Projeto de Analisador Léxico

Conteúdo

1 Introdução

Este documento descreve o desenvolvimento e funcionamento de um analisador léxico para a linguagem OWL2 na sintaxe Manchester. O objetivo do projeto é identificar e categorizar os elementos da linguagem de forma eficiente e robusta, utilizando a ferramenta **Flex** para o reconhecimento de padrões lexicais.

2 Objetivos

Os principais objetivos deste projeto são:

- Desenvolver um analisador léxico capaz de identificar classes, propriedades e indivíduos.
- Reconhecer palavras-chave e tokens específicos da linguagem OWL2.
- Reportar erros lexicais de maneira clara e informativa.
- Servir como base para o desenvolvimento das próximas fases do compilador.

3 Detalhamento do Código do Analisador Léxico

O código do analisador léxico desenvolvido utiliza o **Flex** para identificar e categorizar elementos da linguagem OWL2 na sintaxe Manchester. Abaixo, explicamos os principais componentes do código e suas funcionalidades:

3.1 Estrutura do Código

O código está organizado em seções principais:

- **Definição de bibliotecas e constantes**: Inclui as bibliotecas necessárias (`<iostream>`, `<string>`, `<fstream>`, `<unordered_map>`) e define constantes que representam os tokens da linguagem.
- **Tabela de símbolos**: Utiliza `unordered_map` para armazenar palavras-chave da linguagem e mapear cada uma a uma constante.
- **Mapas para classes, propriedades e indivíduos**: Permitem registrar e contar ocorrências de diferentes elementos do código.

- **Funções auxiliares**: Funções como `toLower` e `inserir` auxiliam na manipulação de strings e inserção de dados em mapas.
- **Definição de padrões lexicais**: Utiliza expressões regulares para identificar diferentes tipos de tokens, como classes, propriedades, indivíduos e palavras-chave.
- **Regras de reconhecimento**: Implementadas após `%%`, especificam como cada padrão é tratado.

3.2 Definição de Tokens e Expressões Regulares

As expressões regulares utilizadas definem os seguintes tokens principais:

- **Classes**: Identificadas pelo padrão `[A-Z][a-zA-Z]*`, representam nomes de classes iniciando com letra maiúscula.
- **Propriedades**: Reconhecidas pelo padrão `[a-z][a-zA-Z]*`, representam propriedades iniciando com letra minúscula.
- **Números**: Identificados pelo padrão `[0-9]+`, permitem capturar valores numéricos.
- **Palavras-chave**: Como `Class:`, `SubClassOf:`, definidas diretamente na tabela de símbolos.

3.3 Processamento de Tokens

Para cada padrão identificado, uma ação é executada:

- **Classes**: São adicionadas ao mapa `classes` e contadas.
- **Propriedades**: São registradas no mapa `propriedades` e contadas.
- **Palavras-chave**: Validadas na tabela de símbolos; caso não sejam reconhecidas, um erro é reportado.
- **Erros**: São capturados e reportados com informações sobre a linha e o conteúdo.

3.4 Funções Auxiliares

- `toLower`: Converte strings para letras minúsculas, garantindo consistência na comparação de tokens.
- `inserir`: Insere tokens identificados nos mapas apropriados.

3.5 Resultados

Ao processar um arquivo de teste, o analisador:

- Identifica corretamente classes, propriedades e indivíduos.
- Reporta erros de sintaxe, como palavras-chave mal formatadas ou simbolizações inválidas.
- Exibe um resumo com as contagens de cada tipo de token.

Essa estrutura garante que o analisador léxico seja eficiente e robusto, servindo como base para as próximas etapas do compilador.

4 Como Usar o Projeto

Para instruções detalhadas sobre como configurar e executar o analisador léxico, acesse o repositório do projeto no GitHub:
https://github.com/usuario/Compiladores.git
No repositório, você encontrará um guia completo com os passos necessários para clonar, compilar e executar o projeto.

```
git clone https://github.com/usuario/Compiladores.git
```

5 Conclusão

O analisador léxico desenvolvido alcança os objetivos propostos, sendo capaz de identificar os principais elementos da linguagem OWL2 na sintaxe Manchester, reportar erros de maneira clara e eficiente, e oferecer uma base sólida para o desenvolvimento do analisador sintático e semântico. Trabalhos futuros incluem a implementação dessas próximas fases e a integração do analisador em um ambiente de compilação completo.