UNIVERSIDAD
COMPLUTENSE
MADRID

Procesadores de Lenguajes

CURSO 2021/22

PRÁCTICA:
PRIMERA FASE

Grupo 5

Álvaro Corrochano López
Álvaro Álvarez Iglesias

Analizador léxico para Tiny(0)

Clases léxicas de Tiny(0)

- **IDEN:** Clase léxica multivaluada que denota un identificador. Admita cadenas que empiecen obligatoriamente por letra pueden contener letras, números y "_".

- **ENT:** Clase léxica multivaluada que denota un número entero. Un número entero es aquel que no contiene parte decimal y no puede empezar por 0 (a excepción del 0). Pueden comenzar opcionalmente con "+" o "-".

- **REAL:** Clase léxica multivaluada que denota un número real. Un número real puede comenzar opcionalmente con "+" o "-", seguidos de una parte entera obligatoriamente y posteriormente por una parte decimal, una parte exponencial o una parte decimal seguida de una exponencial.
  La parte decimal comienza por "." y luego es seguida de números del 0 al 9 sin acabar en 0 (a excepción de que tan solo haya un 0) y la parte exponencial comienza con "e" o "E" seguido de opcionalmente "+" o "-" y obligatoriamente un número entero.

- **BOOL:** Clase léxica multivaluada que denota un booleano. Un booleano puede ser o bien "true" o bien "false".

- **TENT:** Clase léxica univaluada que denota que el identificador a su derecha es de tipo entero. Es siempre de la forma "int".

- **TREAL:** Clase léxica univaluada que denota que el identificador a su derecha es de tipo real. Es siempre de la forma "real".

- **TBOOL:** Clase léxica univaluada que denota que el identificador a su derecha es de tipo booleano. Es siempre de la forma "bool".

- **PAP:** Clase léxica univaluada que denota un paréntesis de apertura. Es de la forma "(".

- **PCIERRE:** Clase léxica univaluada que denota un paréntesis de cierre. Es de la forma ")".

- **IGUAL:** Clase léxica univaluada que denota una asignación (la variable de su izquierda es igual a la expresión a la derecha). Es de la forma "**=**".

- **MAS:** Clase léxica univaluada que denota la operación aritmética suma. Es de la forma "+".

- **MENOS:** Clase léxica univaluada que denota la operación aritmética resta. Es de la forma “-“.

- **POR:** Clase léxica univaluada que denota la operación aritmética multiplicación. Es de la forma “*“.

- **DIV:** Clase léxica univaluada que denota la operación aritmética división. Es de la forma “/“.

- **MENOR:** Clase léxica univaluada que denota el operador relacional “menor que”. Es de la forma “<“.

- **MAYOR:** Clase léxica univaluada que denota el operador relacional “mayor que”. Es de la forma “>”.

- **MENOI:** Clase léxica univaluada que denota el operador relacional “menor o igual que”. Es de la forma “<=“.

- **MAYORI:** Clase léxica univaluada que denota el operador relacional “mayor o igual que”. Es de la forma “>=“.

- **IGIG:** Clase léxica univaluada que denota el operador relacional “igual que”. Es de la forma “==“.

- **NOIG:** Clase léxica univaluada que denota el operador relacional “distinto que”. Es de la forma “!=“.

- **OR:** Clase léxica univaluada que denota la operación lógica or. Es de la forma “or“.

- **NOT:** Clase léxica univaluada que denota la operación lógica not. Es de la forma “not“.

- **AND:** Clase léxica univaluada que denota la operación lógica and. Es de la forma “and“.

- **PUNTOCOMA:** Clase léxica univaluada que representa el separador de instrucciones. Es de la forma “;”.

- **SEPARADOR:** Clase léxica univaluada que representa el separador de secciones (sección de declaraciones y sección de instrucciones). Es de la forma “&&”.

