EAD

EDUCAÇÃO A DISTÂNCIA

UNIDADE IV

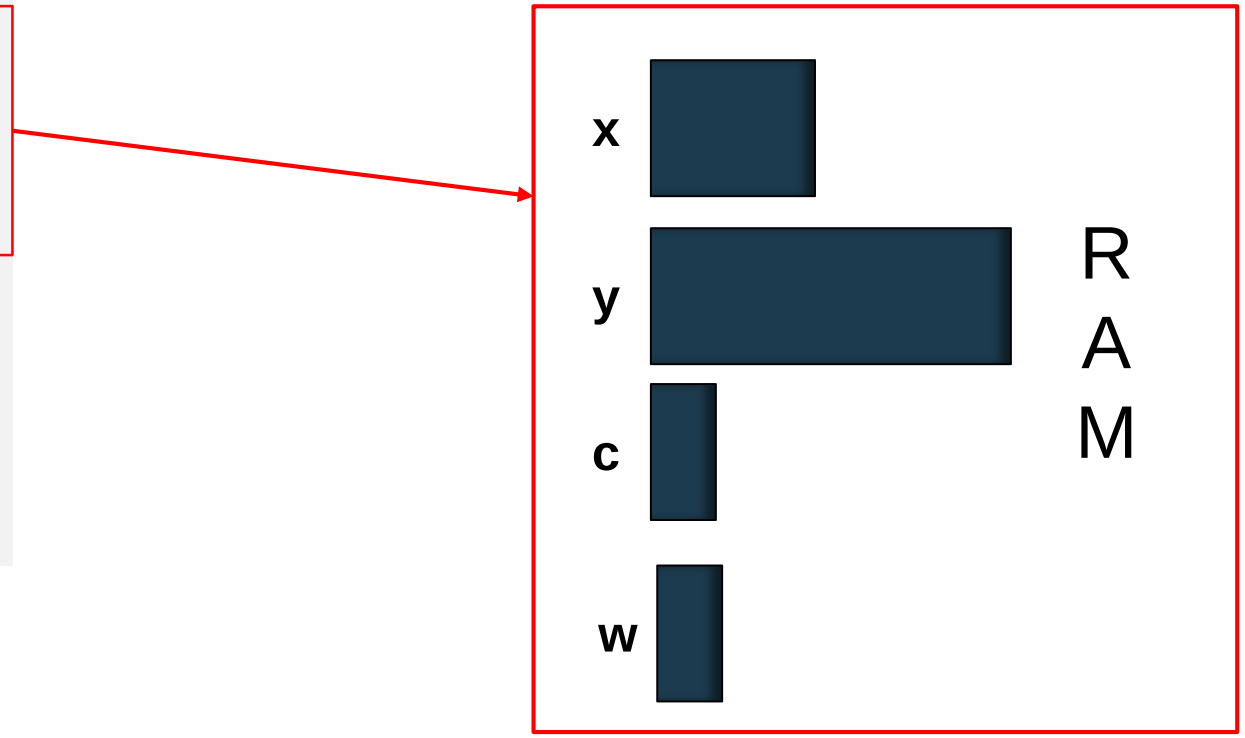
Lógica de Programação e Algoritmos

Profa. Ma. Eliane Santiago

Variáveis

- Variáveis alocam um único endereço de memória.

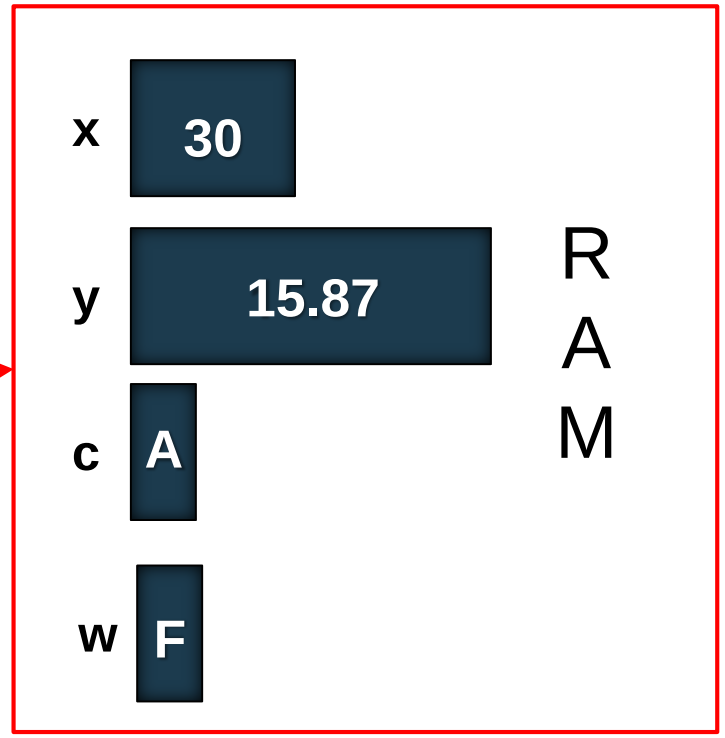
```
1.  Algoritmo " Armazenamento de Dados "
2.  Var //área de declaração de variáveis
3.     x : inteiro
4.     y : real
5.     c : caractere
6.     w : lógico
7.  Inicio
8.     x <- 30
9.     y <- 15.87
10.    c <- “A”
11.    w <- FALSO
12. Fimalgoritmo
```



Variáveis

- Variáveis alocam um único endereço de memória e armazenam um único dado.

```
1.  Algoritmo " Armazenamento de Dados "
2.  Var //área de declaração de variáveis
3.    x : inteiro
4.    y : real
5.    c : caractere
6.    w : lógico
7.  Inicio
8.     x <- 30
9.     y <- 15.87
10.    c <- "A"
11.    w <- FALSO
12. Fimalgoritmo
```



Variáveis simples

- identificador
- endereço de memória
- dado

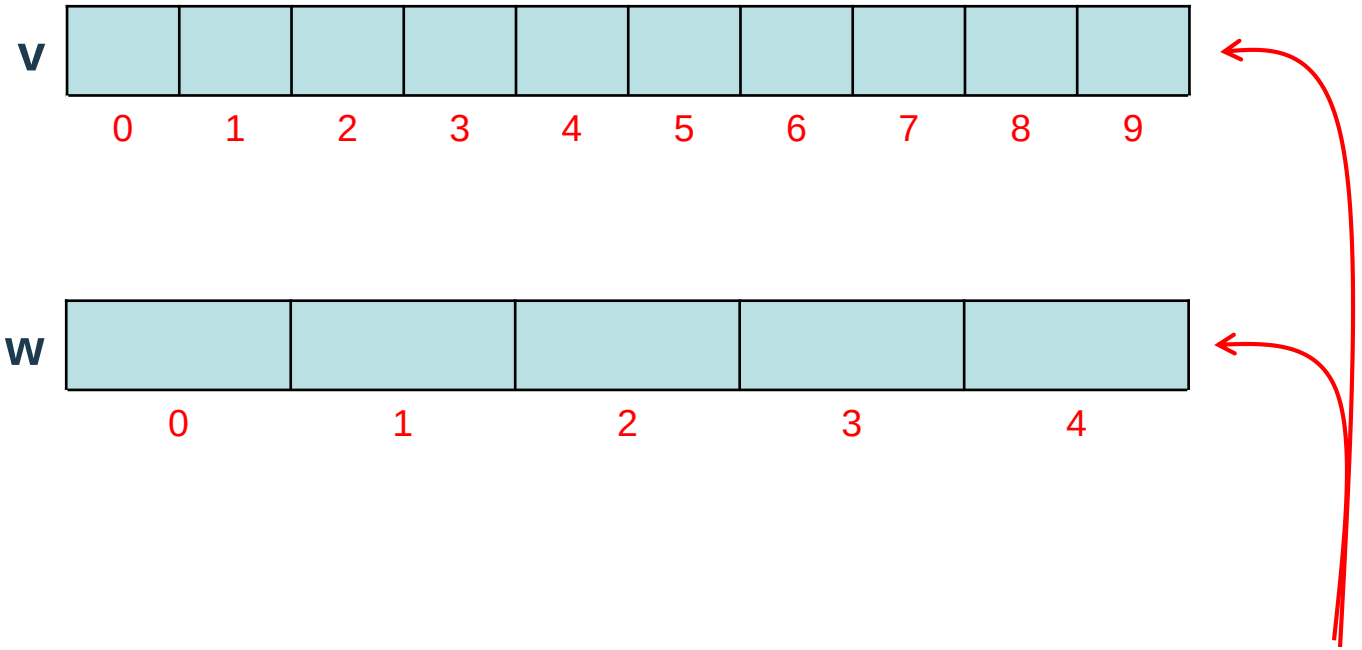
Vetores e Matrizes

- São variáveis compostas porque podem armazenar um conjunto finito de dados do mesmo tipo.
- São estáticas porque requerem especificação do tamanho no momento da declaração e, após alocada a memória, não permite aumentar ou diminuir o tamanho.
- Os dados são indexados.
- Quando se sabe o índice, o acesso ao dado é muito rápido.

