

Arranjos Multidimensionais

Matrizes

**Uso de estruturas de tipo de dados
indexada e homogêneas em C**

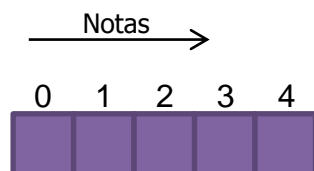
Até agora...

- Você trabalhou com **vetores**, os quais constituem um caso particular de arranjo, e que têm muitas aplicações importantes
- Os arranjos de duas dimensões, ou **matrizes**, ou tabelas, também constituem um caso particular de arranjos, e também têm aplicações importantes na área
- A seguir, você irá estudar os **arranjos bidimensionais**

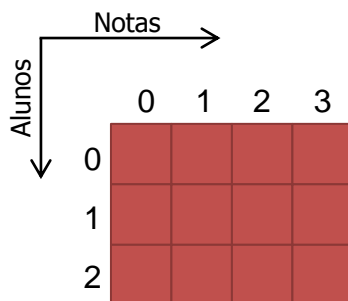
Arranjos Multidimensionais

- A linguagem C nos permite trabalhar com arranjos de várias dimensões:
 - Uma dimensão → vetores
 - Duas dimensões → tabelas ou matrizes
 - Três dimensões ou mais ...

