Compiladores
Análise Léxica
Flex – Gerador de Analisador Léxico

Prof. Dr. Luiz Eduardo G. Martins
(adaptado por Profa Dra Ana Carolina Lorena)
UNIFESP

Flex – Gerador de Analisador Léxico

- O processo de construção de um analisador léxico pode ser automatizado
- Existem vários geradores de analisadores léxicos disponíveis gratuitamente, que ajudam muito no desenvolvimento de um compilador
- Um gerador muito conhecido é o *Flex (Fast Lex)*
  - Distribuído como parte do pacote de compilação *GNU* (produzido pela *Free Software Foundation*)

Flex – Gerador de Analisador Léxico

- Gera sistema de varredura a partir de expressões regulares
  - Implementação de AFD baseada em tabelas (veremos em aulas posteriores)

Flex – Gerador de Analisador Léxico

- Convenções de metacaracteres em *Lex*

Tabela 2.2 Convenções de metacaracteres em Lex

Padrão	Significado
`a`	caractere *a*
`"a"`	caractere *a*, mesmo se *a* for um metacaractere
`\a`	caractere *a* se *a* for um metacaractere **(pela interpretação ANSI –C)**
`a*`	zero ou mais repetições de *a*
`a+`	uma ou mais repetições de *a*
`a?`	um *a* opcional
`a|b`	*a* ou *b*
`(a)`	*a* propriamente dito
`[abc]`	qualquer caractere entre *a*, *b* e *c*
`[a-d]`	qualquer caractere entre *a*, *b*, *c* e *d*
`[^ab]`	qualquer caractere, exceto *a* ou *b*
`.`	qualquer caractere, exceto mudança de linha
`{xxx}`	a expressão regular representada pelo nome *xxx*

Flex – Gerador de Analisador Léxico

- Convenções de metacaracteres em *Lex*
  - Exemplo:

```
nat [0-9]+
signedNat  (+|-)?{nat}
```

Flex – Gerador de Analisador Léxico

- Formato do arquivo de entrada *Flex*
  - O arquivo de entrada é composto por três partes
    - Definições
    - Regras
    - Rotinas do usuário
  - As seções são separadas por dois sinais de porcentagem, que aparecem em linhas separadas, iniciando na primeira coluna

Flex – Gerador de Analisador Léxico

- Formato do arquivo de entrada *Lex*

```
%{                                                    DEFINIÇÕES
    qualquer código C a ser inserido externamente a
    qualquer função      opcional
%}
definições regulares       opcional
%%                                                    REGRAS
expressões regulares seguidas do código C a ser executado quando
houver casamento com a expressão regular correspondente

%%                                                    ROTINAS AUXILIARES
código C para rotinas auxiliares ativadas pela segunda seção, pode
também conter um programa principal        opcional
```

Flex – Gerador de Analisador Léxico

- Nomes internos utilizados por *Flex*

Tabela 2.3 Alguns nomes internos Lex

Nome Interno Lex	Significado/Utilização
`lex.yy.c or lexyy.c`	Arquivo de saída Lex
`yylex`	Rotina de varredura Lex
`yytext`	Cadeia casou com ação corrente
`yyin`	Entrada Lex (padrão: `stdin`)
`yyout`	Saída Lex (padrão: `stdout`)
`input`	Rotina de entrada com reservatório Lex
`ECHO`	Ação básica Lex (imprime `yytext` em `yyout`)

Funciona como getToken

- A documentação completa de *FLEX* está disponivel em http://flex.sourceforge.net/manual/

yytext costuma ser usada para guardar o lexema

Flex

- Exemplo:

```
/*
 * Description: Count the number of characters and the number of lines
 *              from standard input
 * */
%{
int num_lines = 0, num_chars = 0;
%}

%%
\n    ++num_lines; ++num_chars;
.     ++num_chars;
fim   return 0;
%%
main()
{
  yylex();
  printf("# of lines = %d, # of chars = %d\n", num_lines, num_chars);
}
```

Essas linhas (entre %{ e %}) são inseridas diretamente no código C produzido pelo Flex, fora de todos os procedimentos (num_lines e num_chars serão globais)

Código inserido no final
Ativa função yylex (procedimento que implementa o AFD)

Flex

- Exemplo de utilização do *Flex:*
  - Salvar o arquivo anterior como teste.l
  - Digitar no terminal, no diretório do arquivo:

    $ flex teste.l
  - Flex gerará um arquivo C, com o seguinte nome padrão: *lex.yy.c*
  - *lex.yy.c* implementa os procedimentos do AFD gerado a partir das ERs, mas ainda precisa ser compilado e ligado com a biblioteca *libfl*

    $ gcc –o teste lex.yy.c -lfl

Flex

- Exemplo de utilização do *Flex*
  - Podemos executar teste
    - Teste fará a leitura de caracteres de entrada via teclado (yyin = stdin)

Flex

- Exemplo:

```
/*
 * Description: Adiciona números de linhas a um texto
 * */
%{
#include <stdio.h>
int num_line = 1;
%}
line   .*\n
%%
{line} {printf("%5d %s", num_line++,yytext);}
%%
main()
{
  yylex();
  return 0;
}
```

Exercício: fazer os passos anteriores para esse novo exemplo, chamando-o de teste2

Flex

- Resolvendo ambiguidade:
  - Flex sempre casará a subcadeia mais longa com a regra
  - Se duas ou mais regras casarem com subcadeias de mesmo comprimento, Flex selecionará a regra citada antes
  - Se nenhuma regra casa, Flex copia o caractere na saída e segue em frente

