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

Exercícios 06 :: Funções

Instruções Gerais

- Faça cada exercício em uma função distinta, colocando todas em um mesmo arquivo.
- Utilize a extensão .c, o compilador gcc e um editor de sua preferência.
- Não é permitido o uso de funções prontas para resolver os problemas, exceto scanf() e printf(), ou quando mencionado no exercício.

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

   Função: **int isPrime(int n)**

2. 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 1 e max. OBS: Será necessário utilizar a função **rand()** da biblioteca **<stdlib.h>**.

   Função: **void printRandom(int n, int max)**

   Exemplo de uso:

```
printRandom(20, 100);
Saída: 84 87 78 16 94 36 87 93 50 22 63 28 91 60 64 27 41 27 73 37
```

3. Modifique a função do exercício anterior para que sorteie entre -max e max.

   Função: **void printRandom2(int n, int max)**

   Exemplo de uso:

```
printRandom2(20, 100);
Saída: 67 73 55 -69 87 -29 73 85 -1 -57 25 -45 81 19 27 -47 -19 -47 45
-27
```

4. 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
```

5. Escreva uma função que calcula e devolve a soma dos fatoriais até um dado número, $\sum_{i=1}^{n} i!$. Você pode escrever uma solução que utiliza 2 laços aninhados.

   Função: **int factorialSum(int n)**

   Exemplo de uso:

```
int res = factorialSum(5); // 1! + 2! + 3! + 4! + 5! = 153
```

6. Escreva uma nova versão da função do exercício anterior, agora utilizando apenas um único laço para realizar a operação solicitada na função. Dica: observe a representação dos cálculos.

   Função: **int factorialSum2(int n)**

   Exemplo de uso:
   ```
   int res = factorialSum2(5);

   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 isPerfectNumber(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 numbersInFull10(int n)**

   Exemplo de uso:
   ```
   numbersInFull10(-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 numbersInFull100(int n)**

   Exemplo de uso:
   ```
   numbersInFull100(45);
   Saída: Quarenta e Cinco
   ```

10. Escreva uma função que recebe um número inteiro entre 1 e 100 e o escreve em algarismos romanos. Caso o número esteja fora desse intervalo, a função deve informar o erro.
    Referência: https://www.somatematica.com.br/fundam/tabela1.php

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

    Exemplo de uso:

```
intToRoman(47);            Saída: LXVII
```

11. Escreva uma função que imprime uma linha com duas “pontas”. A função receberá como parâmetros:
    a. A largura da linha
    b. O caractere de preenchimento da linha;
    c. O caractere das pontas da linha.

Função: **void printLine(int n, char fill, char edge)**

Exemplo de uso:

```
printLine(10, '-', '+');
Saída: +--------+
```

12. Utilizando a função do exercício anterior, escreva uma nova função que imprime uma caixa, recebendo como parâmetros:
    a. Largura e altura da caixa;
    b. O caractere de preenchimento da caixa;
    c. O caractere das bordas da caixa.

Função: **void printBoxCustom(int width, int height, char fill, char edge)**

Exemplo de uso:

```
printBoxCustom(10, 5, '#', 'o');

o--------o
|########|
|########|
|########|
o--------o
```

13. Escreva uma função que imprime uma caixa com o caractere informado. Utilize apenas um laço para realizar a operação solicitada. A função deve receber as dimensões da caixa (largura e altura).

Função: **void printBoxFilled(int width, int height, char ch)**

Exemplo de uso: `printBoxFilled(4, 9, ‘X’);`

```
XXXXXXXXX
XXXXXXXXX
XXXXXXXXX
XXXXXXXXX
```

14. Escreva uma função que imprime uma caixa malhada com 0’s e 1’s. A função deve receber as dimensões da caixa (largura e altura). Experimente criar duas versões da função: uma resolvendo o problema com dois laços aninhados e outra usando apenas um laço.

Função: **void printBox01(int width, int height)**

Exemplo de uso: `printBox01(4, 9);`

```
010101010
010101010
010101010
010101010
```

15. Escreva uma função que imprime uma caixa progressiva, preenchida com contagens até um dado número. A função deve receber as dimensões da caixa (largura e altura) e o limite da contagem. Experimente criar duas versões da função: uma resolvendo o problema com dois laços aninhados e outra usando apenas um laço.

    Função: **void printBoxProgressive(int width, int height, int max)**

    Exemplo de uso: printBoxProgressive(4, 9, 5);

    ```
    012340123
    401234012
    340123401
    234012340
    ```

16. Escreva uma função que faz a leitura de vários números do terminal, até que seja informado zero. Após, ela deve devolver, via parâmetros de saída:
    a. O maior valor
    b. O menor valor

    Função: **void batchReport(int* max, int* min)**

    Exemplo de uso:

    ```
    int maior, menor;

    batchReport(&maior, &menor);

    Exemplo de números digitados: 4 7 18 16 14 -16 7 13 -10 -2 3 8 11 20 4 7
    // maior: 20
    // menor: -16
    ```

17. Escreva uma função que sorteia e imprime uma dada quantidade de números. O parâmetro n define a quantidade de números e o limite para sorteio. Após, deve devolver, via parâmetros de saída:
    a. A soma dos pares
    b. A quantidade de números primos

    OBS: Será necessário utilizar a função **rand()** da biblioteca **<stdlib.h>**. Você pode utilizar a função do exercício 1 para verificar se um dado número é primo.

    Função: **void randReport(int n, int* evenSum, int* primes)**

    Exemplo de uso:

    ```
    int somaPares, primos;

    // sorteia números de 1 à 20
    randReport(20, &somaPares, &primos);

    Saída: 4 7 18 16 14 16 7 13 10 2 3 8 11 20 4 7 1 7 13 17
    // somaPares: 112
    // primos: 11
    ```