```
algoritmo "exercício vetor"
var vet:vetor [1..12] de inteiro
      i:inteiro
inicio
   para i de 1 ate 12 faca
      escreva ("Entre com vetor[",i,"]: ")
      leia (vet[i])
   fimpara
fimalgoritmo
```

```
algoritmo "exercício vetor"
var vet:vetor [0..11] de inteiro
      i:inteiro
inicio
   para i de 0 ate 11 faca
      escreva ("Entre com vetor[",i+1,"]: ")
      leia (vet[i])
   fimpara
fimalgoritmo
```

Estruturas de dados homogêneas

Exercício 27:

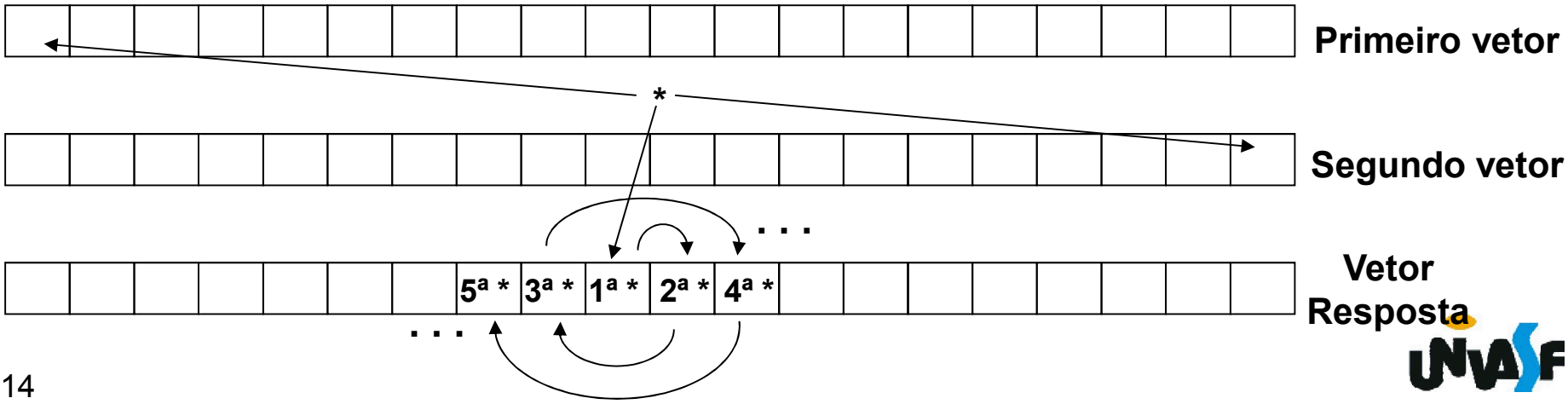
Elabore um algoritmo, com base no exercício anterior, que declare um vetor de inteiros com 12 elementos, o inicialize, com números fornecidos pelo usuário através da entrada padrão, e que, após a inicialização, através de uma pesquisa nos elementos do vetor, retorne na saída padrão os elementos de menor e maior valor, respectivamente.

```
algoritmo "exercício vetor"
var vet:vetor [1..12] de inteiro
    i,maior,menor:inteiro
inicio
   para i de 1 ate 12 faca
      escreva ("Entre com vetor[",i,"]: ")
      leia (vet[i])
   fimpara
   para i de 1 ate 12 faca
      se (i=1) entao
         menor<-vet[i]
         maior<-menor
      senao
         se (maior<vet[i]) entao
            maior<-vet[i]
         senao
            se (menor>vet[i]) entao
               menor<-vet[i]
            fimse
         fimse
      fimse
   fimpara
   escreva ("O menor valor contido no vetor é: ",menor)
   escreval ("O maior valor contido no vetor é: ",maior)
fimalgoritmo
```

Estruturas de dados homogêneas

Exercício 28:

Construa um algoritmo que aplique a operação definida a seguir sobre dois vetores de reais, cada um com vinte elementos inicializados pelo usuário. A operação consiste em multiplicar os elementos de um vetor pelos do outro da seguinte forma: cada elemento do primeiro vetor deve ser multiplicado pelo elemento com posição correspondente no vetor obtido considerando o inverso do segundo vetor. Cada valor resultante das multiplicações deve ser armazenado em um vetor resposta partindo do centro para as bordas.

