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

Exercícios 05 :: Vetores e Funções

Instruções Gerais

- Faça cada exercício em um programa (arquivo) distinto.
- Utilize a extensão .c e o compilador gcc.
- Utilize o editor de sua preferência: Code Blocks, VS Code, Dev C++, etc.

1. Escreva uma função que verifica se um caractere é uma letra (maiúscula/minúscula). Ela deve devolver 1 (true), caso o parâmetro seja uma letra, ou 0 (zero), em caso contrário.

   Função: **`int is_letter(char ch)`**

2. Escreva uma função que verifica se um dado número é primo. Ela deve devolver 1 (true) caso primo ou 0 (zero), caso contrário.

   Função: **`int is_prime(int n)`**

3. Escreva uma função que imprime N números aleatórios. A função receberá como parâmetro a quantidade de números (n) e o limite para sorteio (max). Os números devem ser sorteados entre 0 (incluído) e max (excluído), isto é, **[0..max)**. OBS: Será necessário utilizar a função **rand()** da biblioteca **<stdlib.h>**.

   Função: **`void print_random(int n, int max)`**

   Exemplo de uso:
   ```
   print_random(20, 100);
   Saída: 84 87 78 16 94 36 0 93 50 22 63 2 91 60 64 27 41 27 73 37
   ```

4. Modifique a função do exercício anterior para que sorteie entre min e max, isto é, no intervalo [min..max].

   Função: **`void print_random2(int n, int min, int max)`**

   Exemplo de uso:
   ```
   print_random2(18, 50, 100);
   Saída: 67 73 55 79 87 50 73 85 100 57 85 91 80 69 57 74 88 52
   ```

5. Escreva uma função que calcula e devolve o somatório de um número: $\sum_{i=1}^{n} i$.

   Função: **`int summation(int n)`**

   Exemplo de uso:
   ```
   int res = summation(5); // 1 + 2 + 3 + 4 + 5 = 15
   ```

6. Escreva uma função que calcula e devolve a soma dos fatoriais até um dado número, $\sum_{i=1}^{n} i!$.

Função: **int factorial_sum(int n)**

Exemplo de uso:
```
int res = factorial_sum(5); // 1! + 2! + 3! + 4! + 5! = 153
```

DICA: observe a representação dos cálculos.

```
1! + 2! + 3! + 4! + 5! = 153, é o mesmo que:

1! = 1                 = 1  +
2! = 1 x 2             = 2  +
3! = 1 x 2 x 3         = 6  +
4! = 1 x 2 x 3 x 4     = 24 +
5! = 1 x 2 x 3 x 4 x 5= 120
                         153
```

7. Escreva uma função que verifica se o parâmetro é um **número perfeito.** Um número perfeito é um inteiro positivo que é igual a soma de seus divisores positivos, excluindo o próprio número. Exemplo: 6 tem os divisores 1, 2 e 3 (excluindo o próprio 6), e 1 + 2 + 3 = 6. Logo, 6 é um número perfeito. A função deve devolver 1 (true) se número positivo ou 0 (false), caso contrário.

   Função: **int is_perfect_number(int n)**

8. Escreva uma função que recebe um número inteiro entre -10 e 10 e o escreve por extenso. Caso o número esteja fora desse intervalo, a função deve informar o erro.

   Função: **void number_in_full_10(int n)**

   Exemplo de uso:
   ```
   number_in_full_10(-8);
   Número: Menos Oito
   ```

9. Escreva uma função que recebe um número inteiro entre -100 e 100 e o escreve por extenso. Caso o número esteja fora desse intervalo, a função deve informar o erro.

   Função: **void number_in_full_100(int n)**

   Exemplo de uso:
   ```
   number_in_full_100(45);
   Saída: Quarenta e Cinco
   ```

Vetores + Funções

10. Escreva uma função que recebe um vetor **vet** de tamanho **n** e o imprime em ordem reversa.

    void print_reverse(int n, int vet[n])

11. Escreva uma função que recebe um vetor **vet** de tamanho **n** e imprime apenas os valores pares.

    void print_even(int n, int vet[n])

12. 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 set_positive(int n, int vet[n])

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

13. Escreva uma função que recebe um vetor **vet** de tamanho **n** e devolve a média aritmética simples dos valores contidos.

    int sum_values(int n, int vet[n])

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

14. 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)

15. 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 find_min_max(int n, int vet[n])

16. 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 replace_all(int n, int vet[n], int elem)

17. Escreva uma função que faz a leitura de **n** números inteiros e os coloca no vetor **vet** fornecido. Depois, deve imprimir o vetor em ordem reversa. Considere que o **vet** possui tamanho **n**.

    void read_vector(int n, int vet[n])

18. Escreva uma função que recebe um vetor **vet** de tamanho **n** e inverte os seus elementos.

    void reverse(int n, int vet[n])

19. 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}