



Departamento de Computação

CCENS – UFES

Programação I

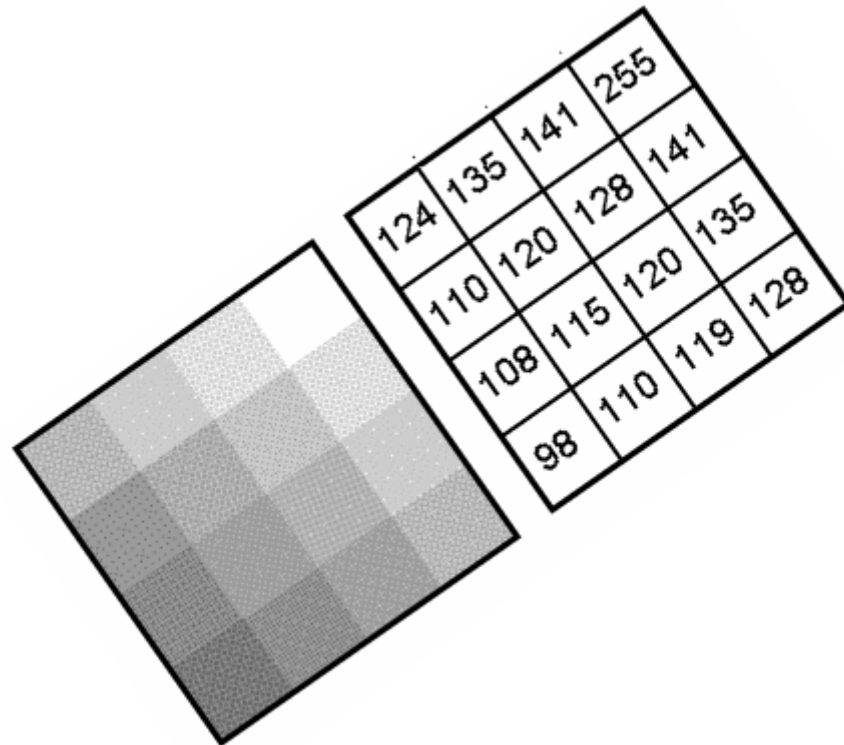
```
public class Boneco2D
{
    private Rectangle quadra;
    private Texture2D textura;
    private bool falado;
    private int altura, largura;

    //construtor
    public Boneco2D(int posX, int posY, Texture2D t)
    {
        // inicializa o boneco com altura e largura proporcionais a suas coordenadas na Tela
        if (posX == 0 || posY == 0)
        {
            posX = 10;
            posY = 10;
        }

        float prop = (float)posX / ((float)T.height);
        altura = (int)Math.Round(T.height * prop);
    }
}
```

❖ Variáveis compostas homogêneas

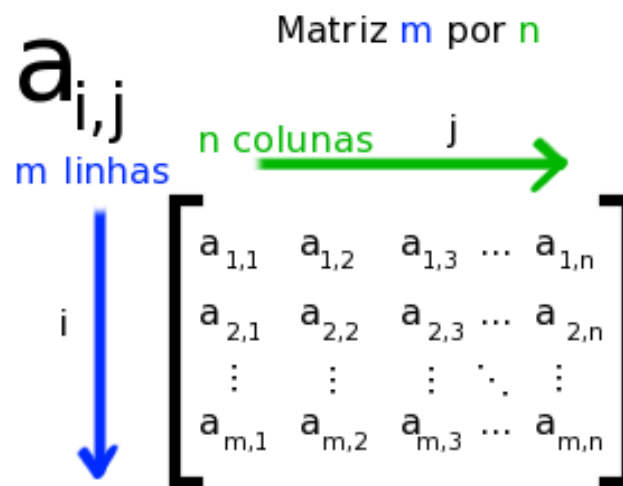
- Multidimensionais (o uso de *arrays* para a representação de matrizes e outras abstrações)



