Mestrado Decisão Económica e Empresarial
COMPUTAÇÃO

Programação em VBA.
Variáveis indexadas. Funções e procedimentos.

Exercícios aula 3 Dúvidas ?

1. Escrever um macro que calcula a média de um conjunto de dados armazenado num vector.
2. Escrever um macro para determinar o produto interno de dois vectores.
3. Escrever um macro que efectue uma pesquisa sequencial num vector
4. Programar um algoritmo de ordenação.
5. Escrever um macro que efectue uma pesquisa binária num vector.

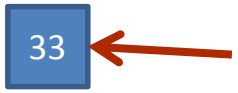
Bubble sort

```
Sub BubbleSort()

   Dim v() As Single, temp As Single
   Dim n As Integer, i As Integer, j As Integer

   n = InputBox("Escreva uma lista de numeros na coluna A. Indique o tamanho da lista:")
   ReDim v(n)

   'Preencher o vector com os dados do Excel
   For i = 0 To n - 1
      v(i) = Cells(i + 1, 1)
   Next i

   For j = 2 To n
      For i = 0 To n - j
         If v(i) > v(i + 1) Then
            temp = v(i + 1)
            v(i + 1) = v(i)
            v(i) = temp
         End If
      Next i
   Next j

   'Escrever no Excel o vector ordenado
   For i = 0 To n - 1
      Cells(i + 1, 2) = v(i)
   Next i

End Sub
```

Variáveis indexadas (relembrar)

- **Array** (vector) **Dim** b(10) **As** Integer

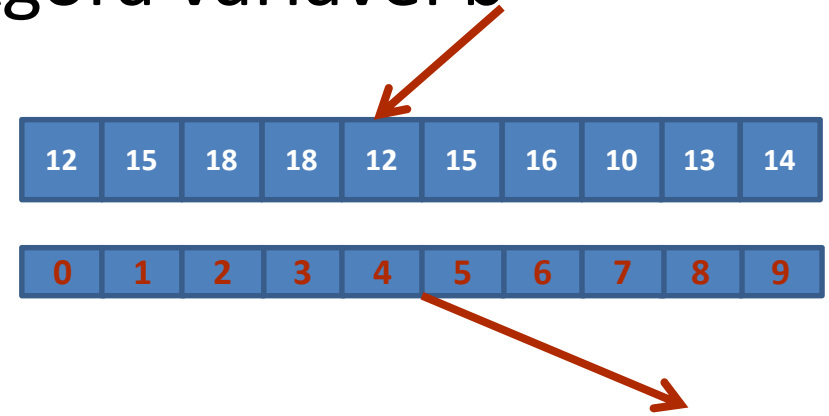
  Permitem armazenar numa variável vários valores desde que sejam todos do mesmo tipo



Agora variável b

b(0)=12

b(1)=15



b(9)=14

Valor do índice deve ser inteiro e por omissão começa em zero

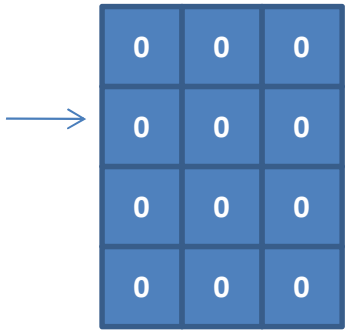
Variáveis indexadas multidimensionais

Dim nome_matriz(num_linhas,num_colunas) **As** Tipo

- **Matrizes**

exemplo

Dim Mat1(4,3) **As** Integer



Mat1=

1	5	7
3	2	4
11	10	6
8	12	13

Mat1(2,2)=6
Mat1(0,0)=1
Mat1(3,2)=13

Variáveis indexadas

- **Manipulação de Matriz ciclos encadeados**

```
Public Sub mat()
   Dim mat(1 To 4, 1 To 4) As Integer
   Dim i As Integer
   Dim j As Integer
   Dim k As Integer
   For i = 1 To 4
      For j = 1 To 4
         mat(i, j) = Cells(i, j)
      Next j
   Next i
   k = 6
   For i = 1 To 4
      For j = 1 To 4
            Cells(i + k, j + k) = mat(i, j)
      Next j
   Next i
End Sub
```

Procedimentos e Funções

- Os procedimentos ou subrotinas destinam-se a realizar um conjunto de tarefas mas não têm necessáriamente que devolver qualquer resultado

