LÓGICA DE PROGRAMAÇÃO

Esp. Marcelo Uemura

INICIAR

Introdução

Nesta unidade, veremos o uso de estruturas de dados compostas homogêneas, do tipo unidimensional, conhecida como vetor, e do tipo multidimensional, que chamamos de matriz.

Até então, as variáveis utilizadas pelas linguagens de programação eram simples, armazenando um espaço de memória para um único valor, de acordo com o seu tipo. Com as estruturas de dados compostas, é possível armazenar um conjunto de valores de modo a serem utilizados como uma tabela, representando um vetor ou matriz.

Nesse sentido, apresentaremos como esses vetores e matrizes podem ser utilizados pelas linguagens de programação, por meio de declaração, carregamento, atribuição, leitura e apresentação.

Vetores

Neste tópico, veremos as estruturas de dados chamadas vetores, que são representadas por variáveis compostas homogêneas. Portanto, estudaremos os princípios básicos das formas de construção de programas, usando estruturas de dados homogênea, o desenvolvimento de programas com adoção de vetores e a avaliação de soluções de programas escritos em uma linguagem de programação.

Variáveis compostas homogêneas

A variável é um elemento que armazena um valor pertencente a um tipo, e pode ser classificada como unitária ou composta. A variável unitária possui apenas um valor único pertencente a um determinado tipo. Já as variáveis compostas são estruturas de dados que representam um conjunto de valores ao mesmo tempo, segundo Medina e Fertig (2006). Essas variáveis podem ser homogêneas (do mesmo tipo) ou heterogêneas (tipos diferentes).

As variáveis compostas homogêneas podem ser chamadas de vetores, compostas por uma estrutura de dados unidimensional, conforme Figura 4.1:

Conforme Forbellone e Eberspächer (2005), para utilizar um vetor é necessário definir em detalhes como é constituído o tipo construído, para, posteriormente, declarar uma variável, associando o identificador dessa ao identificador do tipo vetor.

A sintaxe da definição de um vetor segue a seguinte estrutura (FORBELLONE; EBERSPÄCHER, 2005, p. 69):

tipo identificador = vetor [LI .. LF] de tipo primitivo;

Em que LI representa o limite inicial do vetor e LF o seu limite final. O número de valores dentro do vetor pode ser calculado como LF-LI+1. Por exemplo, a definição de um vetor que contempla 40 valores inteiros:

tipo V = vetor [1..40] de inteiros; // definição do vetor V

Para a declaração de variáveis compostas, declaram-se os vetores de acordo com a sua definição:

V: VA, VB, VC; // declaração de variáveis do tipo vetor V

Na declaração acima, VA, VB e VC são vetores do tipo V, que comportam 40 números inteiros.

Os vetores também são conhecidos como *arrays* (VILARIM, 2004). O preenchimento de cada elemento de um vetor pode ser feito indicando o valor de acordo com a posição (índice) do elemento, ou seja, por meio de uma indexação. Por isso, os vetores também são conhecidos como variáveis indexadas unidimensionais (SOUZA et al., 2011). Por exemplo:

VA[1] = 5;

VA[2] = 8;

VA[3] = 10;

No exemplo apresentado, o primeiro elemento do vetor VA, representado pelo índice 1 (entre colchetes), recebe o valor inteiro 5, o segundo elemento recebe o valor 8 e o terceiro elemento recebe o valor inteiro 10. A seguir, veremos como utilizar vetores no desenvolvimento de programas.

Desenvolvimento de programas com vetores

Para trabalhar o desenvolvimento de programas com vetores, vamos considerar o seguinte problema: queremos apresentar a quantidade de alunos que tiveram a nota acima da média de uma turma de 10 alunos. Para resolver esse problema, podemos resolver através de vetores com o seguinte algoritmo (FORBELLONE; EBERSPÄCHER, 2005, p. 73):