Operações básicas com vetores

- Declarar vetores.
- Atribuir um dado a uma determinada posição do vetor.
- Preencher todas as posições de um vetor.
- Mostrar (escrever) todos os elementos do vetor.

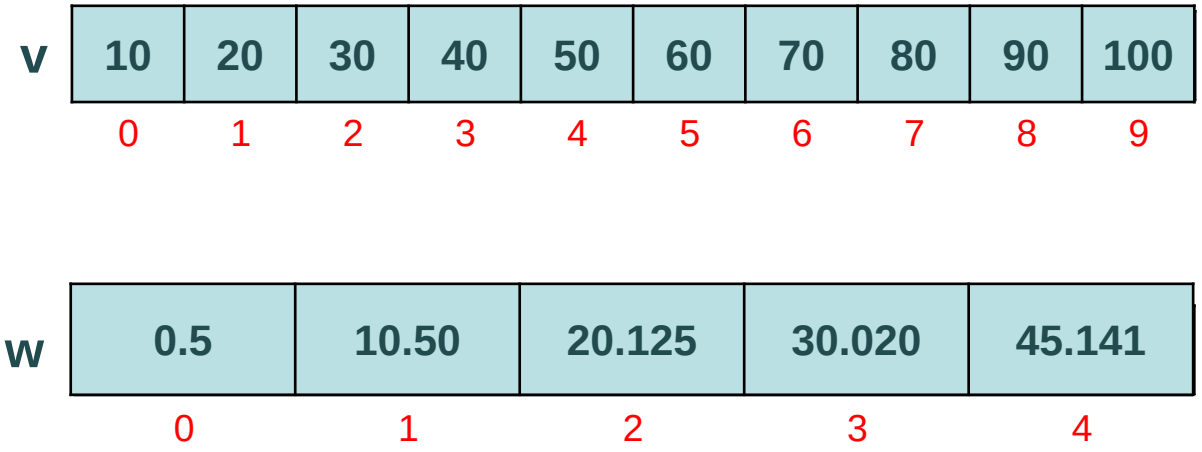
Declaração de vetores



Índices dos vetores

```
1.  Algoritmo " Vetores – Operação 1"
2.  Var //área de declaração de variáveis
3.
4.    v : vetor[0..9] de inteiro
5.    w : vetor[0..4] de real
6.
7.  Inicio
8.
9.
10.
11.
12. Fimalgoritmo
```

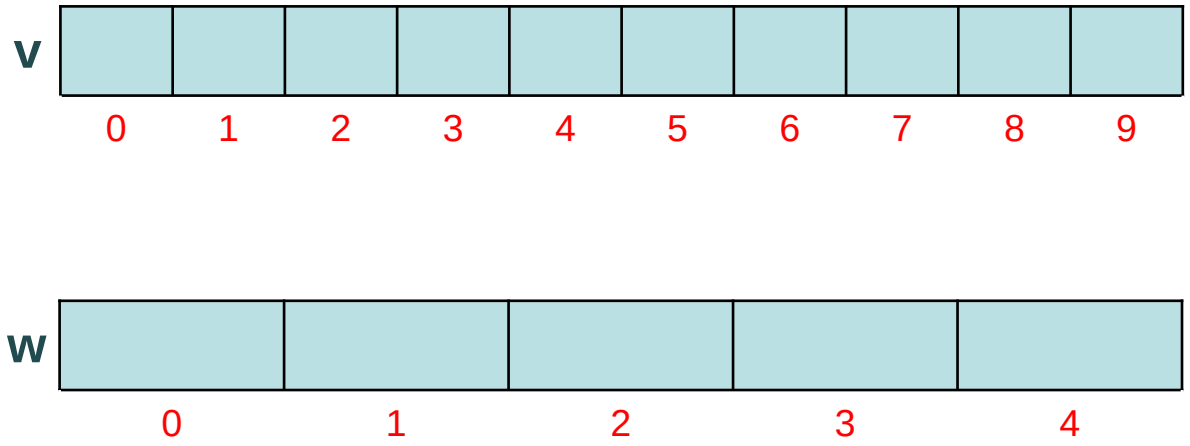
Atribuição de dado a posições específicas do vetor



nome_do_vetor[<posição>] <- <valor>

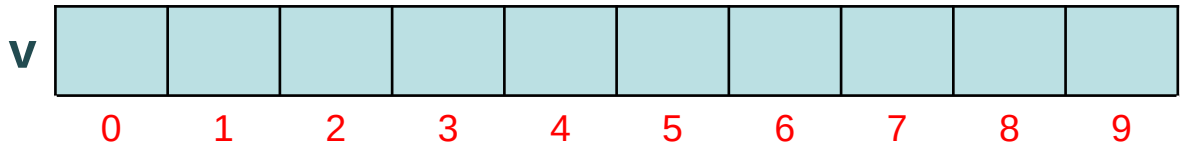
```
1.  Algoritmo " Vetores – Operaçãoo 2"
2.  Var //área de declaração de variáveis
3.
4.     v : vetor[0..9] de inteiro
5.     w : vetor[0..4] de real
6.
7.  Inicio
8.    v[0] <- 10
9.    v[1] <- 20
10.   v[2] <- 30
11.   v[3] <- 40
12.   v[4] <- 50
13.   v[5] <- 60
14.   v[6] <- 70
15.   v[7] <- 80
16.   v[8] <- 90
17.   v[9] <- 100
18.   w[0] <- 0.5
19.   w[1] <- 10.50
20.   w[2] <- 20.125
21.   w[3] <- 30.020
22.   w[4] <- 45.141
23. Fimalgoritmo
```

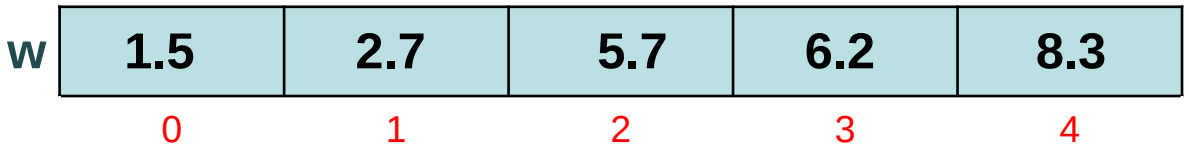
Atribuição em vetores




```
1.  Algoritmo " Vetores – Operação 2"
2.  Var //área de declaração de variáveis
3.    indice : inteiro
4.    v : vetor[0..9] de inteiro
5.    w : vetor[0..4] de real
6.
7.  Inicio
8.    v[0] <- 10
9.    v[1] <- 20
10.   v[2] <- 30
11.   v[3] <- 40
12.   v[4] <- 50
13.   v[5] <- 60
14.   v[6] <- 70
15.   v[7] <- 80
16.   v[8] <- 90
17.   v[9] <- 100
18.   w[0] <- 0.5
19.   w[1] <- 10.50
20.   w[2] <- 20.125
21.   w[3] <- 30.020
22.   w[4] <- 45.141
23. Fimalgoritmo
```

Preencher as posições de um vetor





i	i<=4	v[i]
0	V	w[0] ← 1.5
1	V	w[1] ← 2.7
2	V	w[2] ← 5.7
3	V	w[3] ← 6.2
4	V	w[4] ← 8.3
5	F	

```
1.  Algoritmo " Vetores – Operação 3"
2.  Var //área de declaração de variáveis
3.    i : inteiro
4.    v : vetor[0..9] de inteiro
5.    w : vetor[0..4] de real
6.
7.  Inicio
8.    para i de 0 ate 9 passo 1 faca
9.        escreva("v[", i, "]= ")
10.       leia(v[i])
11.   fimpara
12.
13.
14.
15.   para i de 0 ate 4 passo 1 faca
16.       escreva("w[", i, "]= ")
17.       leia(w[i])
18.   fimpara
19.  Fimalgoritmo
```

