

# MC102 – Algoritmos e Programação de Computadores

Instituto de Computação

UNICAMP

Primeiro Semestre de 2014

# Roteiro

- 1 Laços aninhados
- 2 Matrizes
- 3 Exemplos com matrizes
- 4 Matrizes de caracteres
- 5 Inicialização de matrizes e vetores
- 6 Representação de matrizes por linearização de índices
- 7 Exercícios

## Laços aninhados

- Em muitas situações, torna-se necessário implementar um laço dentro de outro laço.
- Estes comandos são conhecidos como laços aninhados.
- Exemplo de laços aninhados em um trecho de programa:

```
int i, j;
```

```
for (i = 1; i <= 10; i++)  
    for (j = 1; j <= 5; j++)  
        printf("i:%d j:%d\n", i, j);
```

# Laços aninhados

- Já sabemos testar se um determinado número é ou não primo.
- Como podemos imprimir os  $n$  primeiros números primos?

## Laços aninhados

- O trecho de programa abaixo verifica se o valor da variável “candidato” corresponde a um número primo:

```
divisor = 2;
primo = 1;
while ((divisor <= candidato/2) && (primo)) {
    if (candidato % divisor == 0)
        primo = 0;
    divisor++;
}
if (primo)
    printf("%d, ", candidato);
```

# Laços aninhados

```
#include <stdio.h>
int main() {
    int divisor, candidato, primosEncontrados = 0, n, primo;
    printf("Digite o numero de primos a serem calculados: ");
    scanf("%d",&n);

    candidato = 2;
    while (primosEncontrados < n) {

        /* inserir aqui trecho do codigo anterior
           que verifica se candidato eh primo */

        if (primo) {
            printf("%d, ", candidato);
            primosEncontrados++;
        }
        candidato++; /* testa proximo numero candidato */
    }
    return 0;
}
```

# Laços aninhados

```
#include <stdio.h>
int main() {
    int divisor, candidato, primosEncontrados = 0, n, primo;
    printf("Digite o numero de primos a serem calculados: ");
    scanf("%d",&n);

    candidato = 2;
    while (primosEncontrados < n) {
        divisor = 2;
        primo = 1;
        while ((divisor <= candidato / 2) && (primo)) {
            if (candidato % divisor == 0)
                primo = 0;
            divisor++;
        }
        if (primo) {
            printf("%d, ", candidato);
            primosEncontrados++;
        }
        candidato++; /* testa proximo numero candidato */
    }
    return 0;
}
```

## Laços aninhados

- Note que o número 2 é o único número par que é primo.
- Podemos alterar o programa para imprimir o número 2:

```
#include <stdio.h>
```

```
int main() {  
    int divisor, candidato, primosEncontrados = 1, n, primo;  
  
    printf("Digite o numero de primos a serem calculados: ");  
    scanf("%d",&n);  
  
    if (n > 0) {  
        printf("%d, ", 2);  
        ...  
    }  
}
```

## Laços aninhados

- E depois os primos ímpares:

```
...
candidato = 3;

while (primosEncontrados < n) {
    divisor = 2;
    primo = 1;
    while ((divisor <= candidato / 2) && (primo)) {
        if (candidato % divisor == 0)
            primo = 0;
        divisor++;
    }
    if (primo) {
        printf("%d, ", candidato);
        primosEncontrados++;
    }
    candidato = candidato + 2; /* testa proximo numero candidato */
}
}
```

# Laços aninhados

```
#include <stdio.h>
int main() {
    int divisor, candidato, primosEncontrados = 1, n, primo;
    printf("Digite o numero de primos a serem calculados: ");
    scanf("%d",&n);
    if (n > 0) {
        printf("%d, ", 2);
        candidato = 3;
        while (primosEncontrados < n) {
            divisor = 2; primo = 1;
            while ((divisor <= candidato / 2) && (primo)) {
                if (candidato % divisor == 0) primo = 0;
                divisor++;
            }
            if (primo) {
                printf("%d, ", candidato);
                primosEncontrados++;
            }
            candidato = candidato + 2; /* testa proximo numero candidato */
        }
    }
    return 0;
}
```