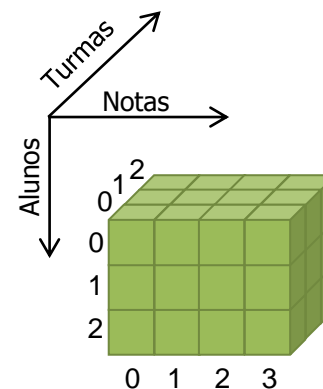
Vetor



**Matriz
Bidimensional**



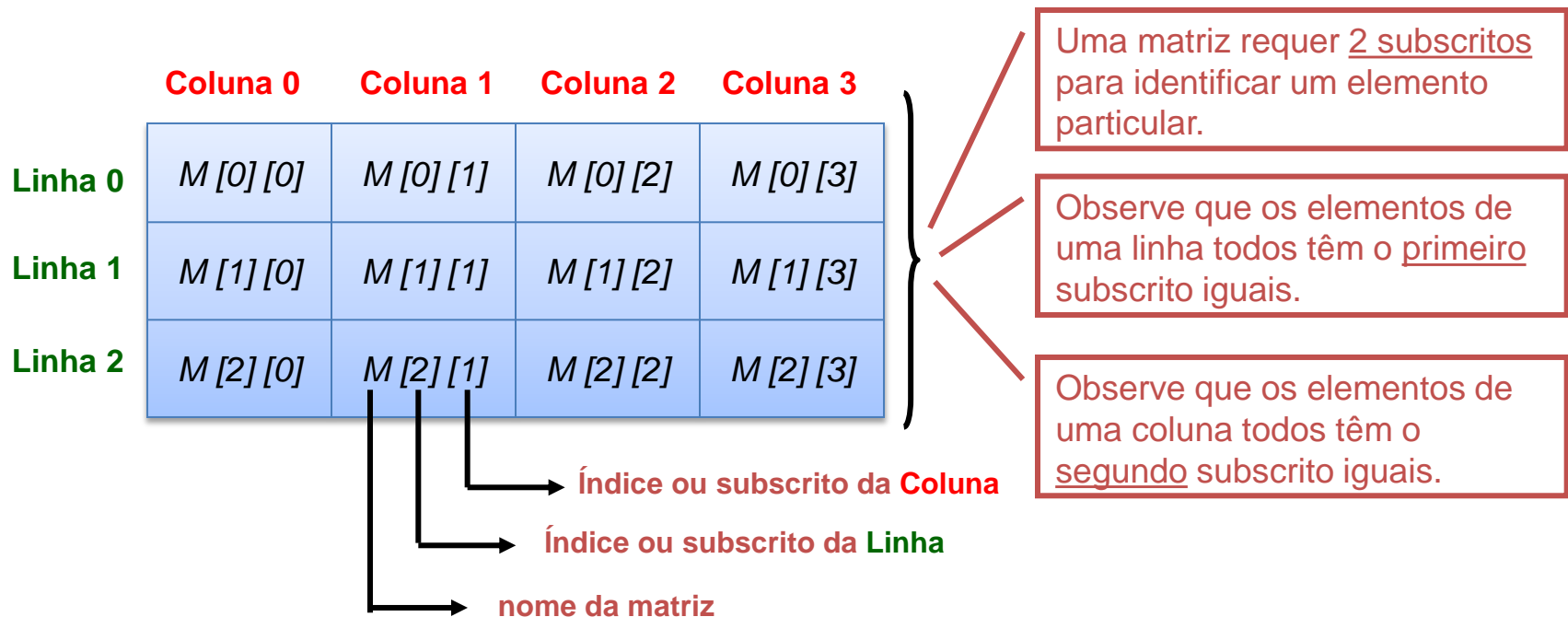
**Matriz
Tridimensional**



Matrizes

- São arranjos bidimensionais, nos quais as informações são organizadas em

Linhas e Colunas



Declaração de Matrizes

Forma geral

tipo **nome** [**dim1**] [**dim2**];

Tipo de dado
(char, int, float, etc.)

Nome da variável
(Escolhido pelo
programador)

**Tamanho de cada
dimensão**
(Quantos valores são
armazenados)

- **Exemplos:**

- Info é uma matriz real de 10 elementos (100 elementos):

float **info**[**10**][**10**];

- Cruzamentos é uma matriz de inteiros de 18x45 elementos:

#define AVENIDAS 18

#define RUAS 45

int **cruzamentos**[**AVENIDAS**][**RUAS**];

Inicialização de Matrizes

- Na declaração

```
int m1[2][2] = {{1, 2}, {3, 4}};
```

Declara uma matriz quadrada 2 x 2 de inteiros e a inicializa da seguinte forma:

$$\longrightarrow m1 = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$$

```
int m2[2][2] = {{1}, {3, 4}};
```

Declara uma matriz quadrada 2 x 2 de inteiros e a inicializa da seguinte forma:

$$\longrightarrow m2 = \begin{bmatrix} 1 & 0 \\ 3 & 4 \end{bmatrix}$$

```
float m3[2][3] = {{1.0, 2.0, 3.0},  
                  {4.0, 5.0, 6.0}};
```

Declara uma matriz 2 x 3 de valores tipo float e a inicializa da seguinte forma:

$$\longrightarrow m3 = \begin{bmatrix} 1.0 & 2.0 & 3.0 \\ 4.0 & 5.0 & 6.0 \end{bmatrix}$$

Inicialização de Matrizes

- Por Leitura
 - Considerando uma matriz de inteiros **m** (4x4)

//Por linhas

```
for (i = 0; i < 4; i++)  
    for (j = 0; j < 4; j++)  
        scanf("%d", &m[i][j]);
```

Neste caso, para cada valor do índice *i* de linha, o índice *j* de coluna varia em toda sua extensão.

Neste caso, para cada valor do índice *j* de coluna, o índice *i* de linha varia em toda sua extensão.

//Por colunas

```
for (j = 0; j < 4; j++)  
    for (i = 0; i < 4; i++)  
        scanf("%d", &m[i][j]);
```

Inicialização de Matrizes

- Por Atribuição
 - Considerando uma matriz de inteiros **m** (4x4)

//Por linhas

```
for (i = 0; i < 4; i++)  
    for (j = 0; j < 4; j++)  
        m[i][j] = i + j;
```

Neste caso, para cada valor do índice *i* de linha, o índice *j* de coluna varia em toda sua extensão.

Neste caso, para cada valor do índice *j* de coluna, o índice *i* de linha varia em toda sua extensão.

//Por colunas

```
for (j = 0; j < 4; j++)  
    for (i = 0; i < 4; i++)  
        m[i][j] = j * 2;
```


Apresentação de Matrizes

- Considerando uma matriz de inteiros **m** (4x4)

//Por linhas

```
for (i = 0; i < 4; i++) {  
    for (j = 0; j < 4; j++)  
        printf("%4d", m[i][j]);  
    printf("\n");  
}
```

Neste caso, para cada valor do índice *i* de linha, o índice *j* de coluna varia em toda sua extensão.

Neste caso, para cada valor do índice *j* de coluna, o índice *i* de linha varia em toda sua extensão.

//Por colunas

```
for (j = 0; j < 4; j++) {  
    for (i = 0; i < 4; i++)  
        printf("%4d", m[i][j]);  
    printf("\n");  
}
```

Exemplos de Aplicação de Matrizes

- Dada uma matriz M (MAXLIN, MAXCOL), preenchê-la por leitura e imprimir:
 1. o maior elemento de cada linha da matriz;
 2. a média dos elementos de cada coluna;
 3. o produto de todos os elementos diferentes de zero;
 4. quantos elementos são negativos;
 5. posição ocupada (linha-coluna) por um elemento cujo valor será lido pelo programa.

