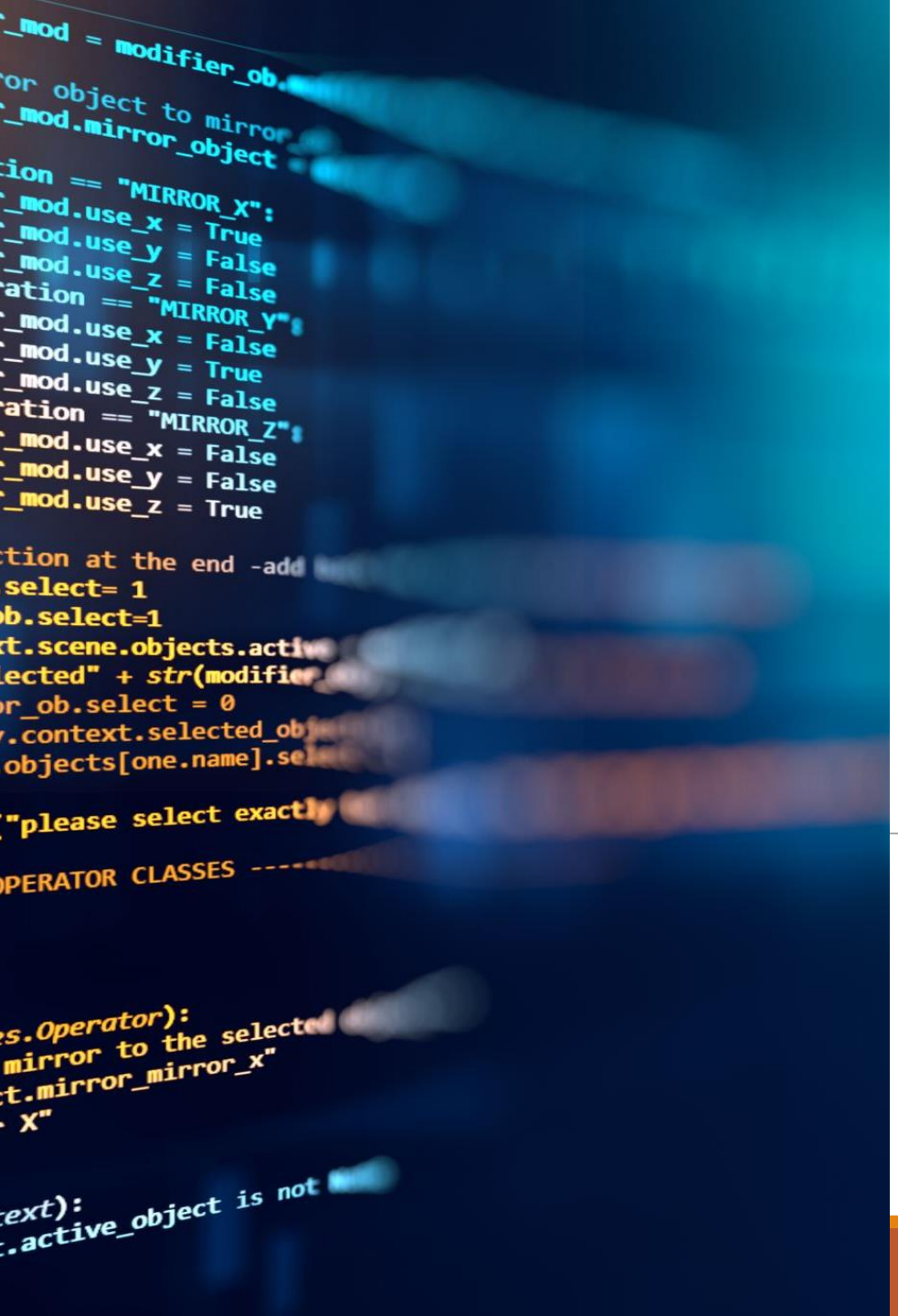


PROGRAMAÇÃO DE COMPUTADORES I

NONA PARTE

CAPÍTULO IV

TIPOS DE DADOS COMPOSTOS

PARTE I - VETORES

Programação de Computadores I

prof. Marco Villaça

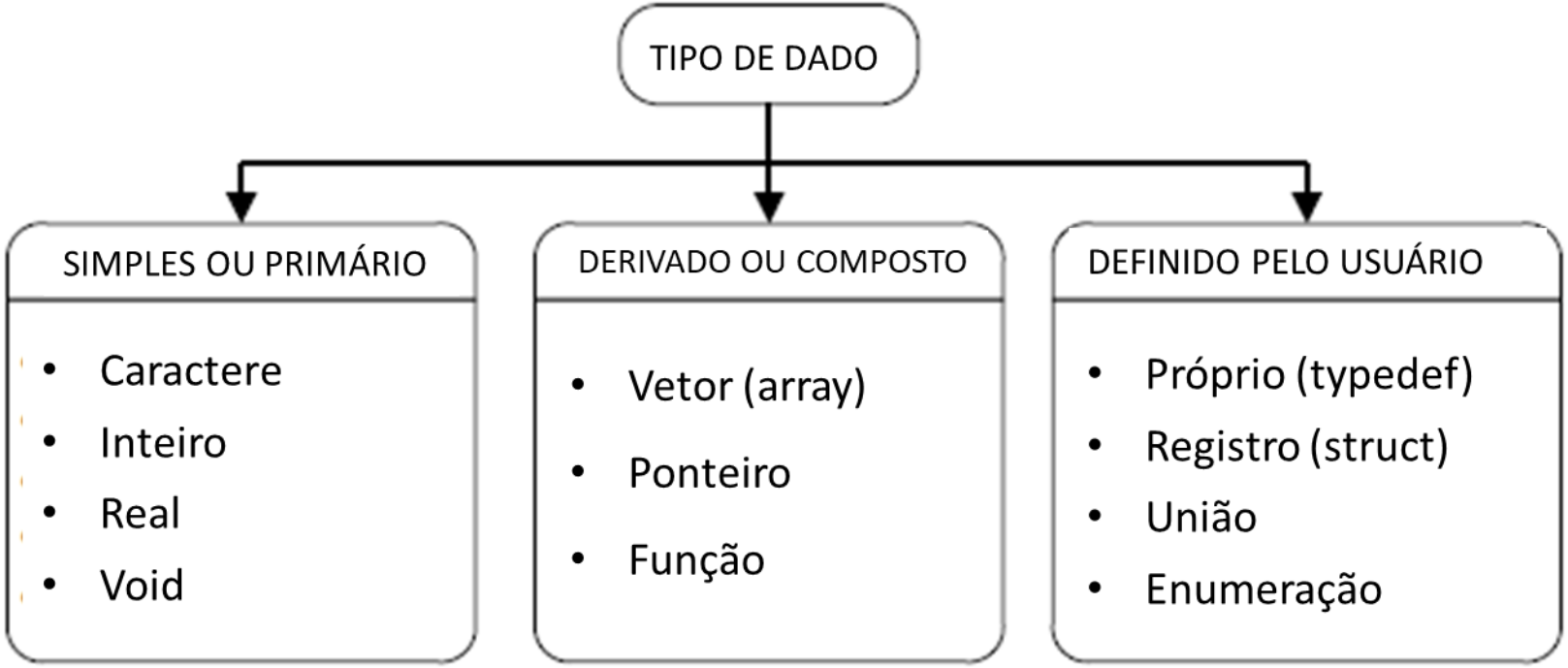
TIPOS DE DADOS COMPOSTOS

- Definidos pelo utilizador:
  - ✓ Um tipo composto pode ser construído em uma linguagem de programação a partir de tipos primitivos (simples) e de outros tipos compostos, em um processo chamado composição.
  - ✓ Composição
    - Conjunto de dados;
    - Exemplos em C:

      int naturais [3] = {1, 2 ,3} (homogêneo)

      struct aluno { char nome[20]; int ano_nasc; }; (heterogêneo)

LEMBRANDO
TAXONOMIA DOS DADOS



VETORES (ARRAYS)

Conjunto de variáveis do mesmo tipo dispostas em posições contíguas de memória e que podem ser acessadas por sua posição relativa, a partir da origem (primeiro elemento), através de um índice.

