Mestrado Decisão Económica e Empresarial
COMPUTAÇÃO

Programação em VBA.
Variáveis indexadas. Funções e procedimentos.

Exercícios aula 3 Dúvidas ?

1. Escreva um macro que escreve 3 inteiros por ordem decrescente.
2. Programe um macro a determinação do custo de uma encomenda sabendo que o preço base é de 100 e que é efectuado um desconto em função da quantidade de acordo com a seguinte tabela:

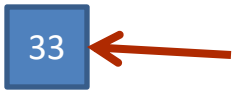
quantidade	desconto
>=150	10%
>= 300	15%
>= 500	20%
>=1000	25%

3. Escreva um macro para determinar se um número é primo.
4. Escreva um macro para determinar o máximo divisor comum.
5. Escreva um macro que informa se um número é par ou ímpar

Variáveis indexadas

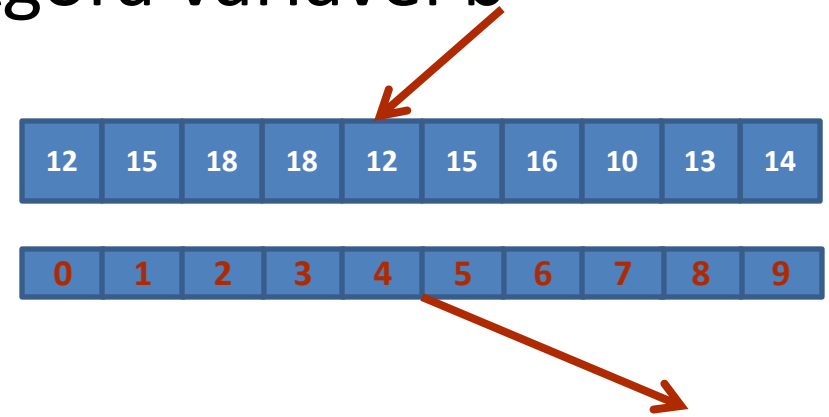
- **Array** (vector)

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



a

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

- **Array** (vector)

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

  Não necessariamente numéricos

  variável nomes

  nomes(0)=“Pedro”

  nomes(1)=“Mariana”

  nomes(6)=“Manuel”

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

Variáveis indexadas

- **Declaração** Estática I

  vector

  Dim nome_do_vector(num_de_elementos) **As** Tipo

  exemplos:

  Dim idade(8) As Integer (de 0 à 7)

  Dim Medias(12) As Double (de 0 à 11)

  Dim nomes(16) As String (de 0 à 15)

  Adicionalmente com Dim os valores numéricos são inicializados a zeros e os alfanuméricos a strings nulas

Variáveis indexadas

- **Declaração** Estática II

  vector

  Dim nome_do_vector(menorIndice **To** maiorIndice)

  As Tipo

  exemplos:

  Dim alunosNovos(100 To 200) As String

  Dim Medias(-100 To 100) As integer

  Adicionalmente com Dim os valores numéricos são inicializados a zeros e os alfanuméricos a strings nulas

Variáveis indexadas

- **Declaração** Dinâmica

  Quando não sabemos a dimensão no vector no momento da sua declaração

  Dim nome_do_vector() **As** Tipo

  Dim dados() As Integer

  Quando é conhecida a dimensão pretendida

  ReDim nome_do_vector()

  ReDim Preserve nome_do_vector()

  para ajustar dimensão sem perder os valores já existentes no vector

Variáveis indexadas

- **Utilidade**

  Exemplo: uma empresa de produtos lacteos tem 100 produtos cada um com o seu preço e pretende actualizar o IVA que era de 6% e passou a ser de 23%.

100 variáveis

Declaração

```
Dim preço1 As Single
Dim preço2 As Single
…
Dim preço100 As Single
Dim actualiza As Single
actualiza =0.23/0.06
```

Actualização

```
preço1 = actualiza*preço1
preço2 = actualiza*preço2
…
preço100 = actualiza*preço100
```

1 variável indexada com 100 elementos

Declaração

```
Dim preços(1 To 100) As Single
Dim actualiza As Single
Dim i As Integer
actualiza =0.23/0.06
```

Actualização

```
For i=1 To 100
        Preços(i) = actualiza*preços(i)
Next i
```

