Jaime Evaristo

Sérgio Crespo

Aprendendo a Programar Programando numa Linguagem Algorítmica Executável (ILA)

Segunda Edição
Capítulo 6
Versão 11052010

6. Variáveis compostas

6.1 Introdução

Nos exemplos 6 e 7 da seção 4.4 discutimos programas para a determinação da média de uma relação de números dados. Para tal, utilizamos uma variável simples para receber os números, sendo que cada vez que um número, a partir do segundo, era recebido o anterior era "perdido". Ou seja, a relação de números não era armazenada. Imagine que a relação fosse uma relação de notas escolares e além da média se quisesse também saber a quantidade de alunos que obtiveram nota acima da média ou uma outra medida estatística (*desvio padrão*, por exemplo) que dependesse da média. Neste caso, haveria a necessidade de que a relação fosse redigitada, o que, além da duplicidade do trabalho, facilitaria os erros de digitação. É importante então que exista uma "variável" capaz de armazenar vários valores simultaneamente de tal forma que se possa acessar cada um deles independentemente de se acessar os demais.

Um outro exemplo que justifica plenamente a necessidade de uma *variável composta* é o caso do exemplo 2 da seção 4.4. Lá queríamos a relação dos divisores de um inteiro dado e estes divisores eram apenas exibidos, não sendo armazenados, como recomendado na seção 2.7.

Uma *variável composta* é um conjunto de variáveis simples do tipo *Numerico*, identificadas pela concatenação de índices entre colchetes ao identificador da variável composta. Como as variáveis simples, as compostas também devem ser definidas no início do programa com a seguinte sintaxe:

Matriz numerico Identificador[Expr 1, Expr 2, ..., Expr n]

onde Expr 1, Expr 2, ..., Expr n são expressões numéricas cujo valor é um número inteiro. Comumente o valor de *n* é referido como sendo a *dimensão* da variável composta e cada variável simples da variável composta é chamada de *componente* da variável composta. Em relação aos valores de *n*, costuma-se denominar a variável composta das seguintes formas: quando $n = 1$, temos um *vetor*; quando $n = 2$, uma *matriz*; quando $n = 3$, uma *matriz tridimensional* e assim sucessivamente.

Para exemplificar um vetor, a declaração

Matriz numerico Vet[1000]

define um conjunto de mil variáveis do tipo numerico Vet[1], Vet[2], ..., Vet[1000] e podemos armazenar em Vet uma relação com até mil números. Já para exemplificar uma matriz, definição

Matriz numerico Mat[20, 30]

define um conjunto de 600 variáveis Mat[1, 1], ..., Mat[1, 30], Mat[2, 1], ..., Mat[2, 30], ..., Mat[20, 1], ..., Mat[20, 30]. Naturalmente, matrizes bidimensionais, como Mat acima, são utilizadas para armazenar *tabelas de dupla entrada*. Para um valor fixo de *i*, dizemos que Mat[i, 1], Mat[i, 2], ... constituem a *linha i* da matriz enquanto que para um valor fixo de *j* Mat[1, j], Mat[2, j], ... constituem uma *coluna j*. O número de linhas e o número de colunas constituem a *ordem* da matriz.

Uma limitação do sistema ILA para a manipulação de variáveis compostas é que as componentes destas variáveis não podem ser argumentos de um comando *Ler*. A solução é definir uma variável simples do tipo *Numerico* para ser passada para o comando *Ler* e em seguida usar um comando de atribuição para armazenar o valor digitado na componente pretendida. Se queremos armazenar um dado de entrada na primeira componente do vetor Vet, utilizamos uma variável Aux e a sequência de comandos

```
Ler Aux
Vet[1] = Aux
```

Feita esta observação, o armazenamento de uma relação de números num *vetor* depende do

fato de que seja ou não conhecida a quantidade de números da relação. Se esta quantidade é conhecida antecipadamente pode-se usar uma estrutura *Para proximo*:

```
Escrever "Digite os elementos da relação"
Para i = 1 ate n
        Ler Aux
        Vet[i] = Aux
Proximo
```

onde *n* é a quantidade de elementos da relação.

Se tal número não é conhecido, pode-se utilizar um *flag*, como foi discutido no exemplo 7 da seção 4.4, e uma estrutura *Faca enquanto*:

```
Escrever "Digite os números (-1 para encerrar)"
Ler Aux
Vet[1] = Aux
i = 1
Faca enquanto Vet[i] <> -1
        i = i + 1
        Ler Aux
        Vet[i] = Aux
Fim_enquanto
i = i - 1
```

Observe que, a partir daí, a variável *i* é importante para o resto do programa pois armazena a quantidade elementos da relação ou o *tamanho do vetor*. Obviamente, o comando i = i - 1 é para "retirar" do vetor o *flag*.

