

# Aula 14: Variáveis compostas - matrizes

---

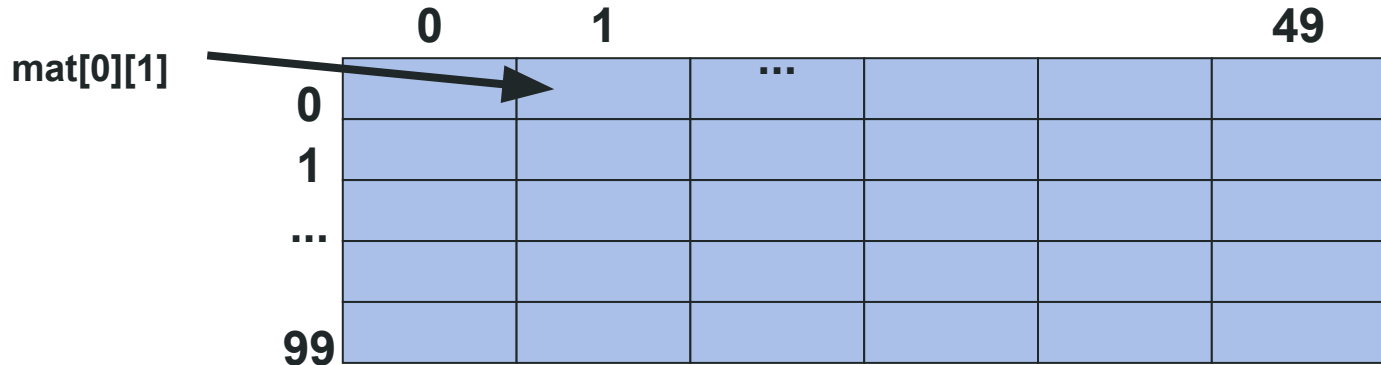
Joice Otsuka

\*Com base no livro Linguagem C: completa e descomplicada, de André Backes

# Matrizes

# Matrizes

- Arrays bidimensionais ou “matrizes”, contêm:
  - Dados organizados na forma de uma tabela de 2 dimensões;
  - Necessitam de dois índices para acessar uma posição: um para indicar a linha e outro para a coluna



Matriz 100x50

Saída Retorno	Sábado set. 28	Domingo set. 29	Segunda-Feira set. 30	Terça-Feira out. 1	Quarta-Feira out. 2	Quinta-Feira out. 3	Sexta-Feira out. 4
Sexta-Feira out. 4	R\$ 1.267	R\$ 1.288	R\$ 1.241	R\$ 1.228	R\$ 1.320	R\$ 1.328	Q
Sábado out. 5	R\$ 1.323	R\$ 1.319	R\$ 1.392	R\$ 1.265	R\$ 1.320	R\$ 1.332	R\$ 1.326
Domingo out. 6	R\$ 1.312	R\$ 1.280	R\$ 1.210	R\$ 1.141	R\$ 1.195	R\$ 1.210	R\$ 1.319
Segunda-Feira out. 7	R\$ 1.306	R\$ 1.307	R\$ 1.313	R\$ 1.210	R\$ 1.291	R\$ 1.210	R\$ 1.210
Terça-Feira out. 8	R\$ 1.319	R\$ 1.293	R\$ 1.250	R\$ 1.223	R\$ 1.304	R\$ 1.223	R\$ 1.319
Quarta-Feira out. 9	R\$ 1.287	R\$ 1.291	R\$ 1.210	R\$ 1.141	R\$ 1.290	R\$ 1.141	R\$ 1.284
Quinta-Feira out. 10	R\$ 1.285	R\$ 1.288	R\$ 1.254	R\$ 1.240	R\$ 1.292	R\$ 1.212	R\$ 1.276

		0	1	2	3	4	5
	CIDADES	Maceió	Manaus	Natal	Palmas	Porto Alegre	Porto Velho
0	Maceió	0	5.491	572	1.851	3.572	4.505
1	Manaus	2.779	0	5.985	4.141	4.563	901
2	Natal	434	2.765	0	2.345	4.066	4.998
3	Palmas	1.383	1.509	1.527	0	2.747	1.712
4	Porto Alegre	2.775	3.132	3.172	2.222	0	3.662
5	Porto Velho	3.090	761	3.179	1.711	2.706	0