Variáveis indexadas

- **Exemplo 1**

```
Sub Xbarra()
  Dim dados(1 To 10) As Integer
  Dim i As Integer
  Dim Soma As Integer
  Dim Média As Double

  For i = 1 To 10
            dados(i) = ActiveSheet.Cells(i, 1)
  Next i
  Soma = 0
  For i = 1 To 10
            Soma = Soma + dados(i)
  Next i
  MsgBox ("Soma=" & Soma)
  Média = Soma / 10
  MsgBox ("Média=" & Média)
End Sub
```

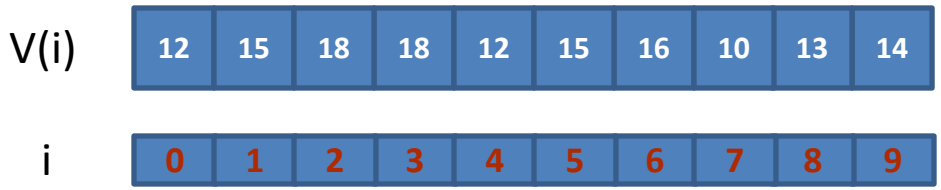
Variáveis indexadas

- **Exemplo 2 (dinâmico)**

```
Sub Xbarra()
  Dim dados() As Integer
  ReDim dados(1 To 10)
  Dim i As Integer
  Dim Soma As Integer
  Dim Média As Double
  For i = 1 To 10:dados(i) = ActiveSheet.Cells(i, 1):Next i
  Soma = 0
  For i = 1 To 10:Soma = Soma + dados(i):Next i
  MsgBox ("Soma=" & Soma)
  Média = Soma / 10
  MsgBox ("Média=" & Média)
  ReDim Preserve dados(1 To 20)
  For i = 11 To 20:dados(i) = ActiveSheet.Cells(i, 1):Next i
  Soma=0
  For i = 1 To 20:Soma = Soma + dados(i):Next i
  MsgBox ("Soma=" & Soma)
  Média = Soma / 20
  MsgBox ("Média=" & Média)
End Sub
```

Algoritmos de pesquisa

- Dado um vector **v** determinar a (1ª) ocorrência (chave/posição) de um determinado elemento.



Pesquisa Sequencial

- Dado um elemento x, percorrer de forma sequencial o vector **v,** comparando cada x a v(i) até encontrar x.

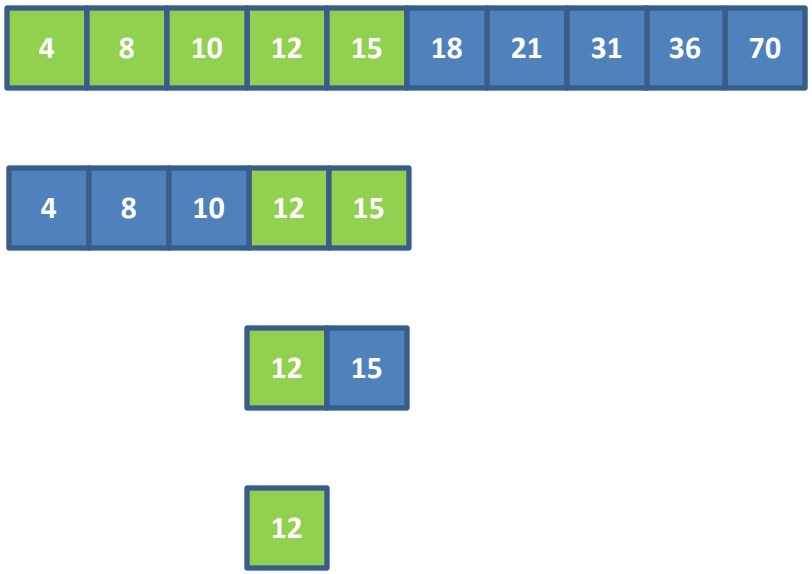
```
Posicao=-1
i=0
Do While  i<=n And Posicao=-1
            If v(i)=x Then
                        Posicao=i
            Else
                        i=i+1
Loop
```

Complexidade O(n)

Pesquisa Binária

- Assume que o vector contém apenas dados numéricos e encontra-se ordenado.
- Divide sucessivamente o vector ao meio até que x seja encontrado.

X=12 V(i)



Pesquisa Binária

```
Esquerda = 0
Direita = n-1
Posicao=-1

Do While Esquerda<=Direita And Posicao=-1
        Meio = (Esquerda + Direita)\2
        If v(Meio)=x Then
                Posicao=Meio
        ElseIf  v(Meio)> x Then
                Direita=Meio-1
        Else
                Esquerda=Meio +1
        End If
Loop
```

Complexidade O(log(n))

Algoritmos de Ordenação

Porquê ordenar vectores?

