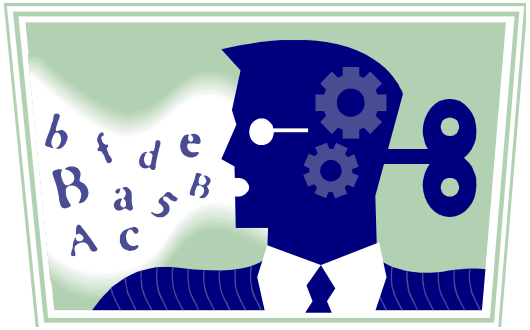
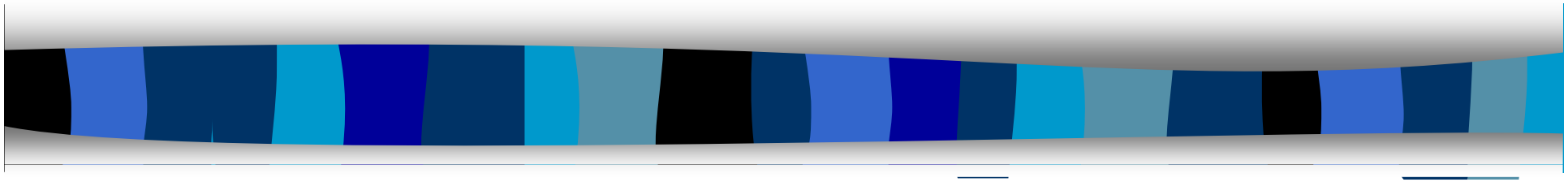


Revisão de APC II



Vetores e Matrizes

Tipos de Dados

- Sistemas de computação têm por finalidade a manipulação de dados
- Um algoritmo, basicamente, consiste na entrada de dados, seu processamento e a saída dos dados computados

- Alguns tipos de dados são considerados básicos, ou primitivos:
  - **Dados Numéricos**: representam valores numéricos diversos
    - Ex: 4, 5.5, 6.1E10, …
  - **Dados Alfanuméricos**: representam valores alfabéticos, numéricos, sinais, símbolos e caracteres
    - Ex: “salário”, “e23”, “%^#”, …
  - **Dados Lógicos**: associados a valores lógicos (*True* ou *False*)
    - Ex: *true*, *false*

Tipos de Dados Compostos

- Dois outros tipos de dados, um pouco mais sofisticado, são extremamente úteis: matrizes e vetores
- Tratam-se de variáveis compostas homogêneas
  - O conteúdo é sempre do mesmo tipo
- Correspondem a posições de memória contíguas
  - Identificadas por um mesmo nome
  - Individualizadas por índices
- Se NOTA contém os seguintes valores:

NOTA	60	70	90	55	91	47	74	86
	1	2	3	4	5	6	7	8

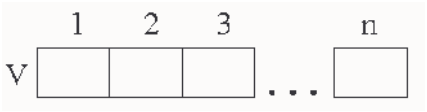
então, NOTA[3] faz referência ao terceiro elemento do conjunto (90)

Vetores

- Também chamado arranjo (array), tipo composto ou agregado
- É uma estrutura de dados homogênea e de acesso aleatório
  - Homogênea: contém elementos de um mesmo tipo;
  - Acesso aleatório: todos seus elementos são igualmente acessíveis a qualquer momento;
- Um elemento específico em uma matriz é acessado através de um índice
- Está disponível na maioria das linguagens
- É usado como base para estruturas de dados mais complexas
- Considere que o próprio acesso a memória do sistema é linear como um vetor

Vetores

- Elementos de um vetor são referenciados com um mesmo *nome*, mas diferenciados por um único *índice*



V = V(1), V(2), …, V(n), onde V(i), com 1 ≤ i ≤ n, é o i-ésimo elemento de V

- Sintaxe: `tipo nome[tamanho];`
- Exemplo:

```
void main(){
   int x[10];  /*reserva 10 elementos inteiros*/
   int i;
   for (i=0; i <10; i++){
     x[i] = i;
   }
}
```

Matrizes

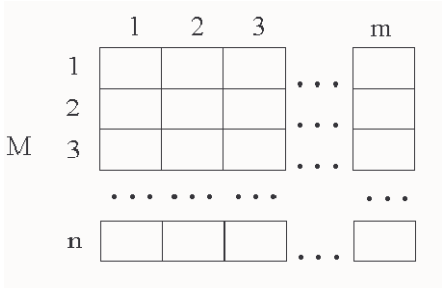
- Arranjo (array) multidimensional. Normalmente, bidimensional;
- Assim como vetores, trata-se de uma estrutura de dados homogênea e de acesso aleatório;
- Necessita de dois ou mais índices
- Também figura na maioria das linguagens;

Matrizes

- Arranjo de várias dimensões de dados do tipo básico, com um mesmo *nome*, mas diferenciado por um *índice* para cada dimensão.

```
     M(1,1)   M(1,2)  …  M(1,m)
M =  M(2,1)   M(2,2)  …  M(2,m)
     …        …       …  …
     M(n,1)   M(n,2)  …  M(n,m)
```

onde M(i,j) representa o elemento da i-ésima linha e j-ésima coluna



- Sintaxe:

```
tipo NomeMatriz[MAX_LINHAS][MAX_COLUNAS];
```

- Exemplo:

```
m[0][1] = 10;
m[1][0] = 20;
printf(Segundo valor de M: %d, m[1][0]);
```

Matrizes

- Geralmente, o acesso aos elementos de uma matriz é feito por um comando de looping (muitos elementos)

```
for (i=0;i<10;i++) {       // percorre a linha
    for (j=0;j<10;j++) {   // percorre a coluna
            a[i][j] = 0;
    }
}
```

- O exemplo acima colocará o valor 0 (zero) para toda a matriz, percorrendo todas as colunas de todas as linhas.

Exercícios:

1. Escrever programa que declare um vetor de 100 elementos numéricos, preenchido aleatoriamente com valores entre 1 e 1000. Depois, deve receber um valor do usuário e verificar se existem elementos iguais a esse valor no vetor. Se existir, escrever as posições em que estão armazenados.

2. Escrever programa que declare duas matrizes A e B, inteiras e bidimensionais 5x5, preenchidas aleatoriamente com valores entre 0 e 10. Depois:
   a. Imprimir as matrizes
   b. Executar a multiplicacao de A por B
   c. Imprimir a matriz resultante