

Lógica de Programação

Estruturas Condicionais e Vetores em
Algoritmos



Diretor Executivo

DAVID LIRA STEPHEN BARROS

Gerente Editorial

CRISTIANE SILVEIRA CESAR DE OLIVEIRA

Projeto Gráfico

TIAGO DA ROCHA

Autoria

IZABELLY MORAIS DE MORAIS

LEANDRO C. CARDOSO

MAX ANDRÉ DE AZEVÊDO SILVA

AUTORIA

Izabelly Moraes de Moraes

Sou licenciada em Ciência da Computação pela Universidade Federal da Paraíba (UFPB), e mestre em Ciência da Computação com ênfase em Engenharia de Software e Linguagens de Programação pela Universidade Federal de Pernambuco (UFPE). Leciono como professora formadora no Instituto Federal de Pernambuco (IFPE) e na Faculdade Pitágoras (João Pessoa, na Paraíba), onde tenho a oportunidade de transmitir minha experiência na área de Tecnologia e Educação. Por isso fui convidada pela Editora Telesapiens a integrar seu elenco de autores independentes. Estou muito feliz em poder ajudar você nesta fase de muito estudo e trabalho. Conte comigo!

Leandro C. Cardoso

Sou graduado em Comunicação Social com habilitação em Design Digital, e mestre em Tecnologias da Inteligência e Design Digital pela Pontifícia Universidade Católica de São Paulo (PUC-SP), com mais de 20 anos de experiência em direção de arte e criação. Passei por empresas como a Laureate International Universities - FMU/Fiam-Faam, a Universidade Anhembi Morumbi e o Centro Paula Souza (Fatec-Etec). Já atuei como analista de desenvolvimento pedagógico sênior, coordenador de curso técnico de Design Gráfico e revisor técnico e validador para curso EAD para clientes Laureate International Universities, DeVry Brasil, Unef, FAESF, Faculdade Positivo, Uninter e Platos Soluções Educacionais S.A. (Krotonn – Universidade Anhanguera). Além disso, sou autor de mais de 24 livros didáticos e um dos organizadores da Maratona de Criação e Design do Curso de Comunicação Visual da Etec Albert Einstein. Sou apaixonado pelo que faço e adoro transmitir minha experiência de vida àqueles que estão iniciando em suas profissões. Por isso fui convidado pela Editora Telesapiens a integrar seu elenco de autores independentes. Estou muito feliz em poder ajudar você nesta fase de muito estudo e trabalho. Conte comigo!

Max André de Azevêdo Silva

Sou formado em Ciência da Computação, pela Universidade Federal da Paraíba (UFPB), e mestre em Ciência da Computação com ênfase em Engenharia de Software pela Universidade Federal da Paraíba (UFPB). Tenho experiência na área de Desenvolvimento de Sistemas Web, Mobile e Jogos Eletrônicos. Atualmente, trabalho como analista de sistemas. Por isso fui convidado pela Editora Telesapiens a integrar seu elenco de autores independentes. Estou muito feliz em poder ajudar você nesta fase de muito estudo e trabalho. Conte comigo!

ICONOGRÁFICOS

Olá. Esses ícones irão aparecer em sua trilha de aprendizagem toda vez que:



OBJETIVO:
para o início do desenvolvimento de uma nova competência;



NOTA:
quando forem necessários observações ou complementações para o seu conhecimento;



EXPLICANDO MELHOR:
algo precisa ser melhor explicado ou detalhado;



SAIBA MAIS:
textos, referências bibliográficas e links para aprofundamento do seu conhecimento;



ACESSE:
se for preciso acessar um ou mais sites para fazer download, assistir vídeos, ler textos, ouvir podcast;



ATIVIDADES:
quando alguma atividade de autoaprendizagem for aplicada;



DEFINIÇÃO:
houver necessidade de se apresentar um novo conceito;



IMPORTANTE:
as observações escritas tiveram que ser priorizadas para você;



VOCÊ SABIA?
curiosidades e indagações lúdicas sobre o tema em estudo, se forem necessárias;



REFLITA:
se houver a necessidade de chamar a atenção sobre algo a ser refletido ou discutido sobre;



RESUMINDO:
quando for preciso se fazer um resumo acumulativo das últimas abordagens;



TESTANDO:
quando o desenvolvimento de uma competência for concluído e questões forem explicadas;

SUMÁRIO

Estruturas Condicionais se Encadeadas	10
Conceitos de Estruturas Condicionais se Encadeadas.....	10
Estrutura Condicional Encadeada	16
Repetição Interativa com Teste no Início do Laço.....	16
Repetição Iterativa.....	23
Estruturas Repetitivas com Cadeias de Caracteres.....	28
Relembrando Funções de Caracteres.....	28
Vetores.....	35
Conceito de Vetor	35

UNIDADE

03

INTRODUÇÃO

Você sabia que a área que utiliza estruturas condicionais encadeadas e iterativas e efetua as operações envolvendo vetores já é considerada um conhecimento intermediário de lógica de programação? Isso mesmo. Para isso, é importante o conhecimento prévio de algoritmização de soluções computacionais, que pode ser considerado como o básico para a programação de computadores, bem como avançar na complexidade dos programas, por meio do encadeamento de estruturas condicionais e estruturas repetitivas (ou iterativas). Em relação a essas estruturas repetitivas, é importante saber o conceito de variáveis unidimensionais lineares, mais popularmente conhecidas como vetores. É essencial destacar que, nessa altura do aprendizado, a teoria da lógica de programação será cada vez menor, pois o grande segredo do aprendizado a partir desta etapa de estudos está no praticar cada vez mais. Entendeu? Ao longo desta unidade letiva você vai mergulhar neste universo!

OBJETIVOS

Olá. Seja muito bem-vindo à Unidade 3. Nosso objetivo é auxiliar você no desenvolvimento das seguintes competências profissionais até o término desta etapa de estudos:

1. Encadear várias estruturas condicionais SE, de acordo com as necessidades de solução algorítmica.
2. Aplicar estruturas de repetição em soluções algorítmicas, envolvendo contagem e acumulação.
3. Aplicar estruturas de repetição em soluções algorítmicas, envolvendo cadeias de caracteres e variáveis alfanuméricas.
4. Explicar o conceito e as aplicações de vetores em soluções algorítmicas.

Estruturas Condicionais se Encadeadas



OBJETIVO:

Ao término deste Capítulo você será capaz de encadear várias estruturas condicionais SE, de acordo com as necessidades de solução algorítmica. Isso será fundamental para o exercício de sua profissão. E então? Motivado para desenvolver esta competência? Então vamos lá. Avante!