- Rapidez na pesquisa
- Agrupamento dos dados em classes
- Calculo de estatísticas descritivas: max, min, mediana, etc.
- Algoritmos mais complexos usam algoritmos de ordenação

Algoritmos de Ordenação

Classificação:

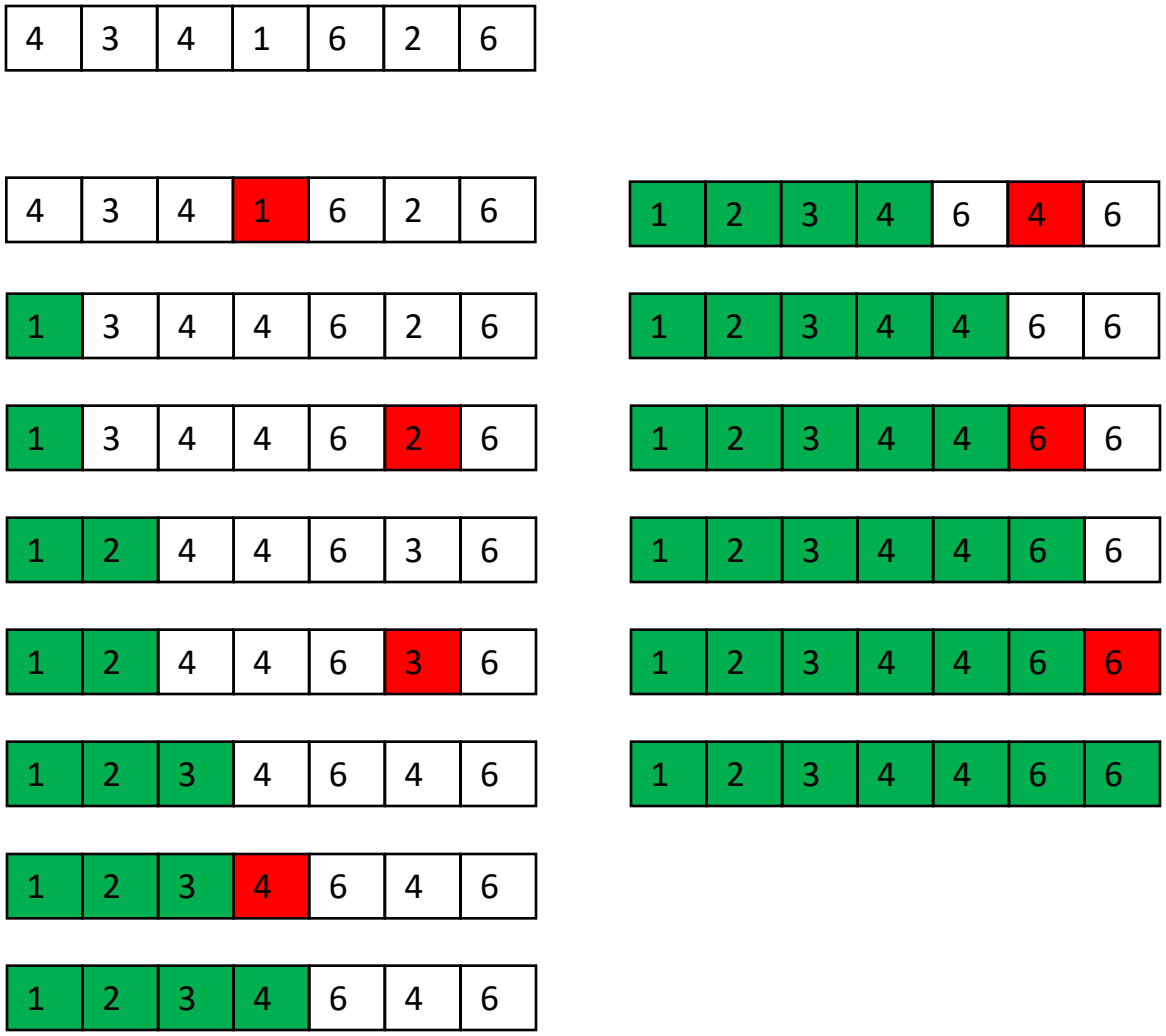
- Complexidade
- Armazenamento de memória
- Estabilidade
- Método de ordenação