- Exemplo :

```
char vetor [10]
```

RESERVADOS 10 ESPAÇOS

Acesso	Conteúdo	Endereço
vetor[9]	-10	0x1.000.009
…	…	…
vetor [2]	- 5	0x1.000.002
vetor [1]	5	0x1.000.001
vetor[0]	10	0x1.000.000

No vetor da tabela o comando

```
printf("%d", vetor[2]);
```

apresentará o valor **–5** na tela

VETORES (ARRAYS)

- Exemplo em C:

```
int fibonacci[20] = {1, 1};
// fibonacci[0] = 1 e fibonacci[1] = 1
for (n = 2; n < 20; n++)
   fibonacci[n] = fibonacci[n-1] + fibonacci[n-2]
```

VETORES (ARRAYS)

- Para uma melhor compreensão da aplicação, imagine o seguinte problema:
  - ✓ Leia e armazene 30 valores de temperatura e calcule a média.
- Como fazer?
  - ✓ Atribuir uma variável para cada valor de temperatura?
    - temp1, temp2, temp3, …, temp30
- A melhor solução é criar um tipo de dado compostos, nesse caso um vetor ou variável composta homogênea unidimensional → temp[30]

TIPOS DE VETORES

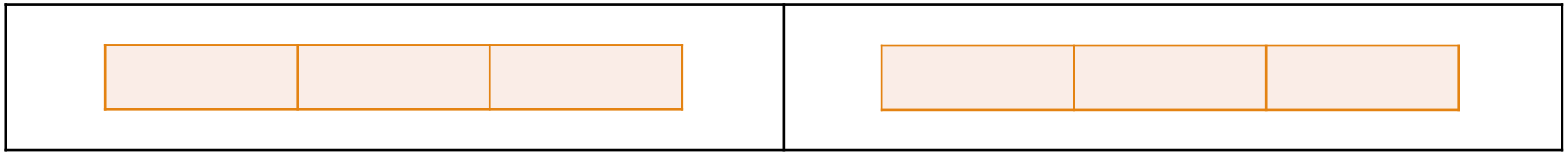
- Vetores estáticos:
  - ✓ Armazenamento do array é estático;
  - ✓ O tamanho do array é estático;
  - ✓ Índices e armazenamento vinculados em tempo de compilação
  - ✓ Vantagem: Eficiência na execução (sem reserva e liberação de memória);
  - ✓ Desvantagem: armazenamento permanece durante toda execução e é necessário saber o tamanho antes da execução do programa

TIPOS DE VETORES

- Vetores dinâmicos:
  - ✓ Armazenamento do array é dinâmico em estruturas específicas de memória, vinculado em tempo de execução;
  - ✓ O tamanho do array pode ser estático ou dinâmico;
  - ✓ Quando o armazenamento e a dimensão são dinâmicos o tamanho pode crescer ou diminuir conforme a necessidade

ÍNDICES DE VETORES

- Número de índices:
  - ✓ C: somente uma dimensão, porém elementos podem ser vetores.



  - ✓ No Scilab a unidade fundamental de dados é uma matriz. As variáveis simples são, na verdade, matrizes com uma única linha e uma única coluna.
    - Uma matriz pode ter qualquer dimensão.

VETORES EM C

- Vetor = array = matriz unidimensional
  - ✓ Declaração

    tipo nomevet [numelem] ;

    ```
    float frequencia[20];
    ```
  - ✓ Índice dos elementos começa em **zero**
  - ✓ Todos os elementos do mesmo tipo
  - ✓ Inicialização:

    ```
    int nat[3] = { 0, 1 , 2 };
    ```

STRINGS EM C

- Vimos que strings são vetores de caracteres (char array)
- Último elemento é \0
  - ✓ Funções da biblioteca padrão consideram essa premissa

EXEMPLO 1

- Calcule a média de um conjunto de até 30 temperaturas inseridas pelo usuário pelo teclado.
- As temperaturas devem ser armazenadas em um vetor.
- Mostre na tela os valores acima da média.

