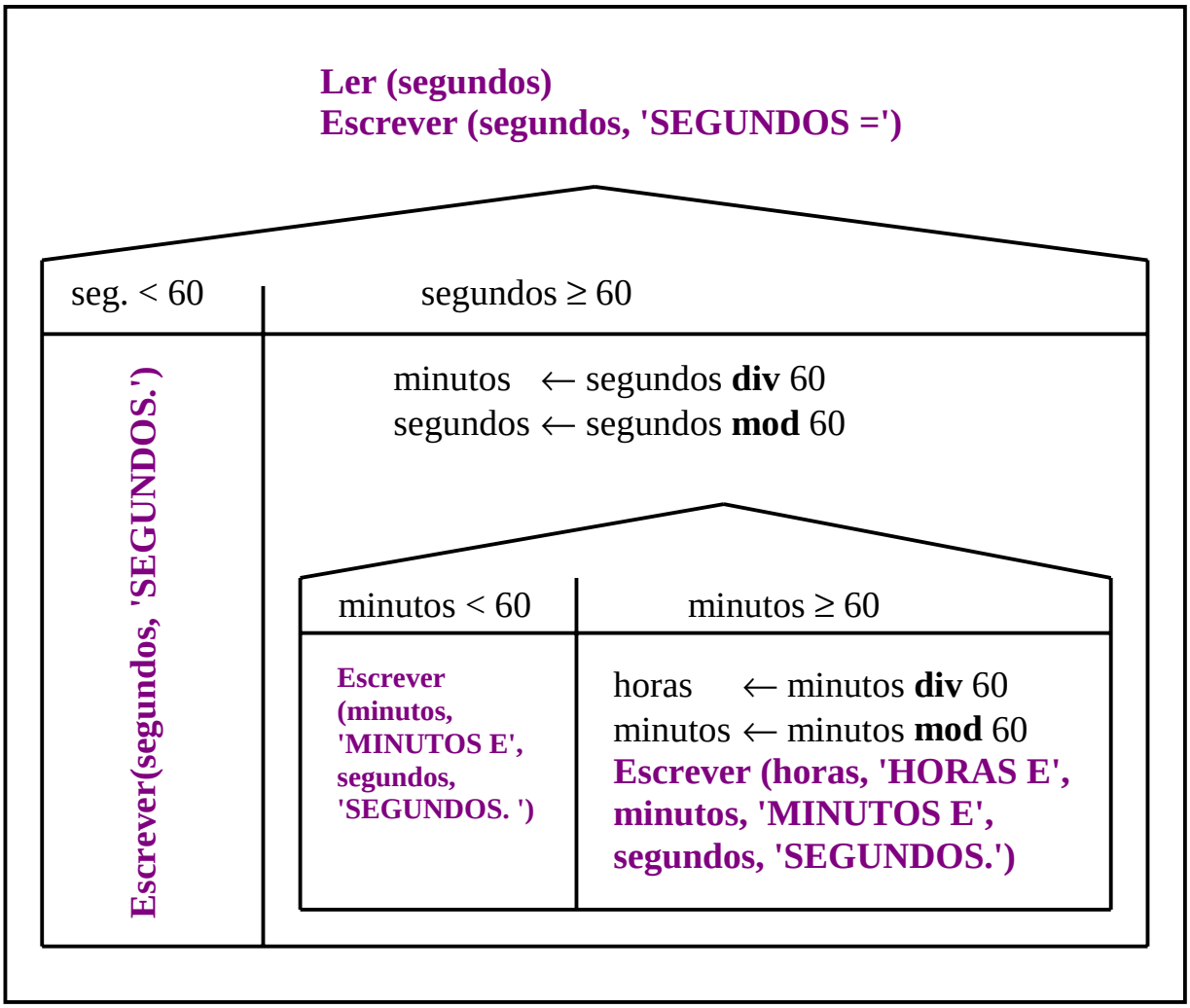
Exercícios sobre o Capítulo III

Problema:

Dado um número inteiro e positivo, representando *segundos*, convertê-lo em *horas*, *minutos* e *segundos*. O resultado deve ser dado numas das formas:

... SEGUNDOS = ... HORAS E ... MINUTOS E ... SEGUNDOS
ou ... SEGUNDOS = ... MINUTOS E ... SEGUNDOS
ou ... SEGUNDOS = ... SEGUNDOS

Diagrama de Estrutura:



Programa:

```
program horas_minutos_segundos (input, output);
type natural = 0..maxint;
var horas, minutos, segundos : natural;

begin  read(segundos);
       write(segundos:6,' SEGUNDOS =');

       if segundos < 60
       then  writeln(segundos:6,' SEGUNDOS .');
       else  begin  minutos:=segundos div 60;
                    segundos:=segundos mod 60;
                    if minutos < 60
                    then writeln(minutos:6,' MINUTOS E',
                                 segundos:6,' SEGUNDOS .')
                    else  begin  horas:= minutos div 60;
                                 minutos:= minutos mod 60;
                                 writeln(horas:6,' HORAS E',
                                      minutos:6,' MINUTOS E',
                                      segundos:6,' SEGUNDOS .')
                          end
             end
end.
```

Problema: Dado: $\mathbf{n} \in \mathbb{N}_0$
Resultado: 2^n

2ª versão do Programa, sem a variável auxiliar i:

```
program potenciade2_versao2 (input, output);
type natural = 0..maxint;
var n, potencia : natural;

begin read(n);
      write('2 elevado a', n:5, ' = ');
      potencia:= 1;
      while  n > 0  do
            begin  n:= n –1;
                  potencia:= potencia * 2
            end;
      (* aqui n = 0 *)
      writeln(potencia:5)
end.
```

$$e^x = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \cdots + \frac{x^n}{n!} + \cdots$$

$$\sin(x) = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \frac{x^9}{9!} - \cdots$$

$$\cos(x) = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \frac{x^6}{6!} + \frac{x^8}{8!} - \cdots$$

Problema: Calcular **sin(x)** e tambem **cos(x)** pelos respectivos desenvolvimentos em **Série de Taylor**, desprezando termos de grandeza inferior a 10^{-8}.

Observações:

Relativamente ao desenvolvimento de $\mathbf{e^x}$:

- **sin(x)** contem os termos de ordem **ímpar**;
- **cos(x)** contem os termos de ordem **par**;
- em ambos os casos, o **sinal** dos termos vai alternando.

Uma solução:

- uma variável lógica para controlar a paridade;
- uma variável inteira para controlar o valor do sinal.

Programa:

```
program seno_e_cosseno (input, output);
const erro = 1.0E-8;
var n, sinal : integer;
    par : boolean;
    x, termo, seno, cosseno : real;

begin write('Escreva um numero real:);
      readln(x);
      (* inicializações *)
      seno:= 0;
      cosseno:= 1;
      n:= 1; par:= false;       (* termo de ordem 1 *)
      termo:= x; sinal:= 1;     (* positivo *)

      while abs(termo) >= erro do
            begin (* decidir onde somar termo válido *)
                  if par
                  then (* somar termo ao cosseno *)
                       cosseno:= cosseno + sinal * termo
                  else begin (* somar termo ao seno *)
                             seno:= seno + sinal * termo;
                             (* e trocar o sinal *)
                             sinal:= -sinal
                       end;

                  (* calcular novo termo *)
                  n:= n + 1;
                  (* e trocar a paridade *)
                  par:= not par;
                  termo:= termo * x / n
            end;

      writeln(' x=', x:9:7, '  seno=', seno:9:7,
                                    '  cosseno=', cosseno:9:7)
end.
```

Alguns resultados:

Para x = π

n	termo	seno	cosseno
1	3.1415927	0.0000000	1.0000000
2	4.9348022	3.1415927	1.0000000
3	5.1677128	3.1415927	-3.9348022
4	4.0587121	-2.0261201	-3.9348022
5	2.5501640	-2.0261201	0.1239099
6	1.3352628	0.5240439	0.1239099
7	0.5992645	0.5240439	-1.2113528
8	0.2353306	-0.0752206	-1.2113528
9	0.0821459	-0.0752206	-0.9760222
10	0.0258069	0.0069253	-0.9760222
11	0.0073704	0.0069253	-1.0018291
12	0.0019296	-0.0004452	-1.0018291
13	0.0004663	-0.0004452	-0.9998995
14	0.0001046	0.0000211	-0.9998995
15	0.0000219	0.0000211	-1.0000042
16	0.0000043	-0.0000008	-1.0000042
17	0.0000008	-0.0000008	-0.9999999
18	0.0000001	0.0000000	-0.9999999
19	0.0000000	0.0000000	-1.0000000

Para x = 0

n	termo	seno	cosseno
1	0.0000000	0.0000000	1.0000000

isto é, o ciclo não chega a ser executado.

Versão “simplificada” do Programa:

```
program seno_e_cosseno (input, output);
const erro = 1.0E-8;
var n, sinal : integer;
    x, termo, seno, cosseno : real;

begin write('Escreva um numero real:);
      readln(x);
      (* inicializações *)
      seno:= 0;
      cosseno:= 1;
      n:= 1;                        (* termo de ordem 1 *)
      termo:= x; sinal:= 1;         (* positivo *)

      while  abs(termo) >= erro  do
            begin  (* somar termo ao seno *)
                   seno:= seno + sinal * termo;

                   (* calcular novo termo *)
                   n:= n + 1;
                   termo:= termo * x / n;
                   (* subtrair termo ao cosseno *)
                   cosseno:= cosseno - sinal * termo

                   (* calcular novo termo *)
                   n:= n + 1;
                   termo:= termo * x / n;
                   (* e trocar o sinal *)
                   sinal:= -sinal
            end;

      writeln(' x=', x:9:7, '  seno=', seno:9:7,
                                    '  cosseno=', cosseno:9:7)
end.
```

Problema: Imprimir a Lista de todos os Números Primos até 1000.

Programa:

```
program ListaPrimosA(output);
const max = 1000;
var   numero, limite, n : 1..max;
      primo : boolean;

begin writeln('Lista dos Números Primos até ', max);
      writeln;
      (* O primeiro Número Primo é 2 *)

      for numero:= 2 to max do
          begin limite:= trunc(sqrt(numero));
                primo:= true;
                n:= 2;

                while primo and (n <= limite) do
                      begin primo:= (numero mod n) <> 0;
                            n:= n+1
                      end;
                (* not primo or (n>limite) *)
                (* primo  ⇒ (n>limite) *)

                if primo
                then writeln(numero)
          end
end.
```

Problema: Imprimir a Lista dos 1000 primeiros Números Primos.

Programa:

```
program ListaPrimosB(output);
const max = 1000;
var   numero, limite, n, conta : 0..max;
      primo : boolean;

begin writeln('Lista dos ', max, ' primeiros Números Primos');
      writeln;
      conta:=0;
      numero:=1;
      (* O primeiro Número Primo é 2 *)

      while  conta < max do
            begin  (* Testar o próximo Número *)
                  numero:= numero+1;
                  limite:= trunc(sqrt(numero));
                  primo:= true;
                  n:= 2;

                  while  primo and (n <= limite) do
                        begin  primo:= (numero mod n) <> 0;
                               n:= n+1
                        end;
                  (* not primo or (n>limite) *)
                  (* primo  ⇒ (n>limite) *)

                  if primo
                  then  begin  writeln(numero);
                               conta:= conta+1
                        end
            end
end.
```

Problema: Listar os **Divisores Próprios** de um dado Número, i.e. excluindo a Unidade e o Número.

Programa:

```
program divisoresA(input, output);
var numero, n, num2 : integer;

begin  writeln('Venha um numero');
       readln(numero);
       (* Cabeçalho da Tabela *)
       writeln('Divisores Proprios de ', numero, ':');

       num2:= numero div 2;

       for  n:=2 to num2 do
            if numero mod n = 0
            then writeln(n)
end.
```

Exemplo:

Divisores Proprios de 36:

2
3
4
6
9
12
18

e se o Número for Primo?

Problema: Decompor um dado Número em todos os **Produtos de Pares** de Divisores Próprios.

