Atividades ADNP
Semana 04 :: Vetores

Instruções Gerais

- Faça cada exercício em uma função distinta.
- Utilize a extensão .c e o compilador de gcc.
- Utilize o editor de sua preferência: Code Blocks, VS Code, Dev C++, etc.

IMPORTANTE: Cada exercício deve ser escrito em uma função distinta. No final do programa, deve haver uma única função main() que fará chamadas às funções dos exercícios, para testes.

1. Escreva uma função que recebe um vetor **vet** de tamanho **n** e o imprime em ordem reversa.
   `void printReverse(int vet[], int n)`

2. Escreva uma função que recebe um vetor **vet** de tamanho **n** e imprime apenas os valores pares.
   `void printEven(int vet[], int n)`

3. Escreva uma função que recebe um vetor **vet** de tamanho **n** contendo números inteiros positivos e negativos. A função deve inverter o sinal dos números negativos, passando-os para positivo.
   `void setPositive(int vet[], int n)`

   `Entrada:{1, -5, 67, -45, -1, -1, 0, 48} → Saída:{1, 5, 67, 45, 1, 1, 0, 48}`

4. Escreva uma função que recebe um vetor **vet** de tamanho **n** e devolve a média aritmética simples dos valores contidos.
   `int sumValues(int vet[], int n)`

   `Entrada: {1, 23, 4, 8, 41, 7, 3} → Saída: 12`

5. Escreva uma função que recebe um vetor **vet** de tamanho **n**, bem como, um elemento **elem** a ser procurado. A função deve retornar a posição (índice) do elemento ou -1 caso ele não esteja no vetor.
   `int find(int vet[], int n, int elem)`

6. Escreva uma função que recebe um vetor **vet** de tamanho **n.** A função deve imprimir o maior e o menor valores contidos no vetor.
   `void findMinMax(int vet[], int n)`

7. Escreva uma função que recebe um vetor **vet** de tamanho **n**, bem como, um elemento **elem** a ser procurado. A função deve substituir todas as ocorrência de **elem** por -1.
   `void replaceAll(int vet[], int n, int elem)`

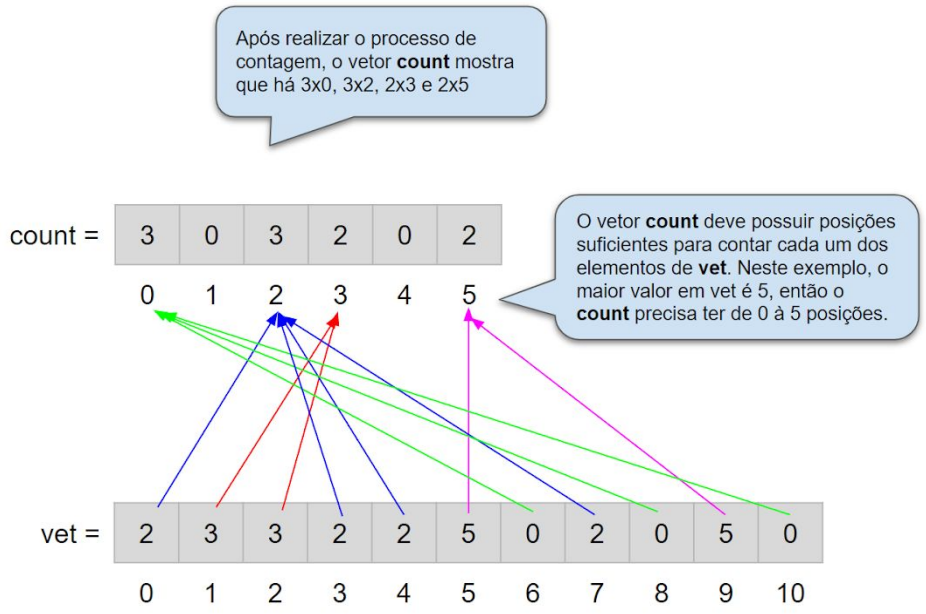
8. Escreva uma função que recebe um vetor **vet** de tamanho **n** e inverte os seus elementos.
   `void reverse(int vet[], int n)`

9. Escreva uma função que faz a leitura de **n** números inteiros e os coloca no vetor **vet** fornecido. Considere que o **vet** possui tamanho **n**.
   `void readVector(int vet[], int n)`

10. Escreva uma função que faz a leitura de **n** números inteiros e os imprime na ordem contrária a que foram digitados.
    `void reverseInput(int n)`

11. Escreva uma função que recebe um vetor **vet** de tamanho **n** preenchido com números naturais (inteiros positivos). A função deve imprimir a quantidade de ocorrências de cada número no vetor. Dica: utilize um vetor **count** para armazenar a contagem de cada elemento no vetor **vet**, relacionando os valores em **vet** com as posições em **count**. Observe a explicação abaixo:
    `void countElements(int vet[], int n)`

Passo 1: encontrar o maior elemento em **vet** (variável max);
Passo 2: criar vetor **count** com max+1 posições;
Passo 3: passar por todos os elementos de **vet**, countando suas ocorrências em **count**;
Passo 4: passar por todos os elementos de **count**, imprimindo os números e suas ocorrências.



12. Escreva uma função que recebe uma quantia de dinheiro **x** e informa a quantidade mínima de cédulas equivalente ao valor. Considere apenas valores inteiros e cédulas de $1, $5, $10, $50 e $100 reais. Dica 1: comece pela maior cédula possível ($100) e passe para uma menor quando não for mais possível dividir **x** pela cédula. Dica 2: use um vetor auxiliar **bills** para armazenar os valores das 5 tipos de cédulas e um outro vetor **count**, para armazenar a contagem de cada cédula. Função **`void minBills(int x)`**

    a. índice 0: notas de $1
    b. índice 1: notas de $5
    c. índice 2: notas de $10
    d. índice 3: notas de $50
    e. índice 4: notas de $100

```
Exemplo:

Quantia? R$ 209↵
2 cédulas de R$100,00
1 cédula de R$5,00
4 cédulas de R$1,00
```