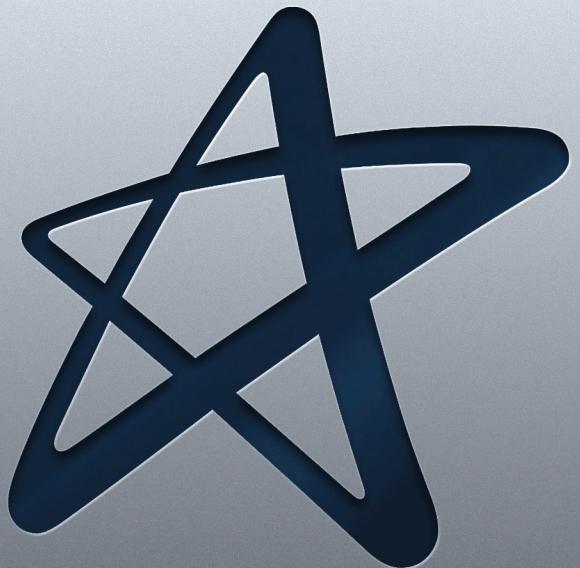


Técnicas de Desenvolvimento de Algoritmos



Cruzeiro do Sul Virtual
Educação a distância

Material Teórico



Vetores e Matrizes

Responsável pelo Conteúdo:

Prof.^a Me. Ana Fernanda Gomes Ascencio

Revisão Textual:

Jaquelina Kutsunugi

Revisão Técnica:

Prof.^a Esp. Margarete Eliane da Silva Almendro

UNIDADE

Vetores e Matrizes



- Conceito de Vetor e Matriz



OBJETIVO DE APRENDIZADO

- Desenvolver algoritmos em pseudocódigo que utilizem as estruturas de dados do tipo vetores e matrizes.

Orientações de estudo

Para que o conteúdo desta Disciplina seja bem aproveitado e haja maior aplicabilidade na sua formação acadêmica e atuação profissional, siga algumas recomendações básicas:



Determine um horário fixo para estudar.

Procure manter contato com seus colegas e tutores para trocar ideias! Isso amplia a aprendizagem.

Aproveite as indicações de Material Complementar.

Conserve seu material e local de estudos sempre organizados.

Seja original!
Nunca plagie trabalhos.

Assim:

- ✓ Organize seus estudos de maneira que passem a fazer parte da sua rotina. Por exemplo, você poderá determinar um dia e horário fixos como seu “momento do estudo”;
- ✓ Procure se alimentar e se hidratar quando for estudar; lembre-se de que uma alimentação saudável pode proporcionar melhor aproveitamento do estudo;
- ✓ No material de cada Unidade, há leituras indicadas e, entre elas, artigos científicos, livros, vídeos e sites para aprofundar os conhecimentos adquiridos ao longo da Unidade. Além disso, você também encontrará sugestões de conteúdo extra no item **Material Complementar**, que ampliarão sua interpretação e auxiliarão no pleno entendimento dos temas abordados;
- ✓ Após o contato com o conteúdo proposto, participe dos debates mediados em fóruns de discussão, pois irão auxiliar a verificar o quanto você absorveu de conhecimento, além de propiciar o contato com seus colegas e tutores, o que se apresenta como rico espaço de troca de ideias e de aprendizagem.



Não se esqueça de se alimentar e de se manter hidratado.



Conceito de Vetor e Matriz

Nesta Unidade VI, continuaremos conhecendo um pouco mais sobre as estruturas que compõem os algoritmos. Salientamos que a implementação dos algoritmos utilizando qualquer Linguagem de Programação permite o teste efetivo destes. Neste curso, os testes serão realizados na ferramenta VisualG.

Na Unidade I, abordamos as definições de algoritmos e as três técnicas mais utilizadas para o desenvolvimento destes. Na Unidade II, foram abordados alguns algoritmos que utilizam apenas estrutura sequencial, ou seja, entrada dos dados, processamento e saída. Na Unidade III, abordamos alguns algoritmos que utilizam estruturas condicionais, também conhecidas como desvios. Na Unidade IV, abordamos alguns algoritmos que utilizam estruturas de repetição, em que um ou vários comandos podem ser repetidos várias vezes. Na unidade V, abordamos as sub-rotinas, ou seja, a modularização dos algoritmos.

