

BCC701 – Programação de Computadores I

Universidade Federal de Ouro Preto

Departamento de Ciência da Computação

www.decom.ufop.br/bcc701

2015/1



Semana 10:

Vetores.

Material Didático Unificado.

Agenda

- Introdução;
- Declaração de vetores;
- Algumas operações com vetores;
- Algumas funções aplicadas a vetores;
- Exercícios.

Introdução;

Declaração de vetores;

Algumas operações com vetores;

Algumas funções aplicadas a vetores;

Exercícios.

Introdução

Conjunto de variáveis

- Em determinadas situações é necessário utilizar várias variáveis, por exemplo:

- Para armazenar três notas de um aluno:

- `Nota1 = input('Digite a nota 1: ');`

- `Nota2 = input('Digite a nota 2: ');`

- `Nota3 = input('Digite a nota 3: ');`

- Ler e imprimir cinco números:

- `for i = 1 : 5`

- `Num = input('Digite um numero: ');`

- `printf('Numero digitado: %g', num);`

- `end`

Conjunto de variáveis

• Em determinadas situações é necessário utilizar várias variáveis, por exemplo:

• Para armazenar três notas de um aluno:

- `Nota1 = input('Digite a nota 1: ');`
- `Nota2 = input('Digite a nota 2: ');`
- `Nota3 = input('Digite a nota 3: ');`

E se de repente for necessário manipular 10 notas de alunos?

• Ler e imprimir cinco números:

• `for i = 1 : 5`

```
Num = input('Digite um numero: ');  
printf('Numero digitado: %g', Num);  
end
```

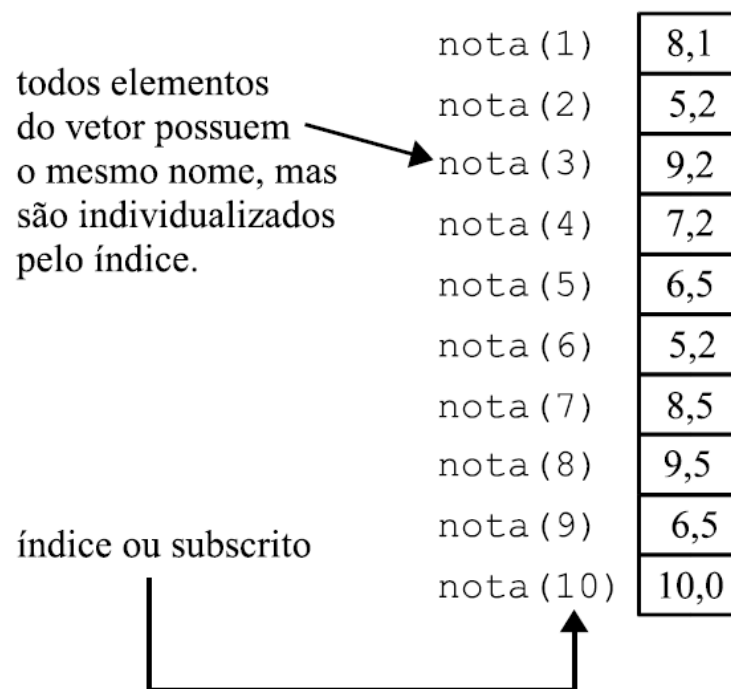
E se de repente for necessário manipular os números digitados depois do for?

O tipo de dados **Vetor**

- Nestes casos, todas as variáveis representam um conjunto de valores, possuem um objetivo em comum e são do mesmo tipo de dados;
- Uma estrutura de dados muito utilizada para armazenar e manipular este tipo de conjunto de variáveis é o **Vetor**;
- Um vetor representa conjuntos ordenados de valores homogêneos (do mesmo tipo), que podem ser números, *strings* e booleanos;
 - A palavra ordenado é empregada no sentido dos valores estarem localizados em posições ordenadas de memória, e não no sentido de estarem respeitando uma relação ($<$, $<=$, $>$, ou $>=$).

O tipo de dados Vetor

- Os itens contidos em um vetor são chamados de **elementos**;
- A posição do elemento no vetor é chamado de **índice** ou subscrito, e é usado para individualizar um elemento do vetor;
- O vetor `nota = [8.1 5.2 9.2 7.2 6.5 5.2 8.5 9.5 6.5 10.0]`,
- pode ser representado
- na memória como
- uma sequência de
- variáveis distintas,
- com o mesmo nome,
- mas diferenciadas
- pelo índice:



Introdução;

Declaração de vetores;

Algumas operações com vetores;

Algumas funções aplicadas a vetores;

Exercícios.

