SINTAXIS DE MODULA2

METASIMBOLOS

Utilizaremos la notación BNF (Backus-Naur Form) para definir las reglas sintácticas que definen el lenguaje Modula-2. En esta notación se emplean los cinco metasimbolos siguientes:

::= Metasimbolo de definición. El elemento a su izquierda se puede desarrollar segun el esquema de la derecha.

| Metasimbolo de alternativa. Puede elegirse uno y sólo uno de los elementos separados por este metasimbolo.

{ } Metasimbolos de repetición. Los elementos incluidos dentro de ellos se pueden repetir cero o más veces.

[ ] Metasimbolos de opción. Los elementos incluidos dentro de ellos pueden ser utilizados o no.

( ) Metasimbolos de agrupación. Agrupan los elementos incluidos en su interior.

UNIDAD DE COMPILACIÓN

```
Unidad_de_compilación ::=          Módulo_programa |
                                   Módulo_definición |
                                   Módulo_implementación

Módulo_programa ::=                Cabecera_módulo
                                   Bloque
                                   Identificador .

Módulo_definición ::=              Cabecera_definición {Definición_de_módulo}
                                   END Identificador .

Módulo_implementación ::=          IMPLEMENTATION Módulo_programa

Cabecera_módulo ::=                MODULE Identificador
                                   [Prioridad] ;
                                   {Lista_importados ;}

Cabecera_definición ::=            DEFINITION MODULE Identificador ;
                                   {Lista_importados ;}
                                   [Lista_exportados ;]

Lista_importados ::=               [FROM Identificador]
                                   IMPORT Lista_de_identificadores

Lista_exportados ::=               EXPORT [QUALIFIED] Lista_de_identificadores

Prioridad ::=                      Expresión_constante

Bloque ::=                         Parte_declarativa
                                   Parte_ejecutiva
                                   END

Definición_de_módulo ::=           Declaración_de_constantes |
                                   TYPE { Identificador [ = Esquema_de_tipo] ; } |
                                   Declaración_de_variables |
                                   Cabecera_subprograma
```

DECLARACIONES

```
Parte_declarativa ::=              {Declaración}

Declaración ::=                    Declaración_de_constantes |
                                   Declaración_de_tipos |
                                   Declaración_de_variables |
                                   Declaración_subprograma |
                                   Declaración_módulo

Declaración_de_constantes ::=      CONST {Asociación_constante ;}

Asociación_constante ::=           Identificador = Expresión_constante
```

SINTAXIS DE MODULA2

```
Declaración_de_tipos ::=          TYPE  {Definición_de_tipo ;}

Definición_de_tipo ::=            Identificador = Esquema_de_tipo

Esquema_de_tipo ::=               Tipo_simple | Tipo_conjunto |
                                  Tipo_formación | Tipo_registro |
                                  Tipo_puntero | Tipo_subprograma

Tipo_simple ::=                   Identificador_de_tipo |
                                  Tipo_enumerado |
                                  Tipo_subrango

Tipo_enumerado ::=                (Lista_de_identificadores)

Tipo_subrango ::=                 [Identificador_de_tipo]
                                  [Expresión_constante .. Expresión_contante]

Tipo_conjunto ::=                 SET OF Tipo_simple

Tipo_formación ::=                ARRAY Tipo_simple{,Tipo simple} OF Esquema_de_tipo

Tipo_registro ::=                 RECORD Secuencia_de_listas_de_campos END

Secuencia_de_listas_de_campos ::= Lista_de_campos {; Lista_de_campos}

Lista_de_campos ::=               [Lista_de_identificadores :
                                  Esquema_de_tipo | Parte_variante]

Parte_variante ::=                CASE [Identificador] : Identificador_de_tipo OF
                                  Variante { | Variante}
                                  [ELSE Secuencia_de_listas_de_campos]
                                  END

Variante ::=                      Lista_de_valores : Secuencia_de_listas_de_campos

Tipo_puntero ::=                  POINTER TO Esquema_de_tipo

Tipo_subprograma ::=              PROCEDURE [Lista_de_tipos_formales]

Lista_de_tipos_formales ::=([[VAR] Tipo_formal{, [VAR] Tipo_formal})
                                  [: Identificador_de_tipo]

Declaración_de_variables ::=      VAR {Lista_de_variables ;}

Lista_de_variables ::=            Lista_de_identificadores : Esquema_de_tipo

Declaración_subprograma ::=Cabecera_subprograma ;
                                  Bloque
                                  Identificador ;

Cabecera_subprograma ::=          PROCEDURE Identificador [Parámetros_formales]

Parámetros_formales ::=           ([Grupo_de_parámetros {; Grupo_de_parámetros}])
                                  [: Identificador_de_tipo]

Grupo_de_parámetros ::=           [VAR] Lista_de_identificadores : Tipo_formal

Declaración_módulo ::=            MODULE Identificador
                                  [Prioridad] ;
                                  {Lista_importados ;}
                                  [Lista_exportados ;]
                                  Bloque
                                  Identificador ;
```