Conceitos de Estruturas Condicionais se Encadeadas

Além de instruções, outras estruturas condicionais podem ser inseridas dentro de uma estrutura SE. Por exemplo, imagine que seja necessário desenvolver uma lógica algorítmica capaz de receber três números inteiros e exibi-los em ordem crescente, independentemente da ordem pela qual eles foram digitados no computador. Apesar de esse problema poder ser solucionado de diversas maneiras diferentes, dificilmente, alguém conseguirá resolvê-lo sem que seja necessário recorrer-se ao encadeamento de estruturas condicionais SE, como podemos ver no exemplo a seguir.

Note que conseguimos inserir cinco estruturas condicionais SE encadeadas em três níveis hierárquicos. Em cada uma dessas estruturas tivemos a oportunidade de incluir instruções tanto na cláusula ENTÃO quanto na cláusula SENÃO (MORAIS; AZEVEDO, 2017a).

```
ALGORITMO
VAR
A, B, C: INTEIRO
LEIA (A, B, C)
SE A < B ENTÃO
    SE B < C ENTÃO
        ESCREVA (A, B, C)
    SENÃO
        SE C < A ENTÃO
            ESCREVA (C, A, B)
        SENÃO
            ESCREVA (A, C, B)
    FIMSE
FIMSE
SENÃO
    SE A < C ENTÃO
        ESCREVA (B, A, C)
    SENÃO
        SE C < B ENTÃO
            ESCREVA (C, B, A)
        SENÃO
            ESCREVA (B, C, A)
    FIMSE
FIMSE
FIMSE
FINALGORITMO
```



IMPORTANTE:

É imprescindível utilizar o método da endentação do texto para que você não se perca quanto aos fechamentos das estruturas encadeadas: para cada SE tem que haver um FIMSE, e convém que estes se posicionem na mesma margem dos seus respectivos SEs.

Para compreendermos melhor a solução dada anteriormente, vamos representar o algoritmo visualmente por meio de diagramas de bloco (ou fluxograma) (MORAIS; AZEVEDO, 2017a).

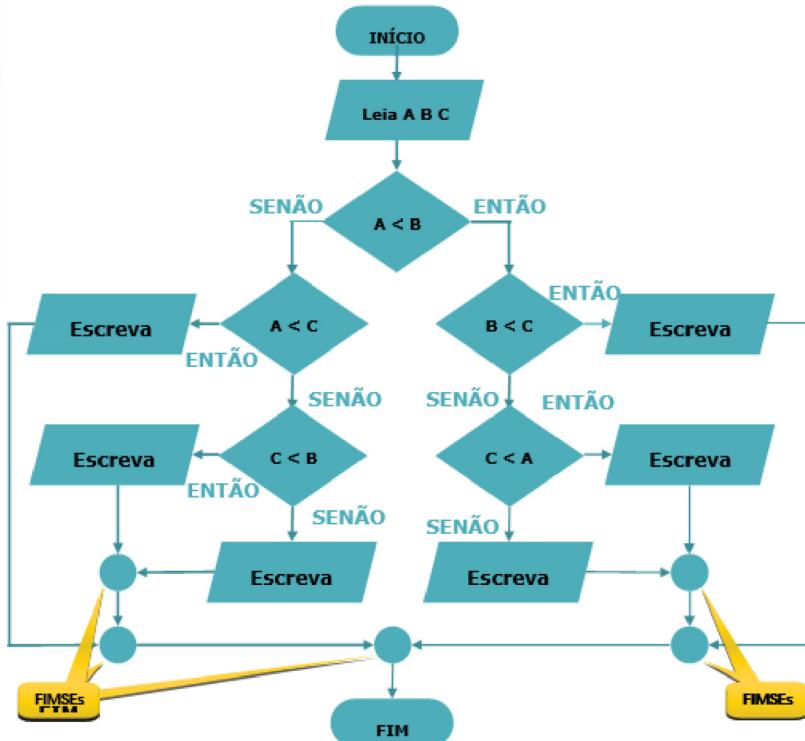


IMPORTANTE:

Perceba que a quantidade de conectores (círculos) que aparecem na Figura 1 é exatamente igual à quantidade de FIMSEs redigidos no algoritmo escrito previamente.

Desse modo, fica mais fácil percorrer as várias situações e decisões que serão tomadas pelo programa em tempo de execução.

Figura 1 – Versão do algoritmo-exemplo em diagrama de blocos



Fonte: Morais e Azevedo (2017a).

Finalmente, vamos testar o nosso algoritmo. Para isso, podemos agir de duas maneiras distintas: ou testamos a lógica desenvolvida visualmente, realizando o que chamamos de “teste de mesa” (também conhecido como “chinês”); ou digitamos o código do algoritmo no editor do VisuAlg e executamos o programa, fazendo, claro, as devidas adequações quanto à sintaxe das instruções e da estrutura condicional SE (MORAIS; AZEVEDO, 2017a).

Figura 2 – Algoritmo que irá exercer a função de colocar três números em ordem crescente

Área dos algoritmos (Edição de código fonte) ->

```

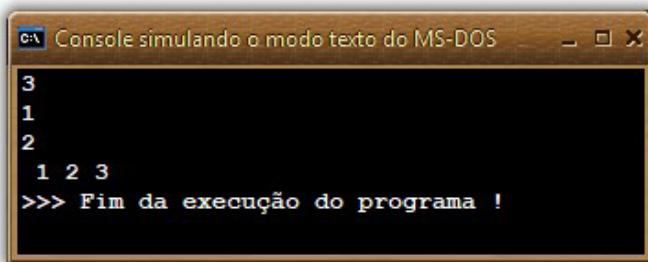
1 Algoritmo "Aula 08 - Exemplo 2"
2     VAR
3         A, B, C: INTEIRO
4         INICIO
5             LEIA (A, B, C)
6             SE A < B ENTÃO
7                 SE B < C ENTÃO
8                     ESCREVA (A, B, C)
9                     SENÃO
10                    SE C < A ENTÃO
11                        ESCREVA (C, A, B)
12                        SENÃO
13                            ESCREVA (A, C, B)
14                            FIMSE
15                        FIMSE
16                    SENÃO
17                        SE A < C ENTÃO
18                            ESCREVA (B, A, C)
19                            SENÃO
20                                SE C < B ENTÃO
21                                    ESCREVA (C, B, A)
22                                    SENÃO
23                                        ESCREVA (B, C, A)
24                                        FIMSE
25                                    FIMSE
26                                FIMSE
27 FimAlgoritmo

```

Fonte: Reprodução do software VisuAlg.