EXEMPLO 2

- Modificar o exemplo anterior para que o programa imprima a média entre as temperaturas mínima e máxima.

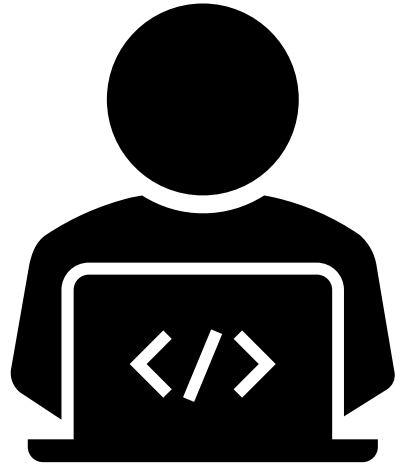
Exercício 1

Fazer um programa para receber uma frase de até 100 caracteres do teclado (não digitar caracteres acentuados) e guardar em um vetor de caracteres (string)

Após o programa deve pedir uma letra qualquer.

Recebida a letra o programa deve responder quantas vezes a letra aparece na frase.

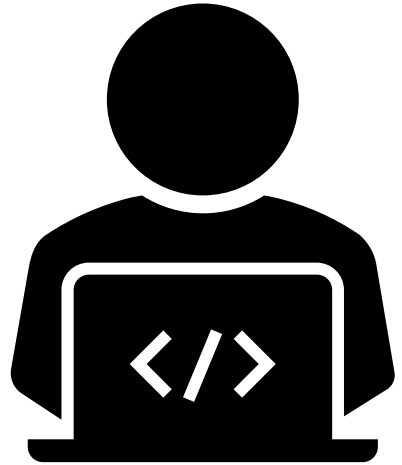
Por exemplo, na frase “To be or not to be, that's the question.” a letra “e” aparece 4 vezes.



Exercício 2

Modificar o Exemplo 1 para obter uma média móvel dos dez últimos valores inseridos. Considere, inicialmente, os dez primeiros valores iguais a zero.

A técnica de média móvel consiste em calcular a média aritmética das k observações mais recentes.



MATRIZES EM C

- Bidimensional
- Declaração

  tipo nomematriz[nlinhas][ncolunas];

  ✓ Exemplo: char mat[3][2];

$$\begin{bmatrix} a & b \\ c & d \\ e & f \end{bmatrix}$$

MATRIZES EM C
ARMAZENAMENTO

- Ordem de linha maior
  - ✓ Os elementos da matriz são armazenados por linha:

    A matriz

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$$

    seria armazenada na memória linear como

    1, 2, 3, 4, 5 , 6, 7, 8, 9

MATRIZES EM C
INICIALIZAÇÃO

- Na declaração, linha a linha:

```
char mat[3][2]={{'a','b'},{'c','d'},{'e','f'}};
```

MATRIZES EM C
INICIALIZAÇÃO

- Entrando com os elementos da matriz:

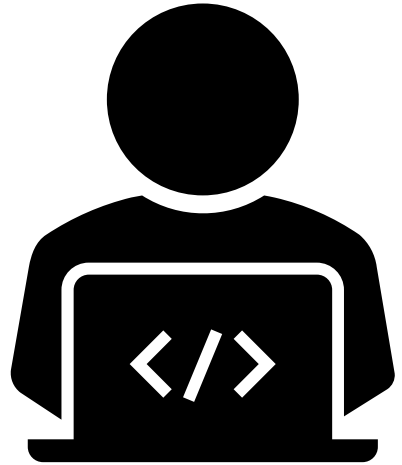
```
printf("Digite as dimensoes (mXn) de A: ");
scanf("%d,%d", &m, &n);
printf("Digite a matriz A:\n");
for (i = 0; i < m; i++)
     for (j = 0; j < n; j++)
     {
          printf("A[%d][%d] = ",i,j);
          scanf(%f", &A[i][j]);
     }
```

EXEMPLO 3

- Calcular a soma dos elementos de uma matriz introduzida pelo usuário por teclado.

