Árboles de Expresión

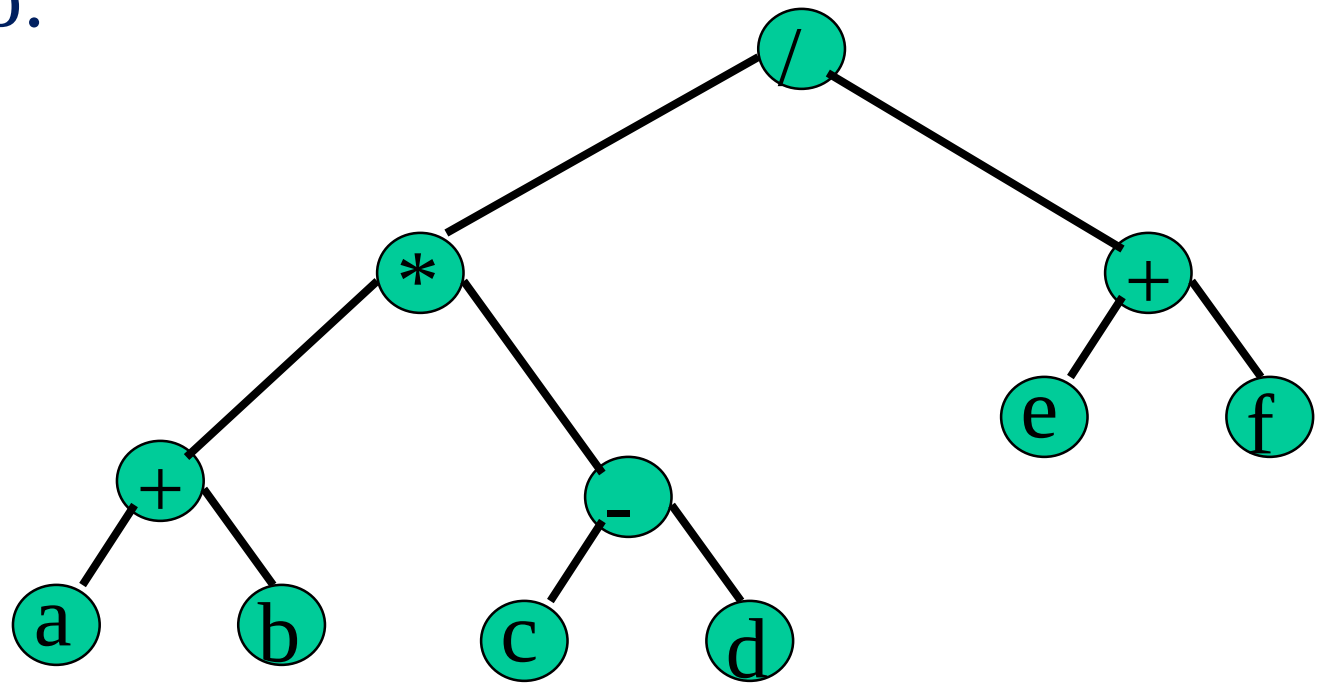
Árbol de Expresión

Es un árbol binario asociado a una expresión aritmética

- Nodos internos representan operadores
- Nodos externos (hojas) representan operandos

Árbol de Expresión

Ejemplo:



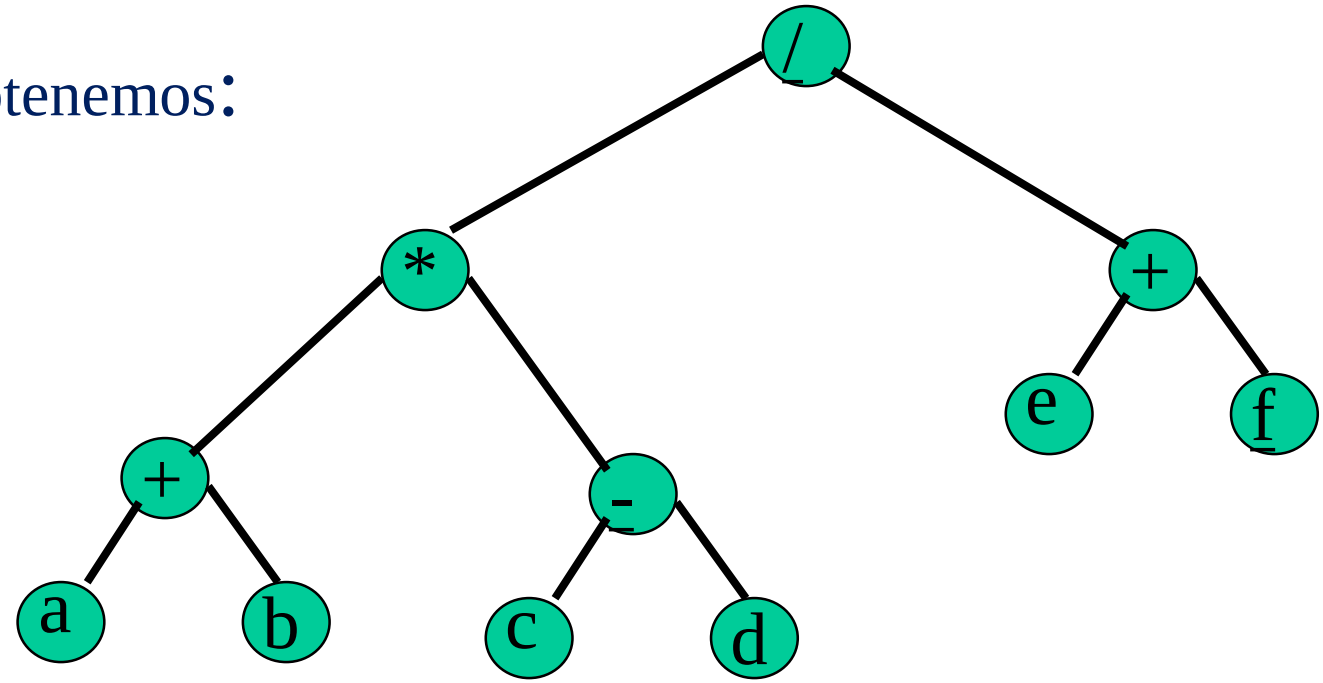
Árbol de Expresión

Aplicaciones:

- En compiladores para analizar, optimizar y traducir programas
- Evaluar expresiones algebraicas o lógicas
- No se necesita el uso de paréntesis
- Traducir expresiones a notación sufija, prefija e infija

Árbol de Expresión

Recorriendo el árbol, obtenemos:



Inorden: (((a + b) * (c – d)) / (e + f))

