Arrays (Matriz / Vetor)

É uma estrutura de dados que contém uma série de dados ordenados, chamados "elementos". Os elementos são referenciados por número ordinal, primeiro elemento é 1, segundo 2... Os elementos podem ser de qualquer tipo, caracter, numérico, data etc.

Você lembra do conceito de Matriz quando nas épocas de colégio, em Matemática??? Pois é, a mesmíssima coisa!

Um *Array* ou matriz é uma variável em clipper que armazena vários valores, onde tais valores são armazenados e consultados através de sua posição na matriz.

Vejamos um exemplo de uma matriz numérica de 4x3 (4 linhas e 3 colunas):

$$B_{4,3} = \begin{pmatrix} 10 & 20 & 40 \\ 15 & 35 & 42 \\ 53 & 25 & 37 \\ 12 & 91 & 33 \end{pmatrix}$$

Que poderia ser representada assim:

$$B_{4,3} = \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \\ a_{41} & a_{42} & a_{43} \end{pmatrix}$$

Onde cada a_{nm} é um valor da matriz. Por exemplo, se você quisesse saber o valor de a_{32} na **matriz B** você encontraria o número 25. Analogamente os outros valores.

Vejamos outro exemplo, agora de uma matriz de uma só coluna, de 3x1 (3 linhas e 1 coluna), também conhecida de *Vetor* (só quando uma coluna):

$$C_{3,1} = \begin{pmatrix} 3 \\ 7 \\ 4 \end{pmatrix}$$

Que poderia ser representada assim:

$$C_{3,1} = \begin{pmatrix} a_{11} \\ a_{21} \\ a_{31} \end{pmatrix}$$

Onde cada a_{nm} é um valor da matriz. Por exemplo, se você quisesse saber o valor de a_{21} na **matriz C** você encontraria o número 7. Analogamente os outros valores.

Agora que você entendeu bem a idéia, vamos traduzir isso para linguagem Clipper:

A **matriz B** poderia ser contruída em Clipper da seguinte maneira, compare com a imagem:

Em Clipper:	Na Matemática:

// definindo a matriz:
B := ARRAY(4,3)
// atribuindo valores
B[1][1] := 10
B[1][2] := 20
B[1][3] := 40
B[2][1] := 15
B[2][2] := 35
B[2][3] := 42
B[3][1] := 53
B[3][2] := 25
B[3][3] := 37
B[4][1] := 12
B[4][2] := 91
B[4][3] := 33
	$B_{4,3} = \begin{pmatrix} 10 & 20 & 40 \\ 15 & 35 & 42 \\ 53 & 25 & 37 \\ 12 & 91 & 33 \end{pmatrix}$

Outra forma em Clipper, atribuição dinâmica de uma matriz:

Em Clipper:	Na Matemática:

// definindo a matriz:
B := {}
// atribuindo valores
AADD(B, {10,20,40})
AADD(B, {15,35,42})
AADD(B, {53,25,37})
AADD(B, {12,91,33})
	$B_{4,3} = \begin{pmatrix} 10 & 20 & 40 \\ 15 & 35 & 42 \\ 53 & 25 & 37 \\ 12 & 91 & 33 \end{pmatrix}$

Para consultar o valor de a_{32} da **matriz B** em clipper você poderia usar:

? B[3][2]

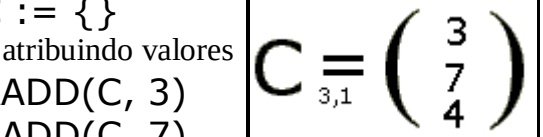
A **matriz C** poderia ser contruída em Clipper da seguinte maneira, compare com a imagem:

Em Clipper:	Na Matemática:

// definindo a matriz:
C := ARRAY(3)
// atribuindo valores
C[1][1] := 3
C[1][2] := 7
C[1][3] := 4
	$C_{3,1} = \begin{pmatrix} 3 \\ 7 \\ 4 \end{pmatrix}$

