FEITEP - FACULDADE DE ENGENHARIA E INOVAÇÃO TÉCNICO PROFISSIONAL

COMPILADORES

CLAUDIO BELO RODRIGUES JUNIOR

LEONARDO MATHEUS BRUMATT DANTAS

ANALISE LÉXICA - MEMELIN

MARINGÁ

2019

Sumário

ANALIZADOR LÉXICO - MEMELIN

1. IDENTIFICAÇÃO

FACULDADE DE ENGENHARIA E INOVAÇÃO TÉCNICO PROFISSIONAL- FEITEP

2. INTRODUÇÃO

Analisador realizado como forma estudantil para a disciplina de compiladores, focado apenas na leitura léxica, utilizando a ferramenta FLEX para seu desenvolvimento.

2.1. FLEX:

Com o intuído de apenas de realizar a parte léxica do compilador, foi utilizado a ferramenta FLEX, também conhecido como fast lexical analyse, um gerador de analisadores lexicais, aonde se permite a leitura de um input stream e executar ações quando esta stream correspondem aos tokens definidos pelo programador.

O formato do arquivo de especificação do flex é constituído em três partes separadas por %%:

definições

%%

regras

%%

código

As definições é aonde da se nomes para as expressões regulares e também pode ser aplicado algumas opções que alteram o comportamento do analisador. O código é realizado na linguagem C que como saída do flex é um .exe em C, assim permitindo definir algumas funções para auxiliar as ações do analisador;

Algumas expressões regulares utilizadas no 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”

3. Definições da linguaguem.

Inicio do codigo:

vamo que vamo

Encerramento do código:

diga tchau lilica

Exemplo:

vamo que vamo

insira seu código aqui ##;
Diga tchau lilica

Printar na tela:

fala um a pra tu ve ve("a a")

Ler:

conta pro tio

for:

roda roda jequeti

If:

que menino?

Elseif:

isso menino

Else:

esse menino e coisado

While:

procurando a graca

Continue:

prossiga

Switch:

achei vacilo

Case:

achei

Default:

achei graca tambem

break:

tururu

Tipos de Dados:

textao (string)
biridin (int)
mai love(float)

Comentarios:

insira seu comentario aqui

Operadores:

" + " : Operador de soma
" - " : Operador de subtração
" * " : Operador de multiplicação
" / " : Operador de divisão

Identificadores

x,var1,var2, ...

Simbologia:

“ (“ : inicia parâmetro

“) ” : fecha parâmetro

“ { “ : início método

“ } ” : fim do método

“ ; ” : fim da linha

“ : ” : dois pontos

“ ! ” : negação

“ == ” : not equal

“ != ” : less

“ < ” : maior que

“ > ” : menor que

“ <= ” : maior igual

“ >= ” : menor igual

“ , ” : virgula

Números:

0-9

3.1. Tabela de tokens.

Expressões	Token
textao("grandao")	t_string
vamo que vamo	t_begin
diga tchau lilica	t_end
fala um a pra tu ve	t_print

Expressões	Token
conta pro tio	t_scan
biridin	t_int
mai love	t_float
que menino?	t_if
isso menino	t_elseif
esse menino e coisado	t_else
procurando a graca	t_while
achei vacilo	t_switch
achei graca tambem	t_deafault
achei	t_case
tururu	t_break
prossiga	t_continue
roda roda jequiti	t_for
identificadores	t_id
## ##	t_comment
:=	t_assignment
0-9	t_number

Expressões	Token
;	t_end_command
(	t_start_expression
)	t_end_expression
{	t_start_block
}	t_end_block
+	t_plus
-	t_minus
*	t_multiply
/	t_share
!	t_not
==	t_equal
!=	t_not_equal
<	t_greater
>	t_less
<=	t_less_equal
>=	t_greater_equal
,	t_comma

3.2. Exemplo do analisador léxico

Codigo utilizado:

```
vamo que vamo

mai love x1;

conta pro tio(x1);


mai love resultado := x1 * 2;

fala um a pra tu ve("O dobro do numero é ");

fala um a pra tu ve(x1);

diga tchau lilica
```

Saida gerada:

```
vamo que vamo
<t_begin, vamo que vamo>
        mai love x1;
<t_float, mai love>
<t_id, x1>
<t_end_command, ;>
        conta pro tio(x1);
<t_scan, conta pro tio>
<t_start_expression, (>
<t_id, x1>
<t_end_expression, )>
<t_end_command, ;>

        mai love resultado := x1 * 2;
<t_float, mai love>
<t_id, resultado>
<t_assignment, :=>
<t_id, x1>
<t_operators, *>
<t_number, 2;>

        fala um a pra tu ve("O dobro do numero é ");
<t_print, fala um a pra tu ve>
<t_start_expression, (>
<t_string, "O dobro do numero é ">
<t_end_expression, )>
<t_end_command, ;>
        fala um a pra tu ve(x1);
<t_print, fala um a pra tu ve>
<t_start_expression, (>
<t_id, x1>
<t_end_expression, )>
<t_end_command, ;>
diga tchau lilica
<t_end, diga tchau lilica>
```

LINK para o repositório github:

https://github.com/ClaudioBelo/analizador-lexico