Universidad Simón Bolívar
Departamento de Computación y Tecnología de la Información
CI3725 – Traductores e Interpretadores
Enero – Marzo 2020

**Version: 27/01/2020**

# Interpretador de `Willy`* <br> Análisis Lexicográfico

La primera etapa del proyecto corresponde al módulo de análisis lexicográfico del interpretador del lenguaje `Willy`*. El analizador lexicográfico debe aceptar como entrada cualquier secuencia de caracteres provenientes del teclado o de un archivo de texto y producir como salida la lista con los *tokens* relevantes reconocidos. Si se reciben caracteres que no corresponden a ningún token del lenguaje `Willy`*, debe producirse un mensaje de error. Tanto los tokens reconocidos como los mensajes de error deben acompañarse de la posición en el archivo de entrada (línea y columna) en la cual han sido encontrados.

Los tokens relevantes son:

- Las palabras claves y funciones predefinidas utilizadas en la sintaxis de `Willy`*.
- Los identificadores utilizados en los mundos y procedimientos. A diferencia de las palabras claves, los identificadores corresponden a un token llamado *TkId*. Este token siempre tendrá asociado como atributos el identificador particular reconocido, y el tipo de identificador: mundo, tipo de objeto, booleano, condición objetivo, o instrucción.
- Cada uno de los símbolos que denotan separadores en `Willy`*.
- Los números enteros que corresponden a un token llamado *TkNum*. Este token siempre tendrá asociado como atributo el valor entero reconocido.

Los espacios en blanco, tabuladores y saltos de línea deben ser ignorados. Si se encuentran dos guiones consecutivos, i.e. –, tanto ellos como todos los caracteres hasta el final de la línea deben ser ignorados, ya que definen un comentario de línea. También debe ignorarse el contenido de los comentarios "en bloque", los cuales se construyen empleando dos llaves '{{' para abrirlos y dos llaves '}}′ para cerrarlos. No se permite anidar comentarios en bloque. Cualquier otro caracter encontrado debe ser reportado como un error.

## Primera Entrega

- La primera entrega consiste de implementar el módulo lexicográfico en el lenguaje de su elección como se discutió en clase. Según el lenguaje elegido, el proyecto consiste en definir las expresiones regulares para la herramienta apropiada de construcción del *lexer*, y uno o más programas que invoquen de forma adecuada al lexer para procesar la entrada.
- Todos los módulos del sistema deben estar completa y correctamente documentados. Tambíen debe entregarse un pequeño informa en formato `.md` en el repositorio Github del grupo, en el directorio correspondiente a la primera entrega.
- Un `Makefile` que permita compilar el programa.
- El programa principal será compilado y utilizado desde la línea de comandos de dos manera diferentes:

1. Si se invoca el programa suministrando exactamente un argumento en la línea, el programa debe intentar abrir ese archivo y procesarlo. El procesamiento debe retornar todos los tokens encontrados o todos los errores lexicográficos detectados; e.g.

```
$ willy test1.txt
... lista de tokens o error ...
$
```

   Si el archivo suministrado como argumento no existe o no puede abrirse, el programa debe terminar indicando el problema. Es inaceptable que el programa aborte o tenga una terminación anormal, e.g.

```
$ willy noexiste.txt
Imposible abrir el archivo noexiste.txt
$
```

2. Si se invoca el programa sin argumentos, el programa debe ofrecer un *prompt* en el cual escribir el nombre del archivo a procesar. El procesamiento debe retornar todos los tokens procesados o todos los errores detectados en la línea.

```
$ willy
Archivo a Interpretar: test.txt ... tokens o errores ...
$
```

3. Los errores lexicográficos deben ser reportados de manera clara y acompañados de la información de línea y columna en el archivo de entrada; e.g.,

```
$ willy test1.txt
Caracter ilegal '%' encontrado en línea 12, columna 42.
$
```

- Actualización sobre formato de salida. Se debe imprimir cada token encontrado en el orden que el lexer lo encontró, agrupando los tokens por línea. Cada token impreso debe tener información sobre el mismo. En el caso de los identificadores y números se debe mostrar el string capturado, para todos los demás tokens se debe imprimir la línea y columna). Por ejemplo, para el programa:

```
1 begin-world␣mimundo
2 ␣␣World␣1␣id
3 ␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣blue
4 end-work
```

  Los números de línea al lado izquierdo son para describir de forma clara la entrada y salida. También hemos representado los espacios de forma explícita para evitar confusiones. La salida debe ser algo similar a lo siguiente:

```
1 TkBeginWorld(linea=1,␣columna=1)␣TkId(valor="mimundo",␣linea=1,␣columna=13)
2 ␣␣TkWorld(linea=2,␣columna=3)␣TkNum(valor=1,␣linea=2,␣columnna=9)␣TkId(valor="id",␣linea=2,␣columna=11)
3 ␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣TkBlue(linea=3,␣columna=15)
4 TkEndWork(linea=4,␣columna=1)
```

  Nótese que se debe respetar la identación del primer token de cada línea de acuerdo a la encontrado en el archivo. Los siguientes tokens pueden ser impresos dejando un (1) espacio de por medio. Las líneas en blanco se deben hacer notar en la salida de su programa: si hay $n$ líneas blancas entre dos líneas con tokens, se debe dejar esa cantidad de líneas en blanco en la salida entre las dos líneas que si poseen tokens. Los comentarios **no** deben ser impresos (ni usados en etapas posteriores) por el lexer, y debe tratarse similar a las líneas en blanco: si un comentario ocupa $n$ líneas (independientemente del tipo de comentario (single-line o multi-line), se debe dejar $n$ líneas en blanco. Por ejemplo, para el siguiente programa:

```
{{
␣comentario␣de␣prueba

␣␣␣␣␣␣␣␣␣␣␣␣}}
begin-world␣mimundo

␣␣World␣1␣id
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣blue␣Place
--␣to␣ignore
end-work␣--␣Comentario␣en␣la␣misma␣línea␣que␣otro␣token.␣Se␣ignora␣el␣comentario␣pero␣se␣procesan␣los␣tokens
```

La salida debe retornar algo similar a lo siguiente:

```


TkBeginWorld(linea=5,␣columna=1)␣TkId(valor="mimundo",␣linea=5,␣columna=13)

␣␣TkWorld(linea=7,␣columna=3)␣TkNum(valor=1,␣linea=7,␣columnna=9)␣TkId(valor="id",␣linea=7,␣columna=11)
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣TkBlue(linea=8,␣columna=15)␣TkPlace(linea=8,␣columna=20)

TkEndWork(linea=10,␣columna=1)
```

- El proyecto se entregara en un repositorio **privado** en donde el preparador y profesor serán colaboradores.
- **Fecha de Entrega:** Martes 28 de Enero 2020 hasta la media noche.
- **Valor:** 5 puntos.