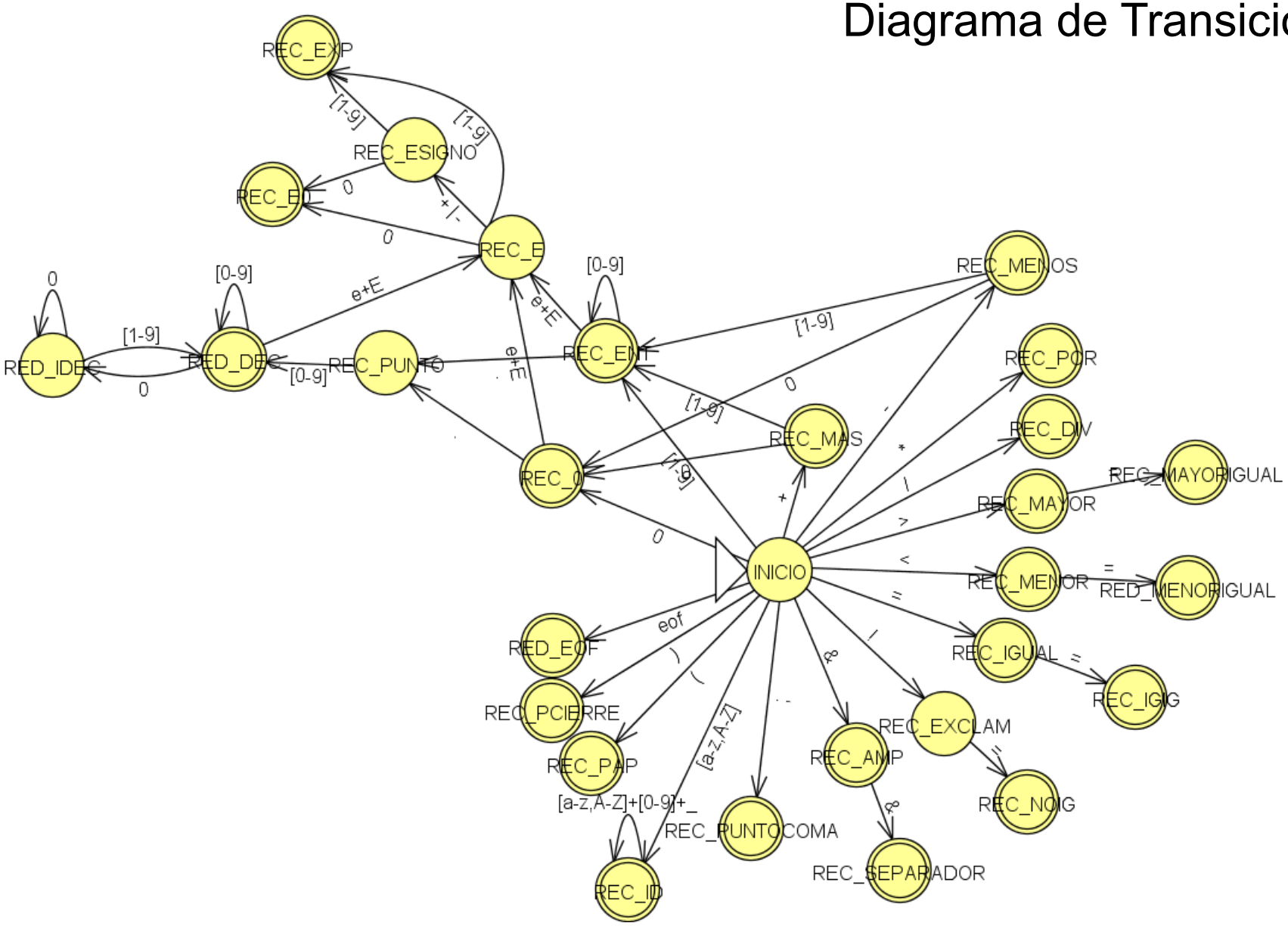
- **EOF:** Clase léxica univaluada que representa el fin de los caracteres.

Especificación formal del léxico del lenguaje mediante expresiones regulares

- **Letra ≡** [a - z ; A - Z]
- **Digito ≡** [0 - 9]
- **Dpos ≡** [1-9]
- **true ≡** true
- **false ≡** false
- **IDEN ≡** Letra(Letra+Digito+_)*
- **ENT ≡** [\+|\-]? (Dpos(Digito)*)+0
- **Pdec ≡** ((Digito*Dpos) + 0)
- **Pexp ≡** (e|E)ENT
- **REAL ≡** {ENT}((\.Pdec+Pexp)+(\.Pdec}{Pexp}))
- **TENT:** int
- **TREAL:** real
- **TBOOL:** bool
- **PAP:** \(
- **PCIERRE:** \)
- **IGUAL: \=**
- **MAS:** \+
- **MENOS:** \-
- **POR:** *
- **DIV:** /
- **MENOR: \<**
- **MAYOR: \>**

- **MENORI:** \<\=
- **MAYORI:** \>\=
- **IGIG:** \=\=
- **NOIG:** \!\=
- **OR:** or
- **NOT:** not
- **AND:** and
- **PUNTOCOMA:** \;
- **SEPARADOR:** &&
- **EOF:** ε

Diagrama de Transiciones



Analizador léxico para Tiny(1)

Clases léxicas de Tiny(1)

- **IDEN:** Clase léxica multivaluada que denota un identificador. Admita cadenas que empiecen obligatoriamente por letra pueden contener letras, números y “_”.

- **ENT:** Clase léxica multivaluada que denota un número entero. Un número entero es aquel que no contiene parte decimal y no puede empezar por 0 (a excepción del 0). Pueden comenzar opcionalmente con “+” o “-”.

