- UNIVERSIDADE DO VALE DO RIO DOS SINOS
CIÊNCIAS EXATAS E TECNOLÓGICAS – Curso de Informática

COMPILADORES I

Disciplina: Compiladores I **E-mail: f*osorio@ unisinos.br***
Professor responsável: *Fernando Santos Osório* **Web: *http://inf.unisinos.br/~osorio/compil.html***
Semestre: 2006/2 **Horário: *Tutoria***

GERADORES DE ANALISADORES LÉXICOS – LEX / FLEX

O **LEX / FLEX** servem para gerar automaticamente programas (usualmente em "C") fazendo a leitura de uma entrada, de modo a varrer um texto e/ou programa ("scanners") a fim de obter uma seqüência de unidades léxicas ("tokens"). Os tokens gerados pelos programas criados pelo LEX/FLEX serão usualmente processados posteriormente por um programa que realizará a analise sintática.

Lex => Gerador de analisadores léxicos (UNIX – Ex.: Lex AT&T, Berkeley BSD)
Flex => Gerador de analisadores léxicos (LINUX / Windows-DOS – GNU Lex)

Entrada: Arquivo de descrição do analisador léxico
Saída: Programa na linguagem "C" que realiza a análise léxica (default: lexyy.c)

Outros geradores de analisadores:
TPly – TP Lex / Yacc => Gera um programa em PASCAL (scanner em Pascal)
JavaCC => Para linguagem Java
Flex++ ou Flexx => Para linguagem C++ (orientado a objetos)

Usando o FLEX:

1. FLEX -o<arq_saida.c> <arq_def>.l
   *.l => Arquivos que contêm as definições das unidades léxicas a serem identificadas
   Contém um conjunto de especificações de ***expressões regulares*** que serão usadas para reconhecer os tokens.
   *.c => Arquivo do programa "C" que implementa o analisador léxico especificado.
   Principais opções do FLEX:
   -i => Case Insensitive (ignora diferença entre maiúscula/minúsculas)
   --version => Exibe a versão atual do programa flex em uso
   -+ => Geração de código de saída em C++
2. GCC <arq_saida.c> -o <arq_executavel> -lfl
   Observações importantes sobre a geração do programa de análise léxica:
   - É necessário linkar (-l) uma bibliotec ("lib") do analisador léxico na compilação do código gerado. O FLEX usa a lib "fl" e o LEX usa a lib "l".
   - O programa gerado pode ser executado, onde usualmente a entrada do texto a ser analisado é feita pela "stdin" (teclado).
3. Executar o programa gerado...

EXPRESSÕES REGULARES – Usada pelo LEX / FLEX:

[0-9] => Reconhece um dígito
[a-zA-Z] => Reconhece uma letra (comum = sem acentos)
[\ \t\n] => Reconhece um espaço em branco ou um tab ou uma nova linha
xxxxx => Reconhece a seqüência de caracteres "xxxxx"

Símbolos especiais:		Exemplo:	
+	=> 1 ou mais ocorrências	[0-9]+	=> Um número
*	=> 0 (nenhuma) ou mais ocorrências	[0-9][0-9]*	=> Um número
?	=> 0 (nenhuma) ou apenas 1 ocorrência	-?[0-9]+	=> Um número com/sem sinal

\n => Reconhece a marca de fim de linha / nova linha
. => Aceita um caracter qualquer de entrada
xxx$ => Reconhece xxx se for seguido de um fim de linha
^xxx => Reconhece xxx se este estiver imediatamente após o início de uma linha
[^x] => Reconhece qualquer caracter menos "x"
[xyz] => Reconhece um dos caracteres "xyz" indicados
[a-z] => Reconhece um caracter pertencente ao intervalo de "a-z"
x{n} => Reconhece um número exato "n" de ocorrência de "x"
x{n,} => Reconhece a ocorrência de no mínimo "n" vezes de "x"
x{n,m}=> Reconhece a ocorrência de "x" entre no mínimo "n" e no máximo "m" vezes
xx|yy => Reconhece a ocorrência de "xx" ou de "yy"
(x|y) => Agrupa (sub)expressões regulares
"x" => Reconhece exatamente o caracter "x" (usado com caracteres especiais). Ex.: "+"

Exemplos de expressões regulares simples:

DIGITO	[0-9]	
LETRA	[a-zA-Z]	
ESPACO	[\ \t\n]	
INTEIRO	[0-9]+	
INTSIGNED	-?[0-9]+	
DECIMAL	[0-9]*\.[0-9]+	=> aceita .33 / não aceita números sem casas decimais
INTOUDEC	([0-9]+)|([0-9]*\.[0-9]+)	
IOUDSIGNED	-?(([0-9]+)|([0-9]*\.[0-9]+))	
NOMEVAR	[a-z][a-z0-9_]*	=> usando opção "case insensitive"...