Declaração de vetores

Definindo todos os elementos

- Utiliza-se colchetes para delimitar todos os elementos;
- Para **vetores de colunas**, utiliza-se espaço ou vírgula para separar os elementos:

- Exemplos:

V1 = [1 2 3 4 5];

V2 = [5,4,3,2,1];

- Resultados (para V1 e V2 respectivamente):

1. 2. 3. 4. 5.

-->

5. 4. 3. 2. 1.

-->

Definindo todos os elementos

- Para **vetores de linhas**, utiliza-se ponto-e-vírgula para separar os elementos:

- Exemplo:

V3 = [1;2;3;4;5];

- Resultado:

1.

2.

3.

4.

5.

-->

Definindo elementos por faixas

- Semelhante à definição dos valores de um laço for:

Vetor = <valor inicial> : <incremento> : <valor final>

- Exemplos:

V4 = 1:5; // o incremento de 1 é opcional

V5 = 5:-1:1;

- Resultados (para V4 e V5 respectivamente):

1. 2. 3. 4. 5.

-->

5. 4. 3. 2. 1.

-->

Definindo vetor de 1's

- Todos os elementos assumirão valor inicial **1**;

Vetor = ones(<linhas>, <colunas>)

- **Vetor**: nome da variável do tipo vetor;
- **ones**: função que retorna uma **matriz*** com valores 1;
- **<linhas>**: número de linhas;
- **<colunas>**: número de colunas;
- *** Matriz** é objeto de estudos do próximo tópico abordado na disciplina;
- Para construir um **vetor**, o número de linhas OU o número de colunas deve ser igual a **um**.

Definindo vetor de 1's

- Exemplos:

- Vetor de colunas (cinco colunas):**

- `c = ones(1, 5)`

- `c = 1. 1. 1. 1. 1.`

-->

- Vetor de linhas (cinco linhas):**

- `c = ones(5, 1)`

- `c = 1.
1.
1.
1.
1.`

-->

Definindo vetor de 0's

- Todos os elementos assumirão valor inicial **0**;

Vetor = zeros(<linhas>, <colunas>)

- **Vetor**: nome da variável do tipo vetor;
- **zeros**: função que retorna uma **matriz*** com valores 0;
- **<linhas>**: número de linhas;
- **<colunas>**: número de colunas;
- *** Matriz** é objeto de estudos do próximo tópico abordado na disciplina;
- Para construir um **vetor**, o número de linhas OU o número de colunas deve ser igual a **um**.

Definindo vetor de 0's

- Exemplos:

- Vetor de colunas (cinco colunas):**

- c = zeros(1, 5)

- c = 0. 0. 0. 0. 0.

-->

- Vetor de linhas (cinco linhas):**

- c = zeros (5, 1)

- c =
0.
0.
0.
0.
0.

-->

Introdução;
Declaração de vetores;
Algumas operações com vetores;
Algumas funções aplicadas a vetores;
Exercícios.

Algumas Operações com vetores

Acesso aos elementos

- Para acessar um elemento específico:

Vetor(<índice>)

- Exemplo:

```
clc;
```

```
V = [10, 20, 30, 40, 50];
```

```
disp(V(3));
```

- Resultado:

30.

-->

- Pode ser aplicado tanto a vetor de coluna quanto de linha:
 - Para **vetor de coluna**, <índice> corresponde ao **número da coluna**;
 - Para **vetor de linha**, <índice> corresponde ao **número da linha**;
- Pode ser usado para modificar o valor: **V(3) = 300**, modifica o valor do **índice 3** de **30** para **300**.

Acesso aos elementos

- Com o acesso a elementos específicos é possível definir o vetor, no exemplo anterior:

```
clc;
```

```
V(1) = 10; V(2) = 20; V(3) = 30; V(4) = 40; V(5) = 50;
```

```
disp(V);
```

- Resultado:

```
10.
```

```
20.
```

```
30.
```

```
40.
```

```
50.
```

```
-->
```

- Note que o resultado é um vetor de linhas.

