Universidad Nacional Autónoma de México
Facultad de Ciencias
Compiladores
Práctica 1

Alumno: Osorio Escandón Huriel / 317204652 / Huriel117@ciencias.unam.mx

1. Ejercicios

1. ¿Qué ocurre si en la primera sección se quitan las llaves al nombre de la macro letra?

   En Flex, las macros se definen en la sección de declaraciones y su sintaxis requiere que el nombre de la macro esté rodeado por llaves cuando se utilizan en la sección de expresiones regulares. Si eliminamos las llaves al nombre de la macro letra en la definición de las expresiones regulares, ocurrirá lo siguiente:

   El código original contiene:

   letra=[a-zA-Z]
   palabra=letra+

   Si eliminamos las llaves, se tiene:

   letra=[a-zA-Z]
   palabra=letra+

   Esto nos da como resultado un error de sintaxis, flex esperará que el nombre de la macro esté rodeado por llaves cuando se use en las expresiones regulares. Sin las llaves, Flex no reconocerá que letra es una macro definida previamente y tratará de interpretar letra literalmente como una expresión regular, lo cual carece de sentido en este contexto y causaría un error.

   Además, cuando intentemos generar el archivo Lexer.java usando jflex, es muy probable que veamos un mensaje de error indicando que letra no es una expresión regular válida. Algo parecido a ünrecognized regular expression.º un error relacionado con la macro no definida.

2. ¿Qué ocurre si en la segunda sección se quitan las llaves a las macros?

   En Flex, las llaves son necesarias para identificar el uso de macros dentro de las expresiones regulares en la segunda sección del archivo .flex. Estas macros se definen en la primera sección y se expanden en las expresiones regulares cuando están dentro de llaves.

   Si quitamos las llaves obtendríamos como resultado errores de interpretación, ya que Flex no reconocerá letra, espacio o digito como macros. Flex tratará de interpretar estas palabras como literales, es decir, buscará directamente las cadenas de texto letra, espacio, y digito en la entrada en lugar de expandir las macros a sus definiciones correspondientes ([a-zA-Z], [ ], [0-9]).

   Además, también podríamos obtener errores de compilación, ya que, flex generará un error durante la compilación, indicando que la expresión regular no es válida porque intentará procesar los nombres de las macros literalmente, lo cual no tiene sentido en el contexto de una expresión regular.

   Se podría obtener un comportamiento incorrecto, ya que, si por alguna razón Flex no muestra un error y se genera el archivo Lexer.java, el escáner no funcionará como se espera.

3. ¿Cómo se escribe un comentario en flex?

   En Flex, los comentarios se pueden escribir como en C/C++ o Java. Se tienen dos opciones:

   La primera, son los comentarios de una sola línea, usando //: todo lo que esté después de // en esa línea se considera un comentario.

   La segunda, comentarios de múltiples líneas, usando /* ... */: todo lo que esté entre /* y */ se considera un comentario que se puede extender a varias líneas.

4. ¿Qué se guarda en yytext?

   En Flex, yytext es una variable especial que contiene la cadena de texto que coincide con la expresión regular actual que el analizador léxico (lexer) ha reconocido.

   yytext guarda exactamente la secuencia de caracteres que hizo coincidir la expresión regular asociada con la acción léxica correspondiente.

   Además, yytext es un array de caracteres (en C/C++) o una cadena de texto (en Java). Es común usar yytext dentro de las acciones léxicas para realizar alguna operación con la cadena reconocida, como imprimirla, almacenarla, o analizarla más a fondo.

5. ¿Qué pasa al ejecutar el programa e introducir cadenas de caracteres y de dígitos en el archivo de entrada?

   Al ejecutar el programa Flex y proporcionar un archivo de entrada que contenga cadenas de caracteres y dígitos, el comportamiento depende de cómo se definieron las reglas en el archivo .flex.

   El comportamiento al ejecutar el programa será el siguiente:

   El analizador léxico leerá el archivo de entrada línea por línea.

   Con la coincidencia de patrones como espacios en blanco, estos (espacios, tabulaciones y saltos de línea) se ignoran debido a la acción léxica vacía en la regla espacio. Esto significa que no se imprimirá nada para estos caracteres.

   Para los dígitos, cada secuencia de uno o más dígitos (digito+) será identificada y se imprimirá un mensaje que indica el número encontrado.

   Para las palabras, cada secuencia de caracteres alfabéticos (palabra) será identificada y se imprimirá un mensaje que indica la palabra encontrada.

6. ¿Qué ocurre si introducimos caracteres como "*.^{en} el archivo de entrada?

   Cuando introducimos caracteres especiales como * en el archivo de entrada, el comportamiento del analizador léxico depende de cómo están definidas las reglas en el archivo .flex. En el código que estamos manejando, los caracteres especiales como * no están explícitamente manejados por ninguna regla, lo que significa que el analizador léxico los tratará de la siguiente manera:

   No se define ninguna regla para caracteres especiales, por lo tanto, esos caracteres no serán reconocidos por el analizador léxico y, como resultado, se ignorarán.

   No se imprime nada para caracteres no manejados, dado que no hay una regla que maneje el carácter *, no se generará alguna salida para estos caracteres. Por lo que, el analizador léxico los ignorará y continuará procesando el resto del archivo de entrada.

7. Modificar al código anterior en un archivo nuevo, de tal manera que reconozca lo siguiente:

   Para poder hacer esto posible, se deben definir las expresiones regulares correspondientes para cada tipo de token.

   EL código de este problema se encuentra en el archivo Analizador.flex

   El cuál se compilo con jflex analizador.flex. Mientras que el archivo JAVA generado se compilo con javac Lexer.java

   - La expresión regular para los hexadecimales en lenguaje Java.
   - 5 palabras reservadas del lenguaje Java.
   - Los identificadores válidos del lenguaje Java, con longitud máxima de 32 caracteres (Sugerencia: use el operador m,n).
   - Los espacios en blanco.