- Uma função destina-se a realizar um conjunto de tarefas e a devolver um resultado

Funções

Function Nome (arg1,arg2,…) Tipo de dado

‘ Lista de instruções

Nome=resultado

End Function

Exemplo:

```
Function  desconto(p As Single,d As Single) As single
          desconto= p*(1-d)
End Function
```

Função factorial

```
Private Sub Combina()
  Dim n As Integer
  Dim p As Integer
  Dim Comb As Double
  n = InputBox("no total quantos elementos")
  p = InputBox("grupos de quantos elementos")
  If n > 0 And p > 0 And n >= p Then
    Comb = factorial(n) / factorial(n - p) / factorial(p)
    MsgBox (" num. de combinações " & Comb)
    Else
    MsgBox (" Erro, dados inválidos ")
  End If
End Sub
```

```
Public Function factorial(n As Integer) As Double
    Dim i As Integer, f As Double
    f = 1
    For i = 2 To n
        f = f * i
    Next i
    factorial = f
End Function
```

Procedimentos

Sub Nome (arg1,arg2,…)

‘ Lista de instruções

End Sub

Call Nome()

```
Public Sub exemplo()
   Dim x As Integer
   Dim y As Integer
   x = InputBox("introduza um inteiro")
   y = InputBox("introduza um inteiro")
   MsgBox ("(" & x & "," & y & ")=>" & " Soma:" & x +y)

Public Sub chamada()
  Call exemplo
End Sub
```

Passagem de parâmetros

- **ByVal** O que é passado à função ou ao procedimento é o valor corrente da variável.

  Dentro da função ou procedimento os valores são alterados mas no retorno ao programa os valores são iguais aos que eram na altura da chamada
- **ByRef** O que é passado à função ou ao procedimento é o endereço da variável, uma autorização para alterar o seu valor. Dentro da função ou procedimento os valores são alterados no retorno ao programa ficam alterados de acordo como dentro do bloco

Exemplo de passagem de argumentos

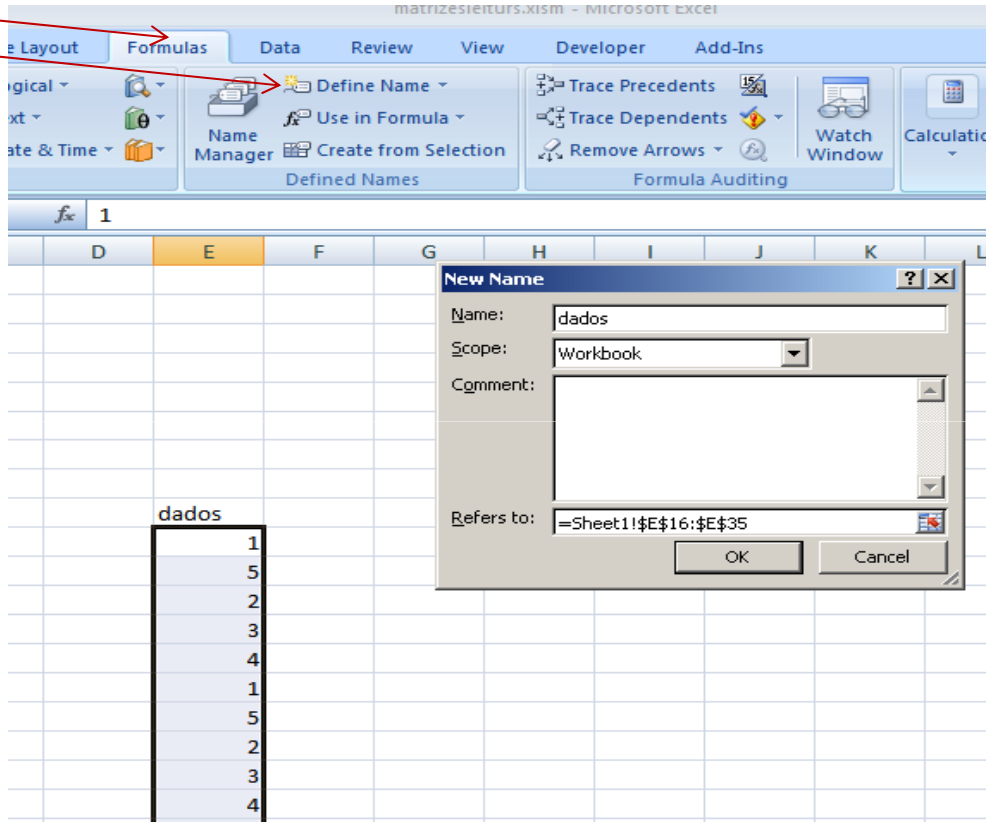
```
Public Sub NãoTroca(ByVal a As Integer, ByVal b As Integer)
Dim temp As Integer
temp = a:a = b:b = temp
End Sub
```

```
Public Sub troca(ByRef a As Integer, ByRef b As Integer)
Dim temp As Integer
temp = a:a = b:b = temp
End Sub
```

```
Public Sub principal()
Dim x As Integer
Dim y As Integer
x = 1:y = 2
MsgBox ("antes de nãoTroca (x,y)=(" & x & "," & y & ")")
Call NãoTroca(x, y)
MsgBox ("depois de nãoTroca (x,y)=(" & x & "," & y & ")")
Call troca(x, y)
MsgBox ("depois de Troca (x,y)=(" & x & "," & y & ")")
End Sub
```