Como uma componente de um vetor pode ser argumento de um comando *Escrever*, a exibição dos elementos de uma relação de números armazenada num vetor é feita de maneira óbvia, bastando "percorrer" o vetor com uma estrutura *Para proximo*.

```
Para i = 1 ate Quant
        Escrever V[i]
Proximo
```

Para o armazenamento de uma matriz é comum se exigir o conhecimento prévio da sua ordem. Com isto pode-se utilizar duas estruturas *Para proximo* aninhadas, a externa para controlar as linhas e a interna para controlar as colunas.

```
Escrever "Digite a ordem da matriz"
Ler m
Ler n
Escrever "Digite (por linha) os elementos da matriz"
Para i = 1 ate m
        Para j = 1 ate n
                Ler Aux
                Mat[i, j] = Aux
        Proximo
Proximo
```

A exibição dos elementos de uma matriz também é feita com duas estruturas *Para proximo* aninhadas.

```
Para i = 1 ate m
        Para i = 1 ate n
                Escrever Mat[i, j]
        Proximo
```

```
Proximo
```

6.2 Exemplos Parte V

1. Um vetor pode ser gerado pelo próprio sistema. Por exemplo, o programa abaixo armazena num vetor os quadrados dos n primeiros números naturais, n dado.

```
Variaveis
        Numerico i, n
        Matriz Numerico Vet[100]
Inicio
        Escrever "Digite n"
        Ler n
        Para i = 1 ate n
                Vet[i] = i^2
        Proximo
        Escrever "Os quadrados dos ", n, "primeiros números naturais são "
        Para i = 1 ate n
                Escrever Vet[i]
        Proximo
Fim
```

2. Seja um programa para determinar a média das notas de uma avaliação de uma turma de uma escola e, em seguida, determinar quantos alunos obtiveram nota maior que a média. Podemos armazenar as notas num vetor, calcular a média destas notas e depois "percorrer" o vetor quantificando o número de componentes maiores que a média. Como foi dito na seção anterior, pode-se percorrer um vetor utilizando-se uma estrutura *Para proximo*.

```
//Programa para determinar a média de uma relação de números e a quantidade de números da relação maiores que a média.
Variaveis
        Numerico Aux, i, j, Media, Soma, Quant
        Matriz numerico Vet[50]
Inicio
        //Entrada dos dados e cálculo da média
        Escrever "Digite os elementos da relação (-1 para encerrar)"
        Ler Aux
        Vet[1] = Aux
        i = 1
        Soma = 0
        Faca enquanto Vet[i] <> -1
                Soma = Soma + Aux
                Ler Aux
                i = i + 1
                Vet[i] = Aux
        Fim_enquanto
        i = i - 1
        Media = Soma/i
        //Determinação do número de componentes maiores que a média
        Para j = 1 ate i
                Se Vet[j] > Media entao
                        Quant = Quant + 1
                Fim_se
```

```
        Proximo
        Escrever "Média da relação: ", Media, "; Maiores que a média: ", Quant
Fim
```

3. Para um programa que determine o maior elemento de uma relação armazenada num vetor pode-se usar o seguinte algoritmo. Supõe-se que o maior elemento é a primeira componente do vetor e, em seguida, percorre-se todo o vetor verificando se alguma componente é maior do que aquele valor que, até o momento, é o maior. Encontrando-se uma componente de valor maior, troca-se o maior valor.

```
//Programa para determinar a maior componente de um vetor
Variaveis
        Numerico Aux, i, j, Maior
        Matriz numerico Vet[50]
Inicio
        Escrever "Digite os elementos da relação (-1 para encerrar)"
        Ler Aux
        Vet[1] = Aux
        i = 1
        Faca enquanto Vet[i] <> -1
                Ler Aux
                i = i + 1
                Vet[i] = Aux
        Fim_enquanto
        i = i - 1
        Maior = Vet[1]
        Para j = 2 ate i
                Se Vet[j] > Maior entao
                        Maior = Vet[j]
                Fim_se
        Proximo
        Escrever "O maior elemento da relação é ", Maior
Fim
```

