Algoritmos e Programação

Aula 24

Estruturas Complexas (Matrizes)

Vetores (review)

Vetores em C são uma **coleção** de **dados** uniformes com **tamanho fixo**, na qual usamos um ***índice numérico*** para acessar elementos.

Vetores (review)

Vetores em C são uma **coleção** de **dados** uniformes com **tamanho fixo**, na qual usamos um ***índice numérico*** para acessar elementos.

Podemos utilizar *vetores* em C para representar, matematicamente, matrizes linha ou coluna:

Vetores (review)

Vetores em C são uma **coleção** de **dados** uniformes com **tamanho fixo**, na qual usamos um ***índice numérico*** para acessar elementos.

Podemos utilizar *vetores* em C para representar, matematicamente, matrizes linha ou coluna:

Código:

```
int v[3] = {2,5,0};
```

Equivalente: $\begin{pmatrix} 2 \\ 5 \\ 0 \end{pmatrix}$ ou $\begin{pmatrix} 2 & 5 & 0 \end{pmatrix}$

Matrizes

Em C, também podemos representar *matrizes* matemáticas, ou seja, uma organização de números em linhas e colunas:

$$\begin{pmatrix} 1 & 7 & 2 \\ 0 & 5 & 4 \\ 0 & 6 & 9 \end{pmatrix}$$

Como fazer isto em C?

Matrizes

Em C, também podemos representar *matrizes* matemáticas, ou seja, uma organização de números em linhas e colunas:

$$\begin{pmatrix} (1 & 7 & 2) \\ (0 & 5 & 4) \\ (0 & 6 & 9) \end{pmatrix}$$

Como fazer isto em C?

Solução: podemos construir ***vetores* de *vetores***, o quais chamamos (adequadamente) de matrizes.

Alocação de Matrizes

A alocação de matrizes é bastante similar à de vetores com a diferença de que temos ***duas indicações*** de ***multiplicidade***:

Alocação de Matrizes

A alocação de matrizes é bastante similar à de vetores com a diferença de que temos ***duas indicações*** de ***multiplicidade***:

Exemplo:

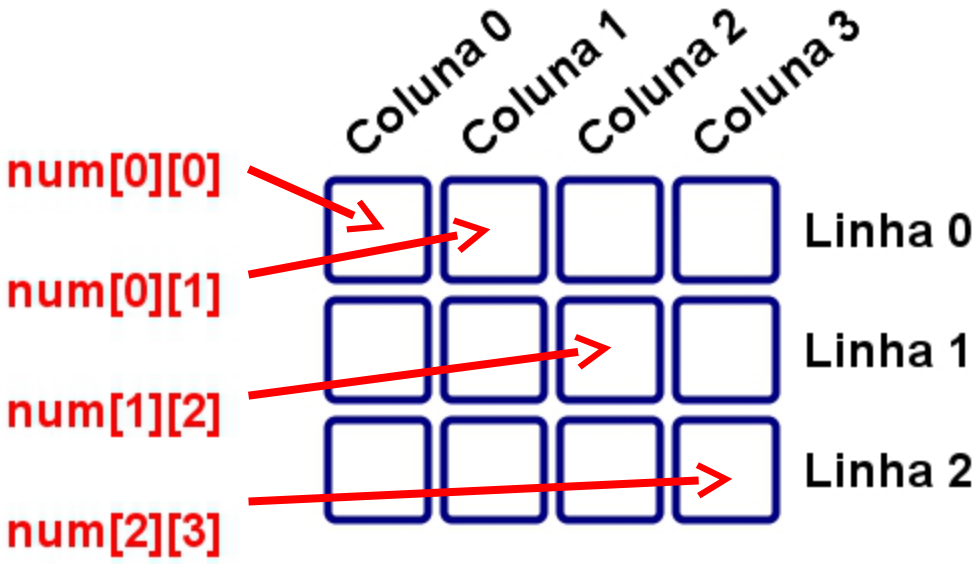
```
#include <stdio.h>
int main() {
    int num[3][4];
}
```

Alocação de Matrizes

A alocação de matrizes é bastante similar à de vetores com a diferença de que temos ***duas indicações*** de ***multiplicidade***:

Exemplo:

```
#include <stdio.h>
int main() {
    int num[3][4];
}
```



Aloca uma matriz de identificador ***num***, com 3 linhas e 4 colunas, onde cada elemento é do tipo *int*.

