ALGORITMOS E PROGRAMAÇÃO DE COMPUTADORES

Disciplina: 113476

Profa. Carla Denise Castanho

Universidade de Brasília – UnB
Instituto de Ciências Exatas – IE
Departamento de Ciência da Computação – CIC

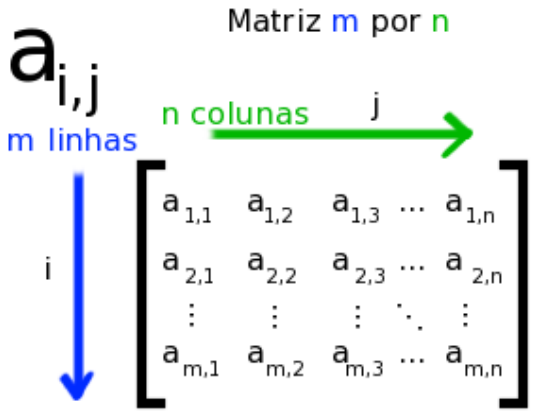
10. MATRIZES

Conjuntos Homogêneos Multidimensionais

- Vimos até aqui vetores como conjuntos homogêneos unidimensionais. Todas as variáveis são do mesmo tipo, e acessamos usando um único índice.
- No entanto, em computação, frequentemente precisamos de vetores **multidimensionais**.
- Por exemplo: para guardar as várias notas de cada aluno de uma turma, seria interessante cada aluno também ser um vetor de notas, *i.e*, precisamos de um **vetor de vetores**.

Matrizes

- Podemos declarar vetores com várias dimensões, tantas quanto forem necessárias.
- No entanto, o caso que usamos com mais frequência é o das **matrizes**, que são vetores **bidimensionais**.
- Uma matriz é uma tabela com ***m* linhas** e ***n* colunas**:



Matrizes

- Declaramos matrizes adicionando uma dimensão ao final do vetor. Veja o exemplo:

Exemplo: Faça um algoritmo que leia uma matriz 5x5 inserida pelo usuário.

```
Algoritmo LeituraMAT
Variáveis
    mat : vetor <pode ser "matriz" também> [5][5] de inteiros
    i, j : inteiro
Início
    Para i ← 0 até 4 faça
        Para j ← 0 até 4 faça
            Escreva ("Informe o elemento ", i + 1, ",", j + 1, " da matriz:")
            Leia (mat[i][j])
        Fim-Para
    Fim-Para
Fim
```

Matrizes

Exemplo: Faça um algoritmo que leia uma matriz MxN inserida pelo usuário, calcula a soma e a média dos elementos.

```
Algoritmo SomaEMedia
Variáveis
    mat : vetor <pode ser "matriz" também> [100][100] de inteiros
    i, j, m, n : inteiro
    soma, media : real
Início
    Leia (m, n)
    Para i ← 0 até m - 1 faça
        Para j ← 0 até n - 1 faça
            Leia (mat[i][j])
        Fim-Para
    Fim-Para
    soma ← 0
    Para i ← 0 até m - 1 faça
        Para j ← 0 até n - 1 faça
            soma ← soma + mat[i][j]
        Fim-Para
    Fim-Para
    media ← soma / (m * n)
    Escreva ("Soma: ", soma, ", Média: ", media, ".")
Fim
```

Matrizes em C

- Como você já deve imaginar, trabalhar com matrizes em C é **muito parecido com vetores**.
- Os índices continuam indo de *zero* até *tamanho-1*, e você só vai adicionar uma nova dimensão (**mais um par de colchetes**).
- Vamos ver um exemplo...

