Міністерство освіти і науки України
Національний технічний університет України
"Київський політехнічний інститут імені Ігоря Сікорського"
Фізико-технічний інститут

«Методи обчислень»

Лабораторна робота №1

Варіант 2

«Розв'язання нелінійних рівнянь»

Виконала:
студентка групи ФБ-95
Гурджия Валерія Вахтангівна

Київ – 2022

Завдання

1. Здійснити в якості допрограмового етапу аналіз та відокремлення коренів за допомогою теорем. Зокрема, визначити кількість дійсних коренів рівняння (теорема Гюа, теорема Штурма), відокремити дійсні корені рівняння (теорема про верхню межу). До аналізу комплексних коренів застосувати теорему про кільце. Результатом цього етапу повинна бути послідовність проміжків, кожен із яких містить лише один дійсний корінь рівняння.

Примітка. Щоб полегшити знаходження поліномів Штурма, дозволяється використовувати функцію deconv() системи Matlab, яка реалізує ділення поліномів із залишком.

2. Програмний етап полягає в тому, щоб уточнити корені рівняння методом бісекції, методом хорд, методом дотичних.
3. Порівняти отримані результати, зробити висновки, який метод приводить до меншої кількості ітерацій і чим це зумовлено.

2	$x^5 - 3x^4 + 7x^2 - 3 = 0$

Допрограмний етап

Теорема про границі усіх коренів (комплексних) рівняння

Нехай $A = \max|a_i|$, i=0,…,n-1; $B = \max|a_i|$, i=1,…,n.

Тоді всі (комплексні) корені рівняння (1) лежать у кільці

$$\frac{|a_0|}{B + |a_0|} \le |x^*| \le \frac{|a_n| + A}{|a_n|}$$

A = max(3,7, 3) = 7   **B** = max(1, 3, 7) = 7

$$r = \frac{3}{3+7} \le |xk| \le \frac{1+7}{1} = R$$

$r = 0.3$

$R = 8$

Теорема про верхню межу додатніх коренів

Нехай $B = \max_i |a_i|$, $a_i < 0$;

$m = \max i : a_i < 0$.

Тоді $R = 1 + \sqrt[n-m]{\frac{B}{a_n}}$ - верхня межа додатніх коренів: $\forall x^* \le R, f(x^*) = 0$.

Для визначення нижньої межі додатніх коренів стосовно вихідного поліному робимо заміну $x = 1/y$, для верхньої та нижньої межі від'ємних коренів – відповідно заміни $x = -1/y$ та $x = -y$.

$$B = \max(3{,}3) = 3$$

$$m = \max(4{,}0) = 4$$

$$R = 1 + \sqrt[5-4]{\frac{3}{1}} = 1 + 3 = 4$$

Верхня межа додатніх коренів: $x^+ \le 4$

Для пошуку нижньої додатної границі робимо заміну $P(\frac{1}{x})$:

$$x^n * \ P_n\left(\frac{1}{x}\right) = x^5 * P_5\left(\frac{1}{x}\right) = x^5 * \left(\frac{1}{x^5} - \frac{3}{x^4} + \frac{7}{x^2} - 3\right) = \ -3x^5 + 7x^3 - 3x + 1$$
$$= \ 3x^5 - 7x^3 + 3x - 1 = 0$$

$$B = \max(7{,}1) = 7$$

$$m = \max(3{,}0) = 3$$

$$R = 1 + \sqrt[5-3]{\frac{7}{3}} = 1 + \sqrt[2]{2.33} = 2.527$$

Нижня межа додатніх коренів: $x^+ \geq \frac{1}{2.527} = 0.443$

Для пошуку нижньої межі від'ємних коренів, робимо заміну $P(-$ x):

$$P_5(-x) = -x^5 - 3x^4 + 7x^2 - 3 = x^5 + 3x^4 - 7x^2 + 3 = 0$$

$$B = \max(7) = 7$$

$$m = \max(2) = 2$$

$$R = 1 + \sqrt[5-2]{\frac{7}{1}} = 1 + \sqrt[3]{7} = 2.913$$

Нижня межа від'ємних коренів: $x^- \geq -2.913$

Для пошуку верхньої межі від'ємних коренів, робимо заміну $P(-\frac{1}{x})$:

$$x^n * \ P_n\left(-\frac{1}{x}\right) = x^5 * P_5\left(-\frac{1}{x}\right) = x^5 * \left(-\frac{1}{x^5} - \frac{3}{x^4} + \frac{7}{x^2} - 3\right) = \ -3x^5 + 7x^3 - 3x - 1$$
$$= \ 3x^5 - 7x^3 + 3x + 1 = 0$$

