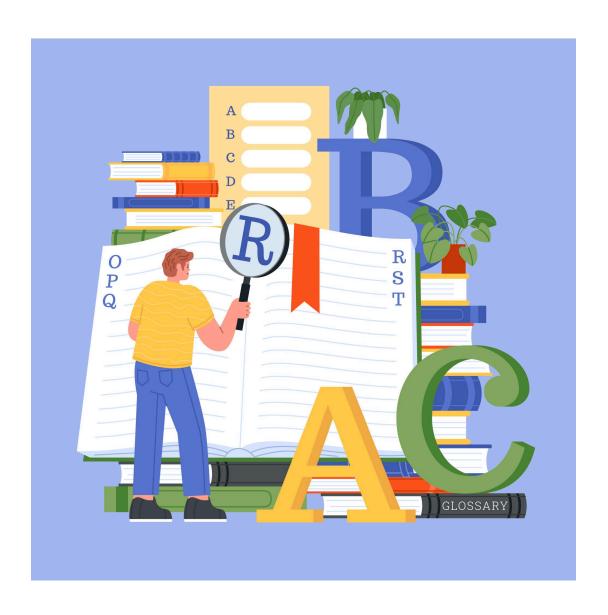
Fase 1: Desarrollo de analizadores léxicos



Javier García Viana
Ibon Malles Altolaguirre
David Peromingo Peromingo
Francisco Prieto Gallego

1. Tiny0

1.1. Clases léxicas

1.1.1. Clases univaluadas

- +: Suma unaria o binaria.
- -: Resta binaria o unaria.
- *: Multiplicación binaria.
- /: División binaria.
- =: Asignación.
- <: Menor binario.
- >: Mayor binario.
- <=: Menor o igual binario.
- >=: Mayor o igual binario.
- !=: Comparación desigual.
- **==:** Comparación igual.
- (: Paréntesis de apertura.
-): Paréntesis de cierre.
- {: Llave de apertura.
- **}:** Llave de cierre.
- ;: Punto y coma.
- **&&:** Terminación de una sección.
- @: Inicio de un nombre.
- int: Declaración de variables enteras.
- real: Declaración de variables reales.
- bool: Declaración de variables booleanas.
- true: Valor booleano verdadero.
- false: Valor booleano falso.

and: 'Y' lógico.

or: 'O' lógico.

not: 'No' lógico.

1.1.2. Clases multivaluadas

Literales enteros: Conjunto de dígitos que representan números enteros. Comienzan, opcionalmente, con un signo + o -. Seguidamente debe aparecer una secuencia de 1 o más dígitos (no se admiten ceros no significativos a la izquierda)

Literales reales: Constan de una parte entera, cuya estructura es como la de los números enteros, seguida de bien una parte decimal, bien una parte exponencial, o bien una parte decimal seguida de una parte exponencial. La parte decimal comienza con un '.', seguido de una secuencia de 1 o más dígitos (no se permite la aparición de ceros no significativos a la derecha). Por último, y también opcionalmente, puede aparecer una parte exponencial (e o E, seguida de un exponente, cuya estructura es igual que la de los números enteros)

Variables/identificadores: Cadenas de caracteres que identifican a una variable. Comienzan necesariamente por una letra o subrayado (_), seguida de una secuencia de cero o más letras, dígitos, o subrayado (_)

1.2. Definiciones regulares

1.2.1. Definiciones auxiliares

 $resta \equiv \backslash -$

```
letra \equiv ([a − z, A − Z] | _)

digitoPositivo \equiv [1 − 9]

digito \equiv {digitoPositivo} | 0

parteEntera \equiv ({digitoPositivo} {digito} * | 0)

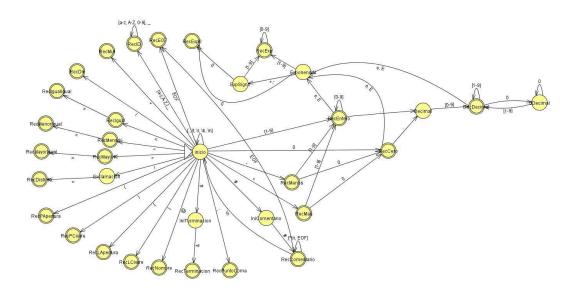
parteDecimal \equiv ({digito} * {digitoPositivo} | 0)
```

```
1.2.2. Definiciones léxicas
identificador \equiv \{letra\} (\{letra\} \mid \{digito\}) *
literalEntero \equiv ( \setminus - | \setminus +)? {parteEntera}
literalReal \equiv \{literalEntero\} \ (\ \{parteDecimal\} \mid ((\ \{parteDecimal\})? \ (e|E)\{literalEntero\}))\}
int \equiv (i|I)(n|N)(t|T)
real \equiv (r|R)(e|E)(a|A)(l|L)
bool \equiv (b|B)(o|O)(o|O)(l|L)
true \equiv (t|T)(r|R)(u|U)(e|E)
false \equiv (f|F)(a|A)(l|L)(s|S)(e|E)
and \equiv (a|A)(n|N)(d|D)
or \equiv (o|0)(r|R)
not \equiv (n|N)(o|O)(t|T)
parentesisApertura \equiv \setminus (
parentesisCierre ≡ \)
llaveApertura ≡ \{
llaveCierre ≡ \}
puntoComa \equiv \;
suma \equiv \ \ +
```

1.2.3. Definiciones de cadenas ignorables:

```
separador \equiv [ , \t, \r, \b, \n]
comentario \equiv \#\#([^\n])^*
```

1.3. Diagrama del analizador léxico



2. Tiny

2.1. Clases léxicas

2.1.1. Clases univaluadas

- +: Suma unaria o binaria.
- -: Resta binaria o unaria.
- *: Multiplicación binaria.
- *I*: División binaria.
- =: Asignación.
- <: Menor binario.
- >: Mayor binario.
- <=: Menor o igual binario.
- >=: Mayor o igual binario.
- !=: Comparación desigual.
- ==: Comparación igual.
- %: Módulo.
- ^: Indirección.
- (: Paréntesis de apertura.
-): Paréntesis de cierre.
- [: Corchete de apertura.
-]: Corchete de cierre.
- **{:** Llave de apertura.
- **}:** Llave de cierre.
- .: Punto.
- ,: Coma.
- ;: Punto y coma.
- &: Referencia.