1. *início*
    a. *tipo Classe = vetor [1..10] de reais; // definição do vetor*
    b. *Classe: VClasse;*
    c. *real: Soma, Média;*
    d. *inteiro: NotaAcima, X;*
    e. *Soma = 0;*
    f. *NotaAcima = 0;*
    g. *para X de 1 até 10 passo 1 faça*
        i. *leia (VClasse[X]); // lê os valores das notas e armazena no vetor*

    h. *fimpara;*
    i. *para X de 1 até 10 passo 1 faça*
        i. *Soma = Soma + VClasse[X]; //acumular soma de valores*

    j. *fimpara;*
    k. *Média = Soma/10 // cálculo da média*
    l. *para X de 1 até 10 passo 1 faça*
        i. *se (VClasse[X] > Média) // nota acima da média*

ii. *então NotaAcima = NotaAcima +1; // contagem de alunos acima da média*
iii. *fimse;*

m. *fimpara;*
n. *escreva(NotaAcima); // número de valores acima da média*

2. *fim*

Neste exemplo, podemos verificar o uso do vetor como uma estrutura de dados para armazenar as notas lidas, e posteriormente, calcular a média e identificar a quantidade de alunos com nota acima da média. A seguir, verificaremos como os vetores podem ser utilizados através de uma linguagem de programação.

1.3 Uso de vetores em linguagem de programação

Na linguagem de programação C, o uso de vetor pode ser feito por meio de definições com *array*. A sintaxe é:

Tipo nome_do_vetor[quantidade de itens]

Exemplo:

int V[10]; //vetor com 10 elementos inteiros

A inicialização do vetor pode ser feita para todos ou alguns elementos. Exemplo:

int V[5] = {1, 2, 3, 4, 5} //inicializando todos os elementos

int V[5] = {1,2,3} //inicializando alguns elementos, nesse caso é equivalente a {1, 2, 3, 0, 0}

int V[] = {1, 2, 3} // inicializando um vetor sem especificar a quantidade de elementos

Para entender o uso de vetores na linguagem C, veremos um exemplo de programa.

```c
#include<stdio.h>
#include<conio.h>
int main(void)
{
float notas[5] = {7, 8, 5, 9.9, 5.2};
// declarando e inicializando o vetor notas

printf("Exibindo os Valores do Vetor \n\n");
printf("notas[0] = %.1f\n", notas[0]);
printf("notas[1] = %.1f\n", notas[1]);
printf("notas[2] = %.1f\n", notas[2]);
printf("notas[3] = %.1f\n", notas[3]);
printf("notas[4] = %.1f\n", notas[4]);

getch();
return 0;
}
```

Fonte: Casavella (2013b).

No exemplo apresentado, o vetor foi inicializado e posteriormente apresentado com a instrução “printf”. No tópico a seguir, serão vistas a leitura, carga e apresentação dos vetores.

atividade

Atividade

Os vetores são variáveis unidimensionais, que permitem o armazenamento de um conjunto de valores do mesmo tipo. Logo, a estrutura de dados dos vetores é:

- ○ **a)** composta homogênea
- ○ **b)** composta heterogênea
- ○ **c)** unitária homogênea
- ○ **d)** unitária heterogênea
- ○ **e)** unitária unidimensional

Vetores - Parte 2

Neste tópico, será apresentado o uso de vetores com relação a sua carga, leitura e apresentação, dentro do contexto da lógica e linguagem de programação.

Carregando vetores

A carga de um vetor pode ser feita baseada no índice de forma individual, conforme mencionado por Souza et al. (2011). O índice também pode ser utilizado para encontrar o valor dentro de um vetor, tendo em vista que é um conjunto de elementos armazenados simultaneamente do mesmo tipo.

O índice é representado dentro de colchetes “[ ]”, indicando um elemento específico dentro de um vetor. Por exemplo, A[1] indica o primeiro elemento de um vetor A, assim como A[2] o seu segundo elemento.

É possível que uma variável receba o valor de um elemento de um vetor, por exemplo:

x = A[3], a variável x recebe o valor do terceiro elemento de um vetor A.

Também podem ser utilizados operadores aritméticos com vetores, como:

A[4] = 2*A[4], o elemento de um vetor recebe o dobro do seu valor.

A[3] = x + A[2], o terceiro elemento do vetor A recebe o valor do segundo elemento somado com a variável x.