4. O programa a seguir exemplifica a possibilidade de que os índices das componentes de um vetor sejam dados através de expressões. O seu objetivo é obter uma relação de elementos "intercalando" os elementos de duas relações dadas. Ou seja, dadas duas relações de números, o programa deve gerar uma terceira relação onde os elementos de ordem ímpar são os elementos da primeira relação e os de ordem par os da segunda. Por exemplo, se as relações são (1, 3, 6, 12) e (2, 5, 10, 15) o programa deve gerar a relação (1, 2, 3, 5, 6, 10, 12, 15). Assim, se as relações dadas forem armazenadas nos vetores V e W e queremos gerar a terceira relação em Vet, deveremos ter Vet[1] = V[1], Vet[2] = W[1], Vet[3] = V[2], Vet[4] = W[2], e assim sucessivamente. Observe que se i é ímpar Vet[i] é a componente de V de ordem $\frac{i+1}{2}$ e se i é par Vet[i] é a componente de W de ordem $\frac{i}{2}$.

```
Variaveis
        Numerico Aux, i, n, m
        Matriz numerico V[250]
        Matriz numerico W[250]
        Matriz numerico Vet[500]
Inicio
        Escrever "Digite o número de elementos das relações"
```

```
        Ler n
        Escrever "Digite os elementos da primeira relação"
        Para i = 1 ate n
                Ler Aux
                V[i] = Aux
        Proximo
        Escrever "Digite os elementos da segunda  relação"
        Para i = 1 ate n
                Ler Aux
                W[i] = Aux
        Proximo
        m = 2*n
        Para i = 1 ate m
                Se Resto(i, 2) = 1 entao
                        Vet[i] = V[(i+2)/2]
                Senao
                        Vet[i] = W[i/2]
                Fim_se
        Proximo
        Para i = 1 ate m
                Escrever Vet[i]
        Proximo
Fim
```

Este exemplo mostra também que a declaração de variáveis compostas exige que cada uma delas seja definida isoladamente. O ILA não aceitaria a definição

```
Variaveis
        Matriz Numerico V[250], W[250], Vet[500]
```

5. Como no exemplo 1, o sistema pode gerar uma matriz. Para exemplificar isto, apresentaremos um programa que gera a *matriz identidade de ordem n*. Para um inteiro positivo dado, a *matriz identidade de ordem n* é a matriz $I_n = (i_{rs})$, de ordem nxn, dada por $i_{rs} = 1$, se r = s, e $i_{rs} = 0$, se $r \neq s$. Esta matriz é muito importante no estudo das matrizes sendo utilizada, por exemplo, para a determinação da *matriz inversa* de uma matriz inversível. Por exemplo, se *n* = 3, temos

$$I_3 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

```
//Programa para gerar a matriz unidade de ordem n
Variaveis
        Numerico k, j, n
        Matriz numerico I[30, 40]
Inicio
        Escrever "Digite a ordem da matriz unidade"
        Ler n
        Para k = 1 ate n
                Para j = 1 ate n
                        Se k = j entao
                                i[k,j] = 1
                        Senao
                                i[k, j] = 0
                        Fim_se
```

```
            Proximo
        Proximo
        Escrever "Matriz unidade de ordem ", n
        Para k = 1 ate n
            Para j = 1 ate n
                Escrever i[k, j]
            Proximo
        Proximo
Fim
```

6. Quando, como no exemplo anterior, o número de linhas é igual ao número de colunas a matriz é dita *matriz quadrada*. Neste caso, os elementos de índices iguais constituem a *diagonal principal*. A soma dos elementos da diagonal principal de uma matriz quadrada é o *traço* da matriz. Como mais um exemplo de programas que manipulem matrizes, o programa abaixo determina o *traço* de uma matriz quadrada dada. Observe que para percorrer a diagonal principal não há necessidade de dupla estrutura *Para proximo*.

```
//Programa para determinar o traço de uma matriz quadrada
Variaveis
    Numerico Aux, i, j, n, Traco
    Matriz numerico A[30, 30]
Inicio
    Escrever "Digite a ordem da  matriz"
    Ler n
    Escrever "Digite os elementos da matriz"
    Para i = 1 ate n
        Para j = 1 ate n
            Ler Aux
            A[i,j] = Aux
        Proximo
    Proximo
    //Determinação do traço da matriz
    Traco = 0
    Para i = 1 ate n
        Traco = Traco + A[i, i]
    Proximo
    Escrever "O traço da matriz dada e ", Traco
Fim
```