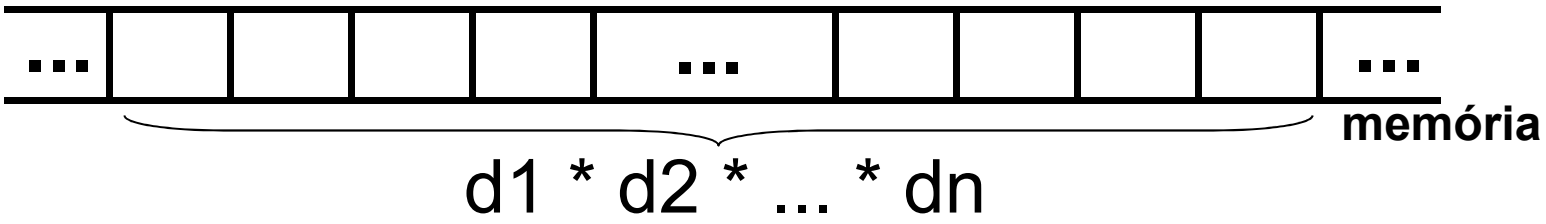


```
algoritmo "exercício vetor"
var vet1:vetor [1..20] de real
    vet2:vetor [1..20] de real
    vetres:vetor [1..20] de real
    i,k:inteiro
inicio
   para i de 1 ate 20 faca
      escreva ("Entre com vetor_1[",i,"]: ")
      leia (vet1[i])
      escreva ("Entre com vetor_2[",i,"]: ")
      leia (vet2[i])
   fimpara
   k<-10
   para i de 1 ate 20 faca
      vetres[k]<-vet1[i]*vet2[21-i]
      se (i%2<>0) entao
         k<-k+i
      senao
         k<-k-i
      fimse
   fimpara
fimalgoritmo
```

Estruturas de dados homogêneas

Vetores Multidimensionais

nome_do_vetor* : vetor [*menor_indice_d1*.. *maior_indice_d1*, *menor_indice_d2*.. *maior_indice_d2*, ..., *menor_indice_dn*.. *maior_indice_dn*] de *tipo_dos_elementos



***Obs.**: d1=maior_indice_d1-menor_indice_d1+1*

d2=maior_indice_d2-menor_indice_d2+1

...

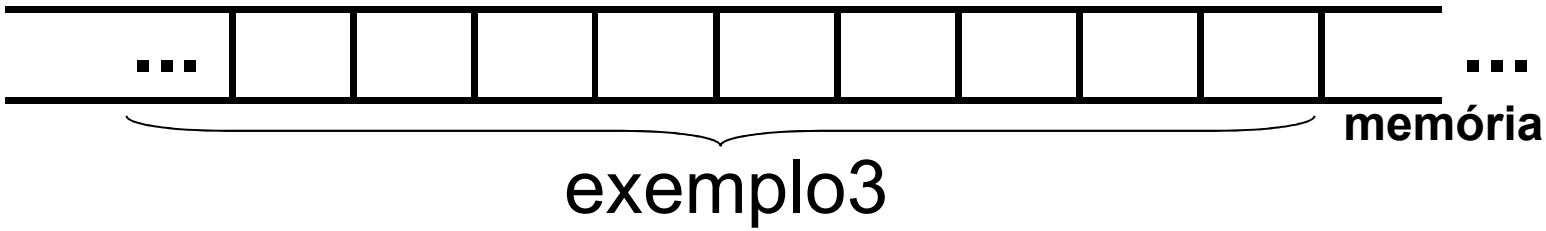
dn=maior_indice_dn-menor_indice_dn+1

Estruturas de dados homogêneas

Vetores Multidimensionais (continuação)

Exemplo:

exemplo3: vetor [0..2, 7..9] de real

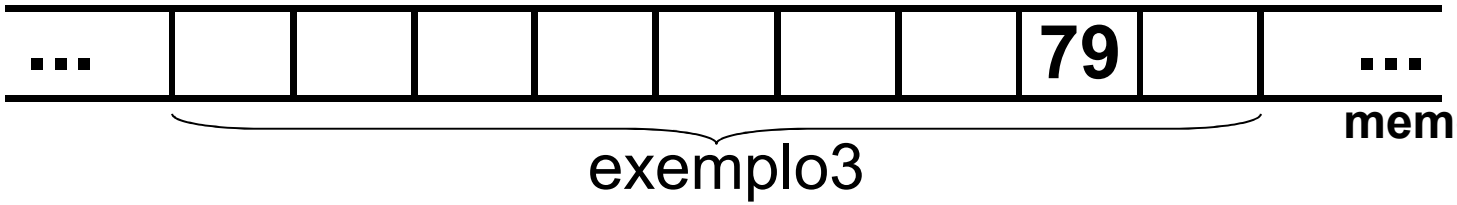


Estruturas de dados homogêneas

Exemplo:

exemplo3 [2,8]<-79

O armazenamento de vetores multidimensionais se dá da seguinte forma: na primeira posição armazena-se o elemento com os menores índices de cada dimensão, no caso do exemplo anterior o elemento referenciado por exemplo3[0,7], seu sucessor é o elemento com o índice mais à direita incrementado em uma unidade (exemplo3[0,8]); quando o referido índice chegar ao seu valor máximo (exemplo3[0,9]) é incrementado o índice que o antecede e o seu valor volta a ser o menor possível (exemplo3[1,7]) e assim sucessivamente. Sendo assim, temos



Estruturas de dados homogêneas

Exemplo:

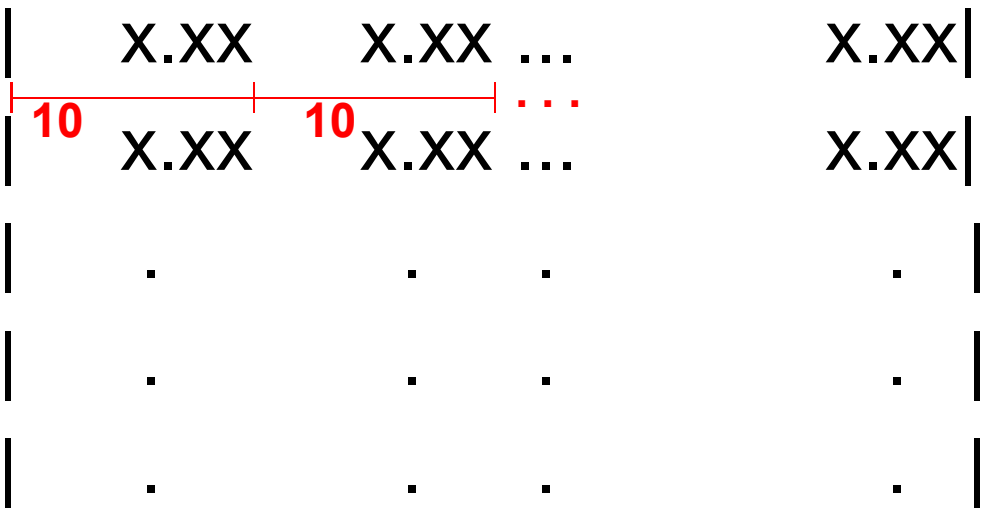
O algoritmo abaixo declara uma matriz 3x4 de inteiros e a inicializa com valores fornecidos pelo usuário.

```
Algoritmo "exemplo matriz"
var
   matriz: vetor [1..3,1..4] de inteiro
   i, j: inteiro
inicio
   para i de 1 ate 3 faca
      para j de 1 ate 4 faca
         escreva ("Entre com matriz[",i, ",",j,"]: ")
         leia (matriz[i,j])
      fimpara
   fimpara
fimalgoritmo
```

Estruturas de dados homogêneas

Vetores Multidimensionais (continuação)

Exercício 29: Construa um algoritmo que declare uma matriz 7x4 de números reais, a inicialize com valores fornecidos pelo usuário através da entrada padrão e a apresente na saída padrão com o layout a seguir:



```
algoritmo "exercício 1 matriz"
var
   matriz: vetor [1..7,1..4] de real
   i, j: inteiro
inicio
   para i de 1 ate 7 faca
      para j de 1 ate 4 faca
         escreva ("Entre com matriz[",i, ",",j,"]= ")
         leia (matriz[i,j])
      fimpara
   fimpara
   para i de 1 ate 7 faca
      escreva ("|")
      para j de 1 ate 4 faca
         escreva (matriz[i,j]:10:2)
      fimpara
      escreval ("|")
   fimpara
fimalgoritmo
```

Estruturas de dados homogêneas

Vetores Multidimensionais (continuação)

Exercício 29b: Adapte a solução gerada para o exercício anterior fazendo com que a mesma não se utilize de laços de repetição aninhados. Ou seja, construa um algoritmo que percorra todos os elementos da matriz utilizando para tal apenas um laço de repetição que itere 28 vezes.

UNIVASF