Exercício 3

Modificar o Exemplo 3 para obter a soma dos elementos da diagonal principal de uma matriz quadrada



MATRIZES DE STRINGSS
INICIALIZAÇÃO

- Declaração

```
char nomevar[num_strings][comprim_strings];
```

- Para acessar uma das strings

```
nomevar[num_string]
```

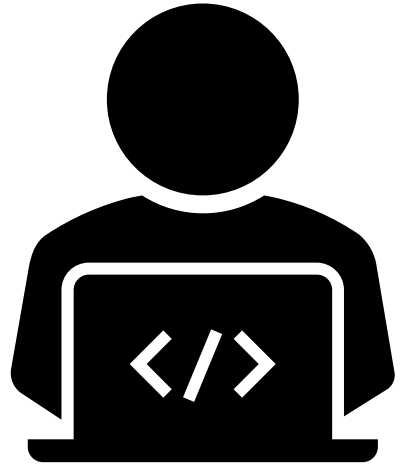
- Exemplo

char passivos [3] [10] = { “resistor”, “indutor”, “capacitor”} ;

EXEMPLO 4

- Fazer um programa para escrever os números de 0 a 9 por extenso

Fazer um programa para escrever os números de 0 a 99 por extenso.



Exercício 4

INICIALIZADORES DESIGNADOS
APENAS C99

```
…
int vet[5] =
{
        [0] = 1,
        [4] = -1
   };
…
```

- Elementos que não são explicitamente inicializados são implicitamente inicializados em zero. Esta sintaxe não era permitida antes de C99.

VETORES DE EXTENSÃO VARIÁVEL
APENAS C99

- **Não confunda com vetores dinâmicos.** O C99 suporta vetores de tamanho variável onde o tamanho é calculado em tempo de execução enquanto processa a definição do vetor.
- No entanto, uma vez criado, o vetor não pode mudar de tamanho.
- Exemplo

```
…
int tam
printf("Qual o tamanho de um vetor: ");
scanf("%d", &tam);
int vetor[tam];
…
```

VETORES NO SCILAB

- Criando um vetor linha:

```
-->v = [1 2 3]
 v  =
    1.    2.    3.
```

- Criando um vetor coluna:

```
-->u = [1; 2; 3]
 u  =
    1.
    2.
    3.
```

VETORES NO SCILAB

- Seja o vetor:

```
--> v = [1 2 3]
 v  =
    1.    2.    3.
```

- Obtendo o primeiro elemento do vetor v:

```
--> v(1)
 ans  =
    1.
```

EXEMPLO 5

- Programa para calcular a média de um conjunto de até 30 temperaturas inseridas pelo usuário pelo teclado

MATRIZES NO SCILAB

- Os elementos de uma matriz devem ser especificados entre colchetes;
- Elementos de uma mesma linha são separados por espaço ou vírgula;
- As linhas são separadas por ponto-e-vírgula.

```
-->A = [1 2 3; 4 5 6]
 A  =

    1.    2.    3.
    4.    5.    6.
```

MATRIZES NO SCILAB
INDEXAÇÃO

- Uma posição em uma matriz $n\ x\ m$ identificada por um par de índices (i, j), onde i é o índice da linha e j o índice da coluna.
- Obtendo o elemento A_{21} da matriz A:

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}$$

```
--->x = A(2,1)

 x  =

        4.
```

MATRIZES NO SCILAB

- Cada linha de uma matriz deve o mesmo número de elementos e cada coluna deve ter o mesmo número de linhas.

```
-->B = [1 2; 4 5 6]
                  !--error 6
Incoerência nas dimensões linha/coluna.
```

MATRIZES NO SCILAB

- No Scilab, matrizes inteiras ou seções de matrizes podem ser manipuladas:

```
A = B    // cópia de toda a matriz y para a matriz x
b = A(:,2) // cópia da 2a coluna da matriz para o
           // o vetor b
A(1:4,2:3) = B(2:5,8:9) // cópia de parte da matriz com
                        // mudanças de índice
```

- Essa característica confere ao Scilab diversas vantagens sobre o C ao operar com matrizes.