É muito comum utilizar estruturas de repetição para carregar dados em um vetor, principalmente quando há uma lógica para os valores a serem atribuídos. Por exemplo:

para x de 1 até 100 passo 1 faça

escreva (A[x], numero);

numero = numero +1;

fimpara;

Nesse exemplo, o vetor A[x] é preenchido com 100 valores da variável número, que é incrementada a cada repetição.

Lendo e mostrando vetores

Assim como a carga de vetores é realizada com base do índice, a leitura também utiliza essa referência. Assim, para fazer a leitura do 4.º elemento de um vetor Z, deve-se fazer a leitura do conteúdo de Z[4].

Assim, considerando que os seguintes elementos de um vetor tenham sido carregados como segue:

Z[1] = 10

Z[2] = 20

Z[3] = 25

Z[4] = 30

Z[5] = 50

A leitura dos elementos de um vetor pode ocorrer com base no índice, como no exemplo a seguir:

x = Z[3]

Nesse caso, a variável x irá receber o valor do 3.º elemento do vetor Z, no caso 25. Também utilizando uma estrutura de laço de repetição, é possível apresentar os valores de um vetor:

para x de 1 até 5 passo 1, faça

escreva ("o número de elemento", x, "é", Z[x])

fimpara;

Assim, nesse laço de repetição, todos os valores do vetor de 5 elementos serão mostrados.

A linguagem de programação C também utiliza dessa sintaxe, procedimento para fazer a leitura e mostrar os valores de um vetor. Vejamos o exemplo, no programa a seguir:

```
1. #include<stdio.h>
2. #include<conio.h>
3. int main(void)
4. {
5.
6. int i;
7. float notas[5] = {7, 8, 5, 9.9, 5.2};
8. // declarando e inicializando o vetor notas
9.
10. printf("Exibindo os Valores do Vetor \n\n");
11.
12. for( i = 0 ; i <= 4; i++)
13. {
14. printf("notas[%d] = %.1f\n",i, notas[i]);
```

```
15. }
16.
17. getch();
18. return 0;
19. }
```

Fonte: Casavella (2013b).

Podemos perceber que as notas armazenadas em um vetor serão apresentadas com base no seu índice dentro de um laço de repetição “for”.

Assim, conseguimos avaliar os princípios dos vetores, a forma de carregamento, leitura e apresentação. No próximo tópico, iremos estudar uma outra modalidade de variáveis compostas homogêneas, porém multidimensionais, denominadas matrizes.

Atividade

Analise o exemplo de código a seguir, que faz o uso dos seguintes vetores:

```
int i;

int n = 5;

float notas[5] = {10, 9.5, 7, 8.5, 8};

float soma = 0;

float media = 0;

for(i=0;  i<n ; i++,)

{

        soma =soma+notas[i];

}

media = soma/n;

return media;
```

Agora, assinale a alternativa que indique qual o valor da variável média após a execução desse trecho de código.

- ○ **a)** 7
- ○ **b)** 8,6
- ○ **c)** 8
- ○ **d)** 9
- ○ **e)** 7,6

Matrizes

Serão apresentados, neste tópico, os princípios básicos das formas de construção de programas usando a estrutura de dados homogênea denominada matriz. Portanto, serão desenvolvidos programas utilizando matriz. Além disso, aprenderemos como avaliá-los utilizando uma linguagem de programação.

Introdução

Os vetores são estruturas de dados compostos homogêneos, indexadas e unidimensionais. Existem estruturas de dados bidimensionais, que são conhecidas como tabelas, arranjos bidimensionais ou, simplesmente, matrizes (SOUZA et al., 2011).

Uma matriz de duas dimensões (bidimensional) é composta por dois índices, que representam linhas e colunas. Na Tabela 4.1, a seguir, temos a representação de uma matriz contendo três linhas e três colunas.

É possível o uso de matrizes com mais de duas dimensões (multidimensional), sendo que as dimensões adicionais correspondem a novos eixos, sendo utilizados nas tabelas, além das linhas e colunas. As matrizes podem ser utilizadas em linguagens de programação para a organização de dados em estruturas baseadas em mais de uma dimensão, diferenciando-as dos vetores. A seguir, veremos com uma matriz pode ser declarada em linguagens de programação.

