algoritmos e programação

com exemplos em Pascal e C

nina edelweiss
maria aparecida castro livi

bookman

as autoras

Nina Edelweiss é engenheira eletricista e doutora em Ciência da Computação pela Universidade Federal do Rio Grande do Sul. Durante muitos anos, lecionou em cursos de Engenharia e de Ciência da Computação na UFRGS, na UFSC e na PUCRS. Foi, ainda, orientadora do Programa de Pós-Graduação em Ciência da Computação da UFRGS. É coautora de três livros, tendo publicado diversos artigos em periódicos e em anais de congressos nacionais e internacionais. Participou de diversos projetos de pesquisa financiados por agências de fomento como CNPq e FAPERGS, desenvolvendo pesquisas nas áreas de bancos de dados e desenvolvimento de software.

Maria Aparecida Castro Livi é licenciada e bacharel em Letras, e mestre em Ciência da Computação pela Universidade Federal do Rio Grande do Sul. Desenvolveu sua carreira profissional na UFRGS, onde foi programadora e analista de sistema, antes de ingressar na carreira docente. Ministrou por vários anos a disciplina de Algoritmos e Programação para alunos dos cursos de Engenharia da Computação e Ciência da Computação. Sua área de interesse prioritário é o ensino de Linguagens de Programação, tanto de forma presencial quanto a distância.

E22a Edelweiss, Nina.
Algoritmos e programação com exemplos em Pascal e C [recurso eletrônico] / Nina Edelweiss, Maria Aparecida Castro Livi. – Dados eletrônicos. – Porto Alegre : Bookman, 2014.

Editado também como livro impresso em 2014.
ISBN 978-85-8260-190-7

1. Informática. 2. Algoritmos – Programação. I. Livi, Maria Aparecida Castro. II. Título.

CDU 004.421

Catalogação na publicação: Ana Paula M. Magnus – CRB 10/2052

7.1 → matrizes

Matrizes são arranjos de duas ou mais dimensões. Assim como nos vetores, todos os elementos de uma matriz são do mesmo tipo, armazenando informações semanticamente semelhantes. No exemplo da Figura 7.2, todos os elementos da matriz `Notas` são do tipo `real` e guardam valores correspondentes a notas de alunos.

Matrizes bidimensionais são arranjos de duas dimensões. São necessários dois índices para acessar cada um de seus elementos, um para cada uma das dimensões. Matrizes de mais de duas dimensões, de forma similar, necessitam de um índice para cada dimensão para definir um elemento de forma única.

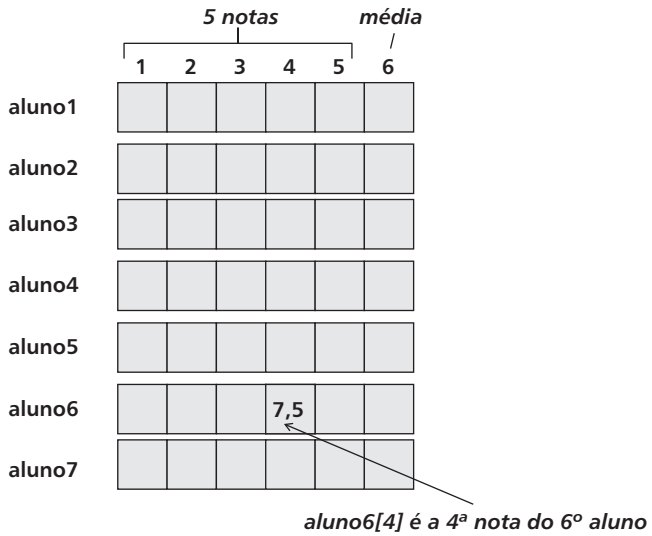

figura 7.1 Um vetor para cada aluno.

