Національний технічний університет України
«Київський політехнічний інститут»
Факультет інформатики і обчислювальної техніки
Кафедра обчислювальної техніки

# Лабораторна робота №4
## З алгоритмів та методів обчислень
Варіант 24

*Виконав:*
Студент групи ІО-32
Попенко Р. Л.
*Перевірив:*
Порєв В. М.

Київ - 2015 р.

**1. Тема завдання:**

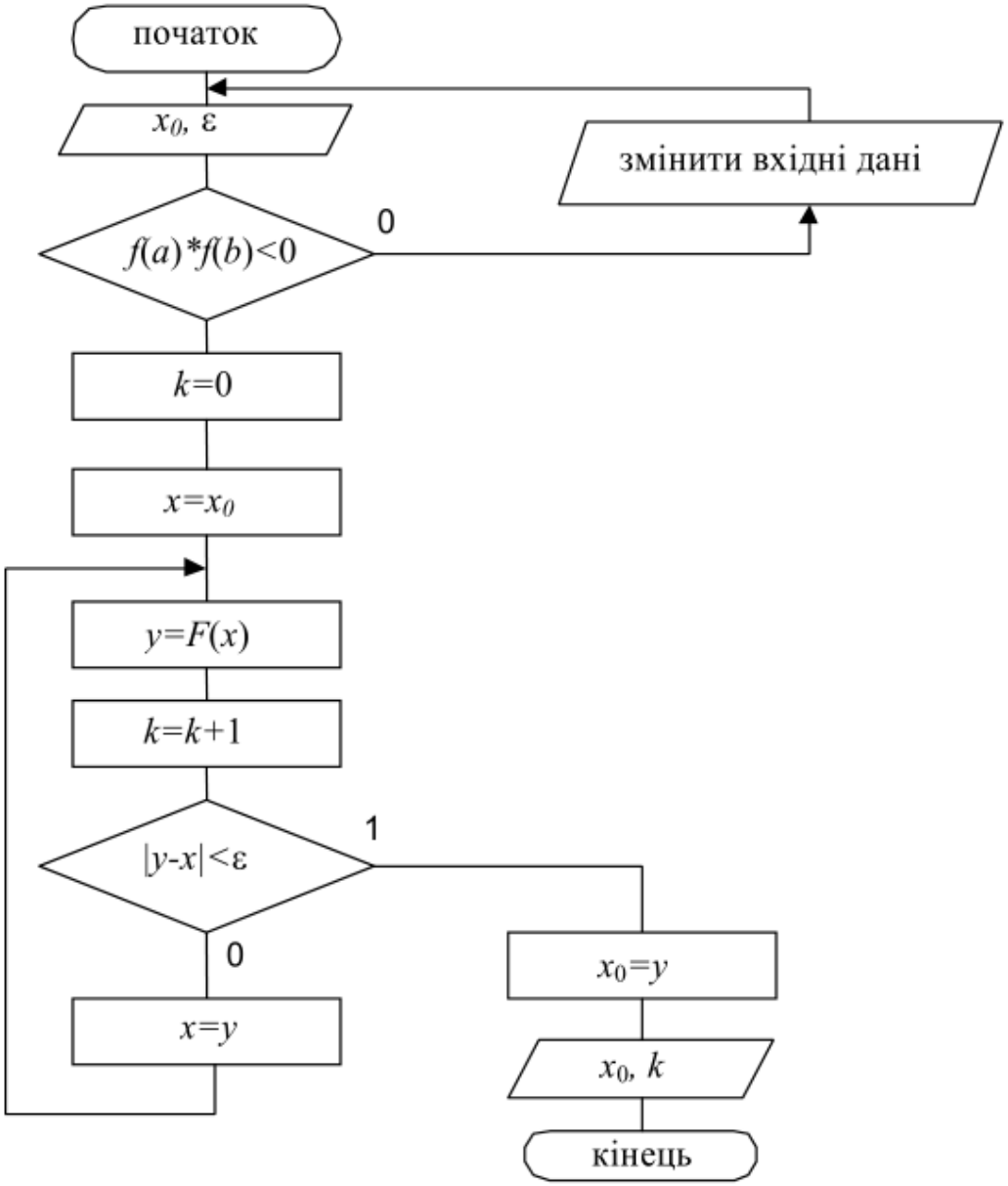
Закріплення знань студентів при вирішенні практичних завдань з розв’язування нелінійних рівнянь. Оволодіння методами і практичними навичками розв’язування нелінійних рівнянь на ЕОМ. Набуття умінь і навичок при програмуванні та налагодженні програм для розв’язування нелінійних рівнянь на комп'ютері.