## Laços aninhados

- Suponha que queremos imprimir todas as possibilidades de resultados ao se jogar 4 dados de 6 faces.
- Para cada possibilidade do primeiro dado, devemos imprimir todas as possibilidades dos 3 dados restantes.
- Para cada possibilidade do primeiro e do segundo dado, devemos imprimir todas as possibilidades dos 2 dados restantes...
- Como poderíamos propor uma solução com laços aninhados?

## Laços aninhados

```
#include <stdio.h>

int main() {
    int d1, d2, d3, d4;

    printf("D1 D2 D3 D4\n");

    for (d1 = 1; d1 <= 6; d1++)
        for (d2 = 1; d2 <= 6; d2++)
            for (d3 = 1; d3 <= 6; d3++)
                for (d4 = 1; d4 <= 6; d4++)
                    printf("%d %d %d %d\n", d1, d2, d3, d4);

    return 0;
}
```

# Matrizes

Suponha que queremos ler as notas de 4 provas para cada aluno e então calcular a média do aluno e a média da turma, sendo o tamanho máximo da turma de 50 alunos.

## Solução:

Criar 4 vetores de tamanho 50 cada. Cada vetor representa as notas dos alunos de uma prova.

```
float prova1[50], prova2[50], prova3[50], prova4[50];
```

# Matrizes

- Agora suponha que o número de provas possa ser igual a 100. Torna-se inconveniente criar 100 vetores diferentes, um para cada prova.
- Para resolver esse problema, podemos utilizar matrizes. Uma matriz é um vetor (ou seja, um conjunto de variáveis de mesmo tipo) que possui duas ou mais dimensões.

## Declarando uma matriz bidimensional

```
<tipo> nome_da_matriz [<linhas>][<colunas>;
```

- Uma matriz bidimensional possui *linhas*  $\times$  *colunas* variáveis do tipo `<tipo>`.
- As linhas são numeradas de 0 a *linhas* - 1.
- As colunas são numeradas de 0 a *colunas* - 1.

## Exemplo de declaração de uma matriz bidimensional

```
int matriz[5][4];
```

	0	1	2	3
0				
1				
2				
3				
4				

## Acesso aos elementos de uma matriz bidimensional

- O acesso a um elemento da matriz bidimensional pode ser feito da seguinte forma:

```
nome_da_matriz [<linha>][<coluna>]
```

- Exemplo: `matriz[1][3]`: refere-se ao elemento na 2ª linha e na 4ª coluna da matriz.
- Lembre-se que, assim como vetores, a primeira posição em uma determinada dimensão começa no índice 0.
- O compilador não verifica se você utiliza valores válidos para a linha ou para a coluna.

## Declarando uma matriz de múltiplas dimensões

```
<tipo> nome_da_matriz [< dim1 >] [< dim2 >] ... [< dimN >];
```

- Essa matriz possui  $dim_1 \times dim_2 \times \dots \times dim_N$  variáveis do tipo `<tipo>`.
- Cada dimensão é numerada de 0 a  $dim_i - 1$ .

## Declarando uma matriz de múltiplas dimensões

- Você pode criar, por exemplo, uma matriz para armazenar a quantidade de chuva em um dado dia, mês e ano:

```
double chuva[31][12][3000];
```

```
chuva[23][3][1979] = 6.0;
```

- O código acima indica que no dia 24/04/1980 choveu 6mm.

