05 - Agregados Homogêneos: Vetores

Notas de Aula

Prof. Francisco Rapchan
www.geocities.com/chicorapchan
rapchan@terra.com.br

Em programação, assim como na álgebra, uma matriz é composta de um conjunto de elementos organizados em linhas e colunas. Neste contexto chamamos de **vetor** uma matriz de apenas uma linha ou apenas uma coluna. Dito de outra forma, um vetor é uma matriz unidimensional (que só tem uma dimensão, ou seja, uma linha ou uma coluna).

Neste capítulo trataremos primeiro de vetores e depois de matrizes com mais dimensões.

VETORES

Como já vimos, declarar uma variável é uma forma de dar um nome para uma posição de memória. Muitas vezes, porém, esta forma de definição de alocação de memória não é suficiente para resolver certos problemas.

Imagine, por exemplo, como construir um programa para ler o nome de 1000 pessoas e mostrar estes mesmos nomes em ordem alfabética. Não seria uma tarefa fácil de realizar usando apenas variáveis simples. Teríamos que definir 1.000 variáveis, uma para cada nome de pessoa. Considere o tamanho do programa e o trabalho braçal necessário para construí-lo. Imagine agora estender o problema para 1.000.000 de nomes. A construção deste algoritmo seria inviável na prática.

Para resolver problemas como este usamos um novo conceito para alocação de memória, o vetor.

Um vetor é uma estrutura de dados composta e homogênea. É composta porque armazena vários elementos (valores) e é homogênea porque todos os elementos devem ser do mesmo tipo (vetor de inteiros, ou vetor de reais, ou vetor de caracteres, etc.).

Exemplo 1. Armazene os números {13,21,9,41,7} em um vetor e depois mostre-os na tela.

```
#include <stdio.h>

int main ()
{
   int v[10];
   int cont;

   v[0] = 13;
   v[1] = 21;
   v[2] = 9;
   v[3] = 41;
   v[4] = 7;

   for (cont=0; cont<=4; cont++)
   {
      printf("%i\n",v[cont]);
   }

   return (0);
}
```

A instrução **`int v[5]`** cria 5 espaços consecutivos de memória numerados de 0 a 4. Cada espaço é chamado de **célula** do vetor.

As atribuições estão colocando valores nas células.

v[0]	v[1]	v[2]	v[3]	V[4]
13	21	9	41	7

No laço for, o valor de cont vai de 0 até 4. Assim, a cada **iteração** (ou ciclo do laço) é mostrado o valor de uma célula do vetor v. Na primeira iteração é mostrado 13, na segunda 21 e assim por diante.

Observe também o uso da forma abreviada de incremento `cont++` que equivale a `cont = cont + 1`

Exemplo 2. Faça um programa que armazene na memória 10 números e depois mostre todos os números armazenados em ordem inversa da que foram digitados.

```
#include <stdio.h>

int main ()
{
   int v[10];
   int cont;

   for (cont=0; cont<=9; cont++)
   {
      scanf("%i",&v[cont]);
   }

   for (cont=9; cont>=0; cont--)
   {
      printf("%i\n",v[cont]);
   }

   return (0);
}
```

A instrução **`int v[10]`** cria um vetor de 10 células numeradas de 0 a 9.

No primeiro laço for, o valor de cont vai de 0 até 9 e, a cada iteração, é solicitado que o usuário digite um valor.

Suponha que o usuário digite os seguintes valores:

{6, 9, 10, 8, 5, 4, 19, 8, 21, 8}

Os valores seriam colocados um a um nas 10 células:

v[0]	v[1]	...	v[8]	V[9]
6	9	...	21	8

No segundo laço for, o valor de cont vai de 9 até 0. Assim, a cada **iteração** é mostrado o valor de uma célula do vetor v começando pela última. Na primeira iteração é mostrado 8, na segunda 21 e assim por diante.

Exemplo 3. Faça um programa que armazene 10 valores de salário em um vetor. Depois, leia um valor de salário e indique quantas vezes ele aparece no vetor.

```
#include <stdio.h>

int main ()
{  //Cria as variáveis locais
   float busca, salarios[10];
   int cont, qtde;

   // Lê os salários e coloca no vetor
   for (cont=0; cont<=9; cont++)
      scanf("%f",&salarios[cont]);

   // Lê o salário que será procurado
   scanf("%f",&busca);

   // Varre o vetor procurando o salário
   qtde = 0; // Inicia a variável qtde
   for (cont=0; cont<=9; cont++)
      if (salarios[cont] == busca)
         qtde++;

   // Mostra o resultado
   printf("Quantidade:%i\n",qtde);

   return (0);
}
```

No vetor `salarios[10]` serão armazenados os 10 salários que o usuário irá digitar.