Inicialização de Matrizes

A inicialização de matrizes é feita da mesma forma que vetores: passamos a coleção completa de valores entre **{ }**, separados por vírgula.

Importante: o preenchimento da matriz é feito linha a linha.

Ex: Inicializar matriz com valores iniciais, colocados entre **{ }**:

```
int a[4][3] = {1,2,3,4,5,6,7,8,9,10,11,12};
```

Este programa aloca uma matriz de $4 \times 3 = 12$ posições, e a preenche com valores de 1 a 12.

$$a = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \\ 10 & 11 & 12 \end{pmatrix}$$

Qual o elemento a[2][1]?

Inicialização de Matrizes

Assim como vetores, podemos declarar matrizes de: **`int`**, **`float`**, **`double`** e **`char`**. Exemplos:

```c
#include <stdio.h>
int main() {
    int m[3][3];
}
```

Declara uma matriz de 3 linhas e 3 colunas cujo conteúdo é lixo de memória.

```c
#include <stdio.h>
int main() {
    int m[3][4] = {1,2,3,4,5,6,6,5,4,3,2,1};
}
```

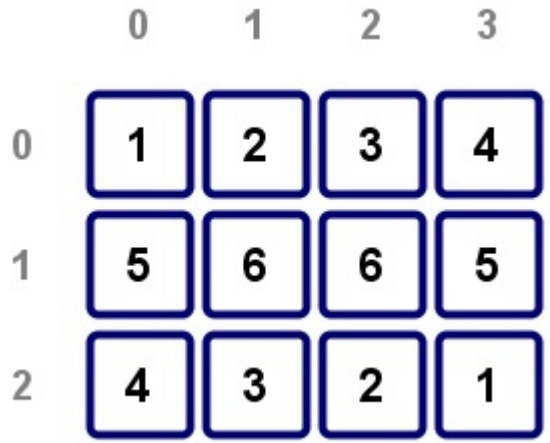
Declara a inicializa uma matriz com 3 linhas e 4 colunas

Inicialização de Matrizes

A declaração abaixo cria uma matriz com 12 números inteiros, dispostos em 3 linhas e 4 colunas, onde todos os elementos do vetor são inicializados na declaração.

```
#include <stdio.h>
int main() {
    int m[3][4] = {1,2,3,4,5,6,6,5,4,3,2,1};
}
```

Representação:

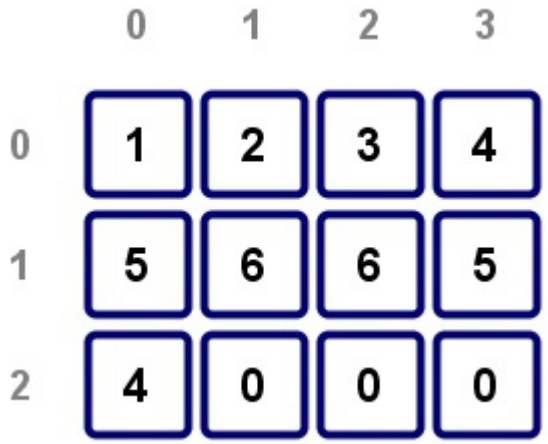


Inicialização de Matrizes

A declaração abaixo cria uma matriz com 12 números inteiros, dispostos em 3 linhas e 4 colunas, porém, alguns elementos não são inicializados na declaração, sendo substituídos por 0 (zero).

```
#include <stdio.h>
int main() {
    int m[3][4] = {1,2,3,4,5,6,6,5,4};
}
```

Representação:



Impressão de Matrizes

Imprime a matriz sem formatação.

```
#include <stdio.h>
#define LINHAS 3     //numero de linhas
#define COLUNAS 4    //numero de colunas
int main() {
    int m[LINHAS][COLUNAS] = {1,2,3,4,5,6,6,5,4,3,2,1};
    int i, j;

    for(i = 0; i < LINHAS; i++) {      //percorre linhas
        for(j = 0; j < COLUNAS; j++) {//percorre colunas
            printf("m[%d][%d] = %d\n", i, j, m[i][j]);
        }
    }
}
```

Saída:

```
m[0][0] = 1
m[0][1] = 2
m[0][2] = 3
m[0][3] = 4
m[1][0] = 5
m[1][1] = 6
m[1][2] = 6
m[1][3] = 5
m[2][0] = 4
m[2][1] = 3
m[2][2] = 2
m[2][3] = 1
```