## Exemplos com matrizes

Lendo uma matriz  $5 \times 4$  a partir da entrada padrão:

```
/* Leitura */
for (i = 0; i < 5; i++)
    for (j = 0; j < 4; j++) {
        printf("Matriz[%d][%d]: ", i, j);
        scanf("%d", &matriz[i][j]);
    }
```

## Exemplos com matrizes

Escrevendo uma matriz  $5 \times 4$  na saída padrão:

```
/* Escrita */  
for (i = 0; i < 5; i++) {  
    for (j = 0; j < 4; j++)  
        printf("%d ", matriz[i][j]);  
    printf("\n");  
}
```

# Exemplos com matrizes

- Ler duas matrizes  $4 \times 3$  e calcular a soma das duas usando uma terceira matriz.

# Exemplos com matrizes

```
#include <stdio.h>

#define LINHAS 4
#define COLUNAS 3

int main() {
    double mat1[LINHAS][COLUNAS], mat2[LINHAS][COLUNAS], mat3[LINHAS][COLUNAS];
    int i, j;

    printf("*** Dados da Matriz 1 ***\n");
    for (i = 0; i < LINHAS; i++)
        for (j = 0; j < COLUNAS; j++) {
            printf("Entre com dado da linha %d e coluna %d: ", i, j);
            scanf("%lf", &mat1[i][j]);
        }

    printf("*** Dados da Matriz 2 ***\n");
    for (i = 0; i < LINHAS; i++)
        for (j = 0; j < COLUNAS; j++) {
            printf("Entre com dado da linha %d e coluna %d: ", i, j);
            scanf("%lf", &mat2[i][j]);
        }
    ...
}
```

# Exemplos com matrizes

...

```
for (i = 0; i < LINHAS; i++)
    for (j = 0; j < COLUNAS; j++) {
        mat3[i][j] = mat1[i][j] + mat2[i][j];
    }
```

```
printf("*** Dados da Matriz 3 ***\n");
for (i = 0; i < LINHAS; i++) {
    for (j = 0; j < COLUNAS; j++)
        printf("%f, ", mat3[i][j]);
    printf("\n");
}
```

```
return 0;
}
```

# Matrizes de caracteres

- Numa matriz bidimensional podemos considerar cada uma das linhas como um vetor, ou seja, uma matriz bidimensional é um vetor de vetores.
- Então, podemos considerar uma matriz bidimensional de caracteres (`char`) como um vetor de strings.
- Sendo assim, podemos, por exemplo, ler ou escrever uma linha inteira de uma matriz bidimensional de caracteres com os comandos `scanf` ou `printf`, respectivamente.
- Neste caso, é importante lembrar que o caractere `'\0'` deve ser adequadamente armazenado na matriz.
- Considere o seguinte exemplo:
  - ▶ Ler, armazenar e imprimir uma lista de palavras. Para cada palavra, também imprimir o seu tamanho.

# Lista de palavras

```
#include <stdio.h>

#define NUM_MAX 50
#define TAM_MAX 20

int main() {
    char palavras[NUM_MAX][TAM_MAX + 1];
    int i, tamanho, n;

    do {
        printf("Entre com o numero de palavras: ");
        scanf("%d", &n);
    } while ((n < 2) || (n > NUM_MAX));

    ...
}
```

## Lista de palavras

...

```
printf("Entre com as %d palavras:\n", n);
for (i = 0; i < n; i++)
    scanf("%s", palavras[i]);

printf("Lista de palavras fornecidas:\n");
for (i = 0; i < n; i++) {
    tamanho = 0;
    while (palavras[i][tamanho] != '\0')
        tamanho++;

    printf("%s (tamanho: %d)\n", palavras[i], tamanho);
}

return 0;
}
```

## Inicialização de matrizes

- Em algumas situações, ao criarmos uma matriz, pode ser útil atribuir valores já na sua criação.
- No caso de vetores, a inicialização é simples: basta atribuir uma lista de valores constantes de mesmo tipo separados por vírgulas e entre chaves.

### Exemplo

```
int vet[5] = {10, 20, 30, 40, 50};
```

- No caso de strings, pode-se atribuir diretamente uma constante.