Declarações, inicialização e impressão da matriz

```
#include <stdio.h>
#include <stdlib.h>
#define MAXLIN 4
#define MAXCOL 3
int main () {
    int m [MAXLIN][MAXCOL]; //declaração da matriz
    int lin, col; //índices de linha e coluna
    int maior, somacol, produto, negativos, valor, achou;
    float mediacol;
    for (lin = 0; lin < MAXLIN; lin++){ //inicialização da matriz
        for (col = 0; col < MAXCOL; col++){
            printf ("Forneca valor inteiro m[%d][%d]: ", lin, col);
            scanf ("%d", &m[lin][col]);
        }
    }
    for (lin = 0; lin < MAXLIN; lin++){ //impressão da matriz
        printf ("Linha %d\t", lin);
        for (col = 0; col < MAXCOL; col++)
            printf ("\t%d", m[lin][col]);
        printf ("\n");
    } //continua ...
}
```

1. Procurando o maior elemento de cada linha...

```
for (lin = 0; lin < MAXLIN; lin++){ //procurando o maior valor de cada linha
    maior = m [lin] [0]; //supõe que o 1º. É o maior
    for (col = 1; col < MAXCOL; col++){
        if (maior < m[lin][col]) //compara maior com os outros
            maior = m[lin][col]; //se existir algum > que maior, troca
    }
    printf ("\n\nMaior valor da linha %d eh %d", lin, maior);
} //continua ...
```

Usa-se uma variável auxiliar “maior” para guardar o maior elemento, linha por linha

2. Calculando a média dos valores de cada coluna...

Usa-se uma variável auxiliar “somacol” acumular a soma dos elementos da coluna, depois calcula-se a média

```
for (col = 0; col < MAXCOL; col++){ //calculando a media da coluna
    somacol = 0; //a cada coluna, inicializa somacol
    for (lin = 0; lin < MAXLIN; lin++){
        somacol += m[lin][col]; //acumula valores da coluna
    }
    printf ("\n\nA media da coluna %d eh %.2f", col, (float)somacol/MAXLIN);
} //continua ...
```

3. Calculando o produto dos valores != 0 ...

Similar a soma, usa-se uma variável auxiliar “produto” para acumular a multiplicação dos elementos

```
produto = 1; //inicializa em 1, pois eh produto!
for (lin = 0; lin < MAXLIN; lin++){ //calculando o produto dos valores != 0
    for (col = 0; col < MAXCOL; col++){
        if ( m[lin] [col] != 0 )
            produto *= m[lin] [col];
    }
}
printf ("\n\nO produto dos valores != 0 eh: %d", produto);
//continua ...
```

4. Calculando o total de valores negativos ...

```
negativos = 0; //inicializa em 0, pois eh contador!
for (lin = 0; lin < MAXLIN; lin++){ //calculando o total de negativos
    for (col = 0; col < MAXCOL; col++){
        if ( m[lin] [col] < 0 )
            negativos += m[lin] [col];
    }
}
printf ("\n\nO total de valores < 0 eh: %d", negativos);
//continua ...
```

Usa-se uma variável auxiliar “negativos” para **contar** os valores menores que zero

5. Procurando a posição ocupada por um elemento cujo valor será lido pelo programa

```
printf ("\n\nForneca valor a ser procurado: ");
scanf ("%d", &valor);
achou = 0;
for (lin = 0; lin < MAXLIN; lin++) //procurando o valor ..
    for (col = 0; col < MAXCOL; col++)
        if ( m[lin] [col] == valor ){
            achou = 1;
            printf ("\n\nValor estah em: [%d] [%d]", lin, col);
        }
if (!achou)
    printf ("\n\nValor não encontrado!");

//acabou
printf ("\n\n");
system ("pause");
return 0;
} //fim de main
```

Compara-se os elementos da matriz um a um com o valor digitado

Álgebra Matricial: Soma e Multiplicação

- Seja o trecho de código que declara as matrizes A, B e C:

```
...  
#define MAX 100  
int A [MAX] [MAX] ;  
int B [MAX] [MAX] ;  
int C [MAX] [MAX] ;  
...
```

- Como efetuar?
 - A Soma: $C = A_{mn} + B_{mn}$
 - A Multiplicação: $C = A_{mp} \times B_{pn}$

