

INSTITUTO FEDERAL DE
EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
RIO GRANDE DO NORTE

CONTEÚDO 04

VETORES E MATRIZES

Disciplina: Algoritmos e POO

Professora: Alba Lopes

alba.lopes@ifrn.edu.br

<http://docente.ifrn.edu.br/albalopes>

INTRODUÇÃO

○ Variável

- Analogia: uma caixa, na qual você pode dar o nome que lhe achar conveniente, e guardar o conteúdo que desejar



- Possui um tipo (caractere, lógico, inteiro ou real)
- O valor dentro da “caixa” que pode ser alterado de acordo com a execução do algoritmo



INTRODUÇÃO

- Agora imagine como ficaria na declaração de variáveis, declarando uma a uma, as 50 variáveis para o nome, depois as variáveis para as médias de cada aluno...

var

```
nota1, nota2, nota3, nota4: real  
nome_aluno1, nome_aluno2, nome_aluno3, ....., nome_aluno50: caractere  
media_aluno1, media_aluno2, media_aluno3, ....., media_aluno50: real
```



INTRODUÇÃO

- O problema começa quando se precisa declarar várias variáveis para atender a um fim.
- **PROBLEMA:** Receber o nome e as 4 notas de 50 alunos de uma escola, e depois **listar** o nome de cada aluno junto com sua média.

```
Digite o nome do aluno: Paulo
  Digite a nota 1 do aluno :50
  Digite a nota 2 do aluno :80
  Digite a nota 3 do aluno :40
  Digite a nota 4 do aluno :60
Digite o nome do aluno: José
  Digite a nota 1 do aluno :80
  Digite a nota 2 do aluno :100
  Digite a nota 3 do aluno :73
  Digite a nota 4 do aluno :70
Digite o nome do aluno:
```

...

```
**** Alunos - Medias****
Paulo - 57.5
José - 80.75
```

...



VETORES

- Em casos como esse que é útil a utilização da **estrutura de dados** conhecida como **vetor**
- Um vetor é uma espécie de caixa com várias divisórias para armazenar coisas (dados)
 - É uma variável que pode armazenar vários valores



VETORES

meuVetor

--	--	--	--	--	--	--	--

medias

10	40	8	26	70	73		
----	----	---	----	----	----	--	--

nomes

Paulo	José	Maria	Ricardo				
-------	------	-------	---------	--	--	--	--



VETORES

- Os vetores são definidos pelo **tipo de dados** que eles devem armazenar e a **quantidade de posições**
- **Exemplo:**
 - Vetor de 8 posições para armazenar números reais
 - Vetor de 40 posições para armazenar caracteres
- Os vetores são estruturas **homogêneas**.
 - Ex: um vetor de inteiros só armazena dados do tipo inteiro



SINTAXE NO VISUALG

○ Declaração:

```
<nome_variavel>: vetor [posInicial..posFinal] de <tipo>
```

○ Exemplo:

```
algoritmo "exemplo_vetores"  
var  
    nome_alunos: vetor [1..50] de caractere  
    media_alunos: vetor [1..50] de real  
inicio
```

Note que são apenas
DOIS PONTOS!



SINTAXE NO VISUALG

○ Preenchendo e acessando um vetor

- As posições dos vetores são identificadas por índices
- Um vetor de 10 posições, por exemplo pode ser representado da seguinte forma:



SINTAXE NO VISUALG

○ Atribuição

```
<nome_variavel> [<posicao>] ← <valor>  
<nome_variavel> [<posicao>] := <valor>  
leia(<nome_variavel> [<posicao>])
```

○ Exemplo:

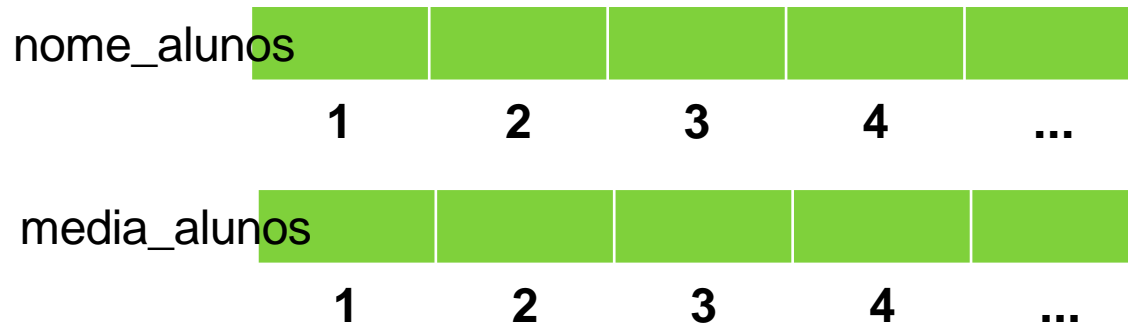
```
algoritmo "exemplo_vetores"  
var  
    nome_alunos: vetor [1..50] de caractere  
    media_alunos: vetor [1..50] de real  
inicio  
    nome_alunos[1] ← "Pedro"  
    leia(nome_alunos[2])  
    nome_alunos[3] := "Joana"  
    media_alunos[1] := 8.5
```



SINTAXE NO VISUALG

○ Exemplo:

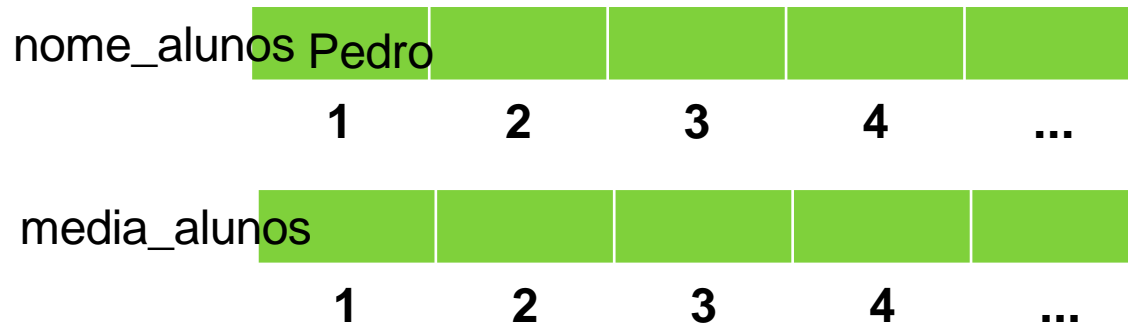
```
algoritmo "exemplo_vetores"  
var  
    nome_alunos: vetor [1..50] de caractere  
    media_alunos: vetor [1..50] de real  
inicio  
    nome_alunos[1] ← "Pedro"  
    leia(nome_alunos[2])  
    nome_alunos[3] := "Joana"  
    media_alunos[1] := 8.5
```



SINTAXE NO VISUALG

○ Exemplo:

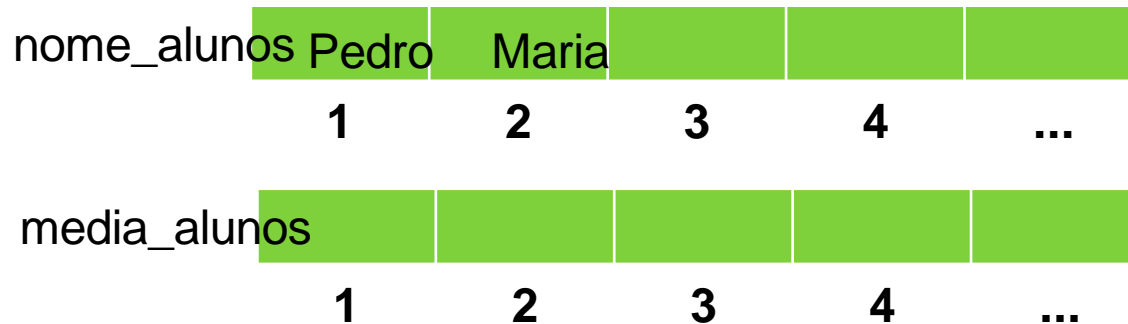
```
algoritmo "exemplo_vetores"  
var  
    nome_alunos: vetor [1..50] de caractere  
    media_alunos: vetor [1..50] de real  
inicio  
    nome_alunos[1] ← "Pedro"  
    leia(nome_alunos[2])  
    nome_alunos[3] := "Joana"  
    media_alunos[1] := 8.5
```



SINTAXE NO VISUALG

○ Exemplo:

```
algoritmo "exemplo_vetores"  
var  
    nome_alunos: vetor [1..50] de caractere  
    media_alunos: vetor [1..50] de real  
inicio  
    nome_alunos[1] ← "Pedro"  
    leia(nome_alunos[2])  
    nome_alunos[3] := "Joana"  
    media_alunos[1] := 8.5
```



SINTAXE NO VISUALG

○ Exemplo:

```
algoritmo "exemplo_vetores"  
var  
    nome_alunos: vetor [1..50] de caractere  
    media_alunos: vetor [1..50] de real  
inicio  
    nome_alunos[1] ← "Pedro"  
    leia(nome_alunos[2])  
    nome_alunos[3] := "Joana"  
    media_alunos[1] := 8.5
```

nome_alunos	Pedro	Maria	Joana		
	1	2	3	4	...
media_alunos					
	1	2	3	4	...



SINTAXE NO VISUALG

○ Exemplo:

```
algoritmo "exemplo_vetores"  
var  
    nome_alunos: vetor [1..50] de caractere  
    media_alunos: vetor [1..50] de real  
inicio  
    nome_alunos[1] ← "Pedro"  
    leia(nome_alunos[2])  
    nome_alunos[3] := "Joana"  
    media_alunos[1] := 8.5
```

nome_alunos	Pedro	Maria	Joana		
	1	2	3	4	...
media_alunos	8.5				
	1	2	3	4	...



SINTAXE NO VISUALG

○ Preenchendo um vetor

- Podemos utilizar um laço de repetição para facilitar o preenchimento dos dados em vetores

○ Exemplo:

```
algoritmo "exemplo_vetores"
var
    numeros: vetor [1..10] de inteiro
    i: inteiro
inicio
    para i de 1 ate 10 faca
        escreva("Digite um valor para ser adicionado ao vetor")
        leia(numeros[i])
    fimpara
fimpara
```



SINTAXE NO VISUALG

○ Preenchendo um vetor

```
algoritmo "exemplo_vetores"  
var  
    numeros: vetor [1..5] de inteiro  
inicio  
    escreva("Digite um valor para a posição 1 do vetor:")  
    leia(numeros[1])  
    escreva("Digite um valor para a posição 2 do vetor:")  
    leia(numeros[2])  
    escreva("Digite um valor para a posição 3 do vetor:")  
    leia(numeros[3])  
    escreva("Digite um valor para a posição 4 do vetor:")  
    leia(numeros[4])  
    escreva("Digite um valor para a posição 5 do vetor:")  
    leia(numeros[5])  
fimpara
```



SINTAXE NO VISUALG

○ Preenchendo um vetor

- Para facilitar, podemos utilizar um laço de repetição!

○ Exemplo:

```
algoritmo "exemplo_vetores"  
var  
    numeros: vetor [1..5] de inteiro  
    i: inteiro  
inicio  
    para i de 1 ate 5 faca  
        escreva("Digite um valor para a posição ", i , "do vetor:")  
        leia(numeros[i])  
    fimpara  
fimpara
```



SINTAXE NO VISUALG

○ Exibindo o conteúdo de um vetor:

```
...  
    escreva("O valor que está na posição 1 é: ", numeros[1])  
    escreva("O valor que está na posição 2 é: ", numeros[2])  
    escreva("O valor que está na posição 3 é: ", numeros[3])  
    escreva("O valor que está na posição 4 é: ", numeros[4])  
    escreva("O valor que está na posição 5 é: ", numeros[5])  
finalgoritmo
```



SINTAXE NO VISUALG

○ Exibindo o conteúdo de um vetor

- Ou podemos utilizar um laço de repetição para facilitar a exibição dos valores de um vetor

○ Exemplo:

```
para i de 1 ate 5 faça
    escreva("O valor que está na posição ", i , " é: ", numeros[i])
fimpara
```



EXEMPLO 1

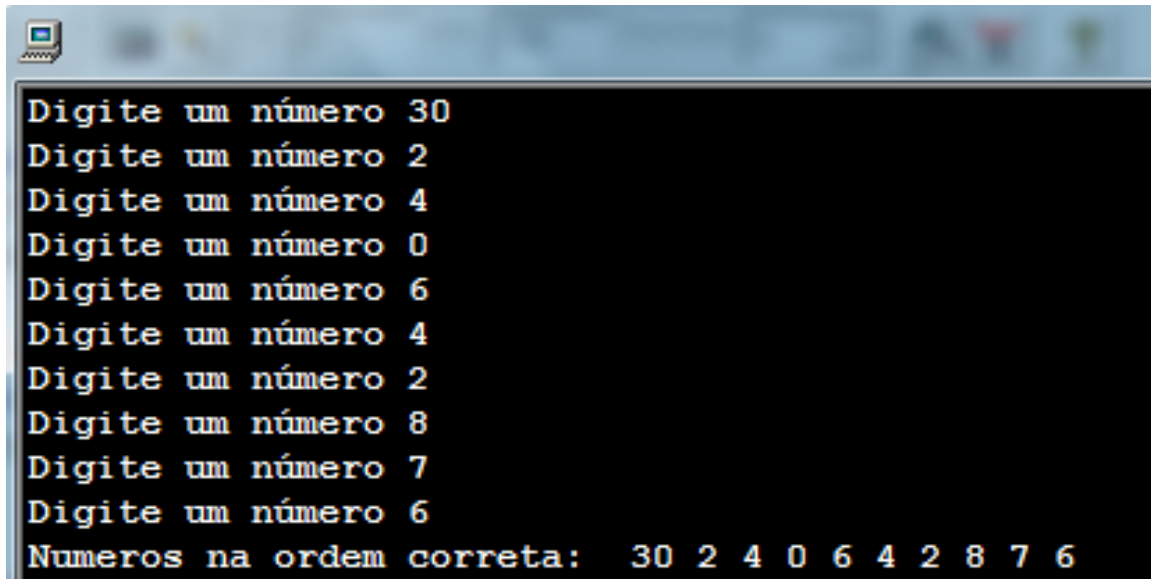
- Criar um algoritmo que leia 10 números pelo teclado e exiba os números na ordem correta que os números foram digitados.

```
algoritmo "vetor"  
var  
    numeros: vetor [1..10] de inteiro  
    i: inteiro  
inicio  
    para i de 1 ate 10 faca  
        escreva("Digite um número ")  
        leia(numeros[i])  
    fimpara  
    escreva("Numeros na ordem correta: ")  
    para i de 1 ate 10 faca  
        escreva(numeros[i])  
    fimpara  
fimalgoritmo
```



