Programação I

Guião da Aula 12

António J. R. Neves

Departamento de Electrónica, Telecomunicações e Informática

Universidade de Aveiro

Ano letivo 2013/2014

Aula Prática 12

Resumo:
- arrays de Strings
- arrays de registos
- arrays bidimensionais

Vimos até aqui que podíamos armazenar numa variável do tipo array vários valores do mesmo tipo. Podemos, no entanto, também ter arrays de tipos referência, sendo que numa variável deste tipo podemos armazenar várias referências de um certo tipos de dados: arrays de Strings arrays de registos (classes) e arrays bidimensionais (arrays de arrays).

A criação deste tipo de arrays necessita de uma declaração semelhante à dos arrays de tipos primitivos, mas cada elemento dos array necessita depois de ser "criado" segundo as regras correspondentes ao tipo de dados em causa.

12.1 Problemas para resolver

Exercício 12.1

Crie um programa que faça uma cópia de um ficheiro de texto para outro ficheiro, mas ordenando o respetivo conteúdo. Essa ordenação deve seguir a regra dos dicionários (lexicográfica). Os nomes dos ficheiros devem ser pedidos ao utilizador e introduzidos através do teclado.

Nota: Tente fazer com que o programa seja robusto, não só detetando a existência do ficheiro original (apresentando uma mensagem de erro quando este não existe), como também a possível existência do ficheiro destino (neste caso fará sentido perguntar ao utilizador se de facto deseja destruir esse ficheiro). Deve também verificar se sobre os ficheiros se podem realizar as operações de leitura e escrita, e se algum deles é um directório (caso em que deve ser apresentada uma mensagem de erro).

Exercício 12.2

Pretende-se escrever um programa que permita o cálculo da soma e multiplicação de duas matrizes. Para tal, o programa começa por fazer a leitura das dimensões das matrizes, seguida da leitura do seu conteúdo. Deve ter-se em consideração as regras para a operações indicadas sobre matrizes.

Exercício 12.3

Pretende-se um programa que permita gerir a avaliação de uma turma de Programação I com no máximo 20 alunos. Para cada aluno deverá ser guardada a seguinte informação:

```
número mecanográfico - valor inteiro;
nome do aluno - texto livre;
notas das várias componentes - 4 valores inteiros no intervalo [0 ... 20];
nota final - valor real, no intervalo [0 ... 20].
```

O programa deve funcionar de forma repetitiva com base num menu de opções que a seguir se apresenta:

```
Gestão de uma turma:
1 - Inserir informação da turma
2 - Mostrar informação de um aluno
3 - Alterar informação de um aluno
4 - Listar os alunos ordenados por nº mec.
5 - Listar os alunos ordenados por nota final
6 - Média das notas finais
7 - Melhor aluno
8 - Terminar o programa
Opção ->
```

O programa deverá permitir as seguintes operações:

a. Introdução da informação associada aos alunos, terminando com a introdução do nº mec. 0. Toda a informação deverá ser pedida ao utilizador com a exceção da nota final que deverá ser calculada pelo programa, assumindo as percentagens de 10%, 30%, 10% e 50% respetivamente para as quatro notas. Nesta opção, a turma deve ser preenchida desde o inicio, ignorando os dados previamente introduzidos.

b. Mostrar informação sobre um aluno com base no nº mec., pedido ao utilizador. Deve informar o utilizador se o aluno não existir.

c. Alterar notas de um aluno cujo nº mec. é pedido ao utilizador. Se o aluno não existir, deve ser acrescentado à turma no caso de ainda não se encontrar preenchida.

d. Mostrar ao utilizador a informação sobre os alunos, ordenada por nº mec.

e. Mostrar ao utilizador a informação sobre os alunos, ordenada por nota final.

f. Calcular e imprimir a média das notas finais da turma.

g. Mostrar ao utilizador a informação sobre o melhor aluno.

