ANÁLISIS LÉXICO:
TINY(1)

GRUPO 15
Integrantes:

- Escobar Suarez, Daniela Alejandra
- Rodríguez Pereira, Diego Alejandro

TINY(1)
Clases Léxicas

Identificador (IDEN): nombre de una variable. Comienzan necesariamente por una letra, seguida de cero (ninguna) o más letras, dígitos o subrayado.
Int (ENT): define el tipo de variable como entera.
Real (REAL): define el tipo de variable como real.
Bool (BOOL): define el tipo de variable como booleana.
True(TRUE):palabra reservada **true**, para variables de tipo bool.
False(FALSE): palabra reservada **false**, para variables de tipo bool.
And (AND): Palabra reservada para aplicar la operación lógica "and" sobre variables de tipo bool.
Or (OR): Palabra reservada para aplicar la operación lógica "or" sobre variables de tipo bool.
Not (NOT): Palabra reservada para aplicar la operación lógica "not" sobre variables de tipo bool.
String (STRING): define el tipo de variable **string**.
Nulo(NULL): identifica la palabra de reservada del valor nulo 'null'.
Procedimiento(PROCEDIMIENTO): definición de la palabra reservada 'proc' que indica procedimiento.
If (IF): define la palabra reservada 'if' para código condicional.
Then (THEN): define la palabra reservada 'then' que indica código del 'if'.
Else (ELSE): define la palabra reservada 'else' complementaria a 'if'.
Endif (ENDIF): define la palabra reservada 'endif' que indica el final de los if y else.
While (WHILE): define la palabra reservada 'while' para bucle condicional.
Do (DO): define la palabra reservada 'do' que indica la sección de código de un bucle while.
Endwhile (ENDWHILE): define la palabra reservada 'endwhile' para indicar final del código del bucle while.
Call (CALL): define la palabra reservada 'call' para indicar la invocación de otro procedimiento.
Record (RECORD): define la palabra reservada 'record' para indicar el tipo registro.
Array (ARRAY): define la palabra reservada 'array' para identificar arreglos.
Of (OF): define la palabra reservada 'of' para indicar el tipo de un array.
Pointer (POINTER): define la palabra reservada 'pointer' para definir el tipo puntero.
New (NEW): define la palabra reservada 'new' para indicar la instrucción de reserva de memoria.
Delete (DELETE): define la palabra reservada 'delete' para indicar la instrucción de liberar la memoria.
Read (READ): define la palabra reservada 'read' para indicar la instrucción de lectura.
Write (WRITE): define la palabra reservada 'write' para indicar la instrucción de escritura.
Nl (Nl): define la palabra reservada 'nl' para indicar el salto de línea.
Var (VAR): define la palabra reservada 'var' para indicar la declaración de variables.
Type (TYPE): define la palabra reservada 'type' para indicar la declaración de nuevos tipos.