**2. Завдання:**

Скласти програму розв’язання нелінійного рівняння, користуючись схемою алгоритму.

| Метод | Номер варіанту | Рівняння | Примітка |
|---|---|---|---|
| Метод ітерацій | 24 | $lnx + (x+1)^3 = 0$ | 0.187 |

**3. Блок схема методу ітерацій:**



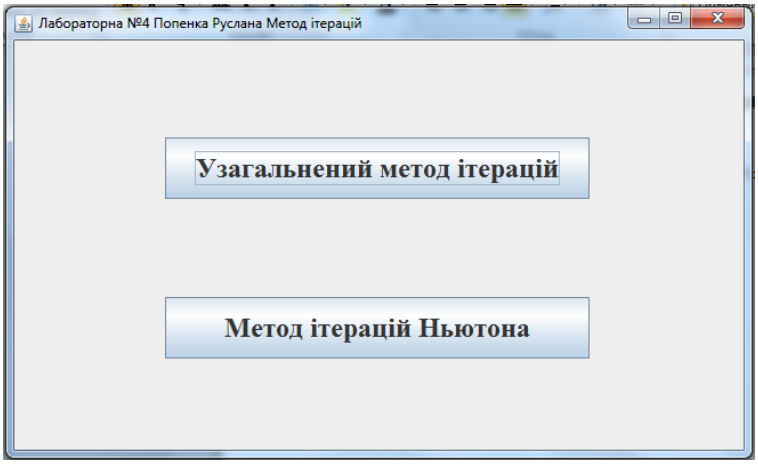

**4. Лістинг програми:**

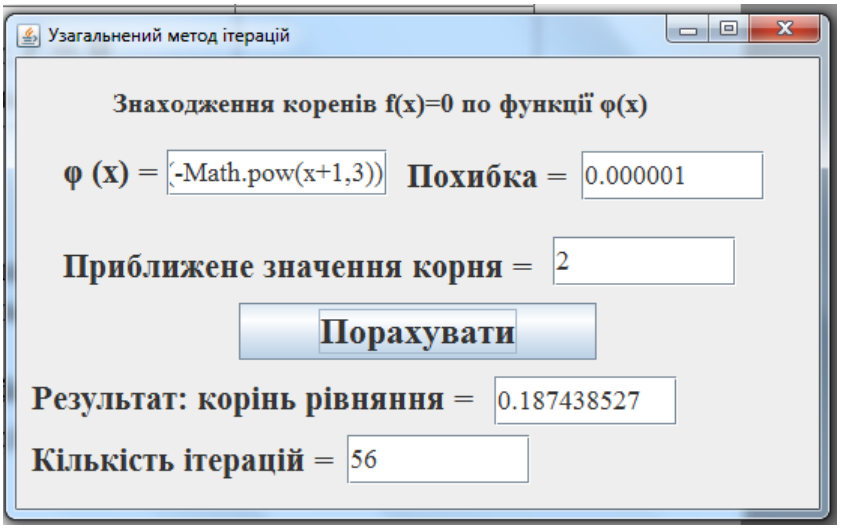
```
package Lab4;
import bsh.EvalError;
import bsh.Interpreter;


public class IterationRoot {
	private Interpreter interpreter = new Interpreter();

	private long iterats;
	private double result;
```

