

# APOI1 – ADS

## Vetores

Prof. SÉRGIO FRANCISCO DA SILVA  
sergio.silva@ifsp.edu.br

# Definição

Um vetor é uma variável composta homogênea unidimensional formada por uma seqüência de variáveis, todas do mesmo tipo, com o mesmo identificador (mesmo nome) e alocadas seqüencialmente na memória. Uma vez que as variáveis têm o mesmo nome, o que as distingue é um índice, que referencia sua localização dentro da estrutura.

# Declaração



**tipo** **nome**[**tamanho**]

onde: **nome** é o nome da variável do tipo vetor, **tamanho** é a quantidade de variáveis que vão compor o vetor e **tipo** é o tipo básico de dados que poderá ser armazenado na sequência de variáveis que formam o vetor.

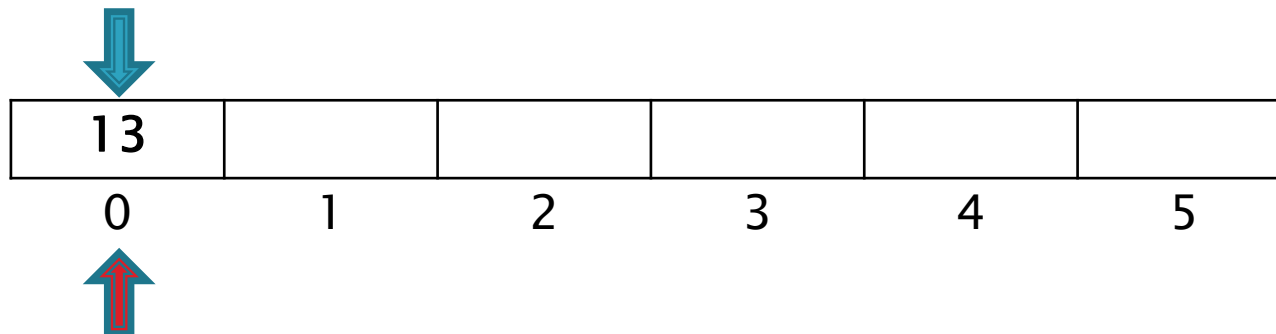
**inteiro** **sorteio**[**6**]





# Atribuição

   
**sorteio**[0] < - 13

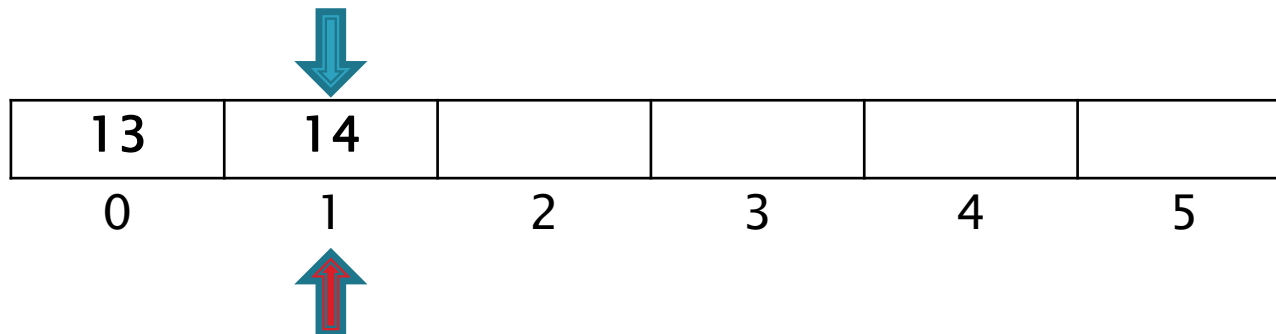
Atribuindo o valor 13 à primeira posição do vetor sorteio, ou seja, a posição de índice 0.





