18 Vetores

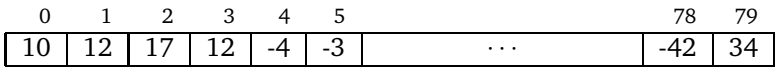
Ronaldo F. Hashimoto e Carlos H. Morimoto

Nessa aula vamos introduzir o tipo **vetor**. Ao final dessa aula você deverá saber:

- Descrever o que são *vetores* na linguagem C.
- Declarar vetores.
- Como acessar elementos de um vetor e percorrer um vetor.
- Utilizar vetores para resolver problemas computacionais.

18.1 Vetores

Vetores são estruturas indexadas utilizadas para armazenar dados de um mesmo tipo: **int**, **char**, **float** ou **double**. O exemplo a seguir é de um vetor de inteiros:



18.1.1 Declaração de Vetores

A declaração de um vetor é feita da seguinte forma:

<tipo_do_vetor> <nome_do_vetor> [<tamanho_do_vetor>];

Exemplos:

- **int** v[80]; ——— 80 é o tamanho do vetor!

  A declaração acima reserva 80 gavetas consecutivas na memória, que corresponde ao tamanho ou número de casas do vetor. Cada gaveta guarda um **int**.

- **float** x[20]; ——— 20 é o tamanho do vetor!

  A declaração acima reserva 20 gavetas consecutivas na memória. Cada gaveta guarda um **float**.

Observação Importante:

1. Na **declaração de vetor**, o que está entre colchetes deve ser um **número constante**.
2. Assim, não é possível fazer algo deste tipo:

```
int n = 20;
float x[n]; /* não é permitido declarar colocando uma variável */
```

ou

```
int n;

printf ("Entre com n>0: ");
scanf ("%d", &n);

float x[n];
```

O correto seria:

```
int n;
float x[20]; /* o correto é declarar sempre um tamanho fixo */
```

18.1.2 Uso de Vetores

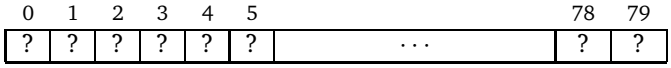
- São usados índices para acessar uma casa de um vetor.
- Um índice é um número natural.
- O índice da **primeira** casa é sempre **zero**.

18.1.3 Exemplo de Uso de Vetores

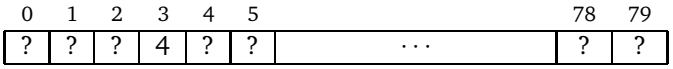
- Exemplo 1:

```
1   # include <stdio.h>
2
3   int main () {
4     int v[80], i;
5
6     v[3] = 4; /* casa de índice 3 do vetor v recebe o inteiro 4 */
7     i = 2;
8     v[i] = 3; /* casa de índice 2 do vetor v recebe o inteiro 3 */
9     v[v[v[i]]] = 10; /* vc saberia dizer qual casa do vetor v
10                        * recebe o inteiro 10?
11                        */
12
13    return 0;
14  }
```

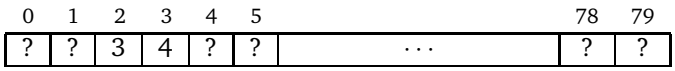
Na Linha 4, o vetor `v` com 80 casas é declarado:



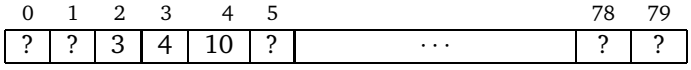
Na Linha 6, casa de índice 3 do vetor v recebe o inteiro 4:



Na Linha 8, casa de índice 2 do vetor v recebe o inteiro 3:



Na Linha 9, temos: `i=2`, `v[i]=3` e `v[v[i]]=v[3]=4`. Desta forma, no comando da Linha 9, a casa de índice 4 do vetor v recebe o inteiro 10:



- Exemplo 2:

```c
# include <stdio.h>

int main () {
  float x[80];
  int i;

  for (i=0; i<80; i++)
    x[i] = 0;

  return 0;
}
```

O programa acima coloca o valor zero em cada uma das casas do vetor `x`.

- Exemplo 3:

  O índice do vetor pode ser uma expressão aritmética, como mostrado a seguir:

```c
# include <stdio.h>

int main () {
  float x[80];
  int i;

  for (i=110; i<190; i++)
    x[i-110] = 0;

  return 0;
}
```

  mas tenha absoluta certeza, porém, de sempre fornecer um índice válido de forma que o resultado da expressão aritmética seja válida (neste exemplo, o resultado da expressão aritmética deve ser um inteiro entre 0 e 79).

18.2 Percorrimento de Vetores

Percorrer um vetor significa varrer o vetor de casa em casa a partir do índice 0 (zero). No percorrimento de um vetor, é necessário saber o número de casas que deve-se fazer este percorrimento. Este número normalmente é guardado em uma variável inteira.

Muitos problemas computacionais que envolvem vetores têm como solução o uso de um padrão para percorrimento de vetores.

