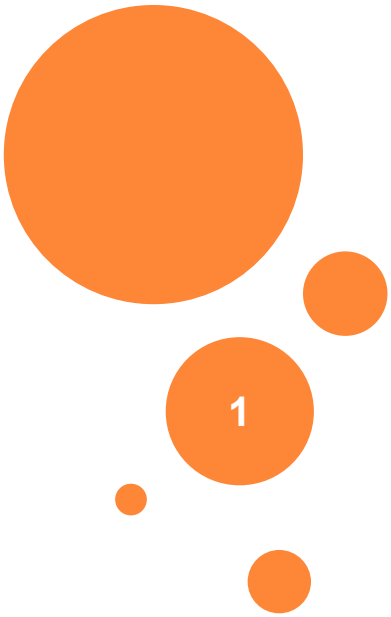


LINGUAGEM C: ARRAY: MATRIZES

Prof. Humberto Razente

Sala 1B144

ARRAYS BIDIMENSIONAIS - MATRIZES

- Os arrays declarados até o momento possuem apenas uma dimensão e, portanto, são tratados como uma lista de variáveis

  - Porém, há casos em que 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. Para isso usamos um array com duas dimensões, ou seja, uma "matriz".

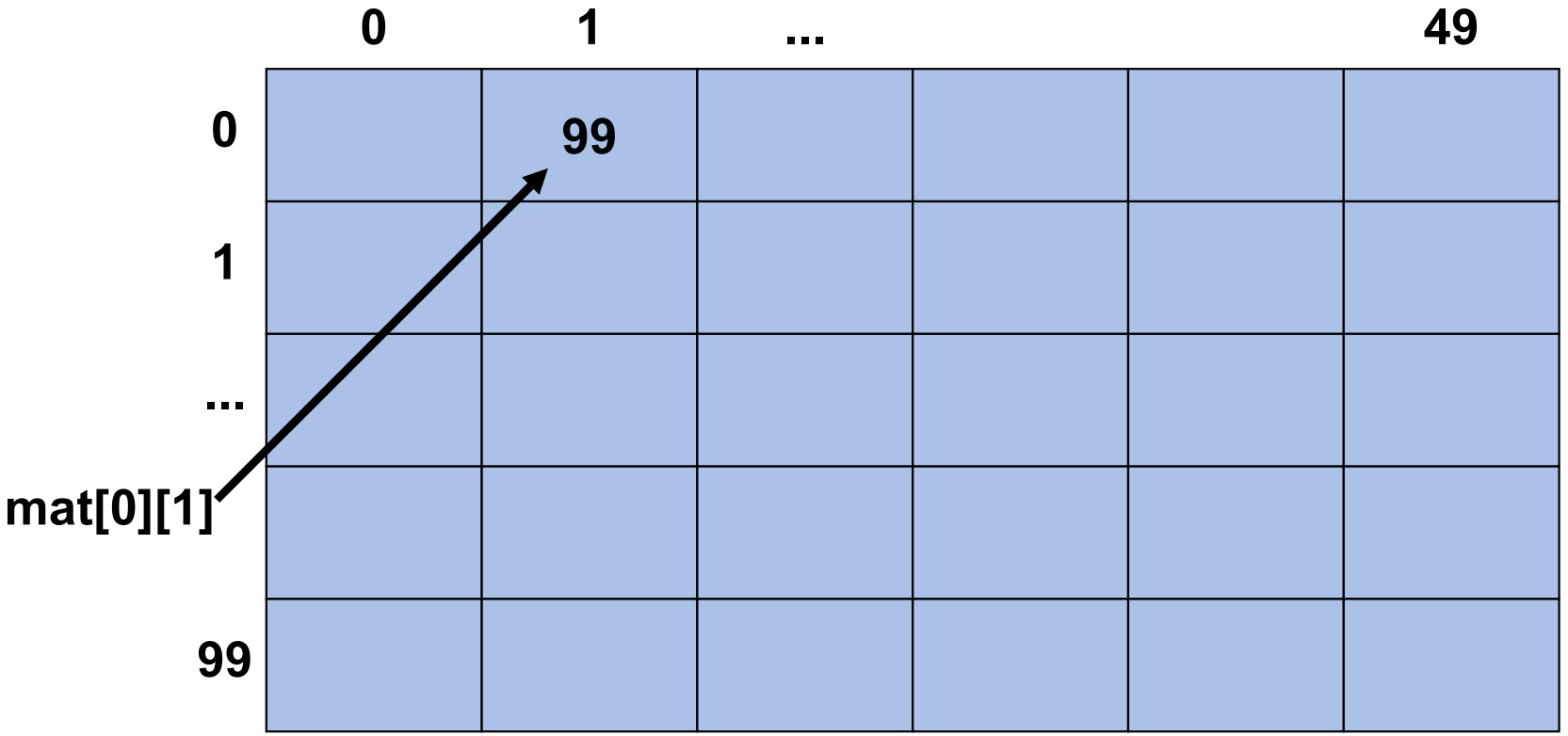
ARRAYS BIDIMENSIONAIS - MATRIZES

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

- Declaração
  - tipo identificador [linhas][colunas];

