UNIVERSIDADE CATÓLICA DOM BOSCO – UCDB

Engenharia de Computação – 7º semestre

Primeiro Trabalho de Compiladores I

Prof. Marcelo Silva Cintra

Primeiro Semestre de 2007

1 Analisador Léxico

O objetivo do trabalho é construir uma função (AnaLex) que realiza a análise léxica de um programa escrito na linguagem de programação PASCAL. A função deve ler o programa PASCAL de um arquivo e identificar os átomos da linguagem. Quando necessário deve ser retornado um atributo que diferencie átomos com o mesmo padrão de formação. As definições regulares para os átomos serão dadas abaixo. O arquivo de entrada deve ser fornecido, sem extensão, para AnaLex através da linha de comando, a rotina deve acrescentar a extensão .pas.

Exemplo: Se o programa a ser analisado possuir o nome **teste.pas,** a execução deve ser:

AnaLex teste

A função AnaLex deve fazer um controle da numeração das linhas do programa fonte e também fazer a eliminação de comentários. Para cada átomo reconhecido, deve-se mostrar, no mínimo, as seguintes informações de cada átomo:

{Número da linha do átomo, Átomo, Atributo (quando o átomo possuir atributo)}

A saída poderá ser mostrada na tela ou gerando-se um arquivo com os átomos reconhecidos.

É **extremamente importante** que seu programa possua **uma função** (analex ou proximoAtomo) que retorne um átomo sempre que for chamada. Esta função não deve escrever o átomo na tela, mas sim produzir um átomo para uma outra função (a *main* por exemplo) que se encarregará de fazer a saída desejada.

1.1 Definição dos Átomos

As palavras chaves da linguagem são consideradas palavras reservadas, ou seja, não podem ser utilizadas como identificadores. Para simplificar a rotina de análise léxica, o reconhecimento de palavras chaves será feito com base na definição regular de identificadores, sendo que a determinação do átomo associado à palavra chave é feita por uma busca na tabela de palavras reservadas. A tabela de palavras reservadas pode ser implementada utilizando-se um vetor de strings, **já ordenado**, contendo todas as palavras reservadas da linguagem PASCAL. Toda vez que um identificador for encontrado, deverá ser feita uma busca binária (θ (lg n)) no vetor de palavras reservadas. Uma busca com sucesso na tabela de palavras reservadas indica que uma palavra chave foi reconhecida e então seu átomo deverá ser retornado. Caso contrário, um identificador foi reconhecido e o mesmo deverá ser inserido na tabela de símbolos, caso ainda não tenha sido inserido previamente.

Palavras Chaves = { AND, ARRAY, BEGIN, CASE, CONST, DIV, DO, DOWNTO, ELSE, END, FOR, FUNCTION, GOTO, IF, LABEL, MOD, NOT, OF, OR, POINTER, PROCEDURE, PROGRAM, RECORD, REPEAT, SET, THEN, TO, TYPE, UNTIL, VAR, WHILE, WITH }

Obs: Pascal não faz distinção entre identificadores grafados com letras maiúsculas ou minúsculas. ARRAY é equivalente à array. Sugere-se que todas as letras sejam convertidas para uma das duas formas antes da busca.

Definições Regulares para Átomos sem Atributos

assign_op → :=	**dotdot** → ..	**dot** → .	**colon** → :
semicolon → ;	**comma** → ,	**lb** → [	**rb** ->]
lp → (	**rp** →)	**equal** → =	**le** → <=
ge → >=	**ne** → <>	**gt** → >	**lt** → <
divide → /	**minus** → -	**times** → *	**plus** → +
simb_pointer → ^	**ender** → @		

Dica: Para os **operadores relacionais** (<, <=, >, >=, =, <>) poderá ser criado um único átomo, RELOP, e o atributo associado a esse átomo especificará qual dos operadores relacionais foi realmente reconhecido (LE para <, GE para >, ...). Exemplo <> deve ser retornado (RELOP, NE).