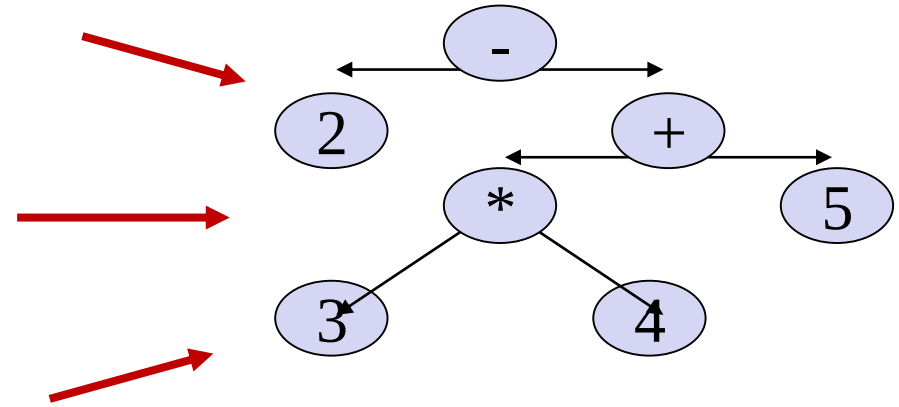
Preorden: /*+ab-cd+ef

Postorden: ab+cd-*ef+/

Construcción de un árbol de expresión

A partir de una:

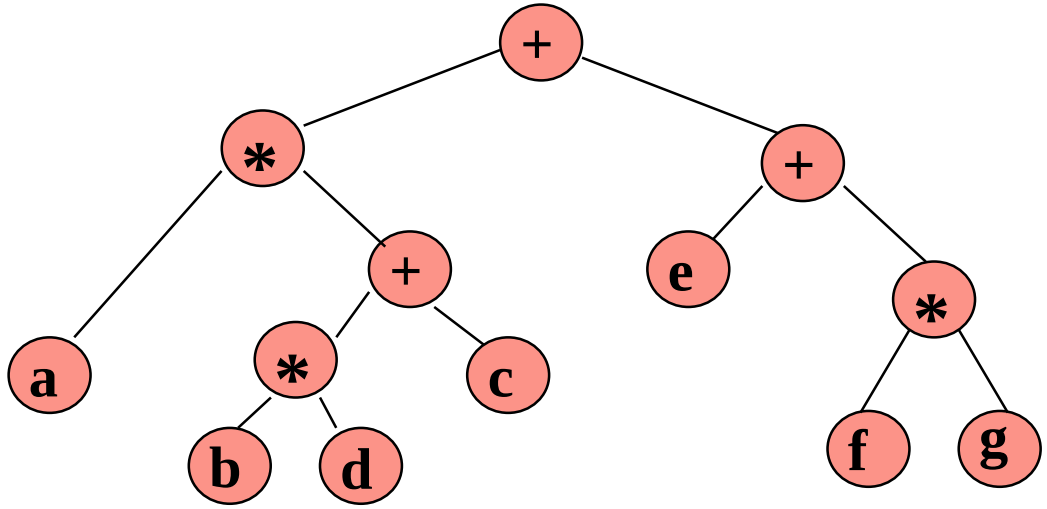
1) Expresión postfija
2) Expresión prefija
3) Expresión infija



Árboles binarios de expresión

Expresión algebraica :

a*(b*d +c) + (e+ f*g)



Expresión **prefija** → + * a + * b d c + e * f g

Expresión **postfija** → a b d * c + * e f g * + +

Expresión **infija** → ((a *((b * d) + c)) + (e + (f * g)))

1) Construcción de un árbol de expresión a partir de una expresión postfija

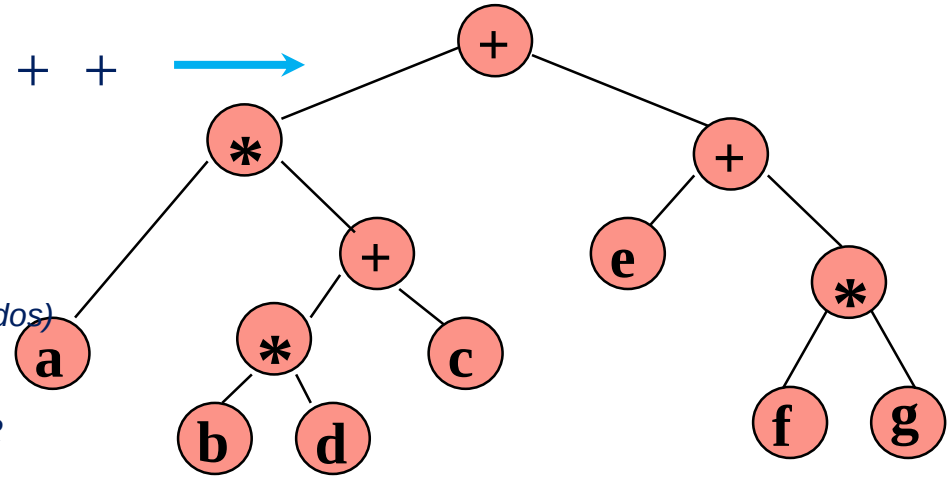
Algoritmo:

tomo un carácter de la expresión
mientras (existe carácter) hacer
si es un ***operando*** *creo un nodo y lo apilo.*
si es un ***operador*** *(lo tomo como la raíz de los dos últimos nodos creados)*
- ***creo*** *un nodo R,*
- ***desapilo*** *y lo agrego como hijo derecho de R*
- ***desapilo*** *y lo agrego como hijo izquierdo de R*
- ***apilo*** *R.*

tomo otro carácter
fin

1) Construcción de un árbol de expresión a partir de una expresión postfija

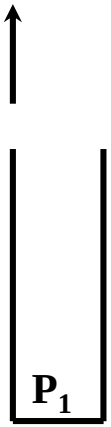
Expresión postfija: a b d * c + * e f g * + + +



Algoritmo:

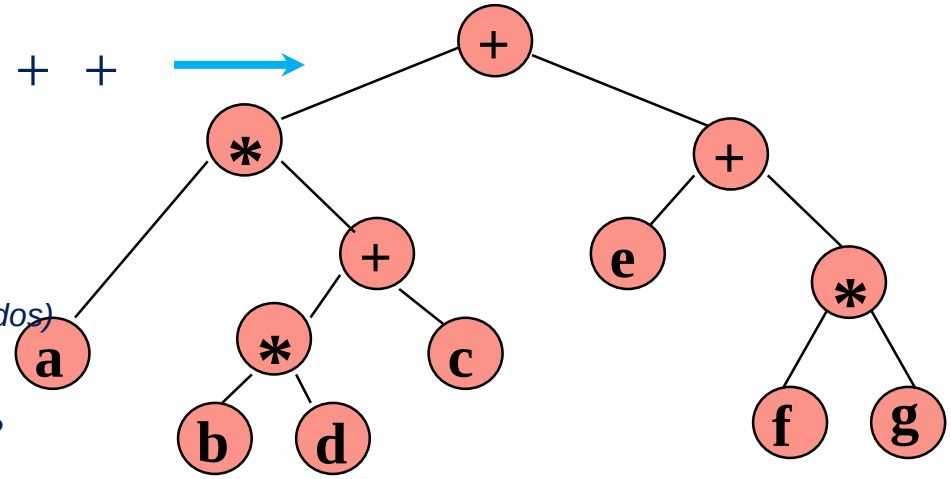
tomo un carácter de la expresión
mientras (existe carácter) hacer
si es un ***operando*** *creo un nodo y lo apilo.*
si es un ***operador*** *(lo tomo como la raíz de los dos últimos nodos creados)*
*- **creo** un nodo R,*
*- **desapilo** y lo agrego como hijo derecho de R*
*- **desapilo** y lo agrego como hijo izquierdo de R*
*- **apilo** R.*
tomo otro carácter
fin

a b d * c + * e f g * + + +



1) Construcción de un árbol de expresión a partir de una expresión postfija

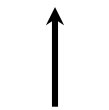
Expresión postfija: a b d * c + * e f g * + + +

