## Consideraciones de YAPar

### Descripción

YAPar – Yet Another Parser – se encuentra basado al estilo de Yacc y toma inspiración sobre la implementación de `ocamlyacc`, herramienta escrita para OCaml. El objetivo principal de YAPar es la generación de Analizadores Sintácticos a partir de la especificación de una Gramática Regular. La llamada esperada sería:

```
yapar parser.yalp -l lexer.yal -o theparser
```

Donde `parser.yalp` es un archivo escrito en Lenguaje YAPar. Esta instrucción genera un archivo – `theparser` – escrito en el lenguaje de su elección, el cual implementa el Analizador Sintáctico a partir de lo definido en `parser.yalp`. Este archivo se utiliza en conjunto con el Lexer generado por YALex previamente, por lo que su llamada a `yapar` debe considerar la ejecución de `yalex` para la implementación de un Generador de Analizadores Léxicos, que provea una tabla de símbolos con información sobre los tokens encontrados. Para ello puede especificar esto directamente en `yapar` con el argumento `-l`, especificando el archivo de Lexer.

### Estructura de un Archivo [yalp]

En este ejemplo se puede observar la estructura de uno de estos archivos

```
/* Definición de parser */

/* INICIA Sección de TOKENS */
%token TOKEN_1
%token TOKEN_2
%token TOKEN_3 TOKEN_4
%token WS
IGNORE WS
/* FINALIZA Sección de TOKENS */

%%

/* INICIA Sección de PRODUCCIONES */
production1:
    production1 TOKEN_2 production2
  | production2
;
production2:
    production2 TOKEN_2 production3
  | production3
;
production3:
    TOKEN_3 production1 TOKEN_4
  | TOKEN_1
;
/* FINALIZA Sección de PRODUCCIONES */
```

- Los comentarios son delimitados por `/*` y `*/`
- La primera sección del archivo debe ser la sección de `TOKENS`. Cada token se define precedido por la palabra reservada `token` en minúscula, seguida del nombre de token en mayúscula. Estos deben hacer match con los tokens producidos por YALex.
- Una línea en esta sección puede contener la declaración de múltiples tokens, separados por un espacio en blanco.
- La palabra reservada `IGNORE` sirve para ignorar `TOKENS` en la ejecución del parsing subsiguiente.
- Para dividir la sección de `TOKENS` y la sección de `PRODUCCIONES`, debe usar el símbolo `%%`
- En la sección de `PRODUCCIONES`, una producción comienza con el nombre asignado para la producción, seguida de dos puntos (`:`), luego puede colocar diversas reglas dentro de esta producción, utilizando minúsculas para los nombres de otras producciones (estos son los no-terminales de la gramática) y utilizando el identificador de `TOKEN` en mayúscula para los tokens correspondientes en su regla (estos son los terminales de la gramática). Separe las reglas con el símbolo `|`. Para finalizar la declaración de la producción, utilice punto y coma (`;`).
- Puede tener la cantidad de producciones que desee. Cuando se llegue a la última línea del archivo, YAPar dejará de interpretar producciones.