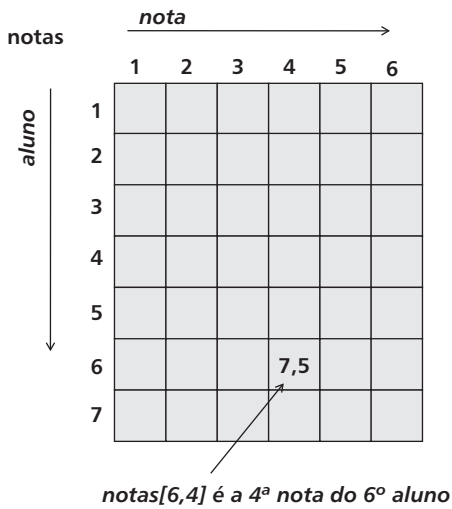

figura 7.2 Uma matriz para todos os alunos.

7.2 ⇢ matrizes bidimensionais

Em uma **matriz de duas dimensões**, ou **bidimensional**, todos os elementos são do mesmo tipo, identificados por um índice para cada dimensão. No mundo real, os teatros são exemplos práticos de matrizes bidimensionais. Uma poltrona, em um teatro, tem a si associada a especificação da fila na qual se situa (fila A, B, C, etc.), bem como a especificação que define a posição da poltrona dentro da fila (1, 2, 3, 4...). Se uma entrada de teatro é rasgada e nela só resta uma dessas duas identificações, ou de fila ou de poltrona, não é possível mais determinar com precisão a poltrona referenciada. Só as duas referências juntas, de fila e poltrona, permitem especificar de forma única uma poltrona no teatro.

7.2.1 declaração de uma matriz bidimensional

Em pseudolinguagem, uma matriz bidimensional é declarada com o seguinte tipo:

```
arranjo [<índice menor da dimensão 1>..<índice maior da dimensão 1>,
        <índice menor da dimensão 2>..<índice maior da dimensão 2> ]
         de <tipo da matriz>)
```

onde `<índice menor da dimensão n>` e `<índice maior da dimensão n>` são os valores limites que definem o intervalo de índices válidos para cada uma das dimensões, e `<tipo da matriz>` é o tipo dos dados passíveis de serem armazenados nessa matriz.

A declaração a seguir corresponde à matriz da Figura 7.2:

```
Variável: notas (arranjo [1..7, 1..6] de real)
```

onde `notas` é o nome comum a todos os elementos da matriz, `1..7` é o intervalo de valores válidos para o índice da primeira dimensão (aluno), `1..6` é o intervalo de valores válidos para o índice da segunda dimensão (nota), e `real` é o tipo da matriz, ou seja, o tipo de todos os seus elementos. Em matrizes bidimensionais, costuma-se dizer que a primeira dimensão corresponde às linhas e a segunda, às colunas.

7.2.2 acesso a um elemento de uma matriz

Para acessar um determinado elemento de uma matriz deve-se usar um índice para cada uma de suas dimensões. Os índices, desde que dentro dos intervalos válidos, podem ser constantes, variáveis ou expressões aritméticas.

A seguir, alguns exemplos de **acesso a um elemento de uma matriz**, considerando a matriz bidimensional `notas` antes declarada:

- primeira nota do primeiro aluno:
  `notas[1,1]`
- primeira nota do aluno 3, considerando i = 3 e j = 1 (sendo `i` e `j` variáveis inteiras):
  `notas[i, j]`
- quinta nota do segundo aluno, considerando i = 4:
  `notas[2, i+1]`

Um elemento de uma matriz é utilizado em comandos da mesma forma que uma variável simples, já que corresponde a um único valor. Por exemplo, o trecho abaixo preenche um elemento por leitura, copia esse valor para outro elemento da matriz, verifica se o valor lido é igual a 10 e, em caso positivo, imprime o valor:

```
ler (notas[1,1])
notas[1,2] ← notas[1,1]
se notas [1,1] = 10
então escrever (notas[1,1])
```

7.2.3 inicialização de matrizes

No que se refere à inicialização, o que foi colocado para vetores vale também para matrizes. Se a matriz é totalmente preenchida por leitura, não é necessário inicializá-la, uma vez que todos os valores anteriores das posições de memória da matriz são descartados quando novos valores nelas são colocados. Mas se a matriz for preenchida apenas parcialmente, restando posições não ocupadas no seu início, meio ou fim, ou se for utilizada em totalizações, cuidados devem ser tomados para garantir que os valores iniciais de todas as suas posições sejam os desejados.

