Procesadores de Lenguajes

Memoria de proyecto - Hito 1: Analizador Léxico

Grupo 10
Sergio Colet García
Laura Martínez Tomás
Rodrigo Souto Santos
Li Jie Chen Chen

Grado en Ingeniería informática
Facultad de Informática
Universidad Complutense de Madrid

2023-2024

Índice general

1 | Tiny(0)

1.1. Introducción

1.2. Clases léxicas

1.2.1. Palabras reservadas

Para poder analizar de manera correcta, será necesario establecer una clase léxica por cada palabra reservada. En el lenguaje de esta práctica, *Tiny(0)*, contamos con 3 palabras reservadas, utilizadas para definir el tipo de las variables. Tendremos pues, una palabra para las variables de tipo booleano, otra para las de tipo entero y una última para las reales. Las palabras son las definidas a continuación, contando cada con una clase léxica.

- *bool* → Variables booleanas.
- *int* → Variables enteras.
- *real* → Variables reales.
- *and* → Conjunción lógica.
- *or* → Disyunción lógica.
- *not* → Negación lógica.

1.2.2. Literales

- **Literales booleanos.** Toma como valor las palabras reservadas *true* o *false*. Su clase léxica será *literalBooleano*.
- **Literales enteros.** Opcionalmente empiezan con un signo más (+) o menos (-), y después debe aparecer una secuencia (que empieza por un número distinto de 0) de 1 o más dígitos. Su clase léxica será *literalEntero*.
- **Literales reales.**

1.2.3. Identificadores

Los identificadores nos sirven para poder ponerle un nombre a las variables. Éstos deben comenzar por un subrayado (_) o una letra, seguida de una secuencia de 0 o más subrayados, dígitos o letras. Su clase léxica será *identificador*.

1.2.4. Símbolos de operación y puntuación

Cada uno de ellos tendrá su propia clase léxica. En el subconjunto del lenguaje en el que trabajamos, *Tiny(0)*, contamos con las siguientes clases:

- **Suma.** Se representa con el símbolo más (+). Su clase léxica será *operadorSuma*.
- **Resta.** Se representa con el símbolo símbolo menos (-). Su clase léxica será *operadorResta*.
- **Multiplicación.** Se representa con el símbolo asterisco (*). Su clase léxica será *operadorMul*.
- **División.** Se representa con el símbolo barra (/). Su clase léxica será *operadorDiv*.
- **Menor.** Se representa con el símbolo menor qué (<). Su clase léxica será *operadorMenor*.
- **Mayor.** Se representa con el símbolo mayor qué (>). Su clase léxica será *operadorMayor*.

- **Igual.** Se representa con el dos símbolos de igualdad seguidos (==). Su clase léxica será *operadorIgual*.
- **Menor o igual.** Se representa con el símbolo menor qué seguido del símbolo de igualdad (<=). Su clase léxica será *operadorMenIgual*.
- **Mayor o igual.** Se representa con el símbolo mayor qué seguido del símbolo de igualdad (>=). Su clase léxica será *operadorMayIgual*.
- **Asignación.** Se representa con el símbolo un símbolo de igualdad (=). Su clase léxica será *operadorAsig*.
- **Paréntesis de apertura.** Se representa con el símbolo del paréntesis de apertura ("(", sin comillas). Su clase léxica será *parentesisAp*.
- **Paréntesis de cierre.** Se representa con el símbolo del paréntesis de cierre (")", sin comillas). Su clase léxica será *parentesisCi*.
- **Punto y coma.** Se representa con el símbolo punto y coma (;). Su clase léxica será *puntoYComa*.
- **Coma.** Se representa con el símbolo coma (,). Su clase léxica será *coma*.

1.3. Especificación formal del léxico

1.3.1. Definiciones auxiliares.

$letra \longrightarrow \mathbf{A}|\mathbf{B}|...|\mathbf{Z}|\mathbf{a}|\mathbf{b}|...|\mathbf{z}$
$digitoPositivo \longrightarrow \mathbf{1}|...|\mathbf{9}$
$digito \longrightarrow digitoPositivo|0$
$parteEntera \longrightarrow digitoPositivodigito*$
$parteDecimal \longrightarrow digito * digitoPositivo$
$parteExponencial \longrightarrow (e|E)[\backslash+|-]parteEntera$

1.3.2. Definiciones de cadenas ignorables.

$separador \longrightarrow \mathbf{SP}|\mathbf{TAB}|\mathbf{NL}$
$comentario \longrightarrow \#\#(\overline{\mathbf{NL}|\mathbf{EOF}})$

1.3.3. Definiciones léxicas.

$bool \longrightarrow \mathbf{bool}$
$int \longrightarrow \mathbf{int}$
$real \longrightarrow \mathbf{real}$
$and \longrightarrow \mathbf{and}$
$or \longrightarrow \mathbf{or}$
$not \longrightarrow \mathbf{not}$
$literalBooleano \longrightarrow \mathbf{true}|\mathbf{false}$
$literalEntero \longrightarrow [\backslash+|-]parteEntera$
$literalReal \longrightarrow [\backslash+|-]parteEntera(.parteDecimal|parteExponencial|.parteDecimalparteExponencial)$
$identificador \longrightarrow (\_|letra)(letra|digito|\_)*$
$operadorSuma \longrightarrow \backslash+$
$operadorResta \longrightarrow -$
$operadorMul \longrightarrow \backslash*$
$operadorDiv \longrightarrow /$
$operadorMenor \longrightarrow <$
$operadorMayor \longrightarrow >$
$operadorIgual \longrightarrow ==$
$operadorMenIgual \longrightarrow <=$
$operadorMayIgual \longrightarrow >=$
$operadorAsig \longrightarrow =$

$parentesisAp \longrightarrow \backslash($
$parentesisCi \longrightarrow \backslash)$
$puntoYComa \longrightarrow ;$
$arroba \longrightarrow @$

1.4. Diseño de un analizador léxico

2 | Tiny

2.1. Introducción

2.2. Clases léxicas

2.2.1. Palabras reservadas

Para poder analizar de manera correcta, será necesario establecer una clase léxica por cada palabra reservada. En el lenguaje de esta práctica, *Tiny(0)*, contamos con 3 palabras reservadas, utilizadas para definir el tipo de las variables. Tendremos pues, una palabra para las variables de tipo booleano, otra para las de tipo entero y una última para las reales. Las palabras son las definidas a continuación, contando cada con una clase léxica.