&&: Terminación de una sección.

@: Inicio de un nombre.

int: Declaración de variables enteras.

real: Declaración de variables reales.

bool: Declaración de variables booleanas.

true: Valor booleano verdadero.

false: Valor booleano falso.

string: Cadena de caracteres.

null: null.

proc: Procedimiento.

if: Evaluación de una condición.

else: Evaluación en caso contrario de que no se cumpla una condición.

while: Bucle while.

struct: Estructura.

new: Reserva de memoria.

delete: Liberación de memoria.

read: Lectura.

write: Escritura.

nl: Nueva línea.

type: Declaración de tipos.

call: Invocación a procedimiento.

and: 'Y' lógico.

or: 'O' lógico.

not: 'No' lógico.

2.1.2. Clases multivaluadas

Literales enteros: Conjunto de dígitos que representan números enteros. Comienzan, opcionalmente, con un signo + o -. Seguidamente debe aparecer una secuencia de 1 o más dígitos (no se admiten ceros no significativos a la izquierda)

Literales reales: Constan de una parte entera, cuya estructura es como la de los números enteros, seguida de bien una parte decimal, bien una parte exponencial, o bien una parte decimal seguida de una parte exponencial. La parte decimal comienza con un '.', seguido de una secuencia de 1 o más dígitos (no se permite la aparición de ceros no significativos a la derecha). Por último, y también opcionalmente, puede aparecer una parte exponencial (e o E, seguida de un exponente, cuya estructura es igual que la de los números enteros)

Literales cadena: Constan de ", seguida de una secuencia de 0 o más caracteres distintos de ", seguida de ". Los caracteres pueden incluir las siguientes secuencias de escape: retroceso (\b), retorno de carro (\r), tabulador (\t), y salto de línea (\n)

Variables/identificadores: Cadenas de caracteres que identifican a una variable. Comienzan necesariamente por una letra o subrayado (_), seguida de una secuencia de cero o más letras, dígitos, o subrayado (_)

2.2. Definiciones regulares

2.2.1 Definiciones auxiliares

```
letra \equiv ([a-z, A-Z] | _)

digitoPositivo \equiv [1-9]

digito \equiv {dígitoPositivo} | 0

parteEntera \equiv ({dígitoPositivo} {dígito} * | 0)

parteDecimal \equiv ({dígito} * {dígitoPositivo} | 0)
```

2.2.2. Definiciones léxicas

while $\equiv (w|W)(h|H)(i|I)(l|L)(e|E)$

```
identificador \equiv \{letra\} (\{letra\} \mid \{digito\}) *
literalEntero \equiv ( \setminus - | \setminus +)? {parteEntera}
literalReal \equiv \{literalEntero\} \ (\ \{parteDecimal\} \mid ((\ \{parteDecimal\})? \ (e|E)\{literalEntero\}))\}
literalCadena ≡ " ([^\"]) * "
int \equiv (i|I)(n|N)(t|T)
real \equiv (r|R)(e|E)(a|A)(l|L)
bool \equiv (b|B)(o|O)(o|O)(l|L)
true \equiv (t|T)(r|R)(u|U)(e|E)
false \equiv (f|F)(a|A)(l|L)(s|S)(e|E)
and \equiv (a|A)(n|N)(d|D)
or \equiv (o|O)(r|R)
not \equiv (n|N)(o|O)(t|T)
string \equiv (s|S)(t|T)(r|R)(i|I)(n|N)(g|G)
null\equiv (n|N)(u|U)(l|L)(l|L)
proc \equiv (p|P)(r|R)(o|O)(c|C)
if \equiv (i|I)(f|F)
else \equiv (e|E)(l|L)(s|S)(e|E)
```

```
struct \equiv (s|S)(t|T)(r|R)(u|U)(c|C)(t|T)
new \equiv (n|N)(e|E)(w|W)
delete \equiv (d|D)(e|E)(l|L)(e|E)(t|T)(e|E)
read \equiv (r|R)(e|E)(a|A)(d|D)
write \equiv (w|W)(r|R)(i|I)(t|T)(e|E)
nl \equiv (n|N)(l|L)
type \equiv (t|T)(y|Y)(p|P)(e|E)
call \equiv (c|C)(a|A)(l|L)(l|L)
parentesisApertura ≡ \(
parentesisCierre ≡ \)
corcheteApertura ≡ \[
corcheteCierre ≡ \]
llaveApertura ≡ \{
llaveCierre ≡ \}
suma \equiv \ \ +
resta \equiv \ \ -
multiplicación \equiv \ \ ^*
división ≡ /
asignación ≡ =
menor \equiv <
menorlgual \equiv <=
mayor \equiv >
mayorlgual ≡ >=
igual ≡ ==
desigual ≡ ! =
módulo ≡ %
indirección ≡ \^
referencia ≡ &
```

```
terminación \equiv &&

nombre \equiv @

punto \equiv \.

coma \equiv \,

puntoComa \equiv \;

2.2.3. Definiciones de cadenas ignorables:

separador \equiv [ ,\t,\r,\b,\n]

comentario \equiv ## ([^\n]) *
```