$$B = \max(7) = 7$$

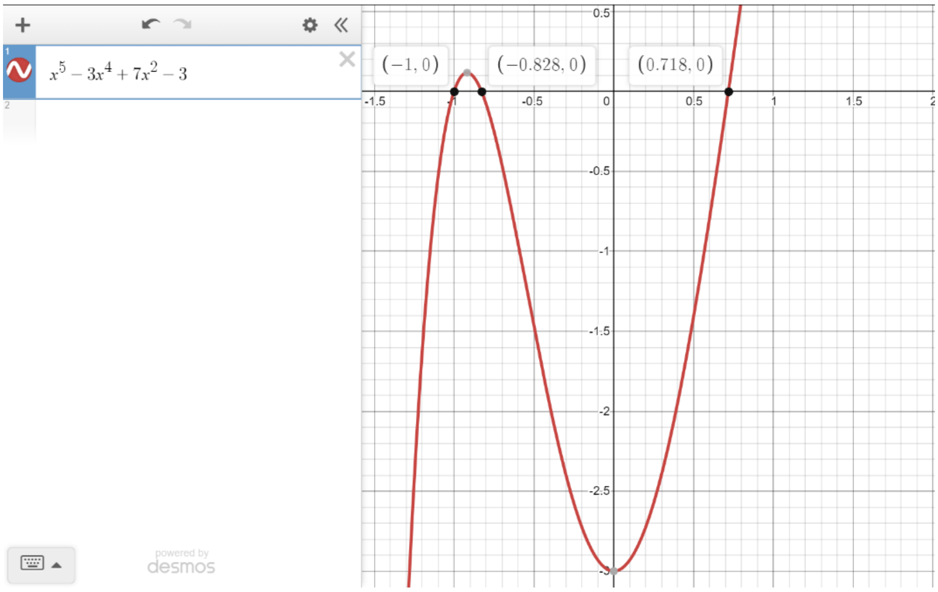
$$m = \max(3) = 3$$

$$R = 1 + \sqrt[5-3]{\frac{7}{3}} = 1 + \sqrt[2]{2.33} = 2.527$$

Верхня межа від'ємних коренів: $x^- \leq -\frac{1}{2.527} = -0.443$

$$0.443 \le x^+ \le 4$$

$$-2.913 \le x^- \le -0.443$$



Теорема Гюа про наявність комплексних коренів

Якщо $\exists\ k$: $0 < k < n$, $a^2_k < a_{k-1} \cdot a_{k+1}$, то рівняння має хоча б одну пару комплексноспряжених коренів.

Нехай k=1: $0^2 > -3 * 7$

Нехай k=2: $7^2 > 0 * 0$

Нехай k=3: $0^2 > 7 * (-3)$

Нехай k=4: $(-3)^2 > 1 * 0$

Усі корені рівняння є дійсними числами

За теоремою Штурма про чередування коренів

$P_0(x) = x^5 - 3x^4 + 7x^2 - 3$

$P_1(x) = 5x^4 - 12x^3 + 14x$

$P_2(x) = \frac{36}{25}x^3 - \frac{21}{5}x^2 - \frac{42}{25}x + 3$

$$P_3(x) = -\frac{1925}{144}x^2 - \frac{475}{72}x + \frac{775}{144}$$

$$P_4(x) = -\frac{196128}{148225}x - \frac{151632}{148225}$$

$$P_5(x) = -\frac{53919956475}{21637233216}$$

Таблиця знаків полінома Штурма:

	-2.913	*-0.443*	*0.443*	*4*
$P_0(x)$	–	–	–	+
$P_1(x)$	+	–	+	+
$P_2(x)$	–	+	+	+
$P_3(x)$	–	+	–	–
$P_4(x)$	+	–	–	–
$P_5(x)$	–	–	–	–
Кількість змін знаку	***4***	***2***	***2***	***1***
Кількість коренів	***2***		***1***	

