Diseño de Compiladores I – Cursada 2018
Trabajo Práctico Nro. 1

La entrega se hará en forma conjunta con el Trabajo Práctico Nro. 2 (Fecha de Entrega: 25-09-2018)

Objetivo

Desarrollar un Analizador Léxico que reconozca los siguientes tokens:

- Identificadores cuyos nombres pueden tener hasta 25 caracteres de longitud. El primer carácter debe ser un "_",y el resto pueden ser letras y dígitos. Los identificadores con longitud mayor deberán ser informados como error, y el token erróneo deberá ser descartado. Las letras utilizadas en los nombres de identificador pueden ser mayúsculas o minúsculas, y el lenguaje será case sensitive. Entonces, el identificador MyVariable, no será igual a myvariable.
- Constantes correspondientes a los temas particulares asignados a cada grupo. Las constantes fuera de rango deberán ser tratadas con la técnica de reemplazo, utilizando para ello, el valor más grande dentro del rango permitido. Tal situación se informará como Warning.
- Operadores aritméticos: "**+**", "**-**" ,"*****", "**/**".
- Operador de asignación: ":**=**"
- Comparadores: "**>=**", "**<=**", "**>**", "**<**", "**=**", "!="
- "**(**" "**)**" "**{**" "}" "**,**" y "**;**"
- Cadenas de caracteres correspondientes al tema particular de cada grupo.
- Palabras reservadas (en minúsculas):
  If, **else**, **end_if**, **print**
- y demás símbolos / tokens indicados en los temas particulares asignados al grupo.

El Analizador Léxico debe eliminar de la entrada (reconocer, pero no informar como tokens al Analizador Sintáctico), los siguientes elementos.

- Comentarios correspondientes al tema particular de cada grupo.
- Caracteres en blanco, tabulaciones y saltos de línea, que pueden aparecer en cualquier lugar de una sentencia.

Analizador Léxico. Especificaciones

a) El Analizador Léxico deberá leer un código fuente, identificando e informando:
   - Tokens detectados en el código fuente. Por ejemplo:
     Palabra reservada **if**
     (
     Identificador **_varx**
     +
     Constante entera **25**
     Palabra reservada **else**
     etc.
   - Errores léxicos detectados en el código fuente, indicando: nro. de línea y descripción del error. Por ejemplo:
     Línea 24 - Warning: Constante entera fuera del rango permitido
   - Contenidos de la Tabla de Símbolos.

Se sugiere la implementación de un consumidor de tokens que invoque al Analizador Léxico solicitándole tokens. En el trabajo práctico 2, esta funcionalidad estará a cargo del Analizador Sintáctico.

b) El código fuente **DEBE SER LEÍDO DESDE UN ARCHIVO**, y el nombre **DEBE PODER SER ELEGIDO** por el usuario del compilador. Se sugiere que la dirección del archivo pueda ser leída como primer argumento de la aplicación en la línea de comandos.

c) La numeración de las líneas de código debe comenzar en 1. Si se implementa una interfaz que permite mostrar o editar el código fuente, incluir alguna manera de identificar el número de cada línea del código.

d) Para la programación se podrá elegirel lenguaje. Para esta elección, tener en cuenta que el analizador léxico se integrará luego a un Parser (Analizador Sintáctico) generado utilizando una herramienta tipo Yacc. Por lo tanto, es necesario asegurarse la disponibilidad de dicha herramienta para el lenguaje elegido.

e) El Analizador Léxico deberá implementarse mediante una matriz de transición de estados y una matriz de acciones semánticas, de modo que cada cambio de estado y acción semántica asociada, sólo dependa del estado actual y el carácter leído.

f) Implementar una Tabla de Símbolos donde se almacenarán identificadores, constantes, y cadenas de caracteres. Es requisito para la aprobación del trabajo, que la tabla sea implementada con una estructura dinámica.

g) La aplicación deberá mostrar, además de tokens y errores léxicos, los contenidos de La Tabla de Símbolos. Puede mostrarse tanto en una interfaz como generarse archivos de texto con este contenido.

Entrega

La forma de entrega (correo electrónico, medio físico, etc.) se pactará con el docente asignado al grupo.
El material entregado debe incluir:

- Ejecutable del compilador (debe poder ejecutarse en una máquina virtual que será provista por la cátedra)
- Código fuente completo del compilador
- Casos de prueba
- Informe

Consideraciones

- Debe controlarse que las constantes estén dentro del rango permitido. Si esta condición no se cumple, se debe considerar que la constante no es válida.
- **Para aquellos grupos que tienen asignados tipos de datos que pueden llevar signo, la distinción del uso del símbolo '-' como operador aritmético o signo de una constante, se postergará hasta el trabajo práctico Nro. 2.**

Informe:

Debe incluir:
- **NRO. DE GRUPO** e Integrantes. Incluir **DIRECCIONES DE CORREO**de todos los integrantes para contacto.
- Temas particulares asignados (esta información deberá repetirse en los informes de los trabajos prácticos subsiguientes).
- Introducción.
- Decisiones de diseño e implementación.
- Diagrama de transición de estados.
- Matriz de transición de estados.
- Descripción del mecanismo empleado para implementar la matriz de transición de estados y la matriz de acciones semánticas.
- Lista de acciones semánticas asociadas a las transiciones del autómata del Analizador Léxico, con una breve descripción de cada una.
- Errores léxicos considerados.