A execução do código no editor do VisuAlg irá resultar nas informações apresentada na Figura 3.

Figura 3 – Resultado de execução do algoritmo no console do computador



The screenshot shows a terminal window titled "Console simulando o modo texto do MS-DOS". Inside the window, the following text is displayed:
3
1
2
1 2 3
->> Fim da execução do programa !

Fonte: Reprodução do software VisuAlg.

Na lógica da programação, as estruturas condicionais SE encadeadas têm um papel importante. Assim, é importante sempre se aprofundar no conhecimento desse assunto.



SAIBA MAIS:

Para se aprofundar no tema deste tópico, é recomendado o acesso a alguns exercícios de algoritmização resolvidos, como os da página a seguir. Para acessar, [clique aqui](#).

É importante salientar que cada linguagem de programação tem as suas particularidades, sendo interessante para cada projeto e linguagem adotada serem aprimorados para evitar resultados inesperados.



RESUMINDO:

E então? Gostou do que lhe mostramos? Aprendeu mesmo tudinho? Agora, só para termos certeza de que você realmente entendeu o tema de estudo deste Capítulo, vamos resumir tudo o que vimos. Você deve ter aprendido que a estrutura condicional SE permite que um bloco de instruções seja executado apenas quando satisfeitas algumas condições. Quando não, outras instruções podem ser executadas. No entanto, essas estruturas condicionais suportam mais que meras instruções: você pode inserir outras estruturas condicionais dentro delas, sem limite de quantidade ou de níveis hierárquicos. Para tanto, é importante entender a sintaxe dessas estruturas complexas, assim como exercitar sua aplicação por meio de alguns exemplos.

Estrutura Condisional Encadeada



OBJETIVO:

Ao término deste Capítulo você será capaz de aplicar estruturas de repetição em soluções algorítmicas envolvendo contagem e acumulação. Isso será fundamental para o exercício de sua profissão. E então? Motivado para desenvolver esta competência? Então vamos lá. Avante!

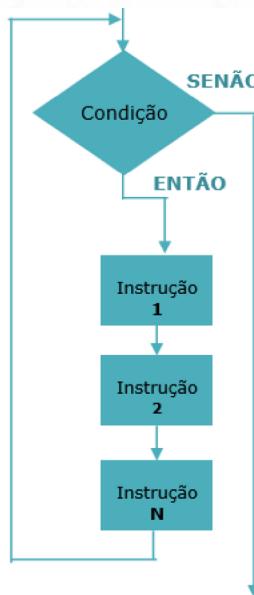
Repetição Interativa com Teste no Início do Laço

Um laço de repetição é uma estrutura que permite a execução de certo bloco de instruções conforme determinada condição. Esses laços podem ser os seguintes (MORAIS; AZEVEDO, 2017b):

- Interativos: quando necessitam da intervenção de um usuário do programa para repetir a próxima ação em um indeterminado número de vezes.
- Iterativos: quando executam as repetições previstas de forma automática.

Inicialmente, trataremos da estrutura de repetição com teste no início, mas o que isso quer dizer? Significa que, para o restante das instruções ser executado, uma condição precisa ser testada, e a repetição da execução dessa instrução dependerá dessa condição.

Figura 4 – Fluxograma da estrutura ENQUANTO-FAÇA



Fonte: Moraes e Azevedo (2017c).

A seguir, podemos visualizar a sintaxe de um laço repetitivo com pré-teste, ou seja, condição no início do laço.

```
ENQUANTO <expressão condicional> FAÇA
    <instrução-1>
    <instrução-2>
    ...
    <instrução-N>
FIMENQUANTO
```

Para exemplificar, imagine que precisamos desenvolver um algoritmo que leia uma sequência de números inteiros e, toda vez que o número digitado for maior que 10, este seja exibido no console do computador. Ao ser digitado o número 999, o programa deverá ser encerrado.

Veja como ficaria esta solução (MORAIS; AZEVEDO, 2017c):

```

ALGORITMO
VAR
N: INTEIRO
LEIA (N)
ENQUANTO N <> 999 FAÇA
SE N > 10 ENTÃO
ESCREVA (N)
FIMSE
LEIA (N)
FIMENQUANTO
FINALGORITMO

```

Perceba que estamos diante de uma estrutura repetitiva interativa, ou seja, a cada looping o usuário interagirá com o programa, no caso, digitando um número inteiro. Ao executarmos esse mesmo algoritmo no VisuAlg, teremos o seguinte código e monitoramento de resultados no console, conforme exemplificado na Figura 5.

Figura 5 – Algoritmo-exemplo interpretado pelo VisuAlg

Área dos algoritmos (Edição do código fonte) ->

```

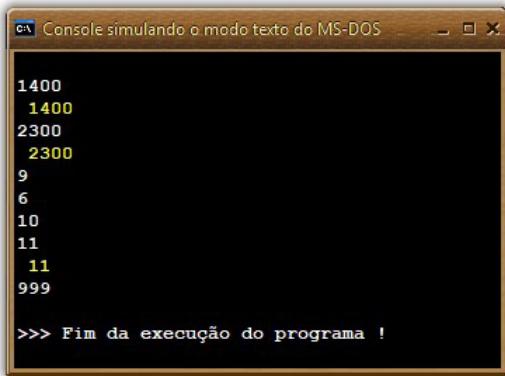
1 Algoritmo "Aula 10 - Exemplo 1"
2     VAR
3         N: INTEIRO
4         INICIO
5             LEIA (N)
6             ENQUANTO N <> 999 FAÇA
7                 SE N > 10 ENTÃO
8                     ESCREVA (N)
9                 FIMSE
10                LEIA (N)
11                FIMENQUANTO
12 FimAlgoritmo

```

Fonte: Reprodução do software VisuAlg.

Após inseridos os códigos com as diretrizes do VisuAlg, haverá o resultado constado nas informações, conforme apresentado na Figura 6.

Figura 6 – Representação do algoritmo-exemplo interpretado pelo VisuAlg



Fonte: Reprodução do software VisuAlg.

Aproveitando o exemplo anterior, imagine que, desta vez, desejamos exibir a quantidade de números digitados que foram maiores que 10, ou seja, que foram impressos no console do computador. Para isso, vamos recorrer a uma variável de memória auxiliar, que iremos chamar de C, em alusão ao termo “contador”.

ALGORITMO
VAR
N, C: INTEIRO
LEIA (N)
ENQUANTO N <> 999 FAÇA
SE N > 10 ENTÃO
ESCREVA (N)
C C + 1
FIMSE
LEIA (N)
FIMENQUANTO
ESCREVA (“Números maiores que 10 foram:”, C)
FIMALGORITMO

Agora, para podermos visualizar melhor, vamos testá-lo no VisuAlg.

