Fase 1: Desarrollo de analizador léxico para Tiny(0) y Tiny

G01
Esther Babon Arcauz
Pablo Campo Gómez
Claudia López-Mingo Moreno
José Antonio Ruiz Heredia

Índice

Implementación manual: la implementación de las palabras reservadas no es adecuada (deben tratarse como casos de identificadores, teniendo en cuenta que son case insensitive).

1. Enumeración de las clases léxicas de Tiny(0):

Aunque no exista una receta universal para identificar las clases léxicas, vamos a seguir algunas guías:

- Cada símbolo de puntuación y cada operador da lugar a una clase léxica univaluada separada.
- Cada palabra reservada da lugar a una clase léxica univaluada separada.
- Cada tipo literal a una clase léxica multivaluada.
- Lo mismo pasa con los identificadores: clase léxica multivaluada identificador.

En Tiny(0) encontramos las siguientes clases léxicas:

- **Variables**: Palabras reservadas que tienen asociadas un valor específico. Comienzan con una letra o un guión bajo, seguido de letras, dígitos o más guiones bajos.
- **Literales enteros**: Número incluido en el conjunto de los números enteros. Pueden incluir un signo opcional (+ o -) seguido de una secuencia de uno o más dígitos, sin ceros no significativos al principio.
- **Literales reales**: Número incluido en el conjunto de los números reales. Pueden tener las siguientes partes:
    - Parte entera. Sigue el formato de los literales enteros.
    - Parte decimal: Comienza con un punto seguido de uno o más dígitos.
    - Parte exponencial: Comienza con ‘e’ o ‘E’ seguido de uno o más dígitos.
- **Literales booleanos**: Representación de un valor de tipo booleano, es decir que se incluye dentro del conjunto de los valores ‘true’ y ‘false’.
- **Operadores**: Símbolos y signos que se utilizan para realizar las siguientes operaciones:
    - Operadores aritméticos: suma, resta, multiplicación, división.
    - Operadores lógicos: and, not, or.
    - Operadores relacionales: mayor, menor, mayor igual, menor igual, igual, desigual.
    - Operador de asignación.
- **Símbolos de puntuación**: Elementos gramaticales utilizados para estructurar y organizar el código.
    - Paréntesis de apertura y paréntesis de cierre.
- **Palabras reservadas**: Palabras especiales que tienen un significado específico en el lenguaje y no pueden ser usadas como identificadores.
    - Tipos de datos: int, real, bool.
    - Valores: true, false.
    - Operadores: and, not, or.

2. Especificación formal del léxico del lenguaje mediante definiciones regulares:

- **Alfabetos**
    - letra ≡ [a-z, A-Z, _]

- digitoPositivo ≡ [1-9]
- digito ≡ {digitoPositivo} | 0
- parteEntera ≡ ({digitoPositivo} {digito}* | 0)
- parteDecimal ≡ \. ({digito}*{digitoPositivo} | 0)
- parteExponencial ≡ [\e, \E] [\+ , \-]? ({digitoPositivo} {digito}* | 0)

- **Palabras reservadas**
  - int ≡ (i|I)(n|N)(t|T)
  - real ≡ (r|R)(e|E)(a|A)(l|L)
  - bool ≡ (b|B)(o|O)(o|O)(l|L)
  - true ≡ (t|T)(r|R)(u|U)(e|E)
  - false ≡ (f|F)(a|A)(l|L)(s|S)(e|E)
  - and ≡ (a|A)(n|N)(d|D)
  - not ≡ (n|N)(o|O)(t|T)
  - or ≡ (o|O)(r|R)

- **Literales**
  - literalEntero ≡ [\+ , \-]? {parteEntera}
  - literalReal ≡ {literalEntero} ({parteDecimal}|{parteExponencial}|({parteDecimal}{parteExponencial}))
  - variable ≡ {letra} ({letra} | {digito})*