MATRIZES NO SCILAB

- Funções para obter a dimensão de uma matriz A:
  - ✓ Número de elementos de A:

```
n = lenght(A)
```

  - ✓ Número de linhas e de columas de A:

```
[l,c] = size(A)
```

MATRIZES NO SCILAB

- Os elementos de uma linha de uma matriz podem, também, podem ser especificados usando-se a notação `vi:passo:vf` onde
  - ✓ vi – primeiro elemento
  - ✓ vf – último elemento
  - ✓ passo – passo (quando omitido vale 1)

```
-->A=[0:2:8;1:2:9]
 A   =
    0.    2.    4.    6.    8.
    1.    3.    5.    7.    9.
```

MATRIZES
C x SCILAB

- Inicializando os elementos de uma matriz (4 x 4) com o valor 0 ou 1
  - ✓ C

```
for(i=0,i<4;i++)
   for(j=0,j<4,j++)
      A[i][j] = 0;
```

  - ✓ Scilab

```
A = zeros(4,4) // matriz 4 x 4 de 0s
A = ones(4,4)  // matriz 4 x 4 de 1s
```

MATRIZES
C x SCILAB

- Criando a matriz (3 x 3) identidade
  - ✓ C

```
for(i=0,i<3;i++)
  for(j=0,j<3,j++){
       if(i=j)A[i][j] = 1;
       else   A[i][j] = 0; }
```

  - ✓ Scilab

```
A = eye(3,3)
```

MATRIZES
C x SCILAB

- Adição de Matrizes – C = A + B
  - ✓ C

```
for(i=0,i<n;i++)
   for(j=0,j<4,j++)
      C[i][j]=A[i][j]+B[i][j];
```

  - ✓ Scilab

```
C = A + B
```

MATRIZES
C x SCILAB

- Produto de Matrizes – C = A * B
  - ✓ C

```
for(i=0; i<n; i++)
  for(j=0; j<n; j++) {
        C[i][j] = 0;
        for(k=0; k<n; k++)
            C[i][j] += A[i][k]*B[k][j];
  }
```

  - Scilab

```
C = A * B
```

MATRIZES
C x SCILAB

- Multiplicação de uma matriz por um escalar, C = a * B
  - ✓ C

```
for(i=0; i<n; i++)
  for(j=0; j<n; j++)
        C[i][j] = a*B[i][j]
        C = A * B
```

  - ✓ Scilab

```
C = a * B
```

MATRIZES
SCILAB

- Colocando um “ .” na frente do operador, os operadores aritméticos *, /, e ^ podem, ainda, serem usados para realizar a operações entre duas matrizes (ou vetores) elemento por elemento:
  - ✓ Exemplo

```
-->A = [1 2; 3 4];
-->B = [2 3; 4 5];
-->A .* B
ans  =
   2.     6.
  12.    20
```

MATRIZES
ALGUMAS FUNÇÕES ÚTEIS DO SCILAB

função	descrição
sum(A)	soma dos elementos de A
prod(A)	produto dos elementos de A
mean(A)	média dos elementos de A
mx = max(A) [mx, i] = max(A)	mx é o valor máximo de A e i é a posição desse valor
mi = min(A) [mx, i] = min(A)	mx é o valor mínimo de A ei é a posição desse valor
gsort(A)	ordena as linhas da matriz em ordem decrescente

MATRIZES
ALGUMAS FUNÇÕES ÚTEIS DO SCILAB

função	descrição
harmean(A)	média harmônica dos elementos de A
geomean(A)	média geométrica dos elementos de A
tabul(A)	frequência dos valores de A
trace(A)	soma dos elementos da diagonal principal de A
det(A)	determinante de A
size(A)	dimensões de A