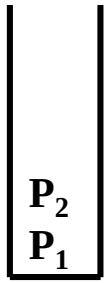


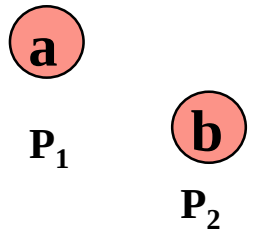
Algoritmo:

tomo un carácter de la expresión
mientras (existe carácter) hacer
si es un ***operando*** *creo un nodo y lo apilo.*
si es un ***operador*** *(lo tomo como la raíz de los dos últimos nodos creados)*
*- **creo** un nodo R,*
*- **desapilo** y lo agrego como hijo derecho de R*
*- **desapilo** y lo agrego como hijo izquierdo de R*
*- **apilo** R.*
tomo otro carácter
fin

a b d * c + * e f g * + + +

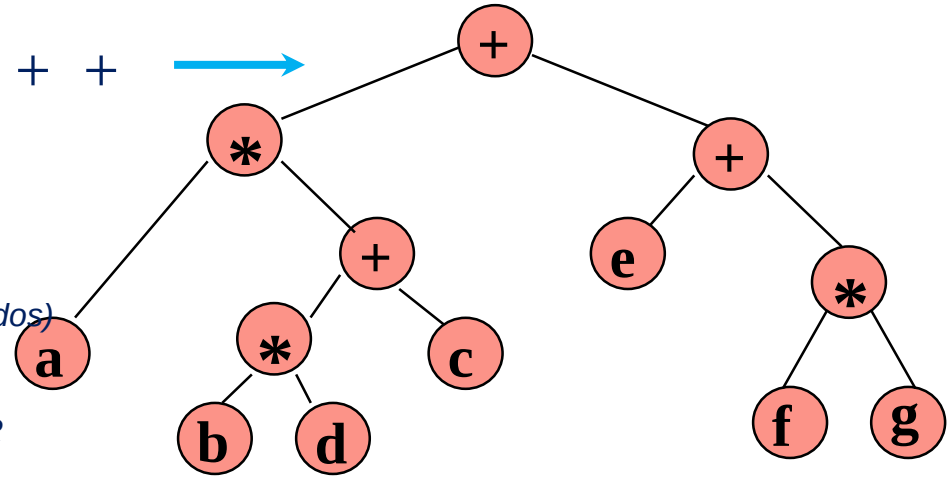






1) Construcción de un árbol de expresión a partir de una expresión postfija

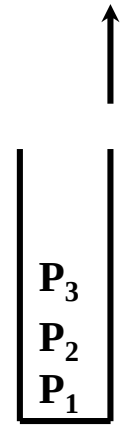
Expresión postfija: a b d * c + * e f g * + + +

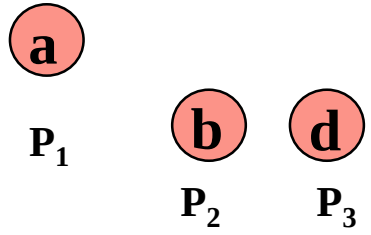


Algoritmo:

tomo un carácter de la expresión
mientras (existe carácter) hacer
si es un ***operando*** *creo un nodo y lo apilo.*
si es un ***operador*** *(lo tomo como la raíz de los dos últimos nodos creados)*
- ***creo*** *un nodo R,*
- ***desapilo*** *y lo agrego como hijo derecho de R*
- ***desapilo*** *y lo agrego como hijo izquierdo de R*
- ***apilo*** *R.*
tomo otro carácter
fin

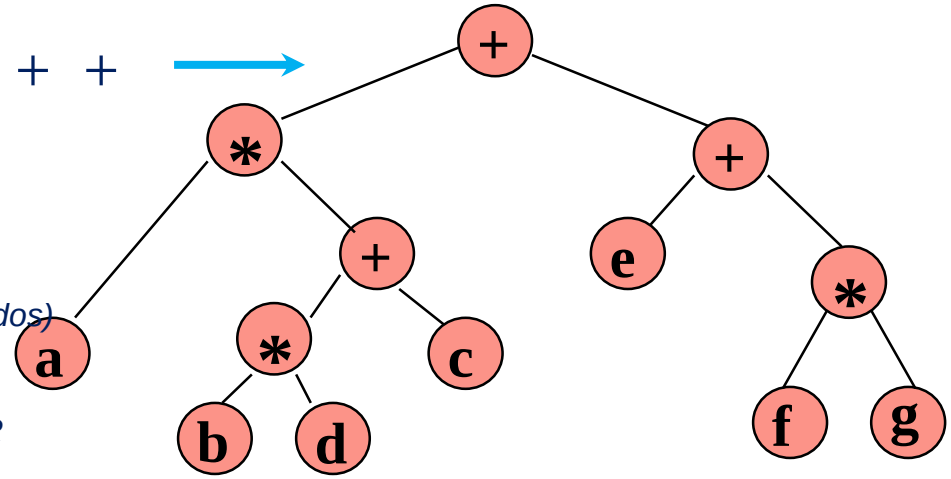
~~a~~ ~~b~~ d * c + * e f g * + + +





1) Construcción de un árbol de expresión a partir de una expresión postfija

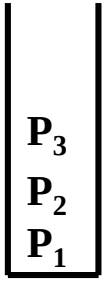
Expresión postfija: a b d * c + * e f g * + + +

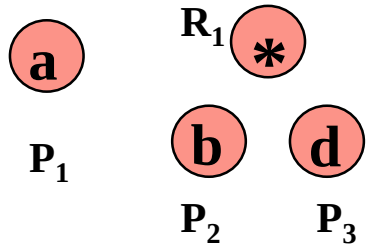


Algoritmo:

tomo un carácter de la expresión
mientras (existe carácter) hacer
si es un ***operando*** *creo un nodo y lo apilo.*
si es un ***operador*** *(lo tomo como la raíz de los dos últimos nodos creados)*
*- **creo** un nodo R,*
*- **desapilo** y lo agrego como hijo derecho de R*
*- **desapilo** y lo agrego como hijo izquierdo de R*
*- **apilo** R.*
tomo otro carácter
fin