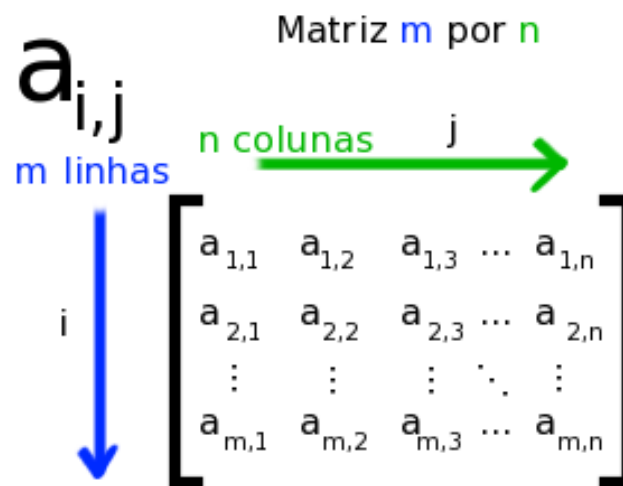
- ❖ Definiremos uma matriz, sem perda de generalidade, como uma tabela de $m \times n$ símbolos que representam valores de algum domínio.
- ❖ As **linhas horizontais** de uma matriz são chamadas **linhas**, e as **linhas verticais** são chamadas **colunas**.



- ❖ Uma matriz com **m** linhas e **n** colunas é chamada de uma matriz **m** -por- **n** (escreve-se **m** x **n**)
 - **m** e **n** são chamadas de suas **dimensões**, **tipo** ou **ordem**.



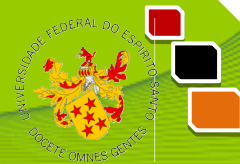
- ❖ Um elemento de uma **matriz A** que está na **i -ésima** linha e na **j -ésima** coluna é chamado de elemento **i,j** ou **(i,j) -ésimo** elemento de A.
 - Referência: $a_{i,j}$ ou $a[i,j]$.



- ❖ Uma matriz onde uma de suas dimensões é igual a **1** geralmente é chamada de **vetor**.
 - Uma matriz $1 \times n$ (**uma linha** e **n** colunas) é chamada de **vetor linha** ou **matriz linha**,
 - Uma matriz $m \times 1$ (**uma coluna** e **m** linhas) é chamada de **vetor coluna** ou **matriz coluna**.



Representação de Matrizes



- ❖ Assim como os *arrays/vetores unidimensionais*, nas linguagens de programação, os elementos do *array* podem estar indexados a partir de 1 (Fortran, MATLAB, R, etc) ou a partir de 0 (C e seus dialetos).
 - Por exemplo, o elemento $a(1,1)$ em Fortran corresponde ao elemento $a[0][0]$ em C.
- ❖ Em C, só podem ser utilizados como números inteiros como índices e a primeira posição do vetor sempre será índice 0 (zero).



- ❖ Esse tipo de *array/vetor* é definido de maneira muito similar ao que já vimos antes.
- ❖ A única coisa que precisamos fazer é incluir mais uma faixa variação de índice:

```
<tipo> <identificador> [d1] [d2];
```

- Onde

- **d1** determina o tamanho da dimensão 1 (**linhas**) e
- **d2** determina o tamanho da dimensão 2 (**colunas**)

Representação de Matrizes



❖ Exemplo

```
int A[4][5];
```

linhas \ colunas	0	1	2	3	4
0	30	0	0	37	0
1	0	44	0	0	0
2	0	0	0	25	0
3	0	0	0	0	0

A[2][3]

