universidade de aveiro deti departamento de electrónica, telecomunicações e informática

Programação I

Folha Exercícios 9
Strings

António J. R. Neves
João Rodrigues
Osvaldo Pacheco
Arnaldo Martins

2018/19

Folha Exercícios 9 - Strings

Resumo:

- **Strings** (sequências de caracteres)
- Classe **Character**; Métodos e operações com caracteres
- Classe **String;** Propriedades e métodos das **Strings**
- Leitura e escrita de **Strings** e caracteres
- **Strings** como argumentos de funções

Existem aplicações informáticas que, para além de necessitarem de processar dados numéricos, também necessitam de processar texto. Em programação, um texto pode ser visto como uma sequência de caracteres. Contudo, uma sequência de caracteres não é simplesmente um conjunto de caracteres pois estes têm particularidades e necessitam de um conjunto de operações específicas para a sua manipulação.

Em Java existe o tipo de dados referência **String** para a manipulação de texto. Este tipo de dados é descrito pela classe **String** que disponibiliza um vasto conjunto de funções para a sua manipulação.

A classe **Character** tem também um papel importante na manipulação de texto, dado que disponibiliza um vasto conjunto de funções para a manipulação de caracteres.

9.1 Problemas para resolver

Exercício 9.1

Pretende-se escrever um programa que leia do teclado uma frase e calcule alguma informação relativamente aos caracteres que a constituem.

1) O programa deve calcular e escrever o número de caracteres minúsculos, o número de caracteres maiúsculos e o número de caracteres numéricos. Pode recorrer aos métodos **Character.isUpperCase(), Character.isLowerCase()** e **Character.isDigit()**.
   A saída de dados deve obedecer ao formato seguinte:

```
Análise de uma frase
Frase de entrada ->########################################
Número de caracteres minúsculos -> ##
Número de caracteres maiúsculos -> ##
Número de caracteres numéricos -> ##
```

2) Crie uma função booleana **isVowel(char c)** para verificar se um carácter c é uma vogal e altere o programa anterior para contar também vogais e consoantes.

```
Número de vogais -> ##
Número de consoantes -> ##
```

Exercício 9.2

Faça uma função que devolva um acrónimo para um dado nome. Por exemplo:

acronimo("Organização das Nações Unidas") devolve **"ONU"**
acronimo("Universidade de Aveiro") devolve **"UA"**

Sugestão: extraia apenas as maiúsculas.

Implemente depois um programa que vá pedindo texto ao utilizador até que este introduza uma linha vazia e, para cada uma, escreva o seu acrónimo com base na função desenvolvida.

Exercício 9.3

Crie uma função que indique quantas palavras contém uma string. Considere que uma palavra é qualquer texto que não contenha espaços, tabs (\t) ou mudanças de linha (\n). Por exemplo:

countWords("isto é 1 frase ") devolve **4.**

Sugestão: pode percorrer a string símbolo a símbolo e manter uma variável que indica se está dentro ou fora de uma palavra. Se estiver "dentro" e aparecer um espaço (ou \t ou \n), passa a estar "fora". Se estiver "fora" e aparecer outro caráter, passa a estar "dentro" e conta mais uma palavra.

Exercício 9.4

As matrículas automóveis em Portugal podem ter um de três padrões: AA-00-00, 00-00-AA ou 00-AA-00.

Crie uma função booleana **matchPattern(str, pattern)** que indique se uma dada string **str** corresponde a um dado padrão **pattern**. Um A no padrão corresponde a qualquer letra maiúscula na string, um 0 corresponde a qualquer dígito, e outros carateres devem coincidir exatamente.

Invocando esta função, escreva um programa que verifique se um texto introduzido pelo utilizador é uma matrícula portuguesa válida.

Exercício 9.5

O número dezassete representa-se em base dez como "17", em base dois como "10001", e em base três como "122". Faça uma função que converta um número inteiro na sua representação numa qualquer base (para bases entre 2 e 10). Por exemplo, pretende-se que:

numToBase(17, 10) devolve **"17"**
numToBase(17, 2) devolve **"10001"**
numToBase(17, 3) devolve **"122"**

Escreva um programa que converte um número dado pelo utilizador nas suas representações em base 2, 3, … , 10.

Lembre-se que pode converter um número para outra base por divisão sucessiva pela base. O resto de cada divisão dá mais um dígito que deve ser concatenado à esquerda dos anteriores.

Exercício 9.6

Na terra do Alberto Alexandre (localmente conhecido por Aubeto Auexande), o dialeto local é semelhante ao português com duas exceções:

- não dizem os R's;
- trocam os L's por U's.

Implemente um tradutor de português para o dialeto do Alberto que permite a introdução de uma frase e apresente a sua tradução. Por exemplo "lar doce lar" deve ser traduzido para "ua doce ua".

Exercício 9.7

Crie uma função que coloque em maiúsculas a primeira letra de cada palavra de uma string:

capitalize("era uma vez") devolve **"Era Uma Vez"**.

Sugestão: pode usar a mesma "máquina de estados" do problema 9.3 para detetar o início de cada palavra e usar o método **Character.toUpperCase** para converter em maiúsculas.

Exercício 9.8

Faça uma função inversa da do problema 9.5. Por exemplo, pretende-se:

baseToNum("10001", 2)	devolve **17**
baseToNum("122", 3)	devolve **17**
baseToNum("17", 10)	devolve **17**

Repare que o primeiro parâmetro é uma string, mas o resultado é um **int** (pode converter o char '3' no valor 3 com a expressão '3'-'0').

Escreva um programa para testar a função. Lembre-se que um número em base *N* só pode ter dígitos na gama 0, …, *N*-1.