Flex

La especificación de flex tiene el siguiente formato:

definiciones

%%

reglas

%%

código del usuario

La sección de definiciones sirve para dar nombres a ciertas ER y así simplificar y hacer más claras otras ER.

La sección de reglas se delimita con %% y es donde indicamos mediante ER los tokens a reconocer y mediante código C las acciones a realizar cada vez que se reconoce un token.

La sección de código de usuario es optativa. En caso de no estar, el segundo delimitador %% no es necesario. Sirve para agregar código que flex copiará tal cual al final del fuente que genere.

Puede haber una función main que típicamente haga al menos un llamado a yylex() o bien puede haber funciones auxiliares a ser invocadas por las acciones de las reglas

**yylex()** es el escáner. Devuelve el token encontrado como un int.

**yytext** variable con el lexema encontrado, en modo flex es un char \* (solo accesible desde flex, se debe copiar en un registro semántico para ser usado en bison)

**yyleng** largo de yytext

**yyin** es el FILE \* de donde el escáner lee la entrada, por defecto stdin

**yyout** es el FILE \* a donde el escáner dirige su salida, por defecto stdout

Reglas: Sintaxis

* patrón acción

**Patrón:** es una ER que no debe estar indentada y se separa de la acción por el primer blanco (espacio o tabulador no encerrado entre comillas o con barra invertida delante).

**Acción:** es código C que debe comenzar en la misma línea. Si es una sentencia compuesta puede extenderse a las líneas siguientes.

Código adicional en Reglas

Si después del inicio de la sección de reglas, marcada por %% y antes de la primer regla se coloca código indentado o entre %{ y %} (estos delimitadores NO identados) dicho código se coloca al inicio de la rutina del escáner.

Así es posible declarar variables locales a la rutina del escáner y/o código que se ejecutará siempre al inicio dicha rutina

Si agrego código después de la primer regla no está definido donde será colocado y suele terminar dando errores de compilación.