Universidade Federal de Ouro Preto – UFOP
Instituto de Ciências Exatas e Biológicas – ICEB
Departamento de Computação – DECOM
Disciplina: Algoritmos e Estrutura de Dados I – CIC102
Professor: David Menotti (menottid@gmail.com)

Lista de Exercícios 06 – 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.

```
algoritmo L6P05;
var
  inteiro: X;

procedimento DIASEMANA(inteiro: NUM);
constante
  SEM[1..7,3] = ("DOM","SEG","TER","QUA","QUI","SEX","SAB");
início
  se ( NUM >= 1 ) e ( NUM <= 7 ) então
    imprima(NUM," corresponde a: ",SEM[NUM])
  senão
    imprima("parametro recebido (",NUM, ") nao correspondente a um dia da semana! ");
  fim-se
fim

início
  imprima("Digite um numero correspondente a um dia da semana: ");
  leia(X);
  DIASEMANA(X);
fim
```

```
algoritmo L6P05;
var
  inteiro: X;

procedimento DIASEMANA(inteiro: NUM);
início
  se ( NUM >= 1 ) e ( NUM <= 7 ) então
    se ( NUM = 1 ) então
      imprima(NUM," corresponde a: DOM");
    fim-se
    se ( NUM = 2 ) então
      imprima(NUM," corresponde a: SEG");
    fim-se
    se ( NUM = 3 ) então
      imprima(NUM," corresponde a: TER");
    fim-se
    se ( NUM = 4 ) então
      imprima(NUM," corresponde a: QUA");
    fim-se
    se ( NUM = 5 ) então
      imprima(NUM," corresponde a: QUI");
    fim-se
    se ( NUM = 6 ) então
      imprima(NUM," corresponde a: SEX");
    fim-se
    se ( NUM = 7 ) então
      imprima(NUM," corresponde a: SAB");
    fim-se
  senão
    imprima("parametro recebido (",NUM,") nao correspondente a um dia da semana! ");
  fim-se
fim

início
  imprima("Digite um numero correspondente a um dia da semana: ");
  leia(X);
  DIASEMANA(X);
fim
```

```
program l6p05;
var
  X: integer;

procedure DIASEMANA(NUM: integer);
const
  SEM: array [1..7] of string[3] =
    ('DOM','SEG','TER','QUA','QUI','SEX','SAB');
begin
  if ( NUM >= 1 ) and ( NUM <= 7 ) then
    writeLn(NUM,' corresponde a: ',SEM[NUM])
  else
    writeLn('parametro recebido (',NUM,') nao correspondente a um dia da semana!');
end;

begin
  write('Digite um numero correspondente a um dia da semana: ');
  readLn(X);
  DIASEMANA(X);
end.
```

```
program l6p05b;
var
  X: integer;

procedure DIASEMANA(NUM: integer);
begin
  if ( NUM >= 1 ) and ( NUM <= 7 ) then
  begin
    if ( NUM = 1 ) then
      writeLn(NUM,' corresponde a: DOM');
    if ( NUM = 2 ) then
      writeLn(NUM,' corresponde a: SEG');
    if ( NUM = 3 ) then
      writeLn(NUM,' corresponde a: TER');
    if ( NUM = 4 ) then
      writeLn(NUM,' corresponde a: QUA');
    if ( NUM = 5 ) then
      writeLn(NUM,' corresponde a: QUI');
    if ( NUM = 6 ) then
      writeLn(NUM,' corresponde a: SEX');
    if ( NUM = 7 ) then
      writeLn(NUM,' corresponde a: SAB');
  end
  else
    writeLn('parametro recebido (',NUM,') nao correspondente a um dia da semana!');
end;

begin
  write('Digite um numero correspondente a um dia da semana: ');
  readLn(X);
  DIASEMANA(X);
end.


function l6p05;
X = input('Digite um numero correspondente a um dia da semana: ');
DIASEMANA(X);

function DIASEMANA(NUM);
SEM = {'DOM';'SEG';'TER';'QUA';'QUI';'SEX';'SAB'};
if ( NUM >= 1 ) & ( NUM <= 7 )
  fprintf(1,'%d corresponde a: %s\n',NUM,SEM{NUM});
else
  fprintf(1,'parametro recebido (%d) nao correspondente a um dia da semana!\n',NUM);
end


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.