7. Naturalmente, o armazenamento de uma matriz que possui alguma propriedade específica pode ser facilitado se esta propriedade for levada em conta no programa. É o caso, por exemplo, de uma *matriz simétrica* (uma matriz quadrada $A = (a_{ij})$ é dita *simétrica* se $a_{ij} = a_{ji}$, quaisquer que sejam i e j). Se Mat[10, 10] é uma variável do tipo *Matriz numerico*, para armazenar em Mat uma matriz simétrica basta digitar os elementos situados acima da diagonal principal e os elementos desta diagonal.

```
//Programa para armazenar e exibir uma matriz simétrica
Variaveis
    Numerico i, j,  n, Aux, x, y
    Matriz numerico Mat[10, 10]
Inicio
    Escrever "Digite a ordem da matriz"
    Ler n
    Escrever "Digite os elementos da matriz, acima da diagonal"
```

```
        Para i = 1 ate n
            Para j = i ate n
                Ler Aux
                Mat[i, j] = Aux
                Mat[j, i] = Aux
            Proximo
        Proximo
Fim
```

8. Como se determina a soma de duas matrizes somando-se os seus elementos de mesmos índices, um programa que receba duas matrizes e determine a soma delas é muito simples.

```
Variaveis
        Numerico Aux, i, j, m1, n1, m2, n2
        Matriz numerico A[30, 40]
        Matriz numerico B[30, 40]
        Matriz numerico Soma[30, 40]
Inicio
        Escrever "Digite a ordem da primeira matriz"
        Ler m1
        Ler n1
        Escrever "Digite os elementos da primeira matriz"
        Para i = 1 ate m1
            Para j = 1 ate n1
                Ler Aux
                A[i,j] = Aux
            Proximo
        Proximo
        Escrever "Digite a ordem da segunda matriz"
        Ler m2
        Ler n2
        Escrever "Digite os elementos da segunda matriz"
        Para i = 1 ate m2
            Para j = 1 ate n2
                Ler Aux
                B[i, j] = Aux
            Proximo
        Proximo
        //Determinação da matriz soma
        Se (m1 = m2) e (n1 = n2) entao
            Para i = 1 ate m1
                Para j = 1 ate n1
                    Soma[i, j] = A[i, j] + B[i, j]
                Proximo
            Proximo
            Para i = 1 ate m1
                Para j = 1 ate n1
                    Escrever Soma[i, j]
                Proximo
            Proximo
        Senao
            Escrever "A soma das matrizes não esta definida"
```

```
        Fim_se
Fim
```

9. Já um programa para multiplicação de matrizes não é tão simples. Na verdade é um programa que é muito útil para o desenvolvimento da lógica de programação. Se $A = (a_{ij})_{mxn}$ e $B = (b_{ij})_{rxs}$, a matriz produto só está definida se n = r e, neste caso, se P = A . B, então $p_{ij} = a_{i1} . b_{1j} + a_{i2} . b_{2j} + ... + a_{in} . b_{rj}$.

Ou seja, o elemento ij da matriz produto é a soma dos produtos dos elementos da linha i da matriz A pelos elementos da coluna j da matriz B, o que exigirá uma terceira estrutura *Para proximo*.

```
Variaveis
        Numerico Aux, i, j, k, m1, n1, m2, n2
        Matriz numerico A[30, 40]
        Matriz numerico B[30, 40]
        Matriz numerico M[30, 40]
Inicio
        Escrever "Digite a ordem da primeira matriz"
        Ler m1
        Ler n1
        Escrever "Digite os elementos da primeira matriz"
        Para i = 1 ate m1
                Para j = 1 ate n1
                        Ler Aux
                        A[i,j] = Aux
                Proximo
        Proximo
        Escrever "Digite a ordem da segunda matriz"
        Ler m2
        Ler n2
        Escrever "Digite os elementos da segunda matriz"
        Para i = 1 ate m2
                Para j = 1 ate n2
                        Ler Aux
                        B[i,j] = Aux
                Proximo
        Proximo
        //Determinação da matriz produto
        Se n1 = m2 entao
                Para i = 1 ate m1
                        Para j = 1 ate n2
                                M[i, j] = 0
                                Para k = 1 ate n1
                                        M[i, j] = M[i, j] + A[i, k]*B[k, j]
                                Proximo
                        Proximo
                Proximo
                Para i = 1 ate m1
                        Para j = 1 ate n2
                                Escrever M[i, j]
                        Proximo
                Proximo
```

```
        Senao
                Escrever "O produto das matrizes não esta definido"
        Fim_se
Fim
```

6.3 Formatação da saída/Interface com o usuário

Mesmo não fazendo da parte da lógica de programação, é interessante que quem está aprendendo a programar saiba que os sistemas para desenvolvimento de programas oferecem recursos para que a saída do programa seja exibida de forma elegante e agradável de se ver e para que o sistema ofereça ao usuário *interfaces* que facilitem a entrada de dados. Isto é obtido através de funções (ou, em alguns sistemas, *procedimentos*) pré-definidos, sendo que, quanto mais sofisticado seja o sistema mais recursos ele oferece. Os sistemas *visuais*, como Visual Basic e Delphi, são os melhores exemplos de sistemas que oferecem recursos espetaculares para a criação de *interfaces* elegantes entre o sistema e o usuário.

