Sistemas e Sinais

Laboratório 0 (parte c)

(MATLAB®: *Cell arrays* e *structures*)

Miguel Pedro Silva e João Reis

Instituto Superior Técnico,
Dep. de Engenharia Mecânica - Secção Sistemas,
Grupo de Controlo Automação e Robótica

Exercício 1

- Escreva uma função que aceita um *cell array* de strings (cadeia de caracteres) e ordena-o de forma ascendente, de acordo com a ordem alfabética.

  Tenha atenção a situações do tipo ‘A’ e ‘a’, que devem ser consideradas como a mesma letra.

- Teste a função com um script.

Exercício 1 (cont.) solução

```
function [celOrd] = ordenaStrings(cel)
%Comentários ... help !!

%Testes à estrutura de entrada
if class(cel)~='cell',
    error('O argumento não é um cell array');
end;

[linhas_cel, colunas_cel] = size(cel);
if (linhas_cel > 1 && colunas_cel > 1)
    error('O cell array deve ser um vector!');
end

if (linhas_cel == 0 || colunas_cel == 0)
    error('O cell array deve ser não-vazio!');
end
```

Exercício 1 (cont.) solução

```
for k=1:length(cel)
    if (strcmp(class(cel{k}), 'char')~=true)
      error('Todos os elementos do cell array devem ser um array de chars!');
    end
end

%Ordenação
[celOrd_aux, indices] = sort(lower(cel));

for k=1:length(cel)
    celOrd{k}=cel{indices(k)};
end
```

Exercício 1 (cont.) *script* de teste

```
% script_ex_1.m
% Testa ordenaStrings()
clc;

% Introduza o número de strings a ordenar
nstr = input('Introduza o número de strings a ordenar:  ');

% Pré-alocação de células
celula = cell(1,nstr); %opcional

for k = 1:nstr
   string = ['Introduza a string nº ' int2str(k) ':  '];
   celula{k} = input(string,'s');
end
```

Exercício 1 (cont.) *script* de teste

```
% Ordena string
stringsOrdenadas = ordenaStrings(celula);

% Apresenta os resultados
fprintf('\nStrings ordenadas:\n');
for k = 1:nstr
   fprintf(' %s\n', stringsOrdenadas{k});
end
```

Exercício 2

- Crie uma função que aceita qualquer número de argumentos de entrada numéricos e soma todos os elementos dos argumentos.

- Teste a função com um *script* de teste.
  - Utilize:

```
a = 1;
b = [1; 1; 1];
c = [1 1 1; 1 1 1; 1 1 1];
d = [1 1 1 1];
```

Exercício 2 (cont.) solução

```
function res = soma_args(varargin)
% SOMA_ARGS recebe um nº arbitrário de args, e soma os valores de
% todos os args. de entrada (todos os elementos)
% Cada elemento pode ser um array de dimensão arbitrária
%
% Chamada: res = soma_args(arg1, arg2, ...);

% Obter o nº de args a somar
n_args = nargin;

% Soma de argumentos
res = 0;
```

Exercício 2 (cont.) solução

```
for k = 1:n_args
   % Obter a dimensão do array do argumento k.
   n_dim = length(size(varargin{k})); %obter o número de linhas
   temp = varargin{k};
   for j = 1:n_dim
      temp = sum(temp);
   end
   % Soma este argumento ao total
   res = res + temp;
end

end % end opcional - fim de função
```

Exercício 2 (cont.) *script* de teste

```
% script_ex_2.m
% Testa soma_args()

clear all;
clc;
close all;
a = 1;
b = [1; 1; 1];
c = [1 1 1; 1 1 1; 1 1 1];
d = [1 1 1 1];

% Calcula a soma de todos os elementos dos argumentos
total = soma_args(a,b,c,d);
fprintf('A soma de todos os elementos dos argumentos é %.2f\n', total);
```

Exercício 3

- Defina a estrutura *ponto* contendo dois campos, x e y. O campo x, contém a posição x do ponto e o campo y a posição y do ponto.

