

**Universidad Nacional Autónoma de
México.**

**Facultad de Estudios Superiores
"Aragón"**

Ingeniería en Computación

Compiladores

**T1. Analisis Léxico
Automata Comentarios y tokens**

Lara Martínez Christian Gael

2608

25 de Febrero del 2025

1. Explicación del Programa y código fuente.

```
tokens.py x prog1.c
1 # Christian Gael Lara Martinez Tarea1, 260225
  Edit | Explain | Test | Document | Fix
  1 usage  Christian Lara
2 def get_etiqueta(t):
3     operadores = '+-*/='
4     minusculas = 'abcdefghijklmnopqrstuvwxyz_'
5     mayusculas = minusculas.upper()
6     if t in especiales:
7         if t in operadores:
8             return 'operador'
9         else:
10            return 'simb esp'
11    else:
12        if t[0] in minusculas or t[0] in mayusculas:
13            return 'ID'
14        elif t[0] in '0123456789':
15            return 'entero'
16    return None
```

Toda esta parte del código clasifica los tokens en diferentes categorías:

- Operador si el token está en '+-*/='
- Símbolo especial si está en caracteres especiales '[{}]'`"#\$%&/()=?¡¿' etc
- Identificadores si el primer carácter del token va de la a-z, A-Z y _
- Números enteros si el primer carácter del token es un dígito de 0-9

Y retorna None si el token no se encuentra en ninguna categoría.

```
19 arch1 = open('prueba.c', 'r')
20 texto = arch1.read()
21 arch1.close()
22 print(texto)
```

En esta parte del código abre el archivo que creamos en lenguaje C además que tiene la propiedad de lectura, lo cierra y lo imprime.

```
24 estado = 'Z'
25 texto2 = ''
26 for letra in texto:
27     if estado == 'Z':
28         if letra == '/':
29             estado = 'A'
30         else:
31             texto2 += letra
32     elif estado == 'A':
33         if letra == '*':
34             estado = 'B'
35         else:
36             estado = 'Z'
37             texto2 += '/'
38     elif estado == 'B':
39         if letra == '*':
40             estado = 'C'
41     elif estado == 'C':
42         if letra == '/':
43             estado = 'Z'
44         elif letra != '*':
45             estado = 'B'
46 print('terminado')
```

En este bloque detectar y eliminar comentarios de múltiples líneas:

1. Estado Z (fuera de comentario)

- Si encuentra /, pasa a A.
- Si no, agrega la letra a texto2 (texto sin comentarios).

2. Estado A (inicio de comentario)

- Si encuentra *, cambia a B (dentro de comentario).
- Si no, regresa a Z y agrega / a texto2.

3. Estado B (dentro de comentario)

- Si encuentra *, pasa a C (posible fin de comentario).

4. Estado C (fin de comentario)

- Si encuentra /, vuelve a Z (comentario finalizado).
- Si no es *, regresa a B (sigue en comentario).

```
52 # aqui empieza lo de los tokens
53 separadores = [' ', '\t', '\n']
54 especiales = '{},;!"#$%&/()=?`~*[]-`^'
55 tokens = []
56 dentro = False
57 token = ''
58 for letra in texto2:
59     if not(dentro):
60         if letra in especiales:
61             tokens.append(letra)
62         elif not(letra in separadores):
63             dentro = True
64             token += letra
65     else:
66         if letra in separadores:
67             tokens.append(token)
68             token = ''
69             dentro = False
70         elif letra in especiales:
71             tokens.append(token)
72             tokens.append(letra)
73             token = ''
74             dentro = False
75         else:
76             token += letra
77
78 for t in tokens:
79     print(t)
```

Este código separa el texto en **tokens**:

1. **Si el carácter es un símbolo especial** -> Lo agrega directamente a tokens.
2. **Si no es un separador (, \t, \n)** -> Inicia la captura de un nuevo token (dentro = True).
3. **Si encuentra un separador mientras forma un token** -> Guarda el token en tokens y reinicia la variable token.
4. **Si encuentra un símbolo especial mientras forma un token** -> Guarda el token y el símbolo en tokens.

Al final se imprimen los tokens.

