Análise léxica

Uma linguagem para especificação de analisadores léxicos

Prof. Edson Alves

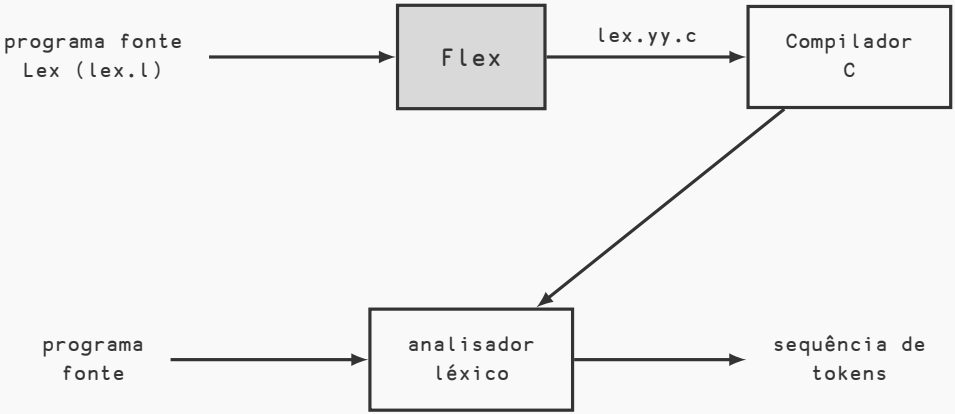
Faculdade UnB Gama

Flex

- Flex (*Fast Lexical Analyzer Generator*) é um programa para a geração de analisadores léxicos
- Ele foi escrito em linguagem C por Vern Paxson por volta de 1987
- Ele pode ser usado em conjunto com um gerador de analisadores sintáticos (por exemplo, o Yacc e o GNU Bison)
- Flex é mais flexível e gera códigos mais rápidos que o Lex, outro programa gerador de analisadores léxicos
- Ele pode ser instalado, em distribuições Linux baseadas no Debian, por meio do comando

```
$ sudo apt-get install flex
```

Fluxo de uso do Flex para geração de analisadores léxicos



Programas Lex

- Programas Lex são salvos em arquivos com extensão `.l` (ou `.lex`)
- Este programas exportam uma função chamada `yylex()` que, ao ser chamada, extraí o próximo token do programa fonte
- O código gerado (arquivo `lex.yy.c`) pode ser usado para gerar um executável independente, ou pode ser compilado como código objeto e ser integrado ao analisador sintático
- Os programas Lex são dividos em três partes: a seção de definições, a seção de regras e a seção de códigos de usuário
- A vantagem do uso de programas Lex é que eles permitem a especificação dos tokens por meio de expressões regulares, e a implementação dos diagramas de transição é feita automaticamente pelo Flex

Seção de definições

- Nesta seção são declaradas variáveis, constantes e definições regulares
- As declarações desta seção deve ser delimitado pelas sequências de caracteres `"%{"` e `"%}"`
- O conteúdo desta seção é copiado diretamente para o arquivo `lex.yy.c`
- As definições regulares devem ser declaradas após esta seção, na forma

```
nome regex
```

- Uma vez definido um nome, ele pode ser usado nas definições regulares subsequentes, desde que sejam delimitados por chaves

Exemplo de seção de declarações

```
%{
    enum {
        LT, LE, EQ, NE, GT, GE,
        IF, THEN, ELSE, ID, NUM, RELOP, END_OF_FILE
    };

    int yylval;
    int instalar_id();
    int instalar_num();
%}
```

```
delim   [ \t\n]
ws      {delim}+
letra   [A-Za-z]
digito  [0-9]
id      {letra}({letra}|{digito})*
num     {digito}+(\.{digito}+)?(E[+-]?{digito}+)?
```

Seção de regras

- Esta seção contém uma série de regras, uma por linha, na forma

```
padrão ação
```

- O padrão não deve estar indentado e deve estar na mesma linha da ação
- O padrão pode conter algum nome presente nas declarações regulares
- Neste caso, o nome deve ser delimitado por chaves
- Esta seção é delimitada pela sequência de caracteres **%%**

Exemplo de seção de declarações

```
%%

{ws}     { printf("WS\n"); /* Nenhum valor retornado */ }
if       { return IF; }
then     { return THEN; }
else     { return ELSE; }
{id}     { yylval = instalar_id(); return ID; }
{num}    { yylval = instalar_num(); return NUM; }
"<"      { yylval = LT; return RELOP; }
"<="     { yylval = LE; return RELOP; }
"="      { yylval = EQ; return RELOP; }
"<>"     { yylval = NE; return RELOP; }
">"      { yylval = GT; return RELOP; }
">="     { yylval = GE; return RELOP; }
<<EOF>> { return END_OF_FILE; }

%%
```

Seção de código de usuário

- Esta seção também é copiada diretamente para o arquivo `lex.yy.c`
- Uma outra alternativa é definir estes códigos em arquivos separados e depois carregar este código na compilação do analisador léxico

```
37 int instalar_id()
38 {
39     // Insere o lexema e o token na tabela de símbolos e retorna o índice da tabela
40     // onde o símbolo foi inserido. O lexema fica armazenado na variável yytext
41     return -1;
42 }
43
44 int instalar_num()
45 {
46     // Insere o valor do lexema na tabela de números e retorna o índice da tabela
47     // onde o número foi inserido. O lexema fica armazenado na variável yytext
48     return -2;
49 }
```

Exemplo de função `main()` para um analisador léxico independente

```
51 // Retorno diferente de zero indica que o scanner deve encerrar ao encontrar EOF
52 int yywrap() { return 1; }
53
54 int main()
55 {
56     while (1)
57     {
58         int token = yylex();
59
60         if (token == END_OF_FILE)
61         {
62             printf("Fim da entrada\n");
63             return 0;
64         }
65
66         printf("Token = %d, yytext = %s, yylval = %d\n", token, yytext, yylval);
67     }
68
69     return 0;
70 }
```

Referências

1. **AHO**, Alfred V, **SETHI**, Ravi, **ULLMAN**, Jeffrey D. *Compiladores: Princípios, Técnicas e Ferramentas*, LTC Editora, 1995.
2. GeeksForGeeks. Flex (Fast Lexical Analyzer Generator), acesso em 04/06/2022.