Impressão de Matrizes

Imprime a matriz com formatação.

```c
#include <stdio.h>
#define LINHAS 3     //numero de linhas
#define COLUNAS 4    //numero de colunas
int main() {
    int m[LINHAS][COLUNAS] = {1,2,3,4,5,6,6,5,4,3,2,1};
    int i, j;

    for(i = 0; i < LINHAS; i++) {      //percorre linhas
        for(j = 0; j < COLUNAS; j++) {//percorre colunas
            printf("%3d", m[i][j]);
        }
     printf("\n");
    }
}
```

Saída:

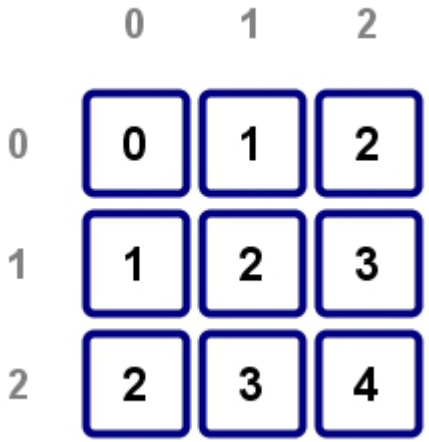
```
  1  2  3  4
  5  6  6  5
  4  3  2  1
```

Inicialização de matrizes

```c
#include <stdio.h>
int main(){
    int m[3][3], i, j;
    for(i = 0; i < 3; i++){
        for(j = 0; j < 3; j++){
            m[i][j] = i + j;
        }
    }
}
```

Para preencher matrizes, normalmente utilizamos **laços dentro** de **laços** (**um** para **linhas**, **outro** para **colunas**).

Este programa aloca uma matriz de 9 inteiros onde cada elemento é a soma dos índices da linha e da coluna:



Inicialização de Matrizes

Preencher uma matriz com valores digitados pelo usuário:

```c
#include <stdio.h>
#define LINHAS 3     //numero de linhas
#define COLUNAS 4    //numero de colunas
int main() {
    int matriz[LINHAS][COLUNAS], i, j;
    for(i = 0; i < LINHAS; i++) {
        for(j = 0; j < COLUNAS; j++) {
            printf("Digite matriz[%d][%d]: ", i, j);
            scanf("%d", &matriz[i][j]);
        }
    }
}
```

Acesso a elementos da matriz

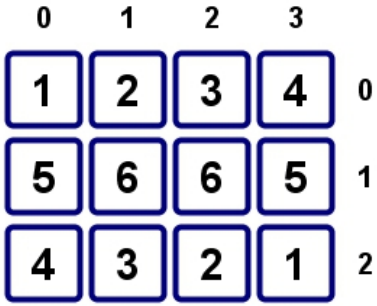
- Assim como em vetores, cada elemento de uma matriz de comporta como uma variável individual do tipo especificado.

  Para acessarmos os elementos da matriz através do índice, utilizamos [ ][ ] para especificar linha e coluna.

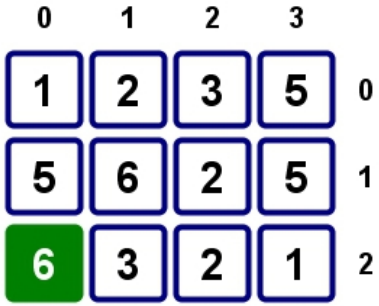
```
a[1][2] = 5;
x = a[0][0] - 23;
a[3][1] = a[0][0] + a[1][0];
```

Assim como em vetores, os índices vão de 0 a N-1, tanto para linha quanto para coluna.

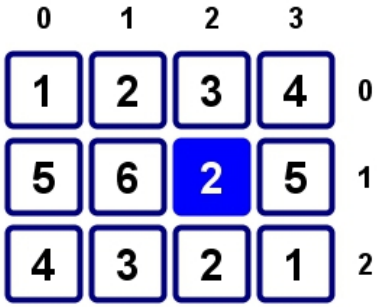
Acesso a elementos da matriz



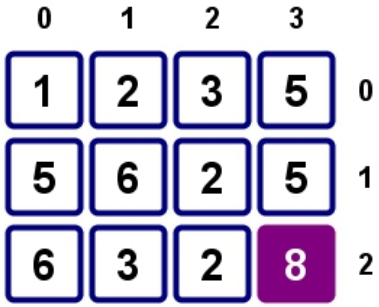
m[m[0][1]][m[0][0]-1] = 6;