```
algoritmo L6P10;
var
  inteiro: X;

função REVERSO(inteiro: NUM) :inteiro;
var
  inteiro: RET, MUL, REV;
início
  REV <- 0;
  enquanto ( NUM <> 0 ) faça
    RET <- NUM mod 10;
    NUM <- NUM div 10;
    REV <- REV * 10 + RET;
  fim-enquanto
  REVERSO <- REV;
fim

função CAPICUA(inteiro: NUM) :lógico;
início
  se ( REVERSO(NUM) = NUM ) então
    CAPICUA <- verdadeiro;
  senão
    CAPICUA <- falso;
  fim-se
fim

início
  imprima("Digite um numero: ");
  leia(X);
  se ( CAPICUA(X) ) então
    imprima(X," eh capicua!");
  senão
    imprima(X," nao eh capicua!");
  fim-se
fim
```

```
program l5p10;
var
  X: integer;

  function REVERSO(NUM: integer) :integer;
  var
    RET, MUL, REV: integer;
  begin
    REV := 0;
    while ( NUM <> 0 ) do
    begin
      RET := NUM mod 10;
      NUM := NUM div 10;
      REV := REV * 10 + RET;
    end;
    REVERSO := REV;
  end;

  function CAPICUA(NUM: integer) :boolean;
  begin
    if (REVERSO(NUM) = NUM) then
      CAPICUA := true
    else
      CAPICUA := false;
  end;

begin
  write('Digite um numero: ');
  readLn(X);
  if CAPICUA(X) then
    writeLn(X,' eh capicua!')
  else
    writeLn(X,' nao eh capicua!');
end.


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.

```
algoritmo L6P15;
var
  inteiro: X, Y, RET;

função VEZDIV(inteiro: X, Y) : inteiro;
var
  inteiro: VEZ, RET;
início
  se ( Y = 0 ) então
    VEZDIV <- -1; {divisao por zero}
  senão
    VEZ <- 0;
    RET <- X mod Y;
    enquanto ( RET = 0 ) faça
      VEZ <- VEZ + 1;
      X   <- X div Y;
      RET <- X mod Y;
    fim-enquanto
    VEZDIV <- VEZ;
  fim-se
fim

início
  imprima("Digite um valor para x: ");
  leia(X);
  imprima("Digite um valor para y: ");
  leia(Y);
  RET <- VEZDIV(X,Y);
  se ( RET = -1 ) então
    imprima("Impossivel calcular, divisao por zero! ");
  senão
    imprima(X, " eh divisivel por ",Y," - ",RET," vez(es) ");
  fim-se
fim
```

```
program l6p15;
var
  X,Y,RET: integer;

function VEZDIV(X,Y: integer) :integer;
var
  VEZ, RET: integer;
begin
  if ( Y = 0 ) then
    VEZDIV := -1 {divisao por zero}
  else
  begin
    VEZ := 0;
    RET := X mod Y;
    while ( RET = 0 ) do
    begin
      VEZ := VEZ + 1;
      X := X div Y;
      RET := X mod Y;
    end;
    VEZDIV := VEZ;
  end;
end;

begin
  write('Digite um valor para x: ');
  readLn(X);
  write('Digite um valor para y: ');
  readLn(Y);
  RET := VEZDIV(X,Y);
  if ( RET = -1 ) then
    writeLn('Impossivel calcular, divisao por zero!')
  else
    writeLn(X,' eh divisivel por ',Y,' - ',RET,' vez(es)');
end.
```

```
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.

```
algoritmo L6P20;
var
  real: A0,A1,A2,X1L,X2L;
  inteiro: RET;

função EQ2(real: A, B, C;var real: X1, X2) :inteiro;
var
  real: DELTA;
início
  se ( A = 0 ) então
    EQ2 <- -1;
  senão
    DELTA <- B * B - 4 * A * C;
    se ( DELTA < 0 ) então
      EQ2 <- 0;
    senão
      se ( DELTA = 0 ) então
        EQ2 <- 1;
        X1 <- -B / ( 2 * A );
        X2 <- X1;
      senão { DELTA > 0 }
        EQ2 <- 2;
        X1 <- ( -B + raiz(DELTA) ) / ( 2 * A );
        X2 <- ( -B - raiz(DELTA) ) / ( 2 * A );
      fim-se
    fim-se
  fim-se
fim

início
  imprima("Equacao do 2o. grau - a0.x^2 + a1.x + a2 = 0");
  imprima("Digite o coeficiente a0: ");
  leia(A0);
  imprima("Digite o coeficiente a1: ");
  leia(A1);
  imprima("Digite o coeficiente a2: ");
  leia(A2);
  RET <- EQ2(A0,A1,A2,X1L,X2L);
  se ( RET = -1 ) então
    imprima("Incosistencia, possivel a0 = 0! ");
  senão
    se ( RET = 0 ) então
      imprima("Nao existe raiz real para tal equacao! ");
    senão
      se ( RET = 1 ) então
        imprima("Uma unica raiz real, x1 = x2 = ",X1L);
      senão
        se ( RET = 2 ) então
          imprima("Duas raizes reais diferentes");
          imprima("x1 = ",X1L);
          imprima("x2 = ",X2L);
        fim-se
      fim-se
    fim-se
  fim-se
fim
```

