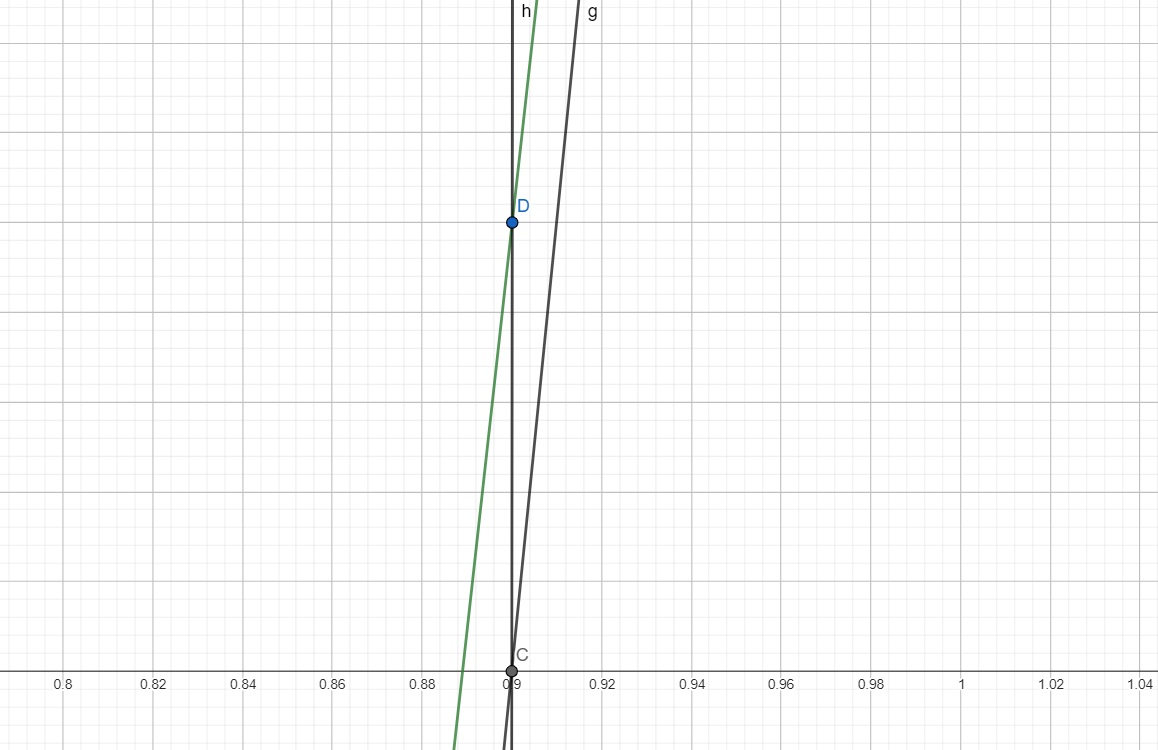


Рис.13. Значення функції в точці C

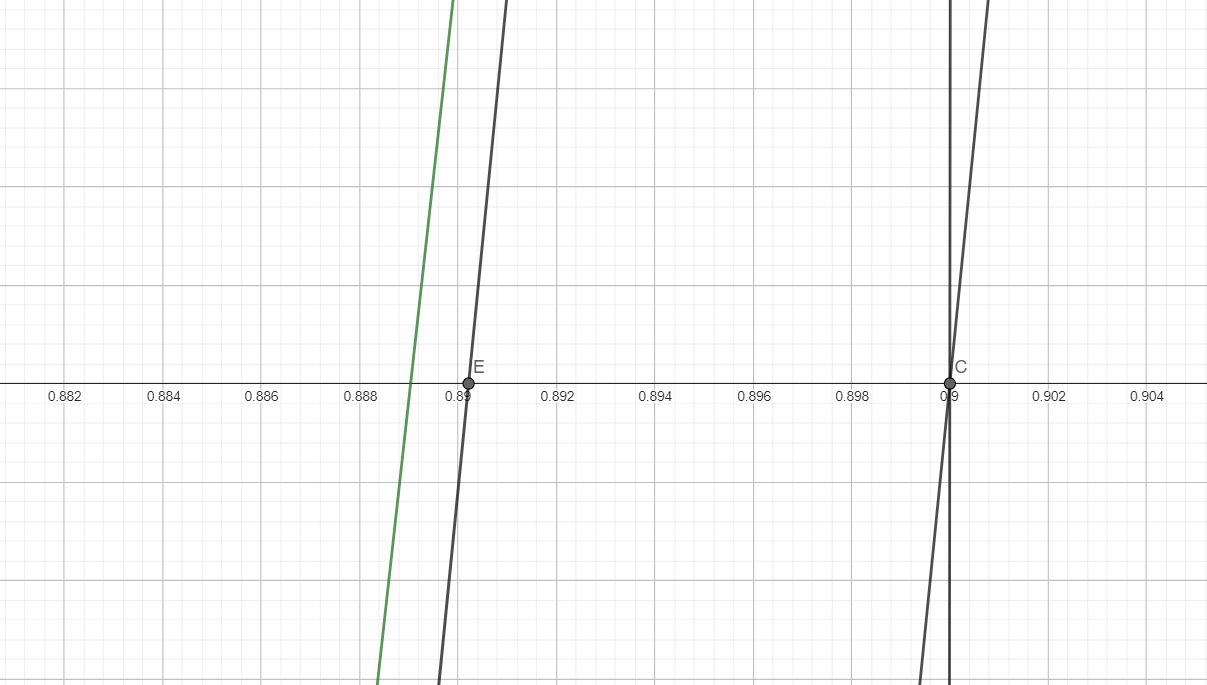


Рис.14. Перетин прямої проведеної через точки A та D з віссю абсцис в точці E

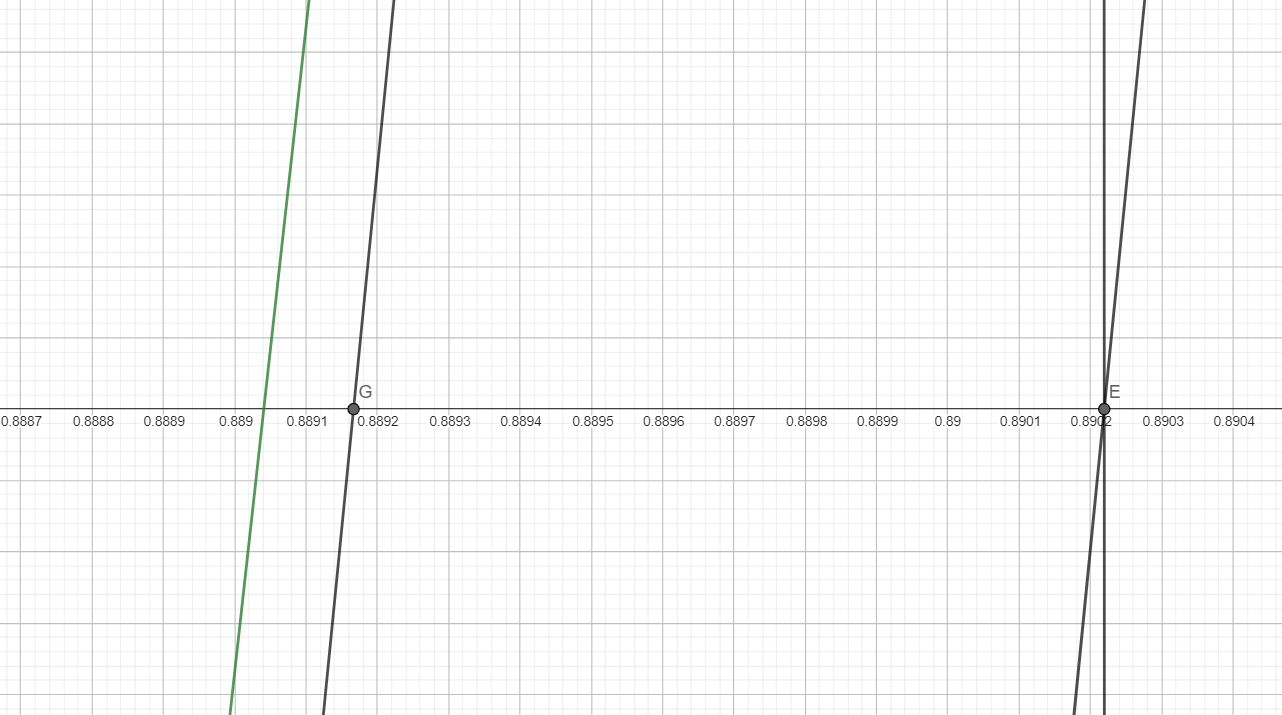


Рис.15. Остання ітерація. Перетин прямої проведеної через точки A та F з віссю абсцис в точці G.

**Висновки**

У результаті виконання лабораторної роботи, визначено дійсний корінь нелінійного алгебраїчного рівняння: - 3 + 12x - 9 = 0 з точністю методом дихотомії та методом хорд. Задана точність отримана методом дихотомії за 10 ітерацій, а за методом хорд за 3 ітерації.

**Додаток**

Назва файлу: **lab1.c**

#include <stdio.h>

#include <Windows.h>

#include <math.h>

double Calc\_Function(double);

void Is\_Correct\_Data(double\*, double\*);

void First\_Locale(double[], double);

int Sign\_Compare(double, double);

void Print\_Interval(double[], double);

void Localization\_with\_dividing(double[], double);

void Localization\_with\_chord(double[], double);

void Print\_Line(int, char);

int main()

{

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

int method = 0, pow\_of\_accuracy = 0;