O trecho a seguir inicializa de forma completa, com zeros, uma matriz de 10 elementos em cada dimensão, com índices no intervalo de 1 a 10:

```
para i de 1 incr 1 até 10 faça
    para j de 1 incr 1 até 10 faça
        valor[i,j] ← 0
```

Dependendo do tipo do problema, por vezes também é possível inicializar, imediatamente antes do uso, apenas as posições da matriz que serão utilizadas.

7.2.4 exemplos de uso de matrizes

Nos exemplos a seguir, sempre que todos os elementos da matriz são acessados, a variação dos índices de linha e coluna é realizada usando comandos `para/faça` encadeados. Embora essa não seja a única solução possível em tais casos, certamente é a mais amplamente utilizada, pois esse encadeamento garante que, para cada valor do índice alterado pelo laço externo, tenha-se toda a variação do índice alterado pelo laço interno, permitindo o acesso ordenado aos elementos das matrizes, linha após linha e, em cada linha, coluna após coluna.

Dada a constante `MAX` e uma matriz quadrada, definida por `tabela(MAX, MAX)`, a seguir são apresentadas algumas operações sobre essa matriz.

a Preenchimento da matriz por leitura, percorrendo linha por linha:

```
para i de 1 incr 1 até MAX faça
  para j de 1 incr 1 até MAX faça
    ler(tabela[i,j])
```

b Escrita da matriz completa, percorrendo linha por linha:

```
para i de 1 incr 1 até MAX faça
  para j de 1 incr 1 até MAX faça
    escrever(tabela[i,j])
```

c Somatório dos elementos da última coluna:

```
soma ← 0
para i de 1 incr 1 até MAX faça
  soma ← soma + tabela[i,MAX]
escrever('Somatório dos elementos da última coluna = ', soma)
```

d Somatório dos elementos da diagonal principal. Observe que neste caso, teoricamente, dois índices deveriam ser usados para acesso à matriz `tabela`, já que ela é bidimensional. No entanto, como os elementos da diagonal principal têm índices com valores iguais para linha e coluna (0,0 – 1,1 – 2,2 – etc.), uma única variável pode ser usada como índice para as duas dimensões:

```
soma ← 0
para i de 1 incr 1 até MAX faça
  soma ← soma + tabela[i,i]
escrever('Somatório dos elementos da diagonal principal = ', soma)
```

e Somatório dos elementos da diagonal secundária. Observar que, a partir do índice da linha, é possível calcular o índice da coluna, o que, mais uma vez, permite resolver o problema usando apenas uma variável como índice:

```
soma ← 0
para i de 1 incr 1 até MAX faça
  soma ← soma + tabela[i,((MAX - i) + 1)]
escrever('Somatório elementos da diagonal secundária = ', soma)
```

f Somatório de todos os elementos à esquerda da diagonal secundária:

```
soma ← 0
para i de 1 incr 1 até MAX faça
  para j de 1 incr 1 até MAX faça
    se j < (MAX - i)
    então soma ← soma + tabela[i,j]
escrever
  ('Somatório dos elementos à esquerda da diagonal secundária = ',
  soma)
```

g Geração de dois vetores, um com o somatório das linhas (`som_lin[MAX]`) e o outro com o das colunas (`som_col[MAX]`):

```
para i de 1 incr 1 até MAX faça  {INICIALIZA VETORES EM ZERO}
  início
  som_lin[i] ← 0
  som_col[i] ← 0
  fim
para i de 1 incr 1 até MAX faça
  para j de 1 incr 1 até MAX faça
    início
    som_lin[i] ← som_lin[i] + tabela[i,j]
    som_col[j] ← som_col[j] + tabela[i,j]
    fim
```

7.3 → matrizes com mais de duas dimensões

Estendendo um pouco mais o exemplo apresentado no início deste capítulo, o que se faria para armazenar as notas de todos os alunos de mais de uma turma em uma única matriz? Supondo que há 2 turmas, cada turma com 7 alunos, com 6 notas por aluno, a declaração da matriz para armazenar os dados correspondentes seria a seguinte:

```
notas_turma (arranjo [1..2, 1..7, 1..6] de real)
```

As três dimensões são, na ordem em que foi feita a declaração: as turmas, os alunos e, finalmente, as notas. A Figura 7.3 mostra a representação espacial da matriz tridimensional `notas`.

