INSTITUTO DE CIÊNCIAS MATEMÁTICAS E DE COMPUTAÇÃO
UNIVERSIDADE DE SÃO PAULO

Introdução à Compilação: Trabalho 1

Professor: Thiago A. Pardo

Disciplina: SSC-0206 – Introdução a Compilação
Curso: Bacharelado em Ciências da Computação
Alunos:

Fábio Henrique Gomes Sikansi	NºUSP: 6792461
Matheus Edson Ramos	NºUSP: 6793177
Vanessa Queiroz Marinho	NºUSP: 6793201

Data: 09/04/2012

São Carlos
2012

Decisões de projeto:

1) Tokens

- Para os números inteiros é retornado o token 'num_integer'.

  Ex.: {10 - num_integer}

- Para os números reais é retornado o token 'num_real'.

  Ex.: {10.99 - num_real}

- Para os identificadores, verifica-se se são uma palavra reservada, o token retornado é a própria palavra reservada. Se não forem, é retornado o token 'id'.

  Ex.: {program - program}

  {minha_variavel - id}

- Para os símbolos de {"=",">=",">","<>","<=","<"} é retornado o token 'operador_comp', essa decisão foi tomada devido a utilização semelhante destes símbolos.

  Ex.: {= - operador_comp}

- Para os símbolos de {"+","-","*","/"} é retornado o token 'operador_mat', essa decisão também foi tomada devido a utilização semelhante destes símbolos.

  Ex.: {+ - operador_mat}

- Por fim, para os símbolos de {";" , "," , ":=", ":", ".", "(", ")"}, o token a ser retornado é o próprio símbolo.

  Ex.: {; - ;}

2) Comentários

O comentário é qualquer sequência de caracteres que se encontram dentro de chaves {}. É possível a utilização de comentários de uma ou várias linhas, pois ao encontrar um '{' consome-se todos os caracteres, inclusive *newline*, até achar o '}'. Essa decisão foi tomada com o objetivo de tornar a leitura do código melhor, pois é mais simples ler um comentário quebrado em várias linhas, do que um comentário grande em uma única linha.

3) Erros

Os seguintes erros são tratados pelo analisador léxico:

- Comentários não fechados.

- Símbolos não pertencentes à linguagem.
  Ex.: @
- Identificadores com tamanho superior ao máximo permitido, que é de 32 caracteres.
- Números com tamanho superior ao máximo permitido. Para os inteiros o tamanho máximo é de 10 dígitos, e para os números reais a parte inteira deve ter no máximo 5 dígitos e a parte real no máximo 6 dígitos.
- Números mal formados.
  Ex.: 12a5, 12.4b4, 12@67
- Identificadores mal formados.
  Ex.: minha_variavel@@@

4) Tabela de Palavras Reservadas

Para implementar a tabela de palavras reservadas foi criado um vetor de strings com 16 posições. Neste vetor as palavras reservadas encontram-se em ordem alfabética. A ordem é a seguinte:

begin

const

else

end

if

integer

procedure

program

readln

real

repeat

then

until

var

while

writeln

As palavras reservadas foram salvas no vetor em ordem alfabética para realizarmos busca binária neste vetor. Portanto, durante o reconhecimento de um identificador verifica-se neste vetor se ele é uma palavra reservada ou não, se for uma palavra reservada a sua posição no vetor é retornada, caso contrário -1 é retornado.

Especificação do Analisador Léxico

Analisando os principais pontos do código:

1) IDENT [a-zA-Z][a-zA-Z0-9]*

Expressão regular responsável pela formação dos identificadores, que são formados por uma letra maiúscula ou minúscula, seguidos de qualquer sequência de letras e dígitos, o tamanho máximo de um identificador é de 32 caracteres.

2) DIGIT [0-9]

Expressão regular que representa um número qualquer de 0 a 9. Esta expressão regular é combinada para identificar os números:

{DIGIT}+ : reconhece os números inteiros, o tamanho máximo aceito é de 10 dígitos.

Ex.: 9999999999

{DIGIT}+"."{DIGIT}+ : reconhece os números reais, sendo que a parte inteira comporta até 5 dígitos, e a parte decimal até 6.

Ex.: 99999.999999

3) Símbolos que são reconhecidos separadamente, com regras simples utilizando expressões regulares:

"="|">="|">"|"<>"|"<="|"<"

"+"|"-"|"*"|"/"

";"
","
":="
":"
"\."
"\("
"\)"

4) \n num_lines++;

Neste trecho, para cada ocorrência do caracter *newline*, o contador do número de linhas é incrementado, esta informação é utilizada para especificar a linha nas mensagens de erro.