Um padrão para percorrer "`n_casas`" de um vetor "`vetor`" é usar um comando de repetição (no caso, vamos usar o comando **for**) com uma variável inteira "`indice`" para o índice das casas do vetor:

```c
for (indice=0; indice < n_casas; indice++) {
  <algum comando usando vetor[indice]>
}
```

18.2.1 Leitura de um Vetor

Para leitura de vetor, devemos ler elemento a elemento usando o padrão de percorrimento.

```c
#include <stdio.h>

int main () {
  float v[100];
  int i, n;

  printf ("Entre com 0<n<=100: ");
  scanf ("%d" &n);

  /* percorrer o vetor v de 0 a n-1 colocando o valor lido pelo teclado */
  for (i=0; i<n; i++) {
    printf ("Entre com v[%d] = ", i);
    scanf ("%f", &v[i]);
  }

  return 0;
}
```

Observe com cuidado a linha do programa utilizada para ler o vetor:

```c
scanf ("%f", &v[i]);
```

A posição `i` do vetor `v`, ou seja, `v[i]`, é utilizada da mesma forma que utizamos qualquer variável até o momento. Essa "variável" é passada para a funcao `scanf` precedida pelo caractere '&'.

18.2.2 Impressão de um Vetor

Para impressão de vetor, devemos imprimir elemento a elemento usando o padrão de percorrimento.

```c
#include <stdio.h>

int main () {
  float v[100];
  int i, n;

  printf ("Entre com 0<n<=100: ");
  scanf ("%d" &n);

  /* percorrer o vetor v de 0 a n-1 imprimindo o valor de cada casa */
  for (i=0; i<n; i++) {
    printf ("v[%d] = %f\n", i, v[i]);
  }

  return 0;
}
```

18.2.3 Observação sobre Percorrimento

Na declaração de um vetor é definido um número fixo de casas, uma vez que sempre deve-se colocar uma constante na definição do número de casas do vetor. Por exemplo:

```c
int v[30];
```

Mas, como podemos ver nos exemplos de percorrimento para ler e imprimir um vetor, um usuário não necessariamente irá usar todas as casas disponíveis do vetor. Note que no padrão de percorrimento deve sempre

existir uma variável indicando quantas casas do vetor estão sendo verdadeiramente usadas (variável `n_casas` do padrão).

Assim, normalmente, em problemas computacionais que envolvem vetores deve-se sempre ter uma variável inteira associada a este vetor que diz quantas casas do vetor estão sendo usadas (por exemplo, variável inteira `n` associada ao vetor `v` nos exemplos de leitura e impressão de vetores).

18.3 Exercícios Comentados

18.3.1 Exercício 1

Dada uma sequência de `0<n<100` números inteiros, imprimi-la na ordem inversa à da leitura.

Exemplo:

Para `n=5`, e a sequência $11, 12, 3, 41, 321$, o programa deve imprimir a saída $321, 41, 3, 12, 11$.

Para resolver esse problema, precisamos armazenar todos os elementos da sequência em um vetor (usando padrão de percorrimento), e depois imprimir esses elementos em ordem inversa (usando padrão de percorrimento em ordem inversa). Observe que sem usar vetor, ou seja, usando apenas variáveis, seria muito difícil resolver esse problema para um valor arbitrário de `n`. Um programa possível, usando vetores, seria:

```
1       # include <stdio.h>
2
3       # define MAX 100
4
5       int main () {
6         int v[MAX], n, i;
7
8         printf ("Entre com 0<n<100: ");
9         scanf ("%d", &n);
10
11        /* percorrer o vetor v do índice 0 a n-1 colocando o valor lido pelo teclado */
12        for (i=0; i<n; i++) {
13          printf ("Entre com v[%d] = ", i);
14          scanf ("%d", &v[i]);
15        }
16
17        /* percorrer o vetor v do índice n-1 a 0 imprimindo o valor de cada casa */
18        for (i=n-1; i>=0; i--) {
19          printf ("v[%d] = %d\n", i, v[i]);
20        }
21
22        return 0;
23      }
```

Note que neste exercício definimos uma constante `MAX` usando o "comando" `define`. Observe que `MAX` é uma constante e não uma variável. Mais sobre definição de constantes, veja o material didático **Alguns Detalhes da Linguagem C**.

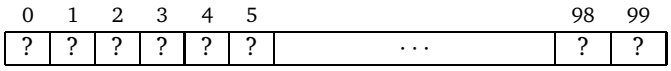
Observe então que o tamanho do vetor é fixo, e deve ser definido antes do programa ser executado. É um erro muito comum entre programadores inexperientes ler um número e utilizá-lo para definir o tamanho do vetor. Algo do tipo:

```
int n;

printf ("Entre com n>0: ");
scanf ("%d", &n);

int v[n];
```

Isto está completamente **ERRADO**! Tome cuidado para você não cometer este **ERRO**!

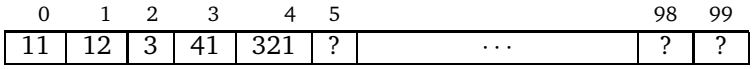
Na solução do problema, no programa, o valor limite foi definido como `MAX` e possui valor 100. Esse programa não pode ser executado para sequências maiores que 100.