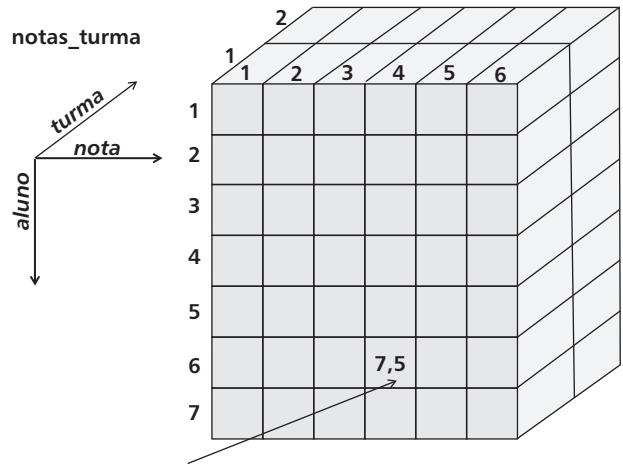

figura 7.3 Matriz tridimensional para notas.

Observa-se, nesse exemplo, que para acessar uma determinada nota é necessário, além do nome da matriz, três informações adicionais que permitam identificar o elemento em cada uma das dimensões, ou seja, um índice para cada uma das três dimensões. Por exemplo, o elemento assinalado na Figura 7.3 requer, para sua identificação, os índices 1, 6 e 4.

Outro problema exige que se armazene, ao longo de 12 meses, a quantidade de itens de um total de 30 produtos que estão dispostos em 10 lojas, cada qual com 5 setores. A matriz para resolver esse problema possuirá quatro dimensões e armazenará a quantidade de itens de cada produto vendido em cada mês, por setor e por loja.

A declaração dessa matriz de quatro dimensões é a seguinte:

```
numitens (arranjo [1..10, 1..5, 1..30, 1..12] de inteiro)
```

As quatro dimensões são, nesta ordem: a loja, o setor, o produto e, finalmente, o mês.

Matrizes multidimensionais podem ter duas ou mais dimensões, embora a grande maioria dos problemas não envolva mais do que três ou quatro. O número de dimensões de uma matriz deverá ser definido em função das necessidades do problema que está sendo analisado e das limitações eventuais da linguagem em uso. Isso porque, embora teoricamente não exista limitação para o número de dimensões de uma matriz, uma implementação particular de uma linguagem pode definir um limite para esse número.

7.5 ⇢ em Pascal

7.5.1 declaração de uma matriz

Um tipo de dado matriz (arranjo de mais de uma dimensão) é declarado em Pascal de forma semelhante ao tipo vetor, incluindo os limites para cada uma das dimensões:

```
array [<limite inferior>..<limite superior>,
       <limite inferior>..<limite superior>, ...] of <tipo>
```

Os limites superior e inferior, índices do primeiro e do último elemento de cada dimensão, devem ser do tipo `inteiro`, `caractere` ou `enumeração` (tipo que será visto no Capítulo 8). Não existem restrições quanto aos valores que podem ser utilizados como índices. O tipo declarado no final é o tipo de todos os elementos da matriz.

Exemplo de declarações de matrizes:

```
var
   nota: array [1..7, 1..6] of real;
   contagem_letras: array ['a'..'j', 1..20] of integer;
```

Em algumas situações, duas variáveis que tenham sido declaradas de forma independente, com tipos não definidos explicitamente e aparentemente iguais, podem não ser reconhecidas como variáveis de mesmo tipo pelo sistema. Por essa razão, recomenda-se que a declaração de uma matriz em Pascal seja feita sempre como está a seguir, definindo primeiro a(s) constante(s) usada(s) na declaração da matriz, em seguida o tipo definido para a matriz e, finalmente, a variável do tipo matriz:

```
const
  MAX1 = 6;
  MAX2 = 8;
type
  matriz = array [1..MAX1, 1..MAX2] of integer;
var
  mat: matriz;
```