- **REAL:** Clase léxica multivaluada que denota un número real. Un número real puede comenzar opcionalmente con “+” o “-”, seguidos de una parte entera obligatoriamente y posteriormente por una parte decimal, una parte exponencial o una parte decimal seguida de una exponencial.
  La parte decimal comienza por “.” y luego es seguida de números del 0 al 9 sin acabar en 0 (a excepción de que tan solo haya un 0) y la parte exponencial comienza con “e” o “E” seguido de opcionalmente “+” o “-” y obligatoriamente un número entero.

- **BOOL:** Clase léxica multivaluada que denota un booleano. Un booleano puede ser o bien “true” o bien “false”.

- **TENT:** Clase léxica univaluada que denota que el identificador a su derecha es de tipo entero. Es siempre de la forma “int”.

- **TREAL:** Clase léxica univaluada que denota que el identificador a su derecha es de tipo real. Es siempre de la forma “real”.

- **TBOOL:** Clase léxica univaluada que denota que el identificador a su derecha es de tipo booleano. Es siempre de la forma “bool”.

- **PAP:** Clase léxica univaluada que denota un paréntesis de apertura. Es de la forma “(“.

- **PCIERRE:** Clase léxica univaluada que denota un paréntesis de cierre. Es de la forma “)“.

- **IGUAL:** Clase léxica univaluada que denota una asignación (la variable de su izquierda es igual a la expresión a la derecha). Es de la forma “=“.

- **MAS:** Clase léxica univaluada que denota la operación aritmética suma. Es de la forma “+“.

- **MENOS:** Clase léxica univaluada que denota la operación aritmética resta. Es de la forma “-“.

- **POR:** Clase léxica univaluada que denota la operación aritmética multiplicación. Es de la forma “*“.

- **DIV:** Clase léxica univaluada que denota la operación aritmética división. Es de la forma “/“.

- **MENOR:** Clase léxica univaluada que denota el operador relacional “menor que”. Es de la forma “<“.

- **MAYOR:** Clase léxica univaluada que denota el operador relacional “mayor que”. Es de la forma “>”.

- **MENOI:** Clase léxica univaluada que denota el operador relacional “menor o igual que”. Es de la forma “<=“.

- **MAYORI:** Clase léxica univaluada que denota el operador relacional “mayor o igual que”. Es de la forma “>=“.

- **IGIG:** Clase léxica univaluada que denota el operador relacional “igual que”. Es de la forma “==“.

- **NOIG:** Clase léxica univaluada que denota el operador relacional “distinto que”. Es de la forma “!=“.

- **OR:** Clase léxica univaluada que denota la operación lógica or. Es de la forma “or“.

- **NOT:** Clase léxica univaluada que denota la operación lógica not. Es de la forma “not“.

- **AND:** Clase léxica univaluada que denota la operación lógica and. Es de la forma “and“.

- **PUNTOCOMA:** Clase léxica univaluada que representa el separador de instrucciones. Es de la forma “;”.

- **SEPARADOR:** Clase léxica univaluada que representa el separador de secciones (sección de declaraciones y sección de instrucciones). Es de la forma “&&”.

- **EOF:** Clase léxica univaluada que representa el fin de los caracteres.

- **CADENA:** Clase léxica multivaluada que representa una cadena de literales. Se trata de una secuencia de caracteres (a excepción de retroceso (\b), retorno de carro (\r), y salto de línea (\n)) de longitud 0 o mayor que comienza y acaba por “.

- **COMENTARIO:** Clase léxica multivaluada que denota un comentario de línea. Un comentario comienza por # seguido de una secuencia de cero o más caracteres, a excepción del salto de línea.

- **MODULO:** Clase léxica univaluada que denota la operación aritmética módulo. Es de la forma “%”.

- **CAP:** Clase léxica univaluada que denota un corchete de apertura. Es de la forma “[“.

- **CCIERRE:** Clase léxica univaluada que denota un corchete de cierre. Es de la forma “]“.

- **LAP:** Clase léxica univaluada que denota una llave de apertura. Es de la forma “{“.

- **LCIERRE:** Clase léxica univaluada que denota una llave de cierre. Es de la forma “}”.

- **PUNTO:** Clase léxica univaluada que denota un punto. Es de la forma “.“.

- **COMA:** Clase léxica univaluada que denota una coma. Es de la forma “,“.

- **FLECHA:** Clase léxica univaluada que denota una flecha. Es de la forma “->“.

- **AMPERSAND:** Clase léxica univaluada que denota un ampersand. Es de la forma “&”.

- **TCADENA:** Clase léxica univaluada que denota que el identificador a su derecha es de tipo cadena. Es de la forma “string”.

- **NULL:** Clase léxica univaluada que representa el tipo null, que representa un puntero sin inicializar. Es de la forma “null”.