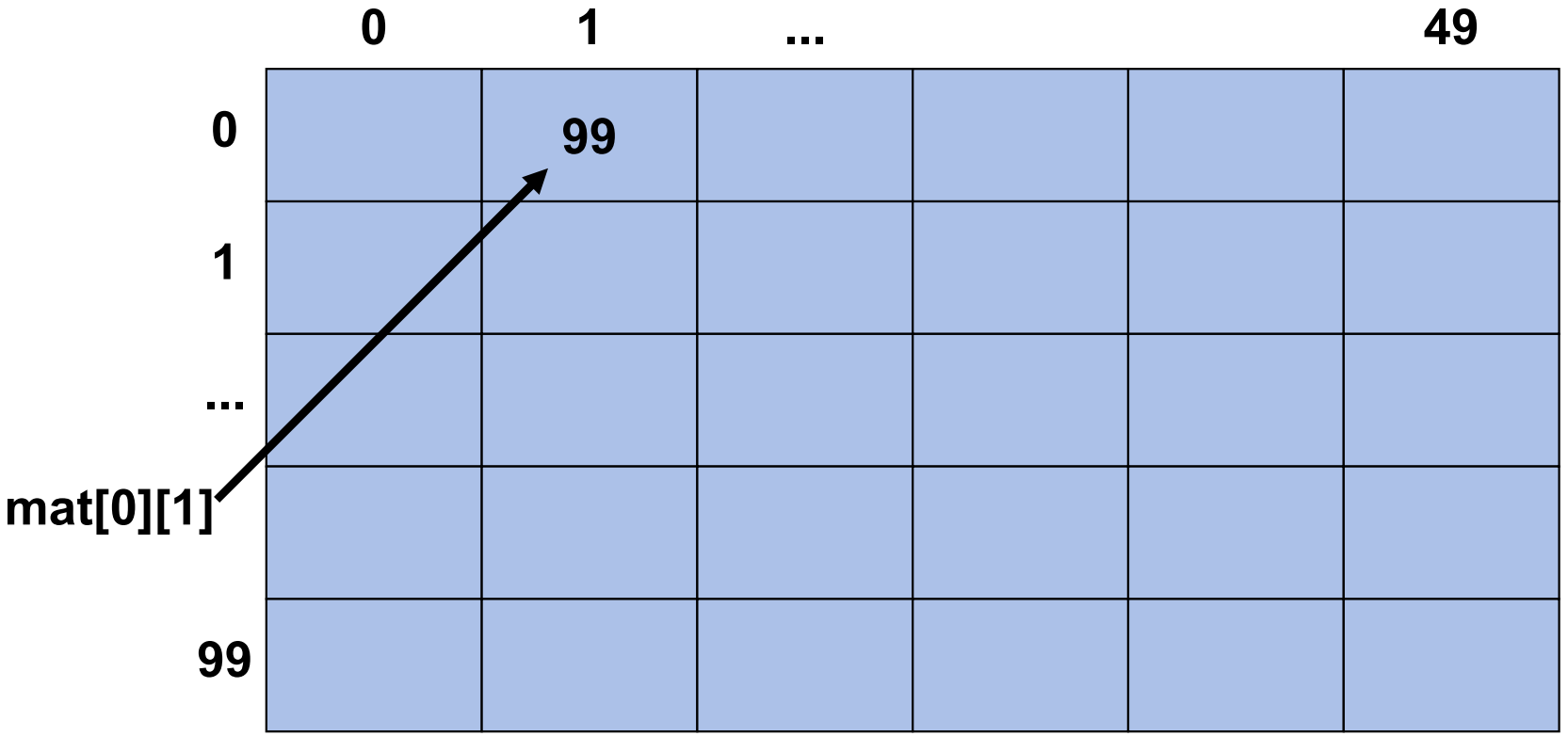
ARRAYS BIDIMENSIONAIS - MATRIZES

- Exemplo
  - Criar uma matriz que tenha 100 linhas por 50 colunas
    - int mat[100][50];
    - mat[0][1] = 99;



ARRAYS BIDIMENSIONAIS - MATRIZES

- Em uma matriz, os elementos são acessados especificando um par de colchetes e índice para cada dimensão da matriz
  - Em linguagem C, a numeração começa em zero



ARRAYS BIDIMENSIONAIS - MATRIZES

- Cada elemento da 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)

ARRAYS BIDIMENSIONAIS - MATRIZES

- Como uma matriz possui dois índices, precisamos de dois comandos de repetição para percorrer todos os seus elementos.

```c
#include <stdio.h>
#include <stdlib.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]);