EXEMPLO 1

- Saída:

A terminal window with a black background and white text. The text shows a sequence of prompts and user inputs for a sorting algorithm. The prompts are "Digite um número" followed by a number. The user inputs are 30, 2, 4, 0, 6, 4, 2, 8, 7, 6. The final line shows the sorted numbers: "Numeros na ordem correta: 30 2 4 0 6 4 2 8 7 6".

```
Digite um número 30
Digite um número 2
Digite um número 4
Digite um número 0
Digite um número 6
Digite um número 4
Digite um número 2
Digite um número 8
Digite um número 7
Digite um número 6
Numeros na ordem correta: 30 2 4 0 6 4 2 8 7 6
```



EXEMPLO 2

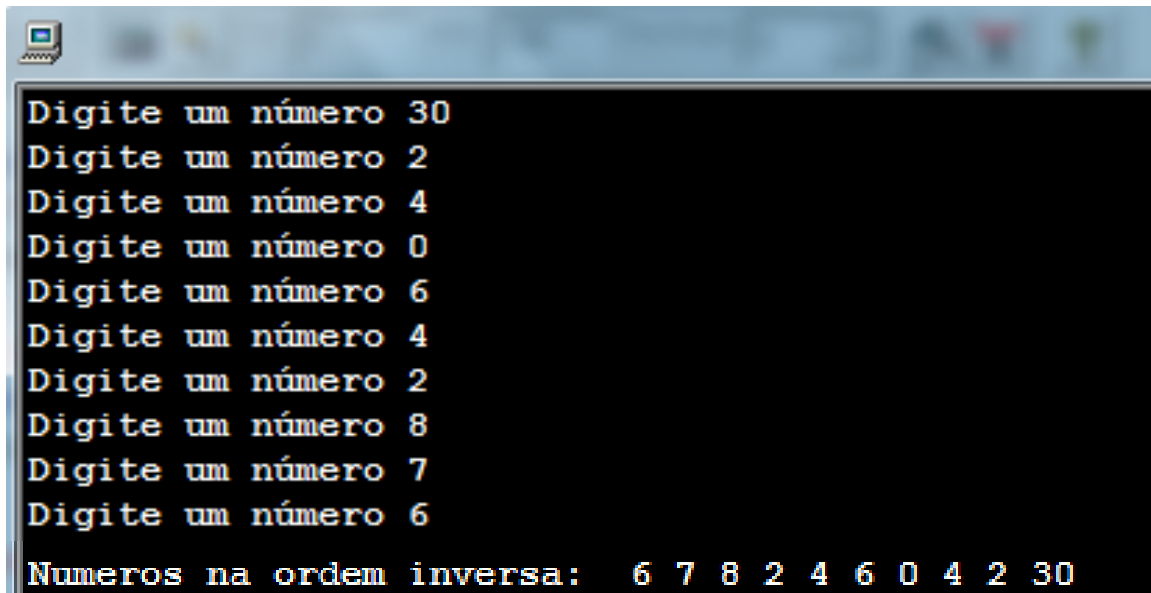
- Criar um algoritmo que leia 10 números pelo teclado e exiba os números na ordem inversa da que os números foram digitados.

```
algoritmo "vetor"  
var  
    numeros: vetor [1..10] de inteiro  
    i: inteiro  
inicio  
    para i de 1 ate 10 faca  
        escreva("Digite um número ")  
        leia(numeros[i])  
    fimpara  
    escreva("Numeros na ordem inversa: ")  
    para i de 10 ate 1 passo -1 faca  
        escreva(numeros[i])  
    fimpara  
fimalgoritmo
```



EXEMPLO 2

- Saída:



```
Digite um número 30
Digite um número 2
Digite um número 4
Digite um número 0
Digite um número 6
Digite um número 4
Digite um número 2
Digite um número 8
Digite um número 7
Digite um número 6
Numeros na ordem inversa: 6 7 8 2 4 6 0 4 2 30
```



EXEMPLO 3

- Escreva um algoritmo que leia um vetor com 10 posições de números inteiros. Em seguida, receba um novo valor do usuário e verifique se este valor se encontra no vetor.



EXEMPLO 3

```
algoritmo "busca_valor"  
var  
  numeros: vetor [1..10] de inteiro  
  i, valor : inteiro  
  encontrou: logico  
inicio  
  para i de 1 ate 10 faca  
    escreva("Digite um número ")  
    leia(numeros[i])  
  fimpara  
  
  escreva("Digite um número para ser buscado no vetor: ")  
  leia(valor)  
  
  encontrou <- falso  
  para i de 1 ate 10 faca  
    se (numeros[i] = valor) entao  
      encontrou <- verdadeiro  
    fimse  
  fimpara  
  
  se encontrou entao  
    escreva("O valor se encontra no vetor")  
  senao  
    escreva("O valor não se encontra no vetor")  
  fimse  
fimalgoritmo
```



EXEMPLO 3 (UM PEQUENO PARÊNTESES)

- As estruturas de repetição (tanto **para**, **enquanto** e **repita**) permitem o uso do comando **INTERROMPA**
 - Esse comando causa a saída imediata do laço de repetição

```
escreva("Digite um número para ser buscado no vetor: ");
leia(valor)

encontrou <- falso
para i de 1 ate 10 faça
    se (numeros[i] = valor) entao
        encontrou <- verdadeiro
        interrompa
    fimse
fimpara

se (encontrou = verdadeiro) entao
    escreva("O valor se encontra no vetor")
senao
    escreva("O valor não se encontra no vetor")
fimse
```

Ao encontrar esse comando, o algoritmo passa a execução para o próximo comando após o laço.

EXERCÍCIOS

1. Crie um algoritmo que leia um vetor de 10 números inteiros. Em seguida, calcule e escreva o somatório dos valores deste vetor.
2. Escreva um algoritmo que leia um vetor com 15 posições de números inteiros. Em seguida, escreva somente os números positivos que se encontram no vetor.
3. Escreva um algoritmo que leia um vetor com 8 posições de números inteiros. Em seguida, leia um novo valor do usuário e verifique se o valor se encontra no vetor. Se estiver, informe a posição desse elemento no vetor. Caso o elemento não esteja no vetor, apresente uma mensagem informando “O número não se encontra no vetor”.



EXERCÍCIOS

5. Escreva um algoritmo que leia **dois** vetores de 10 posições e faça a soma dos elementos de mesmo índice, colocando o resultado em um terceiro vetor. Mostre o vetor resultante.

