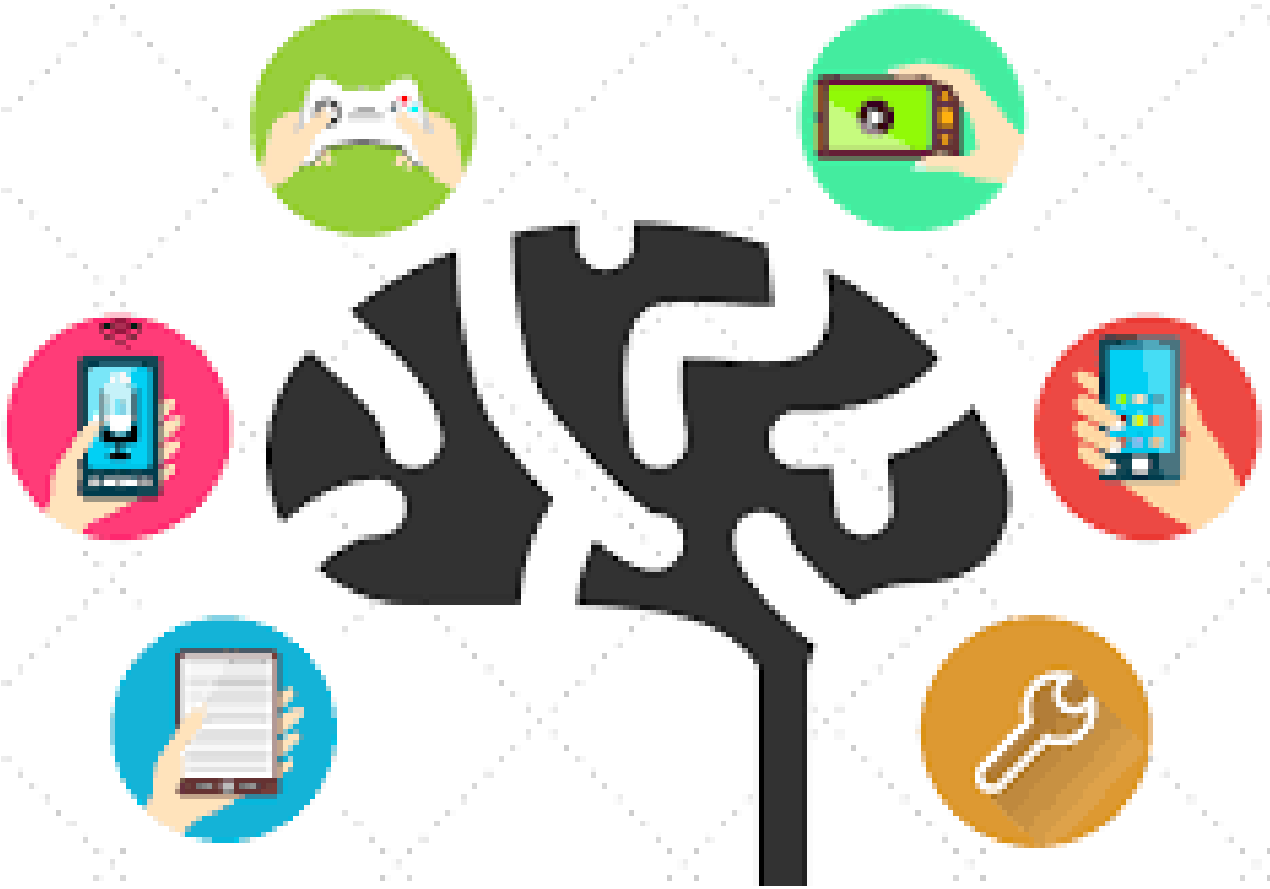
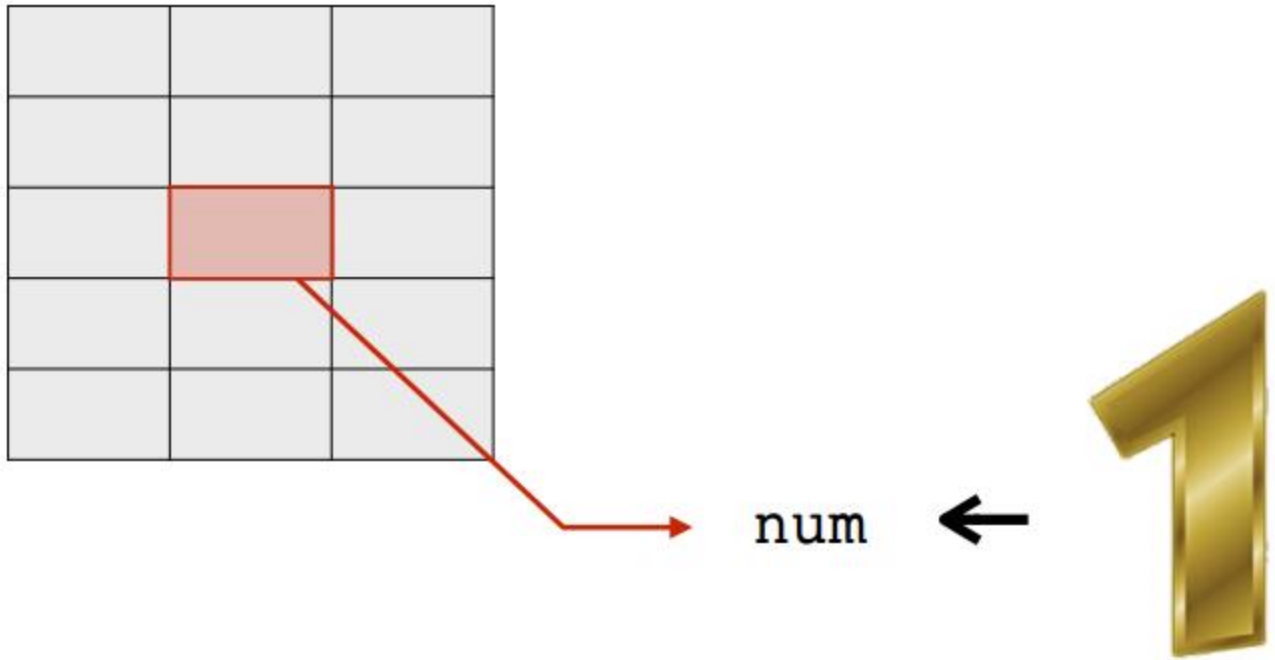