Figura 7 – Implementando variável contadora

The screenshot shows the 'Área dos programas' (Program Area) of the VisuAlg software. The code is written in a pseudocode-like language:

```
1 Algoritmo "Aula 10 - Exemplo 1"
2     VAR
3         N, C: INTEIRO
4         INICIO
5             LEIA (N)
6             ENQUANTO N <> 999 FACÀ
7                 SE N > 10 ENTÃO
8                     ESCREVA (N)
9                     C <- C + 1
10                FIMSE
11                LEIA (N)
12            FIMENQUANTO
13            ESCREVA ("Números maiores que 10 foram:", C)
14 FimAlgoritmo
```

Fonte: Reprodução do software VisuAlg.

Note que apenas os números 11, 15 e 22 foram impressos e, consequentemente, contados pela variável C.

Figura 8 – Visualização da variável contadora

The screenshot shows a terminal window titled 'Console simulando o modo texto do MS-DOS'. The output of the program is displayed:

```
3
5
7
9
11
11
15
15
22
22
999
Números maiores que 10 foram: 3
>>> Fim da execução do programa !
```

Fonte: Reprodução do software VisuAlg.

Perceba que, para essa contagem ter dado certo, foi necessária a soma do número 1 ao valor parcial da variável C em cada looping. Ela iniciou a execução do programa com o valor zero. Ao ser executada a instrução da linha 9 pela primeira vez, ao zero foi somado o número 1. Na segunda rodada, ao 1 foi somado mais 1, ficando 2, e, finalmente, na terceira rodada, ao 2 foi somado 1, ficando 3.

Outro conceito importante é o de “acumulação”. Imagine que, além da contagem, desejamos totalizar os números maiores que 10 no algoritmo anterior. Vamos, então, “lançar mão” de uma nova técnica: a acumulação, diferentemente da contagem, na acumulação, não somamos 1 à variável, mas sim o próprio número lido. Vejamos um exemplo de solução neste sentido:

```
ALGORITMO
VAR
    N, C, S: INTEIRO
    LEIA (N)
    ENQUANTO N > 999 FAÇA
        SE N > 10 ENTÃO
            ESCREVA (N)
            C ← C + 1
            S ← S + N
        FIMSE
        LEIA (N)
    FIMENQUANTO
    ESCREVA ("Números maiores que 10 foram:", C)
    ESCREVA ("e totalizaram:", S)
FIMALGORITMO
```

Testando no VisuAlg, teremos o seguinte resultado, conforme apresentado na Figura 9.

Figura 9 – Acrescentando uma variável acumuladora

Área dos programas (Edição do código fonte) → Nome do arquivo: [AULA 10]

```

1 Algoritmo "Aula 10 - Exemplo 1"
2     VAR
3         N, C, S: INTEIRO
4         INICIO
5             LEIA (N)
6             ENQUANTO N <> 999 FAÇA
7                 SE N > 10 ENTÃO
8                     ESCREVA (N)
9                     C <- C + 1
10                    S <- S + N
11                FIMSE
12                LEIA (N)
13            FIMENQUANTO
14            ESCREVA ("Números maiores que 10 foram:", C)
15            ESCREVA (" , e totalizaram:", S)
16 FimAlgoritmo

```

Fonte: Reprodução do software VisuAlg.

Se você executar este programa passo a passo (linha a linha), e conseguir monitorar os valores assumidos pela variável S que implementamos no algoritmo acima, verá que ela acumulará os números 11, 15 e 22 a cada looping (MORAIS; AZEVEDO, 2017c).

Figura 10 – Visualização após acrescentada uma variável acumuladora

```

3
5
7
9
11
11
15
15
22
22
999
Números maiores que 10 foram: 3, e totalizaram: 48
>>> Fim da execução do programa !

```

Fonte: Reprodução do software VisuAlg.

Na lógica da programação, como em outras áreas da Tecnologia da Informação, é sempre importante executar os testes. Aqui, utilizei aplicativos com os quais vocês têm maior familiaridade, com o VisuAlg.

Repetição Iterativa

Outra estrutura repetitiva, que também pode ser interativa ou iterativa, é o REPITA ATÉ. Trata-se de uma estrutura de repetição munida de um pós-teste de saída do laço, em vez de um pré-teste, como era o caso do ENQUANTO FAÇA. Em termos práticos, o REPITA ATÉ só testa se o fluxo de instruções será percorrido após tê-lo executado, o que não ocorre com o ENQUANTO FAÇA, que testa a condição de permanência antes de executar o bloco de instruções.

```
REPITA
    <instrução-1>
    <instrução-2>
    ...
    <instrução-N>
ATÉ <expressão condicional>
```

Vamos reescrever o algoritmo do exemplo anterior, substituindo a estrutura ENQUANTO FAÇA por um REPITA ATÉ.

ALGORITMO
VAR
N, C, S: INTEIRO
REPITA
LEIA (N)
SE (N > 10) E (N <> 999) ENTÃO
ESCREVA (N)
C ← C + 1
S ← S + N
FIMSE
ATÉ N = 999
ESCREVA ("Números maiores que 10 foram:", C)
ESCREVA ("e totalizaram:", S)
FIMALGORITMO

Observe que essa estrutura não se encerra com um FIMREPITA ou algo do gênero, mas sim com a condição de escape do laço.

**IMPORTANTE:**

A condição do REPITA ATÉ é a negativa da condição do ENQUANTO FAÇA, ou seja, ao passo que o ENQUANTO FAÇA questiona sobre a permanência, o REPITA ATÉ questiona sobre a sua saída do laço.

Já a repetição iterativa é o contrário da interativa, ou seja, trata-se de uma estrutura repetitiva que não interage com o usuário. O exemplo que vimos anteriormente é tipicamente interativo, pois o laço pede algo a ser digitado a cada looping. No entanto, e se já soubermos previamente quantos loopings o laço percorrerá até chegar a seu termo? Por exemplo, imagine que precisamos elaborar um algoritmo que calcule o somatório de todos os números pares até um determinado número digitado pelo usuário. Usando os artifícios que já aprendemos, podemos elaborar o seguinte algoritmo (MORAIS; AZEVEDO, 2017c):

ALGORITMO
VAR
N, C, S: INTEIRO
LEIA (N)
C ← 2
REPITA
S ← S + C
C ← C + 2
ATÉ C > N
ESCREVA ("O somatório é ", S)
FIMALGORITMO

Usando os artifícios que já aprendemos, podemos elaborar o seguinte algoritmo, conforme apresentado na Figura 11.