### Exemplo

```
char st1[100] = "sim, isto eh possivel";
```

# Inicialização de matrizes

- No caso de matrizes bidimensionais, usa-se chaves para delimitar as linhas:

## Exemplo

```
int vet[2][4] = { {10, 20, 30, 40}, {50, 60, 70, 80} };
```

- No caso tridimensional, cada um dos índices da primeira dimensão é uma matriz inteira:

## Exemplo

```
int v3[2][3][4] = {  
    { {1, 2, 3, 4}, {5, 6, 7, 8}, {9, 10, 11, 12} },  
    { {0, 0, 0, 0}, {5, 6, 7, 8}, {0, 0, 0, 0} }  
};
```

# Inicialização de matrizes

```
#include <stdio.h>

int main() {
    int i, j, k;

    int v1[5] = {1, 2, 3, 4, 5};

    int v2[2][3] = { {1, 2, 3}, {4, 5, 6} };

    int v3[2][3][4] = {
        { {1, 2, 3, 4}, {5, 6, 7, 8}, {9, 10, 11, 12} },
        { {0, 0, 0, 0}, {5, 6, 7, 8}, {0, 0, 0, 0} }
    };

    char st1[100] = "exemplo de texto";

    ...
}
```

# Inicialização de matrizes

...

```
printf("Vetor v1:\n");  
for (i = 0; i < 5; i++)  
    printf("%d, ", v1[i]);
```

```
printf("Matriz bidimensional v2:\n");  
for (i = 0; i < 2; i++) {  
    for (j = 0; j < 3; j++)  
        printf("%d, ", v2[i][j]);  
    printf("\n");  
}
```

...

# Inicialização de matrizes

...

```
printf("Matriz tridimensional v3:\n");
for (i = 0; i < 2; i++) {
    for (j = 0; j < 3; j++) {
        for (k = 0; k < 4; k++) {
            printf("%d, ", v3[i][j][k]);
        }
        printf("\n");
    }
    printf("\n");
}
```

```
printf("String st1:\n");
printf("%s", st1);
```

```
return 0;
```

```
}
```

# Linearização de índices

- Podemos usar sempre vetores simples para representar matrizes (na prática o compilador faz isto por você).
- Ao declarar uma matriz como `int mat[3][4]`, sabemos que serão alocados 12 posições de memória associadas com a variável `mat`.
- Poderíamos simplesmente criar `int mat[12]`. Entretanto, perdemos a simplicidade de uso dos índices em forma de matriz.
  - ▶ Você não mais poderia escrever `mat[1][3]`, por exemplo.

# Linearização de índices

- A *linearização de índices* é justamente a representação de matrizes usando-se um vetor simples.
- Entretanto, devemos ter um padrão para acessar as posições deste vetor como se sua organização fosse na forma de matriz.

# Linearização de índices

- Considere o exemplo:

```
int mat[12]; /* ao inves de int mat[3][4] */
```

- Podemos fazer a divisão por linhas da seguinte forma:
  - ▶ Primeira linha: mat[0] até mat[3]
  - ▶ Segunda linha: mat[4] até mat[7]
  - ▶ Terceira linha: mat[8] até mat[11]
- Para acessar uma posição [i][j], podemos usar:
  - ▶ mat[i\*4 + j];  
tal que  $0 \leq i \leq 2$  e  $0 \leq j \leq 3$ .

# Linearização de índices

- De forma geral, seja matriz  $\text{mat}[n*m]$ , representando  $\text{mat}[n][m]$ .
- Para ter acesso à posição correspondente a  $[i][j]$  usamos:
  - ▶  $\text{mat}[i*m + j]$ ;  
tal que  $0 \leq i \leq n - 1$  e  $0 \leq j \leq m - 1$ .
- Note que  $i$  salta blocos de tamanho  $m$  e  $j$  indexa a posição dentro de um bloco.