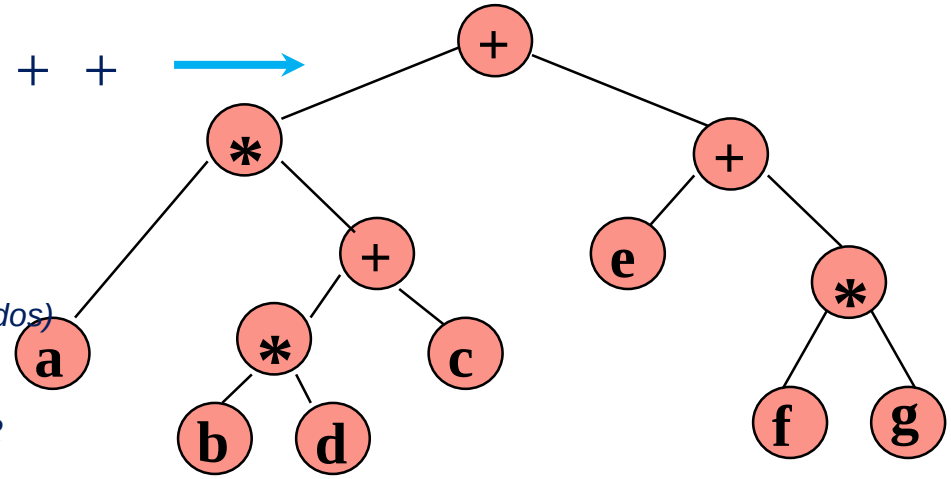
a b d * c + * e f g * + + +





1) Construcción de un árbol de expresión a partir de una expresión postfija

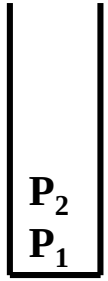
Expresión postfija: a b d * c + * e f g * + +

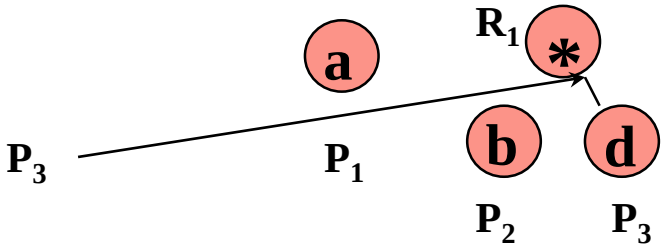


Algoritmo:

tomo un carácter de la expresión
mientras (existe carácter) hacer
si es un ***operando*** *creo un nodo y lo apilo.*
si es un ***operador*** *(lo tomo como la raíz de los dos últimos nodos creados)*
*- **creo** un nodo R,*
*- **desapilo** y lo agrego como hijo derecho de R*
*- **desapilo** y lo agrego como hijo izquierdo de R*
*- **apilo** R.*
tomo otro carácter
fin

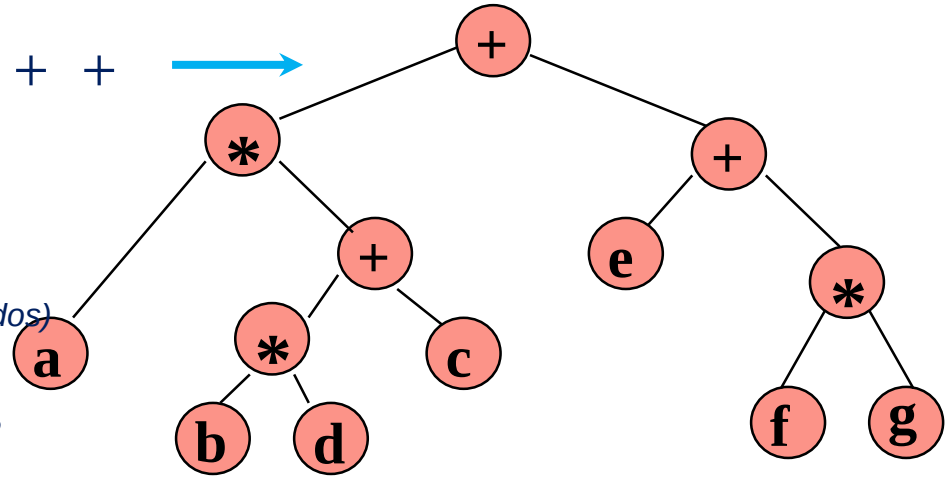
a b d * c + * e f g * + +





1) Construcción de un árbol de expresión a partir de una expresión postfija

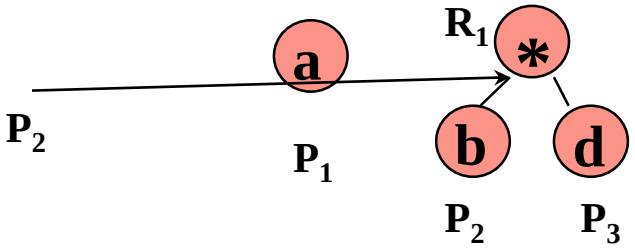
Expresión postfija: a b d * c + * e f g * + + +



Algoritmo:

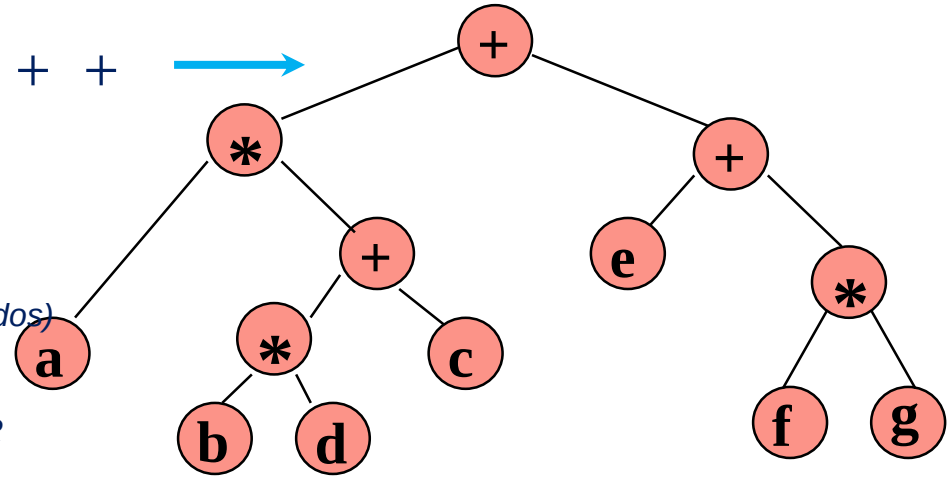
tomo un carácter de la expresión
mientras (existe carácter) hacer
si es un ***operando*** *creo un nodo y lo apilo.*
si es un ***operador*** *(lo tomo como la raíz de los dos últimos nodos creados)*
*- **creo** un nodo R,*
*- **desapilo** y lo agrego como hijo derecho de R*
*- **desapilo** y lo agrego como hijo izquierdo de R*
*- **apilo** R.*
tomo otro carácter
fin

a b d * c + * e f g * + + +



1) Construcción de un árbol de expresión a partir de una expresión postfija

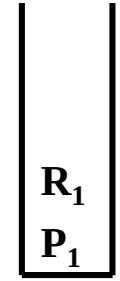
Expresión postfija: a b d * c + * e f g * + +

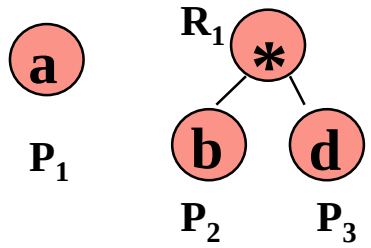


Algoritmo:

tomo un carácter de la expresión
mientras (existe carácter) hacer
si es un ***operando*** *creo un nodo y lo apilo.*
si es un ***operador*** *(lo tomo como la raíz de los dos últimos nodos creados)*
*- **creo** un nodo R,*
*- **desapilo** y lo agrego como hijo derecho de R*
*- **desapilo** y lo agrego como hijo izquierdo de R*
*- **apilo** R.*
tomo otro carácter
fin