LÓGICA DE PROGRAMAÇÃO
AULA 4

Estruturas de Dados Homogêneas

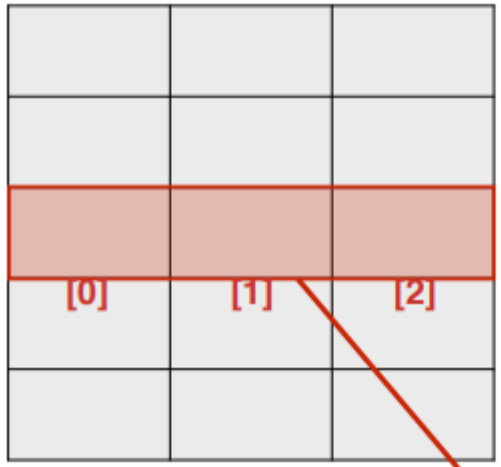
Variáveis Compostas Homogêneas

As **variáveis simples** permitem guardar **apenas um valor** por vez.



Variáveis Compostas Homogêneas

As **variáveis compostas** permitem guardar **vários valores** por vez, um em cada espaço.



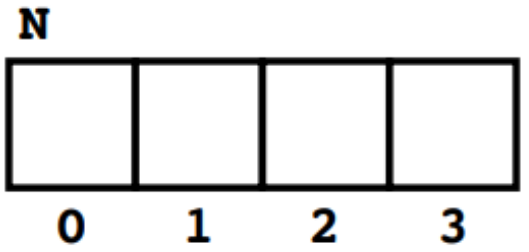
- ✓ **variáveis indexadas**
- ✓ **variáveis subscritas**
- ✓ **arranjos**
- ✓ **vetores**
- ✓ **matrizes**
- ✓ **tabelas em memória**
- ✓ **arrays**

num [1] ← 4

Fundamentos

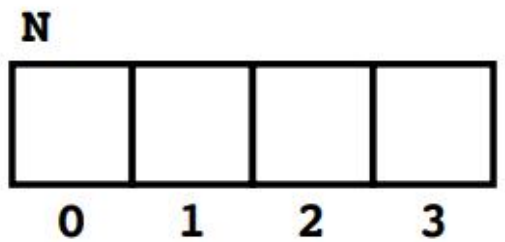
```
Algoritmo Composto
var
  N: vetor[0..3] de Inteiro
inicio
  Escreva("Informe um valor")
  Leia(N[0])
  Escreva("Informe um valor")
  Leia(N[1])
  Escreva("Informe um valor")
  Leia(N[2])
  Escreva("Informe um valor")
  Leia(N[3])
FimAlgoritmo
```