Algoritmos de Ordenação

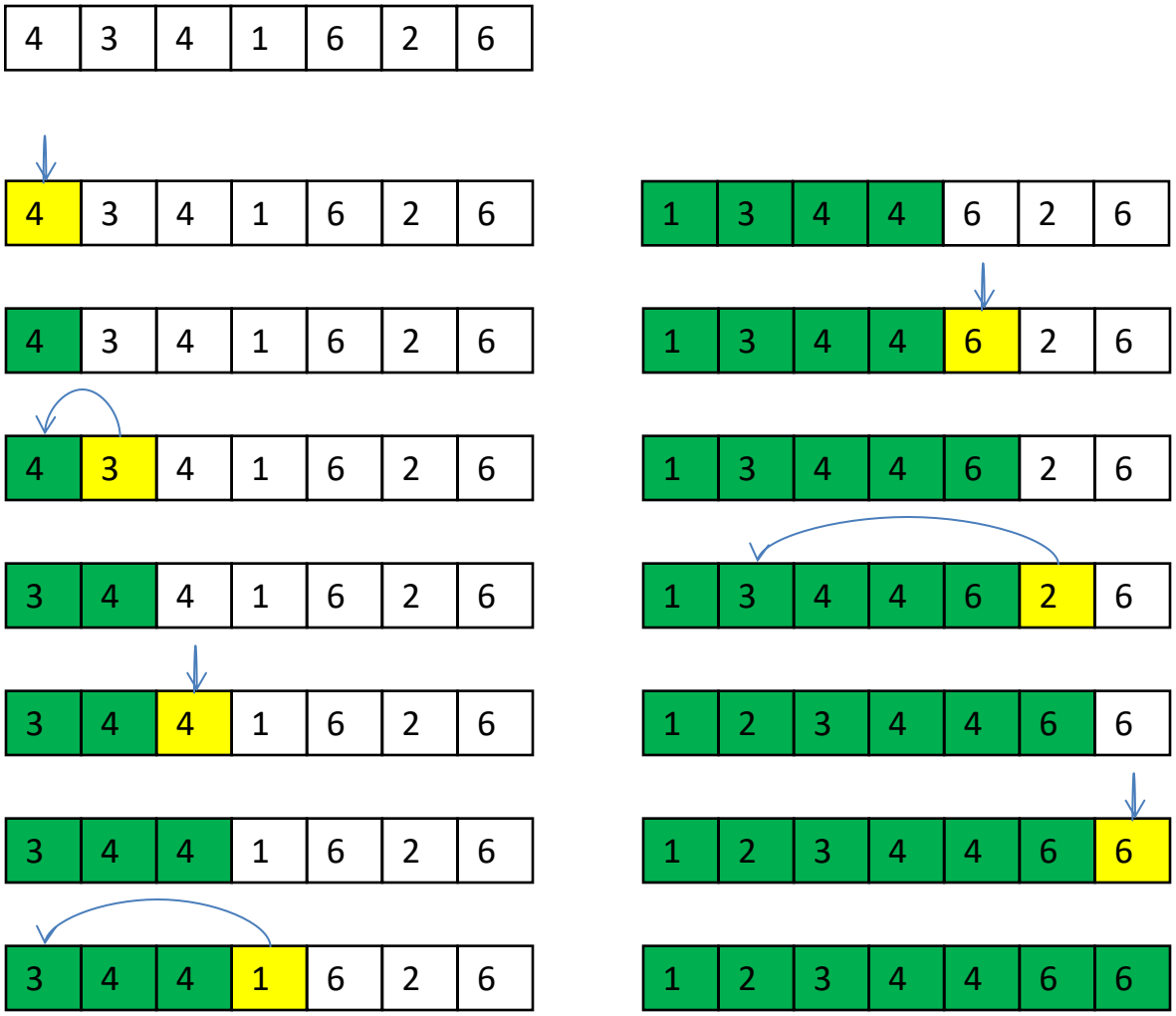
Métodos de ordenação (mais comuns):

- Seleção (Selection Sort)
- Inserção (Insertion Sort)
- Troca (Bubble Sort, Quicksort)