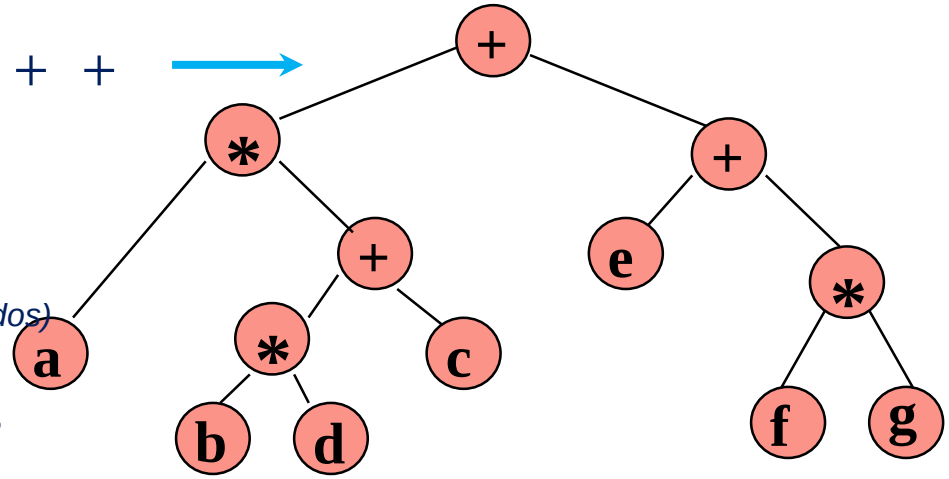
a b d * c + * e f g * + +





1) Construcción de un árbol de expresión a partir de una expresión postfija

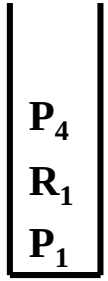
Expresión postfija: a b d * c + * e f g * + + +

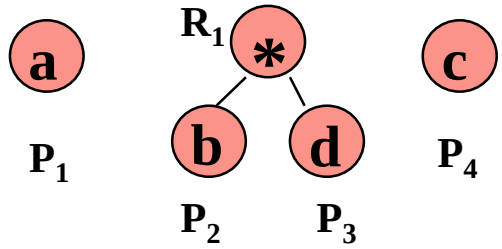


Algoritmo:

tomo un carácter de la expresión
mientras (existe carácter) hacer
si es un ***operando*** *creo un nodo y lo apilo.*
si es un ***operador*** *(lo tomo como la raíz de los dos últimos nodos creados)*
*- **creo** un nodo R,*
*- **desapilo** y lo agrego como hijo derecho de R*
*- **desapilo** y lo agrego como hijo izquierdo de R*
*- **apilo** R.*
tomo otro carácter
fin

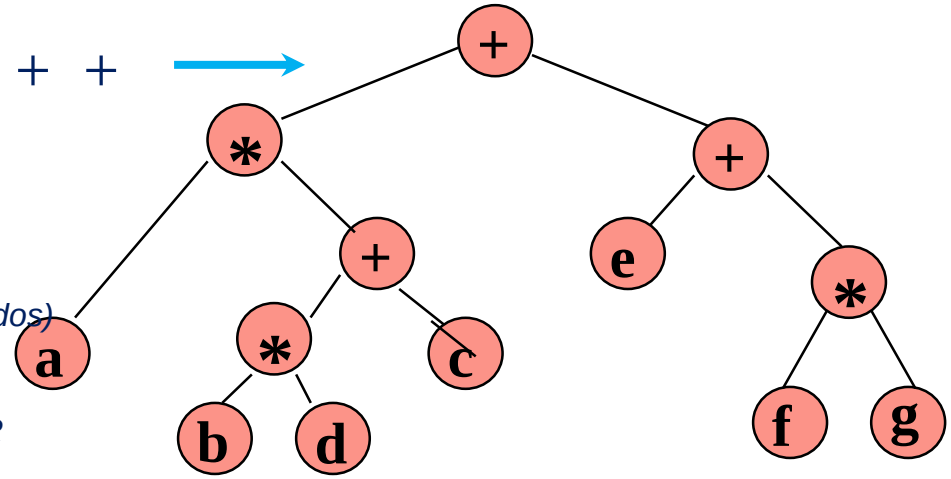
a b d * c + * e f g * + + +





1) Construcción de un árbol de expresión a partir de una expresión postfija

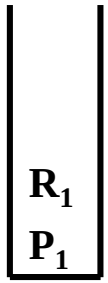
Expresión postfija: a b d * c + * e f g * + + +

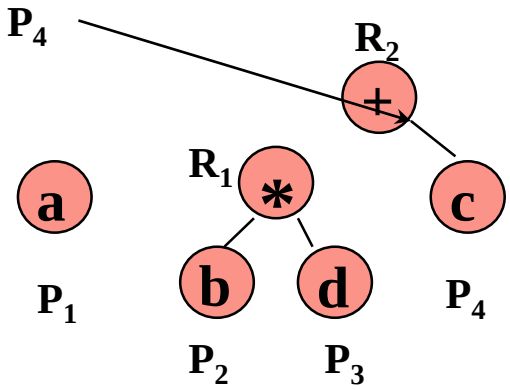


Algoritmo:

tomo un carácter de la expresión
mientras (existe carácter) hacer
si es un ***operando*** *creo un nodo y lo apilo.*
si es un ***operador*** *(lo tomo como la raíz de los dos últimos nodos creados)*
*- **creo** un nodo R,*
*- **desapilo** y lo agrego como hijo derecho de R*
*- **desapilo** y lo agrego como hijo izquierdo de R*
*- **apilo** R.*
tomo otro carácter
fin

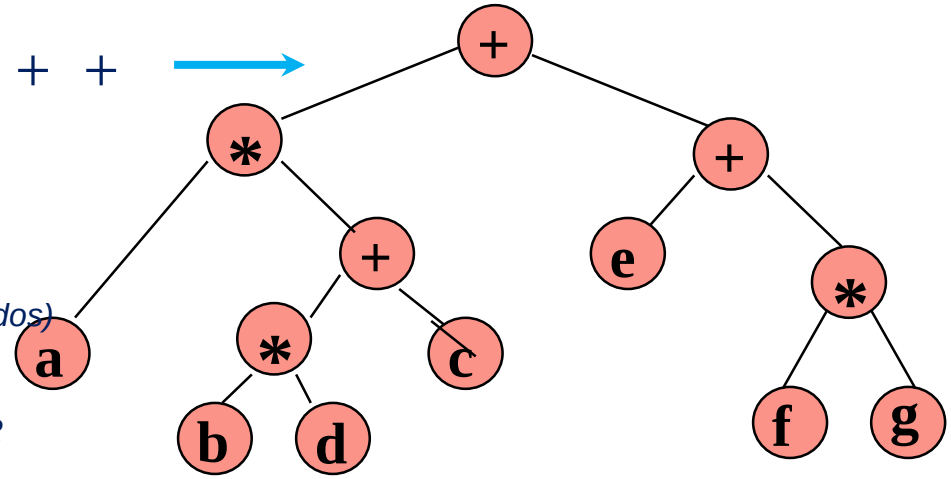
a b d * c + * e f g * + + +





1) Construcción de un árbol de expresión a partir de una expresión postfija

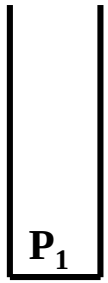
Expresión postfija: a b d * c + * e f g * + + +

