2008-02-26

Operadores ER (Expressões Regulares)

. - qualquer carácter

" * - inicia e finaliza uma string

* → ø ou mais

^ → início de linha

$ → fim de linha

[] → classes.

1)

a)

data = [0-9]{4}\-(0[1-9]|1[0-2])\-(3[01]|[12][0-9]|0[1-9])

ano mês dia

b)

[ab]..[ab]

(a|b)..(a|b)

2)

(.*af.*fg.*k.*)|.*afg.*k.*)

.*(fgk|fgh|fgf|fg|fg□)..*

Sed

Comandos

-n reprime a saída

p imprime

w ~~Ficheiro para ficheiro~~ redirecciona a saída

d remove da saída

s substitui

2008-02-26

Expressões Regulares

Operadores:

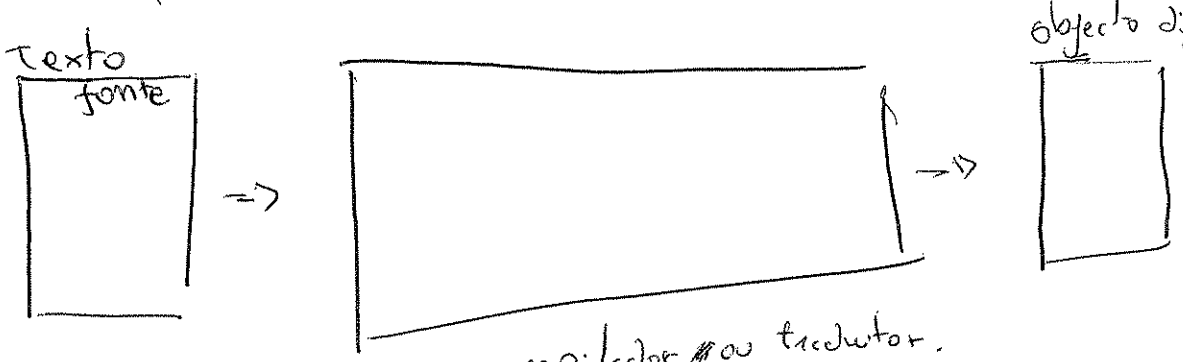
- `.` - qq carácter excepto o '\n'
- `$` - no fim da er faz com que o match seja feito apenas no fim da linha
- `//` - delimitador de strings; os operadores perdem o significado especial
- `\` - retira o significado ao carácter que se lhe seguir
- `//` - faz match c/ a última er compilada
- `*` - 0 ou mais ocorrências do elemento precedente
- `[]` - classe de caracteres < - domínio, ^ inverso
- `^` - no início de uma linha

Comandos do SED:

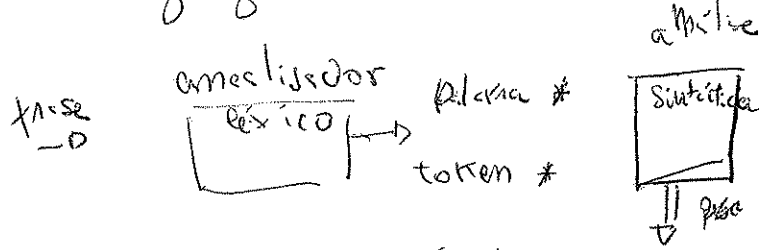
- `s` - substituição
- `a` - acrescentar linhas
- `i` - inserir linhas
- `d` - apagar linhas
- `c` - trocar linhas
- `p` - enviar linhas para output
- `w` - escrever para ficheiro

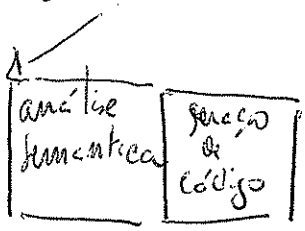
Gilberto Martins Fefoeiros 50201
ES/02/09

Compilador.



linguagens.





análise
léxica

Expressões regulares.

Ø linguagens sem restrição

1 - Gramáticas D.C.