Quando um comando *Escrever* é executado no ILA, o seu argumento é exibido e o *cursor* (traço intermitente que indica a posição onde o próximo caractere a ser exibido, sê-lo-á) vai para a linha e para a coluna seguintes àquelas da posição anterior. Por exemplo, se o vetor (3, 5, 8, 9, 12) estiver armazenado na variável Vet, a estrutura

```
Para i = 1 ate 5
        Escrever Vet[i]
Proximo
```

exibirá na tela

```
3
  5
    8
      9
        12
```

Naturalmente, um vetor não deve ser exibido desta forma! O ILA possui o comando *Posicionar* que, como o seu nome indica, posiciona o cursor num ponto específico da tela do vídeo. Sua sintaxe é:

```
Posicionar Variável 1, Variável 2
```

onde os conteúdos das variáveis 1 e 2 devem se números inteiros, que indicarão, respectivamente, a linha e a coluna em que o cursor será posicionado.

No exemplo anterior, a sequência de comandos

```
Para j = 1 ate 5
        x = x + i
        Posicionar 10, x
        Se j = 1 entao
                Escrever "(", Vet[j], ","
        Senao
                Se j < 5 entao
                        Escrever Vet[j], ","
                Senao
                        Escrever Vet[j], ")"
                Fim_se
        Fim_se
Proximo
```

exibe na tela, exatamente, (3, 5, 8, 9, 12).

O caso de uma matriz é mais grave. Se a matriz

$$\begin{pmatrix} 2 & 3 & 4 & 5 \\ 3 & 4 & 5 & 6 \\ 4 & 5 & 6 & 7 \end{pmatrix}$$

estiver armazenada na variável composta Mat, a estrutura

```
Para i = 1 ate 4
        Para j = 1 ate 3
                Escrever Mat[i, j]
        Proximo
Proximo
```

exibirá na tela os elementos da matriz de seguinte forma:

```
2
  3
    4
      5
        3
          4
            5
              6
                4
                  5
                    6
                      7
```

o que, naturalmente, não é interessante, pois uma matriz só tem interesse se ela estiver escrita em linhas e colunas. Para exibir uma variável composta bidimensional em forma de matriz podemos usar a seguinte estrutura

```
y = 1
Para i = 1 ate m
        x = 1
        Para j = 1 ate n
                Posicionar y, x
                Escrever Mat[i, j]
                x = x + 4
        Proximo
        y = y + 1
Proximo
```

Para a formatação da saída e para a criação de interfaces, o ILA ainda disponibiliza um comando cuja execução limpa uma área da tela do vídeo. Este comando é ativado através da seguinte sintaxe:

Limpar Expr1, Expr2, Expr3, Expr4

onde Expr1, Expr2, Expr3 e Expr4 são expressões que resultam valores inteiros e indicam as coordenadas do canto esquerdo superior (Expr1, Expr2) e do canto direito inferior (Expr3, Expr4) da área a ser limpa. Se Expr1, Erpr2, Expr3, Expr4 forem omitidos, o sistema adota os valores padrões 1, 1, 24 e 80, que são as coordenadas do canto esquerdo superior e as do canto direito inferior da tela. Ou seja, o comando *Limpar* equivale a *Limpar* 1, 1, 24, 80. Aproveitando o ensejo, valores tomados como padrão por um sistema são chamados valores *default*.

Outra instrução útil para a formatação da saída é o comando *Janela* que desenha uma moldura numa posição da tela fixado pelo programador. Este comando é ativado através da seguinte sintaxe:

Janela Expr1, Expr2, Expr3, Expr4

onde, como no comando *Limpar*, Expr1, Expr2, Expr3 e Expr4 são expressões que resultam valores inteiros e indicam as coordenadas do canto esquerdo superior (Expr1, Expr2) e do canto direito inferior (Expr3, Expr4) da moldura.