dimensão

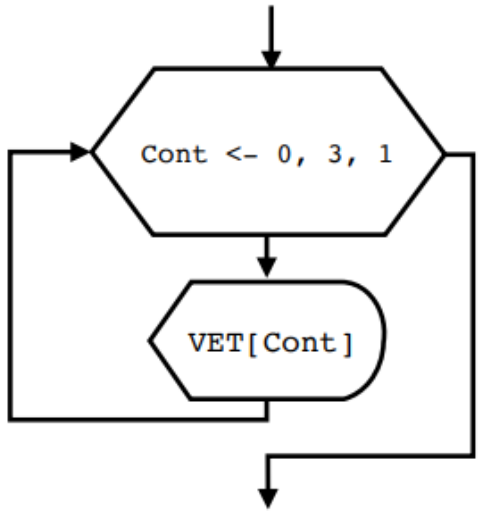


Fundamentos

```
Algoritmo Composto
var
  N: vetor[0..3] de Inteiro
  i: Inteiro
inicio
  Para i <- 0 ate 3 faca
     Escreva("Informe um valor")
     Leia(N[i])
  FimPara
FimAlgoritmo
```



Exibição da Estrutura



VET

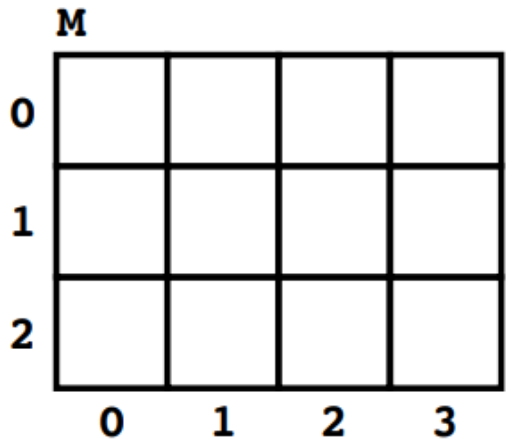
6	4	5	7
0	**1**	**2**	**3**

Variáveis Compostas
Homogêneas Multidimensionais

num [1 , 2] ← 4

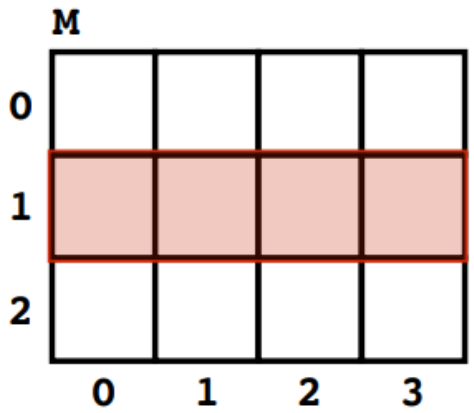
Fundamentos

```
Algoritmo Composto
var
  M: vetor[0..2, 0..3] de Inteiro
  L, C: Inteiro
inicio
  Para L <- 0 ate 2 faca
     Para C <- 0 ate 3 faca
        Escreva("Informe um valor")
        Leia(M[L,C])
     FimPara
  FimPara
FimAlgoritmo
```



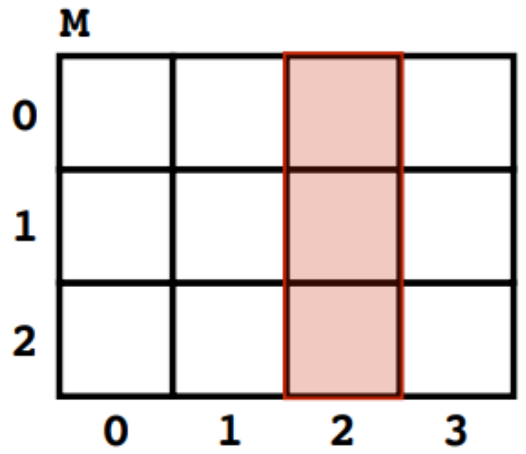
Isolando uma Linha

```
Algoritmo Composto
var
  M: vetor[0..2, 0..3] de Inteiro
  L, C: Inteiro
inicio
  Para C <- 0 ate 3 faca
       Escreva(M[1,C])
  FimPara
FimAlgoritmo
```