Usar as funções do excel

Definir nome => dados



Med = Application.worksheetFunction.Average(Range(“dados”))

Recursividade

Exemplos

- Factorial
- Números de Fibonacci
- Algoritmo de Euclides
- Torres de Hanoi (dupla)

Factorial

factorial(6)
6 * factorial(5)
6 * 5 * factorial(4)
6 * 5 * 4 * factorial(3)
6 * 5 * 4 * 3 * factorial(2)
6 * 5 * 4 * 3 * 2 * factorial(1)
6 * 5 * 4 * 3 * 2 * 1
6 * 5 * 4 * 3 * 2
6 * 5 * 4 * 6
6 * 5 * 24
6 * 120
720

```
Function factorial(n)
  If n = 1 Then
     factorial = 1
  Else
     factorial = n * factorial(n - 1)
  End If
End Function
```

Algoritmo de Euclides

Dados 2 números inteiros m e n, calcular o seu máximo divisor comum mdc(m,n).

Ler m e n (inteiros diferentes de 0);
Enquanto (n!=0)
Torne resto=m mod n
m=n
n=resto
Escrever mdc(m,n)=m; **STOP**

```
'maximo divisor comum
Function mdc(a, b)
  If a = 0 Then
    mdc = b
  Else
    mdc = mdc(b Mod a,a)
  End If
End Function
```

Fibonacci

Mês 0 No início da experiência existe apenas um par de coelhos.

Mês 1 Após um mês, os coelhos acasalaram mas ainda não deram à luz (portanto existe somente um par de coelhos).

Mês 2 Neste mês já a fêmea deu à luz um par de coelhos. Existem agora dois pares de coelhos.

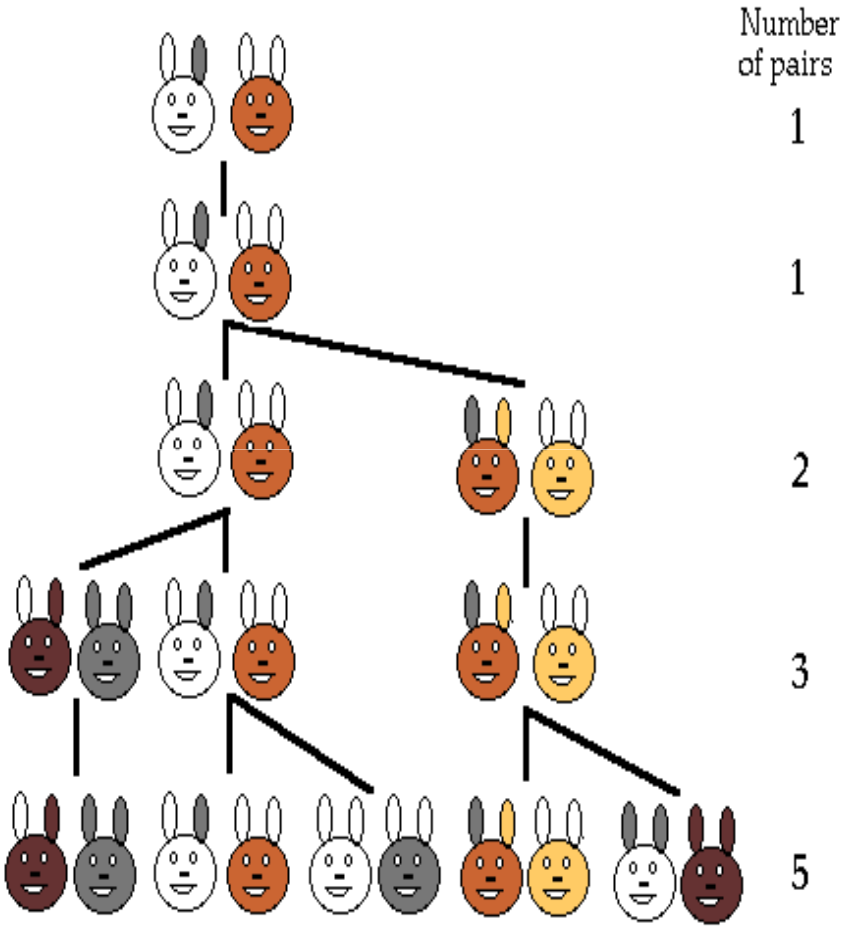
Mês 3 Depois de 3 meses, o par inicial de coelhos dá à luz mais um par de coelhos. No entanto, o segundo par acasala. Isto faz então um total de três pares.

