MCTA028-15: Programação Estruturada

Aula 5: Matrizes (Primeira Parte)

Wagner Tanaka Botelho
wagner.tanaka@ufabc.edu.br / wagtanaka@gmail.com
Universidade Federal do ABC (UFABC)
Centro de Matemática, Computação e Cognição (CMCC)

Introdução

Introdução

- Na aula passada estudamos um *array* com apenas UMA dimensão (**vetor**);
- Alguns casos, uma estrutura com mais de uma dimensão é mais útil:
  - Por exemplo, quando os dados são organizados em uma estrutura de **LINHAS** e **COLUNAS**, como uma **TABELA**:
    - Utiliza-se um *array* com **DUAS** dimensões, ou seja, uma **MATRIZ**.

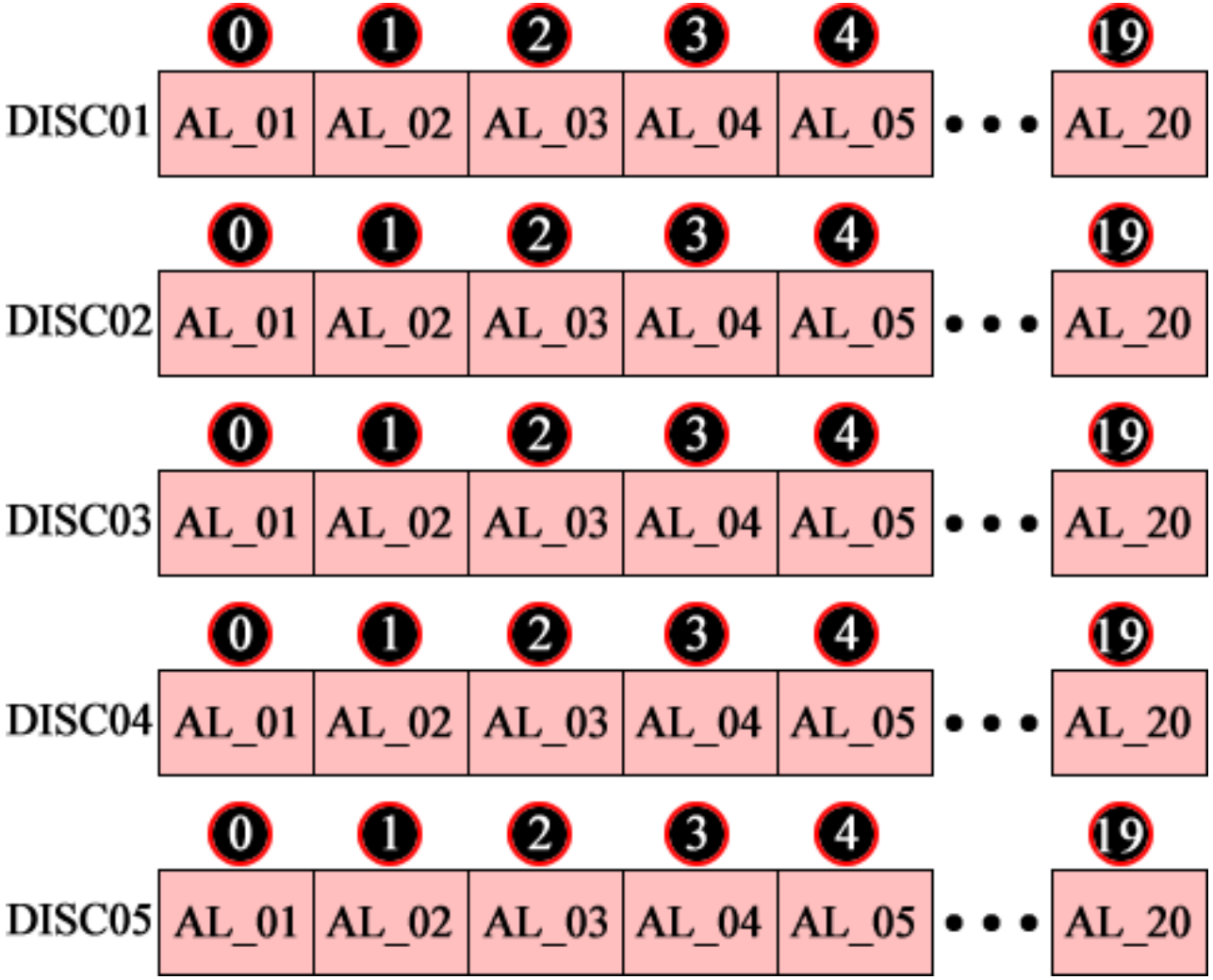
Problema

Problema

- Escreva um programa para armazenar as notas de 20 alunos em 5 disciplinas:
  - Vetores;
  - Matrizes.