Outra forma em Clipper, atribuição dinâmica de uma matriz:

Em Clipper:	Na Matemática:

// definindo a matriz:
C := {}
// atribuindo valores
AADD(C, 3)
AADD(C, 7)
AADD(C, 4)
	

Para consultar o valor de a_{21} da **matriz C** em clipper você poderia usar:

? C[2]

Atenção: Se você colocar uma posição além do tamanho do array, o sistema retornará um erro deBASE/1132 Bound error: array access porque o elemento referenciado não existe. Por exemplo, se fizéssemos ? C[4] na matriz C citada acima, suscitaria este erro, pois C[4] não existe!

Ok, agora vamos para outro passo.

Para se buscar um valor dentro do *Array,* usa-se a função *ASCAN()*, vamos recordar sua sintaxe:

ASCAN(<aARRAY>, <expPROCURA>,[<nINICIO>], [<nCONTAGEM>])

Onde:

aARRAY = Array onde se fará a pesquisa.

expPROCURA = Expressão de procura, pode ser um valor de qualquer tipo (caracter, numerico, data ou lógico) ou um**bloco de código** (*code block*).

nINICIO = Opcional. Valor númerico que representa o elemento inicial de pesquisa, qual a posição do Array que deve iniciar a pesquisa. Se omitido, o valor padrão será 1 (inicio do array)

nCONTAGEM = Opcional. O número de elementos que irá pesquisar a partir de **nINICIO.** Se omitido, o valor padrão será o tamanho do array (LEN(**aARRAY**)).

ASCAN() retorna um valor numérico representando a posição do elemento procurado caso este seja encontrado, caso contrário, retornará 0 (zero).

Agora veja alguns dos diversos modos de buscar um valor dentro de um *Array*, usando a função *ASCAN()* do Clipper:

Exemplo 1: Busca simples.

```
aArray := { "Tom", "Mary", "Sue" }

? ASCAN(aArray, "Mary") // Resultado: 2
? ASCAN(aArray, "mary") // Resultado: 0
? ASCAN(aArray, { |x| UPPER(x) == "MARY" }) // Resultado: 2
```

Exemplo 2: Verificar quantas ocorrências de um valor dentro do *Array.*

```
aArray := { "Tom", "Mary", "Sue","Mary" }

nStart := 1 // Armazena última posição do elemento no Array
nAtEnd := LEN(aArray) // tamanho da matriz
DO WHILE (nPos := ASCAN(aArray, "Mary", nStart)) > 0
    ? nPos, aArray[nPos] // mostra posicao e valor.
    // Atribui nova posicao inicial e testa
    // com tamanho da matriz (evitar erro).
    IF (nStart := ++nPos) > nAtEnd
       EXIT
    ENDIF
ENDDO
```

Obs.: A linha:

DO WHILE (nPos := ASCAN(aArray, "Mary", nStart)) > 0

é o mesmo que:

```
nPOS := ASCAN(aArray, "Mary", nStart)
DO WHILE nPOS > 0
    nPOS := ASCAN(aArray, "Mary", nStart)
```

Exemplo 3: Este exemplo procura uma matriz bi-dimensional usando um bloco de código (code block). A função retornará a posição da linha do elemento procurado. Note que o parâmetro **aVal** no bloco de código é uma matriz e que **aVal[2]** significa **coluna 2** da matriz (alí você colocará o número da coluna que você irá procurar):

```
aArr:={}
CLS
AADD(aArr,{"um","dois"})
AADD(aArr,{"três","quatro"})
AADD(aArr,{"cinco","seis"})
```

? ASCAN(aArr, {|aVal| aVal[2] == "quatro"}) // Retornará 2.

- Se você quiser saber como indexar ***Arrays***, veja a página de *Dicas*!
- Para salvar e restaurar ***Arrays*** para o disco veja as funções na página de *Downloads*.