# Atribuição

   
`sorteio[1] < -14`

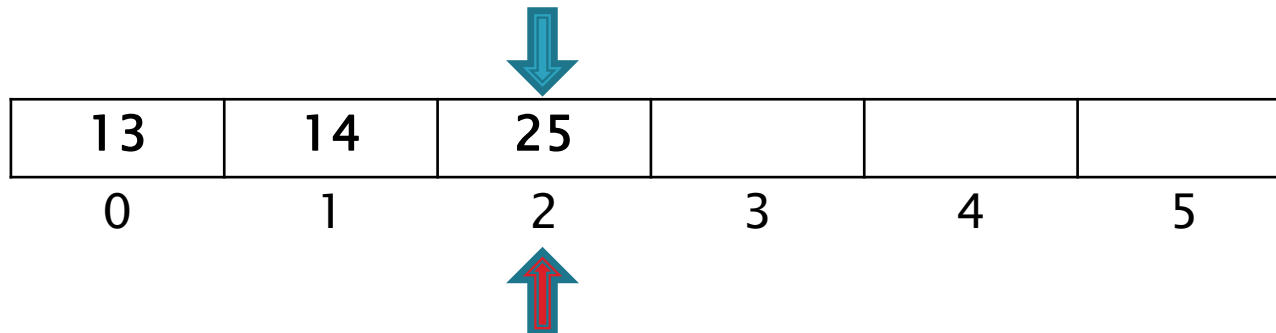
Atribuindo o valor 14 à segunda posição do vetor sorteio, ou seja, a posição de índice 1.





# Atribuição

   
`sorteio[2] <- 25`

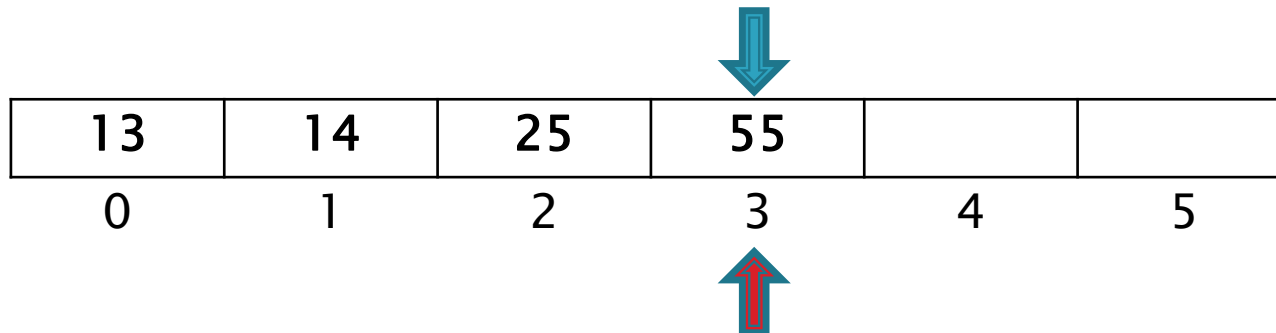
Atribuindo o valor 25 à terceira posição do vetor sorteio, ou seja, a posição de índice 2.



# Atribuição

   
`sorteio[3] <- 55`


Atribuindo o valor 55 à quarta posição do vetor sorteio, ou seja, a posição de índice 3.




# Atribuição

  `sorteio[4] <- 56`

Atribuindo o valor 56 à quinta posição do vetor sorteio, ou seja, a posição de índice 4.





|    |    |    |    |    |   |
|----|----|----|----|----|---|
| 13 | 14 | 25 | 55 | 56 |   |
| 0  | 1  | 2  | 3  | 4  | 5 |







# Atribuição

  `sorteio[5] <- 60`

Atribuindo o valor 60 à sexta posição do vetor sorteio, ou seja, a posição de índice 5.

|    |    |    |    |    |    |
|----|----|----|----|----|----|
| 13 | 14 | 25 | 55 | 56 | 60 |
| 0  | 1  | 2  | 3  | 4  | 5  |



# Inicialização

```
inteiro sorteio[6] <- {13,14,25,55,56,60}
```

Podemos inicializar o vetor atribuindo todos os valores de uma única vez.



# Carregar um vetor

Podemos utilizar uma estrutura de repetição para preencher o vetor.

```
para (x <- 0 até 5 passo 1) faça  
    leia(sorteio[x])  
fimpara
```

# Carregar um vetor

Podemos utilizar uma estrutura de repetição para preencher o vetor.

```
x <- 0
enquanto (x < 6 ) faça
    leia(sorteio[x])
    x <- x + 1
fimenquanto
```

# Recuperar os valores do vetor

Podemos utilizar uma estrutura de repetição para recuperar os valores armazenados em um vetor.

```
para (x <- 0 até 5 passo 1) faça  
    escreva(sorteio[x])  
fimpara
```

# Recuperar os valores do vetor

Podemos utilizar uma estrutura de repetição para recuperar os valores armazenados em um vetor.

```
x <- 0
enquanto (x < 6 ) faça
  escreva(sorteio[x])
  x <- x + 1
fimenquanto
```