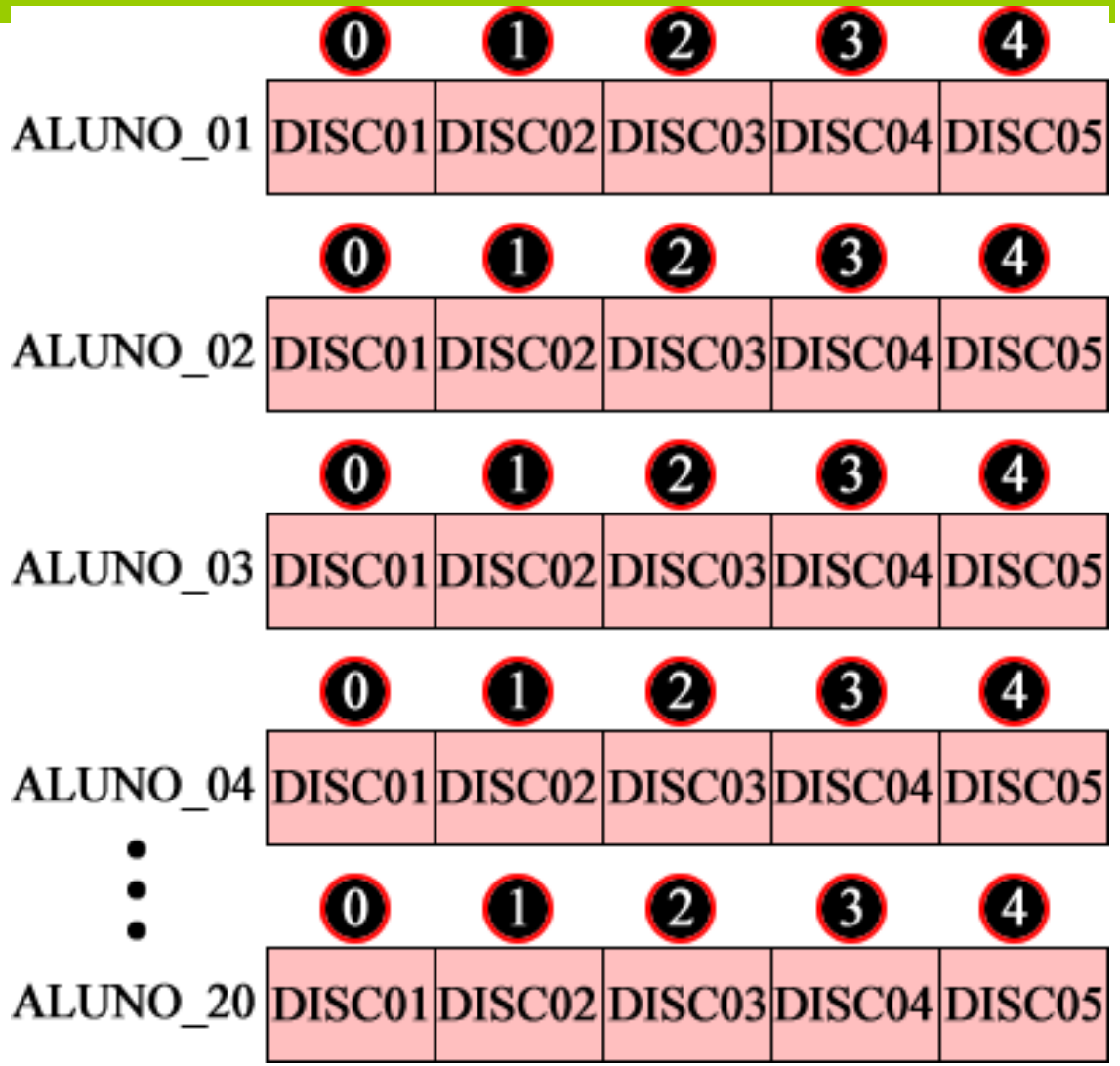
Problema: Vetores

- Escreva um programa para armazenar as notas de 20 alunos em 5 disciplinas:
  - 5 vetores de 20 posições.



