Tecnológico Nacional de México

INSTITUTO TECNOLÓGICO DE PACHUCA

Proyecto: Analizador Léxico

INGENIERÍA EN SISTEMAS COMPUTACIONALES

NOMBRE DE LA ASIGNATURA

Lenguajes y autómatas I

Docente:

Rodolfo Baume Lazcano

Alumnos:

Vera Rodriguez Hector Gabriel

García Lugo Itzel Paola

Diego Quintos Cabeza

Fecha: 20/05/2024

Definición del lenguaje:

- Palabras clave (if, else, while)

- Identificadores (nombres de variables y funciones)
- Operadores (+,-, *,/)
- Números (constantes numéricas)
- Símbolos ((,), {, })
- Espacios en blanco y comentarios (que generalmente se ignoran)

Especificación de tokens:

- Palabras clave:
  if
  else
  while
- Identificadores:
  [a-zA-Z_][a-zA-Z0-9_]*
- Operadores:
  \+, -, *, /
- Números:
  \d+
- Símbolos:
  \(, \), {, },
- Espacios en blanco y comentarios: (que generalmente se ignoran)

Expresiones regulares:

- Identificadores: [a-zA-Z_][a-zA-Z0-9_]*
- Números: \d+
- Operadores: \+, -, *, /
- Símbolos: \(, \), {, }

Tabla de tokens

Token	Clasificación
if	Palabra clave
else	Palabra clave
while	Palabra clave
+	Operador
-	Operador
*	Operador
/	Símbolos
(	Símbolos
)	Símbolos

{	Símbolos
}	Símbolos
0-9	Números
	Espacio en blanco

1.- Palabras clave:
Las palabras clave nos indican tokens que serán reservados para realizar operaciones específicas

- If: un token condicional que si algo sucede entonces el programa realizará una acción
- Else: este token lo utilizaremos en conjunto con el token if el cual nos indica una condición si la condición no se cumple entra en juego este token que si la acción declarada en el if no sucede entonces sucederá otra acción.
- While: El token While nos indica que se generará un ciclo que mientras se cumpla una condición se repetirá una acción hasta que la condición ya no cumpla con los requerimientos

Token	Clasificación	Expresión regular
If	Palabra clave	\bif \b
Else	Palabra clave	\belse \b
While	Palabra clave	\bwhile \b

2.- Identificador
Estos serán utilizados para nombrar las variables, funciones, clases, etc, elementos que serán definidos por el usuario serán palabras que estarán formadas por letras mayúsculas, minúsculas y guión bajo junto con números.

Token	Clasificación	Expresión regular
Identificador	Identificadores	[a-zA-Z_][a-zA-Z0-9_]*

3.- Operador
Los tokens de tipo operado serán operadores aritméticos que nos permitirán realizar las principales operaciones aritméticas:

- Suma: se realizará la suma de dos números
- Resta:se realizará la resta de un número mayor y uno menor
- Multiplicación: Se realiza la multiplicación de 2 números.
- División:se realizará la división de 2 dígitos.

Token	Clasificación	Expresión regular
Operador	Operadores	\+, -, *, /

4.- Símbolo:
Los símbolos utilizados en este lenguaje nos servirán para identificar o definir uno o varios token dentro de una línea de código para encerrarlos y que estos se conviertan en un solo token.

Token	Clasificación	Expresión regular
Símbolo	Símbolos	\(, \), {, }

5.- Número:
Los números serán nuestros principales tokens para realizar las acciones en conjunto de nuestros operadores y nuestras palabras reservadas, podemos encerrar números dentro de los símbolos y realizar operaciones.

- 0, 1, 2, 3, 4, 5, 6, 7, 8, 9

Token	Clasificación	Expresión regular
Símbolo	Símbolos	\d+

6.- Espacios en blanco
Los espacios en blanco serán ignorados y no se marcan dentro del analizador ya que seran espacio sin información

Token	Clasificación	Expresión regular
Espacio	Espacios en blanco	

Código en Python para el analizador léxico:

```
Click here to ask Blackbox to help you code faster
import re

# Definición de tokens y sus expresiones regulares
token_specification = [
    ('NUMBER',   r'\d+'),       # Números enteros
    ('ID',       r'[A-Za-z_][A-Za-z0-9_]*'),  # Identificadores
    ('ASSIGN',   r'='),         # Operador de asignación
    ('OP',       r'[+\-*/]'),   # Operadores aritméticos
    ('LPAREN',   r'\('),        # Paréntesis izquierdo
    ('RPAREN',   r'\)'),        # Paréntesis derecho
    ('SKIP',     r'[ \t]+'),    # Espacios en blanco y tabulaciones
    ('MISMATCH', r'.'),         # Cualquier otro carácter
]

token_re = '|'.join(f'(?P<{pair[0]}>{pair[1]})' for pair in token_specification)
get_token = re.compile(token_re).match

def lex(characters):
    pos = 0
    tokens = []
    while pos < len(characters):
        match = get_token(characters, pos)
        if match is not None:
            type_ = match.lastgroup
            value = match.group(type_)
            if type_ == 'SKIP':
                pass
            elif type_ == 'MISMATCH':
                raise RuntimeError(f'Unexpected character: {value}')
            else:
                tokens.append((type_, value))
```

```
14
15  token_re = '|'.join(f'(?P<{pair[0]}>{pair[1]})' for pair in token_specification)
16  get_token = re.compile(token_re).match
17
18  def lex(characters):
19      pos = 0
20      tokens = []
21      while pos < len(characters):
22          match = get_token(characters, pos)
23          if match is not None:
24              type_ = match.lastgroup
25              value = match.group(type_)
26              if type_ == 'SKIP':
27                  pass
28              elif type_ == 'MISMATCH':
29                  raise RuntimeError(f'Unexpected character: {value}')
30              else:
31                  tokens.append((type_, value))
32              pos = match.end()
33          else:
34              raise RuntimeError(f'Unexpected character: {characters[pos]}')
35      return tokens
36
37  # Ejemplo de uso
38  code = 'x = 42 + y'
39  print(lex(code))
```

Resultados en la terminal:

```
PS C:\Users\paoli\Desktop\Graficación\Codigos\Lenguaje> & C:/Users/paoli/AppData/Local/Micr
osoft/WindowsApps/python3.12.exe c:/Users/paoli/Desktop/Graficación/Codigos/Lenguaje/lengua
je.py
[('ID', 'x'), ('ASSIGN', '='), ('NUMBER', '42'), ('OP', '+'), ('ID', 'y')]
```