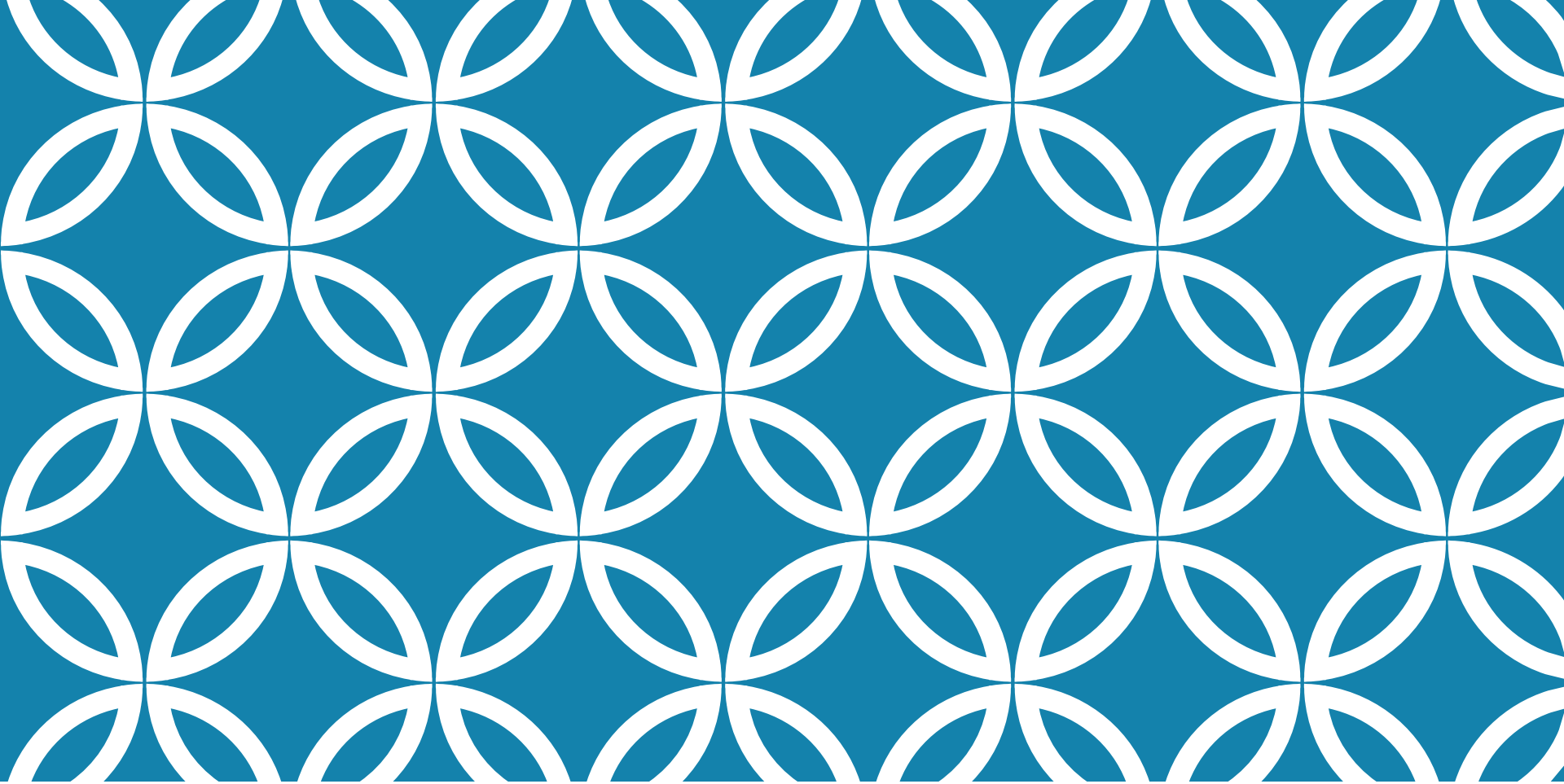

FUNÇÕES E PROCEDIMENTOS

Prof. André Vignatti

PORQUE?

- modularidade
- reaproveitamento de código
- legibilidade do código

CONCEITOS

1. **função** X **procedimento**
2. variáveis**: locais** X **globais**
3. passagem de parâmetros**: valor** X **referência**?

MÁXIMO DIVISOR COMUM

Um método de obter o MDC: **fatorar** os números em **primos**, escolher os fatores primos que se repetem, com a menor potência.

Ex: Calcular o MDC entre 72 e 270

$$72 = 2^3 \times 3^2$$

$$270 = 2^1 \times 3^3 \times 5$$

Assim, o MDC é $2^1 \times 3^2$

MDC: IDEIA PARA O ALGORITMO

1° passo - testar se 2 divide:

$$72 = 2^3 \times 9$$

$$270 = 2^1 \times 135$$

Então mdc(72,270) = 2^1 × **mdc(9,135).**

2° passo - testar se 3 divide:

$$9 = 3^2 \times 1$$

$$135 = 3^3 \times 5$$

Quando algum número é 1, pára

Então, mdc(72,270) = $2^1 \times 3^2 \times$ mdc(1,5) = $2^1 \times 3^2$

PSEUDO-CÓDIGO

```
begin
      read (a,b);
      mdc:= 1;

      (* descobre quantas vezes o 2 divide as duas entradas *)
      cont_a:= {numero de vezes que o 2 divide a};
      cont_b:= {numero de vezes que o 2 divide b};
      menor_cont:= {menor entre cont_a e cont_b};
      mdc:= mdc * {2 elevado a potencia menor_cont};
      a:= a div mdc;
      b:= b div mdc;

      (* repete o processo para todos os impares *)
      primo:= 3;
      while (a <> 1) and (b <> 1) do
      begin
            cont_a:= {numero de vezes que primo divide a}
            cont_b:= {numero de vezes que primo divide b}
            menor_cont:= {menor entre cont_a e cont_b};
            mdc:= mdc * {primo elevado a potencia menor_cont};
            a:= a div mdc;
            b:= b div mdc;
            primo:= primo + 2;          (* passa para o proximo impar *)
      end;
      writeln (mdc);
end.
```

```
program mdc_por_definicao;
var i, a, b, mdc, cont_a, cont_b, menor_cont, primo: integer;
begin
    (* inicializacao das variaveis principais *)
    read (a,b);
    mdc:= 1;

    (* descobre quantas vezes o 2 divide as duas entradas *)
    cont_a:= 0;
    while a mod 2 = 0 do
    begin
        cont_a:= cont_a + 1;
        a:= a div 2;
    end;

    cont_b:= 0;
    while b mod 2 = 0 do
    begin
        cont_b:= cont_b + 1;
        b:= b div 2;
    end;

    (* descobre qual dos contadores eh o menor *)
    if cont_a <= cont_b then
        menor_cont:= cont_a
    else
        menor_cont:= cont_b;

    (* atualiza o mdc para o 2 *)
    i:= 1;
    while i <= menor_cont do
    begin
        mdc:= mdc * 2;
        i:= i + 1;
    end;

    (* repete o processo para todos os impares *)
    primo:= 3;
    while (a <> 1) and (b <> 1) do
    begin
        cont_a:= 0;
        while a mod primo = 0 do
        begin
            cont_a:= cont_a + 1;
            a:= a div primo;
        end;

        cont_b:= 0;
        while b mod primo = 0 do
        begin
            cont_b:= cont_b + 1;
            b:= b div primo;
        end;

        (* descobre qual dos contadores eh o menor *)
        if cont_a <= cont_b then
            menor_cont:= cont_a
        else
            menor_cont:= cont_b;

        (* atualiza o mdc para o primo impar da vez *)
        i:= 1;
        while i <= menor_cont do
        begin
            mdc:= mdc * primo;
            i:= i + 1;
        end;

        (* passa para o proximo impar *)
        primo:= primo + 2;
    end;

    (* imprime o resultado final *)
    writeln (mdc);
end.
```

USANDO FUNÇÕES

```
program mdcpeladefinicao; (* pela definicao de mdc *)
var
   a, b, primo, mdc: longint;
   cont_a, cont_b, menor_cont : integer;

function num_vezes_que_divide(p: integer; var n: longint): integer;
(* codigo da funcao num_vezes_que_divide *)

function potencia(n,p : integer): integer;
(* codigo da funcao potencia *)

begin (* programa principal *)
      read (a,b);
      mdc:= 1;

      cont_a:= num_vezes_que_divide(2,a);
      cont_b:= num_vezes_que_divide(2,b);

      if cont_a <= cont_b then
            menor_cont:= cont_a
      else
            menor_cont:= cont_b;

      mdc:= mdc * potencia(2,menor_cont);
      writeln ('mdc= ',mdc);
      primo:= 3;
      while (a <> 1) and (b <> 1) do
      begin

            writeln (primo);
            cont_a:= num_vezes_que_divide(primo,a);
            cont_b:= num_vezes_que_divide(primo,b);

            if cont_a <= cont_b then
                  menor_cont:= cont_a
            else
                  menor_cont:= cont_b;

            mdc:= mdc * potencia(primo,menor_cont);

            primo:= primo + 2;
      end;
      writeln (mdc);
end.
```

```
function num_vezes_que_divide(p: integer; var n: longint): integer;
var cont: integer;
begin
      (* descobre quantas vezes o 2 divide as duas entradas *)
      cont:= 0;
      while n mod p = 0 do
      begin
            cont:= cont + 1;
            n:= n div p;
      end;
      writeln ('num_vezes_que_divide= ',cont);
      num_vezes_que_divide:= cont;
end;
```

```
function potencia(n,p : integer): integer;
var pot: longint;
    i: integer;
begin
      pot:= 1;
      i:= 1;
      while i <= p do
      begin
            pot:= pot * n;
            i:= i + 1;
      end;
      writeln ('potencia= ',pot);
      potencia:= pot;
end;
```

MDC: MÉTODO DE EUCLIDES

MDC entre 135 e 72 pelo método de Euclides:

$$135 \bmod 72 = 63$$

$$72 \bmod 63 = 9$$

$$63 \bmod \mathbf{9} = \mathbf{0}$$

Problema: Imprimir o Máximo Divisor Comum (MDC) entre dois números dados.

```
program mdcporeuclides;
var a, b, resto: integer;

begin
        read (a,b);
        if (a <> 0) AND (b <> 0)) then
        begin
              resto:= a mod b;
              while resto <> 0 do
              begin
                    a:= b;
                    b:= resto;
                    resto:= a mod b;
              end;
              writeln ('mdc = ', b);
        end
        else
              writeln ('o algoritmo nao funciona para entradas nulas.');
end.
```