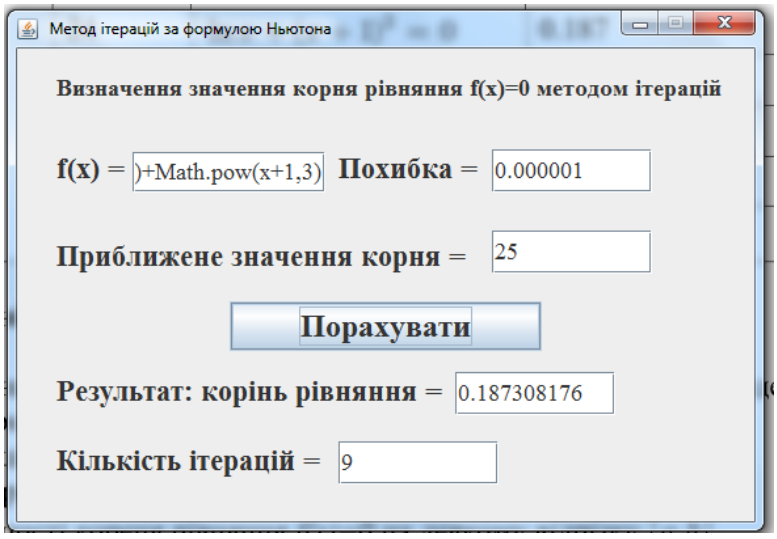
```
	IterationRoot (String func, double x0, double eps){
		try {
			interpreter.eval("f (x) { return "+func+";}");

		} catch (EvalError e) {

			e.printStackTrace();
			int a=1/0;
		}
		iterationRoot(x0,eps);
	}

	IterationRoot (String fi, double x0, double eps, boolean b){
		try {
			interpreter.eval("fi (x) { return "+fi+";}");

		} catch (EvalError e) {

			e.printStackTrace();
			int a=1/0;
		}
		iterationRootOrig(x0, eps);
	}
	private double f (double x){

		try {
			interpreter.eval("result = f ("+Double.toString(x)+");");
			return (double) interpreter.get("result");
		} catch (EvalError e) {
			// TODO Auto-generated catch block
			e.printStackTrace();
			int a=1/0;
		}
		return 0;

	}

	private void iterationRootOrig(double x0,  double eps){
		long n=0;
	  double x=x0,y,b;
	  do {
	         y=fi(x);
	         b=Math.abs(x-y);
	         x=y;
	         n++;
	  }while (b>=eps && n<15000);
	  iterats=n-1;
	  result=x;
	  if(n==14999){
		int a=1/0;
	  }

	}

private double fi (double x){

		try {
			interpreter.eval("result = fi ("+Double.toString(x)+");");
			return (double) interpreter.get("result");
		} catch (EvalError e) {
			// TODO Auto-generated catch block
			e.printStackTrace();
			int a=1/0;
		}
```

```
		return 0;

	}

	public double roundResult (double value, int places) {
		if (places < 0) throw new IllegalArgumentException();

	    long factor = (long) Math.pow(10, places);
	    value = value * factor;
	    long tmp = Math.round(value);
	    return (double) tmp / factor;

	}

	private double df (double x){
		double dx=0.000000001;
		return roundResult((f(x+dx)-f(x))/dx,5);

	}

	private double phi (double x){
		return x-f(x)/df(x);
	}

	public boolean isCals (){
		if ((iterats==-1)||(iterats==0)){
			int a=1/0;
		}
		return true;
	}

	public boolean isCalsOrigin (){
		if ((iterats==-1)||(iterats==0)||(iterats==1)){
			int a=1/0;
		}
		return true;
	}

	public double getResult(){
		return result;
	}

	public long getIter(){
		return iterats;
	}

	private void iterationRoot (double x0,  double eps){
		double iter=0;
		double x=x0;
		double s=x0;
		 for (double i = 1.0; (eps<Math.abs(f(x))); i=i+1.0) {
			if(df(x) == 0)
			    break;
			s=x;
			iter=i;
			x=phi(x);
			if (i==1500.0){
				int a=5/0;
			}
		}
		iterats=(long) (iter-1.0);
		result=s;}

}
```

**5. Робота програми:**




**Аналіз результатів:**

Створена мною програма знаходить корені нелінійних рівнянь з вказаною точністю з указанням приближеного значення кореня. Я реалізував метод ітерацій класичний і той де ітерації реалізовані за допомогою формули дотичних Ньютона. В моїй програмі функції можна вводити при виконанні.