Trabalho 2 Compiladores

Rodrigo Pimenta, Thiago Martinho

12 de outubro de 2015

1 Introdução

Este trabalho teve como objetivo a construção do analisador sintático do compilador proposto para a linguagem GPortugol cujo manual pode ser encontrado em (`http://www.inf.ufes.br/~mberger/Disciplinas/2015_2/Compiladores/manualGPortugol.pdf`). A base desse trabalho de número dois é o analisador léxico construído com o auxilio da ferramenta Flex. Para o desenvolvimento do analisador sintático, foi utilizado a linguagem de programação C e a ferramenta Bison.

2 Analisador Sintático

Um analisador sintático consiste em verificar a sintaxe de um programa escrito em uma determinada linguagem. Caso o código esteja totalmente de acordo com a gramática estabelecidao, nenhuma mensagem de erro é mostrada para o usuário, caso contrário, é informado qual a linha apresenta um erro de sintaxe.

Esses possíveis erros podem ser de natureza diferente como, por exemplo, utilização de palavras reservadas de forma diferente do contexto que elas devem ser empregadas, comandos incompletos, falta de ponto e virgula, entre outros.

A ferramenta Bison (`https://www.gnu.org/software/bison/`) é um gerador de interpretadores que analisa a sintaxe de um arquivo de entrada. Segundo a descrição presente no site da ferramenta, ela se intitula como *um gerador de interpretadores para fins gerais que converte uma descrição gramatical de uma gramática livre de contexto "LALR(1)"para um programa C que analisa aquela gramática. Uma vez que você esteja acostumado com "Bison", você pode usá-lo para desenvolver um ampla quantidade de interpretadores de linguagem, daqueles usados em simples calculadoras de mesa até linguagens de programação complexas.*

Com o auxilio dessa ferramenta, em parceria com o Flex, foi construído neste trabalho um analisador sintático para a linguagem GPortugol de acordo coma gramática definida por nós.

3 Operações Suportadas

Nesta seção, estão listadas e descritas todas as operações suportadas pela gramática construída.

3.1 Declaração de Variáveis

A declaração de variáveis é feita da forma como está presente no manual da linguagem, com exceção da definição do tipo matriz. Neste trabalho, foi considerado que uma matriz é composta de n dimensões e de um tipo. Esse tipo, na definição original da ferramenta, deveria ser escrito no plural, porém para nós ele deve ser declarado da mesma forma que uma variável não-matriz, ou seja, no singular.

O bloco de declaração de variáveis é opcional na construção de um programa, entretanto se este bloco for inserido, pelo menos uma variável deverá ser declarada.

3.2 Comandos de Seleção

Os comandos de seleção presentes na linguagem são o *se-entao-senao-fim-se* e o *avalie-caso-pare-fim-avalie*. O comando *se* pode ser acompanhado ou não de uma cláusula *senao* mas obrigatoriamente deverá ser terminado com *fim-se*. Já o comando *avalie* deverá ter uma expressão a ser avaliada e alguns casos a serem considerados. Esses casos não podem conter expressões lógicas e nem aritméticas, sendo possíveis apenas valores já conhecidos, como inteiros ou lógicos (da mesma forma que na linguagem C) e após o fim de cada bloco de caso, deverá ser inserido o delimitador *pare*.

3.3 Comandos de Repetição

Os comandos de repetição suportados na linguagem são o *enquanto*, o *faca-enquanto* e o *para*. A definição e utilização desses comandos funcionam de acordo com a especificação original da linguagem.

O comando *faca-enquanto* foi inserido na gramática e pode ser utilizado da seguinte forma: *faca: BLOCO DE CÓDIGO enquanto (EXPRESSAO A SER AVALIADA) fim-enquanto*.

3.4 Declaração de Função

As declarações de funções seguem de forma similar a definida pelo manual da linguagem, onde vale destacar que não é possível uma função retornar uma matriz.

A única exceção para declarações de funções é relativo a posição delas. As função devem ser declaradas abaixo da declaração de variáveis (quando houver) e acima do bloco principal do programa.

Não é possível declarar funções de nomes *leia* e *imprima* por serem reservadas da linguagem.

Todas as funções devem possuir uma instrução de retorno e estas não podem conter combinações de chamadas de outras funções com expressões lógicas e/ou aritméticas. Para mais detalhes consulte a seção 3.6.

