

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

# AULA 10

## VETORES

Disciplina: Algoritmos

Professora: Alba Lopes

[alba.lopes@ifrn.edu.br](mailto: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



nomes



# 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:

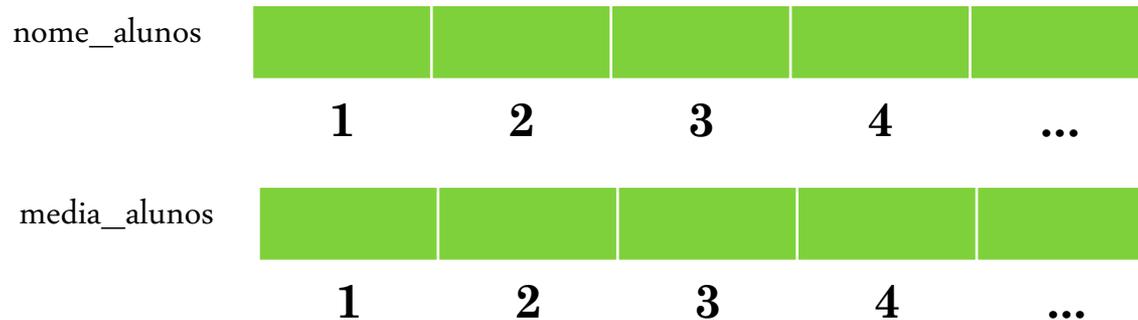
```
algoritmo "exemplo_vetores"  
var  
    nome_alunos: vetor [1..50] de caractere  
    media_alunos: vetor [1..50] de real  
inicio  
    nome_alunos[1] ← "Pedro"  
    leia(nome_alunos[2])  
    nome_alunos[3] := "Joana"  
    media_alunos[1] := 8.5
```



# SINTAXE NO VISUALG

## ○ Exemplo:

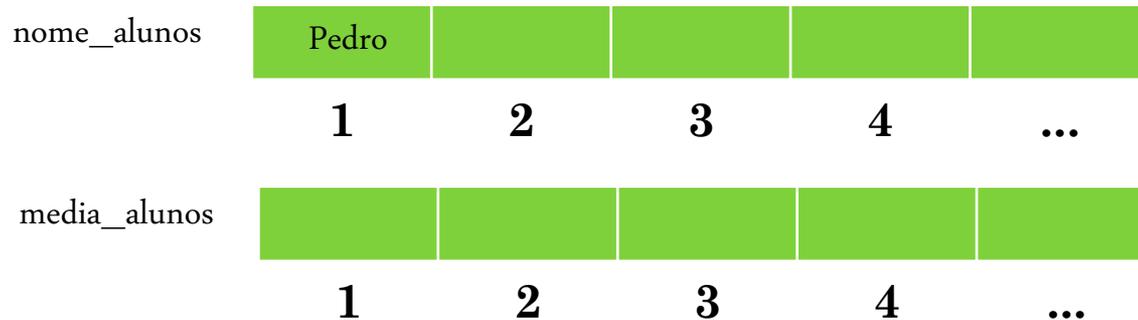
```
algoritmo "exemplo_vetores"  
var  
    nome_alunos: vetor [1..50] de caractere  
    media_alunos: vetor [1..50] de real  
inicio  
    nome_alunos[1] ← "Pedro"  
    leia(nome_alunos[2])  
    nome_alunos[3] := "Joana"  
    media_alunos[1] := 8.5
```



# 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
```

nome_alunos	Pedro	Maria			
	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					
	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])  
fimalgoritmo
```