Cadena (LIT_CADENA): identificación de cadena de caracteres.
Números enteros (NUM_ENTERO): Los números enteros pueden comenzar por un signo o no. Seguidamente debe de aparecer una secuencia de 1 o más dígitos (no se admiten ceros no significativos a la izquierda).
Números reales (NUM_REAL): Los números reales tienen obligatoriamente una parte entera (cuya estructura es similar a la de los números enteros). Seguida de, o una parte decimal, o una parte exponencial, o de una parte decimal seguida de una parte exponencial.
Paréntesis de apertura (PAR_APER): Símbolo para indicar el inicio de un cambio de precedencias o asociatividades de los operadores '('.
Paréntesis de cierre (PAR_CIER): Símbolo para indicar el fin de un cambio de precedencias o asociatividades de los operadores ')'.
Corchete de apertura (CORCHE_APER): indica la apertura del símbolo corchete '{', utilizado para indicar el inicio de bloques de código.
Corchete de cierre (CORCHE_CIER): indica el cierre del símbolo corchete '}',utilizado para indicar el fin de bloques de código.
Corchete rectangular de apertura (CORCHE_RECT_APER): indica el inicio del símbolo de corchete rectangular de apertura '[', indicando el inicio de indexación de array (posición o tamaño del mismo).
Corchete rectangular de apertura (CORCHE_RECT_CIER): indica el inicio del símbolo de corchete rectangular de cierre ']', indicando el fin de indexación de array (posición o tamaño del mismo).
Asignación (IGUAL): Indica que a una instrucción que consta de un identificador, es seguida por el operador de asignación y luego una expresión.
Igual (IGUAL_IGUAL): Combinación de símbolos '==' para expresar la relación de igualdad.
Menor que (MENOR): Símbolo para indicar la operación lógica del 'menor que' (<).
Mayor que (MAYOR): Símbolo para indicar la operación lógica del 'mayor que' (>).
Menor o igual que (MENOR_IGUAL): Combinación de símbolos '<=' para expresar la relación de menor o igual.
Mayor o igual que (MAYOR_IGUAL): Combinación de símbolos '>=' para expresar la relación de mayor o igual.
Distinto (DISTINTO): Símbolo para aplicar la operación relacional "distinto que" sobre dos valores de tipo numérico o de tipo bool.
&& (SEP_SECCION): Con el reconocimiento de esta clase se separa la sección de declaraciones de la sección de instrucciones.
Parámetro por referencia (AMPERSAND): utiliza el símbolo ampersand '&' para indicar el paso de parámetro como puntero.
Flecha (FLECHA): utilización del símbolo '->' para indicar el acceso a registros.
Coma (COMA): es el uso del símbolo ',' para separar los parámetros en las funciones.
Punto y coma (PUNTO_COMA): Con el reconocimiento de esta clase se separa cada una de las declaraciones de la sección de declaraciones ';'.
Mas (MAS): Símbolo que representa la operación aritmética de la suma, al igual que es posible encontrarla como signo de los números enteros o reales '+'.

Menos (MENOS): Símbolo menos(-) que representa el operador binario o unario de la resta '-'.
Multiplicación (POR): Símbolo que representa la operación aritmética de la multiplicación '*'.
División (DIV): Símbolo que representa la operación aritmética de división '/'.
Módulo (MOD): Símbolo del porcentaje '%' para indicar la operación del módulo.
Punto (PUNTO): símbolo de '.' para indicar el acceso a registros.

Definiciones Regulares

Definiciones Auxiliares

Letra → [A-Z] | [a-z]
Digito → digitoPositivo | 0
DigitoPositivo → [1-9]
ParteEntera → digitoPositivo digito* | 0
ParteDecimal → digito* digitoPositivo | 0
ParteExponencial → [e|E] NumeroEntero

Definiciones Cadenas Ignorables

Ignorar → \b | \n | \r
Comentario → #[^\n]*

Definiciones Léxicas

IDEN → letra (letra | [0-9] | _)*
ENT → int
REAL → real
BOOL → bool
TRUE → true
FALSE → false
AND → and
OR → or
NOT → not
STRING→ string
NULL → null
PROCEDIMIENTO → proc
IF → if
THEN → then
ELSE → else
ENDIF → endif
WHILE → while
DO → do
ENDWHILE → endwhile
CALL → call
RECORD→ record
ARRAY → array

OF → of
POINTER → pointer
NEW → new
DELETE → delete
READ → read
WRITE → write
Nl → nl
VAR → var
TYPE → type
LIT_CADENA → “ [^ \b, \n, \r]*”
NUM_ENTERO → [\+|\−]?ParteEntera
NUM_REAL → [\+|\−]?ParteEntera[.ParteDecimal | ParteExponencial |.ParteDecimal ParteExponencial]
PAR_APER → (
PAR_CIER →)
CORCHE_APER → {
CORCHE_CIER → }
CORCHE_RECT_APER → [
CORCHE_RECT_CIER →]
IGUAL → \=
IGUAL_IGUAL → \=\=
MENOR → <
MAYOR → >
MENOR_IGUAL → <\=
MAYOR_IGUAL → >\=
DISTINTO → !\=
SEP_SECCION → &&
AMPERSAND → &
FLECHA → ->
COMA → ,
PUNTO_COMA → ;
MAS → \+
MENOS → \-
POR → *
DIV → /
MOD → %
PUNTO → .