2. Completado de la funcion `get_etiqueta()`, para que reconozca ID, palabras reservadas y tipos de datos.

```
1 usage
2 def es_ID(token):
3     minusculas = 'abcdefghijklmnopqrstuvwxyz'
4     mayusculas = minusculas.upper()
5     if token and (token[0] in minusculas or token[0] in mayusculas):
6         return True
7     return False
8
9 1 usage
10 def es_palabra_reservada(token):
11     reservadas = ['main', 'void', 'int', 'float', 'char', 'for', 'if']
12     return token in reservadas
13
14 1 usage
15 def es_tipo(token):
16     tipos = ['int', 'float', 'char']
17     return token in tipos
18
19 1 usage
20 def es_operador(token):
21     operadores = ['+', '-', '*', '/', '=', '==', '!=', '<=', '>=']
22     return token in operadores
23
24 1 usage
25 def es_simbolo_especial(token):
26     simbolos = '!"#$%&/( )=?¡¿*+{ }[-_ : ; , . '
27     return token in simbolos
```

es_ID(token)

- Comprueba si un token es un identificador (ID), es decir, si comienza con una letra (mayúscula o minúscula).
- Un identificador podría ser el nombre de una variable o función.
- Si el token está vacío, devuelve False.

es_palabra_reservada(token)

- Verifica si el token es una palabra reservada en C (main, void, int, float, char, for, if).

es_tipo(token)

- Identifica si el token es un tipo de dato (int, float, char).

es_operador(token)

- Verifica si el token es un operador matemático o lógico (+, -, *, /, =, ==, !=, <=, >=).

es_simbolo_especial(token)

- Comprueba si el token es un símbolo especial como {}, [], ::.

```
25  def get_etiqueta(token):
26      if es_ID(token):
27          if es_palabra_reservada(token):
28              if es_tipo(token):
29                  return 'tipo'
30              return 'palres'
31          return 'ID'
32      elif es_simbolo_especial(token):
33          if es_operador(token):
34              return 'op'
35          return 'simb_esp'
36      elif token.isdigit():
37          return 'entero'
38      return 'desconocido'
```

get_etiqueta(token)

- Llama a las funciones anteriores para determinar la categoría del token:
 - **Si es un identificador (ID):**
 - Si es una palabra reservada, se clasifica como palres.
 - Si además es un tipo de dato, se clasifica como tipo.
 - De lo contrario, se etiqueta como ID.
 - **Si es un símbolo especial:**
 - Si también es un operador, se clasifica como op.
 - Si no, se clasifica como simb_esp.
 - **Si es un número entero, se clasifica como entero.**
 - **Si no encaja en ninguna categoría, se clasifica como desconocido.**

```
44 texto2 = ''
45 estado = 'Z'
46 for c in texto:
47     if estado == 'Z':
48         if c == '/':
49             estado = 'A'
50         else:
51             texto2 += c
52     elif estado == 'A':
53         if c == '*':
54             estado = 'B'
55         else:
56             estado = 'Z'
57             texto2 += '/' + c
58     elif estado == 'B':
59         if c == '*':
60             estado = 'C'
61     elif estado == 'C':
62         if c == '/':
63             estado = 'Z'
64         elif c != '*':
65             estado = 'B'
```

Eliminación de comentarios (/* ... */)

Parte del código explicado anteriormente en la parte 1 de la tarea.

Por medio de una máquina de estados.

```
69 tokens = []
70 token = ''
71 estado = 'fuera'
72
73 for c in texto2:
74     if estado == 'fuera':
75         if c in '+-*/=!<>[]{};,,:.':
76             tokens.append(c)
77         elif not(c in [' ', '\n', '\t']):
78             estado = 'dentro'
79             token = c
80     else:
81         if c in [' ', '\n', '\t']:
82             tokens.append(token)
83             token = ''
84             estado = 'fuera'
85         elif c in '+-*/=!<>[]{};,,:.':
86             tokens.append(token)
87             tokens.append(c)
88             token = ''
89             estado = 'fuera'
90         else:
91             token += c
```

El código se divide en tokens utilizando otra **máquina de estados**:

- **estado = 'fuera'** → No está dentro de un token.
- Si encuentra un operador o símbolo especial (+-*/=!<>[]{};,,:), lo almacena directamente.
- Si encuentra un espacio, tabulación o salto de línea, termina el token actual.
- Si encuentra un carácter válido, lo agrega al token actual.
- Si el último token no se ha agregado (caso final), se añade a la lista de tokens.