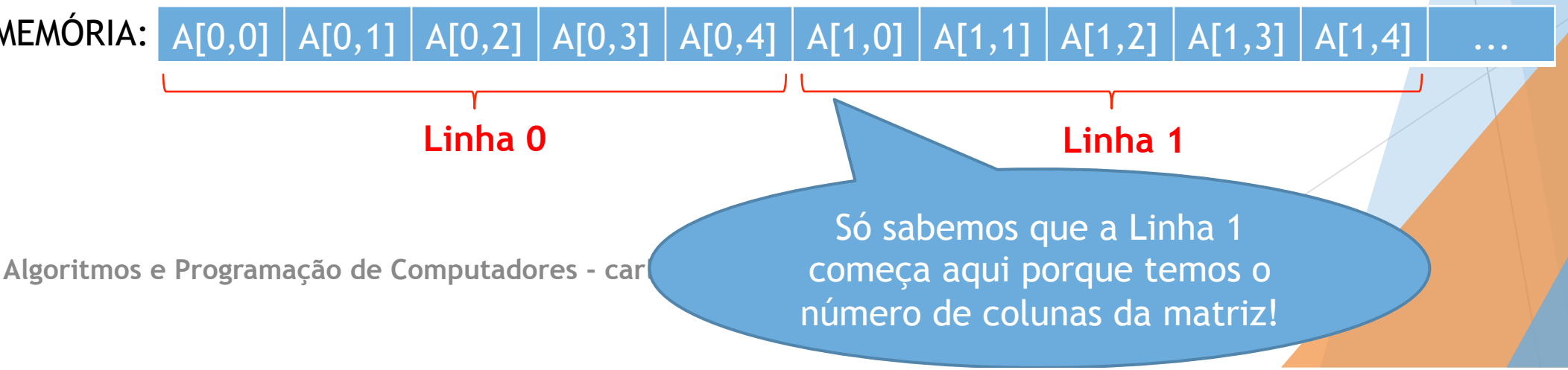
Matrizes em C

Exemplo - Lê e imprime uma matriz na tela.

```c
#include <stdio.h>

int main() {
    int i, j, mat[5][5];

    for (i = 0; i < 5; i++) {
        for (j = 0; j < 5; j++) {
            scanf("%d", &mat[i][j]);
        }
    }

    for (i = 0; i < 5; i++) {
        for (j = 0; j < 5; j++) {
            /* "\t" insere uma marca de tabulação na saída */
            printf("%d\t", mat[i][j]);
        }
        /* ao final de cada linha da matriz, precisamos quebrar uma linha na saída */
        printf("\n");
    }

    return 0;
}
```

Matrizes como Parâmetros de Funções

- Matrizes são passadas para funções da mesma maneira que vetores: um **ponteiro para o primeiro elemento**, no entanto, precisamos tomar cuidado na hora de declará-las como parâmetros de funções.
- A matriz, assim como um vetor, é armazenada em um **bloco contínuo de memória**.
- A função não precisa saber quantos elementos **ao todo** a matriz tem, mas, se não souber o número de **colunas**, como poderá saber onde começa cada linha? Observe o exemplo abaixo para uma matriz de *n* linhas e 5 colunas:



Matrizes como Parâmetros de Funções

- Por isso, na Linguagem C, é necessários **declarar** o parâmetro com o número de colunas alocado (ou seja, do momento que você **declarou a matriz original**). No entanto, você pode utilizar menos linhas e colunas que o declarado:

Exemplo - Declarando uma matriz como parâmetro de função.

```c
#include <stdio.h>

int soma(int mat[][100], int linhas, int colunas) {
    int i, j, s;
    for (i = 0; i < linhas; i++) {
        for (j = 0; j < colunas; j++) {
            s += mat[i][j];
        }
    }
    return s;
}

int main() {
    int i, j, m, n, mat[100][100];
    printf("Informe as dimensões da matriz (linhas, colunas):\n");
    scanf("%d %d", &m, &n);
    for (i = 0; i < m; i++) {
        for (j = 0; j < n; j++) {
            printf("[%d, %d]: ", i + 1, j + 1);
            scanf("%d", &mat[i][j]);
        }
    }
    printf("Soma: %d\n", soma(mat, m, n));
    return 0;
}
```