A partir desta unidade, abordaremos dois novos tipos de dados, conhecidos como vetores e matrizes.

Um vetor é também conhecido como variável composta homogênea unidimensional. Isto quer dizer que se trata de um conjunto de variáveis de mesmo tipo, que possuem o mesmo identificador (nome) e são alocadas sequencialmente na memória. Como as variáveis têm o mesmo nome, o que as distingue é um índice que referencia sua localização dentro da estrutura. (ASCENCIO; CAMPOS, 2012, p. 151)

O vetor é declarado juntamente com as outras variáveis, conforme sintaxe a seguir:

```
var nome_da_variável: vetor[posição_inicial..posição_final] de tipo_dos_dados
```

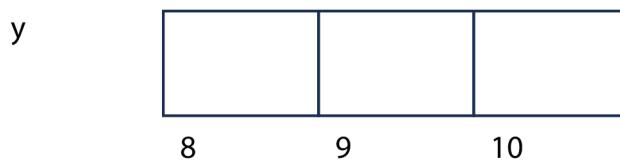
A seguir, alguns exemplos de vetores:

```
var vet: vetor[1..5] de inteiro
```



Acima, podemos ver a criação de um vetor chamado `vet`, que possui cinco posições. Ou seja, foram alocadas cinco porções de memória para armazenamento de números inteiros. Estas porções de memória são contíguas, isto é, seus endereços são sequenciais.

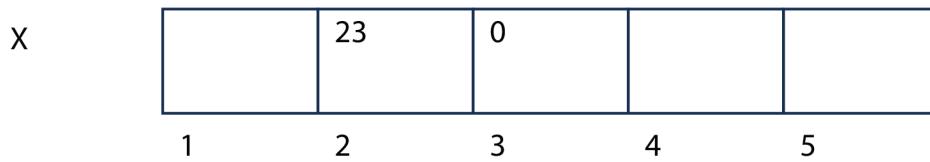
`var y: vetor[8..10] de real`



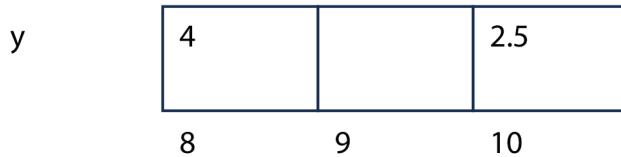
Acima, é possível ver a criação de um vetor chamado `y`, que possui três posições. Ou seja, foram alocadas três porções de memória para armazenamento de números reais. Essas porções de memória são contíguas, isto é, seus endereços são sequenciais.

Para atribuir valores às posições dos vetores, é necessário colocar o nome do vetor e a posição entre colchetes, conforme exemplos abaixo:

`X[2] <- 23`
`X[3] <- 0`



`y[10] <- 2.5`
`y[8] <- 4`



É possível utilizar a estrutura de repetição com variável de controle para carregar todas as posições de um vetor, conforme exemplo a seguir:

```
para i <- 1 até 5 faça [passo 1]
  escreval ("Digite o „,i,“ ° número")
  leia (x[i])
fimpara
```

Nesse exemplo, a estrutura de repetição PARA foi utilizada para garantir que a variável `i` assuma todos os valores possíveis entre 1 e 5 (posições válidas para o vetor `X`). Assim, para cada execução da repetição, será utilizada uma posição diferente do vetor.

Quadro 1 - Simulação

Simulação						
	Memória					Tela
i = 1	x	95				Digite o 1º número 95
i = 2	x	95	13			Digite o 2º número 13
i = 3	x	95	13	-25		Digite o 3º número 25
i = 4	x	95	13	-25	47	Digite o 4º número 47
i = 5	x	95	13	-25	47	Digite o 5º número 0

Fonte: Ascencio e Campos (2012, p. 152)

Da mesma forma que utilizamos a estrutura de repetição para passar por todas as posições de um vetor, a fim de carregá-las com dados digitados pelo usuário, usamos a estrutura de repetição para mostrar os dados armazenados no vetor, conforme exemplo abaixo:

```

para i <- 1 até 5 faça [passo 1]
    escreval ("Conteúdo da posição „,i,“ do vetor ")
    escreval (x[i])
fimpara
  
```



Os vetores são conhecidos como variáveis homogêneas unidimensionais, ou seja, são variáveis com o mesmo nome e o mesmo tipo, mas diferenciadas por um único índice. Assim, não podemos usar um vetor para armazenar os dados de um produto, pois os dados de um produto são: código do tipo inteiro, descrição do tipo literal e preço do tipo real.