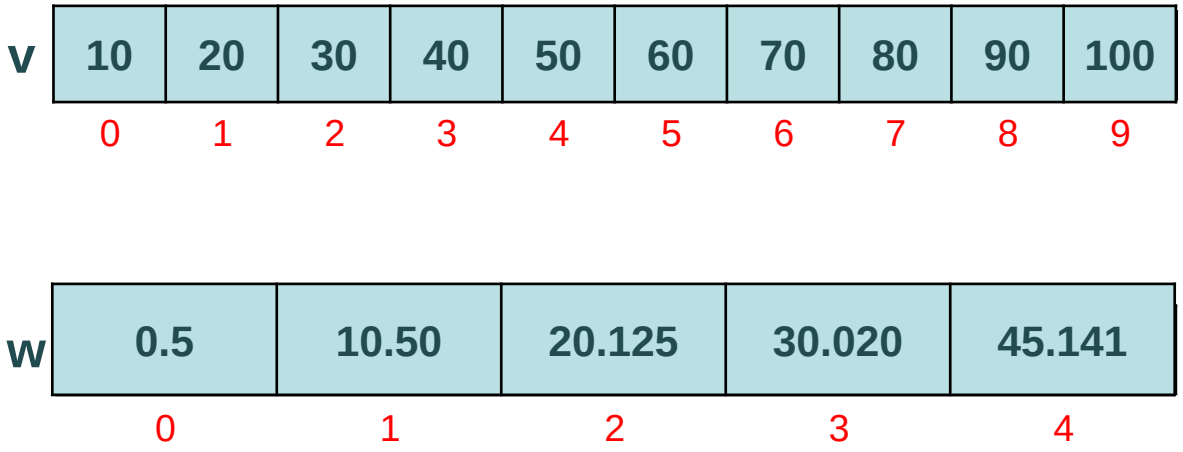
Mostrando os elementos de um vetor

- Os valores são mostrados para cada posição do vetor e a forma de mostrar os elementos na tela é a mesma de qualquer variável ou constante.

```
escreva(<nome do vetor>[<posição>])
escreva("mensagem", <nome do vetor>[<posição>])

escreva("v[",índice, "] = " , v[i])
```

Escrevendo todos os elementos de um vetor




```
1.  Algoritmo " Vetores – Operação 4"
2.  Var //área de declaração de variáveis
3.    indice : inteiro
4.    v : vetor[0..9] de inteiro
5.    w : vetor[0..4] de real
6.
7.  Inicio
8.    para indice de 0 ate 9 passo 1 faca
9.        escreva("v[", indice, "]= ")
10.       leia(v[indice])
11.   fimpara
12.
13.
14.
15.    para indice de 0 ate 4 passo 1 faca
16.        escreva("w[", indice, "]= ")
17.        leia(w[indice])
18.    fimpara
19.  Fimalgoritmo
```

Interatividade

Dado que as médias finais dos 50 alunos da disciplina de LPA foram armazenadas no vetor notas[] após a execução das linhas 6 a 8 do algoritmo dado abaixo. Considerando que a saída esperada após a execução da linha 14 é o cálculo da média geral da turma, o trecho de código que completa o algoritmo e calcula corretamente a média aritmética de todos os elementos do vetor é:

```
1.  Algoritmo " Interatividade "
2.  Var //área de declaração de variáveis
3.    notas[0..49] de real
4.    i, soma, media: real
5.  Inicio
6.   para i de 0 ate 49 passo 1 faca
7.       leia(notas[i])
8.   fimpara
9.   soma<-0
10.  ____________________________________
11.  ____________________________________
12.  ____________________________________
13.  ____________________________________
14.  escreva("A média geral da turma é ", media)
15. Fimalgoritmo
```

Interatividade

```
1.    Algoritmo " Interatividade "
2.    Var //área de declaração de variáveis
3.       notas[0..49] de real
4.       i, soma, media: real
5.    Inicio
6.      para i de 0 ate 49 passo 1 faca
7.          leia(notas[i])
8.      fimpara
9.      soma<-0
10.     ____________________________________
11.     ____________________________________
12.     ____________________________________
13.     ____________________________________
14.     escreva("A média geral da turma é ", media)
15.   Fimalgoritmo
```

a)

```
10.    para i de 0 ate 49 passo 1 faca
11.        soma <- notas[i]
12.    fimpara
13.    media <- soma/49
```

b)

```
10.    para i de 0 ate 49 passo 1 faca
11.        media <- notas[i]/49
12.    fimpara
13.    media <- media/50
```

c)

```
10.    para i de 0 ate 50 passo 1 faca
11.        soma <- notas[i]
12.    fimpara
13.    media <- soma/50
```

d)

```
10.    para i de 0 ate 49 passo 1 faca
11.        notas[i] <- soma
12.    fimpara
13.    media <- soma/49
```

e)

```
10.    para i de 0 ate 49 passo 1 faca
11.        soma <- soma + notas[i]
12.    fimpara
13.    media <- soma/50
```

Resposta

```
1.    Algoritmo " Interatividade "
2.    Var //área de declaração de variáveis
3.       notas[0..49] de real
4.       i, soma, media: real
5.    Inicio
6.      para i de 0 ate 49 passo 1 faca
7.          leia(notas[i])
8.      fimpara
9.      soma<-0
10.     ____________________________________
11.     ____________________________________
12.     ____________________________________
13.     ____________________________________
14.     escreva("A média geral da turma é ", media)
15.   Fimalgoritmo
```

a)

```
10.    para i de 0 ate 49 passo 1 faca
11.        soma <- notas[i]
12.    fimpara
13.    media <- soma/49
```

b)

```
10.    para i de 0 ate 49 passo 1 faca
11.        media <- notas[i]/49
12.    fimpara
13.    media <- media/50
```

c)

```
10.    para i de 0 ate 50 passo 1 faca
11.        soma <- notas[i]
12.    fimpara
13.    media <- soma/50
```

d)

```
10.    para i de 0 ate 49 passo 1 faca
11.        notas[i] <- soma
12.    fimpara
13.    media <- soma/49
```

e)

```
10.    para i de 0 ate 49 passo 1 faca
11.        soma <- soma + notas[i]
12.    fimpara
13.    media <- soma/50
```

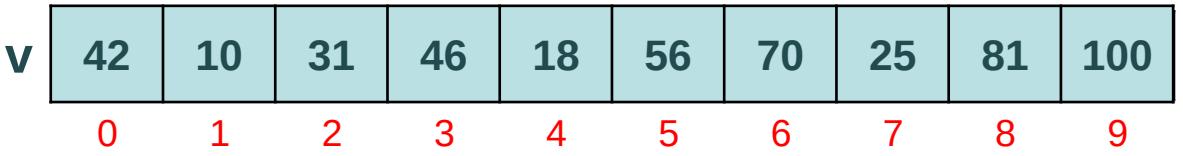
Outras operações com vetores

- Pesquisar um determinado elemento dentro do vetor.
- Concatenar vetores.
- Ordenar os elementos do vetor.

Pesquisar um determinado elemento no vetor

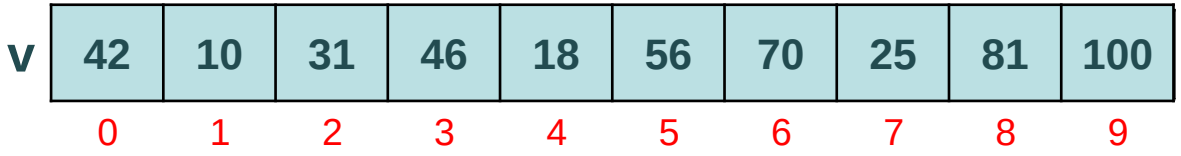
O valor 25 está contido no vetor?

