PRIMEIRA ENTREGA - TRABALHO DE COMPILADORES

DESCRIÇÃO DO PROJETO

Este projeto implementa um analisador léxico e um analisador sintático para um subconjunto da linguagem de programação Ada, utilizando as ferramentas Haskell, Alex (gerador de lexer) e Happy (gerador de parser). O objetivo é criar as primeiras fases de um compilador completo, capazes de transformar código fonte em uma representação estrutural (AST - Abstract Syntax Tree).

COMPONENTES DO SISTEMA

1. ANALISADOR LÉXICO (Lexer.x)

O analisador léxico é responsável por transformar o código fonte Ada em um fluxo sequencial de tokens. Desenvolvido utilizando Alex, este componente reconhece os elementos fundamentais da linguagem.

Elementos Reconhecidos:

- **Estruturas de Controlo**: `procedure`, `is`, `begin`, `end`, `if`, `then`, `else`, `while`, `loop`
- **Operadores Lógicos**: `and`, `or`, `not`
- **Valores Booleanos**: `True`, `False`
- **Funções de E/S**: `Put_Line` (output), `Get_Line` (input)

Categorias de Tokens:

- **Identificadores**: Sequências alfanuméricas que começam com uma letra (ex: `variavel1`, `contador`)
- **Inteiros**: Sequências de dígitos decimais (ex: `123`, `45`)
- **Strings**: Texto delimitado por aspas duplas (ex: `"Hello World"`)
- **Operadores**:
  - Atribuição: `:=`
  - Aritméticos: `+`, `-`, `*`, `/`

    - Comparação: `=`, `<`, `>`
- **Delimitadores**: `(`, `)`, `;`

Funcionalidades Avançadas:

- Tratamento automático de espaços em branco e tabulações
- Suporte a comentários de linha única (iniciados por `--`)
- Rastreamento da posição dos tokens no código fonte
- Geração automática de código Haskell (Lexer.hs)

2. ANALISADOR SINTÁTICO (Parser.y)

O analisador sintático constrói uma Árvore Sintática Abstrata (AST) a partir do fluxo de tokens gerado pelo lexer, aplicando as regras gramaticais definidas para a linguagem. Desenvolvido com Happy, este componente valida a estrutura sintática do código fonte.

Construções Sintáticas Suportadas:

Declarações:

- Declarações de variáveis com inicialização opcional
- Estrutura básica de programas: `procedure [nome] is begin ... end;`

Instruções:

- Atribuições de variáveis: `identificador := expressão;`
- Estruturas condicionais:

```
if condição then
    ...
else
    ...
end if;
```