2 - Gramáticas I C

3 expressões regulares.

aulas.

Chomsky

a
a|b (a ou b)
a.b (a e b)
a* (0 ou a+)
a+ (1 ou + vezes)

POSIX a?

[a-zA-Z]

. (todos os caract. excepto \n)

\n

\ (escape ao caracter)

a) Especificar o identificador (id) de uma ling. de programação.

um C# — ou letra letra = [a-3A-Z]

dígitos.

id = (_ | letra) . [a-3A-Z_\-|.Ø-9]*

no comp. no meio.

ordem ASCII dos caract.

b) Especificar um nº inteiro

Int = (+|-)?[Ø-9]+

se der problemas.

(1+|1-)?[Ø-9]+

c) Especificar um número real ex (+8, -.75, 1.835E+8E-3)

Real = ~~(1+|1-)?~~ [Ø-9]* (. [Ø-9]+)

---

avaliação.

Avaliação: ? 28 março.

P1 - (5%)

P2 - (35%)

} um grupo (3 elem)

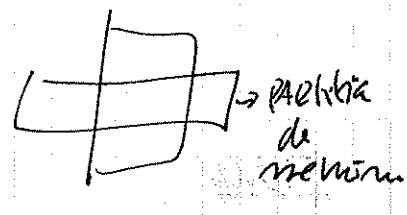
teste (60%)

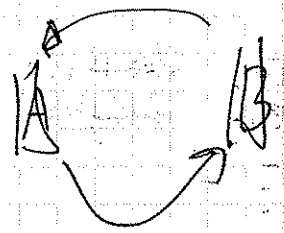
03/03

```
#define LET 101
#define IGUAL 201
/*...*/
%%
[ \t\n]* ;
let     return (LET);
=       return (IGUAL);
'{'     return (CHAV-A);
...
[0-9]+  return (VALOR)
[a-z][a-zA-Z]*  return (VAR)
.       return ERRO
```

pl.lex

PLC






```
-/->
> flex   pl.lex
   ↳ lex.yy.c { int yylex();
-/-
```

myProg.c

```
#include "lex.yy.c"
...
int prox_Simb;
...
int main(){
    prox_simb = yylex();
```

```
switch (prox_Simb)
{
   case LET =  ...
   case VALOR =  ...
}
```

```
> gcc -o myProg -lfl
myProg.c
```