Finalmente, o comando *Cor* que permite se fixar cores distintas para a "frente" do vídeo (textos, molduras, etc.) e para o "fundo" do vídeo. A sua sintaxe é, simplesmente,

Cor Expr1, Expr2

onde Expr1 e Expr2 podem ser variáveis do tipo *Numerico* ou do tipo *Caracter* ou, ainda, valores constante destes tipos de dados, de acordo com a tabela a seguir.

Numerico	Caracter	Numerico	Caracter
0	preto	9	azul_intenso
1	azul	10	verde_intenso
2	verde	11	ciano_intenso
3	ciano	12	vermelho_intenso
4	vermelho	13	magenta_intenso
5	magenta	14	marrom_intenso
6	marrom	15	amarelo
7	cinza	16	branco
8	preto_intenso		

6.4 Exercícios propostos

1. Escreva um programa que armazene uma relação de números na ordem inversa da ordem original. Por exemplo, se a relação dada for (3, 6, 8, 9) o programa deve armazenar (9, 8, 6, 3).

2. Escreva um programa que receba um vetor e o decomponha em dois outros vetores, um contendo as componentes de ordem ímpar e o outro contendo as componentes de ordem par. Por exemplo, se o vetor dado for (3, 5, 6, 8, 1, 4, 2, 3, 7), o vetor deve gerar os vetores (3, 6, 1, 2, 7) e (5, 8, 4, 3).

3. Escreva um programa que receba um vetor de números inteiros e o decomponha em dois outros vetores, um contendo as componentes de valor ímpar e o outro contendo as componentes de valor par. Por exemplo, se o vetor dado for (3, 5, 6, 8, 1, 4, 2, 3, 7) o programa deve gerar os vetores (3, 5, 1, 3, 7) e (6, 8, 4, 2).

4. Um *vetor* do R^n é uma n-upla de números reais (x_1, x_2, ..., x_n), sendo cada x_i chamado de *componente*. A *norma* de um vetor (x_1, x_2, ..., x_n) é definida por $\sqrt{x_1^2 + x_2^2 + ... + x_n^2}$. Escreva um programa que receba um vetor do R^n, n dado, e forneça sua norma.

5. O *produto escalar* de dois vetores do R^n é a soma dos produtos das componentes correspondentes. Isto e, se u = (x_1, x_2, ..., x_n) e v = (y_1, y_2, ..., y_n), o *produto escalar* é $x_1.y_1 + x_2.y_2 ... + x_n.y_n$. Escreva um programa que receba dois vetores do R^n, n dado, e forneça o produto escalar deles.

6. A *amplitude* de uma relação de números reais é a diferença entre o maior e o menor valores da relação. Por exemplo, a *amplitude* da relação 5, 7, 15, 2, 23 21, 3, 6 é 23 - 2 = 21. Escreva um programa que receba uma relação de números e forneça sua *amplitude*.

7. Os *desvios* de uma relação de números reais são as diferenças entre cada número e a média aritmética da relação. O *desvio médio* de uma relação de números reais é a média aritmética dos valores absolutos dos *desvios*. Escreva um programa que receba uma relação de números reais e forneça o seu *desvio médio*.

8. O *desvio padrão* de uma relação de números reais é a raiz quadrada da média aritmética dos quadrados dos desvios. Escreva um programa que receba uma relação de números reais e forneça o seu *desvio padrão*.

9. Escreva um programa que forneça as componentes distintas de um vetor dado. Por exemplo, se o vetor dado for (3, 2, 1, 3, 4, 1, 5, 5, 2) o programa deve fornecer (3, 2, 1, 4, 5).

10. O exemplo 3 da seção 2.9 pedia um programa para extrair o algarismo da casa das unidades de um inteiro dado. Aparentemente esta questão não tem interesse prático. Vejamos um problema cuja solução depende desta questão. Algumas empresas que realizam sorteios de prêmios entre seus clientes o fazem através dos sorteios da loteria federal, sendo ganhador o número formado pelos algarismos das casas das unidades dos números sorteados no cinco prêmios da referida loteria. Por exemplo, se o sorteio da loteria federal deu como resultado os números 2345**1**, 0023**4**, 1123**6**, 0123**5** e 2345**2**, o prêmio da tal empresa seria dado ao cliente que possuísse o bilhete de número **14652**. Escreva um programa que receba os números sorteados pela loteria federal e forneça o número que ganhará o prêmio de acordo com as regras acima (vale observar que o programa não funcionará corretamente se um dos números sorteados for maior que 32767, conforme observação final da seção 2.6. Não se preocupe com isto! O que nos interessa aqui não é o programa e sim a aprendizagem da lógica de programação).