Exemplo:

vetor1	7	4	9	15	20	2	1	4	0	30
vetor2	1	8	3	7	14	9	1	8	11	16
vetorResultado	8	12	12	22	34	11	2	12	11	46

6. Crie um algoritmo que leia um vetor de 20 posições e informe:
- Quantos números pares existem no vetor
 - Quantos números ímpares existem no vetor
 - Quantos números maiores do que 50
 - Quantos números menores do que 7



MATRIZES

- O que é uma matriz?
 - Uma estrutura de dados que contém várias variáveis do mesmo tipo
- Qual a diferença de **vetores** para **matrizes**?
 - Vetores são, na verdade, matrizes de uma única dimensão:

Vetores

1	3	4	6
---	---	---	---

a	maria	jota
---	-------	------

Matrizes

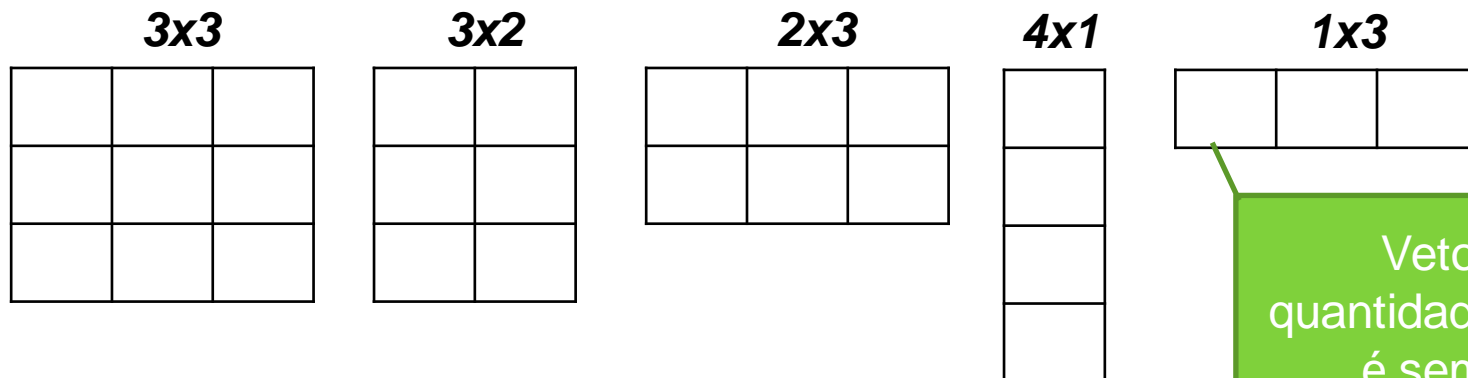
1	3
40	4
6	12

M	J	K
G	A	C
L	Z	H

1.1	7.5	9.2	8.8
9.0	1.3	5.5	7.9

MATRIZES

- As matrizes são, comumente referenciadas através de suas dimensões (quantidade de linhas e colunas)
- A notação comum é: $M \times N$, onde
 - M é a dimensão vertical (**quantidade de linhas**)
 - N é dimensão horizontal (**quantidade de colunas**)
- Exemplo:

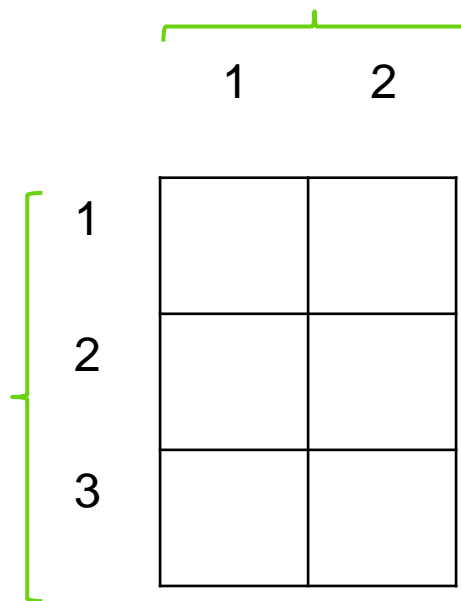


Vetores: a quantidade de linhas é sempre 1!

MATRIZES

○ Notação

- Como referenciar um elemento específico da matriz?
- Exemplo: Matriz 3x2 (*três linhas e duas colunas*)



As linhas
variam de 1
até 3

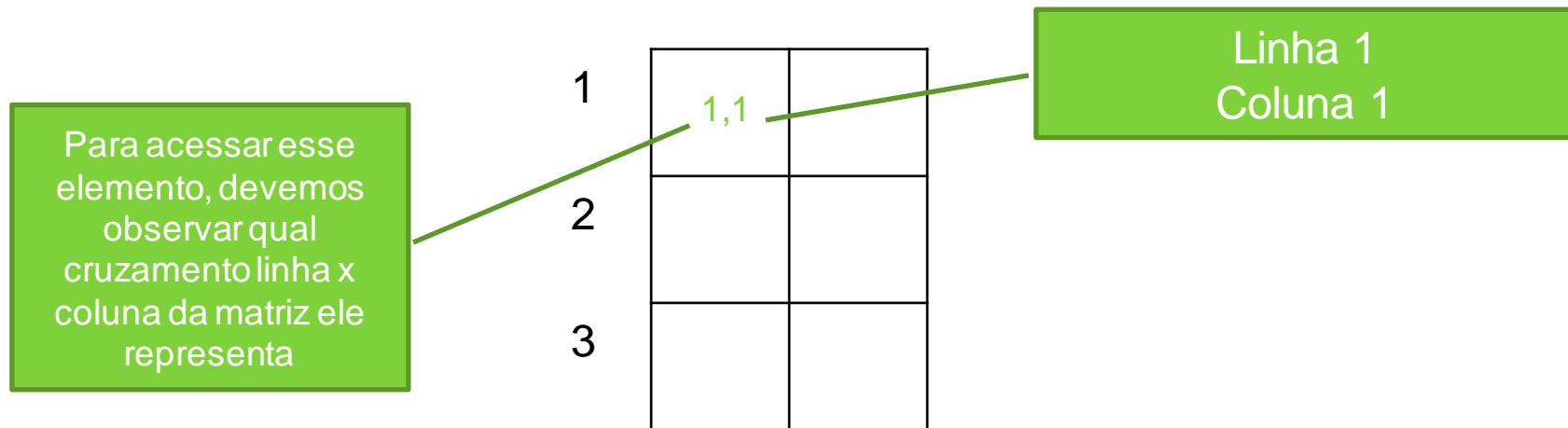
As colunas
variam de 1
até 2



MATRIZES

○ Notação

- Como referenciar um elemento específico da matriz?
- Exemplo: Matriz 3x2 (*três linhas e duas colunas*)



MATRIZES

○ Notação

- Como referenciar um elemento específico da matriz?
- Exemplo: Matriz 3x2 (*três linhas e duas colunas*)

	1	2
1	1,1	1,2
2		
3		

Linha 1
Coluna 2



MATRIZES

○ Notação

- Como referenciar um elemento específico da matriz?
- Exemplo: Matriz 3x2 (*três linhas e duas colunas*)

	1	2
1	1,1	1,2
2	2,1	
3		

Linha 2
Coluna 1

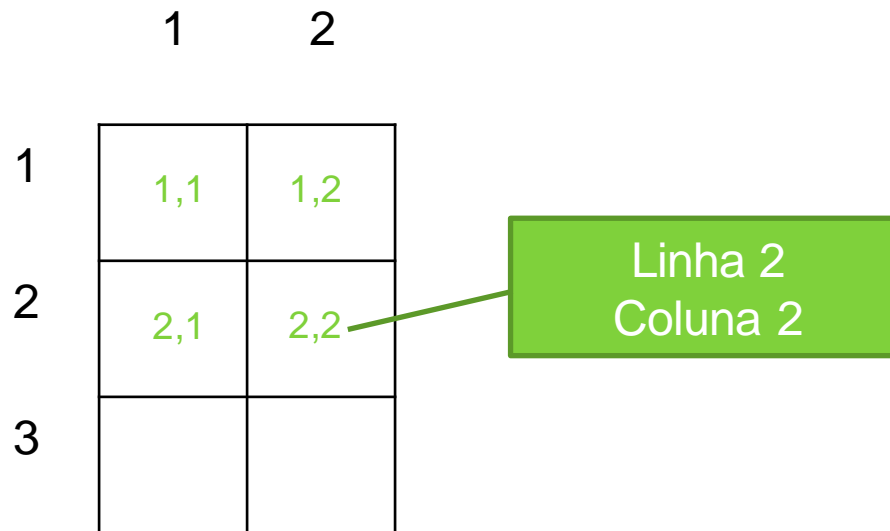


MATRIZES

○ Notação

- Como referenciar um elemento específico da matriz?
- Exemplo: Matriz 3x2 (*três linhas e duas colunas*)

