Atividade 01: Implementação de um Analisador Léxico Simples

Objetivo

Implementar um analisador léxico que leia códigos-fonte em uma linguagem fictícia e gere uma lista de tokens com seus respectivos tipos. Este exercício combina teoria e prática, permitindo que você construa um componente essencial de um compilador em uma linguagem de programação de sua escolha, testando-o com múltiplos exemplos.

Instruções

1. Escolha uma linguagem de programação para implementar o analisador léxico.
2. Escreva um programa que realize a análise léxica dos códigos-fonte fornecidos, identificando e classificando os tokens conforme as regras da linguagem fictícia definidas abaixo.
3. A saída deve ser uma lista de tokens, cada um associado ao seu tipo, exibida em um formato claro (ex.: `(token, tipo)` ou similar, dependendo da linguagem).
4. Ignore espaços em branco, tabulações e comentários (texto após `//` até o final da linha).
5. Teste seu programa com os três códigos-fonte fornecidos e verifique se as saídas estão corretas.

Definição da Linguagem

- **Palavras-chave**: `if` , `else` , `while` , `int` , `float`
- **Identificadores**: Sequências de letras (a-z, A-Z) e dígitos (0-9), começando com uma letra (ex.: `x` , `var123`)
- **Operadores**: `+` , `-` , `*` , `/` , `=` , `!=` , `<=` , `>=`
- **Pontuação**: `(` , `)` , `{` , `}` , `;`
- **Literais**:
    - Números inteiros (ex.: `10` , `42`)

    - Números de ponto flutuante (ex.: `3.14`, `0.0`)
    - Strings: texto entre aspas duplas (ex.: `"texto"`)
- **Comentários**: Texto após `//` até o final da linha (ignorado, não gera tokens)

Códigos-Fonte para Teste

Código-Fonte 1: Simples com Operações Aritméticas

```
int a = 5 + 10 * 2; // soma e multiplicação
float b = 3.14 - 1.5;
a = a + b;
```

Código-Fonte 2: Condicionais com Strings e Comentários

```
int x = 42; // inicialização
if (x != 0) {
    print("x é positivo");
} else {
    print("x é zero"); // mensagem alternativa
}
```

Código-Fonte 3: Loop com Operadores Compostos

```
float temp = 10.5;
while (temp >= 0.0) { // loop decrescente
    temp = temp - 2.5;
    print("temperatura atual");
}
```

Requisitos da Implementação

- Leia cada código-fonte como uma string (pode ser embutido no programa ou lido de um arquivo, conforme a linguagem escolhida).
- Implemente uma lógica que varra o código caractere por caractere para identificar os tokens.
- Trate os seguintes casos:
  - Operadores de múltiplos caracteres (`!=` , `<=` , `>=`) como um único token.
  - Strings como um único token, incluindo todo o conteúdo entre aspas.
  - Números com ponto decimal como literais de ponto flutuante.
- Armazene os tokens em uma estrutura de dados (como uma lista ou array) e exiba cada token com seu tipo para cada código-fonte.

Exemplo de Saída Esperada

Para o trecho do Código-Fonte 1: `int a = 5 + 10 * 2; // soma e multiplicação` , a saída poderia ser:

```
(int, palavra-chave), (a, identificador), (=, operador), (5, literal inteiro), (+, operador), (10, literal inteiro)
```

Dicas

- Use uma estrutura (como um conjunto ou dicionário) para verificar palavras-chave rapidamente.
- Crie uma lógica que acumule caracteres em lexemas (sequências que formam um token) e os classifique ao encontrar um delimitador (espaço, pontuação, etc.).
- Para strings, processe o texto até encontrar a aspa final.
- Para comentários, ignore tudo após `//` até o final da linha.

Tarefa

1. Implemente o analisador léxico na linguagem de sua escolha.
2. Teste-o com os três códigos-fonte fornecidos.
3. Entregue o código completo e as saídas geradas para cada um dos três códigos-fonte, indicando a linguagem utilizada.