Aula prática 12

Vetores: Primeira Parte

Resumo

As atividades propostas nesta prática visam explorar as primeiras noções sobre vetores para o desenvolvimento de aplicações.

Sumário

1 Vetores e matrizes

A unidade fundamental de dados em Scilab é a **matriz**. Uma matriz é semelhante a uma tabela, exceto que uma matriz pode ter qualquer número de dimensões, enquanto uma tabela geralmente tem apenas duas dimensões, que usualmente são chamadas de linhas e colunas.

Por exemplo, as matrizes A, B e C, a seguir, têm dimensões 1×5, 3×1 e 3×2, respectivamente:

$$A = \begin{array}{|c|c|c|c|c|} \hline 3 & 5 & 7 & 12 & 9 \\ \hline \end{array} \qquad B = \begin{array}{|c|} \hline 8 \\ \hline 5 \\ \hline 7 \\ \hline \end{array} \qquad C = \begin{array}{|c|c|} \hline 3 & 8 \\ \hline 12 & 5 \\ \hline 9 & 7 \\ \hline \end{array}$$

A matriz A, que tem apenas 1 linha, é também chamada de **vetor-linha**, ou simplesmente **vetor**. A matriz B, que tem apenas 1 coluna, é também chamada de **vetor-coluna**.

Em Scilab, o valor denotado por uma variável é sempre uma matriz. Em particular, um valor escalar, tal como 2, 11.3 ou `%pi` é visto como uma matriz de dimensão 1×1.

1.1 Criando Matrizes

Uma matriz pode ser especificada simplesmente enumerando os seus elementos. Em Scilab, os elementos de uma matriz devem ser especificados entre colchetes; elementos de uma mesma linha são separados por espaço ou por vírgula (`,`), e as linhas são separadas por mudança-de-linha ou por ponto-e-vírgula (`;`).

Os exemplos a seguir ilustram os comandos Scilab para criação das 3 matrizes `A`, `B` e `C` mostradas anteriormente. Digite cada um desses comandos no *p*rompt do console do Scilab e observe o resultado. Note também, na janela de variáveis, as dimensões das variáveis `A`, `B` e `C` criadas por estes comandos.

```
-->A = [ 3  2*8-5  -7 \%pi ]
 A =

     3.    11.    -7.    3.1415927

-->B = [0.5; 2; 10.3]
 B  =

      0.5
      2.
      10.3

-->C = [ 10, 2, 5; 3, 2.3, 0.6]
 C  =

     10.    2.      5.
     3.      2.3    0.6

-->D = [ 10, 2, 5; 3, 7.2]
 D  =
                        !-- error 6
Inconsistent row/column dimensions
```

Observações

- O valor de um elemento da matriz pode ser especificado como uma expressão arbitrária.
- Cada linha da matriz deve ter o mesmo número de elementos, e cada coluna deve ter o mesmo número de linhas.

1.2 Indexando matrizes

Uma posição em uma matriz `M` de dimensão $n \times m$ é identificada por um par de *índices* (i, j), onde i é o número (ou índice) da linha e j é o número (ou índice) da coluna desta posição na matriz. Por exemplo, `M(2,3)` especifica o elemnto na linha 2, coluna 3 da matriz `M`. Esse mecanismo de *indexação* é ilustrado a seguir.

```
--> A = [10, 20, 30]
 A  =

     10.     20.      30.

---> x = A(1,3)
 x =

     30.
```

A indexação de uma matriz deve ser feita de modo que os valores dos índices estejam compreeendidos dentro dos limtes da dimensão dessa matriz. Isto é, se `M` é uma matriz de dimensão $m \times n$, `M(i,j)` deve ser tal que $1 \leq$ `i` $\leq n$ e $1 \leq$ `j` $\leq m$. Caso contrário, é reportado um erro de índice inválido, como ilustra o exemplo a seguir.

```
-->z = A(2,3)           // erro de indexação: A tem apenas 1 linha
               ! error 21
Invalid index
```

A indexação em um vetor-linha pode ser feita omitindo-se o número da linha e especificando apenas o número da coluna do elemento desejado, conforme ilustrado a seguir. De maneira similar, a indexação em um vetor-coluna pode ser feita omitindo-se o número da coluna e especificando apenas o número da linha do elemento desejado.

```
--> A = [10, 20, 30]
 A  =

     10.     20.      30.

---> y = A(2)
 x =

     20.


---> A(3) = 45
A  =

     10.     20.      45.
```

Observações

- Um índice de linha ou coluna em uma matriz deve ser um valor inteiro positivo, e pode ser especificado como uma expressão arbitrária.