Problema: Vetores

- Escreva um programa para armazenar as notas de 20 alunos em 5 disciplinas:
  - 20 vetores de 5 posições.



Problema: Matrizes

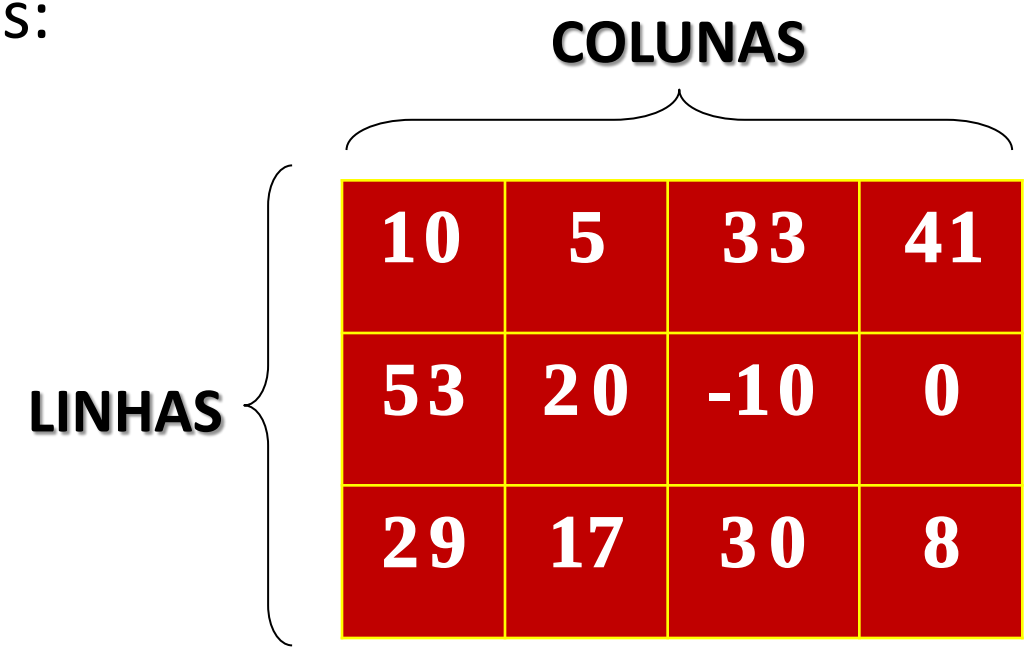
- Escreva um programa para armazenar as notas de 20 alunos em 5 disciplinas:
  - 1 matriz de 20x5.

ALUNOS	DISC01	DISC02	DISC03	DISC04	DISC05
1	5,5	8,3	6,5	10,0	9,5
2	8,0	7,5	8,5	6,5	10,0
3	...	...	...	...	...
4	...	...	...	...	...
5	...	...	...	...	...
6	...	...	...	...	...
7	7,5	8,0	9,0	8,0	8,5
...	6,5	3,5	6,5	4,5	9,0
18	...	...	...	...	...
19	...	...	...	...	...
20	8,0	9,0	8,0	10,0	9,0

Matrizes

- Matrizes são estruturas MULTIDIMENSIONAIS (mais de uma dimensão) capazes de armazenar dados;

- A figura a seguir representa uma matriz BIDIMENSIONAL de números inteiros:



COMO MANIPULAR ESSES DADOS???

Declarando uma Matriz

Declarando uma Matriz

- A forma para declarar uma matriz é:

```
tipo_dado nome_array[nro_linhas][nro_colunas];
```

- Por exemplo, para criar um *array* de **INTEIROS** com 10 linhas e 5 colunas, isto é, uma matriz de inteiros de tamanho **10** × **5**:

```
int mat[10][5];
```

Inicializando uma Matriz

```
void main(){
     int matriz1[2][4] = {1,2,3,4,5,6,7,8};
     int matriz2[2][4] = {{1,2,3,4}, {5,6,7,8}};
}
```