5) [\t]+

Expressão regular que consome os espaços em branco e as tabulações.

6) "{" [código em C...]

Ao encontrar o símbolo de início de comentário, todos os caracteres seguintes são consumidos até encontrar o símbolo de fim de comentário "}". Caso o comentário não seja fechado, o símbolo de fim de arquivo é encontrado retornando um erro.

7) ({IDENT}[^0-9\n\t ;{\/*\+\-<>"<=""">=""<>"\)=\.\:\,\(":="]*)+

Essa expressão regular identifica um identificador mal formado. Sua estrutura é a de uma cadeia de letras e dígitos seguida por um ou mais caracteres não esperados. Todos os caracteres que se encontram entre os colchetes fazem parte do conjunto seguidor da cadeia de letras e dígitos, colocando o caractere '^' antes desses símbolos estamos dizendo que qualquer símbolo que não seja os seguidores descritos é um símbolo não esperado, formando assim um identificador mal formado.

8) ({DIGIT}+[^0-9\n;{\/*\+\-<>"<=""">=""<>"\)=\.]+)+{DIGIT}*

Essa expressão regular identifica um número inteiro mal formado. Sua estrutura é a de uma cadeia de dígitos seguida por um possível conjunto de caracteres não esperados e dígitos. Todos os caracteres que se encontram entre os colchetes fazem parte do conjunto seguidor da cadeia de dígitos, como na regra 7, o caractere '^' antes da classe desses símbolos informa que qualquer símbolo que não pertença aos seguidores é um símbolo não esperado, tendo assim um número inteiro mal formado.

9) ({DIGIT}+[^0-9\n;{\/*\+\-<>"<=""">=""<>"\)=]*)+[\.]({DIGIT}*[^0-9\n;{\/*\+\-<>"<=""">=""<>"\)=\.]*)*

Essa expressão regular identifica um número real mal formado. Sua estrutura é a de uma cadeia de dígitos, seguida por um conjunto misto de caracteres não esperados e dígitos, seguido, por sua vez, pelo símbolo ".", separando a parte inteira da parte decimal. Em seguida, a cadeia é composta por um conjunto possivelmente misto de dígitos e caracteres não esperados. Todos os caracteres que se encontram entre os colchetes fazem parte do conjunto seguidor da cadeia de dígitos, de maneira semelhante às regras 7 e 8.

10) . printf("error-Error at line %d: Unrecognized character : %s\n",num_lines, yytext);

Por fim, caso qualquer outro símbolo que não foi reconhecido antes seja encontrado, é retornado um erro, pois este símbolo não pertence à linguagem.

Compilar e Executar o programa

Para executar o trabalho é necessário ter o programa flex instalado no computador.

1º) Compile o arquivo lex da seguinte maneira:

flex lalg.lex

2º) Após executar a linha acima, o arquivo lex.yy.c será gerado, deve-se compilar o arquivo com a seguinte linha:

gcc lex.yy.c -o out -lfl

3º) Para executar o arquivo gerado no passo anterior, copie o código do exemplo de execução que encontra-se na seção seguinte e salve no arquivo entrada.txt. Redirecione este arquivo para a entrada do arquivo 'out', da seguinte maneira:

./out < entrada.txt

Obs: O trabalho foi desenvolvido em ambiente Linux.

Exemplo de Execução

```
program lalala

var
i, j, k: integer;
r1,r2,r3:real;

begin

{Teste de identificadores mal formados}
a123$##$as := 0;
a#$$# := 1233;
fvd#$ $343:= 112;

i:=0;
j := 1

{Teste de numeros inteiros mal formados}
i := 121#121;
k := 1234567$#$890##aaa1213sss12a;

{Teste com numeros reais mal formados}
r1 := 12#$4.1234;
r2 := 3.141##516;
r3:= 12.#####;

{Teste para identificador com tamanho maior que o permitido}
Aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa

{Teste para numero inteiro com tamanho maior que o permitido}
```

```
99999999999999

{Teste para numero real com tamanho maior que o permitido}
9999999.99999999

{Teste para caracter não pertencente a linguagem}
@&

if (j = i)
 writeln(i);
else if (j >= i);
 writeln(j);
else
 readln(k);
{Temos um comentario
   de multiplas linhas
     aqui}

end.

{Teste para um comentário não fechado ...
```

Bibliografia

Aho, A.V.; Ullman, J.D.; Sethi, R. (1995). Compiladores: Princípios, Técnicas e Ferramentas. Editora LTC.

Manual do Flex.
Disponível em: http://flex.sourceforge.net/manual/

Material de aula do Professor Thiago Pardo.
Disponível em: http://wiki.icmc.usp.br/index.php/Material206t%28taspardo%29