Figura 11 – Algoritmo que calcula a soma de todos os números pares até N

The screenshot shows the VisuAlg software interface. On the left, there is a code editor window titled "Área dos programas (Edição do código fonte) -> Nome do arquivo: [AULA 10 - EXEMPLO 5.ALG]". The code is written in a pseudocode-like language:

```

1 ALgoritmo "Aula 10 - Exemplo 5"
2   VAR
3     N, C, S: INTEIRO
4   INICIO
5     LEIA (N)
6     C <- 2
7     REPITA
8       S <- S + C
9       C <- C + 2
10    ATÉ C > N
11    ESCREVA ("O somatório é ", S)
12 FimAlgoritmo

```

On the right, there is a terminal window titled "Console simulando o modo texto do MS-DOS" showing the execution results:

```

10
O somatório é 30
>>> Fim da execução do programa !

```

Fonte: Reprodução do software VisuAlg.

Para esse tipo de situação, existe uma estrutura repetitiva iterativa que se adéqua perfeitamente ao que se pede: trata-se do PARA FAÇA. A sintaxe do PARA FAÇA é a seguinte:

PARA <variável> DE <inicial> ATÉ <limite> [PASSO <incremento>] FAÇA
<instrução-1>
<instrução-2>
...
<instrução-N>
FIMPARA

Vamos reescrever o algoritmo do exemplo anterior, substituindo a estrutura REPITA ATÉ por uma PARA FAÇA.

ALGORITMO
VAR
N, C, S: INTEIRO
LEIA (N)
PARA C DE 2 ATÉ N PASSO 2 FAÇA
S <- S + C
FIMPARA
ESCREVA ("O somatório é ", S)
FIMALGORITMO

Veja como o algoritmo ficou menor com a estrutura PARA FAÇA. A variável C, quanto parametrizada na estrutura PARA FAÇA, sofre um incremento automático de 2 até N, de 2 em 2, o que dispensa a instrução de contagem que havíamos programado.

**IMPORTANTE:**

Observe que a cláusula PASSO da estrutura repetitiva PARA FAÇA está entre colchetes, isso significa que ela é opcional, se omitida da estrutura PARA FAÇA, o incremento sofrido pela variável contadora será de +1.

Testando no VisuAlg, teremos o seguinte código para essa estrutura, conforme apresentado na Figura 12.

Figura 12 – Estrutura repetitiva iterativa usando o PARA FAÇA N

Área dos programas (Edição do código fonte) -> Non

```

1 Algoritmo "Aula 10 - Exemplo 6"
2     VAR
3         N, C, S: INTEIRO
4         INICIO
5             LEIA (N)
6             PARA C DE 2 ATÉ N PASSO 2 FAÇA
7                 S <- S + C
8             FIMPARA
9             ESCREVA ("O somatório é ", S)
10        FimAlgoritmo

```

Fonte: Reprodução do software VisuAlg.

As estruturas repetitivas são importantes para a lógica de programação. Dessa maneira, é importante estar atualizado sobre o funcionamento desse tipo de estrutura na linguagem de programação adotada nos projetos.



SAIBA MAIS:

Para se aprofundar mais em relação a este tópico, acesse a documentação do VisuAlg. Para acessar, [clique aqui](#).

São vários tipos de repetição, por exemplo, a interativa com teste no início do laço e a repetição iterativa. É importante se atentar às diferenças entre interativa e iterativa.



RESUMINDO:

E então? Gostou do que lhe mostramos? Aprendeu mesmo tudinho? Agora, só para termos certeza de que você realmente entendeu o tema de estudo deste Capítulo, vamos resumir tudo o que vimos. Você deve ter aprendido que as estruturas repetitivas são uns dos pontos importantes da lógica de programação, principalmente ao vivenciar situações em que teremos de aplicar técnicas de iteração (ou de repetição) para solucionarmos problemas envolvendo acumulação e contagem numérica. Para isso, precisaremos que você tenha compreendido muito bem as estruturas condicionais encadeadas, pois seus conceitos são fundamentais para o avançar do conhecimento.

Estruturas Repetitivas com Cadeias de Caracteres



OBJETIVO:

Ao término deste Capítulo você será capaz de aplicar estruturas de repetição em soluções algorítmicas envolvendo cadeias de caracteres e variáveis alfanuméricas. Isso será fundamental para o exercício de sua profissão. E então? Motivado para desenvolver esta competência? Então vamos lá. Avante!

Relembrando Funções de Caracteres

Também conhecidas como “funções string”, as funções de caracteres têm por finalidade ler expressões alfanuméricas (constantes ou variáveis) e retornar valores ou cadeias de caracteres para serem utilizados em instruções ou, ainda, como argumentos de outras funções. Na prática, as funções string são importantes para a atividade de programação pelas seguintes razões (MORAIS; AZEVEDO, 2017b):

- Muitas vezes, necessitamos comparar o valor numérico de uma expressão string com o de outra, ou atribuímos tal valor a uma variável numérica.
- Também é comum precisarmos isolar parte de uma variável string.

São essas as funções string suportadas pelo VisuAlg, mas existem inúmeras outras, dependendo da linguagem de programação.

Quadro 1 – Principais funções string do VisuAlg

Função	Finalidade	Exemplo
Caracpnum()	Retorna o número inteiro ou real equivalente à cadeia de caracteres ou expressão alfanumérica informada como argumento da função, no caso, PALAVRA.	Var PALAVRA: caractere RESULTADO: inteiro PALAVRA ← "1000" RESULTADO ←CARACPNUM(PALAVRA) + 2000 ESCREVA RESULTADO
Numpcarac()	Retorna uma cadeia de caracteres contendo a representação alfanumérica de NUMERO. Essa função faz exatamente o contrário da Caracpnum(), ou seja, converte um NUMERO em uma string.	Var NUMERO: inteiro RESULTADO: caractere NUMERO ← 1000 RESULTADO ←NUMPCARAC(NUMERO) + " reais" ESCREVA RESULTADO
Compr()	Retorna um valor numérico inteiro correspondente ao tamanho da variável, constante ou expressão FRASE. Em outras palavras, essa função serve para contar quantos caracteres existem na expressão, constante ou variável passada como argumento da função.	Var FRASE: caractere RESULTADO: inteiro FRASE ← "Há mais brilhos em nossos olhos que estrelas no céu" RESULTADO ← COMPR(FRASE) ESCREVA RESULTADO
Copia()	Retorna uma cadeia de caracteres contendo uma cópia parcial de FRASE, a partir do caractere situado na posição INICIO, contendo TAMANHO caracteres a partir daquela posição. Os caracteres são numerados da esquerda para a direita, começando de 1.	Var FRASE: caractere RESULTADO: caractere FRASE ← "Há mais brilhos em nossos olhos que estrelas no céu" RESULTADO ← COPIA(FRASE, 9, 7) ESCREVA RESULTADO

Fonte: Morais e Azevedo (2017b).