7.5.2 acesso aos elementos de uma matriz

O acesso aos elementos de uma matriz em Pascal é feito definindo-se um índice para cada uma das dimensões. Esse índice pode ser dado por um valor constante, pelo conteúdo de uma variável ou pelo resultado de uma expressão, devendo ser do mesmo tipo definido para cada um dos índices, na ordem em que foram definidos. Os comandos a seguir acessam elementos das matrizes definidas na seção anterior:

```
{PREENCHER POR LEITURA O ELEMENTO DA LINHA 3, COLUNA 5 DA MATRIZ nota:}
readln(nota [3, 5]);

{SUPONDO A VARIAVEL INTEIRA i = 2, INFORMAR VALOR CONTIDO NA LINHA 1,
COLUNA 4 DA MATRIZ contagem_letras}
writeln(contagem_letras['a', i+2]);

{COPIAR VALOR DA LINHA 2, COLUNA 1 PARA LINHA 1, COLUNA 2 DE mat}
mat[1,2] := mat[2,1];
```

O trecho a seguir imprime a matriz `mat`, linha por linha, separando seus valores adequadamente. São utilizadas as variáveis inteiras `lin` e `col` para percorrer respectivamente as linhas e as colunas da matriz:

```
for lin := 1 to MAX1 do                {PARA CADA LINHA DA MATRIZ}
   begin
   writeln;          {INICIA NOVA LINHA PARA CADA LINHA DA MATRIZ}
   for col := 1 to MAX2 do          {PARA CADA COLUNA DESTA LINHA}
      write(mat[lin, col]:5, ' ')            {ESCREVE UM ELEMENTO}
   end;
```

7.5.3 inicialização de matriz na declaração

É possível inicializar matrizes na declaração, declarando-as como constantes tipadas. Constantes tipadas, apesar do nome, funcionam na realidade como variáveis, uma vez que seus valores iniciais podem ser alterados durante o processamento.

Declaração de uma matriz como constante tipada:

```
const
   <nome da constante> : <tipo da constante> =
((valor1 da 1ª linha, valor2 da 1ª linha , ... , valorn da 1ª linha),
 (valor1 da 2ª linha, valor2 da 2ª linha , ... , valorn da 2ª linha),
 (valor1 da 3ª linha, ...), (...));
```

O número de valores informados não pode ser nem menor nem maior que o número de valores previstos para cada linha da matriz, caso contrário o código não será compilado sem erros.

Exemplo de inicialização de uma matriz por meio de sua declaração como constante tipada:

```
const
   MAXMERC = 5;
   MAXMES = 12;
type
   mt = array [1..MAXMERC , 1..MAXMES] of integer;
const
    matriz : mt = ((0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1),
                   (50, 51, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5),
                   (5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5),
                   (100, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10),
                   (0, 0, 0, 0, 10, 5, 10, 5, 5, 5, 5, 5));
```

7.5.4 atribuição em bloco

Em Pascal é possível fazer uma atribuição em bloco de um arranjo a outro, desde que tenham exatamente a mesma estrutura.

Supondo que `matriz_original` e `matriz_copia` tenham o mesmo número de dimensões e, em cada uma das dimensões correspondentes (na ordem em que foram definidas), o mesmo número de elementos, a atribuição a seguir copia todos os valores de uma para a outra:

```
matriz_copia := matriz_original;
```

7.6 ⋯→ em C

7.6.1 declaração de uma matriz

A declaração de uma matriz multidimensional em C é feita definindo-se, após o nome da matriz, o número de elementos em cada uma das suas dimensões, cada um entre colchetes:

```
<tipo_da_variável> <nome_da_variável>
    [<número de elementos da 1ª dimensão>]
    [número de elementos da 2ª dimensão] ...;
```

onde `<tipo_da_variável>` é o tipo de todos os elementos da matriz e `<nome_da_variável>` é o nome da matriz.

Como os índices em C iniciam obrigatoriamente em 0, os índices válidos de uma dimensão de um arranjo variam de 0 ao número de elementos dessa dimensão menos 1. A declaração em C da matriz `notas_turma` da Figura 7.3 é:

```
real notas_turma [2] [7] [6];
```

Os índices de cada uma das dimensões da matriz `notas_turma`, iniciando sempre em zero, são respectivamente: 0 a 1, 0 a 6 e 0 a 5.

