PROFESSOR: EDUARDO C. XAVIER (SALA 10 - IC1)

MC102 – ALGORITMOS E PROGRAMAÇÃO DE COMPUTADORES

**Lista 4**

1. Crie uma **função** que recebe um vetor e seu tamanho por parâmetro além de dois números inteiros $i$ e $j$ que são índices do vetor (ou seja $0 \leq i, j \leq n$). A função deve então trocar os elementos das posições $i$ e $j$ entre si.

2. Faça um programa que lê um vetor de 30 inteiros e guarda o vetor na ordem inversa que foi lido em um outro vetor de saída.

3. Faça uma **função** que recebe um vetor de inteiros e seu tamanho como parâmetros, e ao final da execução da função o vetor esteja invertido. Utilize a seguinte idéia: troque os elementos da posição 0 e 29 entre si, depois da posição 1 e 28 etc. Pense bem no critério de parada.

4. Escreva uma **função** que recebe um vetor de inteiros e seu tamanho como parâmetros, e devolve a soma dos números pares deste vetor.

5. Faça uma **função** que recebe um vetor de números reais e o seu tamanho por parâmetro e devolve o desvio padrão dos números do vetor usando a seguinte fórmula:

$$\sqrt{\frac{1}{n-1}(\sum_{i=1}^{n} x_i^2 - \frac{1}{n}(\sum_{i=1}^{n} x_i)^2)}$$

   onde $n$ é o número de elementos.

6. Modifique cada um dos seguintes algoritmos vistos em sala, para que estes ordenem um vetor em ordem **decrescente** de valor:

   - selectionSort
   - insertionSort
   - BubbleSort

7. Use cada um dos algoritmos abaixo e mostre passo-a-passo, como visto em aula, os passos para ordenar o vector $(31, 41, 59, 26, 41, 58, 15, 19)$:

   - selectionSort
   - insertionSort
   - BubbleSort

8. Crie uma função para determinar o número total de inversões em um vetor **vet**. Uma inversão existe quando um elemento em uma posição $i < j$ é tal que $vet[i] > vet[j]$. Por exemplo, no vetor $(10, 4, 6, 1, 2)$ existem 4 inversões para o número 10, 2 inversões para o número 4, 2 inversões para o número 6, nenhuma inversão para 1, e nenhuma para o 2. Portanto o total de inversões é 8. Modifique o algoritmo *bubbleSort* para computar o número de inversões em um vetor.

9. Faça um programa que leia uma matriz no máximo $30 \times 30$ e imprima a sua transposta.

10. Faça uma **função** que receba como parâmetros uma matriz quadrada no máximo $30 \times 30$, o seu tamanho $n$ e dois inteiros $i, j$ que são índices de linha e coluna respectivamente da matriz. A função deve devolver a soma total dos elementos da linha $i$ com os elementos da coluna $j$ da matriz.

11. Faça uma **função** que receba como parâmetros uma matriz quadrada no máximo $30 \times 30$, o seu tamanho $n$ e dois inteiros $i, j$. A função deve trocar os conteúdos das linhas $i$ e $j$ desta matriz entre si. Esta é uma operação de matrizes conhecida como permutação de linhas.

12. Escreva uma **função** que, dada uma matriz quadrada de dimensão $n$, verifica se esta é simétrica ou não.

13. Uma matriz quadrada de inteiros é um quadrado mágico se a soma dos elementos de cada linha, a soma dos elementos de cada coluna, a soma dos elementos da diagonal principal e da diagonal secundária são todos iguais. A matriz abaixo é um exemplo de quadrado mágico:

$$\begin{matrix} 3 & 4 & 8 \\ 10 & 5 & 0 \\ 2 & 6 & 7 \end{matrix}$$

    Faça um programa que lê uma matriz quadrada e determina se ela é um quadrado mágico.

14. Escreva um programa que leia duas palavras do teclado e determina se a segunda é um anagrama da primeira. Uma palavra é um anagrama de outra se todas as letras de uma ocorrem na outra, em mesmo número, independente da posição. Exemplos: `ROMA`, `MORA`, `ORAM`, `AMOR`, `RAMO` são anagramas entre si.