1. Inicializar a variável índice com zero.
2. Leia o vetor da primeira à última posição.
3. Para cada posição verifique
4. se o vetor na posição do índice é igual ao valor, então
5. escreva-o
6. Incremente o índice.
7. Volte à linha 2.




```
1.  Algoritmo " Vetores – Operação 4"
2.  Var //área de declaração de variáveis
3.    indice, valor : inteiro
4.    v : vetor[0..9] de inteiro
5.
6.  Inicio
7.    para indice de 0 ate 9 passo 1 faca
8.        escreva("v[", indice, "]= ")
9.        leia(v[indice])
10.   fimpara
11.
12.   escreva("Qual valor deseja pesquisar? ")
13.   leia(valor)
14.
15.   para indice de 0 ate 9 passo 1 faca
16.       se(v[indice] = valor) entao
17.          escreva("Valor encontrado na posição", indice)
18.   fimpara
19. Fimalgoritmo
```

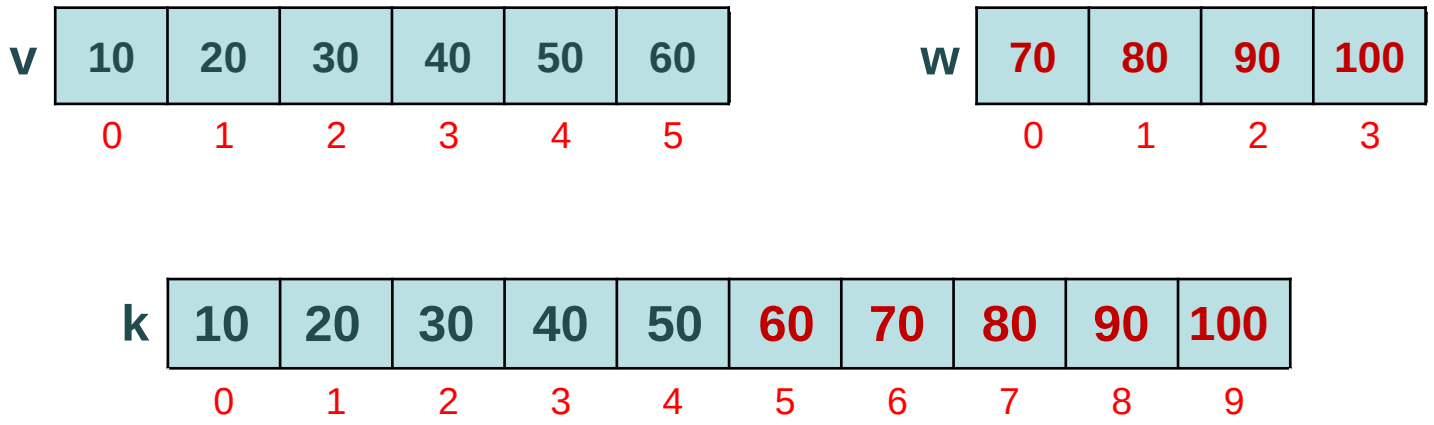
Pesquisar um determinado elemento no vetor




```
1.  Algoritmo " Vetores – Operação 4"
2.  Var //área de declaração de variáveis
3.    indice, valor : inteiro
4.    v : vetor[0..9] de inteiro
5.
6.  Inicio
7.    para indice de 0 ate 9 passo 1 faca
8.        escreva("v[", indice, "]= ")
9.        leia(v[indice])
10.   fimpara
11.
12.   escreva("Qual valor deseja pesquisar? ")
13.   leia(valor)
14.
15.   para indice de 0 ate 9 passo 1 faca
16.       se(v[indice] = valor) entao
17.          escreva("Valor encontrado na posição", indice)
18.   fimpara
19. Fimalgoritmo
```

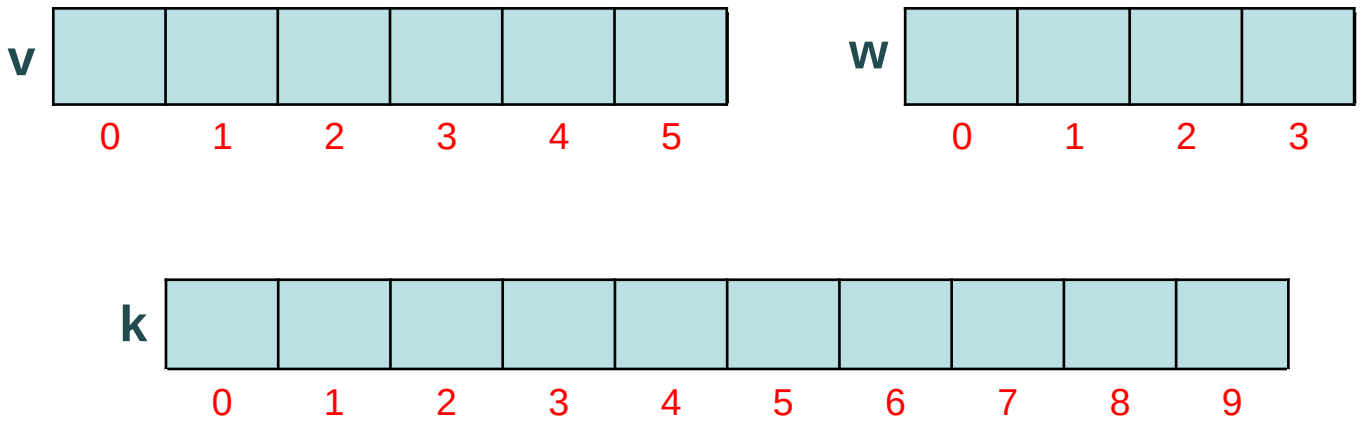
Concatenar dois vetores

- A ideia da concatenação de variáveis é juntar os conteúdos de duas variáveis dentro de uma. Neste exemplo, temos um vetor v de tamanho 6, o vetor w de tamanho 4 e o vetor k de tamanho 10.



Concatenar dois vetores

```
1.  Algoritmo " Concatenar Vetores"
2.  Var //área de declaração de variáveis
3.    i : inteiro
4.    v : vetor[0..5] de inteiro
5.    w : vetor[0..3] de inteiro
6.    k : vetor[0..9] de inteiro
7.  Inicio
8.    aleatorio on
9.    para i de 0 ate 5 passo 1 faca
10.       se (i<4) entao
11.         leia(v[i])
12.         leia(w[i])
13.       senao
14.         leia(v[i])
```




```
15.       fimse
16.    fimpara
17.    aleatorio off
18.    para i de 0 ate 5 passo 1 faca
19.        se (i<=3) então
20.           k[i] <- v[i]
21.           k[i+6] <- w[i]
22.        senao
23.           k[i] <- v[i]
24.        fimse
25.    fimpara
26.  Fimalgoritmo
```

O comando aleatório on/off

```
1.  Algoritmo " Concatenar Vetores"
2.  Var //área de declaração de variáveis
3.    i : inteiro
4.    v : vetor[0..5] de inteiro
5.    w : vetor[0..3] de inteiro
6.    k : vetor[0..9] de inteiro
7.  Inicio
8.    aleatorio on
9.    para i de 0 ate 5 passo 1 faca
10.       se (i<4) entao
11.         leia(v[i])
12.         leia(w[i])
13.       senao
14.         leia(v[i])
15.       fimse
16.   fimpara
17.   aleatorio off
18.   para i de 0 ate 5 passo 1 faca
19.       se (i<=3) então
20.         k[i] <- v[i]
21.         k[i+6] <- w[i]
22.       senao
23.         k[i] <- v[i]
24.       fimse
25.   fimpara
26. Fimalgoritmo
```

- Para não ter que digitar os valores via teclado, poderá acionar o comando aleatório para o preenchimento automático do vetor.

