INSTITUTO TECNOLÓGICO DE NUEVO LEÓN

Lenguajes y Autómatas 2

UNIDAD # 1

EVIDENCIA # 1: Proyecto 1

Catedrático: Ing. Juan Pablo Rosas Baldazo

Nombre: Mauricio Tamez Botello, Diego Fernández

Guadalupe N.L. a 15 de 02 del 2018

Introducción

Este proyecto consta de realizar un programa que nos ayude a la evaluación de expresiones que incluyan operandos números y operadores matemáticos, como suma, resta, multiplicación y división.

También tiene como objetivo al ingresar dicha operación nos pueda dar como resultado los 3 tipos de órdenes, los cuales son: Prefija, Postfija e Infija, con la finalidad de poder ver como es el orden correcto de cada expresión.

Descripción

Se realizó un árbol binario en el cual consta de que cada nodo tiene 2 hijos, ya sean números, letras y operadores matemáticos.

Su objetivo es imprimir las 3 expresiones ya antes mencionadas en la introducción.

Pseudocodigo

```
Package Arbol;
```

```
import java.util.Scanner;
public class Nodo {
   public static void main(String[] args) {
      print ("Digite expresión que desea evaluar");
      String infija, infija1 = Expresión ingresada;
      print ("Resultado y ordenes de la expresión");
   }
   public static double evaluar (String infija, String infija1){
      String prefija= conversión expresión a prefija;
      String posfija = conversión expresión a posfija;
      print ("La expresión posfija es: " + posfija);
      print ("La expresión infija es: " + infija);
      print ("La expresión prefija es: " + prefija);

      return Resultado de la expresión;
   }

     private static String convertirpre(String infija) {
     pila = caracteres de la expresión;

     for (recorrido de la pila){
        char letra = separar carácter por carácter;
        if (si el carácter no es operador) {
           if (si la pila está vacía) {
         (entonces) se apila carácter en la pila;
           } en caso contrario {
              int pe = prioridad en la expresión del operando;
              int pp = prioridad en pila del siguiente operando;
           if (pe > pp){
               apilar operando en pila;
           } en caso contrario {
```

```
            prefija += se desapila el operando de la pila;
            se apila el operando en la pila e ingresa a prefija;
          }
        }
      } en caso contrario {
          prefija += el operador se ingresa a prefija;
      }
    }
  mientras (la pila no esté vacía) {
    prefija += se desapilan los operandos y se ingresan a prefija;
  }
  devuelve el orden de prefija;
}
```

```
    private convertirpos (String infija) {
   pila = número de caracteres de la expresión;
for (recorrido de la pila) {
      char letra = separar carácter por carácter;
      if (si el carácter es operador) {
        if (si la pila está vacía) {
       (entonces) se apila carácter en la pila;
        } en caso contrario {
          int pe = prioridad en la expresión del operador;
          int pp = prioridad en pila del operador;
        if (pe > pp){
           apilar operador en pila;
        } en caso contrario {
          posfija = se desopila el operador de la pila;
          se apila el operador en la pila e ingresa a posfija;
        }
       }
      } en caso contrario {
        posfija = el operando se ingresa a posfija;
      }
     }
   mientras (la pila no esté vacía) {
      posfija = se desapilan los operadores y se ingresan a posfija;
   }

   return devuelve el orden de posfija;
  }
    private static int prioridadEnExpresion (char operador) {
```

```
    if (operador == '^') devuelve ID# 4;
    if (operador == '*' || operador == '/') devuelve ID# 2;
    if (operador == '+' || operador == '-') devuelve ID# 1;
    if (operador == '(') devuelve ID# 5;
    return 0;
}
    private static int prioridadEnPila (char operador) {
    if (operador == '^') devuelve ID# 3;
    if (operador == '*' || operador == '/') devuelve ID# 2;
    if (operador == '+' || operador == '-') devuelve ID# 1;
    if (operador == '(') return 0;
    return 0;
}
  private static double evaluar (String posfija) {
  tamaño pila = cantidad de caracteres de la expresión;
  for (recorrido de la pila) {
    char letra = separar carácter por carácter;
    if (si el carácter no es operador) {
       double num = se apila operando en la pila;
       se apila numero en la pila;
    } en caso contrario {
        double num2 = se desapila el operando;

        double num1 = se desapila el operando;
        double num3 = resultado de la operación;
        apila resultado de la operación en la pila;
    }
  }
  return devuelve resultado de la expresión;
```

```
  }
  private static boolean esOperador (char letra) {
    Si es alguno de los operadores (letra == '*' || letra == '/' || letra == '+' ||
      letra == '-'||letra == '('||letra == ')' ||letra == '^') {
      responde verdadero;
  }
    En caso contrario responde falso;
  }
  private static double operacion (char letra, double num1, double num2) {
     si el operando (letra == '*') regresa una multiplicación;
     si el operando (letra == '/') regresa una división;
     si el operando (letra == '+') regresa una suma;
     si el operando (letra == '-') regresa una resta;
     si el operando (letra == '^') regresa una potencia;
     return 0;
  }
}
```

Arbol.java

```
package Arbol;
public class Arbol {


  se crea variable int n, tope;
  se crea objeto pila [];
  public Arbol (int n) {
     esta n es = tamaño de la pila;
     tope = 0;
     pila = tiene un tamaño = n;
  }
  public boolean estaVacia(){
```

```
    si esta vacía regresa un 0;
  }
  public boolean estaLlena(){
    si está llena regresa el número de elementos en la pila;
  }
  public boolean
    apilar(recibe un
    dato){ if(la pila esta
    llena){
      regr
      esa
       un
      fals
       o;
    }
en caso contrario
    posición de la
    pila actual = dato;
    posición de la
    pila avanza 1;
    regresa un
    verdadero;
  }
  public
    Object
    desapilar
    (){ if(la
    pila esta
    vacía) {
      regresa un valor nulo;
    }
```

```
        posición de la
        pila retrocede 1;
        regresa
        posición de la
        pila;
    }
    public Object
        elementoTope(){
        regresa posición
        anterior de la pila;
    }
}
```

- **Resultado**

Con este programa se obtuvo en pantalla la impresión de la operación ingresada por el usuario en sus respectivos órdenes.

- **Conclusión**

A mi criterio personal es un buen programa que nos ayuda a conocer el resultado final de cada recorrido que hace cada expresión.

El programa es fácil de realizar con la ayuda de videos, ya que en muchos sitios web muestran muchos ejemplos de cómo se realizan y los explica detalladamente cada uno.