Таким чином існує 3 дійсні корені

Програмний етап

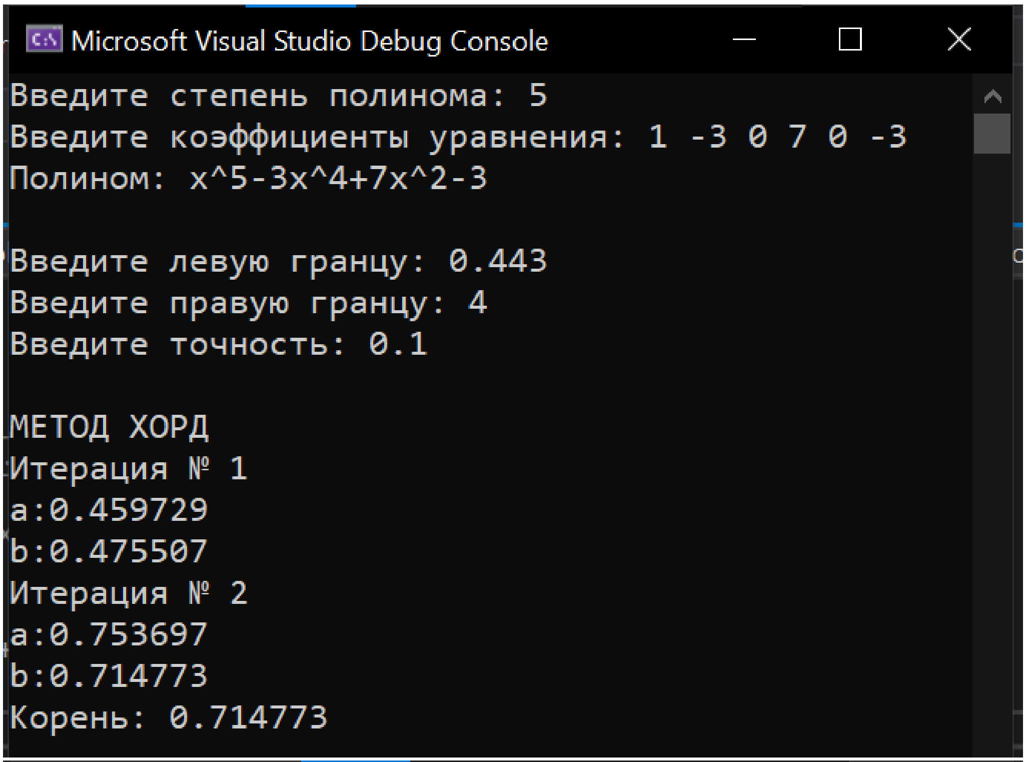
1) Метод бісекції

```
Microsoft Visual Studio Debug Console
Введите степень полинома: 5
Введите коэффициенты уравнения: 1 -3 0 7 0 -3
Полином: x^5-3x^4+7x^2-3

Введите левую границу: 0.443
Введите правую границу: 4
Введите точность: 0.01

МЕТОД БИСЕКЦИИ
Итерация № 1
a: 0.443
b: 4
c: 2.2215
false
f(a)*f(c) = -21.7061
Итерация № 2
a: 0.443
b: 2.2215
c: 1.33225
false
f(a)*f(c) = -7.19289
Итерация № 3
a: 0.443
b: 1.33225
c: 0.887625
false
f(a)*f(c) = -2.07639
Итерация № 4
a: 0.443
b: 0.887625
c: 0.665312
false
f(a)*f(c) = 0.619098
Итерация № 5
a: 0.665312
b: 0.887625
```