A seguir, há um exemplo de algoritmo utilizando vetor. O algoritmo deve receber dez números digitados pelo usuário e mostrar qual o menor.

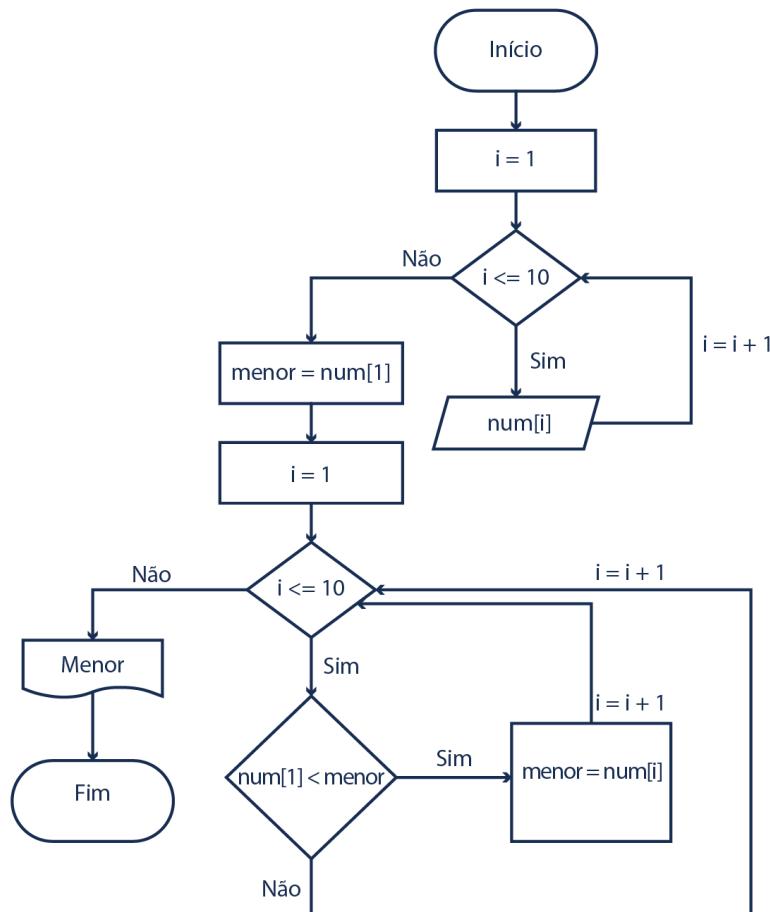


Figura 1 - Fluxograma do algoritmo exemplo de vetor

A seguir, temos um algoritmo exemplo em pseudocódigo:

algoritmo “exemplo1”

```

// Função: receber 10 números e mostrar o menor
// Autor: Ana Fernanda /*revisão Margarete E. S. Almendro
// Data: 24/12/2018      /*07/02/2020
// seção de declarações de variáveis e constantes
  
```

```
var num: vetor[1..10] de inteiro
i, menor: inteiro
inicio
    para i <- 1 até 10 faça [passo 1]
        leia (num[i])
    fimpara
    menor <- num[1]
    para i <- 1 até 10 faça [passo 1]
        se (num[i] < menor) entao
            menor <- num[i]
        fimse
    fimpara
    escreval (menor)
fimalgoritmo
```



Quando falamos em vetores em algoritmos e em programação de computadores, estamos falando de uma variável capaz de armazenar uma quantidade finita de variáveis, todas com o mesmo nome e do mesmo tipo, diferenciadas apenas por um índice. Não estamos falando dos vetores da GEOMETRIA ANALÍTICA.