	0	1	2	3
0	1	2	3	4
1	5	6	7	8

Acessando um Elemento

- Para ACESSAR uma determinada posição da matriz:
  - Deve-se usar dois índices:
    - Primeiro especifica a LINHA;
    - Segundo especifica a COLUNA.

```
int mat[10][5];
mat[0][1] = 9;
mat[3][3]=7;
```

	0	1	2	3	4
0	3	9	0	1	2
1	9	8	22	12	3
2	87	23	7	11	22
3	83	7	9	7	11
4	97	73	86	7	7
5	29	83	7	0	1
6	20	39	8	7	0
7	9	73	2	92	0
8	8	18	23	4	22
9	98	87	3	6	2

Cada dimensão da MATRIZ começa no índice **ZERO** e termina sempre em **N-1**, em que **N** é o número de elementos da matriz.

Entrada de Dados

Matrizes: Entrada de Dados

- Escreva um programa para:
  - INSERIR números inteiros em uma matriz 3 x 3;

*Ex_01.c

```
1   #include<stdio.h>
2   void main(){
3       int mat[3][3];
4
5       for(int i=0; i<3; i++){
6           for(int j=0; j<3; j++){
7               scanf("%i", &mat[i][j]);
8           }
9       }
10  }
```

mat

	0	1	2
0	4	6	1
1	8	3	2
2	7	5	1

i	j	mat[i][j]
~~0~~	~~0~~	mat[0][0]=(4)
	~~1~~	mat[0][1]=(6)
	~~2~~	mat[0][2]=(1)
	~~3~~	
~~1~~	~~0~~	mat[1][0]=(8)
	~~1~~	mat[1][1]=(3)
	~~2~~	mat[1][2]=(2)
	~~3~~	
~~2~~	~~0~~	mat[2][0]=(7)
	~~1~~	mat[2][1]=(5)
	~~2~~	mat[2][2]=(1)
	3	
3		

Saída de Dados

Matrizes: Saída de Dados

- Escreva um programa para:
  - INSERIR números inteiros em uma matriz 3 x 3;
  - IMPRIMIR os números inseridos.

Ex_02.c

```
11      for(int i=0; i<3; i++){
12          for(int j=0; j<3; j++){
13              printf("%i ", mat[i][j]);
14          }
15          printf("\n");
16      }
17  }
```

```
 9      }
10
11      nt i=0; i<3;
12      or(int j=0;
13          printf("
14
15      rintf("\n");
16
17  }
```

mat

	0	1	2
0	4	6	1
1	8	3	2
2	7	5	1

Saída

```
4 6 1
8 3 2
7 5 1
```

i	j	mat[i][j]
0	0	mat[0][0]={4 }
	1	mat[0][1]={6 }
	2	mat[0][2]={1 }
	3	
1	0	mat[1][0]={8 }
	1	mat[1][1]={3 }
	2	mat[1][2]={2 }
	3	
2	0	mat[2][0]={7 }
	1	mat[2][1]={5 }
	2	mat[2][2]={1 }
	3	
3		

Array Com Mais de Duas Dimensões

Array Com Mais de Duas Dimensões

- Vimos como criar arrays com UMA dimensão (vetor) ou DUAS dimensões (matriz);
- A Linguagem C permite criar um array com MAIS de DUAS dimensões:
  - Cada dimensão do array é definida por um par de colchetes na sua declaração.

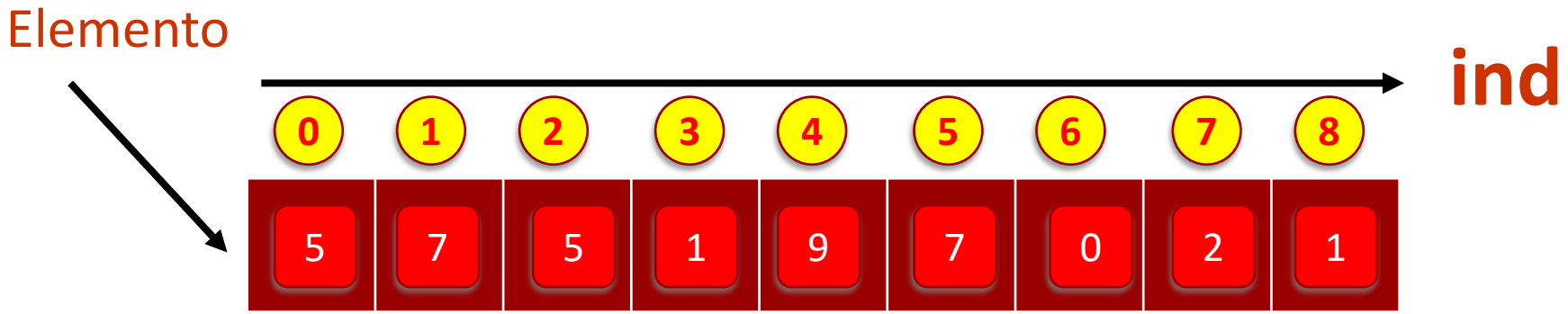
```
#include <stdio.h>
void main(){
   //declara array de int com 1 dimensão
   int vet[5];
   //declara array de float com 2 dimensões
   float mat[5][5];
   //declara array de double com 3 dimensões
   double cub[5][5][5];
   //declara array de int com 4 dimensões
   int X[5][5][5][5];
}
```