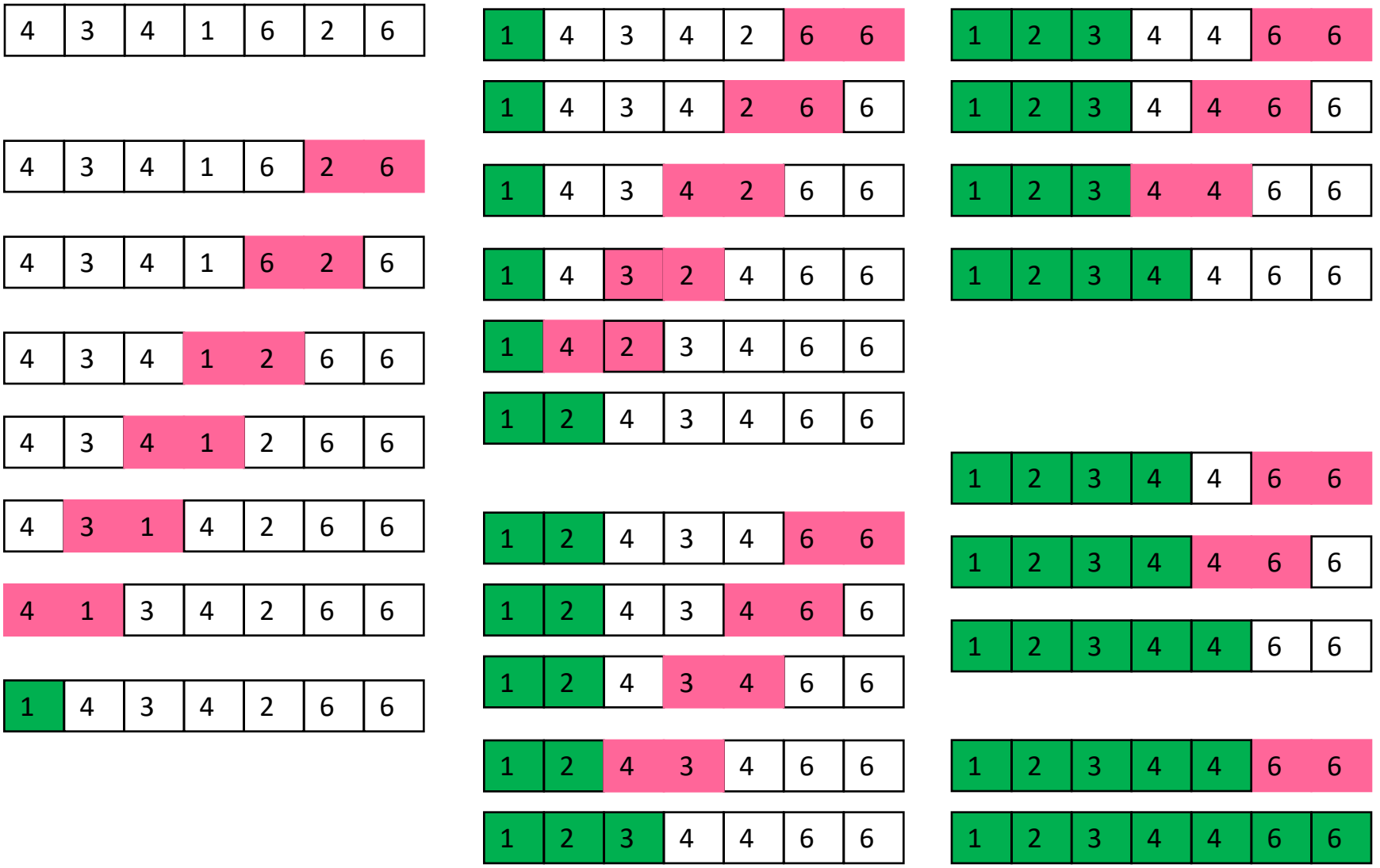
Selection Sort



Insertion Sort



Bubble Sort


4 3 4 1 6 2 6
4 3 4 1 6 2 6
4 3 4 1 6 2 6
4 3 4 1 2 6 6
4 3 4 1 2 6 6
4 3 1 4 2 6 6
4 1 3 4 2 6 6
1 4 3 4 2 6 6
1 4 3 4 2 6 6
1 4 3 4 2 6 6
1 4 3 2 4 6 6
1 4 2 3 4 6 6
1 2 4 3 4 6 6
1 2 4 3 4 6 6
1 2 4 3 4 6 6
1 2 4 3 4 6 6
1 2 3 4 4 6 6
1 2 3 4 4 6 6
1 2 3 4 4 6 6
1 2 3 4 4 6 6
1 2 3 4 4 6 6
1 2 3 4 4 6 6
1 2 3 4 4 6 6

Complexidade

Análise do pior caso:

- No três algoritmos descritos verifica-se que tem O(n^2) complexidade.

- Existem algoritmos mais eficientes com complexidade O(nlog(n)). Exemplos: Mergesort, Heapsort, etc..

- No entanto: Selection sort, Insertion sort e Bubble sort são “rápidos” para vectores de pequena dimensão e fáceis de implementar.

exercícios

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.