# 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

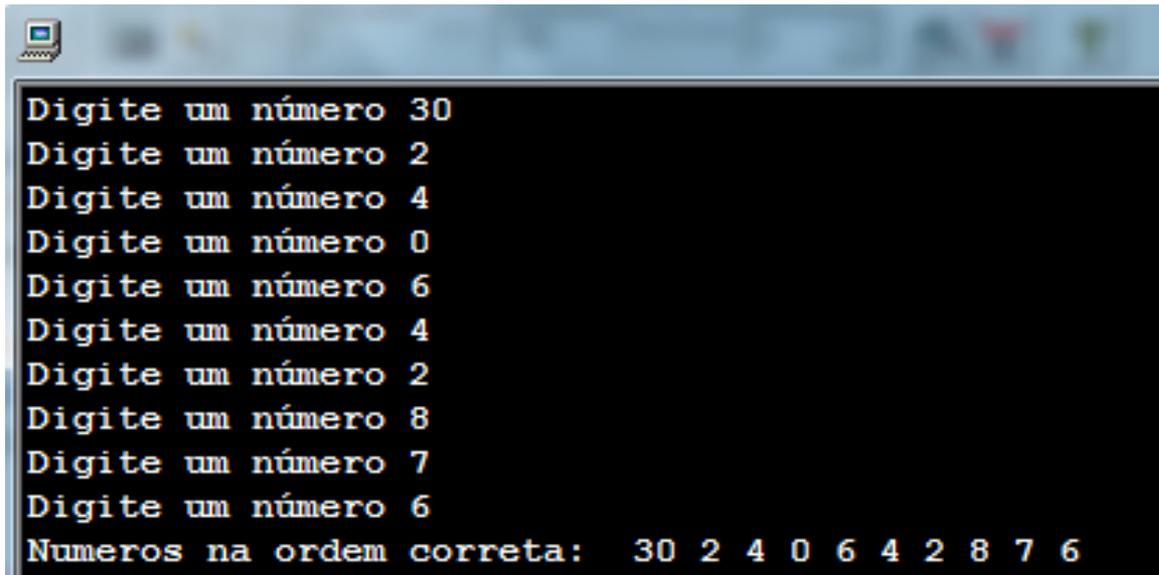
- Criar um algoritmo que leia 10 números pelo teclado e exiba os números na ordem correta que os números foram digitados.

```
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:



```
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

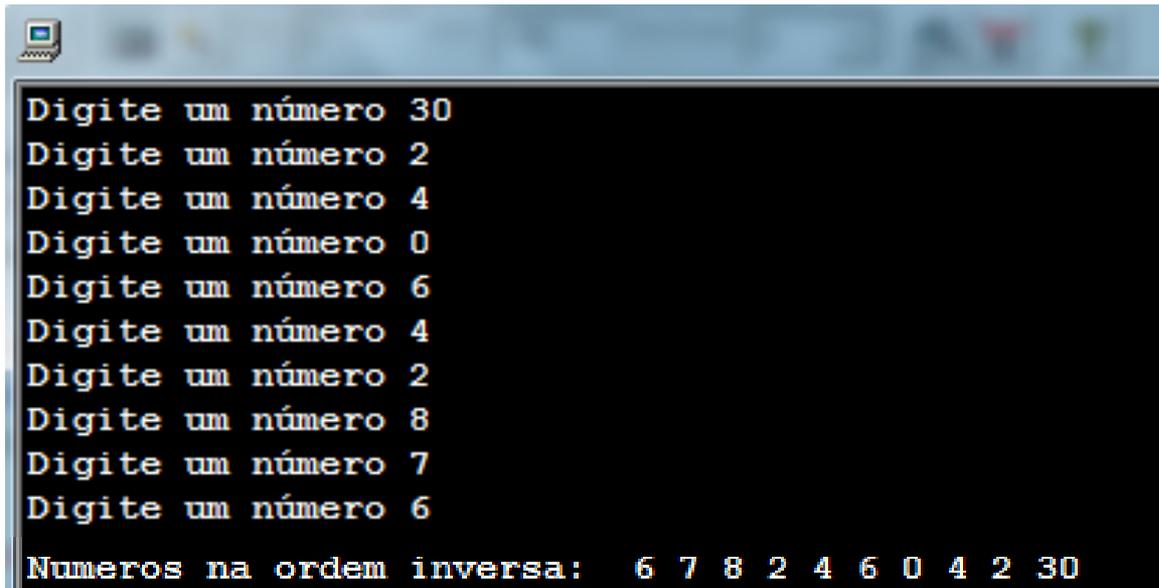
- Criar um algoritmo que leia 10 números pelo teclado e exiba os números na ordem inversa da que os números foram digitados.

```
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

4. 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

5. 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



# EXERCÍCIOS

6. Crie um algoritmo que armazene em um vetor os salários dos 10 funcionários de uma empresa. Em seguida, reajuste o salário de todos os funcionários em 5%.
7. Escreva um algoritmo que leia um vetor de 10 posições e mostre-o. Em seguida, troque o primeiro elemento com o último, o segundo com o penúltimo, o terceiro com o antepenúltimo, e assim sucessivamente. Mostre o novo vetor depois da troca. Ex:

ANTES →	<b>7</b>	<b>4</b>	<b>9</b>	<b>15</b>	<b>20</b>	<b>2</b>	<b>1</b>	<b>4</b>	<b>0</b>	<b>30</b>
DEPOIS →	<b>30</b>	<b>0</b>	<b>4</b>	<b>1</b>	<b>2</b>	<b>20</b>	<b>15</b>	<b>9</b>	<b>4</b>	<b>7</b>