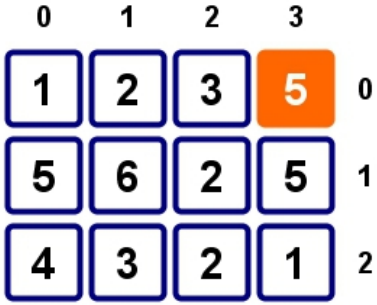
m[1][2] = 2;



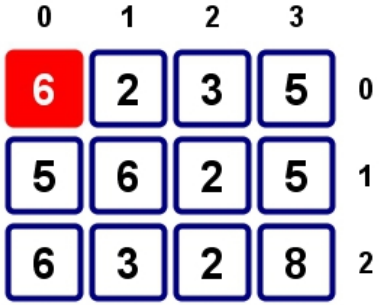
m[m[2][2]][3] = m[1][1]+2;



m[m[2][3]-1][3] = 5;



m[0][2*3-6] = m[2][3]-m[1][2];



Utilização de matrizes

Uma imagem (.jpg), é uma matriz!

- Imagem em preto e branco: utiliza 1 matriz com escalas de cinza entre 0 e 255.

Utilização de matrizes

Uma imagem (.jpg), é uma matriz!

- Imagem em preto e branco: utiliza 1 matriz com escalas de cinza entre 0 e 255.

Utilização de matrizes

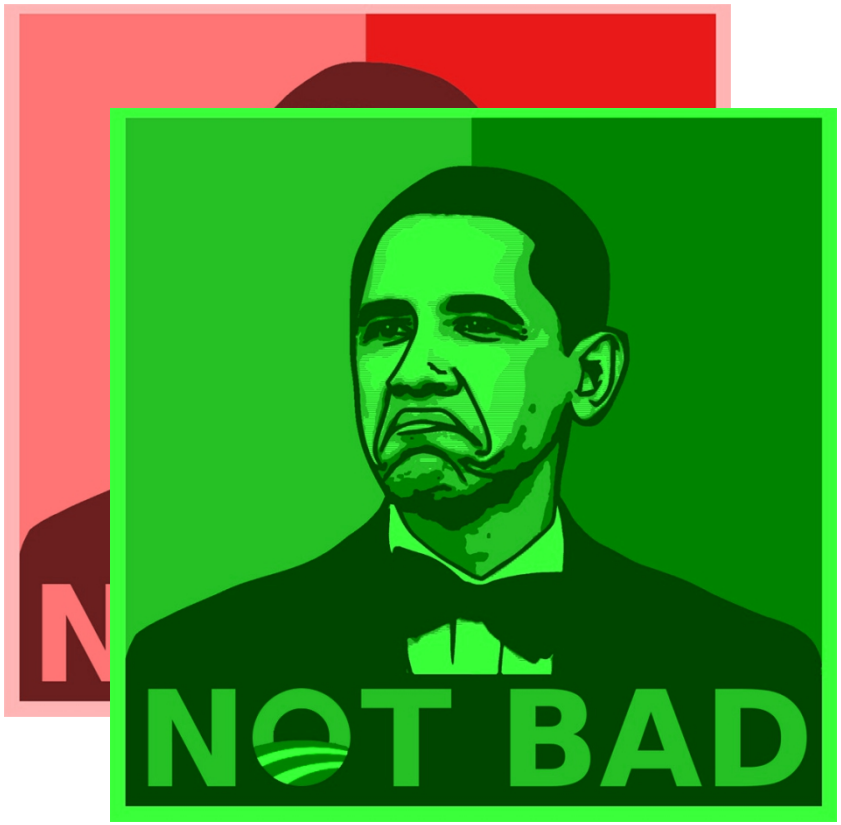
Uma imagem (.jpg), é uma matriz!

- Imagem colorida: utiliza 3 matrizes (R,G,B).

Utilização de matrizes

Uma imagem (.jpg), é uma matriz!

- Imagem colorida: utiliza 3 matrizes (R,G,B).