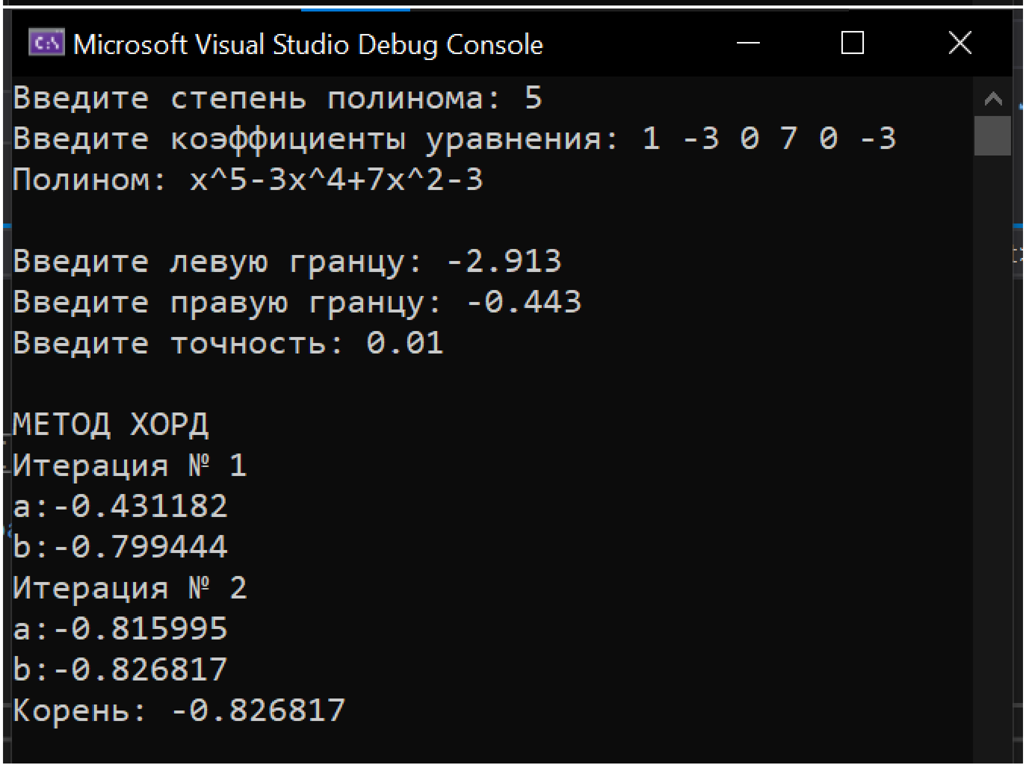
```
Microsoft Visual Studio Debug Console
Итерация № 5
a: 0.665312
b: 0.887625
c: 0.776469
false
f(a)*f(c) = -0.14792
Итерация № 6
a: 0.665312
b: 0.776469
c: 0.720891
false
f(a)*f(c) = -0.00799116
Итерация № 7
a: 0.665312
b: 0.720891
c: 0.693102
false
f(a)*f(c) = 0.0608944
Итерация № 8
a: 0.693102
b: 0.720891
c: 0.706996
false
f(a)*f(c) = 0.0125517
Итерация № 9
a: 0.706996
b: 0.720891
c: 0.713943
false
f(a)*f(c) = 0.00191876
Итерация № 10
a: 0.713943
b: 0.720891
c: 0.717417
true
Корень: 0.717417
```

