Lenguaje mini

Vamos a definir informalmente y mediante ejemplos un lenguaje al que llamaremos “mini” (por lenguaje mínimo).

A efectos de ilustrar la explicación comenzaremos con un ejemplo de un programa correcto en lenguaje mini:

```
programa nombreprog
    entero var1;
    entero var2;
    leer (var1, var2);
    entero total; // comentario
    total << var1 + 3 * -var2;
    var2 << 2 * (5 + 4);
    escribir (total%2);
fin-programa
```

Un programa correcto en `lenguaje` mini comienza con la palabra reservada **`programa`** seguido del nombre del mismo y termina con la palabra reservada **`fin-programa`**. El nombre del programa debe cumplir las reglas de ser un identificador. En medio debe haber una lista, no vacía de sentencias. Para simplificar consideraremos que las declaraciones son un tipo de sentencia. Las declaraciones de variables pueden ocurrir en cualquier lugar, pero debe ser antes de usar esa variable en cualquier otro tipo de sentencia.

Todas las sentencias y declaraciones finalizan con **`;`**
Los espacios en blanco (espacios, tabuladores, nueva línea) se ignoran, si bien sirven como centinelas.

Se puede agregar comentarios, los mismos comienzan con **`//`** y terminan con nueva línea, es decir todo lo que sigue a **`//`** es comentario hasta el final de la línea.

Las variables son todas, implícitamente, enteras. La declaración comienza con la palabra reservada **`entero`** seguida del identificador y finaliza con **`;`**

Los identificadores de las variables deben comenzar con una letra (minúscula o mayúscula, no importa) y luego pueden seguir con letras o dígitos decimales. Las palabras reservadas son en minúsculas tal como se las muestra en este documento.

Las sentencias son muy similares a las de lenguaje micro, de hecho tenemos los mismos tres tipos: **`leer`** es igual a micro, **`escribir`** también es igual y asignación es similar, pero el símbolo para la asignación es **`<<`** y cambian las expresiones.

Las expresiones en lenguaje mini tienen los operadores multiplicación, división y resto mediante los símbolos **`* /`** y **`%`**. Los operadores de suma y resta son **`+`** y **`-`** . En todos los casos la asociatividad es de izquierda a derecha. La precedencia de **`*`** , **`/`** y **`%`** es la misma y es mayor que la de **`+`** y **`-`** que tienen entre ellos igual precedencia.

Además el símbolo **`-`** puede actuar como operador unario para invertir el signo del operando que esté a su derecha. Este operador tiene mayor precedencia que **`* , /`** y **`%`**.

Las siguientes expresiones con correctas en lenguaje mini:

```
-x * y; // primero invierte el signo de x y luego multiplica por y
x + -y; //a x le suma lo que da invertir el signo de y
x--y;   // a x le resta lo que da invertir el signo de y
```

Además tenemos paréntesis para alterar las precedencias. Los operandos además de variables pueden ser constantes enteras, que se forman con dígitos decimales, por tanto solo puede representar números positivos, pero se puede usar el menos unario para invertir el signo si hace falta.

Aspectos semánticos

Las variables deben declararse antes de usarse y no es correcto duplicar una declaración. Por este motivo pueden ocurrir dos tipos de errores semánticos:

- definir más de una vez una variable (redeclarar)
- utilizar en las sentencias una variable no declarada.

Manejo de errores

No es parte del lenguaje en si, pero vamos a describir los tipos de errores léxicos que queremos detectar

- **Error común**: formado por la secuencia de uno o más caracteres inválidos, los que en un autómata hecho a mano ubicaríamos en la columna “otros”, como @ o !. Ejemplo: !::&
- **Error de identificador**: si un identificador es seguido de al menos un carácter inválido y luego es seguido de caracteres inválidos y/o caracteres que pueden formar parte de un identificador. Ejempos: total$, total2@a!1
- **Error de constante**: toda cadena que comience con dígitos y siga con letras o dígitos, por ejemplo 123ab , 56A7