Lista de Exercícios 7 – Modularização (Procedimentos e Funções)

- ***Procedimentos: Passagem de parâmetros.***

5) Escreva um procedimento que receba um número natural e imprima os três primeiros caracteres do dia da semana correspondente ao número. Por exemplo, 7 corresponde à "SAB". O procedimento deve mostrar uma mensagem de erro caso o número recebido não corresponda à um dia da semana. Gere também um algoritmo que utilize esse procedimento, chamando-o, mas antes lendo um valor para passagem de parâmetro.

```
function l6p05b;
X = input('Digite um numero correspondente a um dia da semana: ');
DIASEMANA(X);

function DIASEMANA(NUM);
if ( NUM >= 1 ) & ( NUM <= 7 )
  if ( NUM == 1 )
    fprintf(1,'%d corresponde a: DOM\n',NUM);
  elseif ( NUM == 2 )
    fprintf(1,'%d corresponde a: SEG\n',NUM);
  elseif ( NUM == 3 )
    fprintf(1,'%d corresponde a: TER\n',NUM);
  elseif ( NUM == 4 )
    fprintf(1,'%d corresponde a: QUA\n',NUM);
  elseif ( NUM == 5 )
    fprintf(1,'%d corresponde a: QUI\n',NUM);
  elseif ( NUM == 6 )
    fprintf(1,'%d corresponde a: SEX\n',NUM);
  elseif ( NUM == 7 )
    fprintf(1,'%d corresponde a: SAB\n',NUM);
  end
else
  fprintf(1,'parametro recebido (%d) nao correspondente a um dia da semana!\n',NUM);
end
```

- ***Funções que verificam uma situação, retorno booleano (verdadeiro, falso)***

10) Um número é dito ser **capicua** quando lido da esquerda para a direita é o mesmo que quando lido da direita para a esquerda. O ano 2002, por exemplo, é **capicua**. Então, elabore uma função para verificar se um número possui essa característica. Caso o número seja **capicua**, a função deve retornar 1 e 0 em caso contrário. Escreva também um algoritmo para testar tal função.

```
function l6p10;
X = input('Digite um numero: ');
if CAPICUA(X)
  fprintf(1,'%d eh capicua!\n',X);
else
  fprintf(1,'%d nao eh capicua!\n',X);
end

function REV = REVERSO(NUM);
REV = 0;
while ( NUM ~= 0 )
  RET = mod(NUM,10);
  NUM = floor(NUM / 10);
  REV = REV * 10 + RET;
end

function CAPICUA = CAPICUA(NUM);
if (REVERSO(NUM) == NUM)
  CAPICUA = 1;
else
  CAPICUA = 0;
end
```

- ***Funções que retornam um valor calculado***

15) Criar uma função que verifique quantas vezes um número inteiro ***x*** é divisível por um número inteiro ***y***. A função deve retornar -1 caso não seja possível calcular. Escreva também um algoritmo para testar tal função.

```
function l6p15;
X = input('Digite um valor para x: ');
Y = input('Digite um valor para y: ');
RET = VEZDIV(X,Y);
if ( RET == -1 )
  disp('Impossivel calcular, divisao por zero!');
else
  fprintf(1,'%d eh divisivel por %d - %d vez(es)',X,Y,RET);
end

function VEZ = VEZDIV(X,Y);
if ( Y == 0 )
  VEZ = -1; % divisao por zero
else
  VEZ = 0;
  RET = mod(X,Y);
  while ( RET == 0 )
    VEZ = VEZ + 1;
    X = floor(X / Y);
    RET = mod(X,Y);
  end
end
```

