

# Matrizes

Prof. Denio Duarte

[duarte@uffs.edu.br](mailto:duarte@uffs.edu.br)

Prof. Claunir Pavan

[claunir.pavan@uffs.edu.br](mailto:claunir.pavan@uffs.edu.br)

# Matrizes

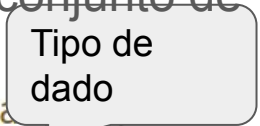
- **Matrizes** são vetores de duas (ou mais) dimensões, ou seja, um vetor onde cada uma das posições também é um vetor
- **C** permite criarmos matrizes de várias dimensões, porém, o mais usual é que uma matriz possua apenas duas dimensões
- Assim como um **Vetor** uma **Matriz** é um conjunto de valores do **mesmo tipo**
- Como declarar:

```
3  int main(){
4      int matriz[10][10];
5      float matriz2[10][10];
6
7      int vetor[1][200];
8
9      return 0;
10 }
```

# Matrizes

- **Matrizes** são vetores de duas (ou mais) dimensões, ou seja, um vetor onde cada uma das posições também é um vetor
- **C** permite criarmos matrizes de várias dimensões, porém, o mais usual é que uma matriz possua apenas duas dimensões
- Assim como um **Vetor** uma **Matriz** é um conjunto de valores do **mesmo tipo**
- Como declarar:

```
3  int main()
4  {
5      int matriz[10][10];
6      float matriz2[10][10];
7
8      int vetor[1][200];
9
10     return 0;
11 }
```



Tipo de dado

# Matrizes

- **Matrizes** são vetores de duas (ou mais) dimensões, ou seja, um vetor onde cada uma das posições também é um vetor
- **C** permite criarmos matrizes de várias dimensões, porém, o mais usual é que uma matriz possua apenas duas dimensões
- Assim como um **Vetor** uma **Matriz** é um conjunto de valores do **mesmo tipo**
- Como declarar:

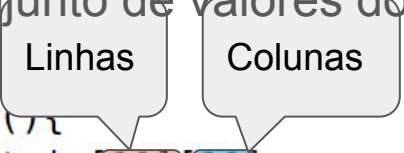
```
3  int main(){  
4      int matriz[10][10];  
5      float matriz2[10][10];  
6  
7      int vetor[1][200];  
8  
9      return 0;  
10 }
```

Nome da  
variável

# Matrizes

- **Matrizes** são vetores de duas (ou mais) dimensões, ou seja, um vetor onde cada uma das posições também é um vetor
- **C** permite criarmos matrizes de várias dimensões, porém, o mais usual é que uma matriz possua apenas duas dimensões
- Assim como um **Vetor** uma **Matriz** é um conjunto de valores do **mesmo tipo**
- Como declarar:

```
3  int main()  
4      int matriz[10][10];  
5      float matriz2[10][10];  
6  
7      int vetor[1][200];  
8  
9      return 0;  
10 }
```



# Matrizes

- **Matrizes** são vetores de duas (ou mais) dimensões, ou seja, um vetor onde cada uma das posições também é um vetor
- **C** permite criarmos matrizes de várias dimensões, porém, o mais usual é que uma matriz possua apenas duas dimensões
- Assim como um **Vetor** uma **Matriz** é um conjunto de valores do **mesmo tipo**
- Como declarar:

```
3  int main  
4      int ma  
5      float  
6  
7      int vetor[1][200];  
8  
9      return 0;  
10 }
```

Mesma coisa que um vetor, já que uma das dimensões é 1

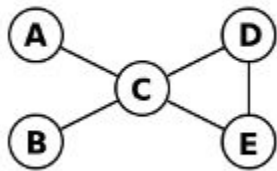
# Matrizes

- Uma matriz pode ser inicializada da seguinte forma:

```
3  int main(){
4      int matriz[3][3] = {
5          |   |   |   |   |   |   |   |   |   |   {00, 01, 02},
6          |   |   |   |   |   |   |   |   |   |   {10, 11, 12},
7          |   |   |   |   |   |   |   |   |   |   {20, 21, 22}
8          |   |   |   |   |   |   |   |   |   |   };
9
10     return 0;
11 }
```

# Matrizes

- Possuem várias utilidades enquanto estrutura de dados.
- Representação de grafos (**spoiler do quarto semestre**):



```
int graph[5][5]={  
    {0,0,1,0,0},  
    {0,0,1,0,0},  
    {1,1,0,1,1},  
    {0,0,1,0,1},  
    {0,0,1,1,0},  
};
```

	A	B	C	D	E
A	0	0	1	0	0
B	0	0	1	0	0
C	1	1	0	1	1
D	0	0	1	0	1
E	0	0	1	1	0

O percurso na matriz pode ser feito fixando a linha e percorrendo as colunas (**duplo for**):

```
for (i=0;i<ROWS;i++)  
    for (j=0;j<COLS;j++)  
        printf("(%d,%d) : %d\n",i,j,m[i][j])
```



# Matrizes

- Dados que podem ser armazenados em forma de tabela:

Produto/Mes	Jan	Fev	Mar	Abr	Mai	Jun
Televisão	500	250	134	320	98	100
Geladeira	124	210	50	70	83	25
Fogão	89	57	74	35	63	95
Ar Condicionado	450	320	120	80	23	43

- Os produtos são representados pelas linhas e os meses pelas colunas
- Temos uma matriz 4x6: `int vprod[4][6]`
- Observe que a matriz armazenará apenas os valores numéricos

# Matrizes

```
#include <stdio.h>
#define ROW 4
#define COL 6
```

```
int main() {
    char prod[ROW][30]={"Televisão","Geladeira","Fogão", "Ar"};
    char meses[COL][4]={"Jan","Fev","Mar","Abr","Mai","Jun"};
    int vprod[ROW][COL]={{500,250,134,320,98,100},{},{},{}};
    int row,col;
    for (row=0;row<ROW;row++)
    {
        printf("%s:\n",prod[row]);
        for (col=0;col<COL;col++)
            printf("    mes: %s quantidade: %d\n",meses[col],vprod[row][col]);
    }
    return 0;
}
```

Produto/Mes	Jan	Fev	Mar	Abr	Mai	Jun
Televisão	500	250	134	320	98	100
Geladeira	124	210	50	70	83	25
Fogão	89	57	74	35	63	95
Ar Condicionado	450	320	120	80	23	43

# Matrizes

- Exercícios
  1. Leia uma matriz 4 x 4 e imprima a diagonal principal.
  2. Leia uma matriz 4 x 4 e escreva a localização (linha e coluna) do maior valor.
  3. Declare uma matriz 5 x 5. Preencha com 1 a diagonal principal e com 0 os demais elementos. Imprima, ao final, a matriz obtida.
  4. Faça um programa que preenche uma matriz 5 x 5 com o produto da posição da linha e da coluna de cada elemento. Em seguida, imprima.
  5. Leia uma matriz 4 x 4 e imprima a triangular superior.