Exercícios para treinamento:
Defina as expressões regulares capazes de reconhecer...

1) Nros. de Telefones no Brasil
2) Placas de Carros Brasileiros
3) ISBN de um livro
4) Endereços IP válidos
5) Prefixos de estações de rádio (e.g. 102.3 MHz)
6) Números romanos
7) Número de matrícula da Unisinos
8) Números reais (qualquer notação, incluindo científica)
9) Tags HTML (padrão)
10) URL de páginas Web
11) Palavras da Língua Portuguesa
12) Strings de um programa em linguagem "C"

TRABALHO PRÁTICO

LISTA DE EXERCÍCIOS 1

Implementar Analisadores LÉXICOS para reconhecer os elementos descritos abaixo, enviando os arquivos do Lex/Flex (*.l) para o professor por e-mail (fosorio@gmail.com) até a sétima semana do semestre.

1. Reconhecedor de:
   1.1. Nro. real com qualquer representação (científica, sinal, etc). Exemplos:
   1 ; 1.0 ; -1 ; -1.0 ; 1.99 ; 0.99 ; 1.0E05 ; 0.99E-05 ...
   1.2. Nros. Romanos: de 0 a 1999
   Incluindo os símbolos: I=1, V=5, X=10, L=50, C=100, D=500, M=1000
   1.3. Números de telefone válidos no Brasil, inclusive com operadora. Exemplos:
   55555555 ; 5555.5555 ; 5555-5555 ; (55) 55555555 ; 55-5555.5555 ; (55) (55) 5555-5555 ;
   +55 (55) 5555.5555 ; 55 55 5555-5555 ; (55 55) 5555-5555 ; (55-55) 5555-5555 ;
   55 XX 55 5555 5555 ; 55 (xx) 55 55555555 ; 0 (xx) 55 5555 5555 ; (+ 55) xx (55) 55555555 ;
   +55 (55) 5555 5555 ; e demais variações como estas.
   Usar google: <Tel Fax 55> ou <Tel Fax 55 xx> para obter exemplos como estes

2. Comando PCC – Pascal Command Count (inspirado no Word Count do Linux - visto em aula).
   Saída do programa: nro. total de caracteres, linhas, palavras e comandos básicos em Pascal encontrados em um programa fonte. Não considerar os comentários entre (* e *) e entre { e }, ignorando também os textos dentro de strings (entre apóstrofes: ‘ ‘). Os comandos básicos do Pascal são todas as ***keywords*** padrão da linguagem pascal.

6. Ocultador de Textos com uso e troca de "estados" (%s) do Lex
   Entrada: blá blá <HIDE>Texto oculto</HIDE> blá blá
   Saída : blá blá <HIDE>XXXXXXXXXXXX</HIDE> blá blá

7. Ocultador/Desocultador com criptografia
   Igual ao exercício acima, mas onde a transformação usa uma função de codificação/decodificação do texto por cripotografia. A criptografia é qualquer função que dado um código A gere um código B, que depois possa ser revertido, ou seja, dado o código B gere de volta o código A.
   Entrada: blá blá <CRIPTO>SEGREDO</CRIPTO> blá blá
   Saída : blá blá <CRIPTO>TFHSFEP</CRIPTO> blá blá

8. Gerador de TOKENS para uma linguagem como a descrita abaixo
   00 - DELIMITADOR , : ; .
   01 - OPERADOR_RELACIONAL < > = >= <= => =< <>
   02 – OPERADOR_ARITMETICO + - * / SQR SQRT SIN COS TAN
   03 - PALAVRA_RESERVADA Begin End For Do If Then Else Elseif Endif While Repeat Until
   04 - TIPO_DADO Integer Real Float Double Char String Boolean
   05 – FUNCAO _nome (palavras começando com underscore)
   06 – VARIÁVEL $nome (palavras começando com um “$”)
   07 – TEXTO “texto” (strings delimitadas entre aspas “”)
   08 – NUMERO 123 (seqüência de dígitos formando um número inteiro, positivo ou negativo)
   09 – NOME nome (palavras compostas por uma seqüência de letras)
   10 – COMENTÁRIO # texto # (texto delimindado pelo “#”)
   99 – INVALIDO demais elementos não presentes nesta descrição