INSTITUTO TECNOLÓGICO DE NUEVO LEÓN

Ingeniería En Sistemas Computacionales

Lenguajes y Autómatas II

U1

Tema: Análisis semántico

Proyecto 1

Profesor.
Juan Pablo Rosas Baldazo

Presenta.

Magaly Guadalupe Leija Sánchez

14480481

Índice

Introducción

Con este proyecto se quería lograr introducir una operación matemática que aceptara, operadores como ^, *, +, -. Y que nos imprimiera las notaciones prefija y posfija.

Y que se pudiera evaluar la operación.

Descripción

Para la realización de este proyecto estuvimos investigando y checando otros ejemplos ya hechos, muchos estaban estructurados de distintas maneras, pero tomamos como base el código de Juan Carlos Zuluaga el cual lo puso en un video de youtube.

Pseudocodigo

```
package evaluadorexpresiones;

import java.util.Scanner;


public class Evaluador {
    public static void main(String[] args) {
        print ("Digite expresión que desea evaluar");
        String infija, infija1 = Expresion ingresada;
        print ("Resultado y ordenes de la expresion");
    }
    public static double evaluar(String infija, String infija1){
        String prefija= convercion expresion a prefija;
        String posfija = convercion expresion a posfija;
        print("La exprecion posfija es: " + posfija);
        print("La exprecion infija es: " + infija);
        print("La exprecion prefija es: " + prefija);


        return Resultado de la expresion;
    }
       private static String convertirpre(String infija) {
       pila = caracteres de la expresion;
```

```
    for (recorrido de la pila){
        char letra = separar caracter por caracter;
        if(si el caracter no es operador){
            if(si la pila esta vacia){
         (entonces) se apila caracter en la pila;
            }en caso contrario{
                int pe = prioridad en la expresion 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 este vacia){
        prefija = se desapilan los operandos y se ingresan a prefija;
    }
    return devuelve el orden de prefija;
}
     private static String convertirpos(String infija) {
```

```
pila = caracteres de la expresion;
   for (recorrido de la pila){
    char letra = separar caracter por caracter;
    if(si el caracter  es operador){
       if(si la pila esta vacia){
     (entonces) se apila caracter en la pila;
       }en caso contrario{
          int pe = prioridad en la expresion del operador;
          int pp = prioridad en pila del siguiente operador;
       if(pe > pp){
           apilar operador en pila;
       }en caso contrario{
          posfija = se desapila 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 este vacia){
    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 expresion;
  for (recorrido de la pila){
    char letra = separar caracter por caracter;
    if(si el caracter 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 operacion;
```

```
            apila resultado de la operacion en la pila;
          }
        }
        return regresa resultado de la expresion;
    }
    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 num1 * num2;
        si el  operando(letra == '/') regresa num1 / num2;
        si el  operando(letra == '+') regresa num1 + num2;
        si el  operando(letra == '-') regresa num1 - num2;
        si el  operando(letra == '^') regresa Math.pow(num1, num2);
        return 0;
    }
}
```

```
package evaluadorexpresiones;
public class Pila {

    se crea variable int n, tope;
    se crea objeto pila[];
    public Pila(int n) {
        esta n es = n de la pila;
        tope = 0;
        pila = tiene un tamaño = n;
    }
    public boolean estaVacia(){
        si esta vacia regresa un 0;
    }
    public boolean estaLlena(){
        si esta llena regresa el numero de elementos en la pila;
    }
    public boolean apilar(resive un dato){
        if(la pila esta llena){
            regresa un falso;
        }
          en caso contrario
        posicion de la pila actual = dato;
        posicion de la pila avanza 1;
        regresa un verdadero;
    }
```

```
    public Object desapilar(){
        if(la pila esta vacia) {
            regresa un valor nulo;
        }
        posicion de la pila retrocede 1;
        regresa posicion de la pila;
    }
    public Object elementoTope(){
        regresa posicion anterior de la pila;
    }
}
```

Experimentación

Primero que nada los errores que nos marcaba al traspasarlo en nuestros equipos, fue un símbolo en while, estuvimos checando varios días hasta que volvimos a poner el video completo y nos percatamos que nos faltaba “!”.

Resultados

Se logró llegar al objetivo del proyecto, al ingresar alguna operación matemática.

Digite expresión que desea evaluar

2*1+5

La expresión posfija es: 21*5+

La expresión infija es: 2*1+5

La expresión prefija es: *2+15

El resultado es: 7.0

Conclusión

Para llegar a que nos diera el resultado, me di cuenta que hay que investigar mucho ya que algunos métodos no los conocía, hay que ver una buena porción de videos para lograr comprender bien la lógica.

Referencias (bibliografía)

Juan Carlos Zuluaga, 17 de abril 2015
https://www.youtube.com/watch?v=xKhA9W1fUZU&t=2016s

Juan Carlos Zuluaga, 20 de marzo 2014
https://www.youtube.com/watch?v=d7UZdz_yGXQ

S.A, 30 de mayo 2012
https://www.youtube.com/watch?v=wciAZR1BnSY