

Exercícios 07 :: Matrizes

Instruções Gerais

- Faça todos os exercícios em um único arquivo .c. Utilize a função main() para fazer chamadas de testes às funções solicitadas pelos exercícios.
- Utilize a extensão .c, o compilador gcc e o editor de sua preferência: VS Code, Dev C++, etc.
 - Alternativamente, utilize <https://replit.com/languages/c>.
- Você pode utilizar as seguintes funções disponíveis na biblioteca padrão de C:
 - rand() <stdlib.h>; printf() e scanf() <stdio.h>; strlen(), strcpy(), strcmp(), strcat() <string.h>

1. Escreva uma função que imprime o conteúdo de uma matriz.

```
void print_matrix(int rows, int cols, int m[rows][cols])
```

```
Ex:      int mat[3][3] = { // 3 linhas x 3 colunas
                {1, 2, 3},
                {4, 5, 6},
                {7, 8, 9}
        };
        print_matrix(3, 3, mat); // imprime do 1 ao 9 na matriz exemplo
```

2. Escreva uma função que imprime o conteúdo de uma matriz ao contrário, isto é, do último elemento para o primeiro. Considerando a matriz do exemplo anterior, a função imprimiria do 9 ao 1.

```
void print_reversed(int rows, int cols, int m[rows][cols])
```

3. Escreva uma função que encontra e imprime o maior e o menor valores contidos em uma matriz.

```
void print_min_max(int rows, int cols, int m[rows][cols])
```

4. Escreva uma função que retorna a média aritmética simples de todos os elementos da matriz.

```
float average(int rows, int cols, int m[rows][cols])
```

5. Escreva uma função que inicia uma matriz com valores inteiros, iniciando em **start** e progredindo com **step**.

```
void init_values(int rows, int cols, int m[rows][cols], int start, int step)
```

```
Ex:      int v[4][6];
        init_values(4, 6, v, 10, 2); // start=10, step=2

        // 10,12,14,16,18,20    << Matriz após chamada da função:
        // 22,24,26,28,30,32
        // 34,36,38,40,42,44
        // 46,48,50,52,54,56
```

6. Escreva uma função que inicia uma matriz com valores aleatórios, sorteados entre **min** e **max**, isto é [min..max]. Utilize a função rand() da biblioteca <stdlib.h> para obter os valores:

```
void init_random(int rows, int cols, int m[rows][cols], int min, int max)
```

```
Ex:      int v[4][6];
          // matriz deve ser preenchida com valores aleatórios entre 5 e 50
          init_random(4, 6, v, 5, 50); // min=5, max=50
```

7. Escreva uma função que retorna o total da linha de maior soma em uma matriz. A função deve somar todos os elementos em cada linha da matriz, para então encontrar a linha de maior soma.

```
int max_line(int rows, int cols, int m[rows][cols])
```

Exemplo:

```
int mat[5][5] = {
    { 1, 2, 3, 4, 5}, // soma: 15
    { 2, 4, 6, 8,10}, // soma: 30 ← esta linha possui a maior soma
    { 1, 2, 3, 4, 7}, // soma: 17
    { 2, 0, 6, 1,10}, // soma: 19
    { 7, 3, 4, 0, 0}  // soma: 14
};
int res = max_line(5, 5, mat);
printf("Maior soma: %d\n", res); // imprime 30
```

8. Escreva uma função que recebe uma matriz e coloca as somas de cada linha na última coluna.

```
void put_totals(int rows, int cols, int m[rows][cols])
```

Exemplo:

```
int mat[4][5] = {
    { 1, 2, 3, 4,  0},
    { 2, 4, 6, 8,  0},
    { 3, 6, 9, 12, 0}, // O último elemento de cada linha receberá a soma
    { 4, 8, 12,16, 0} // da linha.
};
put_totals(4, 5, mat);
```

```
// Após a chamada da função, a matriz ficará desta forma:
// 1, 2, 3, 4, 10
// 2, 4, 6, 8, 20
// 3, 6, 9, 12,30 << O último elemento de cada linha recebeu a soma
// 4, 8, 12,16,40 da linha.
```

9. Escreva uma função que verifica a igualdade entre duas matrizes. Ele deve retornar 1 (true) se as matrizes forem iguais ou 0 (false), caso sejam diferentes. Matrizes iguais possuem os mesmos elementos e as mesmas dimensões.

```
int matrix_equals(int r1, int c1, int m1[r1][c1], int r2, int c2, int m2[r2][c2])
```

10. Escreva uma função que realiza a adição de duas matrizes m1 e m2, colocando o resultado em m3. Considere que todas as matrizes possuem as mesmas dimensões (rows x cols).

```
void sum(int rows, int cols, int m1[rows][cols], int m2[rows][cols],
        int m3[rows][cols])
```

Exemplo para matriz 3x2:

$$\begin{matrix} & m1 & & m2 & & m3 \\ \begin{bmatrix} 1 & 3 \\ 1 & 0 \\ 1 & 2 \end{bmatrix} & + & \begin{bmatrix} 0 & 0 \\ 7 & 5 \\ 2 & 1 \end{bmatrix} & = & \begin{bmatrix} 1+0 & 3+0 \\ 1+7 & 0+5 \\ 1+2 & 2+1 \end{bmatrix} & = & \begin{bmatrix} 1 & 3 \\ 8 & 5 \\ 3 & 3 \end{bmatrix} \end{matrix}$$

11. Escreva uma função que monta a transposta da matriz m1 em m2. Observe que deve haver uma compatibilidade entre as dimensões das matrizes.

```
void transpose(int r1, int c1, int m1[r1][c1], int r2, int c2, int m2[r2][c2]).
```

