Disciplina: Construção de Compiladores

Projeto: Desenvolvimento de um Compilador

Parte II: Analisador Sintático (PARSER)

Sílvio Bandeira
Professor Adjunto

Implemente um parser descendente preditivo recursivo (com procedimentos recursivos), para a linguagem fonte descrita abaixo.

Observação 1: o arquivo a ser compilado será passado ao seu compilador **via argumento da linha de comando**

Observação 2: Imprimir **apenas mensagens de erro.**

Observação 3: A mensagem deve ser **clara e específica** de erro, sempre que for o caso, **e em qualquer fase do compilador**. Formato: "ERRO na linha *n*, coluna *m*, ultimo token lido *t*: *mensagem específica do erro"*

Observação 4: Após o fechamento do bloco do programa (main) não pode haver mais tokens, ou seja, o proximo retorno do scanner deve ser fim_de_arquivo.

1. Introdução

A linguagem tem uma estrutura de blocos tipo C. Sera descrita como uma GLC e com o auxílio da notação EBNF e expressões regulares.

2. Declaracões

Não teremos declaracões de procedimentos nem funções, apenas de variáveis. As declarações devem ser agrupadas no início do bloco, e devem aparecer numa sequência bem definida de modo a facilitar a compilação.

As variáveis podem ser do tipo *int*, *float* ou *char*, e as declarações devem ter o seguinte formato:

```
<decl_var> ::= <tipo> <id> {,<id>}* ";"
```

```
<tipo> ::= int | float | char
```

3. Expressões

Em geral, uma expressão é uma arvore de valores. Em sua forma mais simples, ela é um único valor de um tipo primitivo.

As produções para expressões obedecem à seguinte ordem de precedência:

1. *, /
2. +, -
3. ==, !=, <, >, <=, >=

O aluno deve modificar as produções de modo a eliminar a recursão à esquerda

OBS: Expressões apenas com os operadores *, /, +, - são expressões aritméticas. Expressões com os operadores de comparação (==, !>, <, ...) são expressões relacionais. Não podemos ter mais de um operador relacional em um expressão. Podemos ter expressões aritméticas de qualquer lado de um operador relacional. Mas, não podemos ter expressões relacionais em comandos de atribuição.

4. Programa e Comandos

Um programa é um bloco (como em C). Podemos ter blocos dentro de blocos. Dentro de um bloco as declaracões devem preceder os comandos.
O significado de if, if-else, while e do-while é como na linguagem C padrão.

5. Sintaxe

```
<programa> ::= int main"("")" <bloco>

<bloco> ::= "{" {<decl_var>}* {<comando>}* "}"

<comando> ::= <comando_básico> | <iteração> | if
"("<expr_relacional>")" <comando> {else <comando>}?

<comando_básico> ::= <atribuição> | <bloco>

<iteração> ::= while "("<expr_relacional>")" <comando> | do <comando>
while "("<expr_relacional>")";"
```

<atribuição> ::= <id> "=" <expr_arit> ";"

<expr_relacional> ::= <expr_arit> <op_relacional> <expr_arit>

<expr_arit> ::= <expr_arit> "+" <termo> | <expr_arit> "-" <termo> | <termo>

<termo> ::= <termo> "*" <fator> | <termo> “/” <fator> | <fator>

<fator> ::= “(“ <expr_arit> “)” | <id> | <real> | <inteiro> | <char>

OBS: É preciso alterar Expressão Aritmética(expr_arit), pois do jeito que ela está é recursiva a esquerda, então é preciso fazer uma alteração na gramatica para tornar ela recursiva a direita, ficando assim:

<expr_arit> ::= <termo><expr_arit2>

<expr_arit2> ::= "+" <termo><expr_arit2> | "-" <termo><expr_arit2> | VAZIO

Nota: os símbolos abre e fecha chaves, quando entre aspas, são terminais

6. Tabela de Símbolos

Para as variáveis, sugere-se que a tabela de símbolos seja uma lista encadeada onde os nós serão registros com os atributos das variáveis: lexema e tipo. O aluno pode modificar a tabela se encontrar utilidade para outro tipo de atributo ou se achar necessário incluir constantes com seus tipos e valores.

Como em toda lista encadeada, precisamos de um nó que aponta para a "cabeça" da lista. Chamemos este nó de "tabela". Na ativação de um bloco, guarde o conteúdo de "tabela" e adicione as novas variáveis no inicio da tabela de símbolos. Na desativação, restaure o valor de "tabela", eliminando assim todas as variáveis declaradas nesse bloco. Lembre de desalocar todos os nós com as variáveis do bloco sendo desativado. A busca a partir de "tabela" sempre encontrará o identificador mais recentemente declarado, por isso as variáveis devem ser incluídas no início da tabela de símbolos.