- *bool* → Variables booleanas.
- *int* → Variables enteras.
- *real* → Variables reales.
- *and* → Conjunción lógica.
- *or* → Disyunción lógica.
- *not* → Negación lógica.

2.2.2. Literales

- **Literales booleanos.** Toma como valor las palabras reservadas *true* o *false*. Su clase léxica será *literalBooleano*.
- **Literales enteros.** Opcionalmente empiezan con un signo más (+) o menos (-), y después debe aparecer una secuencia (que empieza por un número distinto de 0) de 1 o más dígitos. Su clase léxica será *literalEntero*.
- **Literales reales.**

2.2.3. Identificadores

Los identificadores nos sirven para poder ponerle un nombre a las variables. Éstos deben comenzar por un subrayado (_) o una letra, seguida de una secuencia de 0 o más subrayados, dígitos o letras. Su clase léxica será *identificador*.

2.2.4. Símbolos de operación y puntuación

Cada uno de ellos tendrá su propia clase léxica. En el subconjunto del lenguaje en el que trabajamos, *Tiny(0)*, contamos con las siguientes clases:

- **Suma.** Se representa con el símbolo más (+). Su clase léxica será *operadorSuma*.
- **Resta.** Se representa con el símbolo símbolo menos (-). Su clase léxica será *operadorResta*.
- **Multiplicación.** Se representa con el símbolo asterisco (*). Su clase léxica será *operadorMul*.
- **División.** Se representa con el símbolo barra (/). Su clase léxica será *operadorDiv*.
- **Menor.** Se representa con el símbolo menor qué (<). Su clase léxica será *operadorMenor*.
- **Mayor.** Se representa con el símbolo mayor qué (>). Su clase léxica será *operadorMayor*.
- **Igual.** Se representa con el dos símbolos de igualdad seguidos (==). Su clase léxica será *operadorIgual*.

- **Menor o igual.** Se representa con el símbolo menor qué seguido del símbolo de igualdad (<=). Su clase léxica será *operadorMenIgual*.
- **Mayor o igual.** Se representa con el símbolo mayor qué seguido del símbolo de igualdad (>=). Su clase léxica será *operadorMayIgual*.
- **Asignación.** Se representa con el símbolo un símbolo de igualdad (=). Su clase léxica será *operadorAsig*.
- **Paréntesis de apertura.** Se representa con el símbolo del paréntesis de apertura ("(", sin comillas). Su clase léxica será *parentesisAp*.
- **Paréntesis de cierre.** Se representa con el símbolo del paréntesis de cierre (")", sin comillas). Su clase léxica será *parentesisCi*.
- **Punto y coma.** Se representa con el símbolo punto y coma (;). Su clase léxica será *puntoYComa*.
- **Coma.** Se representa con el símbolo coma (,). Su clase léxica será *coma*.

2.3. Especificación formal del léxico

2.3.1. Definiciones auxiliares.

$letra \longrightarrow \mathbf{A}|\mathbf{B}|...|\mathbf{Z}|\mathbf{a}|\mathbf{b}|...|\mathbf{z}$
$digitoPositivo \longrightarrow \mathbf{1}|...|\mathbf{9}$
$digito \longrightarrow digitoPositivo|0$
$parteEntera \longrightarrow digitoPositivodigito*$
$parteDecimal \longrightarrow digito*digitoPositivo$
$parteExponencial \longrightarrow (e|E)[\backslash+|-]parteEntera$

2.3.2. Definiciones de cadenas ignorables.

$separador \longrightarrow \mathbf{SP}|\mathbf{TAB}|\mathbf{NL}$
$comentario \longrightarrow \#\#(\overline{\mathbf{NL}|\mathbf{EOF}})$

2.3.3. Definiciones léxicas.

$bool \longrightarrow \mathbf{bool}$
$int \longrightarrow \mathbf{int}$
$real \longrightarrow \mathbf{real}$
$and \longrightarrow \mathbf{and}$
$or \longrightarrow \mathbf{or}$
$not \longrightarrow \mathbf{not}$
$literalBooleano \longrightarrow \mathbf{true}|\mathbf{false}$
$literalEntero \longrightarrow [\backslash+|-]parteEntera$
$literalReal \longrightarrow [\backslash+|-]parteEntera(.parteDecimal|parteExponencial|.parteDecimalparteExponencial)$
$identificador \longrightarrow (\_|letra)(letra|digito|\_)*$
$operadorSuma \longrightarrow \backslash+$
$operadorResta \longrightarrow -$
$operadorMul \longrightarrow \backslash*$
$operadorDiv \longrightarrow /$
$operadorMenor \longrightarrow <$
$operadorMayor \longrightarrow >$
$operadorIgual \longrightarrow ==$
$operadorMenIgual \longrightarrow <=$
$operadorMayIgual \longrightarrow >=$
$operadorAsig \longrightarrow =$
$parentesisAp \longrightarrow \backslash($
$parentesisCi \longrightarrow \backslash)$

$puntoYComa \longrightarrow$;
$arroba \longrightarrow$ @

Índice de figuras