Uma matriz é uma variável composta homogênea multidimensional. Ela é formada por uma sequência de variáveis, todas do mesmo tipo, com o mesmo identificador (mesmo nome), e alocadas sequencialmente na memória. Uma vez que as variáveis têm o mesmo nome, o que as distingue são índices que referenciam sua localização dentro da estrutura. Uma variável do tipo matriz precisa de um índice para cada uma de suas dimensões. (ASCENCIO; CAMPOS, 2012, p. 194)

A matriz é declarada juntamente com as outras variáveis, conforme sintaxe a seguir:

```
var nome_da_variável: vetor[posição_inicial..posição_final, posição_inicial..posição_final] de tipo_dos_dados
```

A seguir alguns exemplos de matrizes:

```
var mat: vetor[1..3, 1..5] de inteiro
```

	1				
mat	2				
	3				
	1	2	3	4	5

Acima, podemos ver a criação de uma matriz chamada mat, que possui 15 posições, ou seja, possui 3 linhas e 5 colunas.

```
var X: vetor[1..2, 4..5] de real
```

X	1			
	2			
	4	5		

Acima, podemos ver a criação de uma matriz chamada X, que possui 4 posições, ou seja, possui 2 linhas e 2 colunas.

Para atribuir valores às posições das matrizes, é necessário colocar o nome da matriz e suas posições (linha e coluna) entre colchetes, conforme exemplos abaixo:

```
mat[1,2] <- 8
mat[2,1] <- 5
mat[3,5] <- 9
```

	1				
mat	2	8			
	5				
	3				9
	1	2	3	4	5

É possível utilizar estruturas de repetição aninhadas com variável de controle para carregar todas as posições de uma matriz, conforme exemplo a seguir:

```
// a variável i será utilizada como primeiro índice, ou
seja, linha

// a variável j será utilizada como segundo índice, ou
seja, coluna

para i <- 1 até 3 faça [passo 1]

    para j <- 1 até 5 faça [passo 1]

        escreval ("Digite o número da linha",i," coluna ",j)
        leia (mat[i,j])

    fimpara

fimpara
```

Os valores assumidos pela variável i estão dentro do intervalo de 1 a 3, ou seja, exatamente o número das linhas da matriz. Por esta razão, a variável i é utilizada para indicar a primeira dimensão, dentro dos colchetes. Para cada valor assumido por i, a variável j assume os valores no intervalo de 1 a 5, ou seja, exatamente o número das colunas. Por esta razão, a variável j é utilizada para indicar a segunda dimensão, dentro dos colchetes. (ASCENCIO; CAMPOS, 2012, p. 196)

Quadro 2 - Simulação das estruturas de repetição para carregar uma matriz

MEMÓRIA		TELA
i	j	
1	1	Digite o número da linha 1 e coluna 1: 12
	2	Digite o número da linha 1 e coluna 2: 9
	3	Digite o número da linha 1 e coluna 3: 3
	4	Digite o número da linha 1 e coluna 4: 7
	5	Digite o número da linha 1 e coluna 5: -23
2	1	Digite o número da linha 2 e coluna 1: 15
	2	Digite o número da linha 2 e coluna 2: 4
	3	Digite o número da linha 2 e coluna 3: 2

MEMÓRIA		TELA
	4	Digite o número da linha 2 e coluna 4: 34
	5	Digite o número da linha 2 e coluna 5: -4
3	1	Digite o número da linha 3 e coluna 1: 3
	2	Digite o número da linha 3 e coluna 2: 45
	3	Digite o número da linha 3 e coluna 3: 3
	4	Digite o número da linha 3 e coluna 4: 0
	5	Digite o número da linha 3 e coluna 5: -3

Fonte: Ascencio e Campos (2012, p. 196)

Assim, podemos imaginar os elementos dispostos em uma estrutura bidimensional, como uma tabela.

Matriz mat	1	12	9	3	7	-23
	2	15	4	2	34	-4
	3	3	45	3	0	-3
		1	2	3	4	5

Da mesma forma que utilizamos estruturas de repetição aninhadas para passar por todas as posições de uma matriz para carregá-las com dados digitados pelo usuário, usamos estruturas de repetição aninhadas para mostrar os dados armazenados na matriz, conforme exemplo abaixo:

```

para i <- 1 até 3 faça [passo 1]
    para j <- 1 até 5 faça [passo 1]
        escreval ("Conteúdo da linha ", i, " coluna ", j)
        escreval (mat[i,j])
    fimpara
fimpara
  
```



As matrizes são conhecidas como variáveis homogêneas multidimensionais, ou seja, são variáveis com o mesmo nome e o mesmo tipo, mas diferenciadas por seus índices. Assim, não podemos usar uma matriz para armazenar os dados de um aluno, pois os dados de um aluno são: código do inteiro e nome do tipo literal.