Всього 10 ітерацій

2) Метод хорд



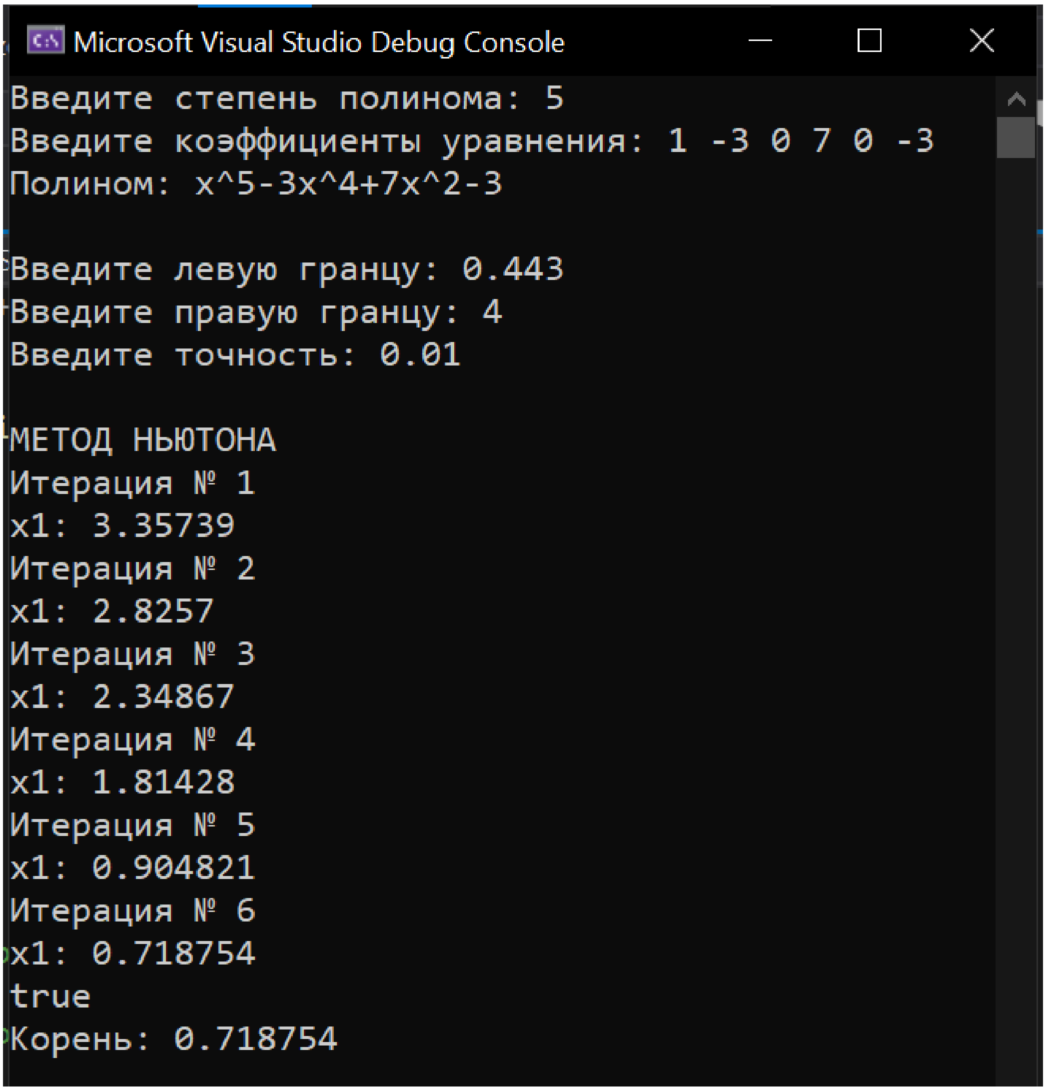
```
Microsoft Visual Studio Debug Console
Введите степень полинома: 5
Введите коэффициенты уравнения: 1 -3 0 7 0 -3
Полином: x^5-3x^4+7x^2-3

Введите левую границу: 0.443
Введите правую границу: 4
Введите точность: 0.1

МЕТОД ХОРД
Итерация № 1
a:0.459729
b:0.475507
Итерация № 2
a:0.753697
b:0.714773
Корень: 0.714773
```



```
Microsoft Visual Studio Debug Console
Введите степень полинома: 5
Введите коэффициенты уравнения: 1 -3 0 7 0 -3
Полином: x^5-3x^4+7x^2-3

Введите левую границу: -2.913
Введите правую границу: -0.443
Введите точность: 0.01

МЕТОД ХОРД
Итерация № 1
a:-0.431182
b:-0.799444
Итерация № 2
a:-0.815995
b:-0.826817
Корень: -0.826817
```

Всього 2 ітерації

3) Метод Ньютона



Всього 6 ітерацій

Microsoft Visual Studio Debug Console

```
Введите степень полинома: 5
Введите коэффициенты уравнения: 1 -3 0 7 0 -3
Полином: x^5-3x^4+7x^2-3

Введите левую границу: -2.913
Введите правую границу: -0.443
Введите точность: 0.01

МЕТОД НЬЮТОНА
Итерация № 1
x1: -2.31325
Итерация № 2
x1: -1.85944
Итерация № 3
x1: -1.52677
Итерация № 4
x1: -1.2942
Итерация № 5
x1: -1.14263
Итерация № 6
x1: -1.05419
Итерация № 7
x1: -1.01258
Итерация № 8
x1: -1.00096
true
Корень: -1.00096
```