Programa:

```
program divisoresB(input, output);
var   numero, n, num2 : integer;
      primo : boolean;

begin  writeln('Venha um numero');
       readln(numero);
       writeln('Produtos dos pares de Divisores Proprios de ',
                                                    numero, ':');
       primo:= true;
       num2:= numero div 2;
       for  n:=2 to num2 do
            if numero mod n = 0
            then  begin  writeln(numero, '=', n, '*', numero div n);
                         primo:= false
                  end;
       if primo
       then writeln(numero, '  E'' um Numero Primo')
end.
```

Exemplos:

Produtos dos pares de Divisores Proprios de 36:

36=2*18
36=3*12
36=4*9
36=6*6
36=9*4
36=12*3
36=18*2

Produtos dos pares de Divisores Proprios de 37:

37 E' um Numero Primo

Problema: Decompor um dado Número nos seus **Divisores Primos**, indicando o grau de divisibilidade de cada um.

Programa:

```
program divisoresC(input, output);
var   numero, n, num2, conta : integer;
      primo : boolean;

begin  writeln('Venha um numero');
       readln(numero);
       writeln('Decomposicao em Factores Primos de ', numero, ':');
       writeln;
       write(numero, ':  ');
       primo:= true;
       num2:= numero div 2;

       for  n:=2 to num2 do

            begin  conta:=0;
                   (* Contagem das divisoes por n *)
                   while  numero mod n = 0 do
                          begin conta:= conta + 1;
                                numero := numero div n
                          end;
                   (* O Numero inicial é destruído *)

                   if conta >= 1
                   then  begin  primo:= false;
                                write(n, '(', conta, ')  ' )
                         end
            end;

       if primo
       then  writeln(numero, '  E" um Numero Primo')
       else  writeln
end.
```

Exemplos:

Decomposicao em Factores Primos de 6:

6: 2(1) 3(1)

Decomposicao em Factores Primos de 36:

36: 2(2) 3(2)

Decomposicao em Factores Primos de 360:

360: 2(3) 3(2) 5(1)

Decomposicao em Factores Primos de 37:

37: E' um Numero Primo

Problemas:

- **Não devia aparecer (1) quando só divide uma vez.**
- **O processo deveria parar quando não existem mais divisores, em vez de continuar sempre até numero div 2.**
- **O ciclo for deve ser substituído por um ciclo while.**

```
program divisoresD(input, output);
var  numero, n, num, num2, conta : integer;
     primo : boolean;

begin writeln('Venha um numero');
      readln(numero);
      writeln('Decomposicao em Factores Primos de ', numero, ':');
      writeln;
      write(numero, ':  ');
      primo:= true;
      num:= numero;
      num2:= numero div 2;
      n:= 2;

      while (num > 1) and (n <= num2) do

            begin conta:=0;
                  (* Contagem das divisoes por n *)
                  while  num mod n = 0 do
                        begin  conta:= conta + 1;
                               num := num div n
                        end;

                  if conta > 0
                  then  begin  write( '   ', n);
                               primo:= false;
                               if conta > 1
                               then write( '(', conta, ')' )
                        end;
                  n:= n + 1
            end;

      if primo
      then  writeln(numero, '  E" um Numero Primo')
      else  writeln
end.
```

Generalização para um dado Intervalo:

2: Numero Primo
3: Numero Primo
4: 2(2)
5: Numero Primo
6: 2 3
7: Numero Primo
8: 2(3)
9: 3(2)
10: 2 5
11: Numero Primo
12: 2(2) 3
13: Numero Primo
14: 2 7
15: 3 5
16: 2(4)
17: Numero Primo
18: 2 3(2)
19: Numero Primo
20: 2(2) 5
21: 3 7
22: 2 11
23: Numero Primo
24: 2(3) 3
25: 5(2)
26: 2 13
27: 3(3)
28: 2(2) 7
29: Numero Primo
30: 2 3 5
31: Numero Primo
32: 2(5)
33: 3 11
34: 2 17
35: 5 7
36: 2(2) 3(2)
37: Numero Primo
38: 2 19
39: 3 13
40: 2(3) 5
41: Numero Primo
42: 2 3 7
43: Numero Primo
44: 2(2) 11
45: 3(2) 5
46: 2 23
47: Numero Primo
48: 2(4) 3
49: 7(2)
50: 2 5(2)
51: 3 17
52: 2(2) 13
53: Numero Primo
54: 2 3(3)
55: 5 11
56: 2(3) 7
57: 3 19
58: 2 29
59: Numero Primo
60: 2(2) 3 5
61: Numero Primo
62: 2 31
63: 3(2) 7
64: 2(6)
65: 5 13
66: 2 3 11
67: Numero Primo
68: 2(2) 17
69: 3 23
70: 2 5 7
71: Numero Primo
72: 2(3) 3(2)
73: Numero Primo
74: 2 37
75: 3 5(2)
76: 2(2) 19
77: 7 11
78: 2 3 13
79: Numero Primo
80: 2(4) 5
81: 3(4)
82: 2 41
83: Numero Primo
84: 2(2) 3 7
85: 5 17
86: 2 43
87: 3 29
88: 2(3) 11
89: Numero Primo
90: 2 3(2) 5
91: 7 13
92: 2(2) 23
93: 3 31
94: 2 47
95: 5 19
96: 2(5) 3
97: Numero Primo
98: 2 7(2)
99: 3(2) 11

```
program divisoresE(input, output);
const max = 100;
var  numero, n, num, num2, conta : integer;
     primo : boolean;

begin writeln('Decomposicoes ate ', max, ':');
      writeln;

      for numero:=2 to max do
          begin write(numero:4, ':  ');
                num:= numero;
                primo:= true;
                num2:= numero div 2;
                n:= 2;

                while  (num > 1) and (n <= num2) do
                       begin conta:=0;
                             while  num mod n = 0 do
                                    begin conta:= conta + 1;
                                          num := num div n
                                    end;

                             if conta > 0
                             then begin  write( '   ', n);
                                         primo:= false;
                                         if conta > 1
                                         then write( '(',conta,')' )
                                  end;
                             n:= n + 1
                       end;

                if primo
                then  writeln(numero, '  Numero Primo')
                else  writeln
          end
end.
```

Problema: Contar os **Divisores Próprios** dos Números de 2 a 500.

Programa:

```
program divisoresF(output);
const max = 500;
var numero, n, num2, conta : integer;

begin  (* Cabeçalho da Tabela *)
       writeln(' Divisores Proprios :');

       for  numero:=2 to max do
            begin  num2:= numero div 2;
                   conta:= 0;
                   for  n:=2 to num2 do
                        if numero mod n = 0
                        then conta:= conta + 1;
                   writeln(numero, conta)
            end
end.
```

Resultados:

```
Divisores Proprios:
   2    0
   3    0
   4    1
   5    0
   6    2
   7    0
   8    2
   9    1
  10    2
  11    0
  12    4
  13    0
  14    2
  15    2
  16    3
...
 485    2
 486   10
 487    0
 488    6
 489    2
 490   10
 491    0
 492   10
 493    2
 494    6
 495   10
 496    8
 497    2
 498    6
 499    0
 500   10
```

Problema: Qual é o Número, entre 2 e 500, que possui mais **Divisores Próprios**?

Programa:

```
program divisoresG(output);
const max = 500;
var   numero, n, num2, conta : integer;
      maxdivisores, maisdivisivel : integer;

begin  maxdivisores:= 0;
       maisdivisivel:= 2;

       for  numero:=3 to max do
            begin  num2:= numero div 2;
                   conta:= 0;
                   for  n:=2 to num2 do
                        if numero mod n = 0
                        then conta:= conta + 1;

                   if conta > maxdividores
                   then  begin  maxdivisores:= conta;
                                maisdivisivel:= numero
                         end
            end;

       writeln('Ganhou o ', maisdivisivel, ' com ',
                                          maxdivisores, ' divisores.')
end.
```

Resultado:

Ganhou o 360 com 22 divisores.

- **Mas será 360 o único número, entre 2 e 500, com 22 divisores?**
- **Se existirem mais, serão maiores ou menores que 360?**