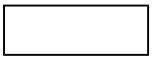
8) "ARRAYS" :

8.1) Arrays uni-dimensionais (ou VETORES):

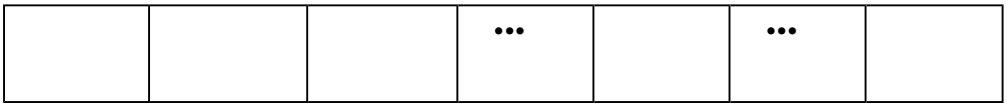
Array é um Tipo de Dados Estruturado. Trata-se de um conjunto com um número fixo de elementos, todos do mesmo tipo. Todo o conjunto é identificado por um único nome, e cada *elemento* (ou *componente*) pode ser referenciado (e acessado) diretamente. Cada componente é *indexado* por um valor de tipo simples (não podendo portanto ser do tipo *real*).

Exemplo:

var X: *real*;
A: *array*[1..10] *of real*;



X



A

A[1] A[2] A[3] A[I] A[10]

Motivação: Calcular a média aritmética de 100 notas e contar a quantidade de notas acima dessa média calculada (ié, da "média da turma").
Pelo que foi visto até aqui, seria preciso digitar todas as 100 notas novamente, ficando sujeito a possíveis erros de digitação... Daí que se faz necessário encontrar de um meio de "armazenar" as notas quando forem digitadas, e de poder acessá-las posteriormente.

Como declarar:

var <lista de identificadores> : *array* [T_1] *of* T_2;

ou

type <identificador-T> = *array* [T_1] *of* T_2;
var <lista de identificadores> : <identificador-T>;

onde: - T_1 é o tipo do índice, e é da forma LI..LS,
sendo LI o "limite inferior" e
LS o "limite superior"

- T_2 é o tipo dos componentes. Pode ser qualquer tipo, inclusive outro "array".

Como fazer referência a (e acessar) componente:

<identificador>[<índice>]

onde: - <identificador> é o mesmo "nome" da declaração
- <índice> é uma expressão que resulta num valor do tipo T_1 (ié, um valor no intervalo LI..LS).

Observações:

1) Outros tipos de dados estruturados que serão estudados posteriormente são: **records**, **files** e **strings**.

2) O número de componentes de um **array** não pode variar durante a execução do programa. Deve-se portanto fazer uma estimativa do seu tamanho máximo, quando da sua declaração.

3) Um outro tipo de dados é o **apontador** (ou **pointer**), que permite definir um conjunto com um número variável de elementos (através de "alocação dinâmica de memória").

4) Um tipo de dados **record** é um conjunto de elementos possivelmente de tipos diferentes.

5) É permitida a atribuição V1 := V2, onde V1 e V2 são vetores.

6) Seja V um vetor. Então, não é permitido o comando **read**(V). Neste caso, deve-se ler um elemento de cada vez. Isto é, deve-se fazer:

```
for I:=1 to N do read(V[I]).
```

Exemplos:

(1)
```
type VETOR = array[1..100] of real;
var NOTA: VETOR;
```

(2)
```
var NOTA:  array[1..100] of real;
```

(3)
```
read(NOTA[I]);
...NOTA[1]:=10;
...NOTA[5]:=MEDIA+1;
...NOTA[I]:=NOTA[I]+1;
```

(4)
```
{leitura de um vetor}
 for I:=1 to 100 do
    begin
      write('digite a ', I, 'ª nota: ');
      readln(NOTA[I])
    end;
```

```
(5) {impressão de um vetor}
      for I:=1 to 100 do write(NOTA[I],‘ ’);

(6) {cálculo da média}
      SOMA:=0;
      for I:=1 to 100 do
        SOMA:=SOMA+NOTA[I];
      MEDIA:=SOMA/100;

(7) {contagem das notas acima da média}
     for I:=1 to 100 do
        if NOTA[I] > MEDIA
           then CONTA:=CONTA+1;

(8) var TEMP: array[-50..50] of real;

(9) var OCORRE: array[‘a’..‘z’] of integer;

(10)type COR = (azul,verde,amarelo,...,vermelho);
      var X: array[azul..vermelho] of integer;

(11)typeMESES =
         (jan,fev,mar,abr,mai,jun,jul,ago,set,out,nov,dez);
      var GASTOS: array[MESES] of real;
```

8.2) Arrays bi-dimensionais (ou MATRIZES):

Num **array bi-dimensional**, cada componente é *indexado* por dois valores, isto é, utiliza dois índices. Daí olharmos pra um array deste tipo como se fosse uma **matriz**.

Como declarar:

var <lista de identificadores>:
array[LI .. LS] *of array*[LI .. LS] *of* T;

ou, melhor:

var <lista de identificadores>:
array[LI .. LS, LI .. LS] *of* T;

De outra maneira:

type <identificador-T> =
array[LI1.. LS1, LI2.. LS2] *of* T;
var <lista de identificadores>: <identificador-T>;

Como fazer referência a (e acessar) componente:

<identificador>[<índice-1>,<índice-2>]

como também:

<identificador>[<índice-1>] [<índice-2>]

onde: - <identificador> é o mesmo "nome" da declaração

- <índice-1> e <índice-2> são expressões que resultam em valores nos intervalos LI1.. LS1 e LI2.. LS2, respectivamente.

Exemplos:

Seja a declaração:

```
var M1, M2: array[1..10,1..10] of real;
```

(1)
```
{leitura de uma matriz}
for I:=1 to NLIN do
  for J:=1 to NCOL do
    begin
      write('M1[',I, ',',J, ']=');
      readln(M1[I,J])
    end;
```

(2)
```
{impressão de uma matriz}
for I:=1 to NLIN do
  begin
    for J:=1 to NCOL do write(M1[I,J]:0:2,' ');
    writeln
  end;
```

(3)
```
{produto de uma matriz por um escalar K}
for I:=1 to NLIN do
  for J:=1 to NCOL do
    M2[I,J]:=K*M1[I,J];
```

8.3) ARRAYS n-dimensionais:

Já que os componentes de um array podem também ser do tipo array, podemos definir um **array** com um número qualquer de *dimensões*! Para cada dimensão devemos utilizar um *índice* diferente.

Um **array n-dimensional** seria declarado com o tipo:

array[T□] *of array*[T□] *of array*[T□] … *of* T□

Exemplo: (n=3)

```
var LIVRO: array[1..500,1..40,1..90] of char;
```

Observação: Neste exemplo estamos considerando um *livro* com 500 páginas, cada página com 40 linhas e cada linha com 90 caracteres (ou

colunas).