Somando Duas Matrizes

```
for (i = 0; i < MAX; i++) {  
    for (j = 0; j < MAX; j++) {  
        C [i][j] = A [i][j] + B [i][j];  
    }  
}
```

Exemplo:

$$\begin{matrix} A & + & B & = & C \\ \begin{bmatrix} 4 & -2 \\ 1 & 0 \end{bmatrix} & + & \begin{bmatrix} 3 & 2 \\ -1 & 5 \end{bmatrix} & = & \begin{bmatrix} 4+3 & -2+2 \\ 1-1 & 0+5 \end{bmatrix} = \begin{bmatrix} 7 & 0 \\ 0 & 5 \end{bmatrix} \end{matrix}$$

Multiplicando Duas Matrizes

$$C_{ij} = \sum_{k=0}^{n-1} A_{ik} * B_{kj}$$

```
for (i = 0; i < MAX; i++) {  
    for (j = 0; j < MAX; j++) {  
        s = 0;  
        for (k = 0; k < MAX; k++) {  
            s += A[i][k] * B[k][j];  
        }  
        C[i][j] = s;  
    }  
}
```

Arranjos com mais de 2 dimensões...

Exemplo:

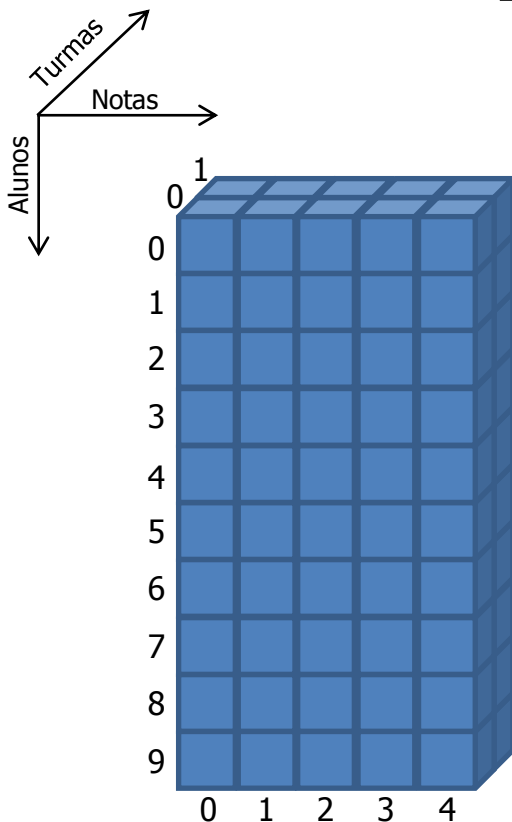
2 turmas

10 alunos em cada turma

5 notas para cada aluno

Ler e armazenar as notas de cada aluno

```
float nt[2][10][5];
```



Declaração arranjos de 3 dimensões

Forma geral

tipo **nome** [**dim1**] [**dim2**] [**dim2**] ;

Tipo de dado
(char, int, float, etc.)

Nome da variável
(Escolhido pelo programador)

Tamanho de cada dimensão
(Quantos valores são armazenados)

- **Exemplo:**

- Armazenar notas de 5 notas de 10 alunos de 2 turmas:

```
#define TURMAS 2
```

```
#define ALUNOS 10
```

```
#define NOTAS 5
```

```
float notas[TURMAS][ALUNOS][NOTAS];
```

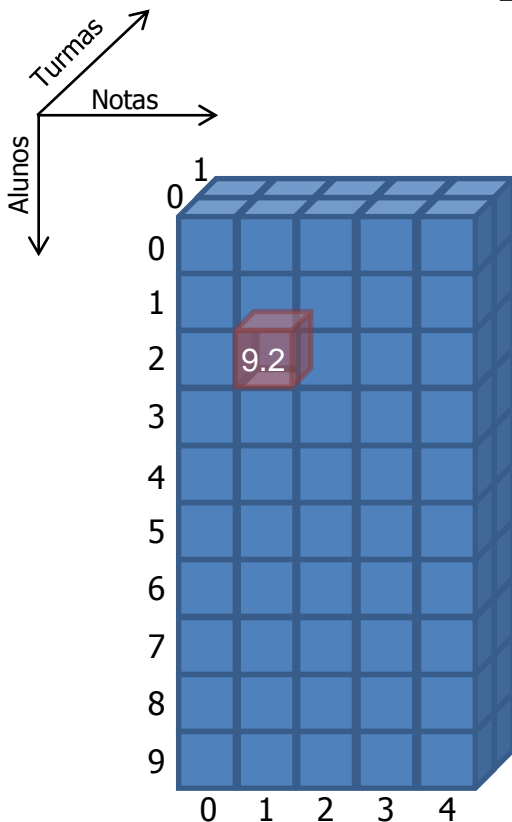
Arranjos com mais de 2 dimensões...