thesis

Eu: Bergamo
Ele: Resposta
⋮

Elimina os prefixos: Eu e Ele

```
%%
Eu:      ;
Ele:     ;
.        Echo;
```

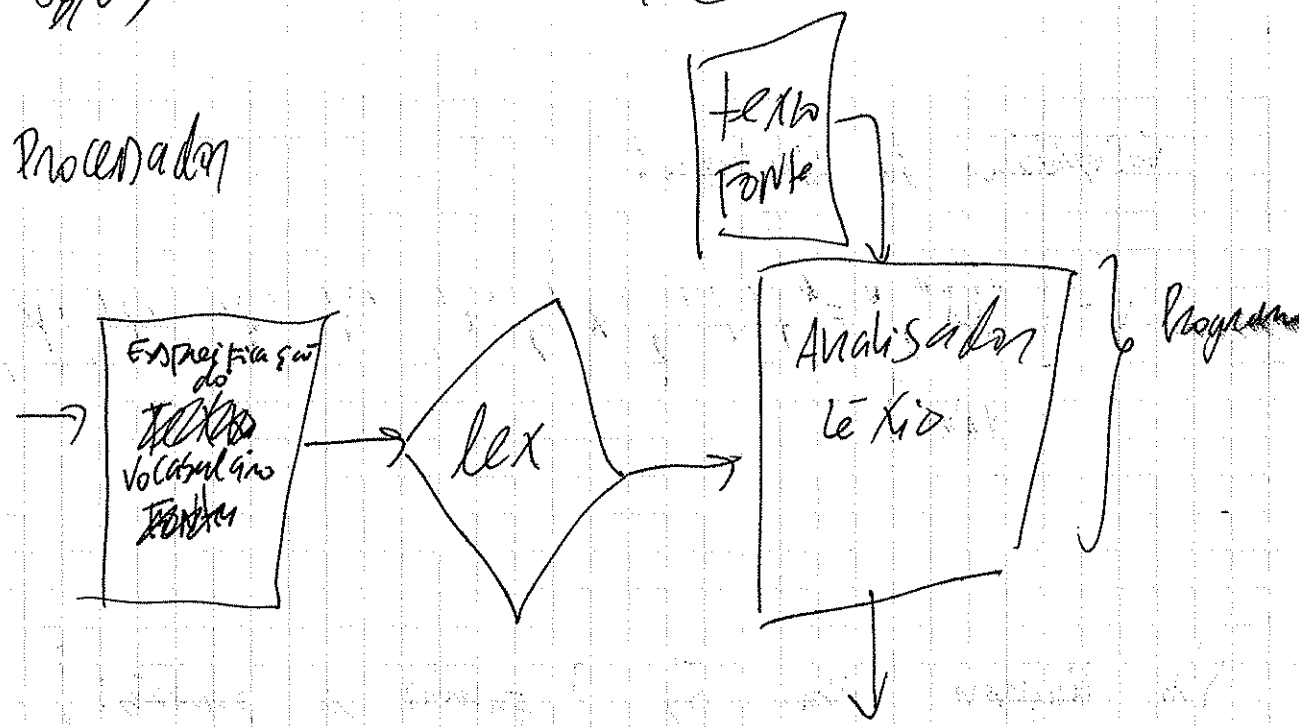
P2.flex

> flex P2.flex

08/03

PLC

Processadores



- Lista de palavras que obedecem à linguagem
- Lista de palavras transformadas

Implementação uma calculadora

Além das operações aritméticas, deve-lhe possuir funcionalidade I/O

```
Programa exemplo
{
    let a = 5
    let b = 7
    x = a ** a + b
    x!
    Y = C? * a/L
}   Y!
```

Vocabulário da linguagem:

$V = \{$ "let", " = ", ' / ', ' * ', ' + ', ' ! ', ' ? ', ' { ', ' } ', var, valor $\}$

{ A linguagem tem 3 tipos de palavras:
- Keywords (palavras reservadas)
- Símbolos
- tokens variáveis }

—//—

```
%{
declarações
%}

%%
exreg    ação —

%%
código C auxiliar
```

; - filtragem (não envia nada para output)

echo ; - apanha o que reconhece e envia para a saída
⇓
Printf("%s", yytext) stdin ⇒ stdout

{
bloco de instruções
}

return ...

3/10/2000 Pedro Araújo nº 24868

LaTeX
↓
x.tex

$ latex x.tex ⟶ $ pdf latex x.tex
↳ x.dvi ↳ x.pdf

$ dvips x.dvi
↳ x.ps

literate programming

\begin{code}
\end{code}

Expressões Regulares

Seja $V = \{a, b, c\}$

- ε é uma expressão regular
- a é " " "
- Se P e Q são expressões regulares então:
  - P | Q
  - P . Q
  - P*

Exercício: Construa expressões regulares para as seguintes linguagens:

a) identificadores duma linguagem de programação

$V = \{$ letra, dígito $\}$

letra . (letra | dígito)*

Nota: Duas expressões regulares dizem-se equivalentes se a linguagem gerada for a mesma.

i) $(P|Q)|R = P|(Q|R)$

ii) $P|\emptyset = \emptyset|P = P$

iii) $P|Q = Q|P$

iv) $P|P = P$

v) $(P.Q).R = P.(Q.R)$

vi) $P.\varepsilon = \varepsilon.P = P$

vii) $P.(Q|R) = P.Q|P.R$

~~viii) $(Q|R).P$~~

~~ix)~~ viii) $P.P^* = P^*.P = P^+$

ix) $P^* = \varepsilon|P^+$

x) $(P|\varepsilon)^+ = P^*$

Exercício:

$V = \{a, e, i, o, u, \text{consoante}\}$ palavra exemplo

a) Expressão regular:

Sequência de letras contendo as 5 vogais por ordem. Cada vogal só deverá ocorrer uma vez na sequência.