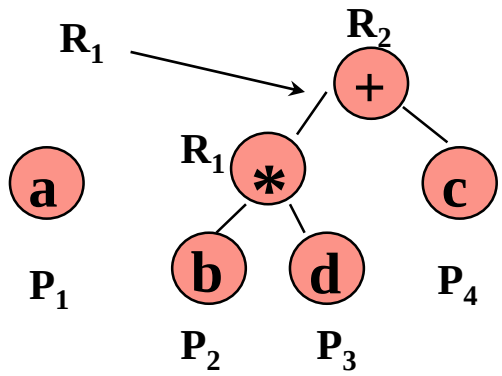


Algoritmo:

tomo un carácter de la expresión
mientras (existe carácter) hacer
si es un ***operando*** *creo un nodo y lo apilo.*
si es un ***operador*** *(lo tomo como la raíz de los dos últimos nodos creados)*
*- **creo** un nodo R,*
*- **desapilo** y lo agrego como hijo derecho de R*
*- **desapilo** y lo agrego como hijo izquierdo de R*
*- **apilo** R.*
tomo otro carácter
fin

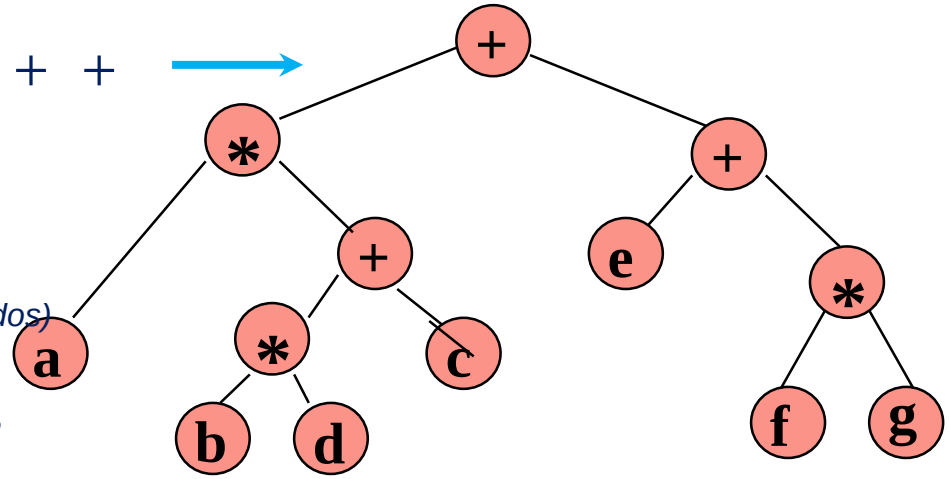
a b d * c + * e f g * + + +





1) Construcción de un árbol de expresión a partir de una expresión postfija

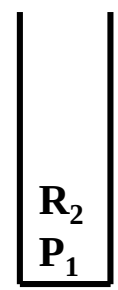
Expresión postfija: a b d * c + * e f g * + +

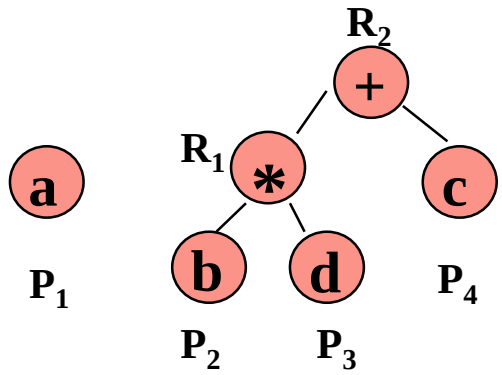


Algoritmo:

tomo un carácter de la expresión
mientras (existe carácter) hacer
si es un ***operando*** *creo un nodo y lo apilo.*
si es un ***operador*** *(lo tomo como la raíz de los dos últimos nodos creados)*
*- **creo** un nodo R,*
*- **desapilo** y lo agrego como hijo derecho de R*
*- **desapilo** y lo agrego como hijo izquierdo de R*
*- **apilo** R.*
tomo otro carácter
fin

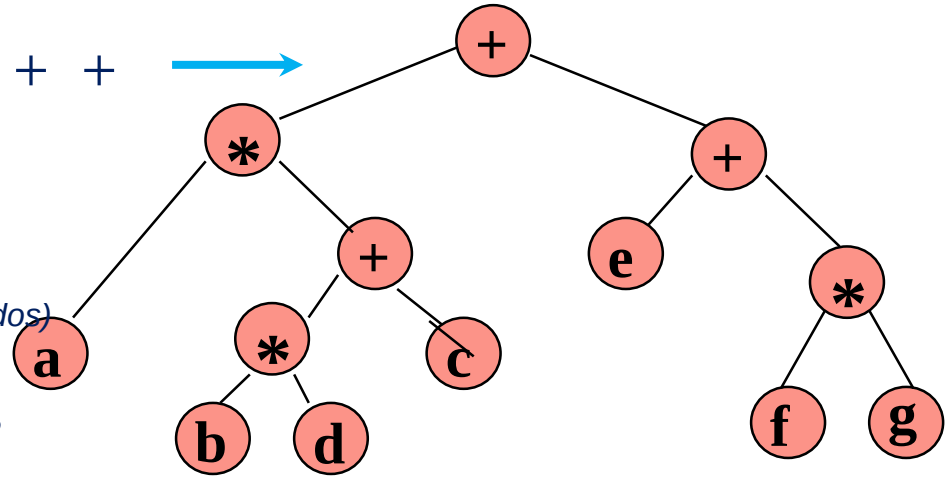
a b d * c + * e f g * + +





1) Construcción de un árbol de expresión a partir de una expresión postfija

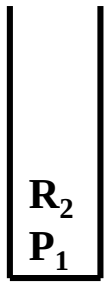
Expresión postfija: a b d * c + * e f g * + +

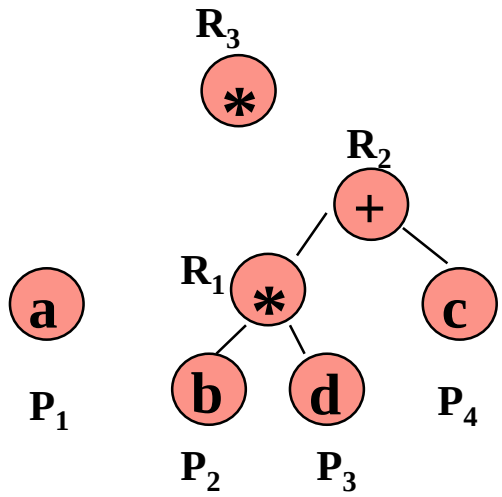


Algoritmo:

tomo un carácter de la expresión
mientras (existe carácter) hacer
si es un ***operando*** *creo un nodo y lo apilo.*
si es un ***operador*** *(lo tomo como la raíz de los dos últimos nodos creados)*
*- **creo** un nodo R,*
*- **desapilo** y lo agrego como hijo derecho de R*
*- **desapilo** y lo agrego como hijo izquierdo de R*
*- **apilo** R.*
tomo otro carácter
fin