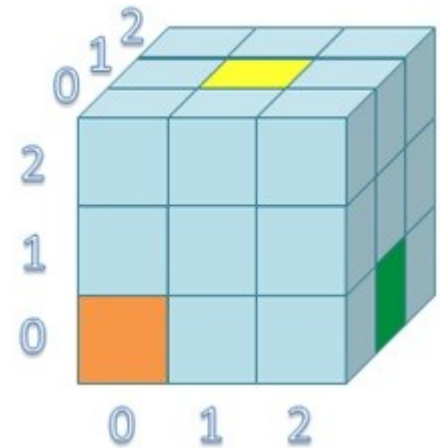
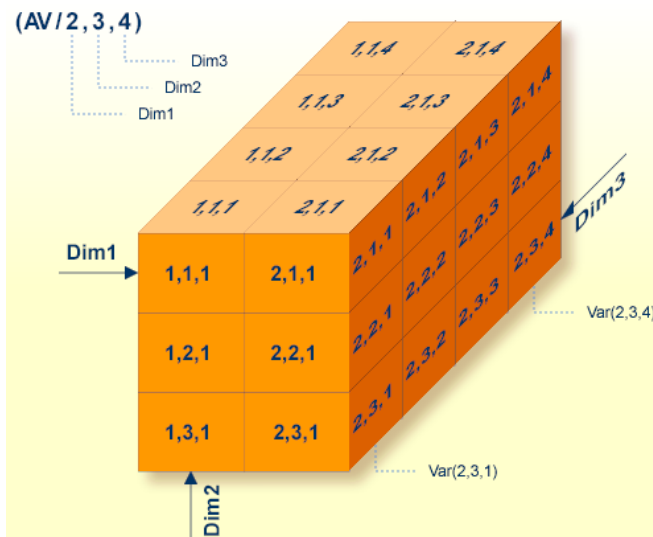
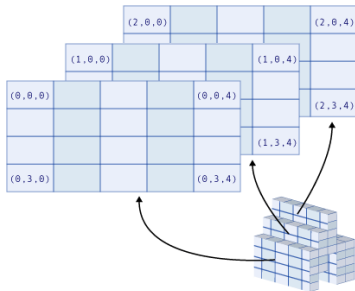


- ❖ Os *arrays* de duas dimensões podem ser usados para
 - armazenar tabelas;
 - fazer cálculos matemáticos (como resolução de sistemas lineares e transformações lineares);
 - fazer desenhos, etc.

Arrays com mais de 2 dimensões

- ❖ E se precisar declarar uma estrutura de dados tridimensional, ou quadridimensional, etc.? basta incluir mais faixas de índices:

```
<tipo> identificador [d1] [d2] [d3] ;
```

