UTFPR - Universidade Tecnológica Federal do Paraná
DACOM - Departamento de Computacão
BCC31A :: Algoritmos

**Exercícios 07 :: Vetores**

Instruções Gerais

- Utilize a extensão .c e o compilador gcc.
- Ao final, compacte tudo em uma pasta em formato ZIP e envie pelo Moodle.
- Faça cada exercício em uma função. Desta forma, todos os exercícios ficarão em um único programa (arquivo). Faça chamadas de teste das funções dentro da função main().
  Veja o exemplo abaixo:

```c
#include <stdio.h>
// Calcula a soma dos elementos do vetor
int vectorSum(int n, int v[n]) {
  int sum = 0;
  for (int i = 0; i < n; i++) {
     sum += v[i];
  }
  return sum;
}
```

```c
int main(void) {
  int v[7] = {1, 23, 4, 8, 41, 7, 3};
  printf("Soma: %d\n", vectorSum(7,v));

  return 0;
}
```

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

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

8. Escreva uma função que recebe um vetor **vet** de tamanho **n** e inverte os seus elementos.
   `void reverse(int n, int vet[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 n, int vet[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 inteiros positivos. A função deve imprimir as ocorrências (contagem) de cada número no vetor. Dica: utilize um vetor **count** para armazenar a contagem de cada elemento no vetor **vet**, relacionando as posições de **count** aos valores em **vet**.
    `void countElements(int n, int vet[n])`

12. Escreva uma função que recebe uma quantia de dinheiro **x** e imprime 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. Use também um outro vetor **count,** para armazenar a contagem de cada cédula.

    ```
    int bills[5] = {1,5,10,50,100};// tabela de tipos de cédulas
    int count[5] = {0,0,0,0,0};    // conta as ocorrências de cada tipos de cédula

    void minBills(int x)

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

13. Escreva uma função que recebe pontos X,Y em um vetor **points** de tamanho **n**. O vetor conterá os pontos sequencialmente: [X1, Y1, X2, Y2, X3, Y3,...]. A função deve informar a distância entre cada cada par de pontos. Distância: sqrt( (x2 - x1)*(x2 - x1) + (y2 - y1)*(y2 - y1) ). Função **square root**: https://www.tutorialspoint.com/c_standard_library/c_function_sqrt.htm).
    `void distances(int n, int points[n])`

14. Escreva uma função que recebe um vetor **days** de tamanho 7 que exibe um histograma da variação da temperatura durante a semana. O vetor **days** armazena as temperaturas para cada dia da semana, iniciando por **domingo**. Por exemplo, se as temperaturas em **days** forem 19, 21, 25, 22, 20, 17 e 15°C, a função deverá exibir:

```
D: >>>>>>>>>>>>>>>>>>
S: >>>>>>>>>>>>>>>>>>>>
T: >>>>>>>>>>>>>>>>>>>>>>>
Q: >>>>>>>>>>>>>>>>>>>>
Q: >>>>>>>>>>>>>>>>>>
S: >>>>>>>>>>>>>>>>
S: >>>>>>>>>>>>>>>
void histogram(int days[])
```

15. Usando a função desenvolvida nos exercício anterior, escreva uma função que recebe o vetor **days** de tamanho 7 e exibe:
    a) As temperaturas média, mínima, máxima;
    b) O histograma.

```
void tempReport(int days[])
```

16. Escreva uma função que recebe um vetor **vet** de tamanho **n**. O vetor contém inteiros positivos e posições livres, marcadas com -1. A função deve desfragmentar o vetor, colocando todos os valores válidos à esquerda.

```
void defrag(int n, int vet[n])

Exemplo:
int v[9] = {1, 6, -1, 9, 4, -1, -1, 2, -1} // vetor original
defrag(9, v);
// v = {1, 6, 9, 4, 2, -1, -1, -1, -1}
```

17. Escreva uma função que recebe três vetores e seus tamanhos. A função deve concatenar (juntar) o conteúdo de **v1** e **v2** em **v3**. Considere que **v3** tem tamanho **n1** + **n2**. Os vetores contém apenas inteiros positivos.

```
void vecConcat(int n1, int v1[n1], int n2, int v2[n2], int v3[])

Exemplo:
int v1[5] = {1,2,3,4,5};
int v2[3] = {2,3,8};
int v3[8];
vecConcat(5, v1, 3, v2, v3);  // v3 = {1,2,3,4,5,2,3,8}
```

18. Escreva uma função que recebe três vetores e seus tamanhos. A função deve realizar a união entre os vetores **v1** e **v2**, colocando os valores em **v3**. Considere que **v3** tem tamanho **n1** + **n2**. Os vetores contém apenas inteiros positivos e **v3** deve ser iniciado com 0.

```
void vecUnion(int n1, int v1[n1], int n2, int v2[n2], int v3[])

Exemplo:
int v1[5] = {1,2,3,4,5};
int v2[3] = {2,3,8};
int v3[8] = {0};
vecUnion(5, v1, 3, v2, v3);   // v3 = {1,2,3,4,5,8,0,0}
```

19. Escreva uma função que recebe três vetores e seus tamanhos. A função deve realizar a intersecção entre os vetores **v1** e **v2**, colocando os valores em **v3**. Considere que **v3** tem tamanho min(**n1**, **n2**). Os vetores contém apenas números naturais (inteiros positivos) e **v3** deve ser iniciado com 0.

```
void vecIntersection(int n1, int v1[n1], int n2, int v2[n2], int v3[])

Exemplo:
int v1[5] = {1,2,3,4,5};
int v2[3] = {2,3,8};
int v3[3] = {0};
vecIntersection(5, v1, 3, v2, v3);  // v3 = {2,3,0}
```