# APOI1 – ADS

## Matrizes

Prof. SÉRGIO FRANCISCO DA SILVA  
sergio.silva@ifsp.edu.br

# Declaração




tipo **nome**[linha][coluna]

inteiro **sorteio**[3][6]







# Atribuição




  
**sorteio**[0][0] <- 13

Atribuindo o valor 13 à primeira posição (0,0) da matriz sorteio.





|   | 0  | 1 | 2 | 3 | 4 | 5 |
|---|----|---|---|---|---|---|
| 0 | 13 |   |   |   |   |   |
| 1 |    |   |   |   |   |   |
| 2 |    |   |   |   |   |   |

# Atribuição




  
**sorteio**[0][1] ← 14

Atribuindo o valor 14 à primeira posição (0,1) da matriz sorteio.





|   | 0  | 1  | 2 | 3 | 4 | 5 |
|---|----|----|---|---|---|---|
| 0 | 13 | 14 |   |   |   |   |
| 1 |    |    |   |   |   |   |
| 2 |    |    |   |   |   |   |

# Atribuição




  
**sorteio**[0][2] <- 25

Atribuindo o valor 25 à primeira posição (0,2) da matriz sorteio.




|   | 0  | 1  | 2  | 3 | 4 | 5 |
|---|----|----|----|---|---|---|
| 0 | 13 | 14 | 25 |   |   |   |
| 1 |    |    |    |   |   |   |
| 2 |    |    |    |   |   |   |

# Atribuição


  
**sorteio**[0][3] ← 55

Atribuindo o valor 55 à primeira posição (0,3) da matriz sorteio.




|   | 0  | 1  | 2  | 3  | 4 | 5 |
|---|----|----|----|----|---|---|
| 0 | 13 | 14 | 25 | 55 |   |   |
| 1 |    |    |    |    |   |   |
| 2 |    |    |    |    |   |   |

# Atribuição




  
`sorteio[0][4] <- 56`

Atribuindo o valor 56 à primeira posição (0,4) da matriz sorteio.





|   | 0  | 1  | 2  | 3  | 4  | 5 |
|---|----|----|----|----|----|---|
| 0 | 13 | 14 | 25 | 55 | 56 |   |
| 1 |    |    |    |    |    |   |
| 2 |    |    |    |    |    |   |

# Atribuição


  
**sorteio[0][5] ← 60**

Atribuindo o valor 60 à primeira posição (0,5) da matriz sorteio.




|   | 0  | 1  | 2  | 3  | 4  | 5  |
|---|----|----|----|----|----|----|
| 0 | 13 | 14 | 25 | 55 | 56 | 60 |
| 1 |    |    |    |    |    |    |
| 2 |    |    |    |    |    |    |

# Atribuição




  
`sorteio[1][0] <- 10`

Atribuindo o valor 10 à primeira posição (1,0) da matriz sorteio.




|   | 0  | 1  | 2  | 3  | 4  | 5  |
|---|----|----|----|----|----|----|
| 0 | 13 | 14 | 25 | 55 | 56 | 60 |
| 1 | 10 |    |    |    |    |    |
| 2 |    |    |    |    |    |    |


# Atribuição

    
**sorteio**[2][5] ← 10

Atribuindo o valor 10 à primeira posição (2,5) da matriz sorteio.



|   | 0  | 1  | 2  | 3  | 4  | 5  |
|---|----|----|----|----|----|----|
| 0 | 13 | 14 | 25 | 55 | 56 | 60 |
| 1 | 10 |    |    |    |    |    |
| 2 |    |    |    |    |    | 10 |

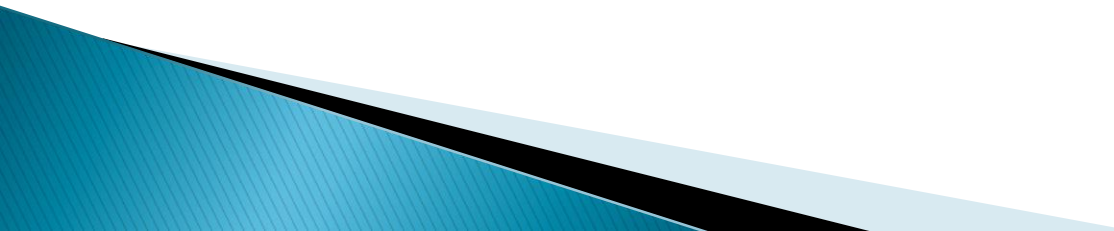




# Carregar uma matriz

Podemos utilizar uma estrutura de repetição para preencher a matriz.

```
para (i <- 0 até 2 passo 1) faça  
  para (j <- 0 até 5 passo 1) faça  
    leia(sorteio[i][j])  
  fimpara  
fimpara
```