Um exemplo prático são problemas envolvendo pesquisas. Você já se perguntou como o buscador da Google consegue analisar de forma tão eficiente o que escrevemos na barra de pesquisa? O algoritmo de busca do Google é considerado um dos marcos revolucionários do mundo digital contemporâneo, tendo chegado a dar origem a uma ciência conhecida hoje como "Marketing de Buscas", entre outras terminologias técnicas.

Figura 13 – Barra de pesquisa do Google



Fonte: Pixabay

Não iremos abordar o algoritmo do Google, mas vamos começar a entender como o sistema de buscas consegue ter acesso às partes dos conteúdos que digitamos na barra de pesquisa. E, para começar, vamos imaginar um algoritmo capaz de ler uma letra ou um símbolo qualquer, e encontrá-lo dentro de um texto digitado, informando ao usuário em que posição do texto ele se encontra. Caso o caractere não seja localizado, a mensagem "Caractere não encontrado no texto" deve ser exibida. Do contrário: "Encontrado o caractere na posição #", onde # é a posição numérica do caractere da esquerda para a direita.

ALGORITMO

```

VAR
TEXTO, SIMBOLO: CARACTERE
POSIC, IND: INTEIRO
ESCREVA "Digite um texto: "
LEIA TEXTO
ESCREVA " – Agora digite uma letra ou algarismo a ser pesquisado: "
LEIA SIMBOLO
PARA IND DE 1 ATÉ COMPR(TEXTO) FAÇA
    SE COPIA (TEXTO, IND, 1) = SIMBOLO ENTÃO
        POSIC ← IND
    FIMSE
FIMPARA
SE POSIC = 0 ENTÃO
    ESCREVA "Caractere não encontrado no texto"
SENÃO
    ESCREVA "Encontrado o caractere na posição ", POSIC
FIMSE
FIMALGORITMO

```

Vamos testar esse algoritmo no VisuAlg. Executando as devidas adaptações sintáticas, teremos o seguinte resultado, como apresentado na Figura 14.

Figura 14 – Algoritmo-exemplo de pesquisa textual

Área dos programas (Edição do código fonte) → Nome do arquivo: [AULA 11 - EXEMPLO 1.ALG] —

```

1 Algoritmo "Aula 11 - Exemplo 1"
2 VAR
3 TEXTO, SIMBOLO: CARACTERE
4 POSIC, IND: INTEIRO
5 INICIO
6 ESCREVA ("Digite um texto: ")
7 LEIA (TEXTO)
8 ESCREVA (" – Agora digite uma letra ou algarismo a ser pesquisado: ")
9 LEIA (SIMBOLO)
10 PARA IND DE 1 ATÉ COMPR(TEXTO) FAÇA
11     SE COPIA(TEXTO, IND, 1) = SIMBOLO ENTÃO
12         POSIC <- IND
13     FIMSE
14 FIMPARA
15     SE POSIC = 0 ENTÃO
16         ESCREVA ("Caractere não encontrado no texto")
17     SENÃO
18         ESCREVA ("Encontrado o caractere na posição ", POSIC)
19     FIMSE
20 FimAlgoritmo

```

Fonte: Reprodução do software VisuAlg.

Primeiramente, perceba que “lançamos mão” da estrutura repetitiva PARA FAÇA, percorrendo toda a cadeia de caracteres contida dentro da variável TEXTO digitada pelo usuário. A variável IND foi criada com o objetivo de representar a posição de cada caractere presente dentro da variável TEXTO, por isso ela teve seu valor variando de 1 até o tamanho total da variável TEXTO, expresso pela função COMPR(TEXTO). Note que, a cada looping do laço PARA FAÇA, foi utilizada uma estrutura condicional SE para perguntar se o caractere corrente (posição IND) era igual ao caractere pesquisado (presente na variável SIMBOLO). Caso afirmativo, esse caractere corrente será atribuído à variável POSIC, que irá guardar a posição do caractere encontrado. Caso o laço chegue até seu término sem ter localizado o caractere pesquisado, nada restará dentro da variável POSIC, exceto seu valor inicial, que é ZERO. Por isso, após a cláusula FIMPARA, o algoritmo está perguntando se POSIC é igual a zero. Se for, é porque o caractere não foi localizado. Do contrário, é só exibir o seu valor final. Vamos testar nosso algoritmo. Primeiramente, digite o texto: “EU TE AMO” e procure pela letra “A”. Depois, basta executar o programa no VisuAlg (MORAIS; AZEVEDO, 2017b).

Figura 15 – Testando com um caractere existente

```

C:\ Console simulando o modo texto do MS-DOS

Digite um texto: EU TE AMO
- Agora digite uma letra ou algarismo a ser pesquisado: A
Encontrado o caractere na posição 7
>>> Fim da execução do programa !
  
```

Fonte: Reprodução do software VisuAlg.

E se digitássemos um caractere inexistente no texto? Experimente a letra “W”, usando o mesmo texto digitado anteriormente. O que você vai ver no console do computador?

Figura 16 – Testando com um caractere inexistente no texto

```

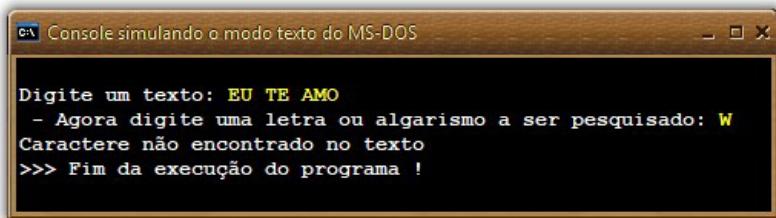
C:\ Console simulando o modo texto do MS-DOS

Digite um texto: EU TE AMO
- Agora digite uma letra ou algarismo a ser pesquisado: W
Caractere não encontrado no texto
>>> Fim da execução do programa !
  
```

Fonte: Reprodução do software VisuAlg.

O que aconteceria se tentássemos localizar a letra "E" na frase "EU TE AMO", utilizando esse algoritmo que acabamos de elaborar? Note que essa letra aparece duas vezes na frase digitada.

