Linguagens de Programação
Departamento de Engenharia Eletrônica e de Computação - DEL-Poli/UFRJ
Prof.: Miguel Elias Mitre Campista

== Laboratório 2: Exercícios de Perl ==

Para esquentar...

Execute no terminal o seguinte programa para verificar se o interpretador Perl está funcionando corretamente:

```
perl –e 'print "hello world!\n";'
```

Caso tudo ocorra bem, faça os seguintes exercícios:

1. Dado um arquivo de texto onde as linhas ímpares possuem o nome de uma pessoa e as linhas pares o sobrenome correspondente ao nome da linha anterior, escreva um programa em Perl que leia todas as linhas do arquivo, armazene as informações em uma hash e imprima a hash resultante na tela. Note que cada elemento da hash é um par chave/valor igual ao par nome/sobrenome de cada contato, ou seja, **`tabela{"nome"}=sobrenome`**. Assuma que o arquivo texto já existe, por exemplo, "**`arquivo.txt`**", e que está no mesmo diretório do programa a ser desenvolvido. Modularize o código e apresente o resultado na tela.

2. Usando o mesmo arquivo da questão anterior, calcule a porcentagem de alunos da turma que possuem o mesmo nome ou sobrenome.

3. Escreva um programa em Perl para obter as seguintes informações de um arquivo de texto: número de linhas, número de palavras, número total de caracteres, número de caracteres diferentes de espaço em branco. Utilize a função **`length`** para contar o número de caracteres de uma string e a função **`split`** para inserir em separado todas as palavras da linha de texto em um array. Assuma que o arquivo texto já existe, por exemplo, "**`arquivo.txt`**", e que está no mesmo diretório do programa a ser desenvolvido. Modularize o código.

   A tabela no final deste arquivo apresenta os padrões utilizados para análise de expressões regulares.

4. Assumindo que o arquivo texto disponível é um e-mail, crie um programa que avalie se o e-mail é um spam ou não. Para isso, defina alguns procedimentos para detecção de palavras ou expressões suspeitas. Saiba que tais palavras ou expressões podem ter sido propositalmente alteradas para não serem detectadas por sistemas anti-spam. O arquivo "**`spam.txt`**" abaixo é um exemplo de e-mail. Modularize o código.

********** **spam.txt** **********

Frete Grátis para todo o Brasil.
Pague sua Compra em até 6x sem juros no cartão.
Ofertas sujeitas a alteração de preço e/ou condições de pagamento sem aviso prévio. Troca ou devolução grátix de produtos em sua embalagem original lacrada e não danificada em até 30 dias corridos após o recebimento da compr@.

CENTRAL DE ATENDIMENTO DE COMPRA

De segunda a sexta, das 09h as 17h (exceto feriados).
E-mail: atendimento@maisspam.com

Para garantir que nossos comunicados cheguem em sua caixa de entrada, adicione o email atendimento@maisspam.com em sua lista de contatos. Sua c0mpra será simplificada.

Caso nao queira mais receber nossos e-mails, clique aqui.

5. Crie um módulo chamado **`spamDetector`** e use pelo menos uma das funções no programa da questão anterior. Não é necessário criar um módulo compilado.

Padrão	Resultado
.	corresponde a um único caractere, exceto newline
\s	corresponde a um caractere whitespace (espaço, tab, newline,...)
\S	corresponde a um caractere não-whitespace
\d	corresponde a um dígito (0-9)
\D	corresponde a um não-dígito
\w	corresponde a um caractere de palavra (a-z, A-Z, 0-9, _)
\W	corresponde a um caractere que não seja de palavra
[aeiou]	corresponde a um único caractere em um dado conjunto
[^aeiou]	corresponde a um único caractere fora de um dado conjunto
(foo|bar|baz)	corresponde a qualquer uma das alternativas especificadas
^	ancora a correspondência para o início de uma string
$	ancora a correspondência para o final de uma string
*	zero ou mais correspondências anteriores
+	um ou mais correspondências anteriores
?	zero ou uma correspondência anterior
{3}	exatamente 3 correspondências anteriores
{3,6}	entre 3 e 6 da correspondência anterior
{3,}	3 ou mais correspondências anteriores