|  |
| --- |
| **Algoritmo Creación de Árbol de Expresión a través de expresión regular**  **Creador: Moises Alonso** |
| Entradas   1. Tokens de la expresión regular (Símbolos terminales “**st**”, meta caracteres operadores incluyendo la concatenación “**op**”) 2. Pila de Tokens llamada “**T**” 3. Pila de árboles llamada “**S**” |
| Salidas   1. Árbol de expresión con el símbolo terminal extendido # |
| Proceso   1. Mientras existan tokens en la expresión regular 2. Obtener **token** 3. Si **token** es un carácter de escape.    1. Si a la expresión regular aún le queda al menos un **token.**       1. Obtener siguiente **token.**       2. Convertir **token** en árbol.       3. Hacer “push” a la pila **S** con el nuevo árbol.    2. De lo contrario       1. Error, se esperaban más tokens. 4. Si **token** es **st**    1. Convertir **st** en árbol    2. Hacer “push” a la pila **S** con el nuevo árbol generado de **st** 5. Sino Si **token** es **“(“**    1. Hacer “push” a la pila **T** con **token** 6. Sino Si **token** es **“)“**    1. Mientras la longitud de **T** sea mayor que **0** y el último dato insertado en **T** sea diferente de **“(“**, hacer:       1. Si Longitud de **T** es igual a **0**          1. Existe error, faltan operandos       2. Si la longitud de **S** es menor a **2**          1. Existe error, faltan operandos       3. Hacer “pop” a **T** y convertirlo en árbol llamado **temp**       4. Hacer “pop” a **S** y asignarlo al hijo derecho de **temp**       5. Hacer “pop” a **S** y asignarlo al hijo izquierdo de **temp**       6. Hacer “Push” de **temp** en la pila **S**    2. Hacer “pop” a **T** con el último dato 7. Sino si **token** es **op**    1. Si **op** es unario       1. Convertir **op** en árbol       2. Si la longitud de **S** es menor que **0**          1. Existe error, faltan operandos       3. Hacer “pop” de **S** y asignarlo como hijo izquierdo       4. Hacer “push” a la pila **S** con el nuevo árbol generado de **op**    2. Sino si **T** no está vacia y el “top” **op** en **T** es diferente a **“(“** y precedencia de **token** es menor o igual a último **op** en **T**       1. Extraer de **T** a **op**, convertirlo en árbol y llamarlo **temp**       2. Si cantidad de elementos en en **S** es menor a 2          1. Existe error, faltan operandos       3. Extraer último árbol de **S** y asignarlo al hijo derecho de **temp**       4. Extraer último árbol de **S** y asignarlo al hijo izquierdo de **temp**       5. Push de **temp** en la pila **S**       6. Push del **token** en la pila **T**    3. Si **op** no es unario Hacer “push” en la pila **T** con **token** 8. De lo contrario    1. Error, no es token reconocido 9. Si aún existen tokens en la expresión regular, ir a paso 2 10. Mientras la longitud de **T** sea Mayor que **0**     1. Hacer “pop” de **T** y crear un nuevo árbol llamado **temp**     2. Si **temp** es **“(“**        1. Existe error, faltan operandos     3. Si longitud de **S** menor que **2**        1. Existe error, faltan operandos     4. Hacer “pop” a la pila **S** y asignarlo como hijo derecho de **temp**     5. Hacer “pop” a la pila **S** y asignarlo como hijo izquierdo de **temp**     6. Hacer “push” a la pila **S** con el árbol **temp** 11. Si longitud de **T** es mayor que **0** ir a paso **9** 12. Si longitud de **S** es diferente de **1**     1. Existe error, faltan operandos 13. Hacer “pop” a **S** y retornar el valor |
|  |

Aldous C., Richard T. & Rogelio G. (2004, 6 de febrero). System and method for securing computer against virus. Recuperado 29 febrero, 2020, de <https://patentimages.storage.googleapis.com/3e/ec/00/5e0b3704bc5c3d/WO2005076101A2.pdf>