3.5 Chamada de Funções

As chamadas de funções podem conter expressões, outras funções, variáveis, literais, valores lógicos, inteiros ou reais como parâmetro. Nessa fase de construção do compilador, não é verificado se existe ou não uma função declarada correspondente.

3.6 Atribuições

Nosso analisador sintático verificará tanto operações aritméticas como operações lógicas (ou relacionais). Durante essa fase de avaliação do compilador, expressões que podem não fazer sentido também são aceitas, uma vez que estão escritas na forma correta. O sentido da expressão será avaliado no próximo trabalho (analisador semântico). Não são aceitas operações que incluem chamadas de funções combinadas com expressões aritméticas ou lógicas como por exemplo $x + fatorial(2);$ e nem expressões que envolvem os operadores $++$ ou $--$.

Alguns exemplos de expressões incorretas mas que são aceitas pelo analisador sintático são:

$x = verdadeiro^3$;
$x = 10 + verdadeiro$;
$x[2] =$ "$exemplo$" $* 7$;

4 A gramática

Neste seção, está descrito de forma mais formal a gramática construída. O símbolo * indica a ocorrência de zero ou mais vezes de um item, o símbolo + indica a ocorrência de uma ou mais vezes e o símbolo ? indica que o item é opcional. A avaliação de uma cadeia de caracteres sempre começa pela regra *ALGORITMO*.

- ALGORITMO : token_algoritmo token_identificador token_simboloPontoVirgula PROGRAMA
- PROGRAMA : (VARIAVEIS)? (DECLARACAO_FUNCAO)? PROGRAMA_PRINCIPAL;
- VARIAVEIS : token_variaveis DECLARACAO_VARIAVEL token_fimVariaveis

- DECLARACAO_VARIAVEL : token_identificador token_simboloVirgula DECLARACAO_VARIAVEL + | token_identificador token_simboloDoisPontos TIPO_VARIAVEIS token_simboloPontoVirgula (DECLARACAO_VARIAVEL+)?
- DECLARACAO_FUNCAO : (DECLARACAO_FUNCAO+)? token_funcao token_identificador token_simboloAbreParentese (PARAMETRO_DECLARACAO_FUNCAO)? token_simboloFechaParentese token_simboloDoisPontos (TIPO_VARIAVEL_PRIMITIVO)? ROTINA_FUNCAO token_fimFuncao
- PARAMETRO_DECLARACAO_FUNCAO : (PARAMETRO_DECLARACAO_FUNCAO token_simboloVirgula +)? token_identificador token_simboloDoisPontos TIPO_VARIAVEL_PRI
- ROTINA_FUNCAO : (DECLARACAO_VARIAVEL)? token_inicio LISTA_COMANDOS token_fim
- COMANDO_RETORNO : token_retorne (EXPRESSAO)? token_simboloPontoVirgula
- TIPO_VARIAVEIS : MATRIZ | TIPO_VARIAVEL_PRIMITIVO
- TIPO_VARIAVEL_PRIMITIVO : token_tipoReal | token_tipoInteiro | token_tipoCaractere | token_tipoLogico | token_tipoLiteral
- MATRIZ : token_tipoMatriz POSICAO_MATRIZ token_de TIPO_VARIAVEL_PRIMITIVO
- POSICAO_MATRIZ : (POSICAO_MATRIZ+)? token_simboloAbreColchete token_identificador token_simboloFechaColchete | (POSICAO_MATRIZ+)? token_simboloAbreColc token_inteiro token_simboloFechaColchete
- PROGRAMA_PRINCIPAL : token_inicio (LISTA_COMANDOS+)? token_fim
- LISTA_COMANDOS : (LISTA_COMANDOS+)? COMANDO_ATRIBUICAO token_simboloPontoVirgula | (LISTA_COMANDOS+)? COMANDO_ENQUANTO | (LISTA_COMANDOS+)? COMANDO_PARA | (LISTA_COMANDOS+)? COMANDO_LEIA | (LISTA_COMANDOS+)? COMANDO_IMPRIMA | (LISTA_COMANDOS+)? COMANDO_CHAMADA_FUNCAO token_simboloPontoVirgula | (LISTA_COMANDOS+)? COMANDO_SE | (LISTA_COMANDOS+)? COMANDO_FACA_ENQUANTO | (LISTA_COMANDOS)+? COMANDO_AVALIE | (LISTA_COMANDOS+)? COMANDO_RETORNO | (LISTA_COMANDOS+)? COMANDO_MAIS_MAIS_MENOS_ME
- COMANDO_ATRIBUICAO : token_identificador token_operadorAtribuicao VALOR_A_SER_ATRIBUIDO | ACESSO_MATRIZ token_operadorAtribuicao VALOR_A_SER_ATRIBUIDO
- VALOR_A_SER_ATRIBUIDO : (VALOR_A_SER_ATRIBUIDO+)? EXPRESSAO | (VALOR_A_SER_ATRIBUIDO+)? COMANDO_CHAMADA_FUNCAO