# Linearização de índices

- Podemos estender para mais dimensões. Seja matriz  $\text{mat}[n*m*q]$ , representando  $\text{mat}[n][m][q]$ .
  - ▶ As posições de 0 até  $(m * q) - 1$  são da primeira matriz.
  - ▶ As posições de  $(m * q)$  até  $(2 * m * q) - 1$  são da segunda matriz.
  - ▶ Assim por diante...
- De forma geral, seja matriz  $\text{mat}[n*m*q]$ , representando  $\text{mat}[n][m][q]$ .
- Para ter acesso à posição correspondente a  $[i][j][k]$ , usamos:
  - ▶  $\text{mat}[i*m*q + j*q + k]$ ;

## Linearização de índices

```
#include <sstdio.h>

int main() {
    int mat[40]; /* representando mat[5][8] */
    int i,j;

    for (i = 0; i < 5; i++)
        for (j = 0; j < 8; j++)
            mat[i*8 + j] = i*j;

    for (i = 0; i < 5; i++) {
        for (j = 0; j < 8; j++)
            printf("%d, ",mat[i*8 + j]);
        printf("\n");
    }

    return 0;
}
```

## Exercícios

Escreva um programa que leia todas as posições de uma matriz  $n \times m$ , em que  $n$  e  $m$  são inteiros dados. Em seguida, para cada posição não nula da matriz, mostre seus índices (de linha e de coluna) e seus valores. No final, mostre o número de posições não nulas da matriz. Exemplo de saída:

```
Matriz[0][3] = 4
```

```
Matriz[0][9] = -12
```

```
Matriz[2][3] = 2
```

```
...
```

```
Matriz[9][8] = 7
```

```
34 posicoes nao nulas
```

## Exercícios

Escreva um programa que leia todos os elementos de uma matriz  $n \times m$  e imprima a matriz e a sua transposta. Exemplo:

Matriz

$$\begin{bmatrix} 11 & 12 & 13 & 14 \\ 21 & 22 & 23 & 24 \\ 31 & 32 & 33 & 34 \\ 41 & 42 & 43 & 44 \\ 51 & 52 & 53 & 54 \end{bmatrix}$$

Transposta

$$\begin{bmatrix} 11 & 21 & 31 & 41 & 51 \\ 12 & 22 & 32 & 42 & 52 \\ 13 & 23 & 33 & 43 & 53 \\ 14 & 24 & 34 & 44 & 54 \end{bmatrix}$$

## Exercícios

Escreva um programa que leia 2 matrizes quadradas ( $n \times n$ ) e imprima as matrizes e a soma delas. Exemplo:

$$\begin{array}{c} \text{A} \\ \left[ \begin{array}{ccccc} 0 & 1 & 0 & 2 & 3 \\ 0 & 1 & 0 & 2 & 3 \\ 0 & 1 & 0 & 2 & 3 \\ 0 & 1 & 0 & 2 & 3 \\ 0 & 1 & 0 & 2 & 3 \end{array} \right] \end{array} + \begin{array}{c} \text{B} \\ \left[ \begin{array}{ccccc} 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 2 & 2 & 2 & 2 & 2 \\ 3 & 3 & 3 & 3 & 3 \end{array} \right] \end{array} = \begin{array}{c} \text{C} \\ \left[ \begin{array}{ccccc} 0 & 1 & 0 & 4 & 3 \\ 1 & 2 & 1 & 3 & 4 \\ 0 & 1 & 0 & 2 & 3 \\ 2 & 3 & 2 & 4 & 5 \\ 3 & 4 & 3 & 5 & 6 \end{array} \right] \end{array}$$