- **PROC:** Clase léxica univaluada que indica que se va a iniciar la definición de una función. Es de la forma “proc”.

- **IF:** Clase léxica univaluada que indica el comienzo de la parte condicional de un bloque if. Es de la forma “if”.

- **THEN:** Clase léxica univaluada que indica el comienzo de la parte de código ejecutable de un bloque if si se cumple la condición. Es de la forma “then”.

- **ELSE:** Clase léxica univaluada que indica el comienzo de la parte de código ejecutable de un bloque if si no se cumple la condición. Es de la forma “else”

- **ENDIF:** Clase léxica univaluada que indica el final de un bloque if. Es de la forma “endif”.

- **WHILE:** Clase léxica univaluada que indica el comienzo de la parte condicional de un bloque while. Es de la forma “while”.

- **DO:** Clase léxica univaluada que indica el comienzo de la parte de código ejecutable de un bloque while siempre y cuando se cumple la condición. Es de la forma “do”.

- **ENDWHILE:** Clase léxica univaluada que indica el final de un bloque while. Es de la forma “endwhile”.

- **CALL:** Clase léxica univaluada que indica la llamada a una función. Es de la forma “call”.

- **TRECORD:** Clase léxica univaluada que indica que la definición a su derecha es de tipo registro (record). Es de la forma “record”.

- **TARRAY:** Clase léxica univaluada que indica el inicio de la declaración de un array. Es de la forma “array”.

- **OF:** Clase léxica univaluada que indica que el array que se está definiendo es del tipo que hay a su derecha. Es de la forma “of”.

- **TPOINTER:** Clase léxica univaluada que denota la palabra reservada “pointer” que representa el tipo pointer.

- **NEW:** Clase léxica univaluada que denota la palabra reservada “new” con la que se reserva memoria para nuevo/estructura.

- **DELETE:** Clase léxica univaluada que denota la palabra reservada “delete” con la que se libera memoria de un puntero.

- **READ:** Clase léxica univaluada que denota la palabra reservada “read” con la que se pide input por consola.

- **WRITE:** Clase léxica univaluada que denota la palabra reservada “write” con la que se escribe por consola.

- **NL:** Clase léxica univaluada que denota la palabra reservada “nl” que escribe un salto de línea.

- **VAR:** Clase léxica univaluada que denota la palabra reservada “var” que indica la definición de una nueva variable.

- **TYPE:** Clase léxica univaluada que denota la palabra reservada “type” que indica la definición de un tipo nuevo/estructura.

Especificación formal del léxico del lenguaje mediante expresiones regulares

- **Letra ≡** [a - z ; A - Z]
- **Digito ≡** [0 - 9]
- **Dpos ≡** [1-9]
- **true ≡** true
- **false ≡** false
- **IDEN ≡** Letra(Letra+Digito+_)*
- **ENT ≡** [\+|\-]? (Dpos(Digito)*)+0
- **Pdec ≡** ((Digito*Dpos) + 0)
- **Pexp ≡** (e|E)ENT
- **REAL ≡** {ENT}((\.Pdec+Pexp)+(\.Pdec}{Pexp}))
- **TENT:** int
- **TREAL:** real
- **TBOOL:** bool
- **PAP:** \(
- **PCIERRE:** \)
- **IGUAL: \=**
- **MAS:** \+
- **MENOS:** \-
- **POR:** *
- **DIV:** /

- **MENOR:** \<
- **MAYOR:** \>
- **MENORI:** \<\=
- **MAYORI:** \>\=
- **IGIG:** \=\=
- **NOIG:** \!\=
- **OR:** or
- **NOT:** not
- **AND:** and
- **PUNTOCOMA:** ;
- **SEPARADOR:** &&
- **EOF:** ε
- **CADENA:** **“**$\overline{[", \ \backslash b, \ \backslash r, \ |n]}$*”**
- **COMENTARIO:** # $\overline{[\backslash n, \ EOF]}$*
- **MODULO:** \%
- **CAP:** \[
- **CCIERRE:** \]
- **LAP:** \{
- **LCIERRE:** \}
- **PUNTO:** \.
- **COMA:** \,
- **FLECHA:** \->
- **AMPERSAND:** &

- **TCADENA:** string
- **NULL:** null
- **PROC:** proc
- **IF:** if
- **THEN:** then
- **ELSE:** else
- **ENDIF:** endif
- **WHILE:** while
- **DO:** do
- **ENDWHILE:** endwhile
- **CALL:** call
- **RECORD:** record
- **TARRAY:** array
- **OF:** of
- **TPOINTER:** pointer
- **NEW:** new
- **DELETE:** delete
- **READ:** read
- **WRITE:** write
- **NL:** nl
- **VAR:** var
- **TYPE:** type