Transposição de vetores

- **Operador apóstrofo ('):** Vetor'

- Transforma um vetor de linha em um vetor de coluna, e
- vice-versa:

```
clc;  
V = [10, 20, 30, 40, 50];  
V = V';  
disp(V);
```

- Resultado:

```
10.  
20.  
30.  
40.  
50.
```

-->

- Também poderia ser feito: $V = [10:10:50]'$, para obter o mesmo resultado anterior;

Operações binárias

- **Soma com escalar:** $V + \langle \text{valor} \rangle$ OU $\langle \text{valor} \rangle + V$:

- Exemplo:

```
clc;
```

```
V1 = [1 2 3 4 5];
```

```
V2 = V1 + 2; // Ou V2 = 2 + V1; (causará o mesmo efeito)
```

```
disp(V2);
```

- Resultado:

```
3. 4. 5. 6. 7.
```

```
-->
```

Operações binárias

- **Soma de vetores:** $V1 + V2$:

- $V1$ e $V2$ devem ser da mesma dimensão;

- Exemplo:

```
clc;
```

```
V1 = [1 2 3 4 5];
```

```
V2 = [5 4 3 2 1];
```

```
Soma = V1 + V2;
```

```
disp(Soma);
```

- Resultado:

```
6. 6. 6. 6. 6.
```

```
-->
```

Operações binárias

- **Subtração com escalar:** Vetor - <valor> OU <valor> - Vetor:

- Exemplo:

```
clc;  
V1 = [1 2 3 4 5];  
V2 = V1 - 1;  
V3 = 1 - V1;  
disp(V2);  
disp(V3);
```

- Resultado:

```
0.  1.  2.  3.  4.  
0. -1. -2. -3. -4.
```

-->

Operações binárias

- **Subtração de vetores: $V1 - V2$:**

- $V1$ e $V2$ devem ser da mesma dimensão;
- Exemplo:

```
clc;  
V1 = [1 2 3 4 5];  
V2 = [5 4 3 2 1];  
Subtracao = V1 - V2;  
disp(Subtracao);
```

- Resultado:

- 4. - 2. 0. 2. 4.

-->

Operações binárias

- **Multiplicação por escalar:** $V * \langle \text{valor} \rangle$ OU $\langle \text{valor} \rangle * V$:

- Exemplo:

```
clc;
```

```
V1 = [1 2 3 4 5];
```

```
V2 = V1 * 2; // Ou V2 = 2 * V1; (causará o mesmo efeito)
```

```
disp(V2);
```

- Resultado:

```
2. 4. 6. 8. 10.
```

```
-->
```

Operações binárias

- **Multiplicação de vetores: $V1 .* V2$:**

- V1 e V2 devem ser da mesma dimensão;
- Exemplo:

```
clc;  
V1 = [1 2 3 4 5];  
V2 = [5 4 3 2 1];  
Mult = V1 .* V2;  
disp(Mult);
```

- Resultado:

5. 8. 9. 8. 5.

-->

Operações binárias

- **Produto interno:** $V1 * V2$:

- V1 é um vetor de colunas e V2 é um vetor de linhas;
- O número de colunas de V1 deve ser igual ao número de linhas de V2;
- Exemplo:

```
clc;
```

```
V1 = [1 2 3 4 5]; // Vetor de colunas
```

```
V2 = [5;4;3;2;1]; // Vetor de linhas
```

```
Mult = V1 * V2;
```

```
disp(Mult);
```

- Resultado:

35.

-->

Operações binárias

- **Divisão por escalar:** $V / \langle \text{valor} \rangle$ OU $\langle \text{valor} \rangle \setminus V$:

- Exemplo:

```
clc;
```

```
V1 = [1 2 3 4 5];
```

```
V2 = V1 / 2; // Ou V2 = 2 \ V1; (causará o mesmo efeito)
```

```
disp(V2);
```

- Resultado:

```
0.5  1.  1.5  2.  2.5
```

```
-->
```

Operações binárias

- **Divisão de vetores à direita: $V1 ./ V2$:**

- $V1$ e $V2$ devem ser da mesma dimensão;
- Exemplo:

```
clc;  
V1 = [1 2 3 4 5];  
V2 = [5 4 3 2 1];  
Div = V1 ./ V2;  
disp(Div);
```

- Resultado:

0.2 0.5 1. 2. 5.

-->

Operações binárias

- **Divisão de vetores à esquerda: $V1 .\ V2$:**

- V1 e V2 devem ser da mesma dimensão;
- Exemplo:

```
clc;  
V1 = [1 2 3 4 5];  
V2 = [5 4 3 2 1];  
Div = V1 .\ V2;  
disp(Div);
```

- Resultado:

5. 2. 1. 0.5 0.2

-->

Operações binárias

- Para mais informações, procure pelos operadores do scilab:
 - Soma (*plus*: +):
 - http://help.scilab.org/docs/5.3.3/pt_BR/plus.html
 - Subtração (*minus*: -):
 - http://help.scilab.org/docs/5.3.3/pt_BR/minus.html
 - Multiplicação (*star*: *):
 - http://help.scilab.org/docs/5.3.3/pt_BR/star.html
 - Divisão (*slash*: \ e *backslash*: /):
 - http://help.scilab.org/docs/5.3.3/pt_BR/slash.html
 - http://help.scilab.org/docs/5.3.3/pt_BR/backslash.html

Introdução;
Declaração de vetores;
Algumas operações com vetores;
Algumas funções aplicadas a vetores;
Exercícios.