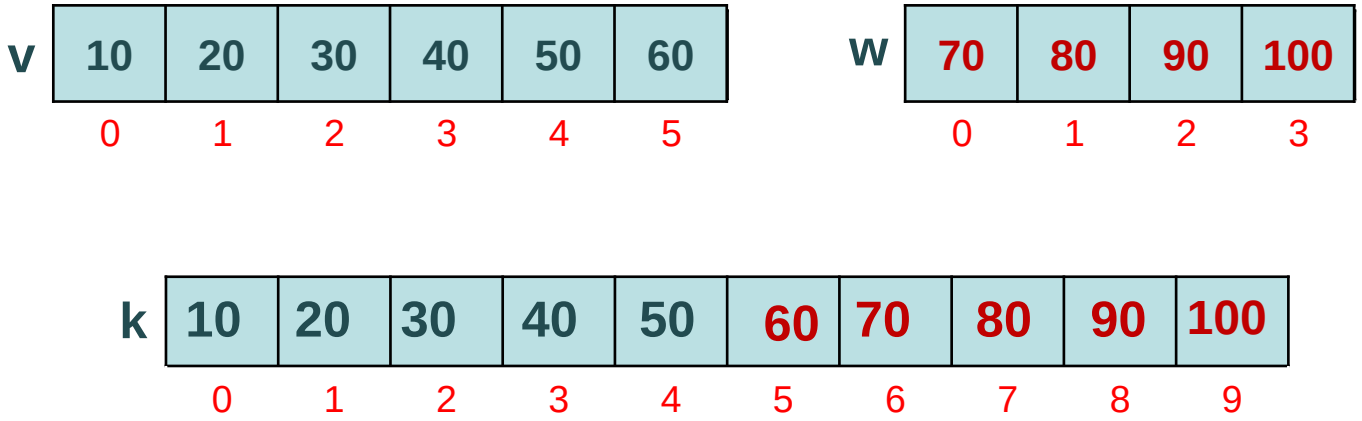
- No bloco entre as linhas 9 e 16, os vetores v[] e w[] estão sendo preenchidos com números aleatórios.

aleatorio on

aleatorio off

Concatenar dois vetores

```
1.  Algoritmo " Concatenar Vetores"
2.  Var //área de declaração de variáveis
3.    i : inteiro
4.    v : vetor[0..5] de inteiro
5.    w : vetor[0..3] de inteiro
6.    k : vetor[0..9] de inteiro
7.  Inicio
8.    aleatorio on
9.    para i de 0 ate 5 passo 1 faca
10.       se (i<4) entao
11.         leia(v[i])
12.         leia(w[i])
13.       senao
14.         leia(v[i])
15.       fimse
16.    fimpara
17.    aleatorio off
18.    para i de 0 ate 5 passo 1 faca
19.        se (i<=3) então
20.           k[i] <- v[i]
21.           k[i+6] <- w[i]
22.        senao
23.           k[i] <- v[i]
24.        fimse
25.    fimpara
26.  Fimalgoritmo
```



Interatividade

Considerando o procedimento p() dado abaixo e as variáveis globais, responda qual alternativa apresenta o propósito do algoritmo.

a) Classifica os dados do vetor em ordem crescente.
b) Classifica os dados do vetor em ordem decrescente.
c) Verifica qual é o maior elemento do vetor.
d) Movimenta os dados do vetor em uma posição.
e) Sobrescreve os dados do vetor em uma posição.

```
Var
    A : vetor[0..9] de inteiro
    i, j, aux : inteiro
```

```
1.  procedimento p()
2.  inicio
3.     para j de 0 ate 9 passo 1 faca
4.        para i de 1 ate 8 passo 1 faca
5.           se (A[i] > A[i+1]) entao
6.               aux <- A[i]
7.               A[i] <- A[i+1]
8.               A[i+1] <- aux
9.           fimse
10.       fimpara
11.    fimpara
12. fimprocedimento
```

Resposta

Considerando o procedimento p() dado abaixo e as variáveis globais, responda qual alternativa apresenta o propósito do algoritmo.

a) Classifica os dados do vetor em ordem crescente.
b) Classifica os dados do vetor em ordem decrescente.
c) Verifica qual é o maior elemento do vetor.
d) Movimenta os dados do vetor em uma posição.
e) Sobrescreve os dados do vetor em uma posição.

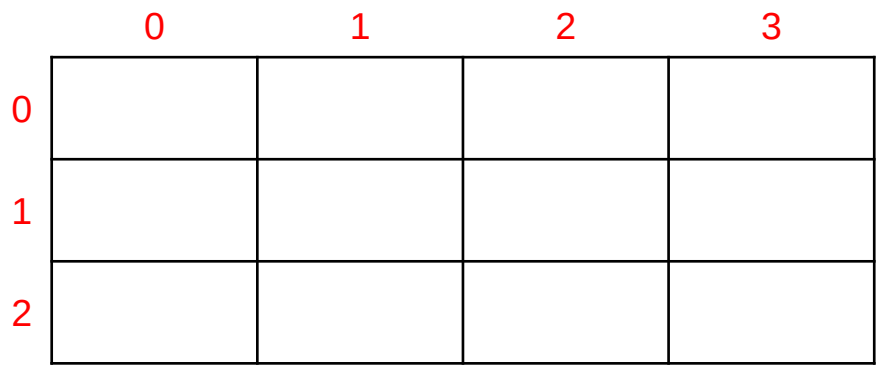
```
Var
    A : vetor[0..9] de inteiro
    i, j, aux : inteiro
```

```
1.  procedimento p()
2.  inicio
3.     para j de 0 ate 9 passo 1 faca
4.        para i de 1 ate 8 passo 1 faca
5.           se (A[i] > A[i+1]) entao
6.               aux <- A[i]
7.               A[i] <- A[i+1]
8.               A[i+1] <- aux
9.           fimse
10.       fimpara
11.    fimpara
12. fimprocedimento
```

Matriz

- Uma matriz é uma alocação de memória com um identificador único, indexada em termos de linhas e colunas, daí o conceito de estrutura de dados composta, homogênea e bidimensional, cujos valores devem ser específicos.

- Exemplo de uma matriz com 3 linhas e 4 colunas.



Operações básicas com matrizes

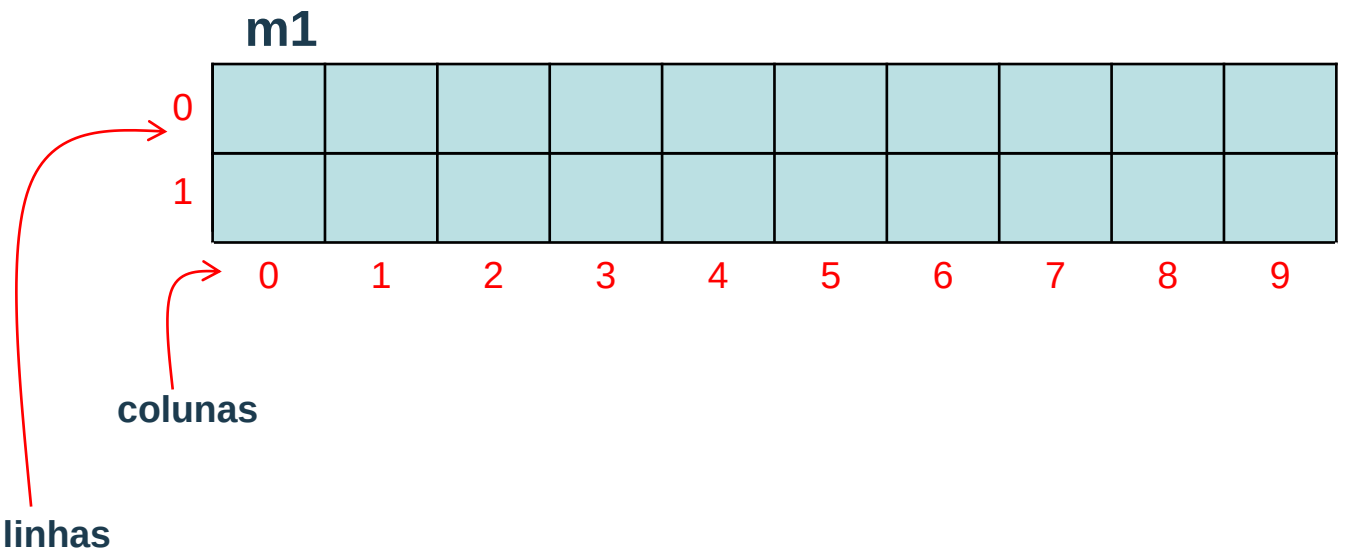
- Declarar matrizes.
- Atribuir um dado a uma determinada posição da matriz.
- Preencher todas as posições da matriz.
- Mostrar (escrever) todos os elementos da matriz.