Após essa digitação o usuário irá digitar o valor do salário que ele quer procurar. Esse valor será armazenado na variável `busca`.

A variável qtde irá armazenar quantos salários iguais ao procurado pelo usuário temos armazenado no vetor `salarios`.

Observe que nesse programa as chaves nas estruturas `for` e `if` foram economizadas.

Em C não é necessário **usar chaves** quando temos apenas uma linha lógica dentro da estrutura. As linhas lógicas são identificadas não pela sua posição mas pelo ponto-e-vírgula (;). Assim, na estrutura abaixo temos apenas uma linha lógica:

```
for (cont=0; cont<=9; cont++)
      if (salarios[cont] == busca)
         qtde++;
```

Observe também o **uso de comentários** usados para facilitar o entendimento do código. Essa prática é muito importante para a **manutenibilidade** do programa que está sendo desenvolvido.

Exemplo 4. Faça um programa que leia 5 números inteiros e armazene em um vetor chamado numeros. Depois, percorra o vetor numeros procurando por números pares. Os números pares encontrados deverão ser armazenados em um vetor chamado pares.

```
#include <stdio.h>

int main ()
{
   int i, j, pares[5], numeros[5];

   // Lê os números
   for (i=0; i<=4; i++)
      scanf("%i",&numeros[i]);

   // Identifica os primos
   j = 0;
   for (i=0; i<=4; i++)
      if (numeros[i] % 2 == 0)
      {  pares[j] = numeros[i];
         j++;
      }

   // Mostra o vetor de primos
   printf ("Sao primos:\n");
   for (i=0; i<=j-1; i++)
      printf("%i\n",pares[i]);

   return (0);
}
```

Em qualquer programa real é muito importante comentar os trechos do código identificando suas funcionalidades. Esta prática facilita muito a identificação de erros (nos testes) e de defeitos (na manutenção).

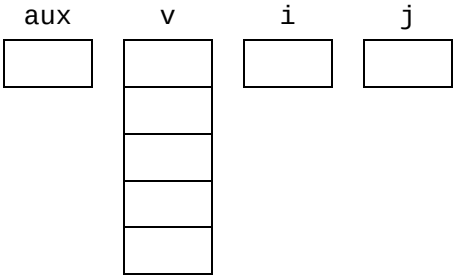
O operador %, retorna o resto da divisão inteira. Assim, 5 % 3 retorna 2 e 6 % 3 retorna 0. Se um número N é par, então N % 2 retornará zero!

Exemplo 5. O que faz o programa abaixo?

```
#include <stdio.h>

int main ()
{
   float aux, v[5];
   int i,j;

   for (i=0; i<=4; i++)
      scanf("%f",&v[i]);

   for (i=0; i<=4; i++)
      for (j=0; j<=3; j++)
         if (v[j] > v[j+1])
         {
            aux = v[j];
            v[j] = v[j+1];
            v[j+1] = aux;
         }

   for (i=0; i<=4; i++)
      printf("%f\n",v[i]);

   return (0);
}
```

Para saber o que esse programa faz podemos usar a técnica do "**Teste de Mesa**". A idéia é fazer manualmente, em uma folha de papel, todos os passos que o computador faria ao executar este programa.

Para isso, crie espaços as variáveis aux, v[5], i, j. Depois execute as instruções alterando o valor dessas variáveis.



Observe que v é um vetor de 5 posições!

Exemplo 6. Leia 5 números e coloque em um vetor. Depois leia um número e, se este número estiver no vetor, retire-o, reorganizando os elementos para ocupar o espaço dele. Ao final, mostre os elementos do vetor resultante.

Exemplo: Suponha um vetor com os seguintes elementos: {1, 2, 3, 2, 4}. Se o usuário digitar 1, deveremos ter no vetor: { 2, 3, 2, 4} e se o usuário digitar 2, deveremos ter: {1, 3, 4}.

```
#include <stdio.h>

int main ()
{  int v[5];    // Vetor de números
   int c, i;    // Indices para percorer o vetor
   int valor;   // Número que será retirado do vetor
   int t = 5;   // Tamanho do vetor

   // Lê os valores
   for (c=0; c < t; c++){
      printf ("Valor: "); scanf ("%i",&v[c]);
   }

   // Identifica o valor a ser retirado do vetor
   printf ("Valor para retirar: "); scanf ("%i",&valor);

   // Varre o vetor retirando valor
   c = 0;
   while (c < t){
      if (v[c] != valor)
         c++;
      else {
         for (i=c ; i < t-1 ; i++)
            v[i]=v[i+1];
         t--;
      }
   }

   // Mostra o vetor resultante
   for (c=0; c < t; c++)
      printf ("%i\n",v[c]);

   return (0);
}
```