	1	2
1	1,1	1,2
2	2,1	2,2
3		

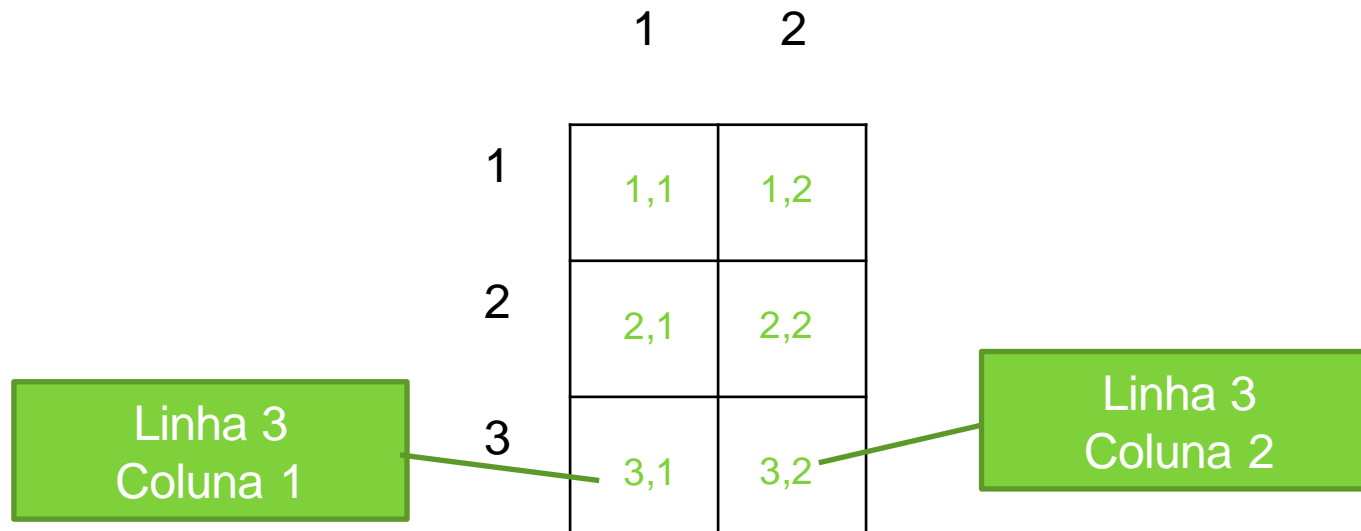


A diagram illustrating a 3x2 matrix. The matrix is represented as a table with 3 rows and 2 columns. The rows are labeled 1, 2, and 3 on the left. The columns are labeled 1 and 2 at the top. The elements in the first two rows are labeled as follows: row 1, column 1 is '1,1'; row 1, column 2 is '1,2'; row 2, column 1 is '2,1'; row 2, column 2 is '2,2'. A green callout box with a pointer indicates the element at row 2, column 2, labeled 'Linha 2 Coluna 2'. A green circle is located in the bottom right corner of the slide.

MATRIZES

○ Notação

- Como referenciar um elemento específico da matriz?
- Exemplo: Matriz 3x2 (*três linhas e duas colunas*)



SINTAXE NO VISUALG

○ Declaração:

```
<nome_variavel>: vetor [li..lf, ci..cf] de <tipo>
```

○ Onde:

- li e lf representam, respectivamente o índice inicial e final das **linhas** e
- ci e cf representam, respectivamente o índice inicial e final das **colunas**



SINTAXE NO VISUALG

○ Exemplo:

- Para declarar uma matriz 3x2 de inteiro

```
algoritmo "exemplo_matriz"  
var  
    exMatriz: vetor [1..3, 1..2] de inteiro  
inicio  
    ...
```

Linhas: o índice das
linhas varia de 1 até 3

Colunas: o índice das
colunas varia de 1 até 2



SINTAXE NO VISUALG

- **Preenchendo e acessando uma matriz**
 - As posições das matrizes são identificados pelos índices das linhas e colunas

- **Atribuição**

```
<nome_variavel> [<linha>, <coluna>] ← <valor>  
<nome_variavel> [<linha>, <coluna>] := <valor>  
leia(<nome_variavel> [<linha>, <coluna>])
```



SINTAXE NO VISUALG

o Exemplo:

```
algoritmo "exemplo_matriz"  
var  
    exMatriz: vetor [1..3, 1..2] de inteiro  
inicio  
    exMatriz[1,1] ← 10  
    leia(exMatriz[1,2])  
    exMatriz[3,1] := 4  
finalgoritmo
```

	1	2
1		
2		
3		

exMatriz



SINTAXE NO VISUALG

o Exemplo:

```
algoritmo "exemplo_matriz"  
var  
    exMatriz: vetor [1..3, 1..2] de inteiro  
inicio  
    exMatriz[1,1] ← 10  
    leia(exMatriz[1,2])  
    exMatriz[3,1] := 4  
finalgoritmo
```

	1	2
1	10	
2		
3		

exMatriz



SINTAXE NO VISUALG

o Exemplo:

```
algoritmo "exemplo_matriz"  
var  
    exMatriz: vetor [1..3, 1..2] de inteiro  
inicio  
    exMatriz[1,1] ← 10  
    leia(exMatriz[1,2])  
    exMatriz[3,1] := 4  
finalgoritmo
```

	1	2
1	10	7
2		
3		

exMatriz



SINTAXE NO VISUALG

o Exemplo:

```
algoritmo "exemplo_matriz"  
var  
    exMatriz: vetor [1..3, 1..2] de inteiro  
inicio  
    exMatriz[1,1] ← 10  
    leia(exMatriz[1,2])  
    exMatriz[3,1] := 4  
finalgoritmo
```

	1	2
1	10	7
2		
3	4	

exMatriz



SINTAXE NO VISUALG

○ Preenchendo uma matriz

- Se quisermos atribuir valores a todas as posições da matriz, podemos fazer:

```
algoritmo "preencher_matrizes"
var
    numeros: vetor[1..3, 1..2] de inteiro
    i: inteiro
inicio
    para i de 1 ate 3 faca //fazer o laço para as linhas
        escreva("Digite o valor para a posicao ", i, ", 1":)
        leia(numeros[i, 1])
        escreva("Digite o valor para a posicao ", i, ", 2":)
        leia(numeros[i, 2])
    fimpara
fimalgoritmo
```

SINTAXE NO VISUALG

○ Preenchendo uma matriz

- Se quisermos atribuir valores a todas as posições da matriz, podemos fazer:

```
algoritmo "preencher"  
var  
    numeros: vetor[1..3, 1..2] de inteiro  
    i, j: inteiro  
inicio  
    escreva("Digite um valor para a posição [1,1]")  
    leia(numeros[1,1])  
    escreva("Digite um valor para a posição [1,2]")  
    leia(numeros[1,2])  
    escreva("Digite um valor para a posição [2,1]")  
    leia(numeros[2,1])  
    escreva("Digite um valor para a posição [2,2]")  
    leia(numeros[2,2])  
    escreva("Digite um valor para a posição [3,1]")  
    leia(numeros[3,1])  
    escreva("Digite um valor para a posição [3,2]")  
    leia(numeros[3,2])  
fimalgoritmo
```



SINTAXE NO VISUALG

○ **Preenchendo uma matriz**

- Entretanto, à medida que a quantidade de elementos da matriz aumenta, fica complicado fazermos manualmente para todas as posições.
- O melhor caminho é utilizar laços de repetição!



