O conteúdo de uma variável ponteiro pode ser passado para outra variável ponteiro (do mesmo tipo), como acontece numa variável comum. Assim, uma variável ponteiro declarada como apontador de dados inteiros deve sempre apontar para dados deste tipo.

O exemplo abaixo mostra:

- a declaração de 2 variáveis de ponteiros para inteiros.
- o uso dos ponteiros *pa* e *pb* para armazenar o endereço de dois dos elementos do vetor.
- a cópia do conteúdo de uma variável ponteiro em outra variável ponteiro (para atualizar um valor no vetor).

```
int main() {
  int lista[10]={1,2,3,4,5,6,7,8,9,10};
  int *pa, *pb;

  pa = &lista[2];
  pb = &lista[3];

  printf("\nValor apontado por pa: %d  (endereco %p)", *pa, &pa);
  printf("\nValor apontado por pb: %d  (endereco %p)", *pb, &pb);

  // Copia do conteúdo de pb em pa
  *pa = *pb;
  printf("\nNovo valor apontado por pa: %d \n", *pa);
}
```

Exercício: Teste o programa acima e codifique a impressão dos elementos do vetor antes e após a cópia do conteúdo apontado por *pb* em *pa*.

Aritmética de Ponteiros

As operações de adição e subtração podem ser aplicadas à variáveis de ponteiro. Desta forma, buscam-se endereços de memória que estejam acima (adição) ou abaixo (subtração) do endereço atual contido no ponteiro.

O exemplo a seguir mostra o incremento de uma variável de ponteiro para que os elementos contidos num vetor possam ser exibidos:

```
int main(){

 int lista[10]={1,2,3,4,5,6,7,8,9,10};
 int *pa;
 int j;

 pa = lista;    // pa recebe o endereço do primeiro elemento do vetor
                // pode-se também escrever: pa = &lista[0];
 for (j=0; j<10; j++){
    printf("\nValor apontado por pa: %d", *pa);
    pa++;
 }
}
```

Exercícios:

1) Criar um vetor de inteiros e incluir valores aleatórios em suas posições. Usando aritmética de ponteiros para o percurso no vetor, faça:

   (a) Encontrar o menor elemento do vetor.

   (b) Calcular a média entre os elementos.

   (c) Encontrar o número de elementos negativos.

   (d) Imprimir os elementos em ordem invertida.

   (e) Usar a função *printf* para retornar o número de bytes alocados a cada posição no vetor. Como as posições no vetor são contínuas, a diferença entre dois endereços consecutivos permite descobrir o tamanho de cada posição.

2) Construa um vetor de números reais e percorra todo o vetor usando dois ponteiros : um começando do início do vetor e outro do final.