Universidade Federal de Goiás
Instituto de Informática
Bacharelado em Ciência da Computação
Compiladores
2025-1

Compilador para a Linguagem Goianinha
Trabalho 2 Análise Sintática e Integração da Análise Sintática com a Análise Léxica

Thierson Couto Rosa

1 Objetivo

Este texto especifica o analisador sintático para a linguagem *Goianinha*. A Seção 2 descreve a gramática da linguagem *Goianinha*. A Seção 3 corresponde à especificação do analisador sintático para a linguagem. Nesse trabalho, além de implementar o analisador sintático usando uma ferramenta do tipo Bison ou YACC, o(a) aluno(a) deve fazer a integração entre o código do analisador sintático e o código do analisador léxico, gerado no trabalho anterior. A seção 4 descreve os detalhes de como a Etapa II do projeto deve ser entregue.

2 Gramática para a linguagem *Goianinha*

A seguir, é apresentada a gramática para a linguagem *Goianinha*. Os não-terminais da gramática iniciam-se com letra maiúscula e o símbolo inicial é *Programa*. Os terminais aparecem em negrito quando são formados por palavras ou por sequência de caracteres. Exemplo: **int**, +, **-** **()**, <=, etc.

Gramática:

$$
\begin{array}{lcl}
Programa & \rightarrow & DeclFuncVar\ DeclProg \\
DeclFuncVar & \rightarrow & Tipo\ \textbf{id}\ DeclVar\ ;\ DeclFuncVar \\
 & & |Tipo\ \textbf{id}\ DeclFunc\ DeclFuncVar \\
 & & |\epsilon \\
DeclProg & \rightarrow & \textbf{programa}\ Bloco \\
DeclVar & \rightarrow & ,\ \textbf{id}\ DeclVar \\
 & & |\ \epsilon \\
DeclFunc & \rightarrow & (\ ListaParametros\ )\ Bloco \\
ListaParametros & \rightarrow & \epsilon \\
 & & |ListaParametrosCont \\
ListaParametrosCont & \rightarrow & Tipo\ \textbf{id} \\
 & & |Tipo\ \textbf{id}\ ,\ ListaParametrosCont \\
Bloco & \rightarrow & \{\ ListaDeclVar\ \ ListaComando\ \} \\
ListaDeclVar & \rightarrow & \epsilon \\
 & & |Tipo\ \textbf{id}\ DeclVar\ ;\ ListaDeclVar \\
Tipo & \rightarrow & \textbf{int} \\
 & & |\textbf{car} \\
ListaComando & \rightarrow & Comando \\
 & & |Comando\ \ ListaComando \\
Comando & \rightarrow & ; \\
 & & |Expr\ ; \\
 & & |\textbf{retorne}\ \ Expr\ ; \\
 & & |\textbf{leia id}\ ; \\
 & & |\textbf{escreva}\ \ Expr\ ; \\
 & & |\textbf{escreva}\ \ \text{“}\textbf{cadeiaCaracteres}\text{”}\ ; \\
 & & |\textbf{novalinha}\ ; \\
 & & |\textbf{se}\ (\ Expr\ )\ \textbf{entao}\ Comando \\
 & & |\textbf{se}\ (\ Expr\ )\ \textbf{entao}\ Comando\ \ \textbf{senao}\ Comando \\
 & & |\textbf{enquanto}\ (\ Expr)\ \textbf{execute}\ Comando \\
 & & |Bloco \\
Expr & \rightarrow & OrExpr \\
 & & |\textbf{id}{=}Expr
\end{array}
$$