    }
  }

  return 0;
}
```

Entradas = 10, 20, 4, 5, 7, 10, 11, 8, 50, 60, 50, 23, 89, 100, 1, 0, 34,6, 20,24

```
for (int i = 0; i < 5; i++)
    for(int j = 0; j < 4; j++)
        scanf("%d",&A[i][j]);
```

	0	1	2	3
0	10	20	4	5
1	7	10	11	8
2	50	60	50	23
3	89	100	1	0
4	34	6	20	24

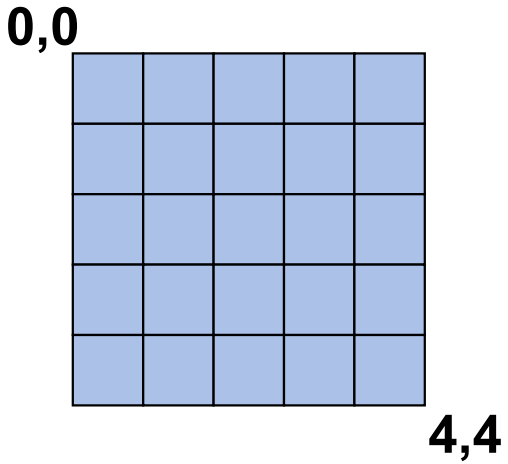
Passo	i	j	A[i,j]
1	0	0	A[0,0] = 10
2	0	1	A[0,1] = 20
3	0	2	A[0,2] = 4
4	0	3	A[0,3] = 5
5	1	0	A[1,0] = 7
6	1	1	A[1,1] = 10
7	1	2	A[1,2] = 11
8	1	3	A[1,3] = 8
9	2	0	A[2,0] = 50
...			
19	4	2	A[4,2] = 20
20	4	3	A[4,3] = 24

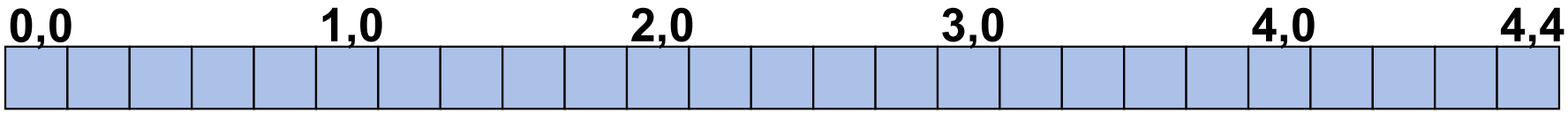
Arrays Multidimensionais

- 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

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];