$\text{consoante}^*.a.\text{consoante}^*.e.\text{consoante}^*.i.\text{consoante}^*.o.\text{consoante}^*.u.\text{consoante}^*$

b) Sequências de letras constituídas por subsequências contendo as 5 vogais por ordem. Cada vogal pode aparecer mais do que uma vez numa subsequência.

$(\text{consoante}^*.a^*.\text{consoante}^*.e^*.\text{consoante}^*.i^*.\text{consoante}^*.o^*.\text{consoante}^*.u^*.\text{consoante}^*)^*$

~~Duas~~ Sequências de zeros e uns que não contêm a subsequência 011.

$1^*.(01|0)^+$

Sequências de zeros e uns com um número par de zeros e um número impar de uns

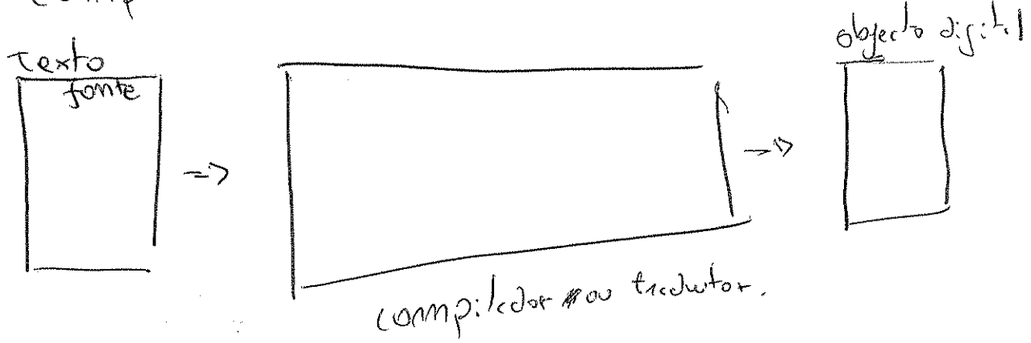
???

Sequências de uns, dois e três, em que nenhuma se repita.

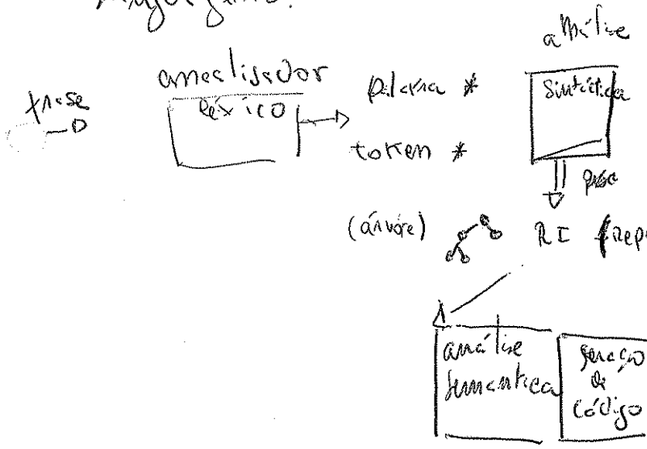
$(1|2|3)\,|\,(1.(23)|2(13)|3(12))\,|\,(1.(2313)|2(1331)|3(1221))$

Gilberto Mattos Fegueiras 50201
ES/02109

compilador.



linguagens.



analise
léxica

Expressões Regulares.

Ø linguagens sem restrição

1 - Gramática de.
2 - Gramática IC
3 expressões regulares.

aulas.

Chaumsky

a
a|b (a ou b)
a.b (a + b)
a* (0 ou a+)
a+ (1 ou + vezes)

POSIX a?
[a-zA-Z]
. (todos os caract. exceto \n)
\n
\ (escape ao caracteres)