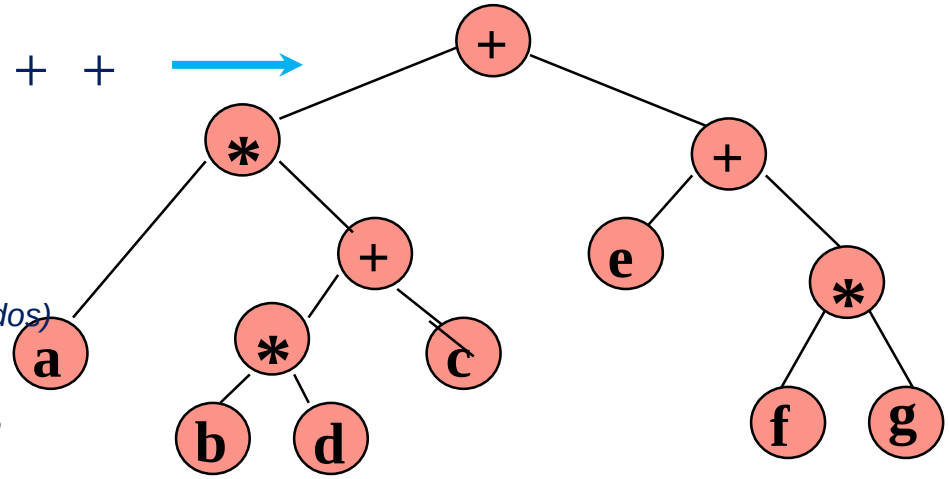
a b d * c + * e f g * + +





1) Construcción de un árbol de expresión a partir de una expresión postfija

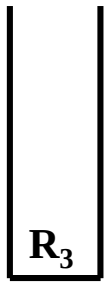
Expresión postfija: a b d * c + * e f g * + +

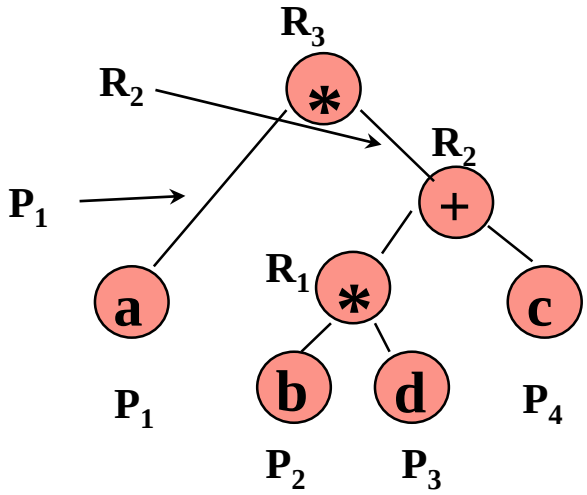


Algoritmo:

tomo un carácter de la expresión
mientras (existe carácter) hacer
si es un ***operando*** *creo un nodo y lo apilo.*
si es un ***operador*** *(lo tomo como la raíz de los dos últimos nodos creados)*
*- **creo** un nodo R,*
*- **desapilo** y lo agrego como hijo derecho de R*
*- **desapilo** y lo agrego como hijo izquierdo de R*
*- **apilo** R.*
tomo otro carácter
fin

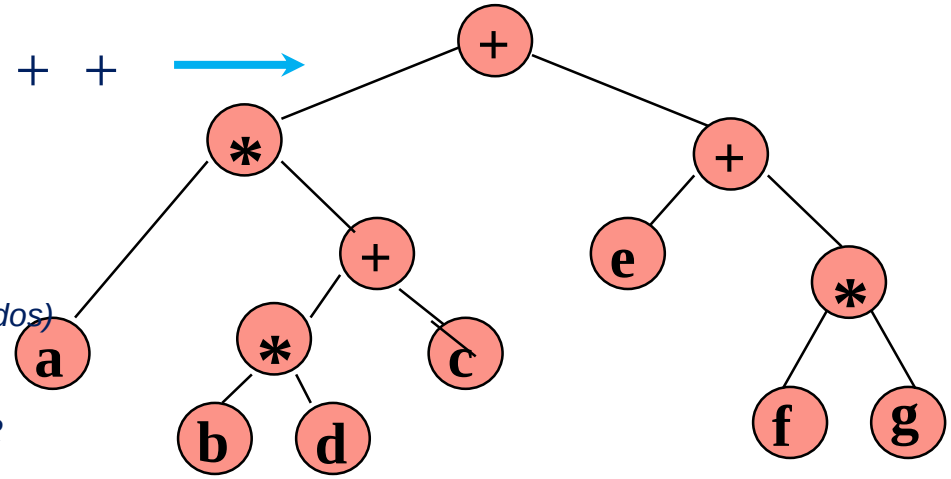
a b d * c + * e f g * + +





1) Construcción de un árbol de expresión a partir de una expresión postfija

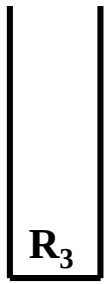
Expresión postfija: a b d * c + * e f g * + +

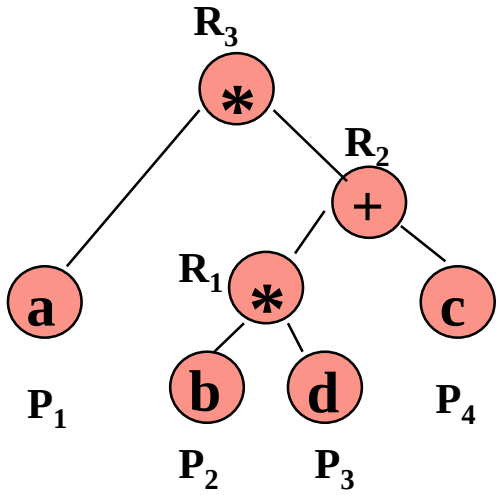


Algoritmo:

tomo un carácter de la expresión
mientras (existe carácter) hacer
*si es un **operando** creo un nodo y lo apilo.*
*si es un **operador** (lo tomo como la raíz de los dos últimos nodos creados)*
*- **creo** un nodo R,*
*- **desapilo** y lo agrego como hijo derecho de R*
*- **desapilo** y lo agrego como hijo izquierdo de R*
*- **apilo** R.*
tomo otro carácter
fin

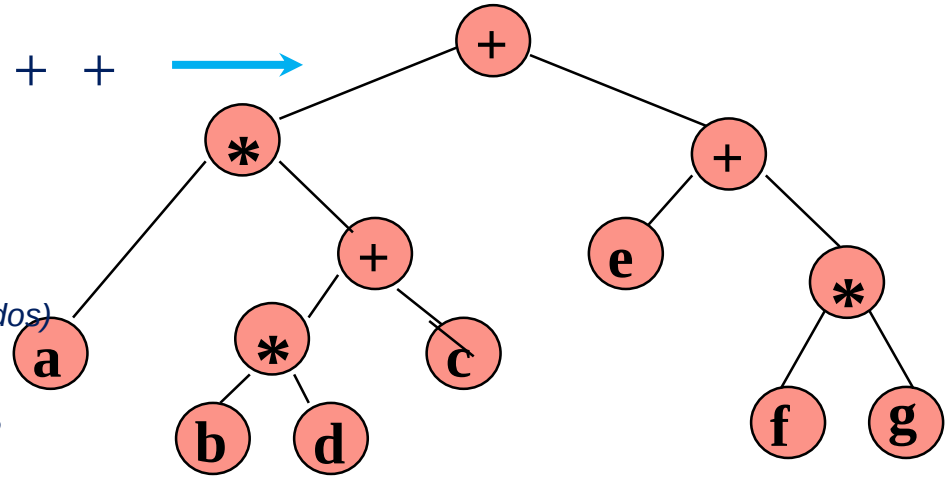
a b d * c + * e f g * + +





1) Construcción de un árbol de expresión a partir de una expresión postfija

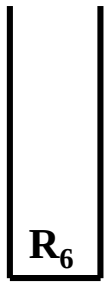
Expresión postfija: a b d * c + * e f g * + +