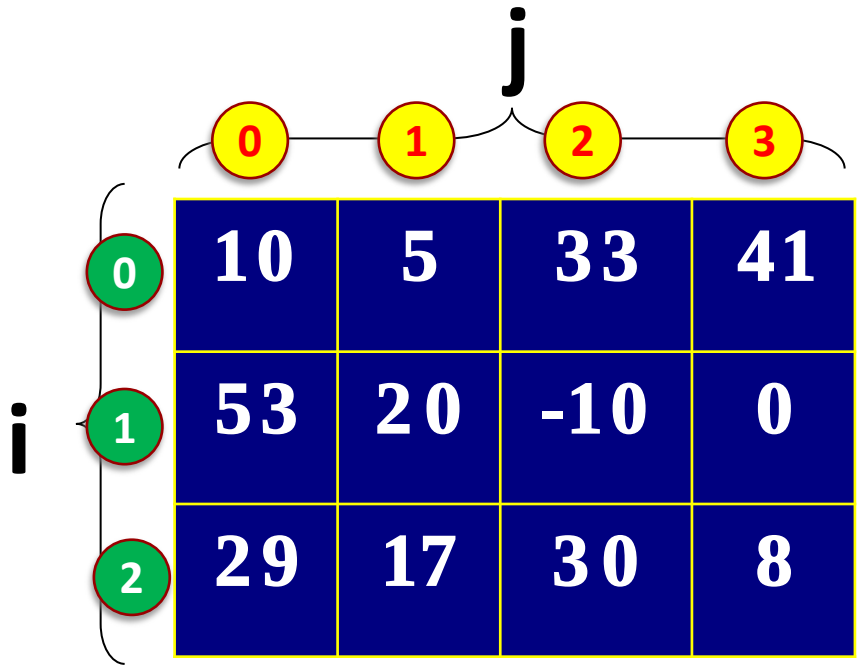
Array Com Mais de Duas Dimensões

```
#include <stdio.h>
void main(){
   int cub[5][5][5];
   int i,j,k;
   //preenche o array de 3 dimensões com zeros
   for (i=0; i < 5; i++){
      for (j=0; j < 5; j++){
         for (k=0; k < 5; k++){
            cub[i][j][k] = 0;
         }
      }
   }
}
```

Reforçar os Conceitos

Vetor x Matriz





Vetor	Matriz
Declarar	
int a[8];	int a[8][8];
Armazenar Valores	
a[0]=10; a[1]=3; a[2]=4;	a[0][0]=1; a[0][1]=2; a[1][0]=5; ....
Armazenar Valores pelo Teclado	
scanf(“%i”, &a[0]); scanf(“%i”, &a[1]); scanf(“%i”, &a[2]);	scanf(“%i”, &a[0][0]); scanf(“%i”, &a[0][1]); scanf(“%i”, &a[1][0]); ....

Referências

- Slides do Prof. Luiz Rozante;
- SALES, André Barros de; AMVAME-NZE, Georges. Linguagem C: roteiro de experimentos para aulas práticas. 2016;
- BACKES, André. Linguagem C Completa e Descomplicada. Editora Campus. 2013;
- SCHILDT, Herbert. C Completo e Total. Makron Books. 1996;
- DAMAS, Luís. Linguagem C. LTC Editora. 1999;
- DEITEL, Paul e DEITEL, Harvey. C Como Programar. Pearson. 2011.