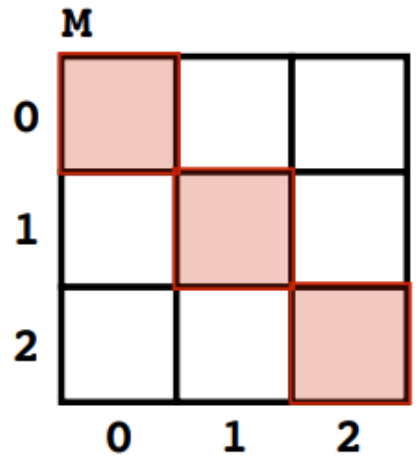
Isolando uma Coluna

```
Algoritmo Composto
var
  M: vetor[0..2, 0..3] de Inteiro
  L, C: Inteiro
inicio
  Para L <- 0 ate 2 faca
       Escreva(M[L,2])
  FimPara
FimAlgoritmo
```



Isolando a Diagonal Principal

```
Algoritmo Composto
var
  M: vetor[0..2, 0..2] de Inteiro
  L, C: Inteiro
inicio
  Para C <- 0 ate 2 faca
       Escreva(M[C,C])
  FimPara
FimAlgoritmo
```



– Considerando a matriz bidimensional TABELA [1..8,1..5], assinale a afirmativa correta.

a) A matriz tem 5 linhas, 8 colunas e 40 elementos.
b) A matriz tem 8 linhas, 5 colunas e no mínimo 40 elementos.
c) A matriz tem 5 linhas, 8 colunas e podem ser armazenados até 40 elementos.
d) A matriz tem 8 linhas, 5 colunas e podem ser armazenados até 40 elementos.

– Dado o seguinte trecho de programa em Português Estruturado,

```
J ← 2
A[1] ← 5
enquanto ( J < 8 ) faça
    A[J] ← A[J – 1] + 3
    J ← J + 1
fim_enquanto
```

assinale a alternativa que contém o valor correto de A[6].

a) 5
b) 8
c) 17
d) 20

– No algoritmo do programa abaixo, em português estruturado, temos as seguintes entradas para os valores da variável X quando requisitados: X[1] = 2, X[2] = 3, X[3] = 8, X[4] = 3. Identifique nas respostas abaixo qual será a saída do programa.

```
programa CONTA

var
        RESULTADO : real
        X : conjunto[1..4] de real
        T : inteiro
        Y : inteiro

início
        Y ← 4
        RESULTADO ← 0
        para T de 1 até 4 passo 1 faça
                leia X[ T ]
                RESULTADO ← RESULTADO + X[ T ]
        fim_para
        RESULTADO ← RESULTADO / Y
        escreva RESULTADO
fim
```

a) 2
b) 8
c) 3
d) 4

Variáveis Compostas Heterogêneas

As **variáveis compostas heterogêneas** ou **registros** permitem definir o **tipo primitivo** de cada posição.

```
Algoritmo Composto
tipo
  Ficha = Registro
             cod: Inteiro
             nome: Caractere
             peso: Real
          FimRegistro
var
  Pessoa: Ficha
inicio
   Leia (Pessoa.cod)
   Leia (Pessoa.nome)
   Leia (Pessoa.peso)
FimAlgoritmo
```