Mês 4 Aos 4 meses, o par original tem mais um par de coelhos. O par nascido no mês 2 também dá à luz. O par de coelhos nascido no mês 3 acasalam,mas ainda não dão à luz. Isto faz um total de cinco pares.

Mês 5 Aos 5 meses, todos os pares que nasceram até há dois meses dão à luz. Isto totaliza oito pares.

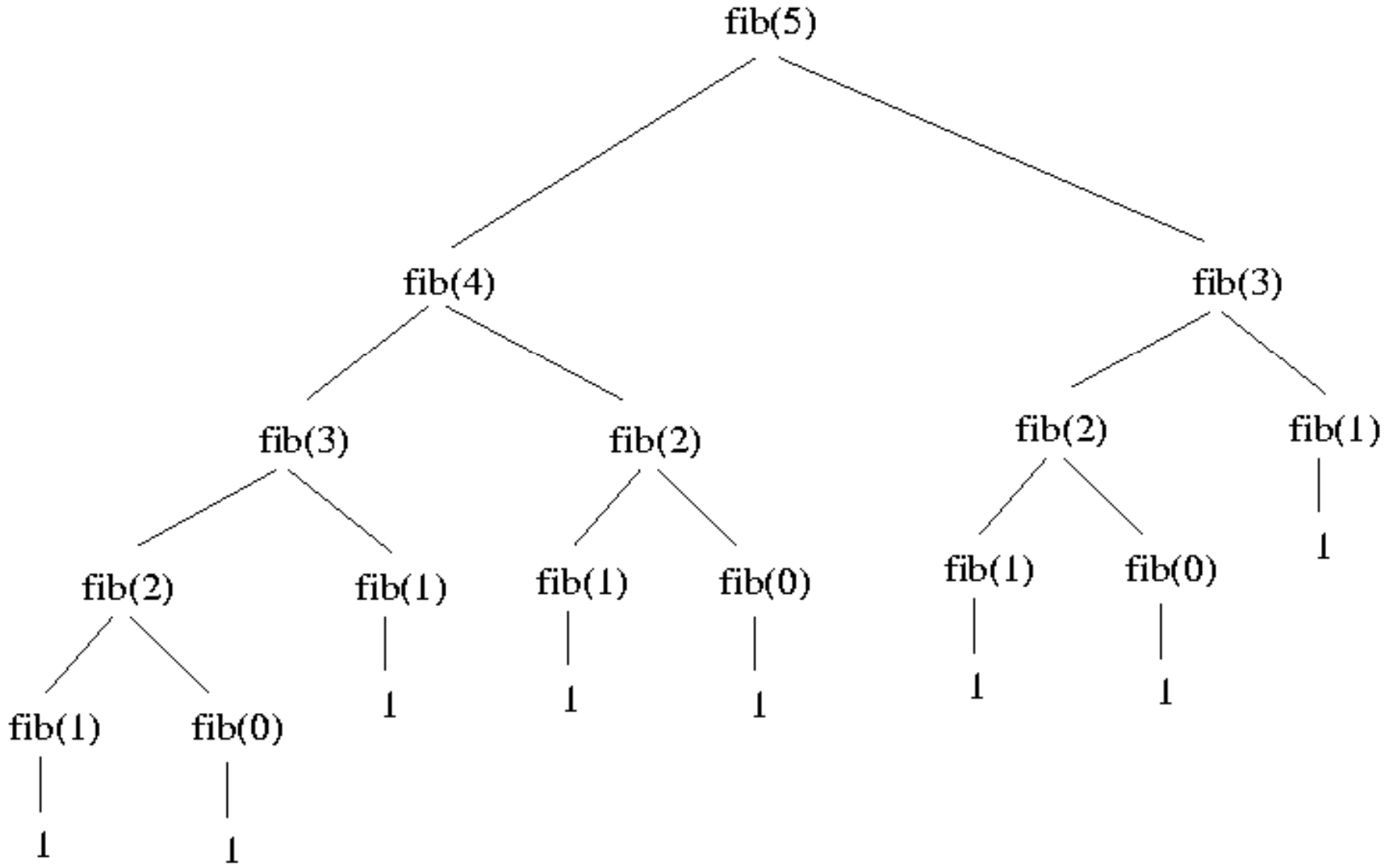
$$fib(0) = 1$$
$$fib(1) = 1$$
$$fib(n) = fib(n-1) + fib(n-2), \text{ para } n > 1$$