SINTAXIS DE MODULA2

SENTENCIAS EJECUTABLES

```
Parte ejecutiva ::=                     [BEGIN
                                        Secuencia_de_sentencias]

Secuencia_de_sentencias ::=             Sentencia {; Sentencia}

Sentencia ::=                           [Sentencia_de_asignación|
                                        Sentencia_de_llamada_a_procedimiento |
                                        Sentencia_IF |
                                        Sentencia_CASE |
                                        Sentencia_WHILE |
                                        Sentencia_REPEAT |
                                        Sentencia_LOOP |
                                        Sentencia_FOR |
                                        Sentencia_WITH |
                                        Sentencia_RETURN |
                                        Sentencia EXIT]

Sentencia_de_asignación ::=             Variable := Expresión

Sentencia_de_llamada_a_procedimiento ::=(Identificador_de_procedimiento | Variable)
                                    [Parametros_de_llamada]

Sentencia_IF ::=                        IF Expresión THEN
                                        Secuencia_de_sentencias
                                        {ELSIF Expresión THEN
                                        Secuencia_de_sentencias}
                                        [ELSE  Secuencia_de_sentencias]
                                        END

Sentencia_CASE ::=                      CASE Expresión OF
                                        Caso {| Caso}
                                        [ELSE
                                        Secuencia_de_sentencias]
                                        END

Caso ::=                                Lista_de_valores : Secuencia_de_sentencias

Sentencia_WHILE ::=                     WHILE Expresión DO
                                        Secuencia_de sentencias
                                        END

Sentencia_REPEAT ::=                    REPEAT
                                        Secuencia_de_sentencias
                                        UNTIL Expresión

Sentencia_LOOP ::=                      LOOP
                                        Secuencia_de_sentencias
                                        END

Sentencia_FOR ::=                       FOR Identificador_de_variable := Expresión
                                        TO  Expresión
                                        [BY Expresión_constante] DO
                                        Secuencia_de_sentencias
                                        END

Sentencia_WITH ::=                      WITH Variable DO
                                        Secuencia_de_sentencias
                                        END
Sentencia_RETURN ::=                    RETURN [Expresión]

Sentencia_EXIT ::=                      EXIT
```

EXPRESIONES

SINTAXIS DE MODULA2

```
Lista_de_valores ::=              Valores {, Valores}

Valores ::=                       Expresión_constante [.. Expresión_constante]

Expresión_contante ::=            Expresión_constante_simple
                                  [Operador_comparador Expresión_constante_simple]

Expresión_constante_simple ::=    [+ | -] Término_constante
                                  {Operador_sumador Término_contante}

Término_constante ::=             Factor_constante
                                  {Operador_multiplicador Factor_constante}

Factor_constante ::=              Identificador constante |
                                  Número |
                                  Ristra |
                                  Conjunto |
                                  (Expresión constante) |
                                  NOT Factor_constante |
                                  Carácter_en_octal

Parámetros_de_llamada ::=         ([Lista de expresiones])

Lista_de_expresiones ::=          Expresión {, Expresión}

Expresión ::=                     Expresión_simple
                                  [Operador_comparador Expresión_simple]

Expresión_simple ::=              [+ | -] Término {Operador_sumador Término}

Término ::=                       Factor {Operador_multiplicador Término}

Factor ::=                        Variable |
                                  Identificador_de_procedimiento |
                                  Identificador_de_función |
                                  Identificador_de_constante |
                                  Número |
                                  Ristra |
                                  Conjunto |
                                  (Expresión) |
                                  NOT Factor |
                                  Carácter_en_octal |
                                  Llamada_a_función

Llamada_a_función ::=             (Identificador_de_función | Variable)
                                  [Parámetros_de_llamada]

Operador_comparador ::=           = | <> | # |  < | > | <= |  > | >= | IN

Operador_sumador ::=              + | - | OR

Operador_multiplicador ::=        * | / | DIV | MOD | & | AND
```

ELEMENTOS BÁSICOS

```
Tipo_formal ::=                   [ARRAY OF] Identificador_de_tipo

Variable ::=                      (Identificador_de_variable |
                                   Identificador_de campo)
                                  {.Identificador_de_campo |
                                  [Lista_de_expresiones] | ^}

Conjunto ::=                      [Identificador_de_tipo] {Lista_de_elementos}

Lista_de_elementos ::=            [Elementos {, Elementos}]
```

SINTAXIS DE MODULA2

```
Elementos ::=                   Expresión_constante[.. Expresión_constante]

Ristra ::=                      “{Carácter}” | ‘{Carácter}’

Número ::=                      Número_entero | Número_real

Número_entero ::=               dígito {dígito} |
                                dígito_octal {dígito_octal}B |
                                dígito_hexadecimal {digito_hexadecimal}H

dígito_hexadecimal ::=          dígito | A | B | C | D | E | F

dígito ::=                      dígito_octal | 8 | 9

dígito_octal ::=                0 | 1 | 2 | 3 | 4 | 5 | 6 | 7

Número_real ::=                 dígito {dígito} . {digito} [escala]

escala ::=                      E [+ | -] dígito {dígito}

Carácter_en_octal ::=           dígito_octal{digito_octal}C

Lista_de_identificadores ::=    Identificador {, Identificador}

Identificador_de_XXXX ::=       Identificador {. Identificador}

Identificador ::=               letra {letra | dígito}

letra ::=                       A | B | C | D | E | F | G | H | I | J | K | L |
                                M | N | O | P | Q | R | S | T | U | V | W | X |
                                Y | Z | a | b | c | d | e | f | g | h | i | j |
                                k | l | m | n | o | p | q | r | s | t | u | v |
                                w | x | y | z |
```