double left\_border = 0, right\_border = 0, interval[2] = { 0 };

const double ACCUARACY = 0.001;

printf("Please! Enter the method of solving expression:\n1)Dichotomy method(Метод ділення відрізка навпіл)\n2)Chord method(Метод хорд)\nYour choice - ");

scanf\_s("%d", &method);

printf("Please! Enter the interval of boundaries(from ... to ...)(Enter two numbers through space: 12 99) - ");

scanf\_s("%lf", &left\_border);

scanf\_s("%lf", &right\_border);

Is\_Correct\_Data(&left\_border, &right\_border);

interval[0] = left\_border;

interval[1] = right\_border;

switch (method)

{

case 1:

printf("Dichotomy method, from %.3f to %.3f with accuracy %.3f:\n", left\_border, right\_border, ACCUARACY);

do

{

First\_Locale(interval, pow(10, pow\_of\_accuracy));

} while (interval[0] == 100 && interval[1] == -100);

Print\_Interval(interval, pow(10, pow\_of\_accuracy));

Localization\_with\_dividing(interval, ACCUARACY);

break;

case 2:

printf("Chord method, from %.3f to %.3f with accuracy %.3f:\n", left\_border, right\_border, ACCUARACY);

do

{

First\_Locale(interval, pow(10, pow\_of\_accuracy));

} while (interval[0] == 100 && interval[1] == -100);

Print\_Interval(interval, pow\_of\_accuracy);

Localization\_with\_chord(interval, ACCUARACY);

break;

default:

printf("Ooops! The program has finished with issues!");

break;

}

}

double Calc\_Function(double number)

{

double result = (pow(number, 3) - 3 \* pow(number, 2) + 12 \* number - 9);

return result;

}

void Is\_Correct\_Data(double\* number1, double\* number2)

{

double temp = 0;

if (\*number2 < \*number1)

{

temp = \*number1;

\*number1 = \*number2;

\*number2 = temp;

printf("Ooops! Incorrect input data, the numbers were swapped! The new interval from %.3f to %.3f\n", \*number1, \*number2);

}

else

{

printf("The input data is correct!\n");

}

}

void First\_Locale(double interval[], double accuracy)

{

double left\_b = interval[0], right\_b = interval[1];

double function = 0, the\_result = Calc\_Function(left\_b), previous\_x = 0;

previous\_x = left\_b;

left\_b += accuracy;

while (left\_b <= right\_b)

{

function = Calc\_Function(left\_b);;

interval[0] = previous\_x;

interval[1] = left\_b;

if (!(Sign\_Compare(the\_result, function)))

{

return;

