Analisador Sintático Descendente Recursivo

O objetivo do trabalho é implementar um Analisador Sintático Descendente Recursivo (ASDR) para linguagem Portugol, baseada no Pascal com palavras reservadas em português. O analisador léxico implementado anteriormente, deverá ser adaptado conforme a necessidade do analisador sintático, a interação entre o analisador léxico e o analisador sintática se dará por meio da função **consome()** vista em sala, ou seja, somente a função **consome()** fará chamadas ao analisador léxico. Caso o aluno não tenha implementado o analisador léxico isso deverá ser feito em conjunto com o ASDR.

Na implementação do ASDR quando for detectado um **erro sintático** ou **léxico**, o ASDR deve-se emitir uma mensagem de erro explicativa e terminar a execução do programa. A mensagem explicativa deve informar a linha do erro, o tipo do erro (léxico ou sintático) e caso seja um erro sintático, deve-se informar qual era o átomo esperado e qual foi encontrado pelo ASDR.

O ASDR deverá iniciar a construção da **Tabela de Símbolos**, assim todo identificador definido em uma seção de declaração é classificado como variável (local ou global), parâmetros, procedimento ou função, e inserido na Tabela de Símbolos. Note que as seções de declaração para identificadores são realizadas nas seguintes produções:

- *declaracao_de_variáveis*
- *declaracao_de_procedimento*
- *declaracao_de_função*

Na inserção de um identificador na Tabela de Símbolos deve-se garantir que o identificador é **único** no escopo corrente, caso não seja único é gerado um erro semântico com a mensagem “identificador já declarado” e termina a execução do programa. Uma **variável** pode ter escopo **global**, quando declaradas no início do programa, ou **local,** quando declaradas dentro de procedimentos e funções. Os **procedimentos** e **funções** tem escopo **global** e os **parâmetros** (declaradas nos procedimentos e funções) são considerados variáveis locais, ou seja, tem escopo **local**. A seguir são apresentados os atributos gerais e específicos para cada um dos tipos de identificadores declarados.

1) Atributos gerais:
   a) sequência de caracteres que gerou o átomo IDENTIFICADOR;
   b) A categoria do atributo (VARIÁVEL GLOBAL, VARIÁVEL LOCAL, PROCEDIMENTO, FUNÇÃO ou PARÂMETRO);
   c) Atributos específicos para **Variáveis** locais e globais:
      i) Tipo da variável definido na produção *tipo*
      ii) Ordem na declaração
   d) Atributos específicos para **Parâmetros**:
      i) Tipo do parâmetro definido na produção *tipo_simples*
      ii) Ordem na declaração
      iii) Passagem (Valor ou referência).
   e) Atributos específicos para **Procedimentos**:
      i) Lista de parâmetros; e
      ii) Lista de variáveis locais;
   f) Atributos específicos para **Funções**:
      i) Tipo de retorno da função;
      ii) Lista de parâmetros; e
      iii) Lista de variáveis locais;

Para armazenar e recuperar os identificadores de forma eficiente, a Tabela de Símbolos deve ser implementada utilizando uma Tabela de Dispersão (*Hash*) como, conforme apresentado no livro *"Compiladores, Princípios, Técnicas e Ferramentas"* dos Autores AHO, SETHI e ULLMAN, página 188.

```
/*
função hashpjw: implementa a dispersão padrão para um hash, usa um numero primo
como tamanho do vetor para minimizar colisões
*/
#define PRIME_NUMER 211
#define EOS '\x0'

int hashpjw( char * s )
{
    char* p;
    unsigned h = 0, g;
    for ( p = s; *p != EOS; p = p + 1 ){
      h = ( h << 4 ) + (*p);
      if ( g = h&0xf0000000 ){
        h = h ^ ( g >> 24 );
        h = h ^ g;
      }
    }
    return h % PRIME_NUMBER;
}
```

Por questão de simplificação o ASDR só armazenará na Tabela de Dispersão dos identificadores com escopo global, os identificadores locais (parâmetros ou variáveis locais) serão armazenados em lista de atributos das funções e/ou procedimentos onde foram declaradas. No final da análise o ASDR deverá listar todos os identificadores armazenados na Tabela de Dispersão, apresentando o identificador, o seu tipo e sua entrada (índice) na Tabela de Dispersão, caso seja uma função e/ou procedimento deverá ser apresentado a sua lista de variáveis locais e parâmetros.

Descrição da Linguagem Pascal

Por conveniência, introduziremos mais uma notação [α] que é equivalente a $\alpha|\lambda$, ou seja, indicará que a cadeia α é opcional. Os *não-terminais* da gramática são nomes em *itálico* e os símbolos **terminais** estão em **negrito** ou entre aspas (Ex: **";"**).

programa ::= **algoritmo identificador** *bloco*

bloco ::= [*declaracao_de_variaveis*]
{ *declaracao_de_função* }
inicio *comando* **fim**

declaracao_de_variaveis ::= **variaveis** { **identificador** *tipos* **";"**}

tipos ::= *tipo_simples* | *tipo_composto*

tipo_simples ::= **inteiro** | **real** | **caractere** | **logico**

tipo_composto ::= **vetor "["**constante_inteiro **".."**constante_inteiro **"]" de** *tipo_simples*

declaracao_de_rotina ::= *declaração_de_funcao* | *declaracao_de_procedimento*

declaracao_de_função ::= **funcao identificador "("** *parametros_formais* **")" ":"** *tipo_simples*
[*declaracao_de_variaveis*] **inicio** *comando* **fim funcao**

declaracao_de_procedimento ::= **procedimento identificador "("** *parametros_formais* **")"**

[*declaracao_de_variaveis*] **inicio** *comando* **fim procedimento**

parametros_formais ::= [**ref**] **identificador** *tipo_simples*
{ **";"** [**ref**] **identificador** *tipo_simples* } | **nada**

comando ::= { *comando_atribuicao* |
chamada_procedimento |
comando_se |
comando_enquanto }

comando_atribuicao ::= **identificador ":="** *expressao* **";"**

chamada_procedimento ::= **identificador "("** *lista_expressao* **")" ";"**

comando_se ::= **se** *expressao* **entao** *comando* [**senao** *comando*] **fim se**

comando_enquanto ::= **enquanto** *expressao* **faca** *comando* **fim enquanto**

lista_expressao ::= *expressao* {**";"** *expressao*} | **nada**

expressao ::= *expressao_simples* [**op_relacional** *expressao_simples*]

expressao_simples ::= [**"+"** | **"–"**] *termo* { *operador_adicao termo* }
operador_adicao ::= **"+"** | **"–"** | **mod** | **ou**

termo ::= *fator* { *operador_multiplicacao fator* }
operador_multiplicacao ::= **"*"** | **div** | **e**

fator ::= **identificador** [**"("** *lista_expressao* **")"**]
| **constante_inteiro**
| **constante_real**
| **constante_frase**
| **constante_caractere**
| **verdadeiro**
| **falso**
| **nao** *fator*
| **"("** *expressao* **")"**

Importante:

O trabalho deve ser implementado na linguagem C/C++ e bem documentado. Você deve entregar o seu programa fonte bem documento através da ferramenta de ensino a distância *blackboard*.

Este trabalho pode ser feito **em dupla**, e evidente você pode "**discutir**" o problema dado com seus colegas, inclusive as "**dicas**" para chegar às soluções, mas você deve ser responsável pela solução final e pelo desenvolvimento do seu programa.