Declaração de matrizes

A declaração de uma matriz pode ser realizada por meio da seguinte sintaxe (FORBELLONE, EBERSPÄCHER, 2005):

tipo identificador = matriz [LI1..LF1, LI2..LF2] de tipo primitivo;

Em que LI1..LF1, LI2..LF2 são os limites dos intervalos de variação dos índices da variável, em que cada par (LI, LF) está associado a um índice da matriz.

O número de elementos de uma matriz pode ser assim calculado:

número de elementos da matriz = (LF1-LI1+1)(LF2-LI2+1)*...*(LFn-LIn+1)*

Exemplos de declaração de matriz:

tipo M = matriz [1..4,1..4] de inteiros;

M: MAT;

Em que MAT tem duas dimensões, contendo 16 elementos.

Na linguagem de programação C, podemos utilizar a seguinte sintaxe para declaração:

tipo identificador [linhas][colunas]

Como exemplo, podemos citar:

float M[4][2];

Em que o valor [4] representa a quantidade de linhas e [2] a quantidade de colunas. Isso nos leva a uma matriz contendo oito elementos.

Atribuição de matrizes

Para a atribuição de matrizes, deve-se ter a matriz declarada previamente. Baseado na combinação dos índices (ex: linha X coluna), pode-se atribuir um valor ao elemento indexado da matriz.

Por exemplo,

tipo M = matriz [1..4,1..2] de reais;

M: MAT;

MAT[1,1] = 1; // atribui o valor 1 para o elemento da primeira linha e primeira coluna.

MAT[1,2] = 3; // atribui o valor 3 para o elemento da primeira linha e segunda coluna.

MAT[2,1] = 2; // atribui o valor 2 para o elemento da segunda linha e primeira coluna.

MAT[2,2] = 4; // atribui o valor 4 para o elemento da segunda linha e segunda coluna.

MAT[3,1] = 5; // atribui o valor 5 para o elemento da terceira linha e primeira coluna.

MAT[3,2] = 7; // atribui o valor 7 para o elemento da terceira linha e segunda coluna.

MAT[4,1] = 6; // atribui o valor 6 para o elemento da quarta linha e primeira coluna.

MAT[4,2] = 8; // atribui o valor 1 para o elemento da quarta linha e segunda coluna.

O mesmo exemplo utilizando a linguagem de programação C:

float M[4][2];

M[0][0] = 1; // atribui o valor 1 para o elemento da primeira linha e primeira coluna.

M[0][1] = 3; // atribui o valor 3 para o elemento da primeira linha e segunda coluna.

M[1][0] = 2; // atribui o valor 2 para o elemento da segunda linha e primeira coluna.

M[1][1] = 4; // atribui o valor 4 para o elemento da segunda linha e segunda coluna.

M[2][0] = 5; // atribui o valor 5 para o elemento da terceira linha e primeira coluna.

M[2][1] = 7; // atribui o valor 7 para o elemento da terceira linha e segunda coluna.

M[3][0] = 6; // atribui o valor 6 para o elemento da quarta linha e primeira coluna.

M[3][1] = 8; // atribui o valor 1 para o elemento da quarta linha e segunda coluna.

Aqui, temos valores numéricos inteiros atribuídos para todos os elementos da matriz mencionada, conforme podemos observar na Tabela 4.2:

Tabela 4.2 – Matriz 4 X 2
Fonte: Elaborada pelo autor.

No tópico a seguir, veremos como carregar, ler e apresentar os elementos de uma matriz.

Atividade

Considere uma matriz contendo a seguinte declaração:

tipo M = matriz [1..4, 1..3, 1..3] de inteiros

Nesse sentido, assinale a alternativa que indique qual a quantidade de elementos desta matriz?

- ○ **a)** 16 elementos
- ○ **b)** 48 elementos
- ○ **c)** 24 elementos
- ○ **d)** 12 elementos
- ○ **e)** 36 elementos

Matrizes - Parte 2

Neste tópico, serão vistas as formas de carregamento, leitura e apresentação das matrizes, em especial por meio da linguagem de programação.