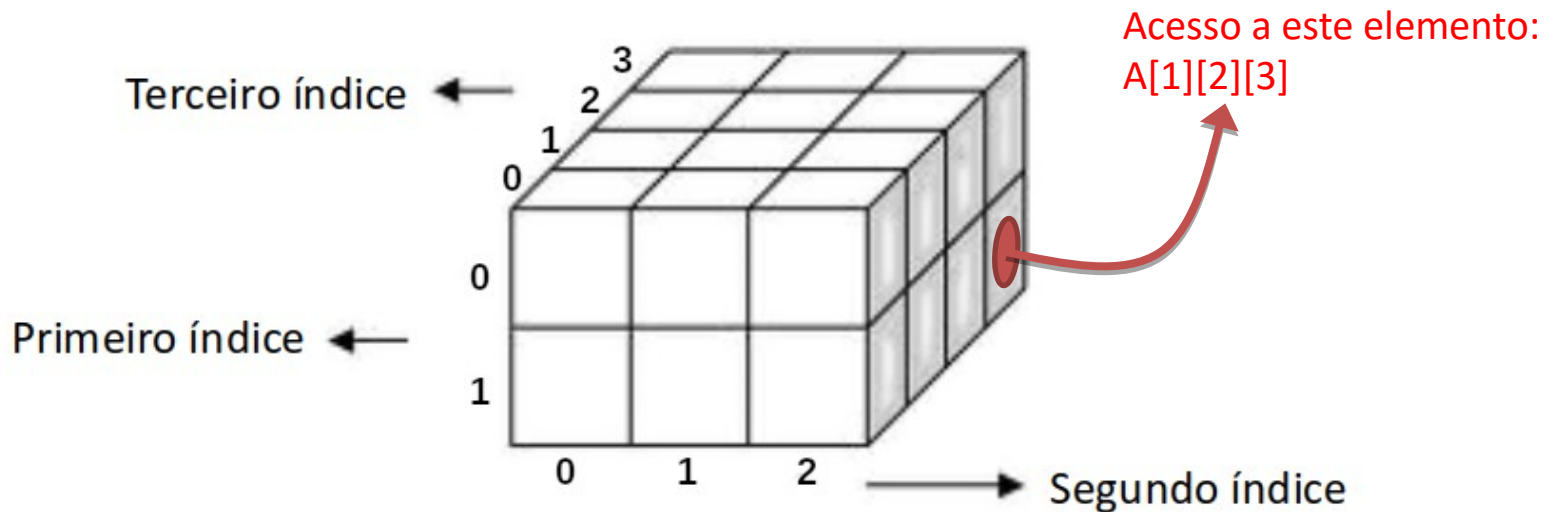


Arrays com mais de 2 dimensões



❖ Exemplo de *array* tridimensional em C com 24 posições (variáveis):

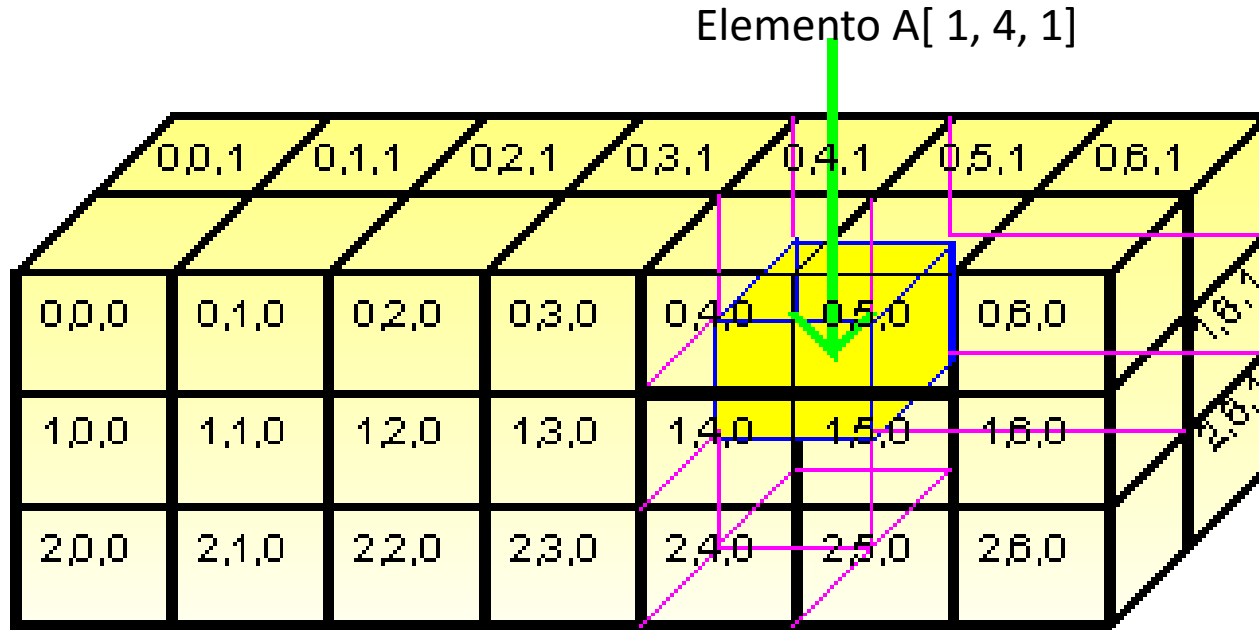
```
int A[2][3][4];
```



Arrays com mais de 2 dimensões

- ❖ Além disso, em um *array*, tridimensional, se usarmos a abstração de um cubo subdividido em várias posições, podemos pensar em posições de armazenamento que estão no “interior” do cubo.

```
int A[3][7][2];
```



❖ Exemplo de aplicação:

- Suponha que desejamos guardar no computador uma lista de alunos com suas respectivas notas em todas as disciplinas cursadas durante o ano.

