Projeto Conceitual do Compilador para a Linguagem Perpertuum2024-1

1. Introdução

Este documento descreve a implementação de um Static Checker para a linguagem Perpertuum2024-1, desenvolvido como parte da disciplina de Compiladores no primeiro semestre de 2024. O objetivo do projeto é realizar a análise léxica e parte da análise sintática para validar textos escritos na linguagem Perpertuum2024-1.

2. Estrutura do Projeto

O projeto é composto pelos seguintes módulos:

1. **alfabeto.py**: Define os caracteres válidos na linguagem Perpertuum2024-1.
2. **automato.py**: Implementa o autômato léxico responsável pela formação de átomos a partir do código fonte.
3. **header.py**: Contém informações sobre a equipe responsável pelo projeto.
4. **lexer.py**: Implementa o analisador léxico, utilizando o autômato léxico para identificar tokens no código fonte.
5. **main.py**: Programa principal que coordena a execução do analisador léxico e a geração de relatórios.
6. **palavras.py**: Define as palavras reservadas e símbolos da linguagem Perpertuum2024-1.
7. **relatorio.py**: Gera relatórios de análise léxica e da tabela de símbolos.
8. **symbol_table.py**: Implementa a tabela de símbolos utilizada durante a análise léxica.
9. **utils.py**: Contém funções utilitárias, como leitura de arquivos e manipulação de strings.

3. Descrição dos Módulos

1. **alfabeto.py**
   - Define duas listas de caracteres: ALFABETO e ALFABETO_STRING, que contêm os caracteres válidos na linguagem.
2. **automato.py**
   - Implementa a classe AutomatoLex, responsável por simular um autômato finito determinístico para a análise léxica.
   - Métodos principais:
     - run: Executa o autômato sobre o código fonte.
     - add_estado: Adiciona estados ao autômato.
     - q0 a q17: Definem as transições entre estados com base nos caracteres de entrada.
3. **header.py**
   - Contém informações sobre a equipe, como nomes, emails e telefones dos integrantes.
4. **lexer.py**
   - Implementa a classe Lexer, que coordena a análise léxica utilizando o AutomatoLex.
   - Métodos principais:
     - filtro: Remove comentários e caracteres inválidos do código fonte.
     - formar_atomo: Utiliza o autômato léxico para formar átomos.
     - is_reservada: Verifica se um lexeme é uma palavra reservada.
     - get_lexeme_id: Atribui IDs únicos a lexemes.
5. **main.py**

    - Programa principal que executa o processo de análise léxica e geração de relatórios.
    - Fluxo principal:
        - Leitura do arquivo fonte.
        - Inicialização do analisador léxico e da tabela de símbolos.
        - Execução do analisador léxico em loop até o fim do arquivo.
        - Geração dos relatórios .LEX e .TAB.
6. **palavras.py**
    - Define uma lista de tuplas PALAVRAS, onde cada tupla contém uma palavra reservada e seu código correspondente.
7. **relatorio.py**
    - Implementa a classe Relatorio, responsável por gerar os relatórios de análise léxica e da tabela de símbolos.
    - Métodos principais:
        - gerar_lex: Gera o relatório .LEX.
        - gerar_tab: Gera o relatório .TAB.
8. **symbol_table.py**
    - Implementa a classe SymbolTable, que gerencia a tabela de símbolos.
    - Métodos principais:
        - add: Adiciona um novo símbolo à tabela.
        - getIndex: Retorna o índice de um símbolo na tabela.
9. **utils.py**
    - Contém funções auxiliares, como:
        - read_file: Lê o conteúdo de um arquivo.
        - uppercase_menos_sequencias: Converte texto para maiúsculas, exceto sequências de escape.
        - analise_escopo: Determina o escopo atual baseado no token anterior.
        - remover_invalidos: Remove caracteres inválidos de uma string.

4. Utilização do Compilador

Para utilizar o compilador, siga os passos abaixo:

1. **Preparar o código fonte**
    - Crie um arquivo com extensão .241 contendo o código na linguagem Perpertuum2024-1.
2. **Executar o programa**
    - Execute o programa main.py. Você será solicitado a fornecer o nome do arquivo (sem a extensão).
3. **Gerar Relatórios**
    - O programa gerará dois arquivos de relatório na mesma pasta do código fonte:
        - .LEX: Relatório da análise léxica, contendo os tokens identificados.
        - .TAB: Relatório da tabela de símbolos, contendo informações sobre os identificadores.