Algumas funções aplicadas a vetores

Dimensão de vetores

[resultado] = length(<Vetor>)

- Retorna a quantidade de elementos do vetor, muito útil para construir laços de repetição para percorrer os elementos do vetor:

```
clc;  
V = [10, 20, 30, 40, 50];  
n = length(V);  
disp(n);
```

- Resultado:

5.

-->

Somatório

[resultado] = sum(<Vetor>)

- Retorna o somatório de todos os elementos do vetor:

```
clc;
```

```
V = [10, 20, 30, 40, 50];
```

```
somatorio = sum(V);
```

```
disp(somatorio);
```

- Resultado:

150.

-->

- Perceba que o resultado é um valor numérico.

Somatório cumulativo

[resultado] = cumsum(<Vetor>)

- Retorna o somatório de todos os elementos do vetor, de forma acumulativa a cada linha/coluna:

```
clc;
```

```
V = [10, 20, 30, 40, 50];
```

```
somatorio = cumsum(V);
```

```
disp(somatorio);
```

- Resultado:

```
10.  30.  60.  100.  150.
```

```
-->
```

- Perceba que o resultado é um vetor.

Produtório

[resultado] = prod(<Vetor>)

- Retorna o produtório de todos os elementos do vetor:

```
clc;
```

```
V = [10, 20, 30, 40, 50];
```

```
produtorio = prod(V);
```

```
disp(produtorio);
```

- Resultado:

```
12000000.
```

```
-->
```

- Perceba que o resultado é um valor numérico.

Produtório cumulativo

[resultado] = cumprod(<Vetor>)

- Retorna o produtório de todos os elementos do vetor, de forma acumulativa a cada linha/coluna:

```
clc;
```

```
V = [10, 20, 30, 40, 50];
```

```
produtorio = cumprod(V);
```

```
disp(produtorio);
```

- Resultado:

```
10.    200.    6000.   240000.  12000000.
```

```
-->
```

- Perceba que o resultado é um vetor.

Elementos únicos

[resultado [, k]] = unique(<Vetor>)

- Retorna um vetor ordenado contendo os elementos únicos de um vetor, adicionalmente retorna um vetor com os índices dos elementos no vetor de entrada:

```
clc;
```

```
V= [60, 30, 40, 50, 20, 20, 30, 10, 70, 80];
```

```
[unicos, indices] = unique(V);
```

```
disp(unicos);
```

```
disp(indices);
```

- Resultado:

10.	20.	30.	40.	50.	60.	70.	80.	← Elementos únicos de V
8.	5.	2.	3.	7.	1.	9.	10.	← Índices dos elementos em V

```
-->
```

União

[resultado [, kA, kB]] = union(<Vetor A>, <Vetor B>)

- Retorna um vetor ordenado contendo a união entre os elementos de dois vetores, adicionalmente retorna vetores com os índices dos elementos em cada vetor de entrada:

clc;

V1 = [60, 30, 40, 50, 20];

V2 = [20, 30, 10, 70, 80];

[uniao, indicesA, indicesB] = **union(V1, V2);**

disp(uniao);

disp(indicesA);

disp(indicesB);

•Resultado:

10. 20. 30. 40. 50. 60. 70. 80.

V2

5. 2. 3. 4. 1.

V1

3. 4. 5.

← Elementos únicos de $V1 \cup V2$

← Índices dos elementos em V2

← Índices dos elementos em V1

Interseção

[resultado [, kA, kB]] = intersect(<Vetor A>, <Vetor B>)

- Retorna um vetor ordenado contendo os elementos em comum de dois vetores, adicionalmente retorna vetores com os índices dos elementos em cada vetor de entrada:

```
clc;  
V1 = [60, 30, 40, 50, 20];  
V2 = [20, 30, 10, 70, 80];  
[intersecao, indicesA, indicesB] = intersect(V1, V2);  
disp(intersecao);  
disp(indicesA);  
disp(indicesB);
```

•Resultado:

```
20. 30.      ← Elementos de  $V1 \cap V2$   
 5.  2.      ← Índices dos elementos em V1  
 1.  2.      ← Índices dos elementos em V2  
-->
```

