

R

Introdução ao
Mundo da
Computação e
Lógica de
Programação

Lógica de Programação

Introdução ao
Mundo da
Computação e
Lógica de
Programação

Vetores

Vetores

Vetor (**array** uni-dimensional) é uma variável que armazena várias variáveis do mesmo tipo. No problema apresentado anteriormente, nós podemos utilizar um vetor de 50 posições para armazenar os nomes dos 50 alunos.

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.

Índices do Vetor						
Dados	1	2	3	4	5	6
	12	37	49	21	67	30

Vetores

Sintaxe:

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

Exemplo:

```
algoritmo "exemplo_vetores"
```

```
var
```

```
    nome_alunos:  vetor [1..10]  de caractere
```

```
    media_alunos: vetor [1..10]  de real inicio
```

Intervalo de valores



Nome do vetor



É preciso colocar os dois ponto ".."



Vetores

Inserindo informações em uma determinada posição do vetor.

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	Carla	Joana	
-------	-------	-------	--

media_alunos

8.5			
-----	--	--	--

Vetores

Preenchendo um vetor

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

Exemplo:

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



Vetores

Exibindo o conteúdo de um vetor

Podemos utilizar um laço de repetição para facilitar a exibição dos valores de um vetor

Exemplo:

```
algoritmo "exibindo_vetores"
```

```
var
```

```
    numeros: vetor [1..5] de inteiro
```

```
    i: inteiro
```

```
inicio
```

```
para i de 1 ate 5 faca
```

```
    escreva("O valor da posição ", i, " é: ", numeros[i])
```

```
fimpara
```

```
fimalgoritmo
```


Exercício de Vetores

Escreva um algoritmo que solicite ao usuário a entrada de 5 números, e que exiba o somatório desses números na tela.

Após exibir a soma, o programa deve mostrar também os números que o usuário digitou, um por linha.



Gabarito do Exercício de Vetores

Algoritmo "preenchendo_vetor"

Var

numeros : vetor [0..4] de inteiro

cont, soma : inteiro

Inicio

para cont de 0 ate 4 faca

 escreval("Entre com um número:")

 leia(numeros[cont])

 soma <- soma + numeros[cont])

fimpara

escreval("A soma dos números é", soma)

escreval()

escreval("Os números digitados foram:")

para cont de 0 ate 4 faca

 escreval(numeros[cont])

fimpara

Fimalgoritmo



Introdução ao
Mundo da
Computação e
Lógica de
Programação

Matrizes

Matrizes

É 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. Já a matriz possui mais de uma dimensão, ou seja, mais de uma linha e coluna.

		Índices das Colunas		
		1	2	3
Índices das Linhas	1	1.7	4.9	2.8
	2	5.3	9.1	7.4

Matrizes

Sintaxe:

<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**.

Matrizes

Preenchendo uma Matriz manualmente

Exemplo:

```
algoritmo "exemplo_matriz"
```

```
var
```

```
exMatriz: vetor[1..3, 1..2] de inteiro
```

Início

```
exMatriz[1,1] <- 10
```

```
leia (exMatriz[1,2])
```

```
exMatriz[3,1] <- 4
```

```
fimalgoritmo
```

exMatriz

10	37
4	

Linhas (3)

Colunas (2)

Matrizes

Preenchendo uma Matriz via teclado

Exemplo:

```
algoritmo "preencher_matrizes"
var
    numeros: vetor[1..3, 1..2] de inteiro
    i: inteiro
inicio
para i de 1 ate 3 faca
    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
```



Funções

É um subprograma que **retorna** um valor. De modo análogo aos procedimentos, sua declaração deve estar entre o final da declaração de variáveis e a linha início do programa principal, e segue a sintaxe abaixo:

```
funcao <nome-de-função> [ (<parâmetros>) ] : <tipo-de-dado>  
início  
<comandos>  
fimfuncao
```


Funções

Exemplo:

Algoritmo "func"

Var

n, m, res : inteiro

funcao soma : inteiro

var aux: inteiro

inicio

aux <- n + m

retorne aux

fimfuncao

Inicio

n <- 4

m <- 9

res <- soma // chama a função

escreva(res)

Fimalgoritmo



Procedimentos

É um subprograma que **não** retorna um valor. De modo análogo aos procedimentos, sua declaração deve estar entre o final da declaração de variáveis e a linha início do programa principal, e segue a sintaxe abaixo:

```
procedimento <nome-de-procedimento> [(<parâmetros>)] : <tipo-de-dado>  
início  
<comandos>  
fimprocedimento
```



Procedimentos

Exemplo:

Algoritmo "proc"

Var

n, m, res : inteiro

procedimento soma

var aux: inteiro

inicio

aux <- n + m

res <- aux

escreva(res)

fimprocedimento

Inicio

n <- 4

m <- 9

Soma // chama o procedimento

Fimalgoritmo

Funções

Variáveis Globais

São aquelas declaradas no início de um algoritmo. São visíveis, ou seja, podem ser utilizadas no algoritmo principal e por todos os outros subalgoritmos.

Variáveis Locais

São aquelas declaradas no início de um subalgoritmo. São visíveis, ou seja, podem ser utilizadas somente pelo subalgoritmo onde foram declaradas. Outros subalgoritmos ou mesmo o algoritmo principal não podem utilizá-las.



Funções

Parâmetros

São canais por onde os dados são transferidos pelo algoritmo chamador a um subalgoritmo, e vice-versa.

Parâmetros Formais

São os nomes simbólicos usados na definição dos parâmetros de um subalgoritmo.

Parâmetros Reais

São aqueles que substituem os parâmetros formais quando da chamada de um subalgoritmo.



Funções

Mecanismos de Passagem de Parâmetros

A substituição dos parâmetros formais pelos parâmetros reais no ato da invocação de um subalgoritmo é denominada de passagem de parâmetros e pode se dar por dois mecanismos distintos: passagem por valor (ou por cópia) e passagem por referência.

Passagem de Parâmetros por Valor

No ato da invocação do subalgoritmo, o parâmetro real é calculado e uma cópia do seu valor é substituída pelo parâmetro formal. Quando o subalgoritmo é executado, as modificações efetuadas no parâmetro formal não afetam o parâmetro real, pois trabalha-se apenas com uma cópia do mesmo.



Funções

Mecanismos de Passagem de Parâmetros

Algoritmo "func_param"

Var

n, m, res : inteiro

funcao soma (n,m : inteiro): inteiro

var aux: inteiro

inicio

aux <- n + m

retorne aux

fimfuncao

Inicio

n <- 7

m <- 3

res <- soma (n, m) // chama a função passando os parâmetros

escreva(res)

Fimalgoritmo



f i t @rederecode | y @recoderede

<https://recode.org.br>