COMPILADORES

TRABALHO 2 -ANALISADOR SINTÁTICO

Nomes: RAs:

Bruno Ferreira Leal 151042161

Paulo Henrique Paim dos Santos 151040745

Introdução

Para o presente trabalho, foi utilizado como fonte as saídas providas pelo analisador léxico desenvolvido no trabalho 1. No entanto, foram adotadas algumas medidas e realizadas algumas alterações no analisador léxico para que o mesmo atendesse as especificações deste segundo trabalho.

Objetivo

Desenvolvimento de um gerador de analisador sintático utilizando a ferramenta GNU Bison em conjunto com a ferramenta de análise léxica Flex.

Linguagem

A linguagem L aceita pelo analisador sintático desenvolvido será especificada pelas regras a seguir.

* Comandos de Início e Término de Programa

Para explicitar o comando de início e termino de programa, foram definidos os comandos início() e fim() para esta função.

Para início do programa, deverá ser executado o comando início(), o qual indica o início do comando. De forma semelhante, para o término do programa, deverá ser chamado o comando fim().

A regra sintática responsável por esta expressão foi definida da seguinte forma:

Inicio:

INICIO corpo\_programa FIM

Na qual corpo\_programa indica os demais comandos que devem ser aceitos pela linguagem.

* Declarações de Variáveis

Para as funções de declarações de variáveis, a linguagem aceita expressões da seguinte forma:

criacao\_variaveis:

VARIAVEL lista\_variaveis

Na qual o token VARIAVEL indica algum tipo primitivo definido na linguagem, tal como, inteiro, real, caractere, logico. O token lista\_variaveis indica a lista de variáveis criadas, podendo ser uma única varivel ou então uma lista de variáveis.

* Comando de Atribuição

Para os comandos de atribuição, foi considerado a atribuição de valores constantes, bem como de expressões matemáticas. Para tanto, a regra de formação do comando de atribuição segue a seguinte regra:

Comando\_atribuicao:

|IDENTIFICADOR ATRIBUICAO INTEIRO INSTRUCAO

|IDENTIFICADOR ATRIBUICAO lista\_comandos\_matematicos INSTRUCAO

No qual IDENTIFICADOR representa a variável na qual será armazenada o valor atribuído, lista\_comandos\_matematicos representa as expressões matemáticas válidas.

* Comando Escolha

Para os comandos de escolha, foi utilizado a seguinte regra:

Comando\_escolha:

|SE ABRE\_CHAVE instrucao\_logica FECHA\_CHAVE ABRE\_COLCHETE comandos FECHA\_COLCHETE

No qual instrução lógica representa a expressão lógica do comando de escolha e comandos representa a lista de comandos contidos dentro da instrução.

* Comando de Repetição

Para a instrução de repetição, foi implementado o comando do laço enquanto (while), que obedece a seguinte expressão:

Comando\_repeticao:

|ENQUANTO ABRE\_CHAVE instrucao\_logica FECHA\_CHAVE ABRE\_COLCHETE comandos FECHA\_COLCHETE

No qual instrução lógica representa a instrução lógica dentro do comando enquanto e a instrução comandos representa a lista de comandos contidos dentro do laco de repetição.

* Chamada de Funções

Para os comandos de chamada de função, a linguagem segue a seguinte regra:

Chamada\_funcao:

|INICIO\_FUNCAO IDENTIFICADOR ABRE\_CHAVE parâmetros FECHA\_CHAVE INSTRUCAO

No qual parâmetros consiste nos parâmetros que a função recebe e INICIO\_FUNCAO representa o comando que inicia uma chamada de função, definido no programa como “função”.

Conteúdo

Segue uma breve descrição dos arquivos enviados:

Parser.l – código do gerador de analisador léxico (Flex)

Parser.y – código do gerador de analisador sintático (Bison)

sample.txt – exemplo de código aceito pela linguagem

Para compilar e executar

Execute a seguinte sequência de comandos:

- Criação do arquivo de análiser léxica:

flex -o Scanner.c Scanner.l

- Criação do arquivo de análise sintática:

bison -d Parser.y

- Criação do analisador

gcc Scanner.c Parser.tab.c -o analisador -lfl

Saída do analisador

O analisador acusará erros de sintaxe como falta de parênteses, chaves, ponto e vírgula, declarações incompletas, etc.

Caso não exista erros sintáticos a análise será concluída com sucesso.