Documentação do Analisador Léxico

1. Introdução

Este documento descreve a implementação de um **Analisador Léxico** com interface gráfica em Python. O objetivo do analisador léxico é processar um código-fonte, identificar seus componentes básicos (tokens) e classificá-los em categorias, como palavras-chave, identificadores, números, operadores, etc.

O analisador foi construído utilizando as bibliotecas **tkinter** para a interface gráfica e **re** (expressões regulares) para a análise léxica. Ele é capaz de processar uma linguagem fictícia onde as palavras-chave foram substituídas por nomes de pedras preciosas no objetivo de criar uma liguagem própria para a matéria.

2. Descrição Geral

O analisador léxico lê o código-fonte de uma caixa de texto, o analisa linha por linha, e exibe os tokens identificados em uma tabela. Cada token é classificado como:

- Palavra-chave
- Número
- Identificador
- Operador
- Delimitado
- String
- Desconhecido (caso o token não se encaixe nas categorias anteriores)

Além disso, o analisador verifica se há erros léxicos no código. Se um token inválido for detectado, ele é registrado como um token "DESCONHECIDO" e é mostrado ao usuário.

3. Funcionalidades

Entrada de Código-Fonte: O código-fonte é digitado pelo usuário em uma caixa de texto de rolagem (scroll).

Análise de Tokens: Cada palavra ou símbolo é identificado e classificado de acordo com as regras definidas para a linguagem.

Interface Gráfica: A interface gráfica facilita o uso, permitindo que o usuário visualize o código, os tokens gerados, e os erros léxicos.

Detecção de Erros Léxicos: Se qualquer token não reconhecido for encontrado, ele será exibido na tabela como "DESCONHECIDO".

4. Componentes do Código

4.1 Bibliotecas Utilizadas

- **tkinter**: Utilizado para criar a interface gráfica. Inclui caixas de texto, botões e a área de exibição de resultados.
- **re**: Biblioteca de expressões regulares usada para identificar padrões no código-fonte, como números, identificadores e strings.

4.2. Tokens Definidos

Palavras-Chave: As palavras-chave da linguagem foram substituídas por nomes de pedras preciosas:

- safira (int)
- diamante (float)
- esmeralda (char)
- topazio (if)
- água-marinha (else)
- ametista (printf)
- granada (scanf)
- rubi (include)

Operadores: Operadores aritméticos e relacionais tradicionais:

- **+, -, *, /, =, ==, !=, <, >, <=, >=**

Delimitadores: Símbolos de pontuação e separação:

- **(,), {, }, ; , ,**

Literais de String: Strings são identificadas pelo padrão **\".*?\".**

4.3. Funções

4.3.1. Funções de Detecção de Tokens

- is_keyword(token):**Verifica se o token é uma palavra-chave.**
- is_delimiter(token):**Verifica se o token é um delimitador.**
- is_operator(token):**Verifica se o token é um operador.**
- is_number(token):**Verifica se o token é um número.**
- is_identifier(token): **Verifica se o token é um identificador válido (uma variável ou função).**

4.3.2. Função de Análise Léxica

lex_analyzer(code): Esta função recebe o código-fonte como entrada e realiza a análise linha por linha. Ela classifica cada token em uma das categorias definidas e armazena o resultado em uma lista. Também rastreia o número da linha para a exibição na tabela.

4.3.3. Função para Exibir Tokens

display_tokens(): Esta função coleta o código digitado pelo usuário, chama o analisador léxico e exibe os tokens classificados em uma tabela. A tabela mostra o número da linha, o lexema, o tipo de token, e o valor associado (se aplicável).

4.4. Estrutura de Dados

Os tokens são armazenados em uma lista, onde cada entrada é uma tupla contendo:

1. **Número da Linha**: A linha onde o token foi encontrado.

2. **Lexema**: O texto do token identificado.

3. **Tipo de Token**: A categoria do token (ex.: PALAVRA-CHAVE, NUMERAL).

4. **Valor**: O valor associado ao token, se aplicável (por exemplo, para números).

5. Interface Gráfica

A interface gráfica foi construída usando o **tkinter** e inclui os seguintes componentes:

Caixa de Entrada de Texto: Área onde o usuário digita o código a ser analisado.

Botão "Analisar Código": Quando clicado, inicia o processo de análise léxica.

Área de Exibição de Resultados: Mostra a tabela de tokens identificados após a análise, com colunas para o número da linha, lexema, tipo de token, e valor.

6. Fluxo de Execução

1. O usuário digita ou cola o código-fonte na caixa de texto.

2. Ao clicar no botão "Analisar Código", a função **display_tokens()** é chamada.

3. A função **lex_analyzer()** processa o código-fonte, identificando os tokens e categorizando-os.

4. O resultado é exibido na área de texto em formato de tabela.

7. Exemplo de Entrada e Saída**

```
safira main() {
   safira number;
   diamante decimal;
   esmeralda letter;

   granada("%d", &number);

   topazio (number > 5) {
      ametista("Número maior que 5\n");
   } água-marinha {
      ametista("Número menor ou igual a 5\n");
   }

   decimal = 3.14;
   letter = 'A';

   ametista("Número: %d\n", number);
   ametista("Decimal: %.2f\n", decimal);
   ametista("Letra: %c\n", letter);
}
```

Saída da Tabela de Tokens:

Linha	Lexema	Token	Valor
1	safira	PALAVRA-CHAVE	None
1	main	IDENTIFICADOR	None
1	(	DELIMITADOR	(
1	)	DELIMITADOR	)
1	{	DELIMITADOR	{
2	safira	PALAVRA-CHAVE	None
2	number	IDENTIFICADOR	None
3	diamante	PALAVRA-CHAVE	None
3	decimal	IDENTIFICADOR	None
...	...	...	...