

# Actividades adicionales

## Algoritmos y Estructuras de Datos

### Tema 4: listas, pilas y colas

#### Actividad 4.7

Esta actividad es la continuación de la actividad anterior. El objetivo es escribir un programa que lea una expresión infija, lo convierta a postfija, y calcula e imprima el resultado final.

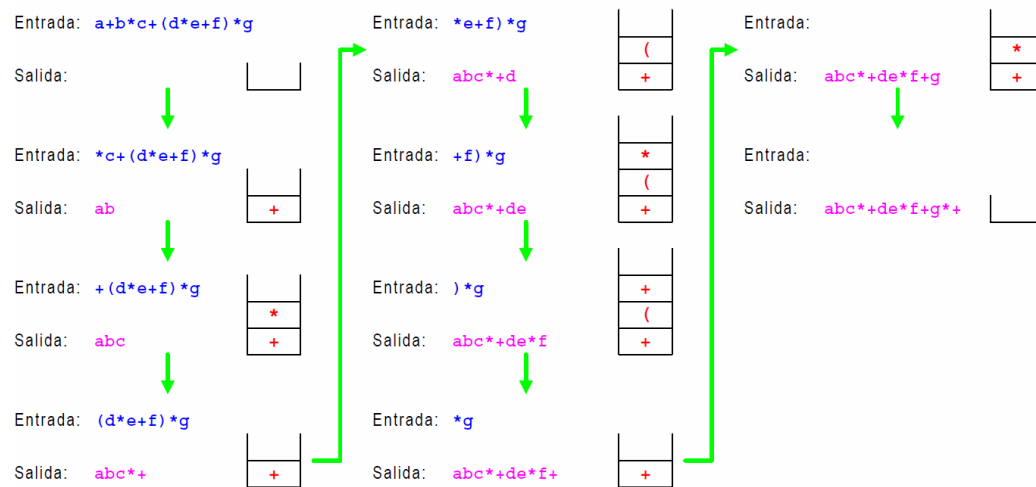
Para interpretar una expresión infija, tenemos dos pasos (ambos se hacen con pilas):

1. La convertimos en postfija. Suponemos que sólo usaremos los operadores  $+$ ,  $*$ ,  $($ ,  $)$  y por lo tanto el algoritmo que veremos sólo funciona para éstos. La precedencia de los operadores es la habitual: la prioridad más baja es para  $+$ , después  $*$  y la más alta para  $($  y  $)$ . El algoritmo requiere un tiempo  $O(n)$ , siendo  $n$  el número de tokens (operadores y operandos) de la expresión
2. Interpretamos la expresión postfija, obteniendo un resultado. El algoritmo es el que ha sido mostrado en la actividad 4.6. Este algoritmo requiere también un tiempo  $O(n)$ , siendo  $n$  el número de tokens de la expresión.

Algoritmo para convertir infija en postfija (ver la figura de la siguiente hoja):

- Inicialmente se tiene una pila vacía, una entrada con la expresión en infijo que queremos convertir en postfijo, y una salida vacía en donde quedará la expresión en postfijo cuando acabemos el algoritmo.
- La pila sólo irá conteniendo operadores.
- Se va recorriendo la expresión de entrada de izquierda a derecha y se van realizando ciertas operaciones, según el token que se vaya encontrando en la expresión:
  - Operando: se coloca en la salida
  - Paréntesis derecho: se sacan operadores de la pila y se llevan a la salida hasta encontrar en la pila un paréntesis izquierdo, que también se saca (pero no se coloca en la salida).
  - Paréntesis izquierdo: se introduce en la pila.
  - Cualquier otro operador: mientras que el operador en la cima de la pila sea un operador de mayor o igual prioridad lo sacamos de la pila y lo ponemos en la salida hasta encontrar otro de estrictamente menor prioridad, un paréntesis izquierdo o hasta que la pila quede vacía. Se introduce el nuevo operador en la pila.
- Cuando se acaba la entrada, se saca todo lo que haya en la pila a la salida
- Tanto la entrada como la salida serán de tipo string.