SINTAXE NO VISUALG

○ Preenchendo uma matriz

- Podemos criar um laço de repetição para variar pelas linhas, por exemplo:

```
algoritmo "preencher"  
var  
    numeros: vetor[1..3, 1..2] de inteiro  
    i, j: inteiro  
inicio  
    para i de 1 ate 3 faça  
        escreva("Digite um valor para a posição [", i, ",1")  
        leia(numeros[i,1])  
        escreva("Digite um valor para a posição [", i, ",2")  
        leia(numeros[i,2])  
    fimpara  
fimalgoritmo
```



SINTAXE NO VISUALG

○ Preenchendo uma matriz

- E podemos ainda incluir um laço de repetição para variar pelas colunas também, por exemplo:

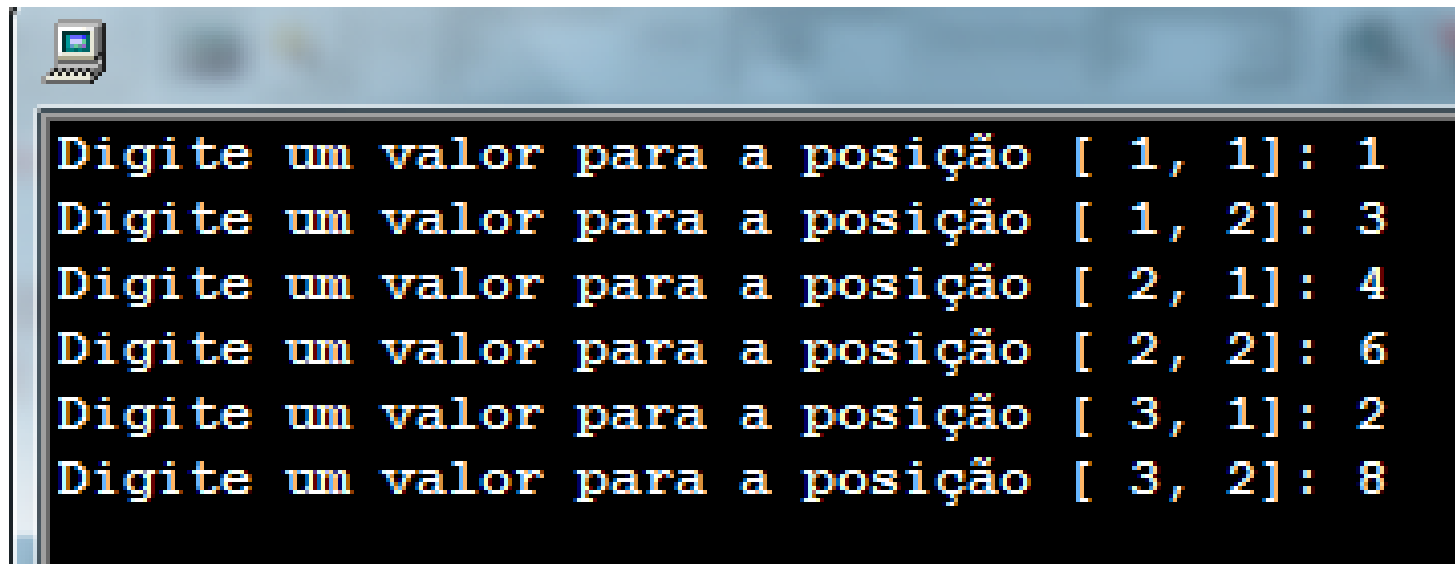
```
algoritmo "preencher"  
var  
    numeros: vetor[1..3, 1..2] de inteiro  
    i, j: inteiro  
inicio  
    para i de 1 ate 3 faca  
        para j de 1 ate 2 faca  
            escreva("Digite um valor para a posição [", i, ",", j, "]: ")  
            leia(numeros[i,j])  
        fimpara  
    fimpara  
fimalgoritmo
```



SINTAXE NO VISUALG

○ Preenchendo uma matriz

- Saída:



```
Digite um valor para a posição [ 1, 1]: 1
Digite um valor para a posição [ 1, 2]: 3
Digite um valor para a posição [ 2, 1]: 4
Digite um valor para a posição [ 2, 2]: 6
Digite um valor para a posição [ 3, 1]: 2
Digite um valor para a posição [ 3, 2]: 8
```



SINTAXE NO VISUALG

○ Exibindo o conteúdo de uma matriz:

```
...  
    escreva("O valor que está na posição [1,1] é: ", numeros[1,1])  
    escreva("O valor que está na posição [1,2] é: ", numeros[1,2])  
    escreva("O valor que está na posição [2,1] é: ", numeros[2,1])  
    escreva("O valor que está na posição [2,2] é: ", numeros[2,2])  
    escreva("O valor que está na posição [3,1] é: ", numeros[3,1])  
    escreva("O valor que está na posição [3,2] é: ", numeros[3,2])  
fimalgoritmo
```



SINTAXE NO VISUALG

○ Exibindo o conteúdo de uma matriz

- Ou podemos utilizar um laço de repetição para facilitar a exibição dos valores de uma matriz
- Criando um laço para percorrer as linhas:

○ Exemplo:

```
para i de 1 ate 3 faca
    escreval("O valor que está na posição [", i, ", 1] é: ", numeros[i, 1])
    escreval("O valor que está na posição [", i, ", 2] é: ", numeros[i, 2])
fimpara
```



SINTAXE NO VISUALG

○ Exibindo o conteúdo de uma matriz

- E podemos ainda incluir um laço de repetição para variar pelas colunas também, por exemplo:

```
para i de 1 ate 3 faca
  para j de 1 ate 2 faca
    escreval("O valor que está na posição [", i,",", " j,"] é: ", numeros[i, j])
  fimpara
fimpara
```