Figura 17 – Testando com um caractere em duplicitade



The screenshot shows a terminal window titled "Console simulando o modo texto do MS-DOS". The text inside the window is:

```
Digite um texto: EU TE AMO
- Agora digite uma letra ou algarismo a ser pesquisado: W
Caractere não encontrado no texto
>>> Fim da execução do programa !
```

Fonte: Reprodução do software VisuAlg.

O resultado seria que a última posição da letra "E" seria exibida. Deve-se atentar para não confundir como o algoritmo irá escrever apenas a posição da primeira letra "E" que aparece no texto, a letra "E" não será localizada no texto ou o algoritmo vai provocar um erro de sintaxe.



SAIBA MAIS:

No VisuAlg, existem algumas funções de caracteres que tratam da representação ASCII dos caracteres de uma expressão string, são elas: ASC() e CARAC(). Para saber mais, acesse as fontes de consulta a seguir.

Tabela ASCII. Para acessar, [clique aqui.](#)

É importante que você enriqueça cada vez mais o seu algoritmo, testando todas as situações possíveis e imagináveis. Depois, implemente outras informações, como um contador de vezes que o caractere pesquisado aparece, com base no exemplo visto anteriormente.

**RESUMINDO:**

E então? Gostou do que lhe mostramos? Aprendeu mesmo tudinho? Agora, só para termos certeza de que você realmente entendeu o tema de estudo deste Capítulo, vamos resumir tudo o que vimos. Você deve ter aprendido que ao avançarmos nos estudos dos algoritmos, para facilitar o aprendizado, é importante a carga teórica. Ao longo deste tópico, foram retomados conhecimentos acerca das estruturas repetitivas, aplicando-os a novas situações, por exemplo, a manipulação de cadeias de caracteres dentro de variáveis alfanuméricas.

Vetores



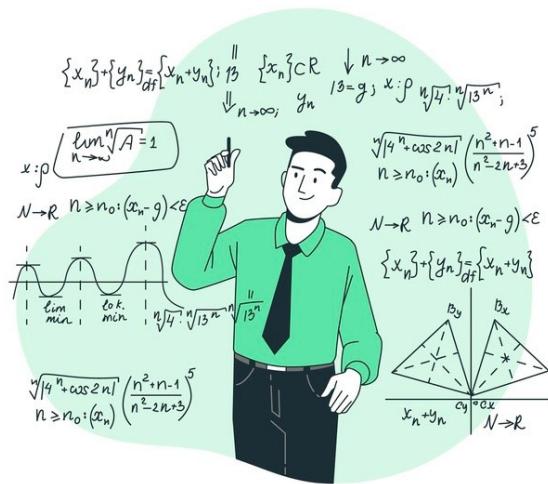
OBJETIVO:

Ao término deste Capítulo você será capaz de entender o conceito e as aplicações de vetores em soluções algorítmicas. Isso será fundamental para o exercício de sua profissão. E então? Motivado para desenvolver esta competência? Então vamos lá. Avante!

Conceito de Vetor

Da matemática que estudamos em nosso ensino médio, sabemos que um vetor é uma linha ou uma coluna de uma matriz, que, por sua vez, representa uma tabela que armazena números ou quaisquer outras informações organizadas em linhas e colunas (LOPES, 2017).

Figura 18 – Vetor é uma linha ou coluna de uma matriz



Fonte: Freepik

Na Ciéncia da Computação, o conceito de vetor não difere muito, no entanto, está associado à definição de variáveis de memória (MORAIS; AZEVEDO, 2017d).



DEFINIÇÃO:

Vetores são variáveis de memória homogêneas capazes de armazenar mais de um dado simultaneamente, indexando-os quanto ao seu posicionamento relativo em sua estrutura.

Assim, por exemplo, se quisermos armazenar as notas que um determinado aluno obteve ao longo de um ano letivo, teremos o seguinte, como exemplificado na Tabela 2.

Tabela 2 – Exemplo de vetor contendo notas obtidas por um aluno

10,0	9,5	6,5	7,0	8,5	4,0	10,0	6,5
1	2	3	4	5	6	7	8

Fonte: Morais e Azevedo (2017d).

Mas, qual é a utilidade prática de um vetor como esse? Para entender melhor, imagine que precisamos elaborar um algoritmo que leia todas as oito notas que um aluno obteve ao longo de seu ano letivo e, depois, calcule a média aritmética que esse aluno conseguiu ao final do último período (MORAIS; AZEVEDO, 2017d).

Figura 19 – Aplicação prática do vetor de um algoritmo que leia as notas que um aluno obteve ao longo de seu ano letivo



Fonte: Freepik

Com os conhecimentos acumulados até o momento, teríamos que criar oito variáveis de memória distintas, uma para cada nota. Teríamos, então, um algoritmo bastante extenso e engessado, pois para adaptá-lo a qualquer variação em termos de quantidade de notas, teríamos que criar ou subtrair mais e mais variáveis. Veja a seguir o algoritmo que lê e calcula a média de oito notas escolares (MORAIS; AZEVEDO, 2017d).

ALGORITMO

VAR

IND: INTEIRO

NOTA1, NOTA2, NOTA3, NOTA4, NOTA5, NOTA6, NOTA7, NOTA8, MED: REAL

ESCREVA "Digite 8 notas: "

LEIA NOTA1, NOTA2, NOTA3, NOTA4, NOTA5, NOTA6, NOTA7, NOTA8

MED <- (NOTA1+NOTA2+NOTA3+NOTA4+NOTA5+NOTA6+NOTA7+NOTA8)/8

ESCREVA "A média é: ", MED

FIMALGORITMO

Utilizando o VisuAlg, será como apresentado na Figura 20.

Figura 20 – Algoritmo-exemplo usando variáveis simples

- Área dos programas (Edição do código fonte) -> Nome do arquivo: [AULA 12 - EXEMPLO 1..]

```

1 Algoritmo "Aula 12 - Exemplo 1"
2 VAR
3 IND: INTEIRO
4 NOTA1, NOTA2, NOTA3, NOTA4, NOTA5, NOTA6, NOTA7, NOTA8, MED: REAL
5 INICIO
6 ESCREVA ("Digite 8 notas: ")
7 LEIA (NOTA1, NOTA2, NOTA3, NOTA4, NOTA5, NOTA6, NOTA7, NOTA8)
8 MED <- (NOTA1+NOTA2+NOTA3+NOTA4+NOTA5+NOTA6+NOTA7+NOTA8)/8
9 ESCREVA ("A média é: ", MED)
10 FimAlgoritmo

```

Fonte: Reprodução do software VisuAlg.