}

else if (the\_result == 0 || function == 0)

{

interval[1] = left\_b + accuracy;

return;

}

the\_result = function;

previous\_x = left\_b;

left\_b += accuracy;

}

interval[0] = -100.0;

interval[1] = -100.0;

return;

}

int Sign\_Compare(double number1, double number2)

{

return (number1 > 0.0f) == (number2 > 0.0f);

}

void Print\_Interval(double borders[], double accuracy)

{

if (borders[0] == -100 && borders[1] == -100)

{

printf("Oops! Something went wrong...\n");

}

else

{

printf("The root is on the interval - [%.3f;%.3f]\n\n", borders[0], borders[1]);

}

}

void Localization\_with\_dividing(double interval[], double accuracy)

{

int count = 1;

double the\_avg = 0, left\_b = interval[0], right\_b = interval[1];

the\_avg = (interval[0] + interval[1]) / 2;

double the\_left\_result = Calc\_Function(interval[0]);

double the\_right\_result = Calc\_Function(interval[1]);

double the\_avg\_result = Calc\_Function(the\_avg);

Print\_Line(83, '-');

printf("\t\t\t\tDICHOTOMY METHOD\n");

Print\_Line(83, '-');

printf("\tn\t   a\t\t   b\t\t   x\t\t   f(x)\n");

while (fabs(left\_b - right\_b) >= accuracy)

{

if (!(Sign\_Compare(the\_left\_result, the\_avg\_result)))

{

right\_b = the\_avg;

}

else if (!(Sign\_Compare(the\_right\_result, the\_avg\_result)))

{

left\_b = the\_avg;

}

printf("\t%d\t%.5f\t\t%.5f\t\t%.5f\t\t%.5f\n", count, interval[0], interval[1], the\_avg, Calc\_Function(the\_avg));

interval[0] = left\_b;

interval[1] = right\_b;

the\_avg = (interval[0] + interval[1]) / 2;

the\_left\_result = Calc\_Function(interval[0]);

the\_right\_result = Calc\_Function(interval[1]);

the\_avg\_result = Calc\_Function(the\_avg);

count++;

}

Print\_Line(83, '-');

printf("\t\tThe approximetly root: %.3f; |a-b| = %.5f\n", the\_avg, fabs(left\_b - right\_b));

Print\_Line(83, '-');

return;

}

void Localization\_with\_chord(double interval[], double accuracy)

{

int count = 1;

double left\_b = interval[0], right\_b = interval[1], the\_left\_result = 0, the\_right\_result = 0, x = 0, prev\_x = 0;

the\_left\_result = Calc\_Function(left\_b);

the\_right\_result = Calc\_Function(right\_b);

x = left\_b - ((the\_left\_result \* (left\_b - right\_b)) / (the\_left\_result - the\_right\_result));

prev\_x = x;

Print\_Line(83, '-');

printf("\t\t\t\tCHORD METHOD\n");

Print\_Line(83, '-');

printf("\tn\t   a\t\t   b\t\t   x\t\t  eps\n");

do

{

if (!(Sign\_Compare(Calc\_Function(x), the\_left\_result)))

{

right\_b = x;

}

else if (!(Sign\_Compare(Calc\_Function(x), the\_right\_result)))

{

left\_b = x;

}

the\_left\_result = Calc\_Function(left\_b);

the\_right\_result = Calc\_Function(right\_b);

prev\_x = x;

x = left\_b - ((the\_left\_result \* (left\_b - right\_b)) / (the\_left\_result - the\_right\_result));

printf("\t%d\t%.5f\t\t%.5f\t\t%.5f\t\t%.5f\n", count, interval[0], interval[1], x, fabs(prev\_x - x));

interval[0] = left\_b;

interval[1] = right\_b;

count++;

} while (fabs(x - prev\_x) >= accuracy);

Print\_Line(83, '-');

printf("\t\tThe approximetly root: %.3f; eps = %.5f\n", x, fabs(prev\_x - x));

Print\_Line(83, '-');

return;

}

void Print\_Line(int number, char symbol)

{

while (number > 0)

{

printf("%c", symbol);

number--;

}

printf("\n");

}