

Flex/Lex

Construindo Analisador Léxico

Validando com um Analisador Sintático Bison



Análise Léxica

É a primeira fase do processo de compilação e sua função é fazer a leitura do programa fonte, caractere a caractere, agrupar os caracteres em lexemas e produzir uma sequência de símbolos léxicos conhecido como tokens.

As principais tarefas de um analisador léxico são:

- Ler os caracteres da entrada;
- Agrupá-los em **lexemas**;
- Produzir **tokens**.

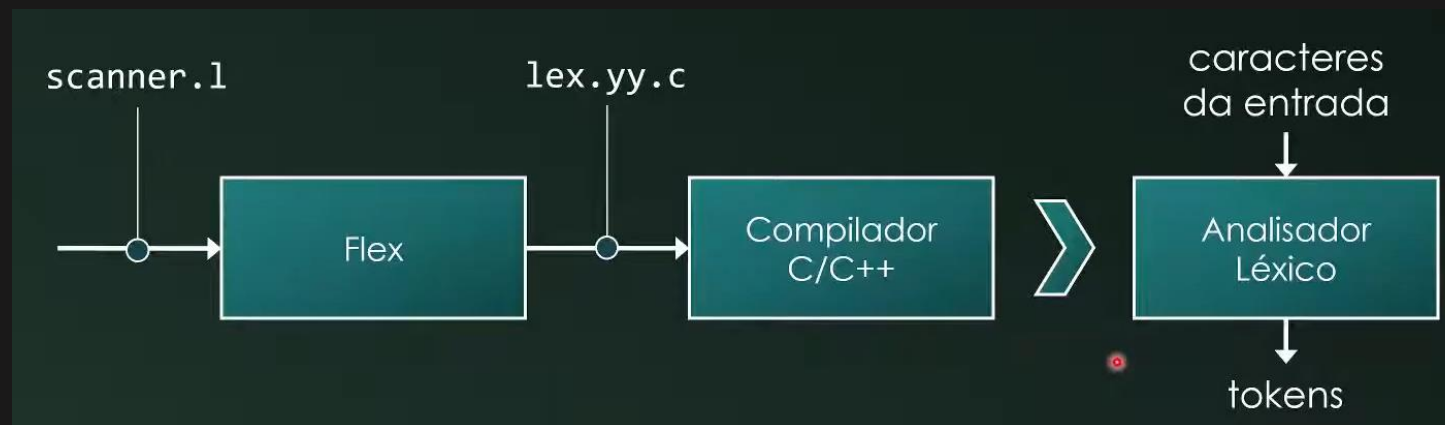
Análise Léxica

Token	Padrão	Lexema	Descrição
<const, >	Sequência das palavras c, o, n, s, t	const	Palavra reservada
<while, >	Sequência das palavras w, h, i, l, e	while, While, WHILE	Palavra reservada
<if, >	Sequência das palavras i, f	If, IF, iF, If	Palavra reservada
<=, >	<, >, <=, >=, ==, !=	==, !=	
<numero, 18>	Dígitos numéricos	0.6, 18, 0.009	Constante numérica
<literal, "Olá">	Caracteres entre ""	"Olá Mundo"	Constante literal
<identificador, 1>	Nomes de variáveis, funções, parâmetros de funções.	nomeCliente, descricaoProduto, calcularPreco()	Nome de variável, nome de função