Declaração de matrizes

```
1.  Algoritmo " Matrizes – Operação 1"
2.  Var //área de declaração de variáveis
3.
4.    m1 : vetor[0..1, 0..9] de inteiro
5.    m2 : vetor[0..4, 0..4] de inteiro
6.
7.  Inicio
8.  :
9.  :
10. :
11. :
12. Fimalgoritmo
```

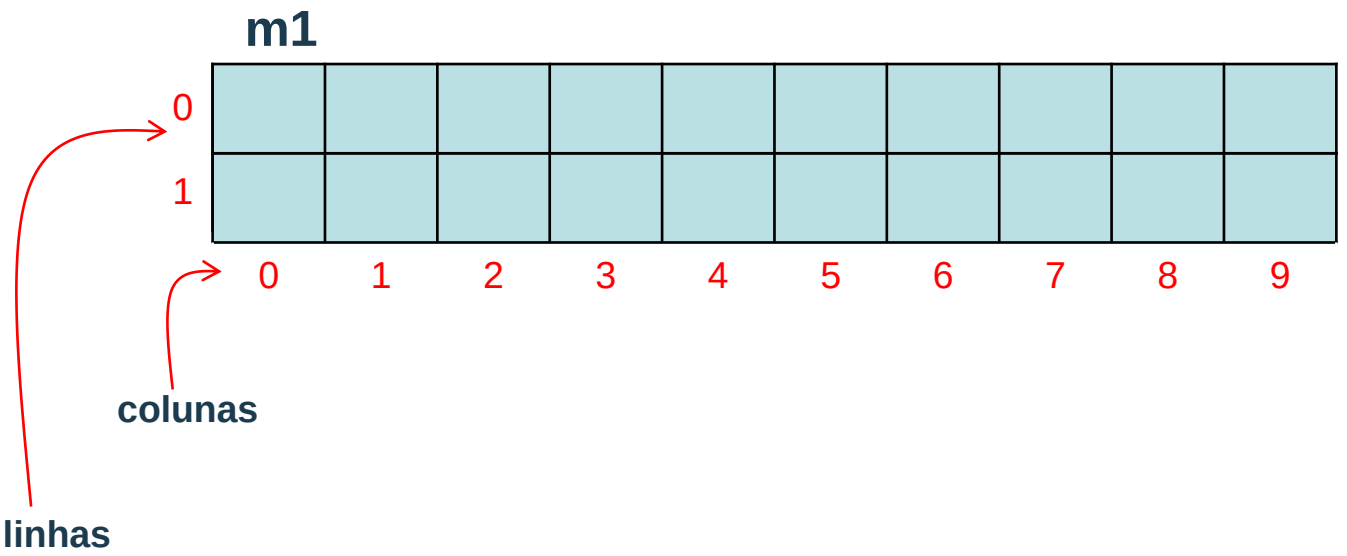
Declaração de matrizes

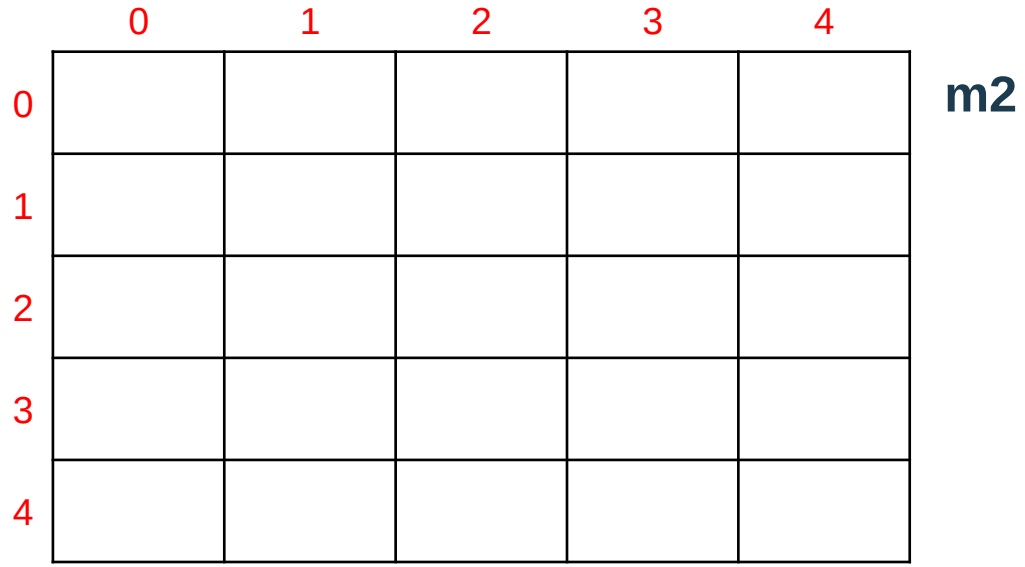
```
1.  Algoritmo " Matrizes – Operação 1"
2.  Var //área de declaração de variáveis
3.
4.    m1 : vetor[0..1, 0..9] de inteiro
5.    m2 : vetor[0..4, 0..4] de inteiro
6.
7.  Inicio
8.  :
9.  :
10. :
11. :
12. Fimalgoritmo
```



Declaração de matrizes

```
1.  Algoritmo " Matrizes – Operação 1"
2.  Var //área de declaração de variáveis
3.
4.    m1 : vetor[0..1, 0..9] de inteiro
5.    m2 : vetor[0..4, 0..4] de inteiro
6.
7.  Inicio
8.  :
9.  :
10. :
11. :
12. Fimalgoritmo
```





Atribuição de dados a posições específicas da matriz

```
1.  Algoritmo " Matrizes – Operação 1"
2.  Var //área de declaração de variáveis
3.
4.    m1 : vetor[0..1, 0..9] de inteiro
5.    m2 : vetor[0..4, 0..4] de inteiro
6.
7.  Inicio
8.    m1[0,0] <- 10
9.    m1[0,1] <- 20
10.   m1[0,2] <- 30
11.   m1[0,3] <- 40
12.   m1[0,4] <- 50
13.   m1[0,5] <- 60
14.   m1[0,6] <- 70
```

```
15.    m1[0,7] <- 80
16.    m1[0,8] <- 90
17.    m1[0,9] <- 100
18.    m1[1,0] <- 15
19.    m1[1,1] <- 25
20.    m1[1,2] <- 35
21.    m1[1,3] <- 45
22.    m1[1,4] <- 55
23.    m1[1,5] <- 65
24.    m1[1,6] <- 75
25.    m1[1,7] <- 85
26.    m1[1,8] <- 95
27.    M1[1,9] <- 105
28.  Fimalgoritmo
```

m1

0	10	20	30	40	50	60	70	80	90	100
1	15	25	35	45	55	65	75	85	95	105
	0	1	2	3	4	5	6	7	8	9

```
nome_do_matriz[<linha>,<coluna>] <- <valor>
```

Atribuição em matrizes

```
1.  Algoritmo " Matrizes – Operação 2"
2.  Var //área de declaração de variáveis
3.    linha, coluna : inteiro
4.    m1 : vetor[0..1, 0..9] de inteiro
5.    m2 : vetor[0..4, 0..4] de inteiro
6.
7.  Inicio
8.    m1[0,0] <- 10
9.    m1[0,1] <- 20
10.   m1[0,2] <- 30
11.   m1[0,3] <- 40
12.   m1[0,4] <- 50
13.   m1[0,5] <- 60
14.   m1[0,6] <- 70
```

```
15.     m1[0,7] <- 80
16.     m1[0,8] <- 90
17.     m1[0,9] <- 100
18.     m1[1,0] <- 15
19.     m1[1,1] <- 25
20.     m1[1,2] <- 35
21.     m1[1,3] <- 45
22.     m1[1,4] <- 55
23.     m1[1,5] <- 65
24.     m1[1,6] <- 75
25.     m1[1,7] <- 85
26.     m1[1,8] <- 95
27.     M1[1,9] <- 105
28.   Fimalgoritmo
```

m1

	0	1	2	3	4	5	6	7	8	9
0	10	20	30	40	50	60	70	80	90	100
1	15	25	35	45	55	65	75	85	95	105

Preencher as posições de uma matriz

Linha	Coluna	linha<=1	coluna<=2	M1[linha,coluna] <- valor
0	0	V	V	m2<[0,0] ← valor
	1		V	m2<[0,1] ← valor
	2		V	m2<[0,2] ← valor
	3		F	
1	0	V	V	m2<[0,0] ← valor
	1		V	m2<[0,1] ← valor
	2		V	m2<[0,2] ← valor
	3		F	
2		F		

Matriz m2

	0	1	2
0			
1			
2			

```
1.  Algoritmo " Matrizes – Operação 2"
2.  Var //área de declaração de variáveis
3.    linha, coluna : inteiro
4.    m1 : vetor[0..1, 0..9] de inteiro
5.    m2 : vetor[0..2, 0..2] de inteiro
6.
7.  Inicio
8.    aleatorio on
9.    para linha de 0 ate 1 passo 1 faca
10.     para coluna de 0 ate 9 passo 1 faca
11.       leia(m1[linha,coluna])
12.     fimpara
13.   fimpara
14.
15.   para linha de 0 ate 1 passo 1 faca
16.     para coluna de 0 ate 2 passo 1 faca
17.       leia(m2[linha,coluna])
18.     fimpara
19.   fimpara
20.   aleatorio off
21.  Fimalgoritmo
```

Mostrando os elementos de uma matriz

- Utilize o comando de saída para mostrar um elemento da matriz.