Carregando matrizes

As matrizes podem ser carregadas “manualmente”, através da inserção de valores para cada elemento representado pelos seus índices. Por exemplo, na linguagem C:

```
int m[3][3] = { 10, 20, 30, 40, 50, 60, 70, 80, 90 };
```

Nesse carregamento, temos:

```
m[0][0] = 10;
```

```
m[0][1] = 20;
```

```
m[0][2] = 30;
```

m[1][0] = 40;

m[1][1] = 50;

m[1][2] = 60;

m[2][0] = 70;

m[2][1] = 80;

m[2][2] = 90;

Outra forma que traz uma visualização melhor para este carregamento, obtendo o mesmo resultado para os elementos com base nos índices:

int m[3][3] = { {10, 20, 30}, {40, 50, 60}, {70, 80, 90} };

O carregamento de matrizes pode ser facilitada através do uso de laços de repetição. Uma das estruturas de repetição é a PARA-ATÉ-FAÇA.

Exemplo:

para i de 1 até 4 faça

para j de 1 até 4 faça

leia (matriz[i,j]);

fimpara;

fimpara;

Este trecho faz a leitura de dados para uma matriz de 4 linhas e 4 colunas, preenchendo, assim, 16 elementos. A seguir, apresentaremos o mesmo código na linguagem de programação C:

for (i=0; i<3; i++)

for (j=0; j<3; j++)

```
{
        scanf("%d", &matriz[i][j]);
}
```

Esse código utiliza a referência da matriz de 4x4 elementos criada.

Lendo e mostrando matrizes

O processo de leitura e apresentação de uma matriz também pode fazer uso de estruturas de repetição, utilizando também os comandos de saída.

Exemplo:

```
para i de 1 até 4 faça
para j de 1 até 4 faça
escreva (matriz[i,j]);
fimpara;
fimpara;
```

Utilizando o mesmo código na linguagem de programação em C:

```
for (i=0; i<3; i++)
    for (j=0; j<3; j++)
    {
        printf("%d", matriz[i][j]);
    }
```

A seguir, temos um exemplo de código utilizando matrizes na linguagem de programação em C:

```c
include<stdio.h>
#include<conio.h>
int main (void )
{
  int matriz[3][3],i, j;

  printf ("\nDigite valor para os elementos da matriz\n\n");

  for ( i=0; i<3; i++ )
    for ( j=0; j<3; j++ )
    {
        printf ("\nElemento[%d][%d] = ", i, j);
        scanf ("%d", &matriz[ i ][ j ]);
    }

  printf("\n\n******************** Saida de Dados ********************* \n\n");

  for ( i=0; i<3; i++ )
     for ( j=0; j<3; j++ )
    {
        printf ("\nElemento[%d][%d] = %d\n", i, j,matriz[ i ][ j ]);
    }

getch();
return(0);
}
```

Fonte: Casavella (2013a).

Nesse exemplo, temos o preenchimento de uma matriz de quatro linhas e quatro colunas, por meio de “scanf” dentro de uma estrutura “for” e, na sequência, a apresentação dos elementos dessa matriz utilizando “printf”.

Atividade

As matrizes apresentam uma estrutura de dados com variáveis compostas homogêneas, com mais de uma dimensão, diferenciando-as dos vetores. Analise o trecho de código a seguir:

```
int main(){

int i, j, m[2][3] = {{10, 20, 30},{40, 50, 60}};

        for(i=0;i<2;i++)

                for(j=0;j<3;j++)

                        m[i][j]=m[i][j]*5;

        for(i=0;i<2;i++){

                for(j=0;j<3;j++)

                        printf("%d ",m[i][j]);

                        printf("\n");

        }

        return 0;

}
```

