

Memoria: Procesadores de Lenguaje - Lenguaje Tiny

Burgos Sosa, Rodrigo	Cassin, Gina Andrea
Estebán Velasco, Luis	Rabbia, Santiago Elias

Curso 2024

Grupo G03

1 Introducción

En el siguiente documento se expondrá una memoria sobre el desarrollo de analizadores léxicos aplicado sobre dos lenguajes de programación, Tiny y Tiny(0) - un subconjunto de Tiny. Se presentará una descripción de las clases léxicas y una especificación formal de ambos lenguajes, y un analizador léxico para Tiny(0).

2 Análisis léxico

La función de un analizador léxico es segmentar el programa de entrada en una secuencia de componentes léxicos o tokens. La primera fase en el desarrollo del analizador léxico (y la fase más importante) es llevar a cabo su especificación léxica. Esto se llevará a cabo a continuación:

2.1 Tiny(0)

2.1.1 Clases léxicas

A continuación se presentan las clases léxicas del lenguaje Tiny(0):

Clases léxicas

- **Identificador (variable):** Comienzan necesariamente por una letra o subrayado (`_`), seguida de una secuencia de cero o más letras, dígitos, o subrayado (`_`).
- **Una clase léxica por cada tipo de variable:**
 - **int:** representa los números enteros.
 - **real:** representa los números reales.
 - **bool:** representa los valores booleanos (`true` o `false`).
- **Literal entero**
- **Literal real**
- **Literal booleano**
- **Una clase léxica por cada operador aritmético:**
 - **+**: suma.

- -: resta.
- *: multiplicación.
- /: división.
- **Una clase léxica por cada operador lógico:**
 - **and**: conjunción.
 - **or**: disyunción.
 - **not**: negación.
- **Una clase léxica por cada operador relacional:**
 - <: menor que.
 - >: mayor que.
 - <=: menor o igual que.
 - >=: mayor o igual que.
 - ==: igual que.
 - !=: distinto que.
- **Una clase léxica por cada símbolo de puntuación:**
 - (: paréntesis izquierdo. Sirve para asociatividad.
 -): paréntesis derecho. Sirve para asociatividad.
 - ;; punto y coma. Sirve para separar declaraciones en la sección de declaraciones, o separar instrucciones en la sección de instrucciones.
 - .: punto. Para los decimales.
 - {: llave izquierda. Indica el inicio de un bloque de código.
 - }: llave derecha. Indica el fin de un bloque de código.
 - &&: doble signo et. Indica el fin de declaraciones.
- **Operador de asignación:** =
- **Operador de evaluación:** @

Cadenas ignorables

- Espacios en blanco.
- Retroceso: `\b`
- Tabulador: `\t`
- Retorno de carro: `\r`
- Salto de línea: `\n`

2.1.2 Especificación formal

Definiciones auxiliares

- `letra` \equiv `[a-z,A-Z]`
- `digito` \equiv `[0-9]`
- `digitoSinCero` \equiv `[1-9]`
- `parteEntera` \equiv `(\{digitoSinCero\} \{digito\}*) | 0`
- `parteDecimal` \equiv `(\{digito\}* \{digitoSinCero\}) | 0`

Definiciones léxicas

- `suma` \equiv `\+`
- `resta` \equiv `\-`
- `mul` \equiv `*`
- `div` \equiv `/`
- `parentesisAbrir` \equiv `\(`
- `parentesisCerrar` \equiv `\)`
- `abrirBloque` \equiv `\{`
- `cerrarBloque` \equiv `\}`
- `separadorDeclaraciones` \equiv `;`
- `finDeclaraciones` \equiv `&&`

- **asignacion** $\equiv \backslash =$
- **menor** $\equiv <$
- **mayor** $\equiv >$
- **menorIgual** $\equiv < \backslash =$
- **mayorIgual** $\equiv > \backslash =$
- **igual** $\equiv \backslash = \backslash =$
- **no igual** $\equiv ! \backslash =$
- **and** $\equiv \text{and}$
- **or** $\equiv \text{or}$
- **not** $\equiv \text{not}$
- **true** $\equiv \text{true}$
- **false** $\equiv \text{false}$
- **tipo entero** $\equiv \text{int}$
- **tipo real** $\equiv \text{real}$
- **tipo booleano** $\equiv \text{bool}$
- **eval** $\equiv @$
- **punto** $\equiv \backslash .$
- **identificador** $\equiv (\{\text{letra}\} \mid _) (\{\text{letra}\} \mid \{\text{dígito}\} \mid _)^*$
- **literalEntero** $\equiv (\backslash + \mid \backslash -)? \{\text{parteEntera}\}$
- **literalReal** $\equiv \{\text{literalEntero}\} ((\backslash . \{\text{parteDecimal}\}) ((\text{e} \mid \text{E}) \{\text{literalEntero}\}))$
 $\mid (\backslash . \{\text{parteDecimal}\}) \mid ((\text{e} \mid \text{E}) \{\text{literalEntero}\})$

Definiciones cadenas ignorables

- **separador** $\equiv [\ , \backslash \text{t}, \backslash \text{r}, \backslash \text{b}, \backslash \text{n}]$