Fibbonacci




```
'calcula o n-ésimo número de Fibonacci
Function Fib(n)
  If n = 0 Or n = 1 Then
        Fib = 1
  Else
        Fib = Fib(n - 1) + Fib(n - 2)
  End If
End Function
```

Fibbonacci


fib(5)
fib(4)
fib(3)
fib(3)
fib(2)
fib(2)
fib(1)
fib(2)
fib(1)
fib(1)
fib(0)
fib(1)
fib(0)
1
1
1
1
1
1
1
1
1

Torres de Hanoi

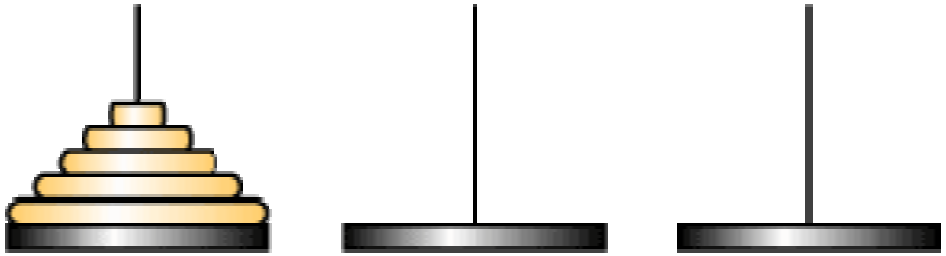
O problema das Torres de Hanói foi inicialmente proposto pelo matemático francês Edouard Lucas, em 1883.

"No grande templo de Brahma em Benares, numa bandeja de metal sob a cúpula que marca o centro do mundo, três agulhas de diamante servem de pilar a sessenta e quatro discos de ouro puro. Incansavelmente, os sacerdotes transferem os discos, um de cada vez, de agulha para agulha, obedecendo sempre à lei imutável de Brahma: Nenhum disco se poderá sobrepor a um menor.
No início do mundo todos os sessenta e quatro discos de ouro, foram dispostos na primeira das três agulhas, constituindo a Torre de Brahma. No momento em que o menor dos discos for colocado de tal modo que se forme uma vez mais a Torre de Brahma numa agulha diferente da inicial, tanto a torre como o templo serão transformados em pó e o ribombar de um trovão assinalará o fim do mundo."