Definições Regulares para Átomos com atributos

letter → [_A-Za-z]

digit → [0-9]

identifier → **letter** (**letter** | **digit**)*

O atributo para *identifier* é um apontador para uma entrada na tabela de símbolos. A tabela de símbolos deve ser implementada como uma tabela de dispersão (*hash table*) com resolução de colisões por encadeamento exterior[1] e com a função de dispersão dada em C abaixo[2]. Quando um *identifier* é reconhecido, deve-se inicialmente fazer uma busca binária no vetor de palavras reservadas para verificar se *identifier* é uma palavra reservada. Se for encontrado, deve-se retornar o átomo associado com a palavra chave. Caso a busca seja sem sucesso, uma nova busca na tabela de símbolos deve ser realizada. Caso o identificador já esteja na tabela de símbolos, o ponteiro para a entrada na tabela de símbolos do identificador deve ser retornado como atributo do identificador. Caso não seja encontrado, o identificador deve ser inserido na tabela de símbolos e então retornar o ponteiro da nova entrada como atributo. Para efeito de visualização neste trabalho, deve ser mostrado, como atributo, na saída, o nome do identificador.

```
#define PRIME 211
#define EOS   '\0'
int hashpjw(char *s)
{
        char            *p;
        unsigned long   h = 0, g;
        for(p = s; *p != EOS; p++)
        {
                h = (h << 4) + (*p);
                g = h & 0xf0000000;
                if(g){
                        h ^= g >> 24;
                        h ^= g;
                }
        }
        return (h % PRIME);
}
```

Função de dispersão

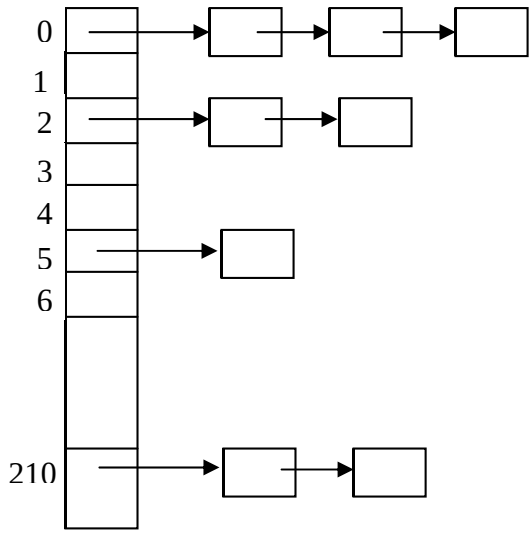

Tabela de Dispersão aberta

[1] Szwarcfiter, Jayme Luiz e Markenzon, Lilian - **Estruturas de Dados e seus Algoritmos**, 1994, 2. ed., Rio de Janeiro: LTC Editora.

[2] Aho, A. V., Sethi, R., Ullman, J. D. **Compiladores: Princípios, Técnicas e Ferramentas**, Rio de Janeiro: LTC, 1995. página 185-188

digits → **digit**$^{+}$

optional_fraction → (.**digits**)?

optional_expoent → ((E | e)(+ | -)? **digits**) ?

num → **digits optional_fraction optional_expoent**

O atributo para **num** é um apontador para a cadeia de caracteres que forma o número.

char1 → { qualquer caractere do alfabeto menos os caracteres de retorno de carro e avanço de linha }

string → '**char1**'

O atributo para **string** é um apontador para a cadeia de caracteres.

character → { qualquer caractere do alfabeto }

comment → **(* character*** ***)** | **{ char1* }**

Para **comment** não deve ser retornado nenhum átomo, mas o controle de linhas deve ser mantido.

Delimitadores

delim → blank | tab | newline

ws → **delim**$^{+}$

Todos os átomos são separados por delimitadores dados pela definição regular **ws**. A definição regular **ws** não está associada com átomos, é simplesmente uma definição auxiliar.

O AnaLex deverá fazer buferização do arquivo de entrada, tal como descrito no livro do Aho, no capítulo 3, seção 3.2. Utilize um par de buffers como mostrado no livro.

Obs: O programa deve ser bem documentado, escrito usando alguma convenção de código. A avaliação poderá ser feita mediante entrevista com os integrantes do grupo. O grupo é de no máximo 2 alunos. Qualquer tentativa de fraude será punida com a nota zero.

```
typedef  enum {
        AND, ARRAY, BEGIN, CASE, CONST, DIV, DO, … }  tipoAtomo;
```

Desta forma, quando um átomo for reconhecido, o seu respectivo código será retornado.

Como vimos, para alguns átomos será necessário capturar informações adicionais que formarão os atributos do átomo. Freqüentemente, os atributos e o átomo são representados por um único tipo estruturado, ou um registro do átomo. Esse registro poderia ser declarado em C como:

```
typedef struct {
        tipoAtomo atomo;
        union {
            char *stringVal;
             int numVal;
         }attr;
    }
```

Bom Trabalho!