Na Linha 6, o vetor `v` com 100 casas é declarado:



A leitura do inteiro `n` (digamos que leia 5, como no exemplo dado no exercício) que indica quantas das 100 casas do vetor `v` que serão de fato utilizadas é feita na Linha 9.

Após a leitura do vetor (Linhas 11 a 15), a estrutura correspondente na memória pode ser representada como:



Note que as casas de índice maior que 4 contêm lixo. Mas, tudo bem, uma vez sabemos o número de casas (guardado na variável `n`=5) que contêm os valores da sequência.

Agora observe a parte do programa para imprimir o vetor invertido (Linhas 17 a 20). Como o vetor foi lido como mostra o esquema acima, devemos percorrer o vetor `v` desde a última posição lida e decrementar até a primeira. A última posição lida nesse caso foi a posição 4 (quatro), e a primeira foi 0 (zero).

18.3.2 Exercício 2

Dados `n`>0 lançamentos de uma roleta (números entre 0 e 36), calcular a frequência de cada número.

Como a gente calcula a frequência de um número? Basta contar quantas vezes ele aparece na sequência. Precisamos, portanto, criar um contador para cada número da sequência possível, ou seja, 37 contadores. Cada contador começa com zero, e toda vez que lancamos a roleta, a gente incrementa apenas o contador correspondente. Ao final, basta calcular as frequências. A solução completa e dada a seguir:

```c
#include <stdio.h>

int main () {
  int freq[37];
  int n, i, x;

  /* zerando o vetor freq */
  for (i=0; i<37; i++)
  freq[i] = 0;

  printf ("Entre com o número de lançamentos n>0: ");
  scanf ("%d", &n);

  for (i=0; i<n; i++) {
    printf ("Entre com um lançamento: ");
    scanf ("%d", &x);
    freq[x] = freq[x] + 1;
  }

  for (i=0; i<37; i++)
    if (freq[i] > 0)
      printf ("O número %d apareceu %d veze(s)\n", i, freq[i]);

  return 0;
}
```

18.3.3 Exercício 3

Um exemplo de como vetores pode ser útil. Um vetor poderia guardar os coeficientes de um polinômio de grau n. Por exemplo:

```c
float a[101];
int n;
```

Dessa forma, $a[0], a[1], \ldots, a[n]$ guardam os coeficientes de um polinômio $p(x) = a[0] + a[1]x + ... + a[n]x^n$. É claro que, neste particular caso, n não pode ser maior que 100.

18.4 Erros Comuns

Ao desenvolver seus programas com vetores, preste atenção com relação aos seguintes detalhes:

- **índices inválidos:** tome muito cuidado, especialmente dentro de um **while** ou **for**, de não utilizar índices negativos ou maiores que o tamanho máximo do vetor.
- **Definição do tamanho do vetor** se faz na declaração do vetor. O tamanho dele é constante, só mudando a sua declaração é que podemos alterar o seu tamanho. Isso significa que podemos estar "desperdiçando" algum espaço da memória que fica no final do vetor. Não cometa o erro de ler `n`, onde `n` seria o tamanho do vetor, e tentar "declarar" o vetor em seguida.

18.5 Exercícios Recomendados

1. Dados dois polinômios reais $p(x) = a_0 + a_1 \cdot x + ... + a_n \cdot x^n$ ($n < 20$) e $q(x) = b_0 + b_1 \cdot x + ... + b_m \cdot x^m$ ($m < 40$) determinar o produto desses polinômios.
2. Dados dois vetores x e y, ambos com n elementos, $n < 50$, determinar o produto escalar desses vetores.

3. Em uma classe há n alunos, $n < 100$, cada um dos quais realizou k provas, $k < 4$, com pesos distintos. Dados n , k, os pesos das k provas e as notas de cada aluno, calcular a média ponderada das provas para cada aluno e a média aritmética da classe em cada uma das provas.

4. Dada uma sequência de 0<n<200 números inteiros, determinar um segmento de soma máxima.

   Exemplo: para n=12 e a sequência

0	1	2	3	4	5	6	7	8	9	10	11	12	...	198	199
5	2	-2	-7	3	14	10	-3	9	-6	4	1	?	...	?	?

   um segmento de soma máxima está entre os índices 4 e 8 e a correspondente soma é 33.

5. Dada uma sequência $x_0, x_1, ..., x_{k-1}$ de números inteiros, com $k < 300$, verifique se existem dois segmentos consecutivos iguais nesta sequência, isto é, se existem i e m tais que:

$$x_i, x_{i+1}, ..., x_{i+m-1} = x_{i+m}, x_{i+m+1}, ..., x_{i+2m-1}$$

   Imprima, caso existam, os valores de i e m.

   Exemplo: para $k = 8$ e a sequência

0	1	2	3	4	5	6	7	8	...	298	299
7	9	5	4	5	4	8	6	?	...	?	?

   existem i=2 e m=2.

6. Dada uma sequência de $n > 0$ números reais, imprimi-los eliminando as repetições.