- COMANDO_ENQUANTO : token_enquanto EXPRESSAO token_faca LISTA_COMANDOS token_fimEnquanto
- COMANDO_PARA token_para token_identificador token_de EXPRESSAO (token_passo NUMERO)? token_faca LISTA_COMANDOS token_fimPara
- COMANDO_SE : token_se EXPRESSAO token_entao LISTA_COMANDOS (token_senao LISTA_COMANDOS)? token_fimSe
- COMANDO_FACA_ENQUANTO : token_faca token_simboloDoisPontos LISTA_COMANDOS token_enquanto token_simboloAbreParentese EXPRESSAO token_simboloFechaParentese token_fimEnquanto
- COMANDO_AVALIE : token_avalie token_simboloAbreParentese token_identificador token_simboloFechaParentese token_simboloDoisPontos AVALIE_CASO token_fimAvalie
- AVALIE_CASO: (AVALIE_CASO+)? token_caso token_identificador token_simboloDoisPontos LISTA_COMANDOS token_pare token_simboloPontoVirgula | (AVALIE_CASO+)? token_caso token_inteiro token_simboloDoisPontos LISTA_COMANDOS token_pare token_simboloPontoVirgula
- COMANDO_LEIA : token_identificador token_operadorAtribuicao token_leia token_simboloAbreParentese token_simboloFechaParentese token_simboloPontoVirgula
- COMANDO_IMPRIMA : token_imprima token_simboloAbreParentese PARAMETROS_FUNCAO token_simboloFechaParentese token_simboloPontoVirgula
- PARAMETROS_FUNCAO : (PARAMETROS_FUNCAO token_simboloVirgula+)? EXPRESSAO | (PARAMETROS_FUNCAO token_simboloVirgula+)? COMANDO_CHAMA
- COMANDO_CHAMADA_FUNCAO : token_identificador token_simboloAbreParentese (PARAMETROS_FUNCAO)? token_simboloFechaParentese
- COMANDO_MAIS_MAIS_MENOS_MENOS : token_identificador token_operadorSomaSo token_simboloPontoVirgula | token_operadorSomaSoma token_identificador token_simboloPontoVirgula | token_identificador token_operadorSubtraiSubtrai token_simboloPontoVirgula | token_operadorSubtraiSubtrai token_identificador token_simboloPontoVirgula
- EXPRESSAO : EXPRESSAO_SIMPLES | EXPRESSAO_SIMPLES OPERADORES_RELACIONAIS EXPRESSAO_SIMPLES
- EXPRESSAO_SIMPLES : TERMO | EXPRESSAO_SIMPLES OPERADORES_BAIXA_PR TERMO
- TERMO : FATOR | TERMO OPERADORES_ALTA_PRECEDENCIA FATOR

- FATOR : token_identificador | token_operadorNao FATOR | NUMERO | ACESSO_MATRIZ | token_verdadeiro | token_falso | token_literal | token_caractere | token_simboloAbreParentese EXPRESSAO token_simboloFechaParentese
- NUMERO : token_inteiro | token_inteiroNegativo | token_real | token_realNegativo
- ACESSO_MATRIZ : token_identificador POSICAO_MATRIZ
- OPERADORES_RELACIONAIS : token_operadorIgualIgual | token_operadorMenor | token_operadorMenorIgual | toen_operadorMaiorIgual | token_operadorMaior | token_operadorDiferente
- OPERADORES_BAIXA_PRECEDENCIA : token_operadorMais | token_operadorMenos | token_operadorOU
- OPERADORES_ALTA_PRECEDENCIA : token_operadorVezes | token_operadorDividir | token_operadorPorcento| token_operadorPotencia | token_operadorE