Distância rodoviária

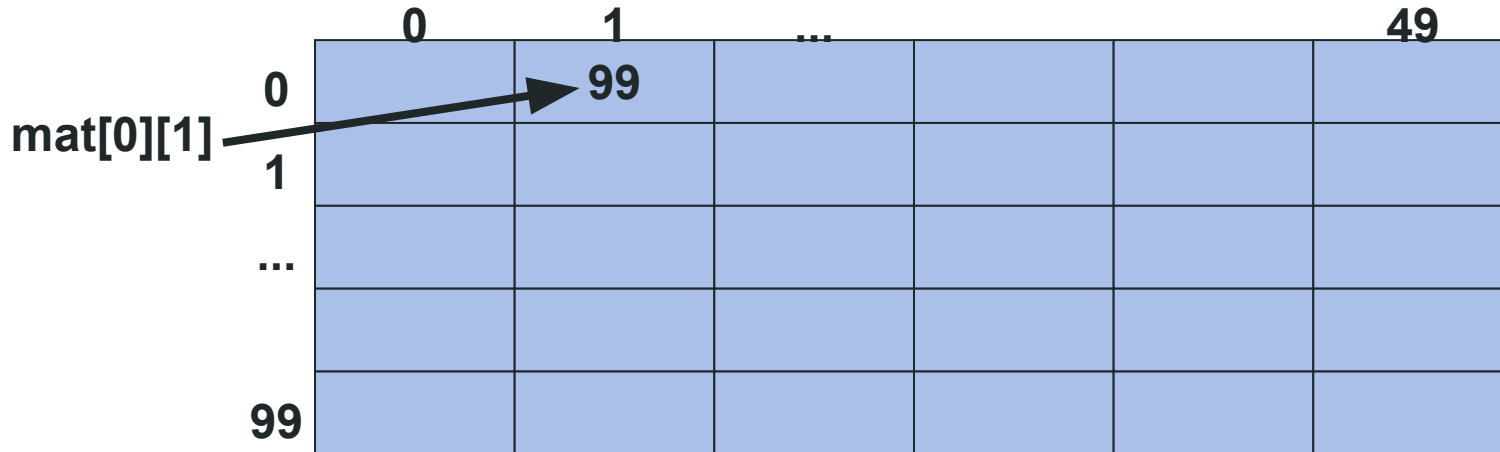
Distância aérea

- Declaração
  - **tipo\_variável** nome\_variável[NroLinhas][NroColunas];
  - **Exemplo:** declaração de uma matriz de inteiros com 100 linhas e 50 colunas

```
int mat[100][50];
```

# Matrizes

- Em uma matriz, os elementos são acessados especificando um par de colchetes e índice para cada dimensão da matriz
  - A numeração dos índices começa sempre de zero
  - Ex: `mat[0][1] = 99;` // atribui valor 99 à posição na linha 0 e coluna 1



	0	1	...				49
0		99					
1							
...							
99							

# Matrizes

- Cada elemento de uma matriz tem todas as características de uma variável e pode aparecer em expressões e atribuições (respeitando os seus tipos)
  - `mat[0][1] = x + mat[1][5];`
  - `if (mat[5][7] > 0) ...`



# Percorrendo matrizes

- Como uma matriz possui dois índices, precisamos de dois loops de repetição para percorrer todos os seus elementos Para percorrer linha a linha, precisamos:
  - Loop externo para percorrer cada linha
    - Loop interno para percorrer as colunas de cada linha

```
#include <stdio.h>
```

```
int main() {  
    int mat[100][50];  
    int i,j;  
    for (i = 0; i < 100; i++){  
        for (j = 0; j < 50; j++){  
            printf("Digite o valor de mat[%d][%d]: ",i,j);  
            scanf("%d",&mat[i][j]);  
        }  
    }  
}
```

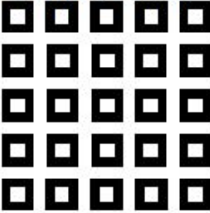
Percurso linha  
por linha

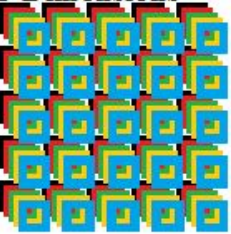
E para percorrer coluna por  
coluna?

# Arrays Multidimensionais

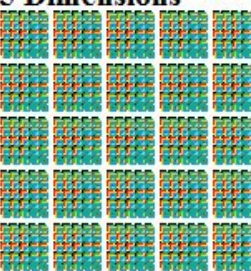
- Arrays podem ter diversas dimensões, cada uma identificada por um par de colchetes na declaração
  - `int vet[5];` // 1 dimensão
  - `float mat[5][5];` // 2 dimensões
  - `double cub[5][5][5];` // 3 dimensões
  - `int X[5][5][5][5];` // 4 dimensões

