**{testes de conteúdo}**

## 12.1 SUBSTRING EM STRING

A linguagem PERL tem uma série de artifícios para testar se um string contém determinado substring ou outras coisas. Este tipo de teste não é muito comum em outras linguagens e deixa meio confusos os programadores tradicionais (coboleiros etc.)

Nos vários exemplos desse capítulo testaremos um string (vamos chamá-lo de BÁSICO) assim composto:

```
$x = "xaxbcdeee78\nfg";
```

e os testes deverão dar sempre positivos, retornando **true**.

Para, por exemplo, verificar se nesse string (em $x) existe o substring **"cd"** usamos a fórmula:

```
$y = ($x=~/cd/);
```

Repare que o substring NÃO aparece entre aspas.

Caso exista (e existe!), em $y teremos **true** ou 1. Caso contrário teremos **false** ou **undef**.

Um print de **true** normalmente mostra: **1** e o de **undef** não aparece.

## 12.2 CARACTERE CURINGA

Digamos que queiramos saber se existe (no nosso string) um "b" seguido de dois caracteres QUAISQUER e depois um "e". Fazemos:

```
$y = ($x=~/b..e/);
```

Para cada caractere curinga temos que colocar um ponto. Podemos explicitar a quantidade com um número dentro de chaves:

```
$y = ($x=~/b.{2}e/);
```

## 12.3 CARACTERE DE GRUPO

E, para saber se existe UM DOS VÁRIOS caracteres: b ou h ou i:

```
$y = ($x=~/[bhi]/);
```

Nosso string básico não contém nem h nem i mas contém b. Donde em $y teremos true ou 1.

## 12.4 QUALQUER ALGARISMO

Para testar a existência de QUALQUER algarismo:

```
$y = ($x=~/\d/);
```

repare que isso não é um teste para ver se $y é um número. Mas caso $x fosse um número o teste também daria positivo.

## 12.4 ESCAPE E BRANCO

Para verificar se contém um escape (\n, por exemplo) ou um branco no meio do string temos:

```
$y = ($x=~/\s/);
```

## 12.5 REPETIÇÃO CONTÍGUA

Vamos chamar de **molde** aquilo que queremos testar se existe dentro do string.

Para verificar se o molde aparece uma determinada quantidade de vezes, contígüamente, temos três expressões:

1. para menos que tantas vezes (inclusive) temos:

   ```
   $y = ($x=~/e{3}/);
   ```

   verificando, tendo como molde o caractere **e** (poderia ser um substring), para 3 vezes ou menos.
2. para mais que tantas vezes (inclusive):

   ```
   $y = ($x=~/e{2,}/);
   ```

   para duas ou mais vezes.
3. no intervalo (inclusive extremos):

   ```
   $y = ($x=~/e{2,5}/);
   ```

   para entre 2 e 5 vezes.

Como no nosso string básico temos: **eee**, todos estes testes dariam positivo.

## 12.7 IGUAIS DESCONHECIDOS

Essa é complicada! Podemos querer saber se um caractere (que não sabemos qual é) que há numa primeira posição se repete em uma outra posição.

Digamos que, no nosso string básico (olha ele aqui de novo):

```
$x = "xaxbcdeee78\nfg";
```

não sabemos que na primeira posição tem um **x** mas queremos saber se na primeira posição temos o mesmo caractere que na terceira.

Definimos o PRIMEIRO DESCONHECIDO com (.) e o IGUAL AO PRIMEIRO DESCONHECIDO COM \1. Veja só a sopa de letrinhas:

```
$y = ($x=~/(.)a\1bcdeee78\nfg/);
```

Como na primeira e na terceira posições temos o mesmo caractere, em $y teríamos true.

Não podemos ter o \1 numa posição seguida de algarismos.

Para testar outros pares temos que usar \2, \3 etc. sempre com (.) para o primeiro do par.

Se os pares não têm só um caractere usamos (.*) no primeiro molde da dupla.

## 12.8 OPÇÕES

Se queremos testar se o string básico contém, por exemplo o substring **ax** ou **cd** fazemos:

```
$y = ($x=~/ax|cd/);
```

No caso de apenas um caractere no molde veja o item 5.

## 12.9 IMPORTANTE!IMPORTANTE!

Se você já está de saco cheio desses hierogrifos não se desespere e guarde com todo carinho o que vamos ensinar agora.

Isso vai ser muito usado no caso de pesquisa em arquivo, quando é dada a chave de um registro (essa chave é colocada nas primeiras posições do dito cujo registro).

O teste é para ver se o string COMEÇA com um determinado substring.

Digamos que queremos saber se $y começa com **xaxb**. Temos:

```
$y = ($x=~/^xaxb/);
```

Vamos fazer mais para frente um exercício importante usando esse truque.

## 12.10 NO FIM

E, para testar um substring no fim de um string temos:

```
$y = ($x=~/fg$/);
```

Espero que você esteja atento e tenha reparado no sinal de $ no fim do molde.

## 12.11 MAIÚSCULAS E MINÚSCULAS

Estes testes todos que vimos levam em consideração se os caracteres estão em maiúscula ou minúscula. Para isso ser ignorado coloque depois da segunda barra a letra: **i**. Assim:

```
$y = ($x=~/^xaxb/i);
```

## 12.12 MOLDE EM VARIÁVEL

É válido que o molde seja o conteúdo de uma variável.Por exemplo:

```
$m = "cd";
$y = ($x=~/$m/);
```

Como em $x temos **cd** em $y teremos: true.