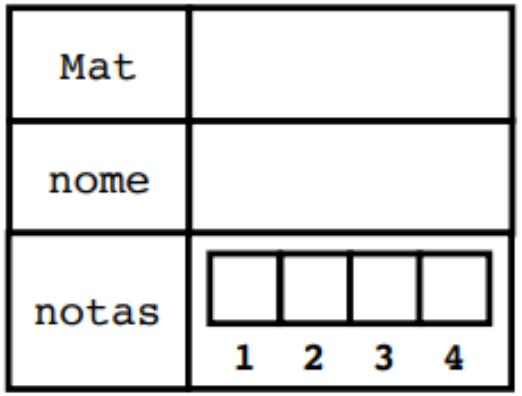
Pessoa

cod	
nome	
peso	

Registros de Vetores

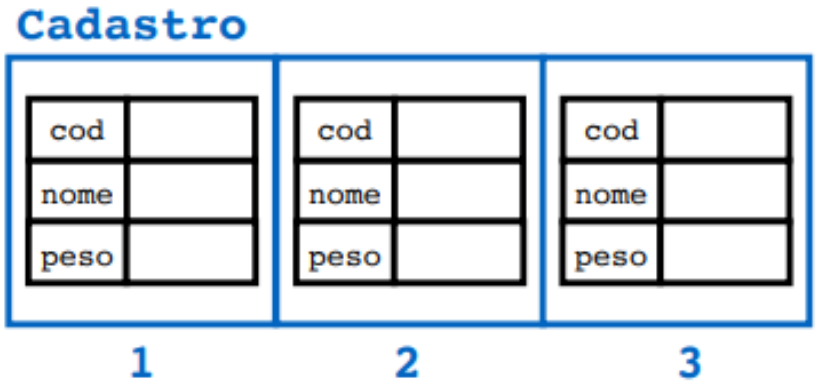
```
Algoritmo Composto
tipo
  Bimestre = vetor[1..4] de Real
  Aluno = Registro
             mat: Inteiro
             nome: Caractere
             notas: Bimestre
          FimRegistro
var
  MeuAluno: Aluno
inicio
   Leia (MeuAluno.nome)
   Leia (MeuAluno.notas[1])
   Leia (MeuAluno.notas[3])
FimAlgoritmo
```

MeuAluno



Vetores de Registros

```
Algoritmo Composto
tipo
  Ficha = Registro
             cod: Inteiro
             nome: Caractere
             peso: Real
          FimRegistro
var
  Cadastro: vetor[1..3] de Ficha
inicio
   Leia (Cadastro[1].nome)
   Leia (Cadastro[2].peso)
FimAlgoritmo
```



Pesquisa Sequencial

Consiste em efetuar a busca da informação desejada **a partir do primeiro elemento** sequencialmente **até o último**.

Localizando a informação no caminho, ela é apresentada. Este método de pesquisa é **lento**, porém **eficiente** caso o vetor encontra-se com seus elementos **desordenados**.

1	5	4	15	25	35	3	18	20	21

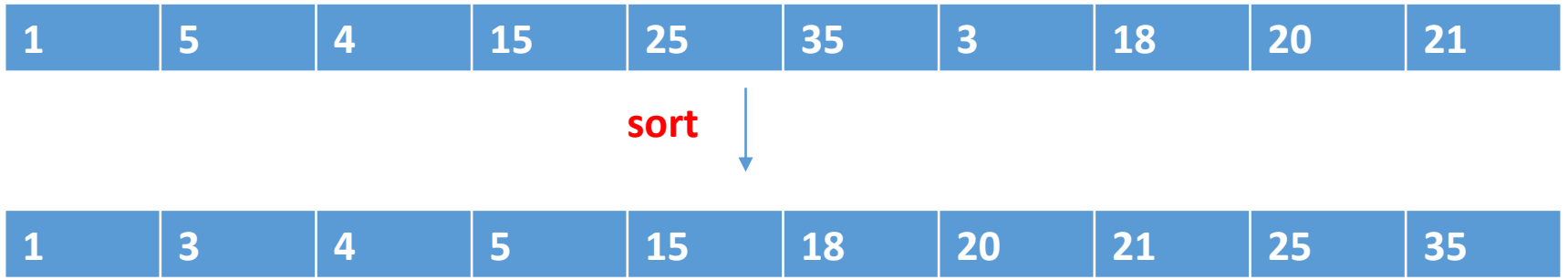
Pesquisa Binária

Divide o vetor em **duas partes** e procura saber se a informação a ser pesquisada está **acima ou abaixo** da linha de divisão.

Se estiver acima, toda a metade abaixo é **desprezada**.

E assim vai sendo executada até **encontrar ou não** a informação pesquisada.

Esse método é em média mais **rápido** que o sequencial, porém exige que a matriz esteja **previamente classificada**.



Método Top-Down

O método mais adequado para a **programação estruturada**.

Antes de iniciar o programa, deverá ter em mente as **tarefas principais** que este deverá executar.

Conhecidas todas as tarefas, imaginar como deverá ser o **programa principal**, que vai controlar todas as outras tarefas.

Tendo definido o programa principal, é iniciado o **processo de detalhamento**. Desta forma são definidos vários algoritmos, um para cada rotina separada.

Método Top-Down

O método **Top-Down** faz com que o programa tenha uma estrutura semelhante a um **organograma**.

A utilização do método **"de cima para baixo"** permite que seja efetuado cada **módulo** do programa em separado.

Outro detalhe a ser considerado é que muitas vezes existem em um programa trechos de códigos que são **repetidos várias vezes**. Esses trechos poderão ser utilizados como sub-rotinas, proporcionando um programa menor e mais fácil de ser alterado num futuro próximo.

Método Top-Down