15. Faça um programa que leia um texto $T$ e uma palavra $p$ do teclado. Em seguida o programa deverá imprimir todas as posições onde ocorrem a palavra $p$ em $T$.

    Se por exemplo $T =$ "duas bananas e 4 abacates. Nao havera mais bananas.", e $p =$"bananas", então o programa deveria imprimir 5 e 43.

16. Escreva um programa que lê uma string de até 50 caracteres, e imprime "Palindromo"caso a string seja um palindromo e "Nao Palindromo"caso contrário. OBS: Um palindromo é uma palavra ou frase, que é igual quando lida da esquerda para a direita ou da direita para a esquerda (assuma que só são usados caracteres minúsculos e sem acentos. Espaços em brancos devem ser descartados). Exemplo de palindromo: saudavel leva duas.

17. Historicamente César foi o primeiro a codificar mensagens. Ele reorganizava o texto de suas mensagens de maneira que o texto parecia não ter sentido. Cada mensagem sempre possuía uma contagem de letras cujo total equivalia a um quadrado perfeito, dependendo de quanto César tivesse que escrever. Assim, uma mensagem com 16 caracteres usava um quadrado de quatro por quatro; se fossem 25 caracteres, seria cinco por cinco; 100 caracteres requeriam um quadrado de dez por dez, etc. Seus oficiais sabiam que deviam transcrever o texto preenchendo as casas do quadrado sempre que uma mensagem aleatória chegasse. Ao fazerem isso, podiam ler a mensagem na vertical e seu sentido se tornaria claro.

Escreva um programa que lê o tamanho de uma string e a string. Depois o programa escreve a mensagem decifrada.

**Exemplo**:

```
36
MEEUMOCSHMSC1T*AGU0A***L2****T*****A
```

Esta mensagem pode ser trancrita em um quadrado perfeito 6x6.

| | | | | | |
|---|---|---|---|---|---|
| M | E | E | U | M | O |
| C | S | H | M | S | C |
| 1 | T | * | A | G | U |
| 0 | A | * | * | * | L |
| 2 | * | * | * | * | T |
| * | * | * | * | * | A |

Lendo cada coluna da matriz (desconsiderando o char '*'), a saída deverá conter:

```
MC102 ESTA EH UMA MSG OCULTA.
```

18. Sudoku é jogado numa malha de 9x9 quadrados, dividida em sub-malhas de 3x3 quadrados, chamada "quadrantes". O objetivo do jogo é preencher os quadrados com números entre 1 e 9 de acordo com as seguintes regras:

    - Cada número pode aparecer apenas uma vez em cada linha.
    - Cada número pode aparacer apenas uma vez em cada coluna.
    - Cada número pode aparecer apenas uma vez em cada quadrante.

    **Exemplo**:

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 9 | 5 | 3 | 4 | 8 | 6 | 2 | 7 | 1 |
| 1 | 2 | 7 | 9 | 3 | 5 | 8 | 4 | 6 |
| 6 | 8 | 4 | 7 | 1 | 2 | 9 | 3 | 5 |
| 5 | 6 | 8 | 3 | 9 | 1 | 4 | 2 | 7 |
| 4 | 9 | 1 | 2 | 6 | 7 | 3 | 5 | 8 |
| 3 | 7 | 2 | 8 | 5 | 4 | 1 | 6 | 9 |
| 7 | 4 | 9 | 5 | 2 | 8 | 6 | 1 | 3 |
| 2 | 3 | 6 | 1 | 7 | 9 | 5 | 8 | 4 |
| 8 | 1 | 5 | 6 | 4 | 3 | 7 | 9 | 2 |

    Escreva um programa que lê um jogo de Sodoku (matriz 9x9, toda preenchida com números de 1 a 9) e verifica se é um jogo válido ou não. Um jogo válido respeita as três regras acima.