A seguir, há um exemplo de algoritmo utilizando matriz. O algoritmo deve receber 6 números digitados pelo usuário, armazenando-os em uma matriz 3 X 2. Em seguida, deve mostrar qual o maior número digitado.

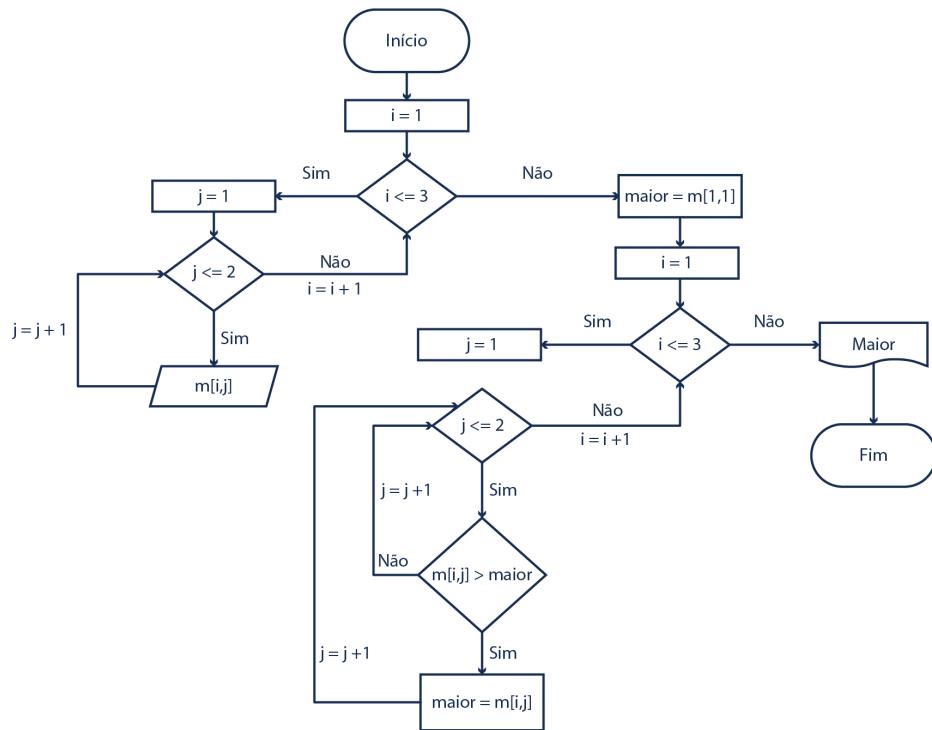


Figura 2 - Fluxograma do algoritmo exemplo de matriz

A seguir, há um algoritmo exemplo em pseudocódigo:

algoritmo "exemplo2"

```

// Função: receber 6 números, armazenar em uma matriz
// mostrar o maior

// Autor: Ana Fernanda /*revisão Margarete E. S. Almendro
// Data: 24/12/2018      /*07/02/2020

// seção de declarações de variáveis e constantes
var m: vetor[1..3, 1..2] de inteiro
i, j, maior: inteiro
inicio
    para i <- 1 até 3 faça [passo 1]
        para j <- 1 até 2 faça [passo 1]
            leia (m[i,j])
        fimpara
    fimpara
    maior = m[1,1]
    para i <- 1 até 3 faça [passo 1]
        para j <- 1 até 2 faça [passo 1]
            se m[i,j] > maior entao
                maior = m[i,j]
            fimse
        fimpara
    fimpara
    escreva (maior)
fim
  
```

```

fimpara
maior <- m[1,1]
para i <- 1 até 3 faça [passo 1]
    para j <- 1 até 2 faça [passo 1]
        se (m[i,j] > maior) então
            maior <- m[i,j]
        fimse
    fimpara
escreval (maior)
fimalgoritmo
  
```



Quando falamos em matrizes em algoritmos e em programação de computadores, estamos falando de uma variável capaz de armazenar uma quantidade finita de variáveis, todas com o mesmo nome e do mesmo tipo, diferenciadas apenas por seus índices. Não estamos falando das matrizes da GEOMETRIA ANALÍTICA, apesar de ambas apresentarem a mesma estrutura de armazenamento.