Este informe deberá ser completado con las consignas indicadas en el Trabajo Práctico 2.

Casos de Prueba

Se debe incluir, **como mínimo**, ejemplos que contemplen las siguientes alternativas:
(Cuando sea posible, agregar un comentario indicando el comportamiento esperado del compilador)
Formato del archivo de texto: TP1_<número de ejemplo>_<subíndice>.txt

1. Constantes con el primer y último valor dentro del rango.
2. Constantes con el primer y último valor fuera del rango.
3. Para números reales, además, mantisa con y sin parte decimal (a-b), con y sin exponente (c-d), con exponente positivo y negativo (e-f).
4. Identificadores de menos y más de 25 caracteres.
5. Identificadores con "_", letras, y dígitos,
6. Identificadores mal definidos.
7. Palabras reservadas escritas en minúsculas y mayúsculas.
8. Comentarios bien y mal definidos.
9. Cadenas bien y mal definidas.

Temas particulares

Cada grupo de trabajo tendrá asignada una combinación de temas particulares.
La información de los temas asignados a cada grupo, estará disponible en: http://diseno-de-compiladores-i.alumnos.exa.unicen.edu.ar/:

1. **Enteros**: Constantes enteras con valores entre -2^{15} y $2^{15} - 1$. Estas constantes llevarán el sufijo "**_i**". Se debe incorporar a la lista de palabras reservadas la palabra **integer**.
2. **Enteros sin signo:** Constantes con valores entre 0 y $2^{16} - 1$. Estas constantes llevarán el sufijo "**_ui**".

   Se debe incorporar a la lista de palabras reservadas la palabra **usinteger**.
3. **Enteros largos:** Constantes enteras con valores entre -2^{31} y $2^{31} - 1$. Estas constantes llevarán el sufijo "**_l**".

   Se debe incorporar a la lista de palabras reservadas la palabra **linteger**.
4. **Enteros largos sin signo:** Constantes enteras con valores entre 0 y $2^{32} - 1$. Estas constantes llevarán el sufijo "**_ul**".

   Se debe incorporar a la lista de palabras reservadas la palabra **uslinteger**.
5. **Flotantes:** Números reales con signo y parte exponencial. El exponente comienza con la letra F mayúscula y puede tener signo. La ausencia de signo implica positivo. La parte exponencial puede estar ausente. El símbolo decimal es el punto ".".

   Ejemplos válidos: 1. .6 -1.2 3.F–5 2.F+34 2.5F1 15. 0.

   Considerar el rango 1,17549435E-38 < x< 3,40282347E38 (incluir el 0,0)

   Se debe incorporar a la lista de palabras reservadas la palabra **single**.
6. **Dobles:** Números reales con signo y parte exponencial. El exponente comienza con la letra D mayúscula y puede tener signo. La ausencia de signo implica positivo. La parte exponencial puede estar ausente. El símbolo decimal es el punto ".".

   Ejemplos válidos: 1. .6 -1.2 3.D–5 2.D+34 2.5D1 13. 0.

   Considerar rango2,2250738585072014E-308 <x < 1,7976931348623157E308(incluir el 0,0)

   Se debe incorporar a la lista de palabras reservadas, la palabra **double**.
7. Incorporar a la lista de palabras reservadas, la palabra **while**.
8. Incorporar a la lista de palabras reservadas, las palabras **loop** y **until**.
9. Incorporar a la lista de palabras reservadas, la palabra **for**.
10. Incorporar a la lista de palabras reservadas, las palabras **case** y **do**.
11. Incorporar el símbolo "**&**", y agregar a la lista de palabras reservadas, las palabras **let** y **mut**.
12. Incorporar a la lista de palabras reservadas, las palabras **void**, **fun** y **return**.
13. Incorporar a la lista de palabras reservadas, la palabra **void**.
14. A definir en Prácticos 2/3.
15. A definir en Prácticos 2/3.
16. A definir en Prácticos 2/3.
17. **Comentarios de 1 línea:** Comentarios que comiencen con doble guión bajo "__" y terminen con el fin de línea.
18. **Comentarios multilínea:** Comentarios que comiencen con "#" y terminen con "#" (estos comentarios pueden ocupar más de una línea).
19. **Cadenas de 1 línea:** Cadenas de caracteres que comiencen y terminen con " ' " (estas cadenas no pueden ocupar más de una línea).
20. **Cadenas multilínea:** Cadenas de caracteres que comiencen y terminen con " ' " . Estas cadenas pueden ocupar más de una línea, y en dicho caso, al final de cada línea, excepto la última, debe aparecer un guión " - ". (En la Tabla de símbolos se guardará la cadena sin el guión, y sin el salto de línea.