Torres de Hanoi

Restrições a obedecer na movimentação dos discos

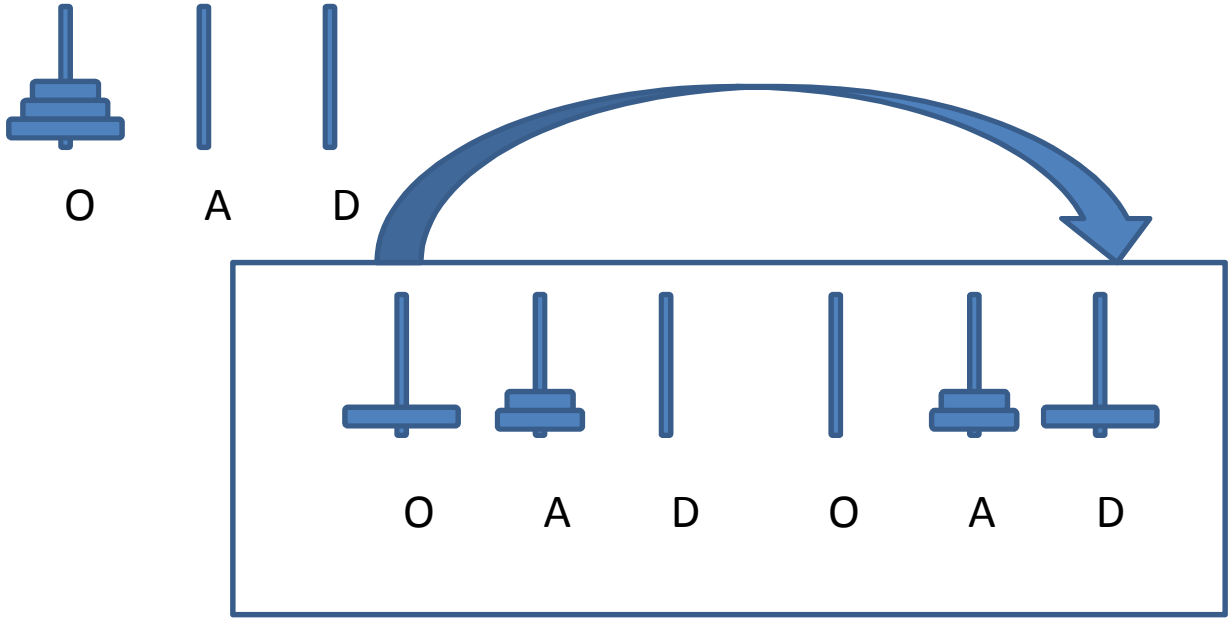
1. apenas se pode mover um único disco por vez;
2. só se podem mover discos que estão no topo;
3. nenhum disco pode ser colocado sobre outro menor;

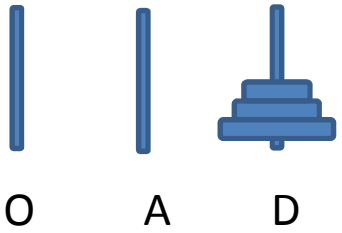


Para mover o disco maior para o poste destino

Todos os discos no poste auxiliar e o poste destino vazio.
=> Transferir todos os outros discos do poste original para o poste auxiliar
o poste Final funciona como auxiliar
Asseguir
=> Transferir todos os outros discos do poste auxiliar para o poste final
o poste Inicial funciona como auxiliar

recursão





Algoritmo

```
Hanoi (n, posteInicial, posteAuxiliar,  posteFinal)
  Se n=1 Então
            MoveDisco(1, posteInicial, posteFinal)

            Senão
            Hanoi (n -1, posteInicial, posteFinal, posteAuxiliar)
            MoveDisco(n, posteInicial, posteFinal)
             Hanoi (n -1 , posteAuxiliar, posteInicial, posteFinal)
```

$$T(n) = \begin{cases} 1 & se \quad n = 1 \\ 2T(n-1)+1 & se \quad n > 1 \end{cases}$$

VBA- Hanoi recursivo

```
Public Sub Hanoi(i As Integer, O As String, A As String, D As String)
If i = 1 Then
   MsgBox ("mover disco " & i & " de " & O & " para " & A)
Else
   Call Hanoi(i - 1, O, D, A)
   MsgBox ("mover disco " & i & " de " & O & " para " & A)
   Call Hanoi(i - 1, D, A, O)
End If

End Sub
```

```
Public Sub jogo()
Dim n As Integer
Dim k As Integer
Dim Origem As String
Dim trabalho As String
Dim destino As String
Origem = "Origem"
trabalho = "Trabalho"
destino = "Destino"
n = 3
Call Hanoi(n, Origem, destino, trabalho)
End Sub
```

Número mínimo de movimentos

Resumindo para n>1 tem-se

$$T(n) = 2^n - 1$$

ou seja, se os monges conseguirem um ritmo de um disco por segundo até que o mundo se desvaneça são necessários 2^{64} segundos, isto é cerca de 584942417 milénios

Podemos estar trânquilos!!!!!