1 Dimension  


2 Dimensions  


3 Dimensions  


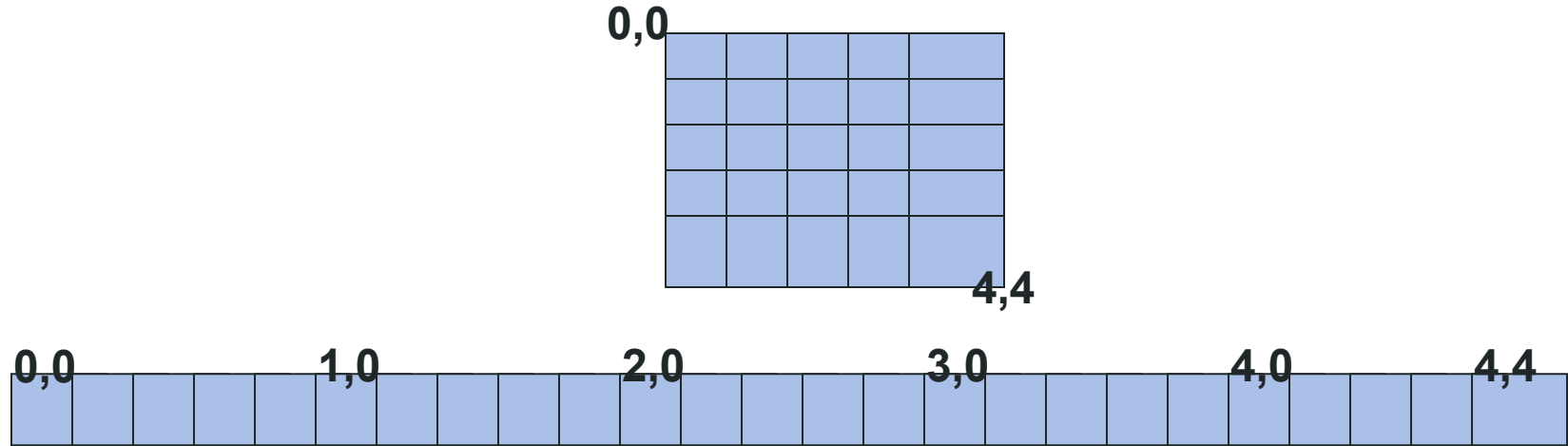
4 Dimensions  


5 Dimensions  


Etc...

# Arrays Multidimensionais

- Apesar de terem o comportamento de estruturas com mais de uma dimensão, na memória os dados são armazenados linearmente:
  - `int mat[5][5];`



# Arrays Multidimensionais

- Um array N-dimensional funciona basicamente como outros tipos de array. Basta lembrar que o índice que varia mais rapidamente é o índice mais à direita.
  - `int vet[5];` // 1 dimensão
  - `float mat[5][5];` // 2 dimensões
  - `double cub[5][5][5];` // 3 dimensões
  - `int X[5][5][5][5];` // 4 dimensões

# Inicialização

- Arrays podem ser inicializados com certos valores durante sua declaração. A forma geral de um array com inicialização é:

**tipo\_da\_variável** nome\_da\_variável [**tam1**] ... [**tamN**] = {dados};

# Inicialização

- A lista de valores é composta por valores (do mesmo tipo do array) separados por vírgula.
- Os valores devem ser dados na ordem em que serão colocados na matriz (linha a linha)

```
float vetor[3] = {1.5, 22.1, 4.56};  
int mat1[3][4] = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12};  
int mat2[3][4] = {{1, 2, 3, 4}, {5, 6, 7, 8}, {9, 10, 11, 12}};
```

# Inicialização sem tamanho

- Inicialização sem especificação de tamanho
  - Nesse tipo de inicialização, o compilador vai considerar o tamanho do dado declarado (elementos atribuídos) como sendo o tamanho do array.
  - Ocorre durante a compilação e não poderá mais ser mudado durante o programa.

# Inicialização sem tamanho

- Inicialização sem especificação de tamanho

```
#include <stdio.h>

int main() {

    //A string mess terá tamanho 36.
    char mess[ ] = "Linguagem C: flexibilidade e poder.";

    //O número de linhas de matrxx será 5.
    int matrxx[ ][2] = { 1,2,2,4,3,6,4,8,5,10 };

    return 0;
}
```

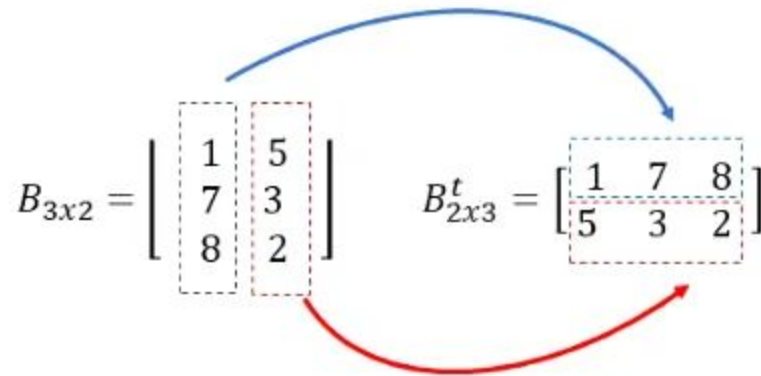


# Exercício

- Leia uma matriz de 3x3 elementos inteiros, imprima a matriz lida e apresente a soma dos seus elementos
- Leia e imprima duas matrizes de números reais de dimensão 2x3. Crie e imprima uma matriz contendo a soma das duas matrizes lidas

# Exercício

- Leia uma matriz de inteiros B (dimensão 3x2) e calcule a matriz transposta  $B^T$  (dimensão 2x3). Imprima B e  $B^T$



# Exercício - Solução

```
int main()
{
    int m[3][2];
    int mt[2][3];
    int i,j;

    for (i=0;i<3;i++){
        for (j=0;j<2;j++){
            scanf("%d",&m[i][j]);
        }
    }
    for (i=0;i<2;i++){
        for (j=0;j<3;j++){
            mt[i][j]=m[j][i];
        }
    }
}
```