

Procesadores de lenguajes

Memoria

G15

Verdaguer Velázquez, Miguel

Rubio Pastor, Cesar Carlos

Antuña Rodríguez , Alejandro

Makitu Koudymba , Fumu Grace

Índices

1 Tiny (0).....	2
1.1 Descripción informal.....	2
Definiciones auxiliares:.....	2
Clases léxicas:.....	3
Cadenas ignorables:.....	3
1.2 Especificación formal del léxico del lenguaje.....	3
Clases auxiliares:.....	3
Clases léxicas:.....	3
Cadenas ignorables:.....	4
1.3 Diagrama de transiciones.....	4
2. Tiny.....	5
2.1 Descripción informal.....	5
Definiciones auxiliares:.....	5
Clases léxicas:.....	5
Cadenas ignorables:.....	6
2.2 Especificación formal del léxico del lenguaje.....	7
Definiciones auxiliares:.....	7
Clases léxicas:.....	7
Cadenas ignorables:.....	8

1 Tiny (0)

1.1 Descripción informal

U->Clase Univaluada

M->clase Multivaluada

Definiciones auxiliares:

digPositivo(M) = Un único dígito entre el 1 y el 9
digito(M) = Un único dígito entre el 0 y el 9
parteEntera(M) = Una serie de dígitos sin ceros al principio
parteDecimal(M) = Un cero seguido de una serie de dígitos sin
ceros al final
parteExponencial(M) = Una e seguida de una serie de dígitos como
un literal entero
segParteReal(M) = La segunda parte de un literal real, bien una
parte decimal, bien una parte exponencial o bien
una parte decimal seguida de una exponencial
letra(M) = Una letra entre la a y la z, mayúscula o minúscula

Clases léxicas:

int(U/I) = Palabra reservada para indicar el tipo int
real(U) = Palabra reservada para indicar el tipo real
bool(U) = Palabra reservada para indicar el tipo bool
and(U) = Palabra reservada como operador and
or(U) = Palabra reservada como operador or
not(U) = Palabra reservada como operador not
TRUE(U) = Palabra reservada para el valor de bool verdadero
FALSE(U) = Palabra reservada para el valor de bool falso
suma(U) = Operador suma
resta(U) = Operador resta
mult(U) = Operador multiplicación
div(U) = Operador divis ión
menorque(U) = Operador menor-que
mayorque(U) = Operador mayor-que

```

menorigual(U) = Operador menor o igual
mayorigual(U) = Operador mayor o igual
igual_comp(U) = Operador igualdad
noigual_comp(U) = Operador no igualdad
par_a(U) = Apertura de paréntesis
par_c(U) = Cierre de paréntesis
puntoycoma(U) = Punto y coma, final de instrucción
igual_op(U) = Operador de asignación de valores
llave_a(U) = Apertura de llave
llave_c(U) = Cierre de llave
dobleat(U) = Fin de declaraciones
arroba(U) = Comienzo de la instrucción de evaluación
litEntero(M) = Un literal entero, puede tener un + o - al
principio y sin ceros no significativos
litReal(M) = Un literal real, separando con punto la parte
decimal y entera y sin ceros no significativos
caracter(M) = Un carácter, letra, dígito o subrayado
iden(M) = Una secuencia de caracteres que comienza por una letra
o subrayado

```

Cadenas ignorables:

```

espacio(M) = Cualquier tipo de espaciado
comentario(M) = Un comentario a ignorar. Comienza con ## y
puede contener cualquier carácter excepto
el salto de línea

```

1.2 Especificación formal del léxico del lenguaje

I -> Case insensitive

Clases auxiliares:

```

digPositivo = [1-9]
digito = {digPositivo}|0
parteEntera = ({digPositivo}{digito})*|0
parteDecimal = \.({digito}{digitoPositivo}|0)
parteExponencial = (e|E){litEntero}
segParteReal = {parteDecimal}|{parteExponencial}|{parteDecimal}
{parteExponencial}
letra = [a-z,A-Z]

```

Clases léxicas:

```

int(I) = [I,I][n,N][t,T]
real (I) = [r,R][e,E][a,A][l,L]
bool (I) = [b,B][o,O][o,O][l,L]
and (I) = [a,A][n,N][d,D]
or (I) = [o,O][r,R]
not (I) = [n,N][o,O][t,T]
null (I) = [n,N][u,U][l,L][l,L]
TRUE (I) = [t,T][r,R][u,U][e,E]
FALSE (I) = [f,F][a,A][l,L][s,S][e,E]
suma = \+
resta = \-
mult = \*
div = /

```

[illegible]

2. Tiny

2.1 Descripción informal

Definiciones auxiliares:

digPositivo(M) = Un único dígito entre el 1 y el 9
digito(M) = Un único dígito entre el 0 y el 9
parteEntera(M) = Una serie de dígitos sin ceros al principio
parteDecimal(M) = Un cero seguido de una serie de dígitos sin
ceros al final
parteExponencial(M) = Una e seguida de una serie de dígitos como
un literal entero
segParteReal(M) = La segunda parte de un literal real, bien una
parte
letra(M) = Una letra entre la a y la z, mayúscula o minúscula