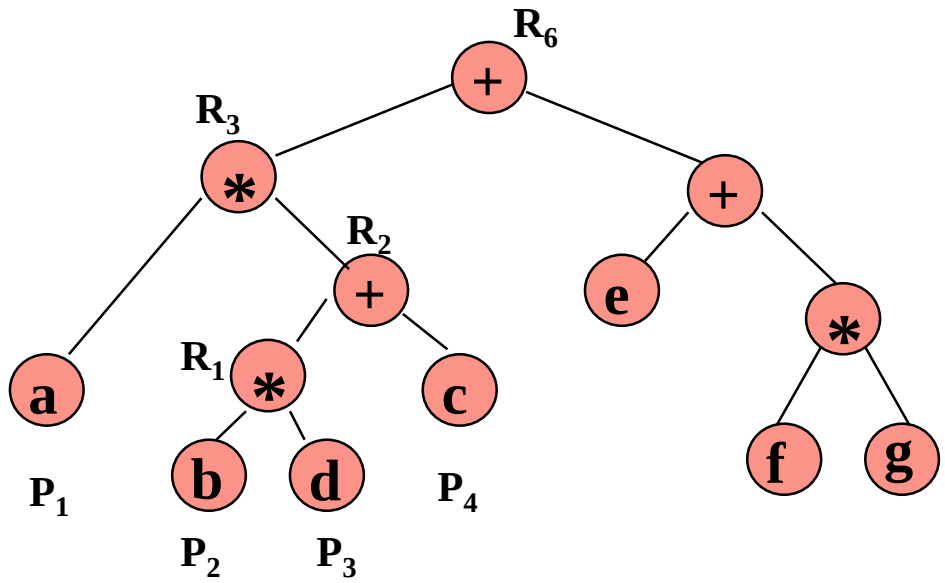


Algoritmo:

tomo un carácter de la expresión
mientras (existe carácter) hacer
si es un ***operando*** *creo un nodo y lo apilo.*
si es un ***operador*** *(lo tomo como la raíz de los dos últimos nodos creados)*
*- **creo** un nodo R,*
*- **desapilo** y lo agrego como hijo derecho de R*
*- **desapilo** y lo agrego como hijo izquierdo de R*
*- **apilo** R.*
tomo otro carácter
fin

a b d * c + * e f g * + +





2) Construcción de un árbol de expresión a partir de una expresión prefija

Algoritmo:

ArbolExpresión (A: ArbolBin, exp: string)

si exp nulo □ *nada.*
si es un operador □ *- creo un nodo raíz R*
- ArbolExpresión (subArbIzq de R, exp (sin 1° carácter))
- ArbolExpresión (subArbDer de R, exp (sin 1° carácter))
si es un operando □ *creo un nodo (hoja)*

3) Construcción de un árbol de expresión a partir de una expresión infija

Expresión infija

(i)

Se usa una pila y se tiene en cuenta la precedencia de los operadores

Expresión postfija

`2+5*3+1`

`253*+1+`

3) Construcción de un árbol de expresión a partir de una expresión infija

Expresión infija

(i)

Se usa una pila y se tiene en cuenta la precedencia de los operadores

Expresión postfija

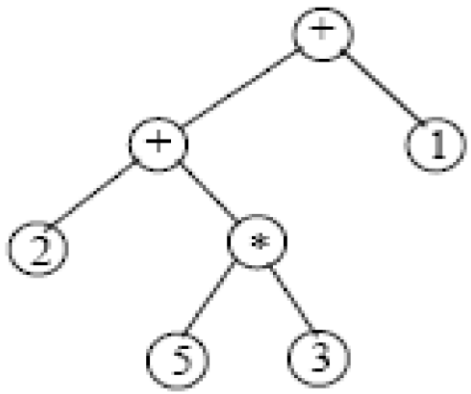
(ii)

Se usa la estrategia 1)

Árbol de Expresión

2+5*3+1

253*+1+



- ***Convertir una expresión infija en árbol de expresión: se debe convertir la expresión infija en postfija (i) y a partir de ésta, construir el árbol de expresión (ii).***

(i) Estrategia del Algoritmo para convertir exp. infija en postfija :

a) si es un operando se coloca en la salida.
b) si es un operador se maneja una pila según la prioridad del operador en relación al tope de la pila

operador con > prioridad que el tope –> se apila
operador con <= prioridad que el tope –> se desapila elemento colocándolo en la salida.
Se vuelve a comparar el operador con el tope de la pila

c) si es un "(" , ")" "(" se apila
")" se desapila todo hasta el "(", incluído éste

d) cuando se llega al final de la expresión, se desapilan todos los elementos llevándolos a la salida, hasta que la pila quede vacía.

Operadores ordenados de mayor a menor según su prioridad:

^ (potencia)

*, / (multiplicación y división)

+, - (suma y resta)

Los " (" siempre se apilan como si tuvieran la mayor prioridad y se desapilan sólo cuando aparece un ") " .

Evaluar un árbol de expresión

Algoritmo:

EvaluarAE (A: ArbolBin)

si dato es ***operador*** □ *EvaluarAE (subArbIzq de A)*

EvaluarAE (subArbDer de A)

si es un operando □ *Retornar el dato del nodo (hoja)*

Evaluar un árbol de expresión

Algoritmo:
Integer EvaluarAE (A: ArbolBin)

*si dato es **operador** //*
valorIzq = EvaluarAE (subArbIzq de A)
valorDer = EvaluarAE (subArbDer de A)
*según el valor **operador** {*
*"**+**" : retornar valorIzq **+** valorDer*
*"**-**" : retornar valorIzq **-** valorDer*
*"*****" : retornar valorIzq ***** valorDer*
*"**/**" : retornar valorIzq **/** valorDer }*

*si es un **operando*** □ *Retornar el dato del nodo (hoja)*

Ejercitación

Árbol binario de expresión

Ejercicio 1.

✓*Dada la siguiente expresión postfija :* ***I J K + + A B * C - *** *, dibuje su correspondiente árbol binario de expresión*

✓*Convierta la expresión* ***((a + b) + c * (d + e) + f) * (g + h)*** *en expresión prefija*

Ejercicio 2.

✓*Dada la siguiente expresión prefija :* **** + I + J K - C * A B*** *, dibuje su correspondiente árbol binario de expresión*

✓*Convierta la expresión* ***((a + b) + c * (d + e) + f) * (g + h)*** *en expresión postfija*