**El lenguaje LM2**

Dentro de la asignatura “Procesadores de lenguajes” se implementará un lenguaje de programación llamado LM2. El nombre del lenguaje es el acrónimo de **L**enguaje **M**odular versión **2**.

En este documento se explicará la naturaleza del lenguaje, aunque no basada estrictamente desde lo formal.

Para describir el lenguaje nos basaremos en un ejemplo y en explicaciones posteriores. El ejemplo presentado es el de continuación (hay que hacer solamente hincapié en su sintaxis):

const M : integer = 7, R : integer = 90;

var N, S : integer;

var A[12] : integer, C[5] : boolean;

procedure PROC1();

var A : integer;

var B : boolean;

begin

B := not C[3] and (N < A or S <> R);

C[1] := true;

while B do

if A <= 10 then

A := A + 1;

end-if;

showLN 'Visualizacion', B, ' ', A;

B := C[3] and A + N < R;

end-while;

end-proc;

procedure PROC2(byref R : integer);

const T : integer = 67;

var W11, W12, Q : integer;

begin

C[2] := false;

S := (S + M) \* 2;

Q := 1;

read W11; {lectura de teclado}

W12 := AW11 \* 2 + S;

while (W12 - 2) <= R + S do

W12 := W12 \* 2;

end-while;

while W12 + M > Y / 2 do

Q := Q \* 2;

W12 := W12 / 2;

if W12 <= R then

R := R - W12;

Q := Q + 1;

end-if;

end-while;

A[1] := Q;

end-proc;

function FUN1(T: integer, byval N2 : integer) : boolean;

var N: integer;

begin

if T > 0 then

N := -45;

else

N := 70;

end-if;

end-func N \* 2 >= 0 or not N2 = 0;

procedure MAIN();

var X: boolean;

begin

PROC1();

S := M + 1;

PROC2(S);

X := FUN1(5, 8);

C[5] := X;

show A[S + 1];

end-proc;

Entre las características importantes del lenguaje podemos notar:

a) Cada programa se conforma de un bloque de componentes globales, luego declaraciones de procedimientos o funciones globales (donde cada uno conforma un bloque), y finalmente un bloque de componentes que conforma el bloque principal (procedimiento **main**). Dentro de un bloque se pueden definir constantes o variables cuyos alcances están determinados por el del bloque que las contiene. Dentro de un procedimiento se pueden definir constantes o variables, que serán locales al procedimiento o función donde se ubican. Las constantes se pueden leer, pero no modificar su contenido.

b) Toda variable usada debe estar declarada como componente de un bloque accesible desde donde se la utiliza. En el caso de existir dos variables de distintos ámbitos con el mismo nombre, tendrá prioridad el ámbito más cercano (primero la variable local y luego la global).

c) Los nombres de variables y de procedimientos y funciones deben ser de una letra, seguida de una determinada cantidad de letras o números.

d) Las variables y constantes solamente podrán ser de tipo **integer** o **boolean**, que se representan internamente con dos bytes. Se pueden armar expresiones de tipo entero utilizando los operadores de suma (+), resta (-), multiplicación (\*) y división entera (/). La suma y la resta tienen la menor precedencia, y se podrán utilizar paréntesis para cambiar la precedencia en la evaluación de una expresión determinada.

Los operadores relacionales son =, <>, >, >=, < o <= y requieren dos expresiones numéricas.

Existen operaciones para armar expresiones booleanas utilizando los operadores **and**, **or** y **not**. En este caso la menor precedencia la tiene el operador **or**, y la mayor el operador **not**.

Las constantes booleanas **true** y **false** son parte del lenguaje.

e) Existen también los **arrays** definidos con una longitud fija -en el lugar de su definición- , y donde sus elementos son de tipo **integer** o **boolean** (todos sus elementos del mismo tipo). Los arrays solamente podrán ser globales. Se podrán acceder a cada uno de sus elementos tanto para obtener su valor contenido como asignables dentro de una operación de asignación. Los subíndices de los arrays pueden ser accedidos como expresiones de tipo **integer** en general, tanto en su uso como expresiones como asignables. Sin embargo no se permiten subíndices con valor negativo.

f) Existe el pasaje de parámetros de expresiones de tipo entero o natural de un procedimiento/función llamador a variables de un procedimiento/función llamado, donde se indica el tipo de pasaje de parámetros antes del nombre de la variable en el parámetro formal. Los tipos de pasajes de parámetros podrán ser **byval** (por valor) o **byref** (por referencia). El pasaje por parámetro por default (si no se indicara) es **byval**. El valor retornado por una función está indicado como la expresión a retornar, que deberá indicarse junto a su última línea (en **end-func**). Las posiciones de un array también se podrán pasar como parámetro por referencia, sin importar si el procedimiento/función que invoca es el procedimiento **main**.

g) Existen estructuras de control similares a los de los lenguajes estructurados: **estructura de decisión** (bloque **if..then..end-if**, **if..then..else..end-if**), y **estructura de iteración** (bloque **while..do..end-while** solamente).

h) Como sentencias simples existen la asignación, la sentencia **show** y **showln**, que visualizan los valores y expresiones escritos a continuación (la sentencia **showln** agrega al final un salto de línea), la sentencia **read**, que lee del teclado un número ingresado para asignarlo a la variable indicada, y la sentencia de llamada a un procedimiento (que no se indica con una sentencia explícitamente).

i) Todas las sentencias de código o de declaración de variables o constantes deben estar finalizadas por el caracter “**;**”.

j) Pueden aparecer literales string (encerrados entre comillas simples) como uno de los parámetros de la sentencia **show** o **showln**. Es el único lugar donde pueden aparecer constantes string.

k) Los espacios y otros caracteres separadores deberán ser ignorados, al igual que los comentarios, que representan textos encerrados entre caracteres { y }.

l) Se deberán realizar chequeos de tipos en forma estática, aunque no deberá existir coerción de tipos (conversiones automáticas de tipos cuando sea necesario).

m) El programa objeto deberá realizar verificaciones en tiempo de ejecución para capturar el acceso a un array con un subíndice fuera de rango. En esos casos, se mostrará un mensaje de error y la ejecución del programa abortará. No hay que preocuparse por los errores de overflow.

n) Asumir que podrán existir llamados recursivos (directos o indirectos) de procedimientos o funciones.