Exemplo:

2 turmas

10 alunos em cada turma

5 notas para cada aluno



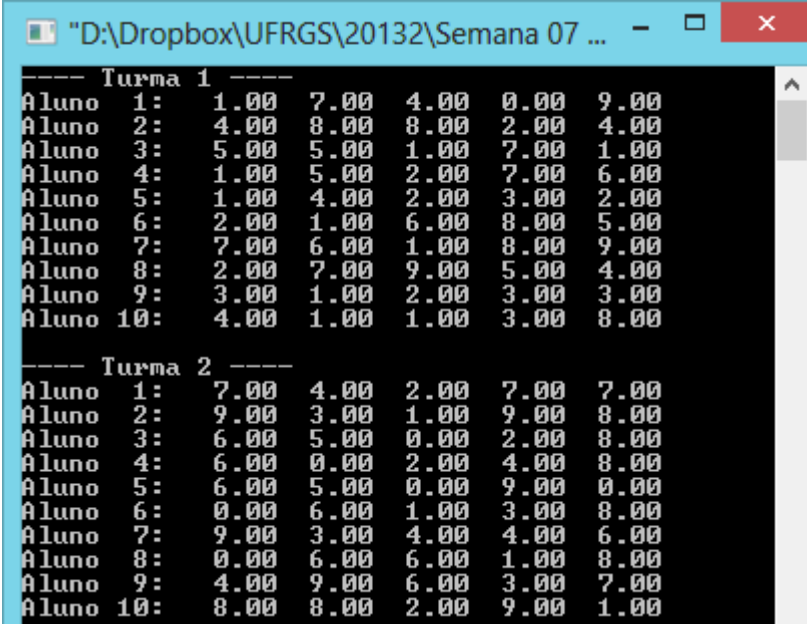
Atribuindo nota 9.2 na segunda nota do terceiro aluno da primeira turma

```
nt[0][2][1] = 9.2;
```

Imprimindo uma matriz 3x3

```
for (i = 0; i < TURMAS; i++){
    printf("---- Turma %d ----\n", i+1);
    for (j = 0; j < ALUNOS; j++){
        printf("Aluno %2d: ", j+1);
        for (k = 0; k < NOTAS; k++){
            printf("%6.2f", notas[i][j][k]);
        }
        printf("\n");
    }
    printf("\n");
}
```

As mesmas regras vão valer para arranjos de quantas dimensões forem necessárias declarar de acordo com cada problema proposto



---- Turma 1 ----					
Aluno 1:	1.00	7.00	4.00	0.00	9.00
Aluno 2:	4.00	8.00	8.00	2.00	4.00
Aluno 3:	5.00	5.00	1.00	7.00	1.00
Aluno 4:	1.00	5.00	2.00	7.00	6.00
Aluno 5:	1.00	4.00	2.00	3.00	2.00
Aluno 6:	2.00	1.00	6.00	8.00	5.00
Aluno 7:	7.00	6.00	1.00	8.00	9.00
Aluno 8:	2.00	7.00	9.00	5.00	4.00
Aluno 9:	3.00	1.00	2.00	3.00	3.00
Aluno 10:	4.00	1.00	1.00	3.00	8.00

---- Turma 2 ----					
Aluno 1:	7.00	4.00	2.00	7.00	7.00
Aluno 2:	9.00	3.00	1.00	9.00	8.00
Aluno 3:	6.00	5.00	0.00	2.00	8.00
Aluno 4:	6.00	0.00	2.00	4.00	8.00
Aluno 5:	6.00	5.00	0.00	9.00	0.00
Aluno 6:	0.00	6.00	1.00	3.00	8.00
Aluno 7:	9.00	3.00	4.00	4.00	6.00
Aluno 8:	0.00	6.00	6.00	1.00	8.00
Aluno 9:	4.00	9.00	6.00	3.00	7.00
Aluno 10:	8.00	8.00	2.00	9.00	1.00

```
return 0;  
} // Fim
```

Dennis, por que
nossos índices
nos arrays
começam em 0?

Ora Ken,
agente nem se
preocupou com
isso!



Ken Thompson & Dennis Ritchie