Flex/Lex

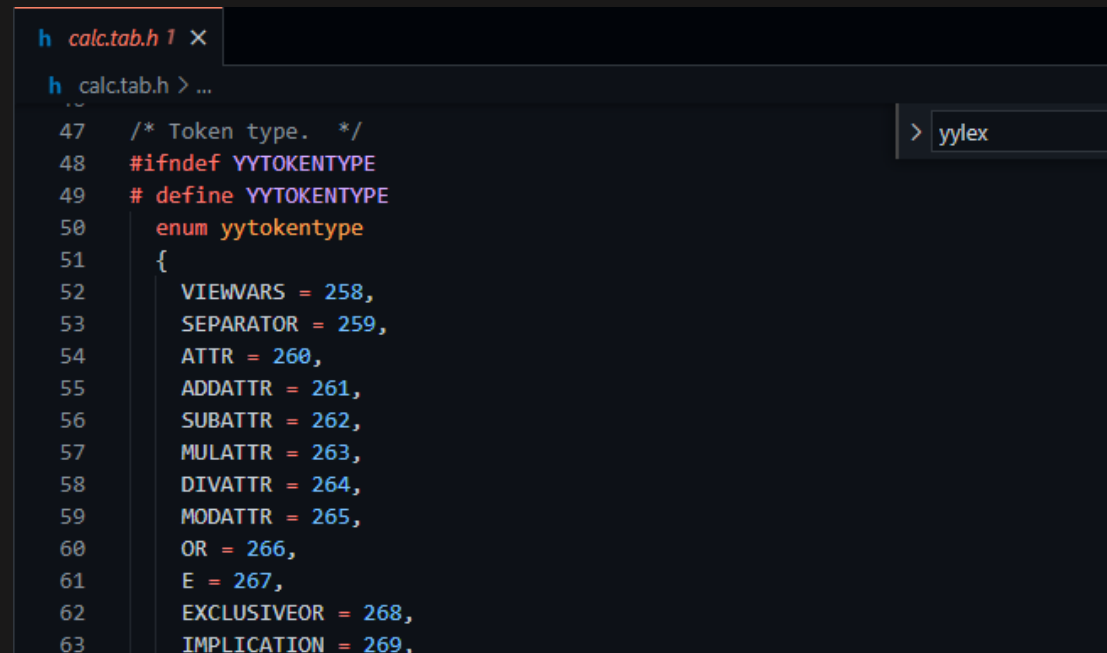
O Flex é uma ferramenta geradora de analisadores léxicos que utiliza de expressões regulares para descrever padrões de tokens.

- Tem extensão **.l**
- Gera código em linguagem C (ou C++) em um arquivo chamado **lex.yy.c**;



Flex/Lex

O Analisador Léxico é uma função `yyllex()` que retorna um inteiro e esse inteiro representa o código para um dos tokens.



```
h calc.tab.h 1 X
h calc.tab.h > ...
47  /* Token type. */
48  #ifndef YYTOKENTYPE
49  # define YYTOKENTYPE
50  enum yytokentype
51  {
52      VIEWARS = 258,
53      SEPARATOR = 259,
54      ATTR = 260,
55      ADDATTR = 261,
56      SUBATTR = 262,
57      MULATTR = 263,
58      DIVATTR = 264,
59      MODATTR = 265,
60      OR = 266,
61      E = 267,
62      EXCLUSIVEOR = 268,
63      IMPLICATION = 269,
```

0 Trabalho

Fazer um analisador léxico, utilizando softwares geradores de analisadores léxicos, para uma calculadora que seja capaz de reconhecer funções matemáticas, tais como multiplicação, divisão, soma, subtração, logaritmo, raiz quadrada, exponenciação, operações lógicas, e inequações, e que também seja capaz de armazenar os seus resultados em variáveis que por sua vez possam ser usadas nos cálculos posteriores, dentro da própria calculadora.

O Trabalho

Um programa Flex possui o seguinte formato:

- Definições;
- Regras;
- Código Fonte.



Definições E Definições Regulares

```

/*****
                                DEFINIÇÕES
*****/
%{
#include "calc.tab.h"
%}

/*****
                                Definições Regulares
*****/
white [ \t]+
digit [0-9]
integer {digit}+
real {integer}("."{integer})?{exponent}?
exponent [eE][+-]?{integer}
lether [a-zA-Z]
identifier ({lether}|_)( {lether}|{digit}|_)*

...
```


Regras

Parte 1



```
/******  
    REGRAS: Padrões reconhecidos  
*****/  
%%  
"log" return LOG;  
"sqrt" return Sqrt;  
"viewvars()" return VIEWVARS;  
  
{white} { }  
{real} { yylval.value = atof(yytext); return NUMBER; }  
{identifier} { sscanf(yytext, "%s", yylval.lexeme); return VAR; }  
  
...
```

Regras

Parte 2

```

," return SEPARATOR;
"=" return ATTR;
"+=" return ADDATTR;
"-=" return SUBATTR;
"*=" return MULATTR;
"/=" return DIVATTR;
"%=" return MODATTR;
"|" return OR;
"&&" return E;
"|||" return EXCLUSIVEOR;
"->" return IMPLICATION;
"==" return EQUAL;
"!=" return DIFF;
"<" return LESS;
"<=" return LESSOREQUAL;
">" return MORE;
">=" return MOREOREQUAL;
"+" return ADD;
"-" return SUB;
"*" return MUL;
"/" return DIV;
%" return MOD;
"^" return POW;
"!" return NOT;
"(" return LBRACKET;
")" return RBRACKET;
"\n" return EOL;
%%

```

Mã no código!