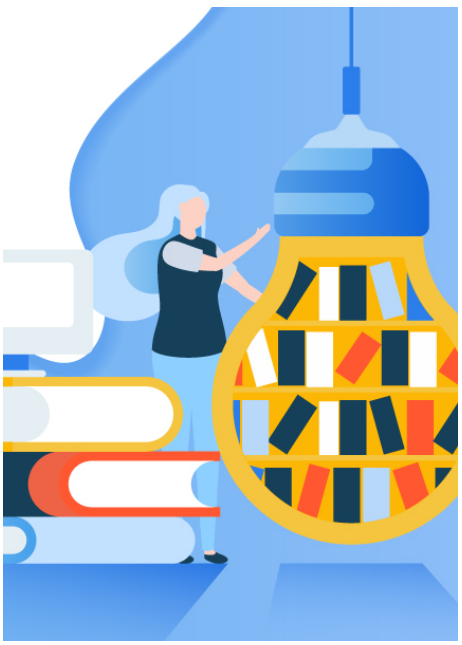
Observando esse código, assinale a alternativa que indique o valor impresso pela instrução printf, quando i = 1 e j = 2.

- ○ **a)** 300
- ○ **b)** 250
- ○ **c)** 200

- ○ **d)** 150
- ○ **e)** 100

indicações

Material Complementar



LIVRO

Algoritmos estruturados

Harry Farrer, Christiano Gonçalves Becker, Eduardo Chaves Faria, Helton Fábio de Matos, Marcos Augusto dos Santos, Miriam Lourenço Maia.

Editora: LTC

ISBN: 978-85-216-1180-6

Comentário: O livro indicado apresenta uma série de exemplos de algoritmos, em especial o capítulo “Estrutura de dados”, em que podem ser complementados os tópicos de estrutura de dados com variáveis compostas unidimensionais (vetores) e multidimensionais (matrizes).

WEB

{PortugolStudio} #11 - Vetores

Ano: 2017

Comentário: O vídeo indicado apresenta a construção de algoritmos com o uso de vetores, por meio da linguagem Portugol, de forma simples e fácil de aplicar.

ASSISTIR

conclusão

Conclusão

Nesta unidade, foram contempladas as estruturas de variáveis compostas homogêneas, unidimensionais (vetores) e multidimensionais (matrizes). Essas variáveis podem ser utilizadas quando um conjunto de dados do mesmo tipo deve ser utilizado.

Destacamos que tanto vetores como matrizes apresentam a característica da indexação, trabalhando com índices para o carregamento, leitura e apresentação dos valores de seus elementos. O número de elementos depende da combinação da quantidade de índices utilizados, como linhas e colunas.

Por fim, vimos que os vetores também são conhecidos como *arrays* em linguagens de programação. E que, para o trabalho de carregamento e leitura de vetores e matrizes, é comum o uso de estruturas de repetição, como “for”.

referências

Referências Bibliográficas

AVILA, W. M. M. Algoritmo: estrutura de vetores e matrizes. **Fábrica de Software**, 18 jun. 2013. Disponível em:

<http://fabrica.ms.senac.br/2013/06/algoritmo-estrutura-de-vetores-e-matrizes/>. Acesso em: 20 abr. 2019.

CASAVELLA, E. Matriz em C. **Intellectuale. Tecnologia & Treinamento**, 2013a. Disponível em: <http://linguagemc.com.br/matriz-em-c/>. Acesso em: 20 abr. 2019.

CASAVELLA, E. Passando um vetor para função em C. **Intellectuale. Tecnologia & Treinamento**, 2013b. Disponível em: <http://linguagemc.com.br/passando-um-vetor-para-funcao-em-c/>. Acesso em: 19 abr. 2019.

ESTRUTURA de dados em Python. **Os Programadores**, 14 abr. 2017. Disponível em: <https://osprogramadores.com/blog/2017/04/14/estruturas-python/>. Acesso em: 20 abr. 2019.

FORBELLONE, A. L. V.; EBERSPÄCHER, H. F. **Lógica de programação** – A construção de algoritmos e estruturas de dados. São Paulo: Prentice Hall, 2005.

MEDINA, M.; FERTIG, C. **Algoritmos e programação**: teoria e prática. São Paulo: Novatec, 2006.

SOUZA, M. A. F.; GOMES, M. M.; SOARES, M. V.; CONCILIO, R. **Algoritmos e lógica de programação**. São Paulo: Cengage Learning, 2011.

VILARIM, G. **Algoritmos** – Programação para iniciantes. Rio de Janeiro: Ciência Moderna, 2004.

IMPRIMIR