Exercício

- Leia uma matriz de 3x3 elementos inteiros e calcule a soma dos seus elementos

EXERCÍCIO - SOLUÇÃO

- Leia uma matriz de 3x3 elementos inteiros e calcule a soma dos seus elementos

```c
#include <stdio.h>
#include <stdlib.h>
int main(){
    int mat[3][3];
    int i, j, soma = 0;
    printf("Digite os elementos da matriz\n");
    for(i=0; i < 3; i++)
        for(j=0; j < 3; j++){
            scanf("%d",&mat[i][j]);
        }

    for(i=0; i < 3; i++)
        for(j=0; j < 3; j++)
            soma = soma + mat[i][j];
    printf("Soma = %d\n",soma);

    return 0;
}
```

Exercício

- Dadas duas matrizes reais de dimensão 2x3, fazer um programa para calcular a soma delas.

EXERCÍCIO - SOLUÇÃO

- Dadas duas matrizes reais de dimensão 2x3, fazer um programa para calcular a soma delas.

```c
#include <stdio.h>
#include <stdlib.h>
int main(){
    float A[2][3], B[2][3], S[2][3];
    int i, j;

    //leia as matrizes A e B...

    for(i=0; i < 2; i++)
        for(j=0; j < 3; j++)
            S[i][j] = A[i][j]+ B[i][j];

    return 0;
}
```

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

```c
#include <stdio.h>
#include <stdlib.h>
int main(){
    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}};
    int mat[3][4] = { {1,2}, {3,4,5}, {6,7,8,9} };
    char str1[10] = {'J','o','a','o'};
    char str2[10] = "Joao";

    char nomes[3][10] = {"Joao", "Maria", "Jose"};

    return 0;
}
```

Inicialização sem tamanho

- Inicialização sem especificação de tamanho

  - Nesse tipo de inicialização, o compilador vai considerar o tamanho do dado declarado como sendo o tamanho do array.

  - Isto ocorre durante a compilação e não poderá mais ser mudado durante o programa.

  - Isto é útil quando não queremos contar quantos caracteres serão necessários para inicializarmos uma string ou quantos elementos há em uma lista de valores

INICIALIZAÇÃO SEM TAMANHO

- Inicialização sem especificação de tamanho

```c
#include <stdio.h>
#include <stdlib.h>
int main(){

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

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

    return 0;
}
```

EXERCÍCIOS

- 1) Declare uma matriz 5 x 5. Preencha com 1 a diagonal principal e com 0 os demais elementos. Escreva ao final a matriz obtida.

- 2) Faça um algoritmo para comparar duas matrizes 3x2. Como resultado, deve ser mostrado se as matrizes são iguais ou se são diferentes. Quando diferentes, devem ser mostradas todas as posições em que elas diferem.

- 3) Crie um programa capaz de ler os dados de uma matriz quadrada de inteiros. Ao final da leitura o programa deverá imprimir o número da linha e da coluna que contém o menor dentre todos os números lidos.

EXERCÍCIOS

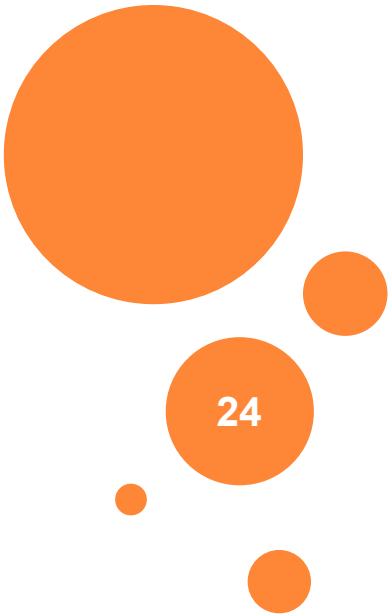
- 4) Crie um programa capaz de criar a transposta de uma matriz. A matriz deve ser lida de teclado.

- 5) Crie um programa que declare uma matriz quadrada. Determine se a matriz é simétrica (uma matriz simétrica é toda a matriz que é igual à sua transposta).

Material Complementar

- Vídeo Aulas
  - Aula 26: Array / Matriz
  - Aula 27: Array Multidimensional

- https://programacaodescomplicada.wordpress.com/indice/linguagem-c/



LINGUAGEM C: ARRAY: VETORES E MATRIZES

Contém slides originais gentilmente disponibilizados pelo Prof. André R. Backes (UFU)