❖ Exemplo de aplicação:

Uma coleção de dados!

- A seguinte tabela mostra uma lista de 6 alunos e as notas em 8 disciplinas.

Disciplinas Nomes	Notas por disciplina								Média
	1	2	3	4	5	6	7	8	
Joãozinho	7.7	8.1	7.0	6.0	8.5	9.5	6.8	9.0	
Dorinha	6,0	9,0	7,0	7,0	10,0	5,0	10,0	6,0	
Luizinho	8,0	6,0	8,0	10,0	5,0	5,0	9,0	9,0	
Mariazinha	10,0	8,0	5,0	8,0	8,0	8,0	9,0	10,0	
Pedrinho	6,0	8,0	7,0	7,0	10,0	10,0	9,0	5,0	
Marquinho	9,0	7,0	6,0	7,0	8,0	5,0	6,0	8,0	

- ❖ Gostaríamos de fazer um programa para armazenar essa tabela no computador e depois poder calcular a **média** de todas as **notas** de todos os **alunos**.
- ❖ Para saber o desempenho da turma poderíamos calcular a **média** das **médias**.



Matrizes – Aplicações – 1



- ❖ Para começar vamos pensar apenas em como armazenar as notas na memória do computador.
 - Para isso vamos utilizar um **array bidimensional**;



❖ Para o nosso problema de notas:

```
int notas[6][8];
```

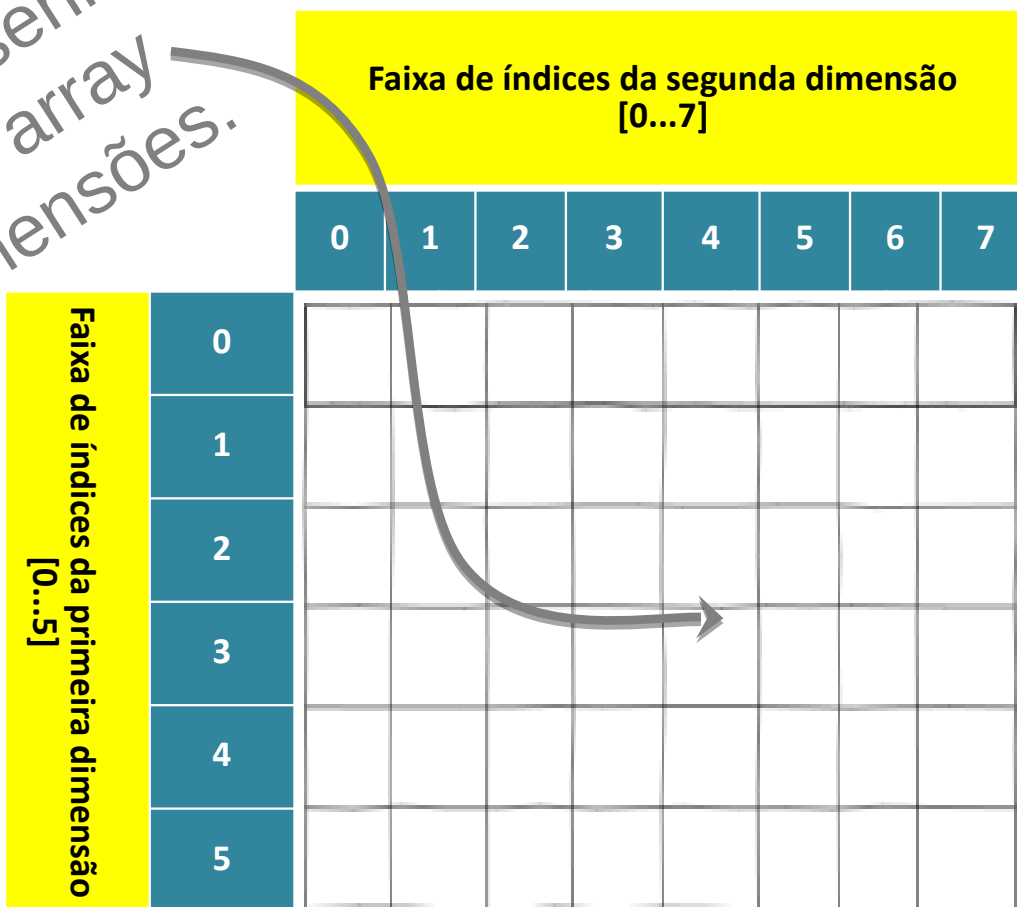
- O número 6 indica a faixa de índices da primeira dimensão do *array* (as linhas);
- O número 8 indica a faixa de índices para a segunda dimensão (as colunas);