Busca (pesquisa)

[indices] = find(<condição>[, <nmax>])

- Retorna um vetor ordenado contendo os elementos de um vetor que atendem à **condição** de entrada (o número de elementos é limitado a **nmax**, o valor -1 (padrão) indica “todos”):

```
clc;
```

```
V = [60, 30, 40, 50, 20, 20, 30, 10, 70, 80];
```

```
encontrados1 = find(V > 50);
```

```
encontrados2 = find(V == 30);
```

```
encontrados3 = find(V == 30 | V == 20);
```

```
disp(encontrados1);
```

```
disp(encontrados2);
```

```
disp(encontrados3);
```

•Resultado:

1. 9. 10.

← Elementos de V maiores de 50

2. 7.

← Elementos de V iguais a 30

2. 5. 6. 7.

← Elementos de V iguais a 30 OU iguais a 20

-->

Ordenação

[resultado, indices] = gsort(<Vetor>[, flag1, flag2])

- Retorna um vetor ordenado contendo os elementos de um vetor, adicionalmente retorna um vetor com os índices dos elementos no vetor de entrada;
 - Utiliza o algoritmo “quick sort”;
- **flag1**: usado para definir o tipo de ordenação, no caso de vetores, recomenda-se utilizar sempre o valor ‘g’ (valor padrão), que significa ordenar todos os elementos;
- **flag2**: usado para definir a direção de ordenação:
 - Valor ‘i’: para ordem crescente;
 - Valor ‘d’: para ordem decrescente (padrão);

Ordenação

[resultado, indices] = gsort(<Vetor>[, flag1, flag2])

•Exemplo:

```
clc;  
V = [60, 30, 40, 50, 20, 20, 30, 10, 70, 80];  
[ordenado1, indice1] = gsort(V);  
[ordenado2, indice2] = gsort(V, 'g', 'i');  
disp(ordenado1); // Ordenação padrão  
disp(indice1); // Índices dos elementos da ordenação padrão  
disp(ordenado2); // Ordenação de forma crescente  
disp(indice2); // Índices dos elementos da ordenação crescente
```

•Resultado:

80.	70.	60.	50.	40.	30.	30.	20.	20.	10.
10.	9.	1.	4.	3.	2.	7.	5.	6.	8.
10.	20.	20.	30.	30.	40.	50.	60.	70.	80.
8.	5.	6.	2.	7.	3.	4.	1.	9.	10.

-->

Introdução;
Declaração de vetores;
Algumas operações com vetores;
Algumas funções aplicadas a vetores;
Exercícios.

Exercícios

Exercícios propostos

1. Faça um programa que preencha um vetor de 10 elementos através de entradas do usuário. Após a definição dos elementos do vetor, calcule a média dos valores.

1. Faça um programa que preencha dois vetores de 10 elementos através de entradas do usuário. Após a definição dos dois vetores, construa um terceiro vetor onde cada elemento corresponde ao dobro da soma entre os elementos correspondentes dos outros dois vetores. Imprima o conteúdo do vetor calculado.

1. Faça um programa que preencha dois vetores de 10 elementos através de entradas do usuário. Após a definição dos dois vetores, construa um terceiro vetor onde cada elemento de *índice ímpar* receba o valor correspondente do *primeiro vetor* e cada elemento de *índice par* receba o valor correspondente do

Exercícios propostos

1. Escreva um programa que preencha um vetor com entradas do usuário. Considere que o usuário definirá apenas valores numéricos positivos, e que, ao desejar encerrar a definição dos elementos ele digite um valor negativo. Após a entrada de todos os elementos do vetor, calcule e imprima o seu somatório, **sem a utilização da função sum**.

1. Escreva um programa semelhante ao anterior, que retorne e imprima o produtório cumulativo, **sem a utilização da função cumprod**.

1. Escreva um programa semelhante aos anteriores, mas que retorne e imprima um vetor contendo apenas os elementos únicos, **sem a utilização da função unique**. *Dicas: Com o vetor preenchido, percorra seus elementos inserindo os elementos únicos em um novo vetor. Um elemento único é aquele que ainda*

Exercícios propostos

1. Definir mais exercícios para utilização de operações e funções.

Lista 4 do prof. David

- Resolução dos exercícios da lista conforme distribuição predefinida.

Próxima aula prática: resolução de exercícios com o Scilab.
Próxima aula teórica: Variáveis Homogêneas - Matrizes.

FIM!
Dúvidas?