EINA - Universidad de Zaragoza
Departamento de Informática e Ingeniería de Sistemas
3º Grado de Ingeniería Informática
30230 Procesadores de Lenguajes. **Curso 2018-2019**
Profesor: Javier Fabra – jfabra@unizar.es

==========

PRÁCTICA 4: Análisis semántico.
Parte II: Implementación de un analizador semántico para MiniLeng

==========

I. OBJETIVOS

En esta práctica vas a aplicar los conceptos vistos en clase para análisis semántico, con el objetivo de implementar el analizador semántico de un lenguaje estructurado sencillo como es MiniLeng.

II. CONTENIDO

ATENCIÓN: Para realizar esta segunda parte de la práctica, es requisito indispensable haber completado todas las prácticas anteriores

En esta práctica vamos a continuar trabajando con la tabla de símbolos para implementar **un analizador semántico del lenguaje Minileng**. A continuación se detalla la semántica del lenguaje. Considera también las características de MiniLeng que se introdujeron en la Práctica 2.

1. Se permiten comentarios de una línea, comenzados por % y **terminados por el fin de línea**. **Se deben permitir también los comentarios multilínea, que comienzan por '%%' y terminan por '%%'**

2. **Todo bloque debe contener, al menos, una instrucción.** No se permiten procedimientos ni bloques que no contengan, al menos, una instrucción. El punto y coma no representa ninguna instrucción.

3. Se permite la declaración y uso de variables simples, tanto globales como locales, de tres tipos: **entero, carácter y booleano**. La declaración de variables sigue la sintaxis de C, y permite más de una variable en cada declaración:

```
entero i, j, k;
caracter c, d, e;
booleano v, f;
entero z;
```

4. Ningún símbolo puede llamarse como las palabras reservadas. Es decir, no se permite declarar una variable o una función de nombre 'entero' o 'programa', por ejemplo.

5. Se permite el uso de cadenas de caracteres constantes, aunque **solamente para escritura**:

```
escribir("Escribe un numero: ");
```

NOTA: Las cadenas y los caracteres simples deben ir siempre entre comillas dobles, no se admite la utilización de comillas simples.

6. Se permite la declaración y uso de acciones (procedimientos) anidadas con paso de **parámetros simples** tanto por valor como por referencia. Si el procedimiento no tiene parámetros se debe permitir tanto especificar los paréntesis sin contenido como no poner los paréntesis. En el caso de paso de parámetros por valor, se precederá la signatura del parámetro con la palabra reservada **val**, mientras que en el paso por referencia se utilizará la palabra **ref**. Es obligatorio especificar si el parámetro se pasa por valor o por referencia.

```
accion procesar( val entero i, j, k; ref caracter c, d, e;
                 val booleano f);
```

7. Los parámetros que se pasan por valor no se pueden utilizar en la parte izquierda de una asignación, ni como argumento de la operación 'leer()', ni pasarse como argumento por referencia a un procedimiento.

8. Los parámetros que se pasan por referencia se pueden utilizar en una expresión o como argumento de la operación 'escribir()', y pasarse como argumento por valor a un procedimiento. Sin embargo, se corre el riesgo de que el parámetro no haya sido inicializado (más adelante veremos cómo solucionar y detectar esta situación).

9. La operación de lectura sólo admite como argumentos variables o parámetros que se puedan asignar y del tipo entero o carácter (¡o desconocido!). No se le puede pasar como argumento una expresión o un parámetro por valor.

10. Se permite la escritura de variables simples. Como salida, la operación de escritura mostrará por pantalla el valor entero, y las cadenas "Verdadero" o "Falso" en el caso de booleanos ("Verdadero" si el valor del booleano es 1, "Falso" si es 0).

11. Si en una expresión aparece un identificador que no ha sido declarado, se dará un mensaje de error semántico **y se introducirá en la tabla de símbolos con el tipo DESCONOCIDO**. El tipo a propagar en la expresión se deja a vuestra elección.

12. Deben controlarse los siguientes errores de desbordamiento en las expresiones, **como mínimo**:

    a. División y módulo por cero cuando la expresión es constante.
    b. El argumento de la operación 'entacar()' debe ser del tipo entero y comprendido entre el 0 y el 255 (ver http://www.asciitable.com/ para saber los caracteres a los que se refiere este rango), cuando la expresión es constante.
    c. Todos los errores de desbordamiento adicionales que se os ocurran se considerarán mejoras realizadas sobre la práctica.

13. Todos los bloques de instrucciones (en los procedimientos) terminan con la palabra reservada "`Fin`", así como el programa principal (que al fin y al cabo es un bloque de instrucciones de nivel 0).

14. El compilador debe continuar el proceso cuando detecte un error semántico (igual que ocurría con los errores sintácticos), si bien puede detener la compilación y terminar la ejecución si el error es léxico. En caso de error, se mostrará la línea, columna, y un mensaje lo más detallado posible sobre el error.

III: TAREAS

Considerando la semántica del lenguaje, deberás:

- Implementar el análisis semántico relativo a la declaración y utilización de variables globales y locales, la utilización de expresiones, y la declaración e invocación de acciones en todas las sentencias y estructuras del lenguaje.
- Debes modificar la gramática para implementar de forma explícita la precedencia y asociatividad de los operadores (en caso de que no lo tengas ya), así como para añadir las características del lenguaje descritas en este guión.
- Debes diseñar una **batería de pruebas de 5 programas** que ilustren el funcionamiento de la práctica. Los programas no tienen por qué ser complejos, sino que deben mostrar los puntos y características a destacar de la práctica realizada. Se valorará la capacidad de síntesis y madurez a la hora de desarrollarlos.
- Debes comprobar el correcto funcionamiento de la práctica con los programas que se os proporcionaron en prácticas anteriores. Si la práctica no funciona con los programas propuestos **no se superará.** Si los supera, podéis estar seguros de que, como mínimo, la práctica está aprobada.
- Debes tomar las decisiones de diseño que consideres oportunas para que tu compilador funcione adecuadamente. A este respecto, se valorará la capacidad de síntesis y demostración en la evaluación de la práctica.

Hay que considerar un aspecto importante sobre el "formato" del compilador, y es que **su ejecución debe realizarse pasándole como parámetro el nombre del fichero a compilar (sin la extensión .ml). La ejecución debe ejecutarse desde la línea de comandos**.

Además, genera **cinco** programas de prueba (con extensión .ml) que hayas realizado para comprobar el correcto funcionamiento de la práctica. No es necesario que sean muy complejos, pero sí que demuestren la corrección de la práctica de forma clara y concisa.