Observe a Figura 3, a seguir, que representa as duas estruturas estudadas nesta Unidade VI, ou seja, vetores e matrizes.

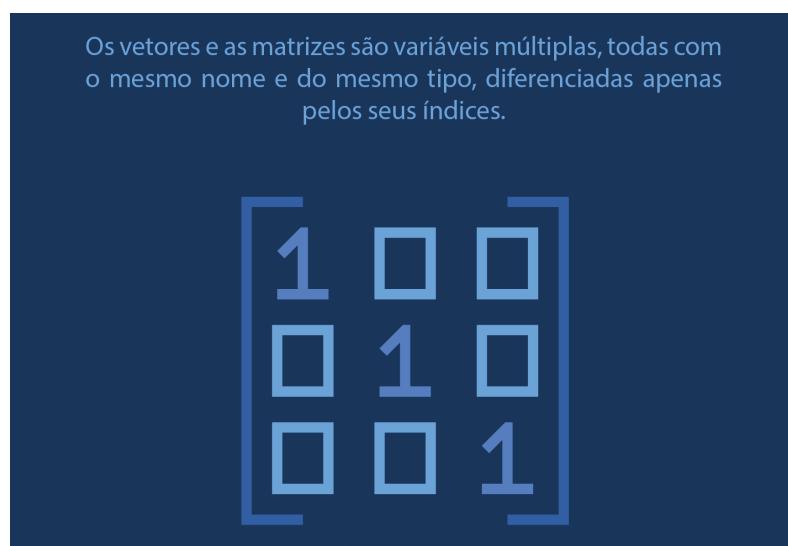


Figura 3 - Vetores e Matrizes

Fonte: sn333g / 123RF

Com a utilização das estruturas do tipo vetores e matrizes, podemos desenvolver algoritmos que necessitem do armazenamento, mesmo que temporário, de conjuntos de dados, sem a necessidade de declarar uma variável específica para cada elemento do conjunto.

Material Complementar

Indicações para saber mais sobre os assuntos abordados nesta Unidade:



Livros

Lógica de Programação com Pascal

O livro **Lógica de Programação com Pascal**, no Capítulo 7, mostra a teoria sobre vetores e matrizes e vários exemplos. Ao final desse capítulo, há uma lista de exercícios a resolver.

SCENCIO, A. F. G. **Lógica de Programação com Pascal**. São Paulo: Pearson, 1999.

Fundamentos da Programação de Computadores

O livro **Fundamentos da Programação de Computadores**, nos Capítulos 7 e 8, mostra a teoria sobre vetores e matrizes e vários exemplos. Ao final desses capítulos, existem duas listas de exercícios, uma resolvida e a outra a resolver.

ASCENCIO, A. F. G.; CAMPOS, E. A. V. **Fundamentos da Programação de Computadores**. 3 ed. São Paulo: Pearson, 2012.

Algoritmos: Teoria e Prática

O livro **Algoritmos, Teoria e Prática**, do autor Thomas Cormen, é um clássico na área de Computação e aborda o desenvolvimento de algoritmos na teoria e na prática.

CORMEN, T. H. **Algoritmos, Teoria e Prática**. Rio de Janeiro: Campus, 2012.

Algoritmos

O livro **Algoritmos**, de José Augusto Manzano, é muito indicado para quem está começando a desenvolver algoritmos, tendo em vista que possível sequência didática de aprendizagem e uma linguagem apropriada para iniciantes.

MANZANO, J. A. N. G. **Algoritmos**. 28 ed. São Paulo: Pearson, 2016.

Referência

ASCENCIO, A. F. G.; CAMPOS, E. A. V. **Fundamentos da Programação de Computadores**. 3. ed. São Paulo: Pearson, 2012.



Cruzeiro do Sul
Educacional