11. Escreva um programa que insira um valor dado num vetor dado numa posição dada. Por exemplo se o vetor dado for (3, 8, 5, 9, 12, 3), o valor dado for 10 e a posição dada for 4, o programa deve fornecer (3, 8, 5, **10**, 9, 12, 3).

12. Escreva um programa que insira um valor dado num vetor ordenado dado de modo que o vetor continue ordenado. Por exemplo, se o vetor dado for (2, 5, 7, 10, 12, 13) e o valor dado for 6, o programa deve fornecer o vetor (2, 5, **6**, 7, 10, 12, 13).

13. Escreva um programa que delete uma componente de ordem dada de um vetor dado. Por exemplo, se o vetor dado for (2, 5, 7, 10, 12, 13) e a componente a ser deletada for a de ordem 4, programa deve fornecer o vetor (2, 5, 7, 12, 13).

14. Escreva um programa que, dadas duas relações de números, cada uma delas com números distintos, forneça os números que aparecem nas duas listas. Por exemplo. se as relações forem (9, 32, 45, 21, 56, 67, 42, 55) e (24, 42, 32, 12, 45, 11, 67, 66, 78), o programa deve fornecer o vetor (32, 45, 67, 42).

15. Escreva um programa que, dado um vetor ordenado, forneça a maior diferença entre duas componentes consecutivas, fornecendo também as ordens das componentes que geraram esta maior diferença. Por exemplo, se o vetor dado for (3, 5, 9, 16, 17, 20, 26, 31), o programa deve fornecer como maior diferença o valor 7 (16 - 9), e as ordens 4 e 3.

16. Uma avaliação escolar consiste de 50 questões objetivas, cada uma delas com 5 opções, (1, 2, 3, 4 e 5), sendo apenas uma delas verdadeira. Escreva um programa que receba a sequência de respostas corretas, o *gabarito*, e corrija um cartão-resposta dado.

17. Escreva um programa que forneça o valor numérico de um polinômio P(x) dado, para um valor de x dado. Por exemplo, se o polinômio dado for $P(x) = x^3 + 2x - 1$ e o valor de x dado for 2, o programa deve fornecer $P(2) = 2^3 + 2x2 - 1 = 11$.

18. O(s) valor(es) de maior frequência de uma relação de valores numéricos é(são) chamado(s) *moda* da relação. Escreva um programa que receba uma relação de notas escolares maiores do que zero e menores do que ou iguais a 10, com uma casa decimal, e forneça a(s) moda(s) desta relação. Por exemplo, se a relação de notas for (8,0; 3,5, 4,5; 8,0; 6,0; 4,5; 6,0; 3,5; 2,5; 6,0; 9,0) o programa deve fornecer o valor 6,0 (frequência 3).

19. Escreva um programa que receba um número inteiro n e forneça o número formado pelos algarismos de n escritos na ordem inversa. Por exemplo se o número dado for 3876, o programa deve fornecer 6783.

20. A matemática prova que a conversão de um número do sistema decimal para o sistema binário pode ser feita através de divisões sucessivas do número e dos quocientes sucessivamente obtidos por 2, sendo então o número binário dado pela sequência iniciada por 1 e seguida pelos

restos obtidos nas divisões sucessivas, na ordem inversa em que são obtidos. Por exemplo, para se converter 22 do sistema decimal para o sistema binário temos: Resto(22, 2) = 0; Resto(11, 2) = 1; Resto(5, 2) = 1; Resto(2, 2) = 0 e, portanto, 22s = $(10110)_2$. Escreva um programa que converta um número positivo dado no sistema decimal de numeração para o sistema binário, usando o algoritmo acima.

21. O exercício 10 da seção 4.5 solicitava um programa que determinasse a *decomposição em fatores primos*, fornecendo os fatores primitivos e suas respectivas *multiplicidades*. Na ocasião os fatores primos e suas multiplicidades eram apenas exibidos não sendo armazenados. Modifique o programa referido para que os fatores primos e as suas multiplicidades sejam armazenados, antes de serem exibidos.

22. A Universidade Federal de Alagoas adota o sistema de verificação de aprendizagem listado no exemplo 5 da seção 3.3, com o adendo de que terá direito a uma *reavaliação* um aluno que obtiver uma nota inferior a 7,0 em algum bimestre. Neste caso, a nota obtida na reavaliação substitui a menor das notas bimestrais obtidas. Escreva um programa que, recebendo as notas das avaliações bimestrais e, se for o caso, a nota da reavaliação e, se for o caso, a nota da prova final, forneça a média final de um aluno da UFAL e a sua condição em relação à aprovação.