- ***Funções retornando mais de um parâmetro***

20) Construa uma função, que receba três coeficientes relativos à uma equação de segundo grau ($a.x^2 + b.x + c = 0$) e calcule suas raízes através da fórmula de báscara:

$$x = \frac{-b \pm \sqrt{\Delta}}{2a} \qquad \Delta = b^2 - 4ac$$

A função deve levar em conta a possibilidade da existência de nenhuma, uma ou duas raízes. A função deve retornar o número de raízes ou -1 em caso de inconsistência. Os valores das raízes devem ser retornados. Construa também um algoritmo para utilizar a função construída.

```
function l6p20;
disp('Equacao do 2o. grau - a0.x^2 + a1.x + a2 = 0');
AZ = input('Digite o coeficiente a0: ');
AU = input('Digite o coeficiente a1: ');
AD = input('Digite o coeficiente a2: ');
[RET,X1L,X2L] = EQ2(AZ,AU,AD);
if ( RET == -1 )
    disp('Incosistencia, possivel a0 = 0!');
elseif ( RET == 0 )
    disp('Nao existe raiz real para tal equacao!');
elseif ( RET == 1 )
    fprintf(1,'Uma unica raiz real, x1 = x2 = %f\n',X1L);
elseif ( RET == 2 )
  disp('Duas raizes reais diferentes!');
  fprintf(1,'x1 = %f\n',X1L);
  fprintf(1,'x2 = %f\n',X2L);
end

function [EQ2,X1,X2] = EQ2(A,B,C);
if ( A == 0 )
  EQ2 = -1;
  X1 = -1;
  X2 = -1;
else
  DELTA = B * B - 4 * A * C;
  if ( DELTA < 0 )
    EQ2 = 0;
    X1 = -1;
    X2 = -1;
  elseif ( DELTA == 0 )
    EQ2 = 1;
    X1 = -B / ( 2 * A );
    X2 = X1;
  else % DELTA > 0
    EQ2 = 2;
    X1 = ( -B + sqrt(DELTA) ) / ( 2 * A );
    X2 = ( -B - sqrt(DELTA) ) / ( 2 * A );
  end
end
```

- ***Transformações***

25) Crie uma função que realize a conversão da escala Kelvin (***K*** - escala absoluta) para a escala Fahrenheit (***F***). Sabe-se que 273K equivale a 32ºF e a cada variação de 10 unidades na escala Kelvin equivale a 18 na escala Fahrenheit. A função deve retornar zero caso não seja possível realizar a conversão e um em caso contrário. Crie também um algoritmo para testar tal função.

```
function L6P25;
KL = input('Entre com a temperatura na escala Kelvin: ');
[VAL,FL] = CONVKF(KL);
if ( VAL == 0 )
  fprintf(1,'Impossivel calcular, temperatura Kelvin negativa!\n');
else
  fprintf(1,'A correspondente temperatura na escala Fahrenheit eh %.2f\n',FL);
end

function [RET,F] = CONVKF(K);
% 273K - 32F, 373K - 212F
if ( K < 0 )
  RET = 0;
  F = 0;
else
  RET = 1;
  F = ( 5 * 212 - 9 * (373-K) ) / 5;
end
```

- ***Funções recursivas***

30) O fatorial de um número ***n***, inteiro e positivo, pode ser definido recursivamente, ou seja:

$$n! = \begin{cases} 1 & se \quad n = 0 \\ n.(n-1)! & se \quad n \geq 1 \end{cases}$$

Então, pede-se que seja criada uma função recursiva que calcule o fatorial de um número n. A função deve retornar -1 caso não seja possível calcular o fatorial Além disso, crie um algoritmo que leia um valor, utilize a função criada para calcular o fatorial e imprima o valor computado.

```
function l6p30;
I = input('Digite um numero: ');
RET = FAT(I);
if ( RET == -1 )
  fprintf(1,'Impossivel calcular o fatorial de %d\n',I);
else
  fprintf(1,'O Fatorial de %d eh %d\n',I,RET);
end

function F = FAT(N);
if ( N < 0 )
  F = -1;
elseif (N == 0)
  F = 1;
else
  F = N * FAT(N-1);
end
```