$$
\begin{array}{lcl}
OrExpr & \rightarrow & OrExpr \ \textbf{ou} \ AndExpr \\
 & & |AndExpr \\
 & & \\
AndExpr & \rightarrow & AndExpr \ \textbf{e} \ EqExpr \\
 & & |EqExpr \\
 & & \\
EqExpr & \rightarrow & EqExpr == DesigExpr \\
 & & |EqExpr \ \textbf{!=} \ DesigExpr \\
 & & |DesigExpr \\
 & & \\
DesigExpr & \rightarrow & DesigExpr < AddExpr \\
 & & | \ DesigExpr > AddExpr \\
 & & | \ DesigExpr >= AddExpr \\
 & & | \ DesigExpr <= AddExpr \\
 & & |AddExpr \\
 & & \\
AddExpr & \rightarrow & AddExpr + MulExpr \\
 & & |AddExpr - MulExpr \\
 & & |MulExpr \\
 & & \\
MulExpr & \rightarrow & MulExpr * UnExpr \\
 & & |MulExpr \ / \ UnExpr \\
 & & | \ UnExpr \\
 & & \\
UnExpr & \rightarrow & -PrimExpr \\
 & & |! \ PrimExpr \\
 & & |PrimExpr \\
 & & \\
PrimExpr & \rightarrow & \textbf{id} \ (\ ListExpr \ ) \\
 & & | \ \textbf{id} \ (\ ) \\
 & & | \ \textbf{id} \\
 & & | \ \textbf{carconst} \\
 & & | \ \textbf{intconst} \\
 & & | \ (\ Expr \ ) \\
 & & \\
ListExpr & \rightarrow & Expr \\
 & & | \ ListExpr \ , \ Expr
\end{array}
$$

3 Analisador Sintático

A função do compilador que implementa o analisador sintático pode ser gerada automaticamente utilizando-se um gerador de analisadores sintáticos ou *parsers*. Neste trabalho poderá ser utilizado um dos seguintes geradores: YACC ou o Bison. Ambos utilizam o método LALR para a geração do analisador sintático. O trabalho de implementação do analisador sintático consiste na preparação do arquivo de entrada para o gerador de analisador sintático e na adaptação da função analisador sintático gerada para que possa funcionar utilizando a função

analisador léxico obtida no trabalho 1.

O programa principal que chama a função analisador sintático deve receber o nome do arquivo a ser compilado como parâmetro de entrada. Especificamente, dado que o programa executável do analisador sintático tenha o nome *goianinha*, e supondo que o arquivo de entrada seja *teste.g*, deve ser possível executar a análise sintática do arquivo através do seguinte comando em uma *shell* do Linux: *./goianinha teste.g*

Deve ser implementado o corpo da função yyerror(), de tal modo que erros sintáticos detectados pela função yyparse() sejam emitidos na tela do computador, juntamente com o número da linha onde o erro foi detectado. A mensagem de erro deve iniciar com a palavra **ERRO:** seguida por um espaço.

4 Informações Sobre a Implementação e a Entrega

O trabalho é individual e deve ser entregue até o dia 11/05/2025 via tarefa criada na Plataforma Turing. Devem ser entregues:

- O código de entrada para o gerador de analisador léxico utilizado (arquivo com extensão ".l").
- O código de entrada para o gerador de analisador sintático utilizado (arquivo com extensão ".y").
- O *Makefile* contendo:
  - Comandos de execução do gerador de analisador léxico (Flex ou JFlex) para conversão do arquivo de entrada do gerador de analisador léxico em programas em C.
  - Comandos de execução do gerador de analisador sintático (YACC ou Bison) para conversão do arquivo de entrada do gerador de analisador sintático.
  - Comandos de compilação e link-edição para compilar e ligar o programa principal com os códigos dos analisadores léxicos e sintáticos gerados.

Os itens acima devem estar agrupados em um arquivo do tipo *tar* compactado com o utilitário gzip e submetidos como uma tarefa a ser criada no ambiente Moodle da disciplina. os códigos gerados devem estar preparados para executarem no sistema operacional Linux (ambiente onde os trabalhos serão avaliados). O programa executável deve ter o nome *goianinha*.

Importante:

Cópias idênticas ou modificadas dos códigos ou de partes dos códigos são consideradas plágios. **Plágio é crime**. O aluno que cometer plágio em seu trabalho receberá nota zero no mesmo.