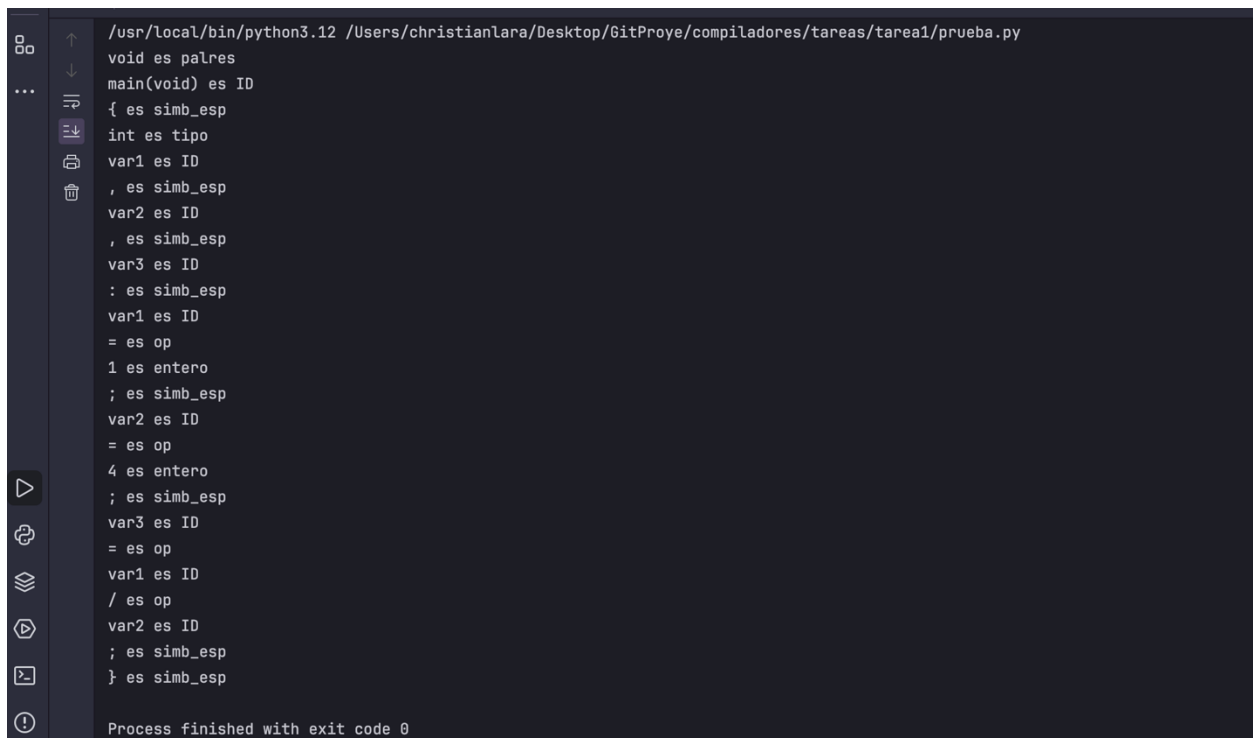

```

93     if token:
94         tokens.append(token)
95
96     for t in tokens:
97         etiqueta = get_etiqueta(t)
98         print(f'{t} es {etiqueta}')

```

se recorre la lista de tokens y se les asigna una categoría usando `get_etiqueta()`, luego se imprimen en formato

Salida:



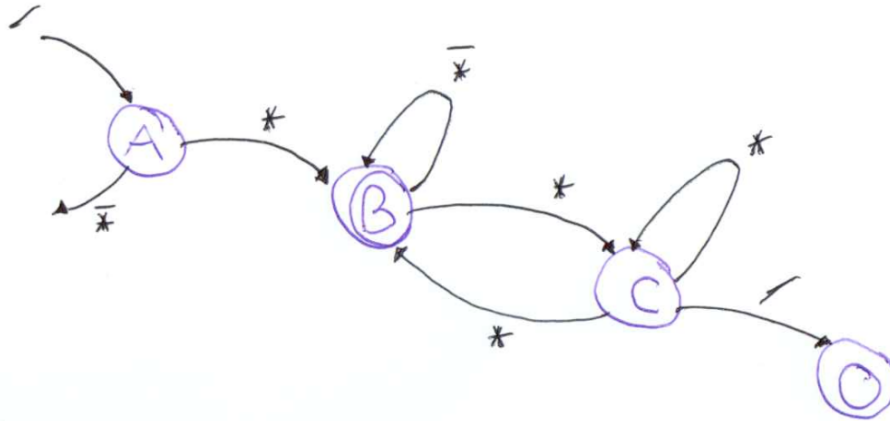
```

/usr/local/bin/python3.12 /Users/christianlara/Desktop/GitProye/compiladores/tareas/tarea1/prueba.py
void es palres
main(void) es ID
{ es simb_esp
int es tipo
var1 es ID
, es simb_esp
var2 es ID
, es simb_esp
var3 es ID
: es simb_esp
var1 es ID
= es op
1 es entero
; es simb_esp
var2 es ID
= es op
4 es entero
; es simb_esp
var3 es ID
= es op
var1 es ID
/ es op
var2 es ID
; es simb_esp
} es simb_esp

Process finished with exit code 0

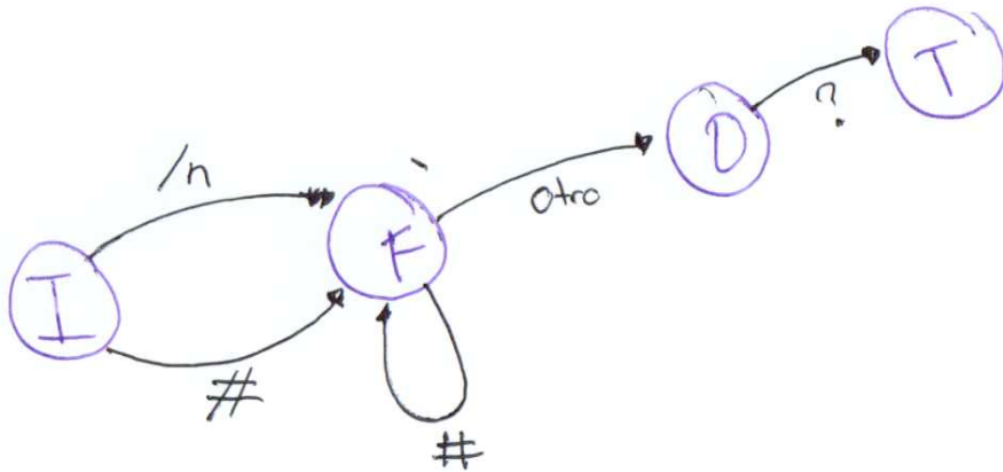
```

Automata Comentarios:



```
24 estado = 'Z'
25 texto2 = ''
26 for letra in texto:
27     if estado == 'Z':
28         if letra == '/':
29             estado = 'A'
30         else:
31             texto2 += letra
32     elif estado == 'A':
33         if letra == '*':
34             estado = 'B'
35         else:
36             estado = 'Z'
37             texto2 += '/'
38     elif estado == 'B':
39         if letra == '*':
40             estado = 'C'
41     elif estado == 'C':
42         if letra == '/':
43             estado = 'Z'
44         elif letra != '*':
45             estado = 'B'
46 print('terminado')
```

Automata Tokens:



```
73 for c in texto2:
74     if estado == 'fuera':
75         if c in '+-*/=<>[]{};,:. ':
76             tokens.append(c)
77         elif not(c in [' ', '\n', '\t']):
78             estado = 'dentro'
79             token = c
80     else:
81         if c in [' ', '\n', '\t']:
82             tokens.append(token)
83             token = ''
84             estado = 'fuera'
85         elif c in '+-*/=<>[]{};,:. ':
86             tokens.append(token)
87             tokens.append(c)
88             token = ''
89             estado = 'fuera'
90         else:
91             token += c
92
93 if token:
94     tokens.append(token)
```