- Utilize uma estrutura de repetição encadeada para percorrer todos os elementos da matriz.

```
escreva(<nome da matriz>[<linha>,<coluna>])
escreva("mensagem", <nome da matriz>[<linha,coluna>])

escreva(" m[",índice, "] = " , v[i])
```

Escrevendo todos os elementos da matriz

m1

	0	1	2	3	4	5	6	7	8	9
0	0	1	2	3	4	5	6	7	8	9
1	1	2	3	4	5	6	7	8	9	10
2	2	3	4	5	6	7	8	9	10	11
3	3	4	5	6	7	8	9	10	11	12

```
1.  Algoritmo " Matrizes – Operação 3"
2.  Var //área de declaração de variáveis
3.    linha, coluna : inteiro
4.    m1 : vetor[0..1, 0..9] de inteiro
5.
6.  Inicio
7.    para linha de 0 ate 1 passo 1 faca
8.      para coluna de 0 ate 9 passo 1 faca
9.        m1[linha,coluna] ← linha+coluna
10.     fimpara
11.   fimpara
12.
13.
14.
15.   para linha de 0 ate 1 passo 1 faca
16.     para coluna de 0 ate 9 passo 1 faca
17.       escreva(m1[linha,coluna])
18.     fimpara
19.   fimpara
20.  Fimalgoritmo
```

Pesquisando um determinado elemento na matriz

```
1.  Algoritmo " Pesquisa elemento numa matriz "
2.  Var
3.      linha, coluna, x: inteiro
4.       M: vetor[0..4, 0..4] de inteiro
5.
6.  Inicio
7.      //preenchendo a matriz
8.      para linha de 0 até 4 passo 1 faca
9.         para coluna de 0 até 4 passo 1 faca
10.           escreva("M[", linha, "][", coluna, "]=  ")
11.           leia(M[linha,coluna])
12.        fimpara
13.     fimpara
14.
15.    //pesquisando um dado elemento no vetor
16.    escreva(" Qual valor deseja pesquisar: ")
17.    leia(x)
18.    para linha de 0 até 4 passo 1 faca
19.       para coluna de 0 ate 4 passo 1 faca
20.           se(M[linha,coluna] = x) entao
21.               escreval(" valor encontrado na posicao [",
                                          linha, ",", coluna, "]")
22.           fimse
23.        fimpara
24.    fimpara
25.  Fimalgoritmo
```

Interatividade

A matriz Notas[][] foi declarada da seguinte forma:

```
notas : vetor[0..29, 0..2] de inteiro
```

As duas primeiras colunas armazenam as médias bimestrais de cada aluno, sendo 1º e 2º Bimestres, respectivamente. Cada linha representa um aluno e pede-se o código que armazena na última coluna com a média aritmética de cada aluno. Assinale a alternativa que implementa este requisito.

Interatividade

a)

```
para linha de 0 ate 29 passo 1 faca
    notas[linha,2] <- ((notas[linha,0]+notas[linha,1])/2)
fimpara
```

b)

```
para coluna de 0 ate 2 passo 1 faca
    notas[2,coluna] <- ((notas[linha,0]+notas[linha,1])/2)
fimpara
```

c)

```
para linha de 0 ate 2 passo 1 faca
     para coluna de 0 ate 29 passo 1 faca
          notas[linha,coluna] <- notas[linha,coluna])/2
     fimpara
fimpara
```

d)

```
para linha de 0 ate 29 passo 1 faca
    para coluna de 0 a 2 passo 1 faca
      notas[linha, 3] <- ((notas[linha,0]+notas[linha,1])/2)
fimpara
```

e)

```
para coluna de 0 ate 29 passo 1 faca
  para linha de 0 ate 2 passo 1 faca
       notas[linha,2] <- ((notas[linha,0]+notas[linha,1])/2)
  fimpara
fimpara
```

Resposta

a)

```
para linha de 0 ate 29 passo 1 faca
    notas[linha,2] <- ((notas[linha,0]+notas[linha,1])/2)
fimpara
```

b)

```
para coluna de 0 ate 2 passo 1 faca
    notas[2,coluna] <- ((notas[linha,0]+notas[linha,1])/2)
fimpara
```

c)

```
para linha de 0 ate 2 passo 1 faca
     para coluna de 0 ate 29 passo 1 faca
          notas[linha,coluna] <- notas[linha,coluna])/2
     fimpara
fimpara
```

	0	1	2
0	10.0	8.0	
1	3.0	2.5	
2	8.0	6.0	
:			
29	9.0	5.0	

d)

```
para linha de 0 ate 29 passo 1 faca
    para coluna de 0 a 2 passo 1 faca
       notas[linha, 3] <- ((notas[linha,0]+notas[linha,1])/2)
fimpara
```

e)

```
para coluna de 0 ate 29 passo 1 faca
  para linha de 0 ate 2 passo 1 faca
       notas[linha,2] <- ((notas[linha,0]+notas[linha,1])/2)
  fimpara
fimpara
```

Resposta

a)

```
para linha de 0 ate 29 passo 1 faca
    notas[linha,2] <- ((notas[linha,0]+notas[linha,1])/2)
fimpara
```

b)

```
para coluna de 0 ate 2 passo 1 faca
    notas[2,coluna] <- ((notas[linha,0]+notas[linha,1])/2)
fimpara
```

c)

```
para linha de 0 ate 2 passo 1 faca
     para coluna de 0 ate 29 passo 1 faca
          notas[linha,coluna] <- notas[linha,coluna])/2
     fimpara
fimpara
```

	0	1	2
0	10.0	8.0	9.0
1	3.0	2.5	2.75
2	8.0	6.0	7.0
:			
29	9.0	5.0	7.0

d)

```
para linha de 0 ate 29 passo 1 faca
    para coluna de 0 a 2 passo 1 faca
       notas[linha, 3] <- ((notas[linha,0]+notas[linha,1])/2)
fimpara
```

e)

```
para coluna de 0 ate 29 passo 1 faca
  para linha de 0 ate 2 passo 1 faca
       notas[linha,2] <- ((notas[linha,0]+notas[linha,1])/2)
  fimpara
fimpara
```

Aplicações com matrizes

- Suponha que você tenha em uma matriz M1 5x10, compreendendo os dados de 50 peças em produção, e outra matriz M2 3x10, com os dados de outras 30 peças, e precise juntar todos os dados numa única matriz M3 8x10.

M1	0	1	2	3	4	5	6	7	8	9
0	1	2	3	4	5	6	7	8	9	10
1	11	12	13	14	15	16	17	18	19	20
2	21	22	23	24	25	26	27	28	29	30
3	31	32	33	34	35	36	37	38	39	40
4	41	42	43	44	45	46	47	48	49	50

M2	0	1	2	3	4	5	6	7	8	9
0	1	2	3	4	5	6	7	8	9	9
1	9	8	7	6	5	4	3	2	1	0
2	1	2	3	4	5	6	7	8	9	9

- A matriz M3 é preenchida com os dados das matrizes M1 e M2, de modo sequencial.