MATRIZES
SCILAB

- Como saber mais sobre matrizes no Scilab:
  - ✓ Matrizes elementares:

https://help.scilab.org/docs/6.1.0/pt_BR/section_f0a673dd193b7455aad6db094bf96c78.html

  - Manipulação de matrizes:

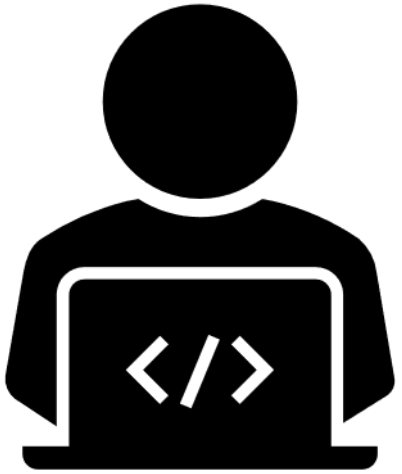
https://help.scilab.org/docs/6.1.0/pt_BR/section_b6ebe288c4978496cece52f2b916c97a.html

  - Operações com matrizes:

https://help.scilab.org/docs/6.1.0/pt_BR/section_02d25d3f6893ff8fc54c7f9fe59ed5a2.html

Exercícios

1. Dado um vetor {a} de n elementos, faça um programa C/Scilab que ordene e imprima o vetor em ordem decrescente.
2. Dada um Matriz [A] (m x n), faça um programa C/Scilab que obtenha a transposta de [A].



Exercícios

3. Dado dois vetores quaisquer de números positivos, {A} e {B}, faça um programa em C/Scilab que imprima, sem repetição, na tela todos os elementos comuns aos dois vetores.

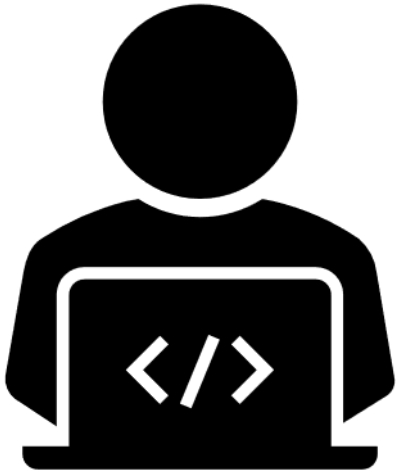
   A marca de fim de vetor é -1.

   Vetores para teste:

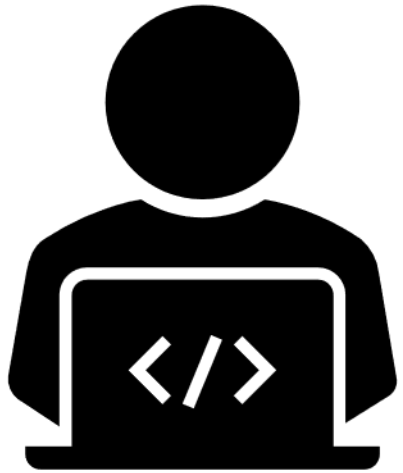
   A[30]= {1, 2, 5, 7, 11, 12, 13, 20, 21, 12,-1} ;

   B[30]= {1, 3, 5, 8, 12, 15, 16, 21, 23, 5, 20,-1} ;

   RESPOSTA: {1, 5, 12, 20, 21}



4. Faça um programa em C/Scilab que faça a troca de linhas de uma matriz [A] de m x n. Inicialmente, atribuir aos elementos da matriz o valor da linha + coluna. Imprimir a matriz e solicitar ao usuário quais linhas quer trocar.



Exercícios

Bibliografia e crédito das figuras

OUALLINE, S. Practical C Programming. 3a ed. O'Reilly, 1997.

SEBESTA, R. Conceitos de Linguagens de Programação. 5a ed. Porto Alegre: Bookman, 2003.

http://help.scilab.org/docs/6.1.0/pt_BR/index.html