Compilador para Subconjunto de Kotlin

0. Identificação

Grupo: T02_G05

Nomes: André Filipe Carvalho Guedes Borges Adrêgo
Bruno Alexandre Magalhães Costa

1. Introdução

Este projeto implementa um compilador simples para um subconjunto da linguagem Kotlin, focando em variáveis, tipos básicos (inteiro, ponto flutuante, booleano, string), operações aritméticas e lógicas, estruturas de controle (if, while), e entrada/saída (readln, print). Este compilador usa **Alex** para análise léxica e **Happy** para análise sintática, escrito em Haskell.

2. Objetivos do Projeto

- Desenvolver uma análise léxica para identificar tokens.
- Implementar um parser que reconheça a estrutura da linguagem e gere uma árvore sintática abstrata (AST).
- Definir nós da AST que representem corretamente as expressões e comandos do subconjunto de Kotlin.
- Implementar um tratamento básico de erros de parsing.

3. Estrutura do Projeto

- **Lexer.x**: Define os tokens e as expressões regulares.
- **Parser.y**: Define a gramática da linguagem, com precedências e AST.
- **Main.hs**: Função principal para inicializar a análise.
- **README.pdf**: Documento de descrição detalhada do projeto.

4. Funcionalidades da Linguagem

A linguagem aceita:

- **Declaração de variáveis e constantes**: val para constantes imutáveis e var para variáveis mutáveis.
- **Tipos suportados**: Int, Float, Boolean, String.
- **Operadores**:
    - **Aritméticos**: +, -, *, /, %.
    - **Lógicos**: &&, ||, !.
    - **Comparativos**: ==, !=, <, <=, >, >=.
- **Controle de Fluxo**: if, else, while.
- **Entrada e Saída**: readln(), print().

5. Instruções para Compilação e Execução

Para compilar e executar o projeto:

1. Gere o analisador léxico:
   alex Lexer.x
2. Gere o parser:
   happy Parser.y
3. Compile o projeto com o GHC:
   ghc Main.hs -o compilador
4. Execute o compilador:
   ./compilador

6. Análise Léxica (Lexer)

O arquivo Lexer.x define tokens utilizando expressões regulares, mapeando palavras-chave (if, while, val, var), operadores (+, -, *, &&, ||), e literais (true, false, números, strings). Tokens reconhecidos incluem:

- **Palavras-chave**: if, else, while, fun, main.
- **Tipos**: Int, Float, Boolean, String.
- **Operadores e delimitadores**: +, -, *, /, (,).

7. Análise Sintática e AST

A análise sintática é realizada pelo Parser.y, que cria a AST representando a estrutura hierárquica do código. Cada expressão e comando são representados como nós de tipos específicos, por exemplo:

- **Nó para Variáveis**: VarDecl e ValDecl.
- **Nó para Estruturas de Controle**: IfNode, IfElseNode, WhileNode.

- **Nó para Operações**: AddNode, SubNode, MultNode, DivNode, AndNode, OrNode.

8. Tratamento de Erros

O compilador apresenta tratamento básico de erros de sintaxe na função parseError em Parser.y. Quando um token inválido é encontrado, uma mensagem de erro descritiva é exibida com o token inesperado.

9. Conclusão

Este projeto de compilador para um subconjunto da linguagem Kotlin proporcionou uma visão prática dos processos fundamentais envolvidos no desenvolvimento de compiladores, incluindo a análise léxica, a análise sintática e a geração de uma árvore sintática abstrata (AST). O projeto alcançou com sucesso os objetivos de reconhecer e processar estruturas típicas de uma linguagem de programação.