1.3 Obtendo as dimensões de uma matriz

As seguintes funções podem ser usadas, em Scilab, para determinar o número de elementos ou as dimensões de uma matriz.

função	**descrição**
n = length(A)	número de elementos de A
[l,c] = size(A)	número de linhas e colunas de A

Exemplo:

```
-->A = [ 1 8 9 5; 2 3 4 0 ]
 A  =

    1.    8.    9.    5.
    2.    3.    4.    0.

-->dim = size(A)
 dim  =

      2.    4.

-->[lin,col] = size(A)
 col  =

    4.
 lin  =

    2.

-->length(A)
 ans  =

     8.

-->length(A(1,:))
 ans  =

      4.

-->length(A(:,2))
ans  =

      2.
```

Observações

- O resultado da expressão `(size(M)`, onde `M` é uma matriz de n dimensões, ($n \geq 2$) consiste de um vetor de n valores.
- O resultado de uma expressão `(size(M)` pode ser atribuído a uma variável, tal como na atribuição `dim = size(C)` acima.
- O resultado de uma expressão `(size(M)`, onde `M` é uma matriz de n dimensões, pode também ser atribuído a um vetor de n variáveis `[`d_1`,` d_2`, ...,` d_n`]`, sendo que cada variável d_i, para $1 \leq i \leq n$, irá conter o número de elementos da dimensão i da matriz (tal como no último exemplo acima).

2 Exemplo de aplicação

Escrever um programa em Scilab para ler uma sequência de notas digitadas pelo usuário, e calcular e exibir a média aritmética das notas, bem como a quantidade de notas que estão acima desta média. A entrada deve ser encerrada quando o usuário digitar um valor negativo.

Exemplo de execução da aplicação

```
Cálculo da média
--------------------------
digite a nota: 6.8
digite a nota: 9.0
digite a nota: 8.1
digite a nota: 4.5
digite a nota: 7.0
digite a nota: 1.9
digite a nota: 6.0
digite a nota: 10.0
digite a nota: 6.8
digite a nota: 7.5
digite a nota: -1

média: 6.76
7 notas estao acima da média
```

```
clc;
clear;
printf("Cálculo da média\n");
printf("--------------------------\n");

// entrada das notas
quant = 0;
vet = [];             // vetor nulo
nota = input("digite a nota: ");
while nota >= 0 do
    quant = quant + 1;
    vet(quant) = nota;
    nota = input("digite a nota: ");
end

// foi digitada alguma nota válida?
if quant > 0 then
    // cálculo da média
    soma = 0;
    for i = 1:quant do
        soma = soma + vet(i);
    end
    media = soma / quant;
    printf("\nmédia: %g\n", media);

    // contar as notas acima da média
    acima_media = 0;
    for i = 1 : quant do
        if vet(i) > media then
            acima_media = acima_media + 1;
        end
    end
    printf("%g notas estao acima da média\n", acima_media);
else
    printf("nenhuma nota foi digitada!\n");
end
```

3 Exercícios

Tarefa 1: Entrada e saída de vetores

Faça um programa que leia um vetor com cinco posições para números reais e, depois, um código inteiro. Se o código for zero, finalize o programa; se for 1, mostre o vetor na ordem direta; se for 2, mostre o vetor na ordem inversa.

Exemplo de execução da aplicação

```
digite um elemento: 2
digite um elemento: 4
digite um elemento: 0
digite um elemento: 8
digite um elemento: 5

digite o código (0, 1 ou 2): 2

elementos do vetor em ordem inversa:
5 8 0 4 2
```

Solução:

```
clc;
clear;

n = 5;

for i = 1:n do
    vet(i) = input("digite um elemento: ");
end

printf("\n");
codigo = input("digite o código (0, 1, ou 2): ");
printf("\n");

if codigo == 1 then
    printf("elementos do vetor em ordem direta:\n");
    for i = 1:n do
        printf("%g ", vet(i));
    end
elseif codigo == 2 then
    printf("elementos do vetor em ordem inversa:\n");
    for i = n:-1:1 do
        printf("%g ", vet(i));
    end
elseif codigo <> 0 then
    printf("código inválido");
end
```

Tarefa 2: Cálculo com elementos de vetores

Codifique um programa que preencha dois vetores de seis elementos através de entradas pelo teclado. Após a definição dos dois vetores, construa um terceiro vetor onde cada elemento é igual ao dobro da soma entre os elementos correspondentes dos outros dois vetores. Imprima o conteúdo do vetor calculado.