- Escreva uma função *dist* que aceita dois pontos, e retorna a distância entre eles no plano cartesiano. Verifique o número de argumentos de entrada da sua função.

- Teste a função com um *script*.

Exercício 3 (cont.) solução

```
function distancia = dist(ponto1, ponto2)

% A função DIST aceita dois pontos representados
% num estrutura e calcula a distância entre eles.
% Os campos da estrutura que representa o ponto
% devem ser os seguintes:
%
% x          -- posição x do ponto
% y          -- posição y do ponto
%
% Chamada: distancia = dist(ponto1, ponto2);

% Verifica o nº de argumentos
msg = nargchk(2,2,nargin);
error(msg);
```

Exercício 3 (cont.) solução

```
% Verifica tipo dos argumentos
if ((strcmp(class(ponto1),'struct')~=true) || ...
        (strcmp(class(ponto2),'struct')~=true))
    error('Os dois argumentos devem ser estruturas');
end

% Calcula a distância
distancia = sqrt( (ponto1.x - ponto2.x) .^2 ...
          + (ponto1.y - ponto2.y) .^2 );
```

Exercício 3 (cont.) *script* de teste

```
% script_ex_3.m
% Testa dist()

clear all;
clc;
close all;

% Cria as estruturas dos pontos
p1.x = 3.5;
p1.y = 2;
p2.x = -2;
p2.y = 0;

% Calcula a distancia
d = dist(p1,p2);

% Apresenta o resultado
fprintf('Ponto 1 = (%.4f,%.4f) \n', p1.x, p1.y);
fprintf('Ponto 2 = (%.4f,%.4f) \n', p2.x, p2.y);
fprintf('Distância = %.4f\n', d);
```

Exercício 3 (cont.) *script* de teste

```
%continuação
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Carrega a estrutura pontos
load pontos.mat

% Calcula a distancia
d = dist(pontos.p1,pontos.p2);

% Apresenta o resultado
fprintf('Ponto 1 = (%.4f,%.4f) \n', pontos.p1.x, pontos.p1.y);
fprintf('Ponto 2 = (%.4f,%.4f) \n', pontos.p2.x, pontos.p2.y);
fprintf('Distância = %.4f\n',d);
```

Exercício 3 (cont.) solução

```
%continuação
%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Cria um array de estruturas
p(1).x = 3.5;
p(1).y = 2;
p(2).x = -2;
p(2).y = 0;

% Calcula a distancia
d = dist(p(1),p(2));

% Apresenta o resultado
fprintf('Ponto 1 = (%.4f,%.4f) \n', p(1).x, p(1).y);
fprintf('Ponto 2 = (%.4f,%.4f) \n', p(2).x, p(2).y);
fprintf('Distância = %.4f\n',d);
```

Exercício 4

- Considere um cell array, denominado OsMeusCDs, que possui em cada posição uma estrutura com os seguintes dados de um CD:
  - Genero
  - Artista
  - Titulo
  - Ano
- Escreva uma função, denominada *procuraCDs*, para procurar neste cell array todos os CDs mais recentes que um dado ano (inclusivé) e que retorne os seus titulos num novo cell array.

Exercício 4 (cont.) solução

```
function celula = procuraCDs(coleccaoCDs, anoAprocurar )
% comentários ...

c ={};

indice = 1;

for k = 1:length(coleccaoCDs)

    umCD = coleccaoCDs{k};

    if umCD.ano >= anoAprocurar
       celula{indice} = umCD.titulo;
       indice = indice + 1;
    end

end
```

Exercício 4 (cont.) *script* de teste

```
% script_ex_4.m
% Testa procuraCDs()

clear all;
clc;
close all;

% Carrega o cell array composto por estruturas
load OsMeusCDs.mat

% Pede ano e faz a procura
var_ano = input('Introduza o ano a partir do qual quer os resultados: ');
res = procuraCDs(OsMeusCDs, var_ano);

% Apresenta o resultado
disp(['Titulo dos CDs gravados desde o ano ' int2str(var_ano) ': ' ]);
disp(res);
```