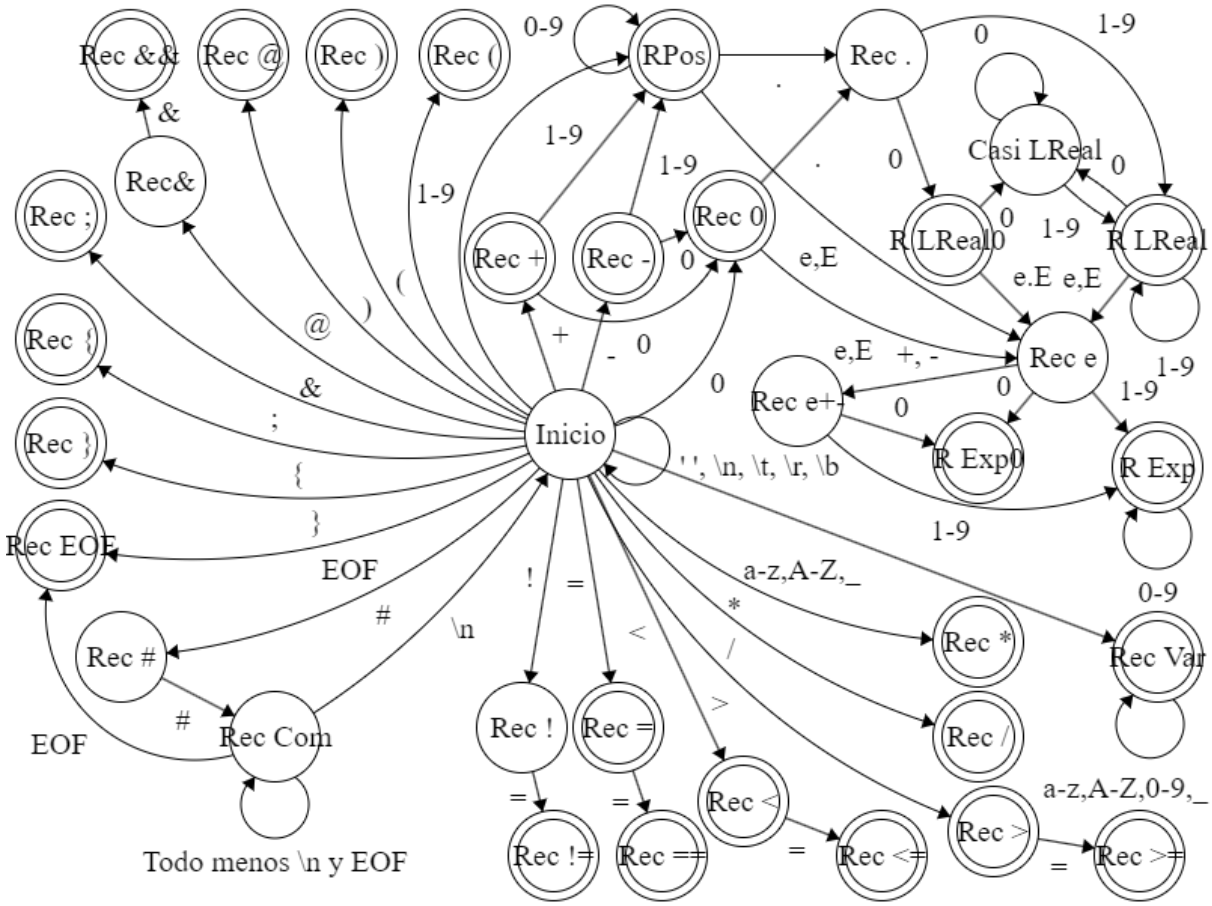
- **Operadores**
  - suma ≡ \+
  - resta ≡ \-
  - mul ≡ *
  - div ≡ /
  - mayor ≡ \>
  - menor ≡ \<
  - mayorIgual ≡ \>=
  - menorIgual ≡ \<=
  - igual ≡ ==
  - desigual ≡ !=
  - asig ≡ =

- **Símbolos de puntuación**
  - parA ≡ \(
  - parC ≡ \)
  - arroba ≡ \@
  - ampersand2 ≡ \&&
  - puntoYComa ≡ \;
  - llaveA ≡ \{

- llaveC ≡ \}
-

- **Cadenas ignorables**
  separador ≡ [,\t, \r, \b, \n]
  comentario ≡ ##([^\n,EOF])*

3. Diseño de un analizador léxico para el lenguaje mediante un diagrama de transiciones:



4. Enumeración de las clases léxicas de Tiny:

Aunque no exista una receta universal para identificar las clases léxicas, vamos a seguir algunas guías:

- Cada símbolo de puntuación y cada operador da lugar a una clase léxica univaluada separada.
- Cada palabra reservada da lugar a una clase léxica univaluada separada.
- Cada tipo literal a una clase léxica multivaluada.
- Lo mismo pasa con los identificadores: clase léxica multivaluada identificador.

