COMPILADORES

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.

Por exemplo, a expressão regular que define o grupo de palavras reservadas no analisador léxico do trabalho 1, indicado por:

reservada "se"|"senao"|"para"|"enquanto"|"int"|"real"|"string"|"main"|"void"

Foi desmembrada em expressões individuais para a cada palavra reservado pudesse ser associado um token correspondente.

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 é 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

Execução

Para compilar e executar, deve-se executar a seguinte sequência de comandos (terminal linux):

- 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
```

- Execução do analisador considerando um arquivo de entrada:

```
$ ./analisador < sample.txt
```

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.