7.6.2 acesso aos elementos de uma matriz

O acesso aos elementos de uma matriz em C é feito por índices, um para cada dimensão, que podem ser valores constantes, variáveis ou expressões, preferencialmente de tipo `integer` ou `char`.

No trecho a seguir, a matriz `mat_val` é apresentada linha por linha, com as variáveis `linha` e `coluna` sendo usadas respectivamente como índices de linha e coluna e as constantes `MAX1` e `MAX2` como valores máximos de índices:

```
for (linha = 0; linha < MAX1; linha++)
    {
      for (coluna = 0; coluna < MAX2; coluna++)
           printf ("%6.2f    ", mat_val[linha][coluna]);
      printf("\n");
    }
```

7.7 → dicas

usar índices fora de intervalo válido. O erro mais frequente no uso de matrizes é a tentativa de empregar índices com valores que gerem acessos fora dos limites das matrizes. Esse tipo de erro, por implicar no acesso a áreas indevidas de memória, produz por vezes resultados muito estranhos.

verificar correção de dados de entrada usados como índices. Todo dado de entrada que deva estar dentro de um intervalo válido de valores só deve ser aceito se estiver correto. Esse cuidado é particularmente importante quando o dado de entrada vier a ser usado como índice de matrizes.

escrever índices de matrizes multidimensionais em C. Na escrita dos índices de uma matriz em C, devem ser colocados pares de colchetes cercando os índices de cada dimensão. Escrever todos os índices dentro de um único par de colchetes, apenas separados por vírgulas, mesmo que não resulte em erro de sintaxe, resulta em erro lógico.

escrever índices obedecendo à ordem das dimensões das matrizes. Cuidar para referenciar corretamente cada uma das dimensões das matrizes, na ordem em que foram definidas. Por exemplo, se no exemplo da Figura 7.3 as dimensões foram definidas na ordem turmas/alunos/notas, uma referência ao elemento 1,3,2 está se referindo à 2ª nota do 3º aluno da 1ª turma. Errar a ordem dos índices resulta no acesso e utilização de elementos diferentes dos que se quer.

usar constantes. É altamente recomendável utilizar constantes nos limites das dimensões das matrizes e no código que as acessa, sempre que pertinente. Isso facilita os trabalhos de teste e torna o código mais genérico.

evitar processamento desnecessário. Um exemplo típico de processamento desnecessário é o acesso aos elementos da diagonal principal de uma matriz bidimensional. Os valores de índice desses elementos são iguais: 1 1, 2 2, 3 3, etc. Embora soluções com a variação de dois índices sejam por vezes adotadas, um só índice é suficiente para acessar seus elementos.

não expor desnecessariamente o usuário a particularidades da linguagem de programação utilizada. Por exemplo, na linguagem C os índices partem de zero. Mas, no mundo real, o usuário conhece valores que iniciam em 1 e vão até um valor limite. Nesses e em outros casos semelhantes, ao interagir com o usuário, é necessário fazer os ajustes necessários de modo a garantir que ele opere com a informação da forma como está habituado.

7.8 ⇢ testes

testar com versões reduzidas da matriz. Sempre que possível, testar os programas com a matriz reduzida a um pequeno conjunto de valores. Se funcionar para um número reduzido de elementos por dimensão, também funcionará quando esse número for aumentado.

testar com o volume mais próximo possível do real. Fazer pelo menos um teste com o volume aproximado de dados que deve ser o realmente utilizado. Há questões que só surgem com um volume maior de dados.

testar valores limite. Nos testes, verificar, sobretudo, os itens que vão até o limite dos intervalos de valores dos índices.

encontrar e solucionar erros. Verificar inicialmente a declaração da matriz. A maior parte dos problemas surge de declarações incorretas ou inconsistentes com o uso da matriz. Verificar os valores que estão sendo utilizados nos índices. Índices fora de intervalo podem provocar erros em áreas sem relação com as matrizes. Verificar se os valores de índices usados nas dimensões coincidem com o previsto na declaração da matriz.

Encerra aqui o trecho do livro disponibilizado para esta Unidade de Aprendizagem. Na Biblioteca Virtual da Instituição, você encontra a obra na íntegra.