```
program l6p20;
var
  A0,A1,A2,X1L,X2L: real;
  RET: integer;

function EQ2(A,B,C: real;var X1,X2: real) :integer;
var
  DELTA: real;
begin
  if ( A = 0 ) then
    EQ2 := -1
  else
  begin
    DELTA := B * B - 4 * A * C;
    if ( DELTA < 0 ) then
      EQ2 := 0
    else if ( DELTA = 0 ) then
    begin
      EQ2 := 1;
      X1 := -B / ( 2 * A );
      X2 := X1;
    end
    else {DELTA > 0 }
    begin
      EQ2 := 2;
      X1 := ( -B + SqRt(DELTA) ) / ( 2 * A );
      X2 := ( -B - SqRt(DELTA) ) / ( 2 * A );
    end;
  end;
end;

begin
  writeLn('Equacao do 2o. grau - a0.x^2 + a1.x + a2 = 0');
  write('Digite o coeficiente a0: ');
  readLn(A0);
  write('Digite o coeficiente a1: ');
  readLn(A1);
  write('Digite o coeficiente a2: ');
  readLn(A2);
  RET := EQ2(A0,A1,A2,X1L,X2L);
  if ( RET = -1 ) then
    writeLn('Incosistencia, possivel a0 = 0!')
  else if ( RET = 0 ) then
    writeLn('Nao existe raiz real para tal equacao!')
  else if ( RET = 1 ) then
    writeLn('Uma unica raiz real, x1 = x2 = ',X1L:5:4)
  else if ( RET = 2 ) then
  begin
    writeLn('Duas raizes reais diferentes');
    writeLn('x1 = ',X1L:5:4);
    writeLn('x2 = ',X2L:5:4);
  end
end.
```

```
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.

```
algoritmo L6P25;
var
  real : FL, KL;

função CONVKF(real : K; var real : F) : inteiro;
início
  { 273K - 32F, 373K - 212F}
  se ( K < 0 ) então
    CONVKF <- 0;
  senão
    CONVKF <- 1;
    F <- ( 5 * 212 - 9 * (373-K) ) / 5;
  fim-se
fim

início
  leia(KL);
  se ( CONVKF(KL,FL) = 0 ) então
    imprima("Impossivel calcular, temperatura Kelvin negativa!");
  senão
    imprima("A correspondente temperatura na escala Fahrenheit eh ",FL);
  fim-se
fim
```

```
program L6P25;
var
  FL, KL : real;
function CONVKF(K : real; var F : real) : integer;
begin
  if ( K < 0 ) then
    CONVKF := 0
  else
  begin
    { 273K - 32F, 373K - 212F}
    CONVKF := 1;
    F := ( 5 * 212 - 9 * (373-K) ) / 5;
  end;
end;
begin
  write('Entre com a temperatura na escala Kelvin: ');
  readLn(KL);
  if ( CONVKF(KL,FL) = 0 ) then
    writeLn('Impossivel calcular, temperatura Kelvin negativa!')
  else
    writeLn('A correspondente temperatura na escala Fahrenheit eh ',FL:3:2);
end.
```

```
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.

```
algoritmo L6P30;
var
  inteiro: I, RET;

função FAT(inteiro: N) :inteiro;
início
  se ( N < 0 ) então
    FAT <- -1;
  senão-se (N = 0) então
    FAT <- 1;
  senão
    FAT <- N * FAT(N-1);
  fim-se
fim

início
  imprima("Digite um numero: ");
  leia(I);
  RET <- FAT(I);
  se ( RET = -1 ) então
    imprima("Impossivel calcular o fatorial de ",I);
  senão
    imprima("O Fatorial de ",I," eh ",RET);
  fim-se
fim
```

```
program l6p30;
var
  I: integer;
  RET: real;

  function FAT(N : integer) :real;
  begin
    if ( N < 0 ) then
      FAT := -1
    else if (N = 0) then
      FAT := 1
    else
      FAT := N * FAT(N-1);
  end;

begin
  write('Digite um numero: ');
  readLn(I);
  RET := FAT(I);
  if ( RET = -1 ) then
    writeLn('Impossivel calcular o fatorial de ',I)
  else
    writeLn('O Fatorial de ',I,' eh ',RET:1:0);
end.
```

```
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
```