Exemplo para matriz 2x3:

$$\begin{matrix} & m1 & & m2 \\ \begin{bmatrix} 1 & 2 & 3 \\ 0 & -6 & 7 \end{bmatrix}^T & = & \begin{bmatrix} 1 & 0 \\ 2 & -6 \\ 3 & 7 \end{bmatrix} \end{matrix}$$

12. Escreva uma função que troca as diagonais de uma matriz. Considere que a matriz é quadrada, com dimensões d.

```
void switch_diagonals(int d, int m1[d][d]).
```

Exemplo para matriz 3x3:



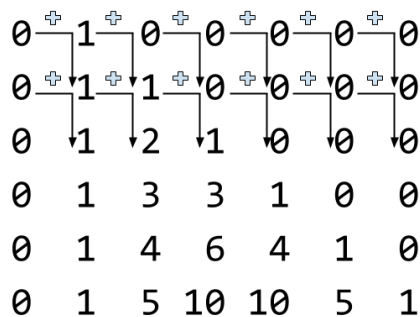
13. Escreva uma função que cria o Triângulo de Pascal em uma matriz e depois o imprime. A função deve receber o número de linhas desejado para o triângulo.

```
void pascal_triangle(int n).
```

DICA: Na função, declare uma matriz com n linhas e $n+1$ colunas (`int m[n][n+1]`). Inicie a primeira linha com zeros, exceto na posição 1. No triângulo de Pascal, cada dois números consecutivos (na mesma linha) devem ser somados, e o resultado deve ser colocado na posição abaixo, isto é, $m[i+1][j+1] = m[i][j] + m[i][j+1]$

$$\begin{matrix} 2_{1,2} & 1_{1,3} \\ & 3_{2,3} \end{matrix} \implies m[1+1][2+1] = m[1][2] + m[1][2+1]$$

Ex: lines = 6



14. Escreva uma função que verifica se uma matriz contém um triângulo superior e devolve 1 (true) ou 0 (false). Considere que a matriz é quadrada, com dimensões d. Dica: nesta matriz, todos os elementos do triângulo esquerdo inferior, nos quais row > col, devem ser iguais a zero.

`int check_upper_triangle(int d, int m[d][d]).`

Exemplo para matrizes 2x2 e 3x3

$$\mathbf{A} \begin{bmatrix} 1 & 2 \\ 0 & 4 \end{bmatrix} \quad \mathbf{B} \begin{bmatrix} 1 & 2 & 3 \\ 0 & 5 & 6 \\ 0 & 0 & 9 \end{bmatrix}$$

15. Escreva uma função que verifica se uma matriz é identidade e devolve 1 (true) ou 0 (false).

`int check_identity(int rows, int cols, int m[rows][cols]).`

Exemplo para matriz 3x3

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

16. Escreva uma função que verifica se uma matriz é simétrica e devolve 1 (true) ou 0 (false). Em uma matriz simétrica os valores são espelhados em relação à diagonal principal. Considere que a matriz é quadrada, com dimensões d.

`int check_symmetric(int d, int m[d][d]).`

Exemplo para matriz 3x3

$$\begin{bmatrix} 1 & 2 & 3 \\ 2 & 4 & 5 \\ 3 & 5 & 8 \end{bmatrix} = \begin{bmatrix} 1 & 2 & 3 \\ 2 & 4 & 5 \\ 3 & 5 & 8 \end{bmatrix}$$

17. Escreva uma função que imprime um array de **n** strings de até **len-1** letras.

```
void print_strings(int n, int len, char list[n][len]).
```

```
Ex:      char v[6][20] = {"John", "Mary", "Jake", "Finn", "Ada", "Michael"};
        print_strings(6, 20, v);
```

18. Escreva uma função que imprime as palavras de maior e menor tamanhos em um array de **n** strings de até **len-1** letras.

```
void print_strings_minmax(int n, int len, char list[n][len]).
```

```
Ex: char v[6][20] = {"Joh", "Mariane", "Jak", "Samuel", "Ada", "Michelangelo"};
    print_strings_minmax(6, 20, v);
```

```
// A função imprimirá:
// Menor: Joh, Jak, Ada
// Maior: Michelangelo
```

19. Escreva uma função que retorna a quantidade de palavras em um array de **n** linhas de texto (strings de até **len-1** letras). Considere que poderá haver mais de um espaço entre cada palavra, bem como, espaços no início ou final de cada string.

```
int count_words(int n, int len, char lines[n][len]).
```

Ex:

```
char v[2][200] = {
    " Computer programming is the process of designing and building an
    executable computer program for accomplishing a specific computing task.",
    " Programming involves tasks such as analysis, generating algorithms,
    profiling algorithms accuracy and resource consumption, and the
    implementation of algorithms in a chosen programming language."
};
int count = count_words(2, 200, v);
printf("Numero de palavras: %d\n", count);
```

20. Escreva uma função que conta e devolve o número de ocorrências de uma palavra **word** em um array de **n** linhas de texto (strings de até **len-1** letras).

```
int count_occurrences(int n, int len, char lines[n][len], char word[]).
```

Ex:

```
char v[5][200] = {
    "computer programming is the process of designing and building an
    executable computer program for accomplishing a specific computing task.",
    "programming involves tasks such as analysis, generating algorithms,
    profiling algorithms accuracy and resource consumption, and the
    implementation of algorithms in a chosen programming language.",
    "the source code of a program is written in one or more programming
    languages.",
```

"the purpose of programming is to find a sequence of instructions that will automate the performance of a task for solving a given problem.",
"the process of programming thus often requires expertise in several different subjects, including knowledge of the application domain, specialized algorithms, and formal logic."

};

int count = count_occurrences(5, 200, v, "programming");

printf("Numero de ocoorrecncias: %d\n", count);