Matrizes como Parâmetros de Funções

- Por isso, na Linguagem C, é necessários **declarar** o parâmetro com o número de colunas alocado (ou seja, do momento que você **declarou a matriz original**). No entanto, você pode utilizar menos linhas e colunas que o declarado:

Exemplo - Declarando uma matriz como parâmetro de função.

```c
#include <stdio.h>

int soma(int mat[][100], int linhas, int colunas) {
    int i, j, s;
    for (i = 0; i < linhas; i++)
        for (j = 0; j < colunas; j++) {
            s += mat[i][j];
        }
    }
    return s;
}

int main() {
    int i, j, m, n, mat[100][100];
    printf("Informe as dimensões da matriz (linhas, colunas):\n");
    scanf("%d %d", &m, &n);
    for (i = 0; i < m; i++) {
        for (j = 0; j < n; j++) {
            printf("[%d, %d]: ", i + 1, j + 1);
            scanf("%d", &mat[i][j]);
        }
    }
    printf("Soma: %d\n", soma(mat, m, n));
    return 0;
}
```

Note como **declaramos** o mesmo tamanho **declarado na main**. No entanto, **passamos** o número de linhas e colunas **efetivamente utilizados** (informados pelo usuário).

Matrizes como Vetores

- Tudo bem, mas e se eu não quiser que minha função precise saber o número de colunas? Isso faz com que **uma função precise saber detalhes de outra**. E a modularização como fica?
- Na disciplina “Estrutura de Dados”, você vai aprender a **alocar dinamicamente** matrizes (e vetores). Por enquanto, existe um jeito bem mais simples.
- Vamos fazer exatamente o que Linguagem C faz “por debaixo dos panos” para você: **representar uma matriz como um vetor**.

Matrizes como Vetores

- Nós já vimos que a matriz é na verdade um **bloco** contínuo de memória. **Uma linha logo após a outra**.
- Dado o índice da linha *i*, como sabemos em que posição de memória começa essa linha? Simples! Todas as linhas têm o mesmo número de posições. Esse valor é dado pelo número de colunas:

  < *início da linha i* > = i * < *nro de colunas* >

- Lembrando que o i começa com *zero*. A Linha 0 começa sempre na posição 0.
- E agora, dado que eu sei onde começa a *i-ésima* linha, como pegar o *j-ésimo* elemento (ou seja, o elemento na Coluna *j*)? Simples, basta somar j à posição inicial da linha:

  < *elemento [i][j]* > = i * <*nro de colunas*> + j

Matrizes como Parâmetros de Funções

Exemplo - Declarando uma matriz como um vetor no parâmetro de função.

```c
#include <stdio.h>

int soma(int mat[], int linhas, int colunas) {
    int i, j, s;
    for (i = 0; i < linhas; i++) {
        for (j = 0; j < colunas; j++) {
            s += mat[i * colunas + j];
        }
    }
    return s;
}

int main() {
    int i, j, m, n, mat[100 * 100];
    printf("Informe as dimensões da matriz (linhas, colunas):\n");
    scanf("%d %d", &m, &n);
    for (i = 0; i < m; i++) {
        for (j = 0; j < n; j++) {
            printf("[%d, %d]: ", i + 1, j + 1);
            scanf("%d", &mat[i * n + j]);
        }
    }
    printf("Soma: %d\n", soma(mat, m, n));
    return 0;
}
```

Matrizes - Exercícios

1. Escrever um algoritmo que possui 3 funções:
   - uma função para ler uma matriz 4x4 de inteiros;
   - outra função para multiplicar a diagonal principal da matriz por um valor K (K deve ser lido no programa principal e passado para a função juntamente com o ponteiro para a matriz);
   - e uma última função para mostar a matriz resultante na tela.

Obs: as 3 funções devem ser chamadas no programa principal.