M3	0	1	2	3	4	5	6	7	8	9
0	1	2	3	4	5	6	7	8	9	9
1	11	12	13	14	15	16	17	18	19	20
2	21	22	23	24	25	26	27	28	29	30
3	31	32	33	34	35	36	37	38	39	40
4	41	42	43	44	45	46	47	48	49	50
5	1	2	3	4	5	6	7	8	9	10
6	11	12	13	14	15	16	17	18	19	20
7	21	22	23	24	25	26	27	28	29	30

Algoritmo para junção das matrizes M1 e M2

```
1. Algoritmo "Concatenar Matrizes"
2. var
3.  //declaração das variáveis
4.  M1: vetor [1..5, 1..10] de inteiro
5.  M2: vetor [1..3, 1..10] de inteiro
6.  M3: vetor [1..8, 1..10] de inteiro
7.
8.  i, linha, coluna: inteiro
9. Inicio
10.    i ← 0;
11.
12.    //preencher a matriz M1
13.    para linha de 1 a 5 passo 1 faca){
14.        para coluna de 0 a 10 passo 1 faca){
15.            M1[linha,coluna] ← i+1;
16.            i ← i+1
```

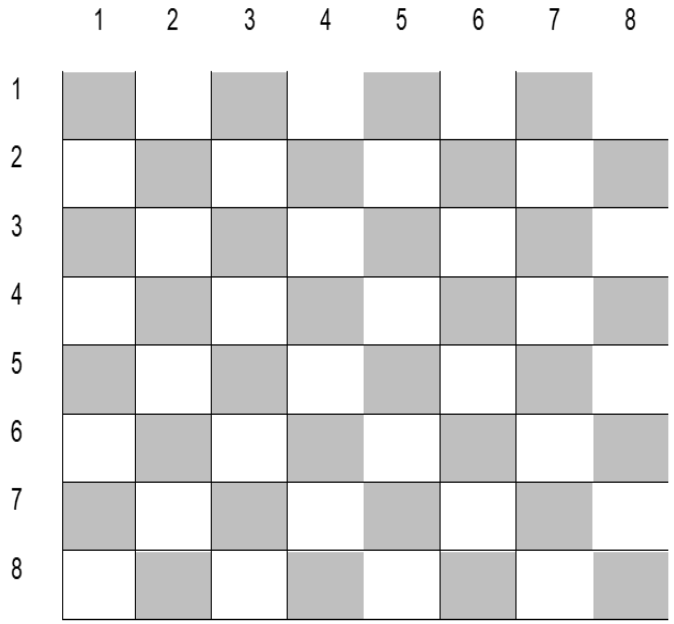
```
16.                escreva(" "
17.    fimpara
18.    escreval("  ", M1[linha,coluna]);
19. fimpara
20.
21. //preencher a matriz M2
22. para linha de 1 a 3 passo 1 faca){
23.    para coluna de 0 a 10 passo 1 faca){
24.            i ← i+1
25.        M2[linha,coluna] ← i;
26.            escreval("  ", M2[linha,coluna]);
27.    fimpara
28.    escreval("\n ");
29. fimpara
30.
31. //preencher a Matriz M3 conforme enunciado
32. para linha de 1 a 5 passo 1 faca){
33.    para coluna de 0 a 10 passo 1 faca){
34.        se(linha<3) entao
35.            M3[linha,coluna] = M1[linha,coluna];
36.            M3[linha+5,coluna] = M2[linha,coluna];
37.        }
38.        senao{
39.            M3[linha,coluna] = M1[linha,coluna];
40.        fimse
41.    fimpara
42. fimpara
```

Algoritmo para junção das matrizes M1 e M2

```
1. Algoritmo “Concatenar Matrizes”
2. var
3.  //declaração das variáveis
4.   M1: vetor [1..5, 1..10] de inteiro
5.   M2: vetor [1..3, 1..10] de inteiro
6.   M3: vetor [1..8, 1..10] de inteiro
7.
8.   i, linha, coluna: inteiro
9. Inicio
10.    i ← 0;
11.
12.    //preencher a matriz M1
13.    para linha de 1 a 5 passo 1 faca){
14.        para coluna de 0 a 10 passo 1 faca){
15.            M1[linha,coluna] ← i+1;
16.             i ← i+1
   ⋮
43.    //mostrar todos os dados da matriz M3
44.    para linha de 1 a 8 passo 1 faca){
45.        para coluna de 0 a 10 passo 1 faca){
46.                escreva(M3[linha][coluna]);
47.        fimpara
48.        escreval(" ");
49.    fimpara
50. Fimalgoritmo
```

Exemplo de aplicação

- Implementação de jogos de tabuleiro.




```
Algoritmo "Tabuleiro de Xadrez"
Var
   //declaração das variáveis
   matriz_tabuleiro: vetor[0..7,0..7] de caractere
   i, j: inteiro

procedimento tabuleiro()
inicio
   //Laço de repetição para preencher as posições do Peao
   para i de 0 ate 7 passo 1 faca
          tabuleiro[1][i] <- "Peão"
          tabuleiro[6][i] <- "Peão"
   fimpara

    //Preenchimento das posições da Torre, Cavalo, Bispo, Dama e Rei
    Tabuleiro[0][0] <- "Torre"
    Tabuleiro[0][1] <- "Cavalo"
    Tabuleiro[0][2] <- "Bispo"
    Tabuleiro[0][3] <- "Dama"
    Tabuleiro[0][4] <- "Rei"
    Tabuleiro[0][5] <- "Bispo"
    Tabuleiro[0][6] <- "Cavalo"
    Tabuleiro[0][7] <- "Torre"
fimprocedimento

Inicio
   tabuleiro()
Fimalgoritmo
```

Exemplo

Deseja armazenar na matriz C os dados das matrizes A e B, respeitando a seguinte regra:

- os dados da matriz A serão armazenados nas linhas pares da matriz C.
- os dados da matriz B serão armazenados nas linhas ímpares da matriz C.

- Considere que as matrizes A e B possuem dimensão 5x5 e a matriz C 10x5.

```
Var
  A : vetor[0..4,0..5] de inteiro
  B : vetor[0..4,0..5] de inteiro
  C : vetor[0..9,0..5] de inteiro
  linha, coluna, i, j : inteiro
Inicio
i <- 0
j <- 0
para linha de 0 a 9 passo 1 faca
     para coluna de 0 a 5 passo 1 faca
          se ((linha mod 2)=0)
             C[linha][coluna] <- A[i][coluna]
             i ← i+1
          senão
             C[linha][coluna] <- A[j][coluna]
             j ← j+1
          fimse
     fimpara
  fimpara
Fimalgoritmo
```

	0	1	2	3	4	5
0						
1						
2						
3						
4						
5						
6						
7						
8						
9						

Exemplo

Deseja armazenar na matriz C os dados das matrizes A e B, respeitando a seguinte regra:

- os dados da matriz A serão armazenados nas linhas pares da matriz C.
- os dados da matriz B serão armazenados nas linhas ímpares da matriz C.

- Considere que as matrizes A e B possuem dimensão 5x5 e a matriz C 10x5.

```
Var
  A : vetor[0..4,0..5] de inteiro
  B : vetor[0..4,0..5] de inteiro
  C : vetor[0..9,0..5] de inteiro
  linha, coluna, i, j : inteiro
Inicio
i <- 0
j <- 0
para linha de 0 a 9 passo 1 faca
     para coluna de 0 a 5 passo 1 faca
          se ((linha mod 2)=0)
             C[linha][coluna] <- A[i][coluna]
             i ← i+1
          senão
             C[linha][coluna] <- B[j][coluna]
             j ← j+1
          fimse
     fimpara
  fimpara
Fimalgoritmo
```

C	0	1	2	3	4	5
0						
1						
2						
3						
4						
5						
6						
7						
8						
9						

Interatividade

Matrizes e Vetores são estruturas de dados estáticas, compostas e homogêneas

Porque

Matrizes e Vetores alocam várias posições de memória, usam o mesmo identificador, uma vez declarada, o tamanho não poderá ser redimensionado, e todos os dados são do mesmo tipo.

A respeito das duas asserções, assinale a resposta correta.

a) As duas asserções são proposições verdadeiras e a segunda é a justificativa correta da primeira.
b) As duas asserções são proposições verdadeiras, mas a segunda não é justificativa da primeira.
c) A primeira asserção é falsa, mas a segunda é verdadeira.
d) A primeira asserção é verdadeira, mas a segunda é falsa.
e) Ambas as asserções são falsas.

Resposta

Matrizes e Vetores são estruturas de dados estáticas, compostas e homogêneas

Porque

Matrizes e Vetores alocam várias posições de memória, usam o mesmo identificador, uma vez declarada, o tamanho não poderá ser redimensionado, e todos os dados são do mesmo tipo.

A respeito das duas asserções, assinale a resposta correta.

a) As duas asserções são proposições verdadeiras e a segunda é a justificativa correta da primeira.

b) As duas asserções são proposições verdadeiras, mas a segunda não é justificativa da primeira.

c) A primeira asserção é falsa, mas a segunda é verdadeira.

d) A primeira asserção é verdadeira, mas a segunda é falsa.

e) Ambas as asserções são falsas.

ATÉ A PRÓXIMA!