2.1.3 Diagrama de transiciones



2.2 Tiny

2.2.1 Clases léxicas

A continuación se presentan las clases léxicas del lenguaje Tiny:

Clases léxicas

- **Identificador (variable):** Comienzan necesariamente por una letra o subrayado (`_`), seguida de una secuencia de cero o más letras, dígitos, o subrayado (`_`).
- **Una clase léxica por cada tipo de variable:**
 - **int:** representa los números enteros.
 - **real:** representa los números reales.
 - **bool:** representa los valores booleanos (`true` o `false`).
 - **string:** representa las cadenas de caracteres.
 - **array:** representa los arreglos.
- **Literal entero**
- **Literal real**
- **Literal booleano**
- **Literal cadena**
- **Una clase léxica por cada operador aritmético:**
 - **+**: suma.
 - **-**: resta.
 - *****: multiplicación.
 - **/**: división.
- **Una clase léxica por cada operador lógico:**
 - **and:** conjunción.
 - **or:** disyunción.
 - **not:** negación.
- **Una clase léxica por cada operador relacional:**

- <: menor que.
- >: mayor que.
- <=: menor o igual que.
- >=: mayor o igual que.
- ==: igual que.
- !=: distinto que.

- **Una clase léxica por cada símbolo de puntuación:**

- (: paréntesis izquierdo.
-): paréntesis derecho.
- ;: punto y coma.
- ,: coma.
- .: punto.
- { : llave izquierda.
- } : llave derecha.
- &: signo et simple.
- &&: doble signo et.
- [: corchete izquierdo.
-]: corchete derecho.
- %: porcentaje.
- ^: acento circunflejo.

- **Operador de asignación: =**

- **Operador de evaluación: @**

- **Una clase léxica por cada palabra reservada:**

- **null**: representa el valor nulo.
- **proc**: palabra reservada para definir un procedimiento.
- **if**: palabra reservada para definir una condición.
- **else**: palabra reservada para definir una condición alternativa.
- **while**: palabra reservada para definir un bucle.
- **struct**: palabra reservada para definir una estructura.

- **new**: palabra reservada para instrucción de reserva de memoria.
- **delete**: palabra reservada para instrucción de liberación de memoria.
- **read**: palabra reservada para instrucción de lectura.
- **write**: palabra reservada para instrucción de escritura.
- **nl**: palabra reservada para instrucción de nueva línea.
- **type**: palabra reservada para declaración de tipo.
- **call**: palabra reservada para instrucción de invocación a procedimiento.

Cadenas ignorables

- **Espacios en blanco.**
- **Retroceso:** \b
- **Tabulador:** \t
- **Retorno de carro:** \r
- **Salto de línea:** \n
- **Comentarios:** comienzan con ## y terminan con un salto de línea.

2.2.2 Especificación formal

Definiciones auxiliares

- **letra** \equiv [a-z,A-Z]
- **digito** \equiv [0-9]
- **digitoSinCero** \equiv [1-9]
- **parteEntera** \equiv ({digitoSinCero} {digito}*) | 0
- **parteDecimal** \equiv ({digito}* {digitoSinCero}) | 0

Definiciones léxicas

- `suma` \equiv `\+`
- `resta` \equiv `\-`
- `mul` \equiv `*`
- `div` \equiv `/`
- `parentesisAbrir` \equiv `\(`
- `parentesisCerrar` \equiv `\)`
- `abrirBloque` \equiv `\{`
- `cerrarBloque` \equiv `\}`
- `tamañoAbrir` \equiv `[`
- `tamañoCerrar` \equiv `]`
- `finDeclaraciones` \equiv `&&`
- `asignacion` \equiv `\=`
- `menor` \equiv `<`
- `mayor` \equiv `>`
- `menorIgual` \equiv `<\=`
- `mayorIgual` \equiv `>\=`
- `igual` \equiv `\=\=`
- `no igual` \equiv `!\=`
- `and` \equiv `and`
- `or` \equiv `or`
- `not` \equiv `not`
- `true` \equiv `true`
- `false` \equiv `false`

- **modulo** \equiv %
- **puntero** \equiv ^
- **bitwiseAnd** \equiv &
- **tipo entero** \equiv int
- **tipo real** \equiv real
- **tipo booleano** \equiv bool
- **tipo string** \equiv string
- **tipo array** \equiv array
- **creacionTipo** \equiv type
- **null** \equiv null
- **procedimiento** \equiv proc
- **if** \equiv if
- **else** \equiv else
- **while** \equiv while
- **estructura** \equiv struct
- **new** \equiv new
- **delete** \equiv delete
- **read** \equiv read
- **write** \equiv write
- **nl** \equiv nl
- **call** \equiv call
- **eval** \equiv @
- **punto** \equiv \.
- **identificador** \equiv ({letra} | _) ({letra} | {dígito} | _)*
- **literalEntero** \equiv (\+ | \-)? {parteEntera}
- **literalReal** \equiv {literalEntero}((\.{parteDecimal})((e | E){literalEntero}))
| (\.{parteDecimal}) | ((e | E){literalEntero})

Definiciones cadenas ignorables

- **separador** $\equiv [, \backslash t, \backslash r, \backslash b, \backslash n]$
- **comentario** $\equiv \#\#([\backslash n, \text{EOF}])^*$