Exercício 12.4

Tendo como base o programa desenvolvido no exercicio 1, pretende-se agora que escreva no monitor o número de vogais, o número de consoantes, o número de caracteres minúsculos, o número de caracteres maiúsculos, o número de caracteres numéricos e o número de palavras, considerando os espaços e os sinais de pontuação como separadores (use o código desenvolvido em aulas anteriores).

```
Frase a analisar? ##############################################
Frase a analisar? ###################################
. . .

Análise estatística

Número total de vogais -> ###
Número total de consoantes -> ###
Número total de caracteres minúsculos -> ###
Número total de caracteres maiúsculos -> ###
Número total de caracteres numéricos -> ###
Número total de palavras -> ###
```

Exercício 12.5

Pretende-se escrever um programa que leia do teclado uma lista de frases, até um máximo de 10 frases ou até ser lida a frase "fim" e escreva no monitor as frases lidas pela ordem inversa com que foram introduzidas. Não só deve escrever primeiro a última frase introduzida, como a frase deve aparecer ao contrário. Exemplo de utilização do programa:

```
Frase 1: ola
Frase 2: aveiro
Frase 3: P1
Frase 4: fim

Resultado:
1P
orieva
alo
```

Exercício 12.6

Pretende-se um programa que permita gerir uma biblioteca. Assume-se que existe um máximo de 200 livros e que se pretende manter para cada um deles a seguinte informação:

```
cota - código com 20 caracteres alfanuméricos;
autor - texto com um máximo de 40 caracteres;
título - texto com um máximo de 60 caracteres;
data de aquisição - constituída por dia, mês e ano;
estado atual do livro -  carácter (requisitado 'R', livre 'L' e condicionado 'C').
```

O programa deve funcionar de forma repetitiva com base num menu de opções que a seguir se apresenta:

```
Gestão de uma biblioteca
1 - Introduzir livros
2 - Remover um livro
3 - Apagar a base da dados
4 - Verificação de cotas repetidas
5 - Alterar o estado de um livro
6 - Listar os livros requisitados
7 - Listar os livros ordenados pela cota
8 - Listar os livros ordenados pela data
9 - Terminar o programa
Opção ->
```

O programa deverá permitir as seguintes operações:

a. Introdução de livros na biblioteca, terminando com a introdução de uma cota vazia. Nesta opção, a base de dados deve ser preenchida desde o inicio, ignorando os dados previamente introduzidos. Assuma que o utilizador introduz uma data válida e garanta apenas a validação dos restantes campos, segundo o enunciado.

b. Remover um livro com base na cota, pedida ao utilizador. Deve informar o utilizador se o livro não existir.

c. Apagar a base de dados.

d. Verificação das cotas existentes na base de dados e em caso de cotas repetidas pedir ao utilizador nova cota para o livro em causa. A ordem com que as cotas são analisadas não é relevante. No final desta ação devemos garantir que não temos dois livros com a mesma cota.

e. Para alterar o estado de um livro, começa-se por pedir a cota e, caso exista, em função do seu estado atual o livro pode ser libertado (se requisitado ou condicionado), requisitado (se livre) ou colocado em acesso condicionado (se livre).

f. Mostrar ao utilizador os livros requisitados.

g. Mostrar ao utilizador os livros ordenados por cota.

h. Mostrar ao utilizador os livros ordenados por data.

Exercício 12.7

Pretende-se um programa que permita gerir uma agenda. Assume-se que existe um máximo de 100 contactos e que se pretende manter para cada um deles a seguinte informação:

```
nome - texto livre;
morada - texto livre;
telefone - numero inteiro;
email - texto que signifique um email.
```

O programa deve funcionar de forma repetitiva com base num menu de opções que a seguir se apresenta:

```
Gestão de uma agenda:
1 - Inserir um contacto
2 - Pesquisar contacto por nome
3 - Eliminar um contacto
4 - Visualizar todos os contactos
5 - Visualizar contactos ordenados pelo nome
6 - Validar endereços de email
7 - Apagar a agenda
8 - Terminar o programa
Opção ->
```

O programa deverá permitir as seguintes operações:

a. Introdução de um novo contacto na agenda. Não é necessário fazer validações na introdução dos dados.

b. Mostrar informação sobre um contacto com base no nome, parcial ou completo, pedido ao utilizador. Deve informar o utilizador se o aluno não existir. Sugere-se a utilização da função indexOf da classe String.

c. Eliminar um contacto da agenda com base no número de telefone, pedido ao utilizador. Deve informar o utilizador se o contacto não existir.

d. Mostrar ao utilizador os contactos da agenda pela ordem com que foram introduzidos.

e. Mostrar ao utilizador os contactos ordenados pelo nome. Nesta operação deve ter o cuidado de manter a base de dados original, isto é, deve manter a ordem original de introdução.

f. Verificação dos endereços de email existentes na agenda e, em caso de erro, pedir ao utilizador novo email para o contacto em causa. Um email válido só pode conter caracteres alfanuméricos, um símbolo '@' e os símbolos '.' e '_'.

g. Apagar a agenda.