Ejemplo: transformar "a+b\*c+(d\*e+f)\*g" en "abc\*+de\*f+g\*+"



Vamos a construir una clase que tendrá al menos los siguientes métodos:

- Método público "evaluar()": se le pasa por parámetro un texto (objeto de la clase "string") que contiene la expresión infija a evaluar y nos devuelve un int como resultado. Este método llamará a los siguientes en el orden adecuado.
- Método privado "convertirInfijoAPostfijo()". A partir del texto recibido (la expresión en infijo), convertimos la expresión en postfijo y la metemos sus tokens en orden en un string.
- Método privado "evaluarPostfijo()". A partir de string que genera el método anterior (la expresión en postfijo), la evalúa y devuelve un int con el resultado.

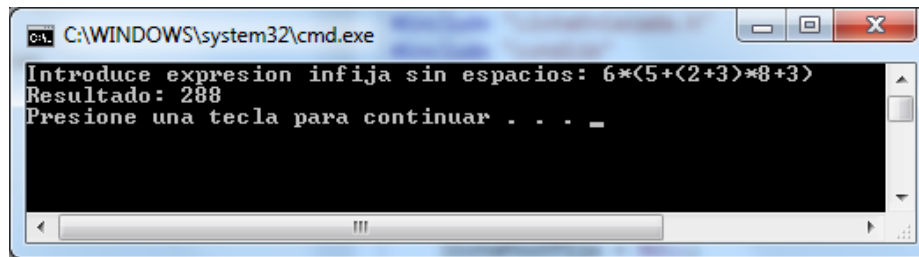
La función "evaluar()" tendrá que realizar varios pasos:

1. Convertir la expresión infija en postfija, utilizando una pila. Para ello, llamará a "convertirInfijoAPostfijo()"
2. Evaluar la expresión postfija, utilizando otra pila. Para ello, llama a "evaluarPostfijo()".

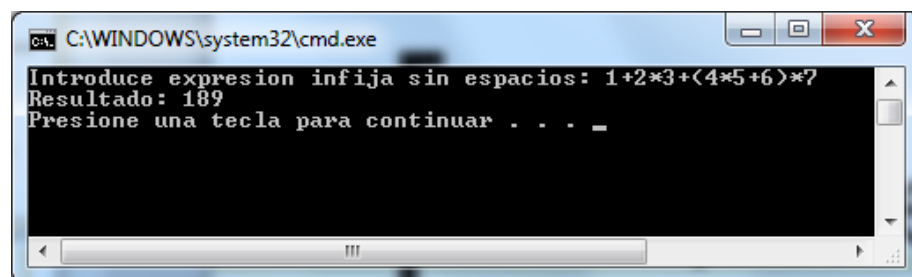
Para simplificar el ejercicio, ten en cuenta lo siguiente:

- No es necesario comprobar que el usuario introduce bien los números y operaciones.
- El usuario introduce la expresión en infijo por teclado sin espacios entre los tokens. Todo seguido. Tampoco es necesario comprobar que el usuario introduce bien esto (lo asumimos).
- Para simplificar el ejercicio, no es necesario comprobar con asserts las siguientes precondiciones: que una expresión en infijo o postfijo está bien formada, o que cada número de la expresión tiene sólo un dígito, o que los operadores son sólo los mencionados. Sí habrá que describirlas en los comentarios de los métodos adecuados.

Pruébalo con "2+3" (¡te tiene que dar 5!), y con los siguientes dos casos más complicados:



```
C:\WINDOWS\system32\cmd.exe
Introduce expresion infija sin espacios: 6*(5+(2+3)*8+3)
Resultado: 288
Presione una tecla para continuar . . . _
```



```
C:\WINDOWS\system32\cmd.exe
Introduce expresion infija sin espacios: 1+2*3+(4*5+6)*7
Resultado: 189
Presione una tecla para continuar . . . _
```