**El algoritmo de Shuting Yard**

Es un método para convertir una expresión matemática de notación infija (donde los operadores están entre los operandos) a notación postfija (también conocida como notación polaca inversa o RPN). Este algoritmo es muy útil en el contexto de expresiones regulares y calculadoras de expresión debido a su capacidad para gestionar la precedencia y la asociación de los operadores.

**Prácticamente el algoritmo a seguir fue:**

**Inicialización:**

Se tienen tres estructuras de datos:

Una lista de salida (output), inicialmente vacía.

Una pila de operadores (operators), inicialmente vacía.

Una lista de tokens que representa la expresión infija.

**Lectura de tokens:**

Se lee cada token de la expresión infija uno por uno.

**Manejo de tokens:**

Si el token es un operando (letra o dígito):

Se agrega directamente a la lista de salida (output).

Si el token es un operador:

Mientras haya un operador en la cima de la pila (operators) que tenga mayor o igual precedencia que el operador actual y el operador de la pila no sea un paréntesis izquierdo, se saca el operador de la pila y se agrega a la lista de salida.

Luego, se pone el operador actual en la pila.

Si el token es un paréntesis izquierdo ('('):

Se empuja a la pila de operadores.

Si el token es un paréntesis derecho (')'):

Se sacan operadores de la pila y se agregan a la lista de salida hasta que se encuentre un paréntesis izquierdo.

Se descarta el paréntesis izquierdo.

Finalización:

Después de leer todos los tokens, si quedan operadores en la pila, se sacan y se agregan a la lista de salida.

Ejemplo de Expresión Regular:

Tomemos la expresión regular (a|b)\* y veamos cómo se convierte a notación postfija utilizando el algoritmo de Shunting Yard.

Entrada:

(a|b)\*

Proceso Detallado:

Inicialización:

Lista de salida: []

Pila de operadores: []

Tokens: [ '(', 'a', '|', 'b', ')', '\*' ]

Lectura y manejo de tokens:

Token: (

Es un paréntesis izquierdo, se agrega a la pila.

Pila: [ '(' ]

Salida: []

Token: a

Es un operando, se agrega a la lista de salida.

Pila: [ '(' ]

Salida: [ 'a' ]

Token: |

Es un operador, se agrega a la pila.

Pila: [ '(', '|' ]

Salida: [ 'a' ]

Token: b

Es un operando, se agrega a la lista de salida.

Pila: [ '(', '|' ]

Salida: [ 'a', 'b' ]

Token: )

Se sacan operadores de la pila hasta encontrar (.

Se saca | y se agrega a la lista de salida.

Se descarta (.

Pila: []

Salida: [ 'a', 'b', '|' ]

Token: \*

Es un operador, se agrega a la pila.

Pila: [ '\*' ]

Salida: [ 'a', 'b', '|' ]

Finalización:

Se sacan los operadores restantes de la pila.

Se saca \* y se agrega a la lista de salida.

Pila: []

Salida: [ 'a', 'b', '|', '\*' ]

Salida:

[ 'a', 'b', '|', '\*' ]