Flex – Gerador de Analisador Léxico

- Exemplo de utilização do *Flex*
  - Reconhecimento de cadeias que atendam a especificação de números inteiros nos formatos
    - Decimal
    - Octal
    - Hexadecimal
    - Binário

- Exemplo de utilização do *Flex*

```
DIGIT [0-9]
%%
[1-9]{DIGIT}*       printf("DEC");
0[0-7]*         printf("OCT");
0x[0-9A-Fa-f]+          printf("HEX");
0b[01]+         printf("BIN");
<<EOF>>             return 0;
%%
int main(int argc, char *argv[])
{
 FILE *f_in;
 if (argc == 2)
 {
  if(f_in = fopen(argv[1],"r"))  yyin = f_in;
  else  perror(argv[0]);
 }
 else  yyin = stdin;
 yylex();
 return(0);
 }
```

- Arquivo de entrada (programa), a ser lido pelo gerador de analisador léxico (*Flex)*
- Este arquivo deve ter extensão .l
- No exemplo, o nome do arquivo é *unsint.l*

Flex – Gerador de Analisador Léxico

- Exemplo de utilização do *Flex*
  - Ao digitar:

    $ flex unsint.l
  - Flex gerará o analisador léxico, com o seguinte nome padrão *lex.yy.c*
  - *lex.yy.c* implementa os procedimentos do analisador léxico, mas ainda precisa ser compilado e ligado com a biblioteca *libfl*

    $ gcc –o aliss lex.yy.c -lfl

Flex – Gerador de Analisador Léxico

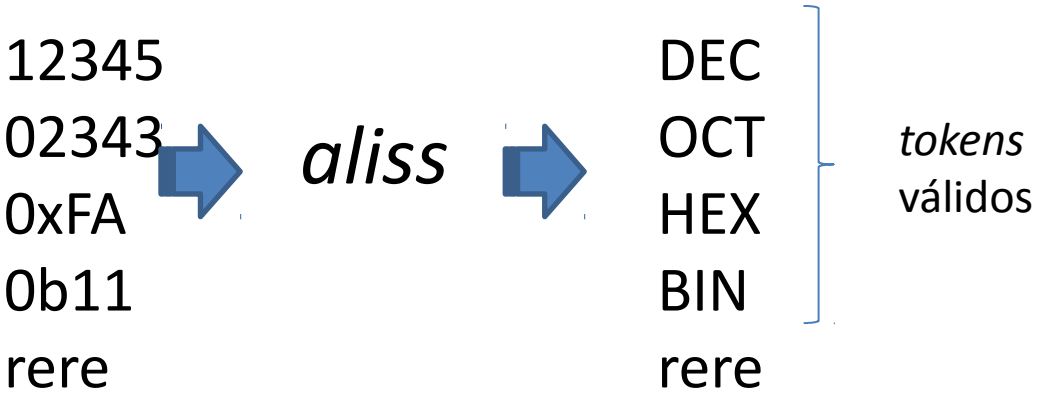
- Exemplo de utilização do *Flex*
  - Podemos executar *aliss* com ou sem um arquivo texto de entrada
    - Sem arquivo de entrada

      aliss fará a leitura de caracteres de entrada via teclado (yyin = stdin)
    - Com arquivo de entrada

      aliss fará a leitura de caracteres de entrada via arquivo (yyin = f_in)

Flex – Gerador de Analisador Léxico

- Exemplo de utilização do *Flex*
  - Considere um arquivo de entrada que contenha as seguintes cadeias de caracteres:



Flex – Linguagem Tiny

```
digit [0-9]
number {digit}+
letter [a-zA-Z]
id {letter}+
newline \n
space [ \t]+
%%
“if” {return IF;}
“then” {return THEN;}
“else” {return ELSE;}
“end” {return END;}
“repeat” {return REPEAT;}
“until” {return UNTIL;}
“read” {return READ;}
“write” {return WRITE;}
“:=” {return ASSIGN;}
“=” {return EQ;}
“<” {return LT;}
“+” {return PLUS;}
```

Flex – Linguagem Tiny

```
"-" {return MINUS;}
"*" {return TIMES;}
"/" {return OVER;}
"(" {return LPAREN;}
")" {return RPAREN;}
";" {return SEMI;}
{number} {return NUM;}
{id} {return ID;}
{newline} {lineno++;}
{space} {/* skip */}
"{" { char c;
      do{
        c = input();
        if(c == '\n') lineno++;
      }while(c!='}');
    }
{return ERROR}
%%
```

Flex – Linguagem Tiny

```
TokenType getToken(void){
    static int firstTime = TRUE;
    TokenType currentToken;
    if(firstTime){
        FirstTime = FALSE;
        Lineno++;
       yyin = source;
       yyout = listing;
   }
   CurrentToken = yylex();
   strncpy(tokenString,yytext,MAXTOKENLEN);
   if(TraceScan){
       fprintf(listing,"\t%d: ",lineno);
       printToken(currentToken,tokenString);
  }
  return currentToken;
}
```

Flex – Exercício

Projete um programa Flex para contar entrada de números, identificadores e símbolos especiais em um arquivo

Entrega via Moodle

Flex – Gerador de Analisador Léxico

- Bibliografia consultada

Capítulo 2 e Apêndice B - LOUDEN, K. C. **Compiladores: princípios e práticas.** São Paulo: Pioneira Thompson Learning, 2004

RICARTE, I. **Introdução à Compilação.** Rio de Janeiro: Editora Campus/Elsevier, 2008.