Projeto de Compiladores - Parte 1: Análise Léxica e Sintática

Descrição Geral

Este projeto corresponde à primeira parte de um compilador desenvolvido para um subconjunto da linguagem de programação Kotlin. A tarefa principal consistiu na implementação de um analisador léxico (Lexer.x) e de um analisador sintático (Parser.y) que deve construir e devolver a Árvore Sintática Abstrata (AST.hs) que representa o programa kotlin fornecido como input.

Ficheiros principais

- **Lexer.x**: Contém a definição do analisador léxico, responsável por identificar os tokens no código de input e classificá-los conforme as regras léxicas definidas para o subconjunto pedido da linguagem Kotlin.

- **Parser.y**: Define o analisador sintático, que utiliza os tokens gerados pelo lexer para construir a árvore sintática abstrata (AST). Este implementa as regras gramaticais necessárias para o subset da linguagem Kotlin pedido.

- **AST.hs**: Define a estrutura da árvore sintática abstrata (AST) para representar a estrutura hierárquica do código Kotlin processado. Este ficheiro inclui os tipos e construtores necessários para representar as diferentes expressões e comandos do subconjunto pedido da linguagem.

Ficheiros de Teste

- **testLexer.hs**: Primeiramente utilizado para realizar testes de funcionamento do analisador léxico, e para garantir que os tokens fossem identificados corretamente.

- **testAST.hs**: Criado para testar e verificar a construção correta da AST a partir de expressões e comandos válidos do subconjunto da linguagem (também inclui a estrutura do Lexer.x como fazia o testLexer.x).

Funcionalidade Implementada

1. **Análise Léxica (Lexer)**:
    - Identifica e classifica tokens, incluindo operadores aritméticos, expressões booleanas, variáveis, expressões condicionais, while loops (if, else, while, print, readln), e delimitadores.

2. **Análise Sintática (Parser)**:
    - Constrói a AST para expressões e comandos do subconjunto Kotlin especificado, incluindo:
        - **Expressões**: Aritméticas, booleanas, print e readln.
        - **Comandos**: Declarações de variáveis, atribuições, expressões condicionais (if-then-else) e while loops.
    - O ficheiro Parser.y estabelece a gramática e as regras de construção da AST a partir dos tokens fornecidos pelo lexer.

3. **AST (Abstract Syntax Tree)**:
    - O ficheiro AST.hs define a estrutura de dados que armazena a AST, representando a hierarquia do código Kotlin processado. Essa árvore serve como a base para as próximas etapas de análise e execução do código.

Como Executar

Para compilar e testar o projeto, é preciso de se certificar de ter um o Haskell configurado e atualizado, para se encontrar no ambiente correto.

1. Compilar os ficheiros principais:

alex Lexer.x happy
Parser.y ghci

2. Executar os ficheiros de teste e testar com um ficheiro de teste: Exemplo:

:l testLexer.hs # Testa o lexer main
(está uma pasta dentro da pasta source com ficheiros .kt para teste)
exs/ex1.kt

Ou: :l testAST.hs # Devolve a análise Léxica e a AST do ficheiro .kt
main
exs/ex8.kt

Estrutura das Pastas

- **src/**: Contém os ficheiros principais como o Lexer.x, Parser.y, AST.hs e os ficheiros de teste mencionados anteriormente.
- **exs/**: Pasta com os ficheiros.kt para utilizar no testAST.hs e testLexer.hs

Referências

- **Subconjunto simples da Linguagem Kotlin**: https://kotlinlang.org/docs/basicsyntax.html
- Site para verificar as regras gramaticais e o conjunto de expressões suportadas e pedidas pelo professor neste projeto.