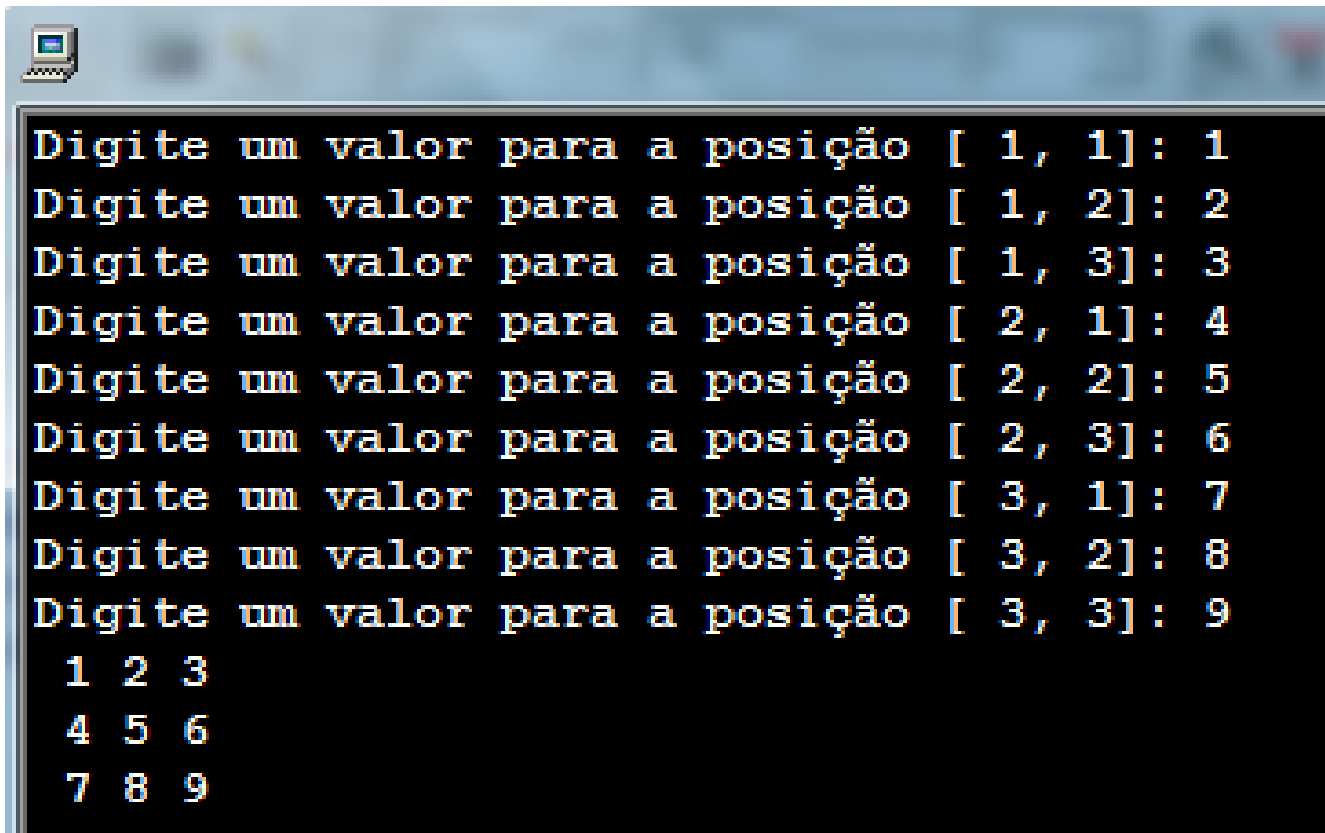
EXEMPLO 1

- Criar um algoritmo que leia uma matriz 3x3 e exiba a matriz preenchida:

```
algoritmo "exemplo01"  
var  
  numeros: vetor[1..3, 1..3] de inteiro  
  i, j: inteiro  
inicio  
  para i de 1 ate 3 faca  
    para j de 1 ate 3 faca  
      escreva("Digite um valor para a posição [", i, ",", j, "]: ")  
      leia(numeros[i,j])  
    fimpara  
  fimpara  
  
  para i de 1 ate 3 faca  
    para j de 1 ate 3 faca  
      escreva(numeros[i, j])  
    fimpara  
  escreval  
  fimpara  
finalgoritmo
```

EXEMPLO 1

- Saída:



```
Digite um valor para a posição [ 1, 1]: 1
Digite um valor para a posição [ 1, 2]: 2
Digite um valor para a posição [ 1, 3]: 3
Digite um valor para a posição [ 2, 1]: 4
Digite um valor para a posição [ 2, 2]: 5
Digite um valor para a posição [ 2, 3]: 6
Digite um valor para a posição [ 3, 1]: 7
Digite um valor para a posição [ 3, 2]: 8
Digite um valor para a posição [ 3, 3]: 9

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



EXEMPLO 2

- Criar um algoritmo que leia uma matrizes 3x3. Em seguida, exiba a som dos elementos de cada uma das linhas. Ex:

1	2	2
3	2	3
4	1	1

Soma Linha 1 = 5

Soma Linha 2 = 8

Soma Linha 3 = 6



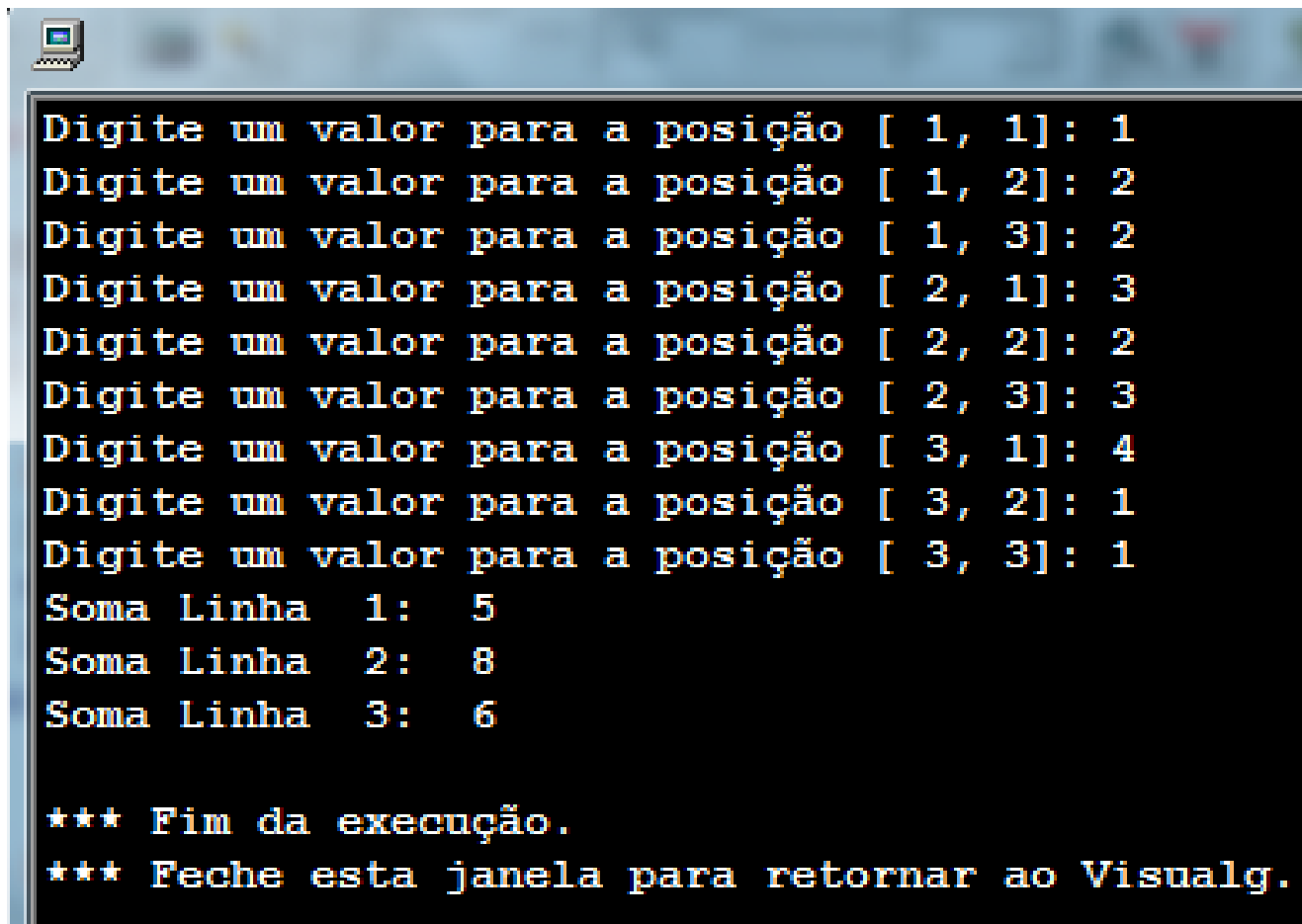
EXEMPLO 2

○ Resolução:

```
algoritmo "exemplo01"  
var  
  numeros: vetor[1..3, 1..3] de inteiro  
  i, j: inteiro  
  soma: inteiro  
inicio  
  para i de 1 ate 3 faca  
    para j de 1 ate 3 faca  
      escreva("Digite um valor para a posição [", i, ",", j, "]: ")  
      leia(numeros[i,j])  
    fimpara  
  fimpara  
  
  para i de 1 ate 3 faca  
    soma <- 0  
    para j de 1 ate 3 faca  
      soma <- soma + numeros[i, j]  
    fimpara  
    escreval ("Soma Linha ", i, ": ", soma)  
  fimpara  
fimalgoritmo
```

EXEMPLO 2

- Saída:



```
Digite um valor para a posição [ 1, 1]: 1
Digite um valor para a posição [ 1, 2]: 2
Digite um valor para a posição [ 1, 3]: 2
Digite um valor para a posição [ 2, 1]: 3
Digite um valor para a posição [ 2, 2]: 2
Digite um valor para a posição [ 2, 3]: 3
Digite um valor para a posição [ 3, 1]: 4
Digite um valor para a posição [ 3, 2]: 1
Digite um valor para a posição [ 3, 3]: 1
Soma Linha 1: 5
Soma Linha 2: 8
Soma Linha 3: 6

*** Fim da execução.
*** Feche esta janela para retornar ao Visualg.
```



EXEMPLO 3

- Escreva um algoritmo que leia uma matriz 4x3. Em seguida, receba um novo valor do usuário e verifique se este valor se encontra na matriz. Caso o valor se encontre na matriz, escreva a mensagem “O valor se encontra na matriz”. Caso contrário, escreva a mensagem “O valor NÃO se encontra na matriz”.



EXEMPLO 3

algoritmo "exemplo03"

var

numeros: vetor[1..4, 1..3] de inteiro

i, j, buscar: inteiro

achou: logico

inicio

para i de 1 ate 4 faca

para j de 1 ate 3 faca

escreva("Digite um valor para a posição [", i, ",", j, "]: ")

leia(numeros[i,j])

fimpara

fimpara

escreva("Digite um valor para ser buscado na matriz: ")

leia(buscar)

achou <- falso

para i de 1 ate 4 faca

para j de 1 ate 3 faca

se (numeros[i,j] = buscar) entao

achou <- verdadeiro

fimse

fimpara

fimpara

se achou=verdadeiro entao

escreva("O número se encontra na matriz.")

senao

escreva("O número NÃO se encontra na matriz.")

fimse

fimalgoritmo



EXEMPLO 3

- Saída:

```
Digite um valor para a posição [ 1, 1]: 1
Digite um valor para a posição [ 1, 2]: 2
Digite um valor para a posição [ 1, 3]: 3
Digite um valor para a posição [ 2, 1]: 4
Digite um valor para a posição [ 2, 2]: 5
Digite um valor para a posição [ 2, 3]: 6
Digite um valor para a posição [ 3, 1]: 7
Digite um valor para a posição [ 3, 2]: 8
Digite um valor para a posição [ 3, 3]: 9
Digite um valor para a posição [ 4, 1]: 10
Digite um valor para a posição [ 4, 2]: 11
Digite um valor para a posição [ 4, 3]: 12
Digite um valor para ser buscado na matriz: 7
O número se encontra na matriz.
*** Fim da execução.
*** Feche esta janela para retornar ao Visualg.
```

```
Digite um valor para a posição [ 1, 1]: 1
Digite um valor para a posição [ 1, 2]: 2
Digite um valor para a posição [ 1, 3]: 3
Digite um valor para a posição [ 2, 1]: 4
Digite um valor para a posição [ 2, 2]: 5
Digite um valor para a posição [ 2, 3]: 6
Digite um valor para a posição [ 3, 1]: 7
Digite um valor para a posição [ 3, 2]: 8
Digite um valor para a posição [ 3, 3]: 9
Digite um valor para a posição [ 4, 1]: 10
Digite um valor para a posição [ 4, 2]: 11
Digite um valor para a posição [ 4, 3]: 12
Digite um valor para ser buscado na matriz: 15
O número NÃO se encontra na matriz.
*** Fim da execução.
*** Feche esta janela para retornar ao Visualg.
```



EXERCÍCIOS

1. Crie um algoritmo que leia uma matriz 5x5. Em seguida, conte quantos números pares existem na matriz.
2. Crie um algoritmo que leia uma matriz 3x3 e calcule a soma dos valores das colunas da matriz. Ex:

1	2	2
3	2	3
4	1	1

Soma Coluna 1 = 8

Soma Coluna 2 = 5

Soma Coluna 3 = 6



EXERCÍCIOS

3. Crie um algoritmo que calcule a média dos elementos de uma matriz 5x2.
4. Crie um algoritmo informe qual o maior e qual o menor elemento existente em uma matriz 6x3.
5. Crie um algoritmo que leia uma matriz 3x3 e crie uma segunda matriz que inverta as linhas e colunas da primeira matriz. Ex:

Matriz

1	2	3
4	5	6
7	8	9

Matriz Invertida

1	4	7
2	5	8
3	6	9



EXERCÍCIOS

6. Crie um algoritmo que leia duas matrizes 2x5 e crie uma terceira matriz também 2x5 com o valor da soma dos elementos de mesmo índice. Ex:

Matriz1 + Matriz2 = Matriz3

1	2
3	2
4	1
5	5
1	2

2	4
5	3
7	7
4	4
1	9

3	6
8	5
11	8
9	9
2	11



EXERCÍCIOS

7. Crie um algoritmo que calcule a soma dos valores da diagonal principal de uma matriz 5x5. Veja a **diagonal principal** da matriz destacada no exemplo abaixo:

1	2	5	1	4
3	2	4	2	3
4	1	2	3	7
5	5	2	4	9
1	2	4	5	1

SOMA = 10



EXERCÍCIOS

8. Crie um algoritmo que verifique se uma matriz é **triangular superior**. Uma matriz é triangular superior se todos os elementos abaixo da **diagonal principal** são iguais a 0.

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



EXERCÍCIOS

9. Crie um algoritmo que verifique se uma matriz é **triangular inferior**. Uma matriz é triangular inferior se todos os elementos abaixo da **diagonal principal** são iguais a 0.

1	0	0	0	0
3	2	0	0	0
4	1	2	0	0
5	5	2	4	0
1	2	4	5	1