Matrizes – Aplicações – 1

Para o problema das notas,
será necessário reservar **6 linhas** e **8 colunas**

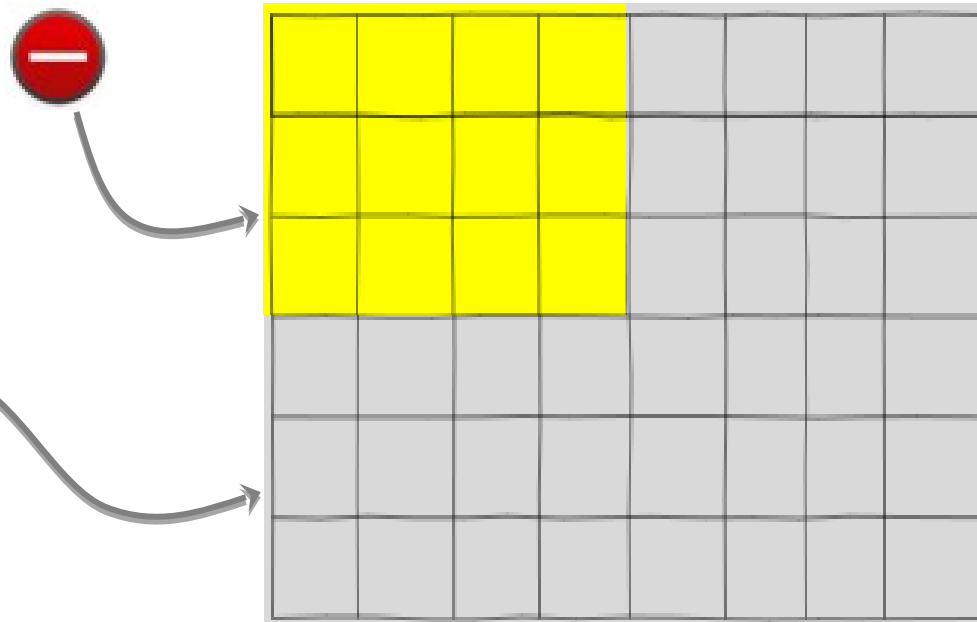
Este é nosso desenho
abstrato de um array
com duas dimensões.

Vulgo: Matriz



❖ OBSERVAÇÃO:

- Em algumas situações, se não soubermos com antecedência qual será o tamanho da estrutura de dados, podemos declarar uma dimensão maior, deixando assim uma “folga” no *array* que represente a “matriz”.



Fiquem aí que uma hora posso precisar!

- ❖ Para manter todas as informações da tabela no computador, poderíamos armazenar também os **nomes** dos alunos em um *array* de *String* (*char**) (um vetor) e as médias dos alunos num *array* de números reais (outro vetor).



Matrizes – Aplicações – 1


- Agora todos os dados poderão ser armazenados!!!

Nomes(0...5)

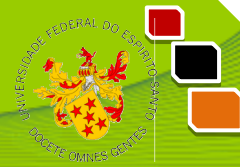
Notas (0...5,0...7)

Médias (0..5)

	0	1	2	3	4	5	6	7	
0									
1									
2									
3									
4									
5									



Matrizes – Aplicações – 1



❖ Mais um detalhe...

