COMUNICA BR | ACESSO À INFORMAÇÃO | PARTICIPE | LEGISLAÇÃO | ÓRGÃOS DO GOVERNO

Marcar como feito

Descrição Visualizar envios

Data de entrega: segunda-feira, 8 set. 2025, 23:55
Arquivos requeridos: scan.lex (Baixar)
Tipo de trabalho: Trabalho individual

Crie um arquivo LEX que reconheça os tokens descritos a seguir:

Token	Lexemas	Padrão
_ID	a b _1 ab1 $tab $_ $1	Identificadores podem ou não começar com $, mas $ só pode aparecer como primeiro caractere. Caracteres válidos são letra, _ ou dígito, sendo que dígito não pode aparecer como primeiro caractere.
_INT	1 221 0	Números inteiros
_FLOAT	0.1 1.028 1.2E-4 0.2e+3 1e3	Números de ponto flutuante e em notação científica
_FOR	for For fOr	for, *case insensitive*
_IF	if IF	if, *case insensitive*
_MAIG	>=	>=
_MEIG	<=	<=
_IG	==	==
_DIF	!=	!=
_COMENTARIO	/* Um comentário */ /* Outro comentário */ // Comentário até o final da linha // /*Esse comentário anula o início /* Esse comentário foi terminado! // */	Um comentário pode se estender por mais de uma linha, e não pode haver comentário dentro de comentário. Não deve juntar comentários que estão separados.
_STRING	"hello, world" 'hello, world' "Aspas internas com \" (contrabarra)" "ou com "" (duas aspas)" 'd''água' "d'água" 'd\'água'	Uma string começa e termina com aspas. Se houver aspas dentro da string devemos usar contrabarra ou duas aspas. Uma string não pode ir além do final da linha. Podemos usar aspas simples ou aspas duplas, mas se começou com aspas simples deve terminar com aspas simples.
_STRING2	`hello, world` `hello, world` `Hora atual: ${agora} horas!`	Aspas invertidas. Podem se estender para além do final da linha. Nesse caso a string deve conter os espaços em branco e a quebra de linha ('\n'). É permitido dentro de string o ${ID}, que deve ser identificado (a string termina antes do "$" e recomeça após o "}". Isso deve retornar 3 tokens: _STRING, _EXPR e _STRING.

Identificadores (letras, '$' ou sublinhado seguido de letras, sublinhado e dígitos), números, strings entre aspas duplas, entre aspas simples e comentários. Declare uma variável '**string lexema**' e coloque nela o lexema que casou com o padrão e deve ser retornado, possivelmente com alguma alteração (por exemplo, tirar as aspas).

Assuma a seguinte declaração para os tokens:

```
enum TOKEN { _ID = 256, _FOR, _IF, _INT, _FLOAT, _MAIG, _MEIG, _IG, _DIF, _STRING, _STRING2, _COMENTARIO, _EXPR };
```

A compilação será feita com o seguinte comando:

```
lex scan.lex
g++ -Wall -std=c++17 main.cc -lfl
```

OBS: Algumas distribuições do linux utilizam bibliotecas com outro nome, possivelmente: "-lf" ou "-ll".

O programa "main.cc" usado é o seguinte:

```
#include <stdio.h>
#include <string>

using namespace std;

enum TOKEN { _ID = 256, _FOR, _IF, _INT, _FLOAT, _MAIG, _MEIG, _IG, _DIF, _STRING, _STRING2, _COMENTARIO };

extern "C" int yylex();
extern "C" FILE *yyin;

void yyerror(const char* s);

#include "lex.yy.c"

auto p = &yyunput; // Para evitar uma warning de 'unused variable'

int main() {
  int token = 0;

  while( (token = yylex()) != 0 )
    printf( "%d %s\n", token, lexema.c_str() );

  return 0;
}
```

Arquivos requeridos

scan.lex

```
1  /* Coloque aqui definições regulares */
2
3  WS  [ \t\n]
4
5
6  %%
7      /* Padrões e ações. Nesta seção, comentários devem ter um tab antes */
8
9  {WS}    { /* ignora espaços, tabs e '\n' */ }
10
11 .       { return *yytext;
12           /* Essa deve ser a última regra. Dessa forma qualquer caractere isolado será retornado pelo seu código ascii. */ }
13
14 %%
15
16 /* Não coloque nada aqui - a função main é automaticamente incluída na hora de avaliar e dar a nota. */
```

VPL