28. [Sebesta, 2000] Escreva um procedimento SOMADOR, na linguagem de programação de sua preferência, que faz a adição de dois *arrays* de números inteiros. Ele deve ter somente dois parâmetros, os quais têm dois *arrays* a serem adicionados. O segundo *array* também conterá o *array* da soma na saída. Ambos os parâmetros devem ser passados por referência. Teste SOMADOR com a chamada SOMADOR (A, A) em que A é um *array* a ser adicionado a si mesmo. Explique os resultados de executar esse programa.

```
program Teste;

type
  vector = array of integer;

var
  { Inicializando um array de 10 elementos
    Observação: o tamanho do array está na posição 0 }
  a: vector = (10, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10);
  i: integer;

{ Os compiladores Pascal são capazes de determinar o tamanho definido de todos
  os arrays usados como parâmetros no momento em que os programas são compilados.
  Os tipos de array não-restringidos podem ser parâmetros formais. Um tipo de
  array não-restringido é aquele em que as faixas de índices não são fornecidas
  na definição do tipo de array. As definições de variáveis de tipos de array
  não-restringido devem incluir faixas de índices. O código em um subprograma
  passado como um array não-restringido pode obter informações sobre a faixa
  de índices do parâmetro real associado com esses parâmetros formais. }

procedure somador (var x, y : vector);
  var
    i, length : integer;

  begin
    if (x[0] < y[0]) { Identificar o tamanho do menor array }
      then length := x[0]
      else length := y[0];

    for i := 1 to length do
      y[i] := x[i] + y[i];
  end;

begin
  somador(a, a);

  for i := 1 to a[0] do
    writeln ('a[', i, '] = ', a[i]);
end.
```

```c
#include <stdio.h>

/* O problema com o método de passar arrays como parâmetros é que ele não permite
   que o programador escreva uma função que possa aceitar arrays de tamanhos
   diferentes. Uma nova função deve ser escrita para cada array com um tamanho
   diferente. Isso, de fato, impede a escrita de funções flexíveis efetivamente
   reusáveis se as funções lidarem com arrays de tamanhos distintos. */

void somador(int x[10], int y[10])
{
  for (int i = 0; i < 10; i++)
    y[i] = x[i] + y[i];
}

/* Uma maneira de contornar o problema é passar o array como um ponteiro, e o
   tamanho do array como parâmetro. Então, a função pode avaliar a função de
   associação de armazenamento do array utilizando a aritmética de ponteiros
   cada vez que um elemento do array precisar ser referenciado. */

void somador2(int *x, int *y, int length_x, int length_y)
{
  int length = length_x;

  if (length_x > length_y)
    length = length_y;

  for (int i = 0; i < length; i++)
    y[i] = x[i] + y[i];
}

int main()
{
  int a[10] = { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 };

  somador(a, a);

  for (int i = 0; i < 10; i++)
    printf("%d\n", a[i]);

  int b[10] = { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 };

  somador2(b, b, 10, 10);

  for (int i = 0; i < 10; i++)
    printf("%d\n", b[i]);

  return 0;
}
```

```
public class Teste
{
  /* O Java usa uma técnica para passar arrays como parâmetros similar à
     do Pascal. Em Java, arrays são objetos. Cada um deles herda uma constante
     nomeada (length) fixada no tamanho do array quando o objeto array é
     criado. */

  public static void somador(int x[], int y[])
  {
    int length = x.length;

    if (x.length > y.length)
      length = y.length;

    for (int i = 0; i < length; i++)
      y[i] = x[i] + y[i];
  }

  public static void main(String[] args)
  {
    int a[] = { 1, 2, 3, 4, 5, 6, 7, 8, 10 };

    somador(a, a);

    for (int i = 0; i < a.length; i++)
      System.out.println(a[i]);
  }
}
```