En Tiny encontramos las siguientes clases léxicas:

- **Variables**: Palabras reservadas que tienen asociadas un valor específico. Comienzan con una letra o un guión bajo, seguido de letras, dígitos o más guiones bajos.
- **Literales enteros**: Número incluido en el conjunto de los números enteros. Pueden incluir un signo opcional (+ o -) seguido de una secuencia de uno o más dígitos, sin ceros no significativos al principio.
- **Literales reales**: Número incluido en el conjunto de los números reales. Pueden tener las siguientes partes:
    - Parte entera. Sigue el formato de los literales enteros.
    - Parte decimal: Comienza con un punto seguido de uno o más dígitos.
    - Parte exponencial: Comienza con ‘e’ o ‘E’ seguido de uno o más dígitos.
- **Literales cadena:** Representa un valor de tipo string, comienza con comillas dobles (“) seguidas por la cadena vacía o una cadena que contiene cualquier caracter que no sean comillas y termina con comillas dobles.
- **Operadores**: Símbolos y signos que se utilizan para realizar las siguientes operaciones:
    - Operadores aritméticos: suma, resta, multiplicación, división, módulo.
    - Operadores lógicos: and, not, or.
    - Operadores relacionales: mayor, menor, mayor igual, menor igual, igual, desigual.
    - Operador de asignación.
- **Símbolos de puntuación**: Elementos gramaticales utilizados para estructurar y organizar el código.
    - Paréntesis de apertura y cierre, punto y coma, corchete de apertura y cierre, llave de apertura y cierre, punto, & , &&, arroba
- **Palabras reservadas**: Palabras especiales que tienen un significado específico en el lenguaje y no pueden ser usadas como identificadores.
    - Tipos de datos: int, real, bool, string, struct.
    - Valores: true, false, null.
    - Operadores: and, not, or.
    - Condicionales: if, else, while
    - Instrucciones: proc, new, delete, read, write, nl, type, call

5. Especificación formal del léxico del lenguaje mediante definiciones regulares:

- **Alfabetos**
  - letra ≡ [a-z, A-Z, _]
  - digitoPositivo ≡ [1-9]
  - digito ≡ {digitoPositivo} | 0
  - parteEntera ≡ ({digitoPositivo} {digito}* | 0)
  - parteDecimal ≡ \. ({digito}*{digitoPositivo} | 0)
  - parteExponencial ≡ [\e, \E] [\+ , \-]? ({digitoPositivo} {digito}* | 0)

- **Palabras reservadas**
  - int ≡ (i|I)(n|N)(t|T)
  - real ≡ (r|R)(e|E)(a|A)(l|L)
  - bool ≡ (b|B)(o|O)(o|O)(l|L)
  - string ≡ (s|S)(t|T)(r|R)(i|I)(n|N)(g|G)
  - null ≡ (n|N)(u|U)(l|L)(l|L)
  - proc ≡ (p|P)(r|R)(o|O)(c|C)
  - if ≡ (i|I)(f|F)
  - else ≡ (e|E)(l|L)(s|S)(e|E)
  - while ≡ (w|W)(h|H)(i|I)(l|L)(e|E)
  - struct ≡ (s|S)(t|T)(r|R)(u|U)(c|C)(t|T)
  - new ≡ (n|N)(e|E)(w|W)
  - delete ≡ (d|D)(e|E)(l|L)(e|E)(t|T)(e|E)
  - read ≡ (r|R)(e|E)(a|A)(d|D)
  - write ≡ (w|W)(r|R)(i|I)(t|T)(e|E)
  - nl ≡ (n|N)(l|L)
  - type ≡ (t|T)(y|Y)(p|P)(e|E)
  - call ≡ (c|C)(a|A)(l|L)(l|L)
  - true ≡ (t|T)(r|R)(u|U)(e|E)
  - false ≡ (f|F)(a|A)(l|L)(s|S)(e|E)
  - and ≡ (a|A)(n|N)(d|D)
  - not ≡ (n|N)(o|O)(t|T)
  - or ≡ (o|O)(r|R)

- **Literales**
  - identificador ≡ {letra}({letra}|{digito})*
  - literalEntero ≡ [\+ , \-]? {parteEntera}
  - literalReal ≡ {literalEntero} ({parteDecimal}|{parteExponencial}|({parteDecimal}{parteExponencial}))
  - literalCadena≡ ”([^”])*”

- **Operadores**
    - suma ≡ \+
    - resta ≡ \-
    - mul ≡ *
    - div ≡ \/
    - mod ≡ \%
    - mayor ≡ \>
    - menor ≡ \<
    - mayorIgual ≡ \>=
    - menorIgual ≡ \<=
    - igual ≡ \==
    - desigual \≡ !=
    - asig ≡ \=
    - parA ≡ \(
    - parC ≡ \)
    - puntoYComa ≡ \;
    - punto ≡ \.
    - corcheteA ≡ \[
    - corcheteC ≡ \]
    - llaveA ≡ \{
    - llaveC ≡ \}
    - arroba ≡ \@
    - ampersand≡ \&
    - ampersand2≡ \&&

- **Cadenas ignorables**

  separador ≡ [,\t, \r, \b, \n]
  comentario ≡ ## [^\n]*