## Exercícios

Escreva um programa que leia 2 matrizes quadradas ( $n \times n$ ) e imprima as matrizes e o produto delas. Exemplo:

$$\begin{array}{c} \text{A} \\ \left[ \begin{array}{ccccc} 0 & 1 & 0 & 2 & 3 \\ 0 & 1 & 0 & 2 & 3 \\ 0 & 1 & 0 & 2 & 3 \\ 0 & 1 & 0 & 2 & 3 \\ 0 & 1 & 0 & 2 & 3 \end{array} \right] \end{array} \times \begin{array}{c} \text{B} \\ \left[ \begin{array}{ccccc} 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 2 & 2 & 2 & 2 & 2 \\ 3 & 3 & 3 & 3 & 3 \end{array} \right] \end{array} = \begin{array}{c} \text{C} \\ \left[ \begin{array}{ccccc} 14 & 14 & 14 & 14 & 14 \\ 14 & 14 & 14 & 14 & 14 \\ 14 & 14 & 14 & 14 & 14 \\ 14 & 14 & 14 & 14 & 14 \\ 14 & 14 & 14 & 14 & 14 \end{array} \right] \end{array}$$

## Exercícios

Escreva um programa leia uma matriz quadrada  $n \times n$  e verifique se ela é uma matriz triangular inferior. Exemplo:

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 5 & 0 & 0 & 0 \\ 0 & 4 & 4 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 9 & 0 & 2 & 3 \end{bmatrix}$$

## Exercícios

Escreva um programa leia uma matriz quadrada  $n \times n$  e verifique se ela é uma matriz triangular superior. Exemplo:

$$\begin{bmatrix} 1 & 0 & 8 & 9 & 8 \\ 0 & 5 & 0 & 0 & 0 \\ 0 & 0 & 4 & 0 & 7 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 3 \end{bmatrix}$$

## Exercícios

Escreva um programa leia uma matriz quadrada  $n \times n$  e verifique se ela é uma matriz diagonal. Exemplo:

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 5 & 0 & 0 & 0 \\ 0 & 0 & 4 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 3 \end{bmatrix}$$

## Exercícios

Escreva um programa leia uma matriz quadrada  $n \times n$  e verifique se ela é uma matriz triangular inferior, triangular superior ou diagonal. Exemplo:

Triangular Inferior

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 5 & 0 & 0 & 0 \\ 0 & 4 & 4 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 9 & 0 & 2 & 3 \end{bmatrix}$$

Triangular Superior

$$\begin{bmatrix} 1 & 0 & 8 & 9 & 8 \\ 0 & 5 & 0 & 0 & 0 \\ 0 & 0 & 4 & 0 & 7 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 3 \end{bmatrix}$$

Diagonal

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 5 & 0 & 0 & 0 \\ 0 & 0 & 4 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 3 \end{bmatrix}$$

# Matriz triangular inferior, triangular superior ou diagonal

```
#include <stdio.h>

#define MAX 10

int main() {
    int matriz[MAX][MAX];
    int i, j, n, inferior = 1, superior = 1;

    scanf("%d", &n);

    for (i = 0; i < n; i++)
        for (j = 0; j < n; j++)
            scanf("%d", &matriz[i][j]);

    ...
}
```

## Matriz triangular inferior, triangular superior ou diagonal

```
for (i = 0; i < n; i++)
    for (j = 0; j < n; j++)
        if ((matriz[i][j]) && (i != j)) {
            if (i > j)
                superior = 0;
            else
                inferior = 0;
        }

if (superior && inferior)
    printf("Matriz diagonal\n");
else if (superior)
    printf("Matriz triangular superior\n");
else if (inferior)
    printf("Matriz triangular inferior\n");

return 0;
}
```

## Matriz triangular inferior, triangular superior ou diagonal

```
for (i = 0; (i < n) && (inferior || superior); i++)
    for (j = 0; (j < n) && (inferior || superior); j++)
        if ((matriz[i][j]) && (i != j)) {
            if (i > j)
                superior = 0;
            else
                inferior = 0;
        }

if (superior && inferior)
    printf("Matriz diagonal\n");
else if (superior)
    printf("Matriz triangular superior\n");
else if (inferior)
    printf("Matriz triangular inferior\n");

return 0;
}
```