Mas, se usarmos o conceito de vetor, poderemos economizar a declaração de oito variáveis, trocando-as por apenas uma: a variável tipo vetor NOTAS[]. Veja agora a sintaxe da declaração de um vetor para declarar uma variável do tipo vetor em um algoritmo (compatível com a linguagem VisuAlg) (MORAIS; AZEVEDO, 2017d):

```
<nome da variável>: VETOR [1..#] DE <tipo da variável>
```

Exemplos:

```
VAR
```

```
NOTAS: VETOR [1..8] DE REAL
```

Uma vez declarada, uma variável do tipo vetor pode ser utilizada livremente, da mesma maneira que uma variável simples, tendo, porém, o cuidado de sempre se referir a um de seus elementos, e não a ela como um todo. Assim, para atribuir uma nota 9,0 ao quinto elemento do vetor NOTAS, teremos (MORAIS; AZEVEDO, 2017d):

```
NOTAS[5] ← 9.0
```

Ao número 5 ("cinco") que usamos entre colchetes damos o nome de "índice do vetor", que referencia o seu quinto elemento. Cada elemento de um vetor se comporta como uma variável simples, ou seja, pode receber valores atribuídos e participar como operandos em todo e qualquer tipo de expressão, respeitando sempre o seu tipo.



IMPORTANTE:

Sendo uma variável homogênea, o vetor não pode ter elementos de diferentes tipos, em outras palavras, uma vez declarado como numérico inteiro, todos os seus elementos só poderão receber valores numéricos inteiros. No entanto, em algumas linguagens de programação, como o Visual Basic, os vetores são tratados como variáveis heterogêneas, podendo receber dados de diferentes tipos para seus elementos.

É importante se atentar aos problemas envolvendo vetores. Você lembra do último algoritmo que construímos? Aquele que lê oito notas, calcula e exibe a média? Que tal reescrevê-lo envolvendo um vetor? A seguir será apresentado o mesmo algoritmo usando vetor no lugar de variáveis simples.

```

ALGORITMO
VAR
IND: INTEIRO
MED: REAL
NOTAS: VETOR [1..8] DE REAL
PARA IND DE 1 ATÉ 8 FAÇA
    LEIA NOTAS[IND]
    MED ← MED + NOTAS[IND]
FIMPARA
    ESCREVA "A média é: ", MED/8
FINALGORITMO

```

Vamos testar esse programa usando o VisuAlg, como é apresentado na Figura 21.

Figura 21 – Testando no VisuAlg

- Área dos programas (Edição do código fonte) -> Novo

```

1 Algoritmo "Aula 12 - Exemplo 2"
2 VAR
3 IND: INTEIRO
4 MED: REAL
5 NOTAS: VETOR [1..8] DE REAL
6 INICIO
7 PARA IND DE 1 ATÉ 8 FAÇA
8     LEIA (NOTAS [IND])
9     MED <- MED + NOTAS [IND]
10 FIMPARA
11 ESCREVA ("A média é: ", MED/8)
12 FimAlgoritmo

```

Fonte: Reprodução do software VisuAlg.

Recorremos aqui à estrutura de repetição FAÇA PARA, em que, a cada looping do laço, o algoritmo recebeu uma nota do aluno e, ao mesmo tempo, acumulou-o na variável MED. Após o término do laço, foi só exibir o valor acumulado das notas dividido por oito notas, o que resulta na média aritmética das oito notas digitadas (MORAIS; AZEVEDO, 2017d).

Figura 22 – Resultado do teste do algoritmo

The screenshot shows a terminal window titled "Console simulando o modo texto do MS-DOS". The window contains the following text:
10
9,5
6,5
7,0
8,5
4,0
10
6,5
A média é: 7.75
>>> Fim da execução do programa !

Fonte: Reprodução do software VisuAlg.

Note que a principal vantagem desse método está na flexibilidade do algoritmo, que, com pequenas alterações, funcionará para qualquer quantidade de notas a serem digitadas. E então! Gostou de trabalhar com vetores? A partir agora é só praticar.



IMPORTANTE:

Enriqueça cada vez mais o seu algoritmo, testando todas as situações possíveis e imagináveis.

Existe uma infinidade de situações nas quais os vetores são perfeitamente aplicáveis. Que tal incrementar este algoritmo que acabamos de construir, deixando a quantidade variável de notas?



SAIBA MAIS:

Acesse este material de aprofundamento sobre vetores e matrizes. Para acessar, [clique aqui.](#)

Para incrementar o algoritmo, deixando a quantidade variável de notas, basta incluir uma linha de instrução solicitando este quantitativo do usuário, logo no início do algoritmo.



RESUMINDO:

E então? Gostou do que lhe mostramos? Aprendeu mesmo tudinho? Agora, só para termos certeza de que você realmente entendeu o tema de estudo deste Capítulo, vamos resumir tudo o que vimos. Você deve ter aprendido que para chegarmos ao ponto bastante relevante dos estudos algorítmicos, é importante saber que, ao longo da evolução das ciências da computação, sentiu-se necessidade de expandir o conceito de variáveis de memória, pois estas somente conseguiam armazenar um dado por vez. Por exemplo, se quiséssemos guardar números telefônicos, precisaríamos de uma variável diferente para cada um dos números. Mas, desse modo, como desenvolver uma lógica para ler e imprimir todos os números telefônicos que dispomos? Com os conhecimentos que acumulamos até aqui, a única ideia factível é armazenar todos eles em uma variável do tipo caractere e, depois, utilizar um algoritmo de pesquisa para destrinchar os números telefônicos isoladamente. Para ficar mais fácil, basta o entendimento do conceito e as aplicações dos vetores.

REFERÊNCIAS

BROOKSHEAR, G. J. **Ciência da Computação:** Uma Visão Abrangente. Porto Alegre: Bookman, 2013.

FORBELLONE, A. L. **Lógica de programação:** a construção de algoritmos e estrutura de dados. São Paulo: Pearson, 2005.

MORAIS, I.S.; AZEVEDO M. **Lógica de programação:** estruturas condicionais SE encadeadas. Recife: Unissau, 2017a.

MORAIS, I.S.; AZEVEDO M. **Lógica de programação:** estruturas repetitivas com cadeias de caracteres. Recife: Unissau, 2017b.

MORAIS, I.S.; AZEVEDO M. **Lógica de programação:** estruturas repetitivas. Recife: Unissau, 2017c.

MORAIS, I.S.; AZEVEDO M. **Lógica de programação:** vetores. Recife: Unissau, 2017d.

SEBESTA, R. W. **Conceitos de linguagens de programação.** São Paulo: Bookman, 2011.

