7 - ARRAYS

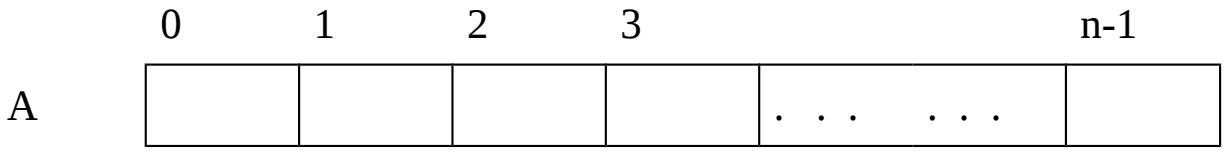
É um tipo de dado estruturado.

7.1 - Características Básicas

- Coleção de elementos do mesmo tipo, referenciados com o mesmo nome → arranjos
- Cada elemento é identificado por um índice.
- Pode ter de uma a varias dimensões.
- Arrays unidimensionais são conhecidos pelo nome de vetor.
- Em C, um vetor de tamanho n possui índices de 0 a n-1.

- Exemplo:

Seja o vetor A, com n elementos:



- Declaração:

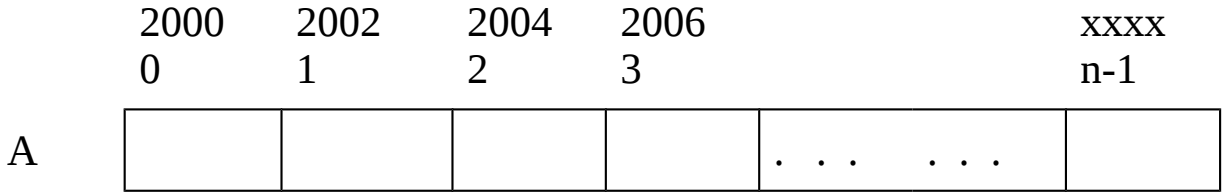
  < tipo > < nome > [<tamanho>]

- Exemplos:

```
char mensagem[10];      /* vetor de caracteres  de tamanho 10 */
int  A[20];             /* vetor de inteiros de tamanho 20 */
float  notas[40];       /* vetor  de reais  de tamanho 40 */
```

- Os elementos de um vetor estão em posições contíguas na memória.
- Exemplo:

  Seja o vetor A, com n elementos inteiros (ocupando 2 bytes)



- Os vetores podem ser inicializados em sua declaração.
- Ex.1:

int vet[5] = {3, 2, 1, 6, 4}

ou

int vet[ ] = {3, 2, 1, 6, 4}

obs.: no segundo caso, o tamanho de vet é determinado pelo número de elementos.

0	1	2	3	4
3	2	1	6	4

- Ex. 2: char v[5] = {‘a’ , ‘b’ , ‘c’}

0	1	2	3	4
‘a’	‘b’	‘c’		

Observações:

A inicialização pode ocorrer somente no momento da declaração.

```
int vet [5];
vet  = {3, 2, 1, 6, 4};      /* erro de compilação */
```

Os valores utilizados não podem ser resultados de expressões.

```
int vet [5], x= 3;
vet  = {3, 2, 1, 6, 4+x };      /* erro de compilação */
```

7.2 - Leitura e Escrita

- Elementos de um array podem ser tratados como variáveis individuais.

- A leitura e escrita é feita de maneira análoga a de variáveis individuais.
- Exemplo:

  Seja o seguinte programa:

```
#include <stdio.h>

main( )
{
  int i;
  float  soma = 0.0,     notas [5];
  for ( i = 0 ;  i < 5  ;  i ++ )
        {
        printf (“Digite a nota do aluno  %d “, i + 1);
        scanf (“% f “, &notas[i]);
        soma + = notas[i];
        }
  printf ( “Média da turma =  % 6.2f “, soma/5);
}
```

- O nome do array é um apontador para o endereço do primeiro elemento do array (ex.: notas).

  Cada elemento do array é uma variável (ex.: notas[i]).

- C não faz verificação de limites. Se for ultrapassado o limite do array, valores podem ser armazenados em qualquer lugar da memória, podendo até “apagar” outras informações.

- Exemplo:

  Seja o seguinte programa:

  ```
  int  vet[5];
    .
    .
  scanf ( “% d   % d”, &vet[5], &vet[6];
  ```

  - Essa leitura está inconsistente, pois não existe vet[5] e vet[6].

- O programador deve tomar cuidado com os limites do array pois muitas vezes o compilador não emite mensagem de erro

7.3 – Arrays Multidimensionais

- Podemos construir arrays com duas ou mais dimensões.

- Declaração:

  < tipo > <nome> [<tamanho 1>] [< tamanho 2>] ... [ ]

  Exemplo: Seja a seguinte declaração:

  ```
  char  M[3] [4]
  ```

  A representação gráfica da matriz acima é a seguinte:

	0	1	2	3
0				
1				
2				

- É possível declarar e inicializar no mesmo comando

  Exemplo:

```
int matriz [2] [3] = {{11, 15, 18}  ,      /* matriz [0]  */
                      { 21, 24, 27} };   /*  matriz [1] */
```

o trecho acima é equivalente a:

```
int matriz [2] [3] = {11, 15, 18, 21, 24, 27};
```

O efeito produzido pelo comando acima é o seguinte:

	0	1	2
0	11	15	18
1	21	24	27

Os valores são armazenados em endereços consecutivos de memória, na seguinte ordem:

matriz[0][0], matriz[0][1], matriz[0][2], matriz[1][0], matriz[1][1], matriz[1][2].

- Para um array bidimensional com x linhas e y colunas, contendo elementos de um determinado tipo base, a seguinte expressão fornece o número de bytes necessários para armazená-lo:

nº bytes = x * y * sizeof(tipo base)

- Para acessar os elementos é necessário especificar o índice de linha e o de coluna:

Exemplo: Para a matriz declarada como char M[3] [4], tem-se:

	0	1	2	3
0	M[0][0]	M[0][1]	M[0][2]	M[0][3]
1	M[1][0]	M[1][1]	M[1][2]	M[1][3]
2	M[2][0]	M[2][1]	M[2][2]	M[2][3]

O seguinte trecho de programa ilustra a leitura de uma matriz bidimensional contendo elementos inteiros:

```
int i, i,  matex[3][4];
main( )
{
   for ( i = 0 ;  i < 3  ;  i ++ )
      for (j = 0; j < 4; j++ )
      {
        printf (“Digite elemento da linha  %d e coluna %d \n”,i,j);
        scanf (“% d “, &matex[i][j]);

      }
}
```

- Um array de dimensão k, onde o número de elementos em cada dimensão é n_0, n_1, … n_{k-1}, respectivamente, pode ser imaginado como um array de dimensão n_0, cujos elementos são arrays de dimensão k-1.

Exemplo:

```
int M [3][4] = { {11, 15, 18, 19},
                 {21, 24, 27, 28},
                 {31, 33, 35, 37} };
```

pode ser imaginado como um array unidimensional de 3 elementos do tipo int[], ou seja, arrays de int; cada um dos 3 elementos é um array de 4 elementos do tipo int:

```
M[0]  ---> {11, 15, 18, 19}
M[1]  ---> {21, 24, 27, 28}
M[2]  ---> {31, 33, 35, 37}
```