Notas (0..5,0..7)

	0	1	2	3	4	5	6	7
0								
1								
2								
3								
4								
5								

❖ Como faço para manipular todos os dados em um *array* com duas dimensões?

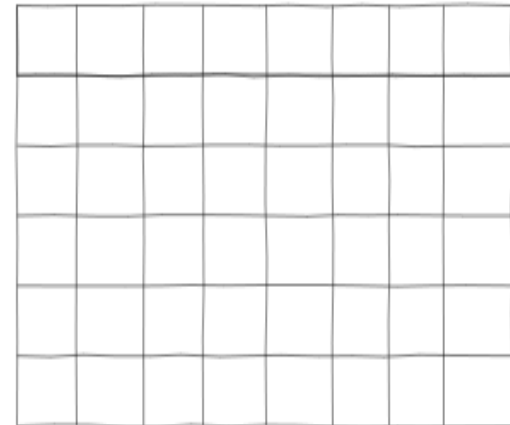


Matrizes – Aplicações – 1

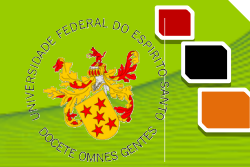


Temos que ler todos os dados da primeira linha, da segunda, da terceira...

Precisaremos de duas variáveis para representar os índices

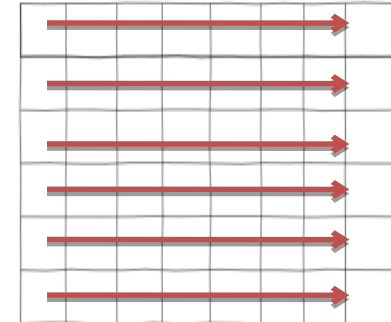


Matrizes – Aplicações – 1



Algoritmo básico:

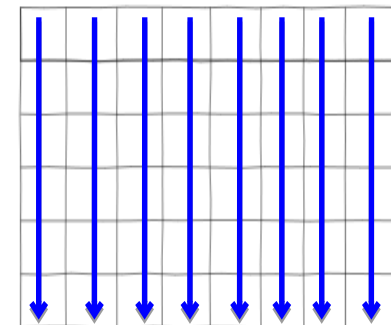
- Para **cada linha**:
 - Percorra **todas** as **colunas**
 - Processando os dados



```
for (i=0;i<6;i++)  
  for (j=0;j<6;j++)  
    notas[i][j]
```

OU:

- Para **cada coluna**:
 - Percorra **todas** as **linhas**
 - Processando os dados

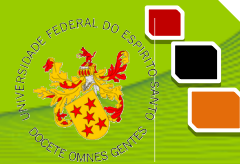


```
for (i=0;i<6;i++)  
  for (j=0;j<6;j++)  
    notas[j][i];
```



- ❖ O próximo slide mostra o código fonte de um programa que faz a leitura dos nomes, das notas dos alunos, calcula as medias, guarda em um *array* e exhibe os resultados na tela do computador.
 - Obs: O código não exhibe mensagens para informar o usuário sobre o que está sendo solicitado como entrada, mas relaxe, isto é apenas um exemplo para quem está aprendendo a programar, na prática você vai querer colocar todas as mensagens necessárias para que o usuário não fique perdido ao usar seus programas.

Matrizes – Aplicações – 1



```
#include <stdio.h>
int main() {
    float notas[6][8], media[6], soma;
    char nomes[6][20];
    int i, j;

    for (i=0;i<6;i++) {
        printf("Digite o nome do aluno %d: ", i);
        scanf("%s", nomes[i]);
    }

    for (i=0;i<6;i++)
        for (j=0;j<8;j++) {
            printf("Digite a nota do aluno %s na disciplina %d: ", nomes[i], j);
            scanf("%f", &notas[i][j]);
        }

    // vamos somar as notas dos alunos:
    for (i=0;i<6;i++) {
        soma = 0;
        for (j=0;j<8;j++) {
            soma += notas[i][j];
        }
        media[i] = soma/8;
    }

    // média das médias:
    soma = 0;
    for (i=0;i<6;i++)
        soma += media[i];

    // Sumário:
    for (i=0;i<6;i++)
        printf("Média do aluno %d - %s: %.2f\n",
            i, nomes[i], media[i]);

    printf("Média da turma: %.2f\n", soma/6);
}
```