Exemplo de execução da aplicação

```
digite os elementos do primeiro vetor:
digite um elemento: 1
digite um elemento: 2
digite um elemento: 3
digite um elemento: 4

digite os elementos do segundo vetor:
digite um elemento: 5
digite um elemento: 6
digite um elemento: 7
digite um elemento: 8

elementos do vetor calculado:
12 16 20 24
```

Solução:

```
clc;
clear;

n = 4;

printf("digite os elementos do primeiro vetor:\n");
for i = 1:n do
    u(i) = input("digite um elemento: ");
end

printf("\n");
printf("digite os elementos do segundo vetor:\n");
for i = 1:n do
    v(i) = input("digite um elemento: ");
end

for i = 1:n do
    w(i) = 2*(u(i) + v(i));
end

printf("\n");
printf("elementos do vetor calculado:\n");
for i = 1:n do
    printf("%g ", w(i));
end
```

Tarefa 3: Contagem de números negativos, nulos e positivos

Codificar um programa Scilab que leia um vetor de n valores. A seguir, o programa determina e exibe quantos elementos são nulos, positivos e negativos.

A quantidade n de elementos no vetor é determinada pelo usuário.

Exemplo de execução da aplicação

```
Digite a quantidade de elementos do vetor:5
Início da leitura dos elementos do vetor...
Elemento 1:
  digite o valor --> 10
Elemento 2:
  digite o valor --> 20
Elemento 3:
  digite o valor --> 30
Elemento 4:
  digite o valor --> 0
Elemento 5:
  digite o valor --> -13

Vetor original:
10  20  30  0  -13

Elementos nulos     --> 1
Elementos positivos --> 3
Elementos negativos --> 1
```

Solução:

```
clc;
clear;
n = input("Digite a quantidade de elementos do vetor: ");
printf("Início da leitura dos elementos do vetor...\n");
for i = 1 : n
    printf("Elemento %g: ", i);
    vetor(i) = input("  digite o valor --> ");
end
// IMPRESSÃO DO VETOR ORIGINAL
printf("\nVetor original:\n");
for i = 1 : n
    printf("%g  ", vetor(i));
end
//
contNulo = 0;
contPos = 0;
contNeg = 0;
for i = 1:n
    if vetor(i) == 0 then
        contNulo = contNulo + 1;
    elseif vetor(i) < 0 then
        contNeg = contNeg + 1;
    else     // É POSITIVO
        contPos = contPos + 1;
    end
end
printf("\n\nElementos nulos     --> %g", contNulo);
printf("\nElementos positivos --> %g", contPos);
printf("\nElementos negativos --> %g", contNeg);
```

Tarefa 4: Temperaturas máximas

As temperaturas máximas diárias (em ^{o}F) para Chicago e São Francisco durante o mês de agosto de 2009 são dadas nos vetores abaixo (dados da Administração Nacional Oceânica e Atmosférica dos EUA).

```
TCH = [75 79 86 86 79 81 73 89 91 86 81 82 86 88 89 ...
       90 82 84 81 79 73 69 73 79 82 72 66 71 69 66 66];

TSF = [69 79 70 73 72 71 69 76 85 87 74 84 76 68 79 ...
       75 68 68 81 72 79 68 68 69 71 70 89 95 90 66 69];
```

Escreva um programa para determinar quantos dias, e em que datas, no mês dado, a temperatura foi a mesma nas duas cidades.

Observações:

- Exemplificando, no quinto dia de agosto a temperatura em Chicago foi de 79^{o}F, e em São Francisco foi de 72^{o}F.
- Os vetores `TCH` e `TSF` já estão definidos. O usuário não precisa fazer nenhuma entrada de dados.

Exemplo de execução da aplicação

```
Datas em que ocorreram a mesma temperatura:  2 19 30
Quantidade de dias que ocorreram a mesma temperatura: 3
```

Solução:

```
clear; clc;

TCH = [75 79 86 86 79 81 73 89 91 86 81 82 86 88 89 ...
       90 82 84 81 79 73 69 73 79 82 72 66 71 69 66 66];

TSF = [69 79 70 73 72 71 69 76 85 87 74 84 76 68 79 ...
       75 68 68 81 72 79 68 68 69 71 70 89 95 90 66 69];

printf("Datas em que ocorreram a mesma temperatura: ");

contador = 0;

for i = 1 : length(TCH)
    if TCH(i) == TSF(i) then
        printf(" %d", i);
        contador = contador + 1;
    end
end

printf("\nQuantidade de dias que ocorreram a mesma temperatura: %g\n", ...
       contador);
```