Clases léxicas:

int(U) = Palabra reservada para indicar el tipo int
real(U) = Palabra reservada para indicar el tipo real
bool(U) = Palabra reservada para indicar el tipo bool
string(U) = Palabra reservada para indicar el tipo string
and(U) = Palabra reservada como operador and
or(U) = Palabra reservada como operador or
not(U) = Palabra reservada como operador not
null(U) = Palabra reservada para la expresión básica vacía
TRUE(U) = Palabra reservada para el valor de bool verdadero
FALSE(U) = Palabra reservada para el valor de bool falso
proc(U) = Palabra reservada para declarar un procedimiento
if(U) = Palabra reservada para indicar el comienzo de una
instrucción if
else(U) = Palabra reservada para indicar la segunda parte de una
instrucción if-else
while(U) = Palabra reservada para indicar el comienzo de una
instrucción while
struct(U) = Palabra reservada para declarar tipos como
estructuras
new(U) = Palabra reservada para la instrucción de reserva de
memoria
delete(U) = Palabra reservada para la instrucción de liberación
de memoria
read(U) = Palabra reservada para la instrucción de lectura de
memoria
write(U) = Palabra reservada para la instrucción de escritura en
memoria
nl(U) = Palabra reservada para la instrucción de nueva línea
type(U) = Palabra reservada para las declaraciones de tipo
call(U) = Palabra reservada para las llamadas a procedimientos
suma(U) = Operador suma
resta(U) = Operador resta
mult(U) = Operador multiplicación
div(U) = Operador división
porcentaje(U) = Operador módulo entero

menorque(U) = Operador menor-que
 mayorque(U) = Operador mayor-que
 menorigual(U) = Operador menor o igual
 mayorigual(U) = Operador mayor o igual
 igual_comp(U) = Operador igualdad
 noigual_comp(U) = Operador no igualdad
 par_a(U) = Apertura de paréntesis
 par_c(U) = Cierre de paréntesis
 puntoycoma(U) = Punto y coma, final de instrucción
 igual_op(U) = Operador de asignación de valores
 corch_a(U) = Apertura de corchetes
 corch_c(U) = Cierre de corchetes
 punto(U) = Punto, operador de acceso a registro
 capirote(U) = Operador de indirección
 coma(U) = Coma, separador de listas
 llave_a(U) = Apertura de llave
 llave_c(U) = Cierre de llave
 at(U) = Operador de referencia
 dobleat(U) = Fin de declaraciones
 arroba(U) = Comienzo de la instrucción de evaluación
 litEntero(M) = Un literal entero, puede tener un + o - al principio y sin ceros no significativos
 litReal(M) = Un literal real, separando con punto la parte decimal y entera y sin ceros no significativos
 caracter(M) = Un carácter, letra, dígito o subrayado
 iden(M) = Una secuencia de caracteres que comienza por una letra o subrayado
 litCadena(M) = Un literal de tipo cadena, comienza por una doble comilla y continúa con una secuencia de 0 o más caracteres, terminado en otra doble comilla

Cadenas ignorables:

espacio(M) = Cualquier tipo de espaciado
 comentario(M) = Un comentario a ignorar. Comienza con ## y puede contener cualquier carácter excepto el salto de línea

2.2 Especificación formal del léxico del lenguaje

I -> Case insensitive

Definiciones auxiliares:

digPositivo = [1-9]
 digito = {digPositivo}|0
 parteEntera = ({digPositivo}{digito}*|0) letra = [a-z,A-Z]

```

parteDecimal = \.({digito}*{digitoPositivo}|0)
segParteReal = {parteDecimal}|{parteExponencial}|{parteDecimal}
{
    parteExponencial = (e|E){litEntero}
    letra = [a-z,A-Z]
}

```

Clases léxicas:

```

int (I) = [i,I][n,N][t,T]
real (I) = [r,R][e,E][a,A][l,L]
bool (I) = [b,B][o,O][o,O][l,L]
string (I) = [s,S][t,T][r,R][i,I][n,N][g,G]
and (I) = [a,A][n,N][d,D]
or (I) = [o,O][r,R]
not (I) = [n,N][o,O][t,T]
null (I) = [n,N][u,U][l,L][l,L]
TRUE (I) = [t,T][r,R][u,U][e,E]
FALSE (I) = [f,F][a,A][l,L][s,S][e,E]
proc (I) = [p,P][r,R][o,O][c,C]
if (I) = [i,I][f,F]
else (I) = [e,E][l,L][s,S][e,E]
while (I) = [w,W][h,H][i,I][l,L][e,E]
struct (I) = [s,S][t,T][r,R][u,U][c,C][t,T]
new (I) = [n,N][e,E][w,W]
delete (I) = [d,D][e,E][l,L][e,E][t,T][e,E]
read (I) = [r,R][e,E][a,A][d,D]
write (I) = [w,W][r,R][i,I][t,T][e,E]
nl (I) = [n,N][l,L]
type (I) = [t,T][y,Y][p,P][e,E]
call (I) = [c,C][a,A][l,L][l,L]
suma = \+
resta = \-
mult = \*
div = /
porcentaje = %
menorque = <
mayorque = >
menorigual = <=
mayorigual = >=
igual_comp = ==
noigual_comp = !=
par_a = \ (
par_c = \ )
puntoycoma = ;
igual_op = =
corch_a = \ [
corch_c = \ ]
punto = \ .
capirote = \ ^
coma = \ ,
llave_a = \ {
llave_c = \ }
at = \ &

```



```
dobleat = &&
arroba = @

litEntero = ([\+, \-])? {parteEntera}
litReal = {litEntero} {segParteReal}
caracter = {letra} | {digito} | _
iden = ({letra} | _) {caracter} *
litCadena = "([^\0F]) *"
```

Cadenas ignorables:

```
espacio = [\b, \r, \t, \n]
comentario = ##([^\n, \0F]) *
```