Всього 8 ітерацій

Код програми

```cpp
#include <iostream>
#include <cmath>
#include <string>
#include <vector>
using namespace std;
vector<int> coeficients;

void ShowPolinom(vector<int> coef) {
    cout << "Полином: ";
    for (int i = coef.size() - 1; i >= 0; i--) {
        if ((coef[i] < 0) || (i == coef.size() - 1)) {
            if (coef[i] != 0 && coef[i] != 1) {
                if (i > 1) {
                    cout << coef[i] << "x^" << i;
                }
                else if (i == 1) {
                    cout << coef[i] << "x";
                }
                else if (i == 0) {
                    cout << coef[i] << endl;
                }
            }
            else if (coef[i] == 1) {
                if (i > 1) {
                    cout << "x^" << i;
                }
                else if (i == 1) {
                    cout << "x";
                }
                else if (i == 0) {
                    cout << endl;
                }
            }
        }
        else {
            if (coef[i] != 0 && coef[i] != 1) {
                if (i > 1) {
                    cout << "+" << coef[i] << "x^" << i;
                }
                else if (i == 1) {
                    cout << "+" << coef[i] << "x";
                }
                else if (i == 0) {
                    cout << "+" << coef[i] << endl;
                }
            }
            else if (coef[i] == 1) {
                if (i > 1) {
                    cout << "+" << "x^" << i;
                }
                else if (i == 1) {
                    cout << "+" << "x";
                }
                else if (i == 0) {
                    cout << endl;
                }
            }
        }
    }
    cout << endl;
```

```
}

float Polinom(vector<int> coef,  float x) {

    float sum = 0;
    for (int i = 0; i < coef.size(); i++) {
        sum += pow(x, i) * coef[i];
    }
    return sum;
}


vector<int> Pohidna(vector<int> coef) {
    vector<int> pohidna;
    for (int i = 1; i < coef.size(); i++) {
        pohidna.push_back(i * coef[i]);
    }
    return pohidna;
}


int i = 0;
float BesectionMethod(vector<int> coef, float a, float b, float epsilon) {
    float c = (a + b) / 2;
    float fc = Polinom(coef, c);
    float fa = Polinom(coef, a);
    float fb = Polinom(coef, b);

    if (fa * fb < 0) {
        cout << "Итерация № " << ++i << endl;
        cout << "a: " << a << "\nb: " << b << "\nc: " << c << endl;

        if (abs(fc) > epsilon) {
            cout << "false\n";
            cout << "f(a)*f(c) = " << fa * fc << endl;
            if (fa * fc < 0) {
                BesectionMethod(coef, a, c, epsilon);
            }
            else if (fa * fc > 0) {
                BesectionMethod(coef, c, b, epsilon);
            }
        }
        else {
            cout << "true\n";
            cout << "Корень: " << c << endl;
            return c;
        }
    }
    else {
        cout << "Не сходится\n";
    }
}


float ChordMethod(vector<int> coef, float a, float b, float epsilon) {
    while (abs(Polinom(coef, b)) > epsilon)
    {
        a = b - ((b - a) * Polinom(coef, b)) / (Polinom(coef, b) - Polinom(coef, a));
        b = a - ((a - b) * Polinom(coef, a)) / (Polinom(coef, a) - Polinom(coef, b));
        cout << "Итерация № " << ++i << "\na:" << a << "\nb:" << b << endl;
    }
    cout << "Корень: " << b << endl;
```

```
    return b;
}

void NewtonMethod(vector<int> coef, float a, float b, float epsilon) {
    float x;
    float fa = Polinom(coef, a) * Polinom(Pohidna(Pohidna(coef)), a);
    float fb = Polinom(coef, b) * Polinom(Pohidna(Pohidna(coef)), b);
    if (Polinom(coef, a) * Polinom(Pohidna(Pohidna(coef)), a) > 0) {
        x = a;
    }
    else if (Polinom(coef, b) * Polinom(Pohidna(Pohidna(coef)), b) > 0) {
        x = b;
    }
    else {
        cout << "Не сходится\n";
        return;
    }
    int i = 0;
    while (abs(Polinom(coef, x)) > epsilon) {
        float x1 = x - Polinom(coef, x) / Polinom(Pohidna(coef), x);
        cout << "Итерация № " << ++i << endl;
        cout << "x1: " << x1 << endl;
        x = x1;
    }
    cout << "true\n";
    cout << "Корень: " << x << endl;
}

int main()
{
    setlocale(LC_ALL, "ru");
    int stepinPolinomy;
    cout << "Введите степень полинома: ";
    cin >> stepinPolinomy;
    cout << "Введите коэффициенты уравнения: ";
    for (int i = 0; i <= stepinPolinomy; i++) {
        int coef;
        cin >> coef;
        coeficients.insert(coeficients.begin() + 0, coef);
    }
    ShowPolinom(coeficients);

    float a, b, epsilon;
    cout << "Введите левую границу: ";
    cin >> a;
    cout << "Введите правую границу: ";
    cin >> b;
    cout << "Введите точность: ";
    cin >> epsilon;

    cout << "\nМЕТОД БИСЕКЦИИ\n";
    BesectionMethod(coeficients, a, b, epsilon);
    cout << "\nМЕТОД ХОРД\n";
    ChordMethod(coeficients, a, b, epsilon);
    cout << "\nМЕТОД НЬЮТОНА\n";
    NewtonMethod(coeficients, a, b, epsilon);

    return 0;
}
```

Висновок: під час виконання лабораторної роботи я навчилася шукати проміжки, на яких лежать корені, та програмно знаходити корені на них. Я помітила, що найменш точним є метод хорд, а найбільш точним є метод Ньютона.