# Recuperar uma matriz

Podemos utilizar uma estrutura de repetição para recuperar os valores de uma matriz.

```
para (i <- 0 até 2 passo 1) faça  
  para (j <- 0 até 5 passo 1) faça  
    escreva(sorteio[i][j])  
  fimpara  
fimpara
```

# APOI1 – ADS

## Procedimentos e Funções

Prof. SÉRGIO FRANCISCO DA SILVA  
[sergio.silva@ifsp.edu.br](mailto:sergio.silva@ifsp.edu.br)

# Subprogramas

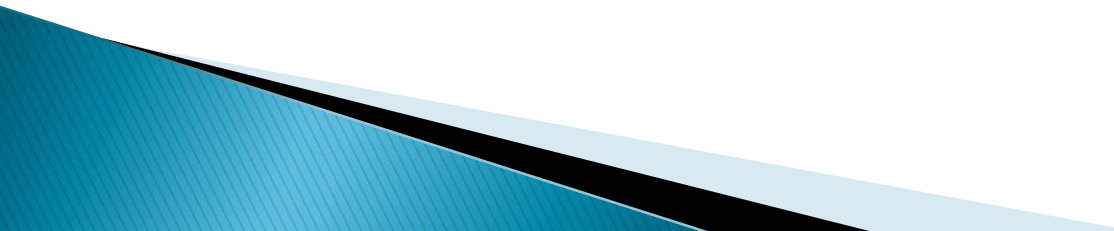
- ▶ *Subprograma* é um programa que auxilia o programa principal através da realização de uma determinada sub-tarefa. Também costuma receber os nomes de *sub-rotina*, *procedimento*, *método* ou *módulo*.
- ▶ Os subprogramas são chamados dentro do corpo do programa principal como se fossem *comandos*. Após seu término, a execução continua a partir do ponto onde foi chamado. É importante compreender que a chamada de um subprograma simplesmente gera um *desvio provisório no fluxo de execução*.

# Procedimento

- ▶ Procedimento é um subprograma que não retorna nenhum valor.

```
procedimento Soma()  
  var  
  inteiro a,b  
  inicio  
    leia(a)  
    leia(b)  
    escreva(a + b)  
fimprocedimento
```

# Função

- ▶ Há um caso particular de subprograma que recebe o nome de *função*.
  - ▶ Uma *função*, além de executar uma determinada tarefa, retorna um valor para quem a chamou, que é o resultado da sua execução.
  - ▶ Por este motivo, a chamada de uma função aparece no corpo do programa principal como uma *expressão*, e não como um comando.
- 

# Função

```
funcao Soma(inteiro x, inteiro y) inteiro  
var  
  inteiro resultado  
inicio  
  resultado <- x + y  
  retorne(resultado)  
fimfuncao
```

# Chamada no programa principal

```
procedimento soma()  
funcao soma(inteiro x, inteiro y) inteiro  
Algoritmo "Soma 2 numeros"  
Var  
    inteiro resultado  
leia(x)  
leia(y)  
resultado <- soma(x,y)      //função soma(x,y)  
escreva(resultado)  
soma()    //procedimento soma()  
fimAlgoritmo
```



# Desafio 1

- 23. Uma empresa possui ônibus com 48 lugares (24 nas janelas e 24 no corredor). Faça um programa que utilize dois vetores para controlar as poltronas ocupadas no corredor e na janela. Considere que zero representa poltrona desocupada e um representa poltrona ocupada.

|          |   |   |   |   |     |    |    |    |
|----------|---|---|---|---|-----|----|----|----|
| Janela   | 0 | 1 | 0 | 0 | ... | 1  | 0  | 0  |
|          | 1 | 2 | 3 | 4 | ... | 21 | 22 | 24 |
| Corredor | 0 | 0 | 0 | 1 | ... | 1  | 0  | 0  |
|          | 1 | 2 | 3 | 4 | ... | 21 | 22 | 24 |

Esse programa deve controlar a venda de passagens da seguinte maneira:

- ◆ o cliente informa se deseja poltrona no corredor ou na janela e, depois, o programa deve informar quais poltronas estão disponíveis para a venda;
- ◆ quando não existirem poltronas livres no corredor, nas janelas ou, ainda, quando o ônibus estiver completamente cheio, deve ser mostrada uma mensagem.