❖ Soma de duas matrizes:

- Dadas duas matrizes A e B , determinar a soma de A e B .
- ❖ A soma de duas matrizes é bastante simples, apenas deve-se somar os elementos correspondentes de cada matriz e o resultado pode ser armazenado em uma outra matriz.
- ❖ Faça um programa para fazer essa soma.



❖ Como fazer:

- Para cada linha i e coluna j o elemento da matriz resultante será calculado como:
 - $C[i][j] \leftarrow A[i][j] + B[i][j];$
- Logo, deve-se elaborar um programa que:
 - “Leia” as dimensões das matrizes
 - “Leia” as matrizes
 - Calcule a matriz soma conforme a expressão acima.
 - Exiba o resultado
- Nota:
 - Para poder somar duas matrizes, elas devem ter as mesmas dimensões.



- ❖ Os elementos de cada linha devem ser digitados deixando um espaço, como mostra o seguinte exemplo:

Exemplo de entrada:

Digite a dimensão das matrizes:

3 3

Digite os elementos da matriz A:

1 4 2

0 5 1

2 2 8

Digite os elementos da matriz B:

1 2 3

1 0 1

2 3 2



```
#include <stdio.h>
int main() {
    float matrizA[10][10], matrizB[10][10], soma[10][10];
    int linhas, colunas, i, j;

    printf("Digite as dimensões das matrizes:\n");
    scanf("%d %d", &linhas, &colunas);

    printf("Digite os elementos da Matriz A:\n");
    for (i=0;i<linhas;i++)
        for (j=0;j<colunas;j++)
            scanf("%f", &matrizA[i][j]);

    printf("Digite os elementos da Matriz B:\n");
    for (i=0;i<linhas;i++)
        for (j=0;j<colunas;j++)
            scanf("%f", &matrizB[i][j]);
    //fazendo a soma:
    for (i=0;i<linhas;i++)
        for (j=0;j<colunas;j++)
            soma[i][j] = matrizA[i][j]+matrizB[i][j];

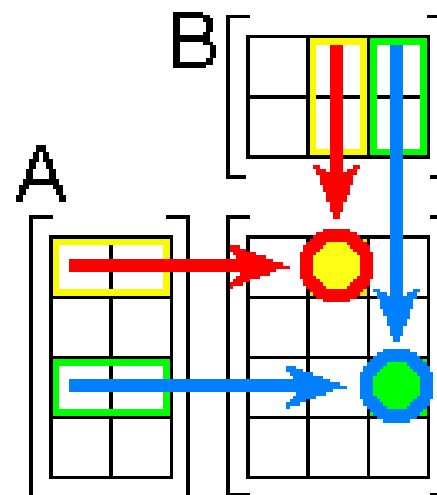
    printf("Resultado:\n");
    for (i=0;i<linhas;i++) {
        for (j=0;j<colunas;j++)
            printf("%6.2f ",soma[i][j]);
        printf("\n");
    }
}
```

❖ Produto matricial

- Na figura abaixo é ilustrado como calcular o elemento (1,2) e o elemento (3,3) de AB se A é uma matriz 4×2 , e B é uma matriz 2×3

$$(AB)_{1,2} = \sum_{r=1}^2 a_{1,r} b_{r,2} = a_{1,1} b_{1,2} + a_{1,2} b_{2,2}$$

$$(AB)_{3,3} = \sum_{r=1}^2 a_{3,r} b_{r,3} = a_{3,1} b_{1,3} + a_{3,2} b_{2,3}$$



- ❖ Tarefa: Façam um programa que faça o produto matricial de duas matrizes