Utilização de matrizes

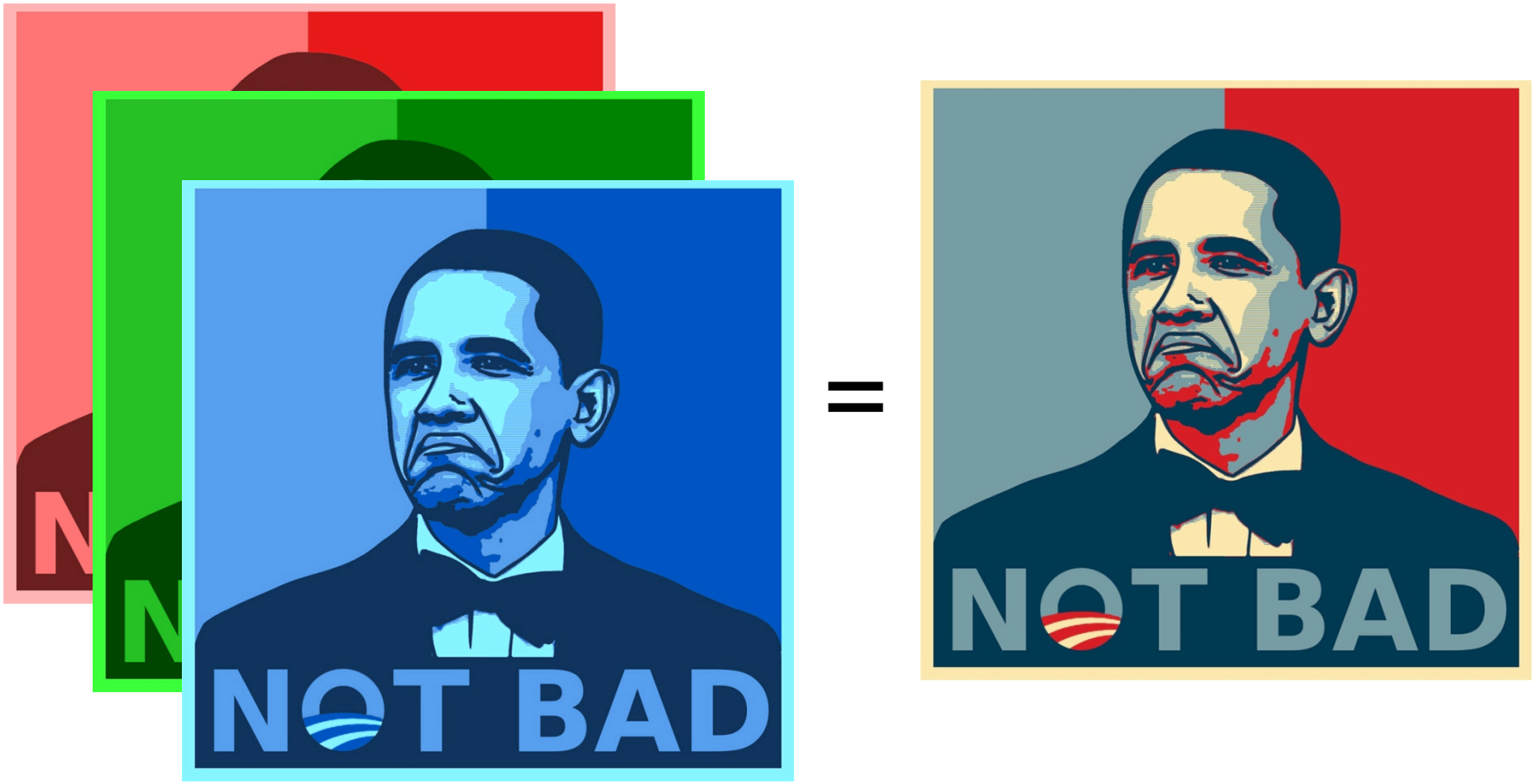
Uma imagem (.jpg), é uma matriz!

- Imagem colorida: utiliza 3 matrizes (R,G,B).

Utilização de matrizes

Uma imagem (.jpg), é uma matriz!

- Imagem colorida: utiliza 3 matrizes (R,G,B).



Exercícios

1. Escreva um programa em C que declara uma matriz 3x2 não inicializada. O programa deve ler os elementos da matriz do usuário, e após escrever a matriz lida (de preferência em formato matricial).
2. Escreva um programa em C que aloca uma matriz de 3x3 de inteiros, inicializada na declaração (escolha você os valores), e calcule e escreva a quantidade de números ímpares que ocorrem na matriz.
3. Escreva um programa em C que declara uma matriz 4x4, inicializada na declaração e a imprime formatada apenas com a diagonal principal.
4. Escreva um programa em C que aloca três matrizes 3x3: M1, M2 e M3. As duas primeiras devem ser inicializadas na definição. A matriz M3 deve ser calculada como M1 + M2, e escrita para o usuário.
5. Escreva uma programa em C que lê uma matriz 3x3 do usuário e escreve seu determinante.