23. Escreva um programa que forneça a *transposta* de uma matriz dada.

24. Um dos métodos para a se estudar as soluções de um *sistema linear de n equações a n incógnitas* aplica *operações elementares sobre as linhas da matriz dos coeficientes*, sendo a permuta de duas linhas uma destas operações elementares. Escreva um programa que permute as posições de duas linhas de uma matriz dadas.

25. Uma matriz quadrada é dita *triangular* se os elementos situados acima de sua diagonal principal são todos nulos. Escreva um programa que receba uma matriz quadrada e verifique se ela é *triangular*.

26. O exemplo 7 da seção 6.2 apresentou um programa para armazenar uma matriz simétrica. Este exercício quer algo contrário: escreva um programa que verifique se uma matriz dada é simétrica.

27. Escreva um programa que receba uma matriz e totalize suas colunas. Por exemplo, se a matriz dada for $\begin{pmatrix} 2 & 6 \\ 5 & 8 \\ 6 & 9 \end{pmatrix}$, o programa deve fornecer a matriz $\begin{pmatrix} 2 & 6 \\ 5 & 8 \\ 6 & 9 \\ 13 & 23 \end{pmatrix}$. Naturalmente um programa deste seria utilizado para totalizar as colunas de uma tabela de valores numéricos.

28. Escreva um programa que determine as médias de cada uma das linhas de uma matriz. Por exemplo, se a matriz dada for

$$\begin{pmatrix} 3 & 7 & 4 & 6 \\ 5 & 2 & 3 & 4 \\ 2 & 6 & 3 & 1 \end{pmatrix},$$

o programa deve fornecer a matriz

$$\begin{pmatrix} 3 & 7 & 4 & 6 & 5{,}0 \\ 5 & 4 & 5 & 4 & 4{,}5 \\ 2 & 6 & 5 & 1 & 3{,}5 \end{pmatrix}$$

29. Escreva um programa que determine o menor valor de cada uma das linhas de uma matriz dada, fornecendo o índice da coluna que contém este menor valor. Por exemplo, se a matriz dada for $\begin{pmatrix} 3 & 7 & 4 & 6 \\ 5 & 2 & 3 & 4 \\ 2 & 6 & 3 & 1 \end{pmatrix}$, o programa deve fornecer uma tabela do tipo

Linha	Menor valor	Coluna
1	3	1
2	2	2
3	1	4

Um programa como este poderia receber os preços de diversos produtos praticados por vários

supermercados e forneceria, para cada produto, o menor preço e o supermercado que pratica este melhor preço.

30. No exemplo 7 da seção anterior vimos como armazenar uma matriz simétrica. Na prática, uma matriz deste tipo ocorre, por exemplo, numa tabela de distâncias entre cidades, como a seguinte tabela que dá as distâncias aéreas, em Km, entre as capitais dos estados nordestinos (Aracaju, Fortaleza, João Pessoa, Maceió, Natal, Recife, Salvador, São Luís, Teresina).

	A	F	JP	M	N	R	S	SL	T
A	0	812	438	210	550	398	267	1218	1272
F	812	0	562	730	444	640	1018	640	432
JP	418	562	0	284	144	110	758	1208	987
M	210	730	294	0	423	191	464	1220	1126
N	550	414	144	423	0	252	852	1064	843
R	398	640	118	191	252	0	654	1197	935
S	267	1018	758	464	852	654	0	1319	1000
SL	1218	640	1208	1220	1064	1197	1319	0	320
T	1272	432	987	1126	843	935	1000	320	0

Imagine que uma companhia de transporte aéreo estabeleça que uma viagem entre duas cidades que distem mais de 400 Km deve ter uma escala. Escreva um programa que armazene uma tabela das distâncias aéreas entre n cidades e dadas duas cidades determine, se for o caso, a cidade em deve se realizar uma escala para que o percurso seja o menor possível. Por exemplo, nas condições estabelecidas, a viagem entre Maceió e São Luís deve ter uma escala em Fortaleza (o percurso Maceió/Fortaleza/São Luís é de 1370 Km; o percurso, por exemplo, Maceió/Recife/São Luís é de 1388 Km)

31. Esta questão não envolve variáveis compostas. Ela se encontra neste capítulo para se exercitar formatação de saída. Escreva um programa para gerar uma tabuada para multiplicação, exibindo-a na forma usual de tabuadas.