Exercícios – Aula Prática VII

Estruturas de dados homogêneas

As estruturas de dados homogêneas que estudaremos são os vetores também conhecidos com arranjos.

Vetores nada mais são que matrizes.

Matriz (Álgebra) -> Arranjo retangular de elementos de um conjunto.

É importante ressaltar que vetores de qualquer dimensão são caracterizados por terem todos os seus elementos pertencentes ao mesmo tipo de dado.

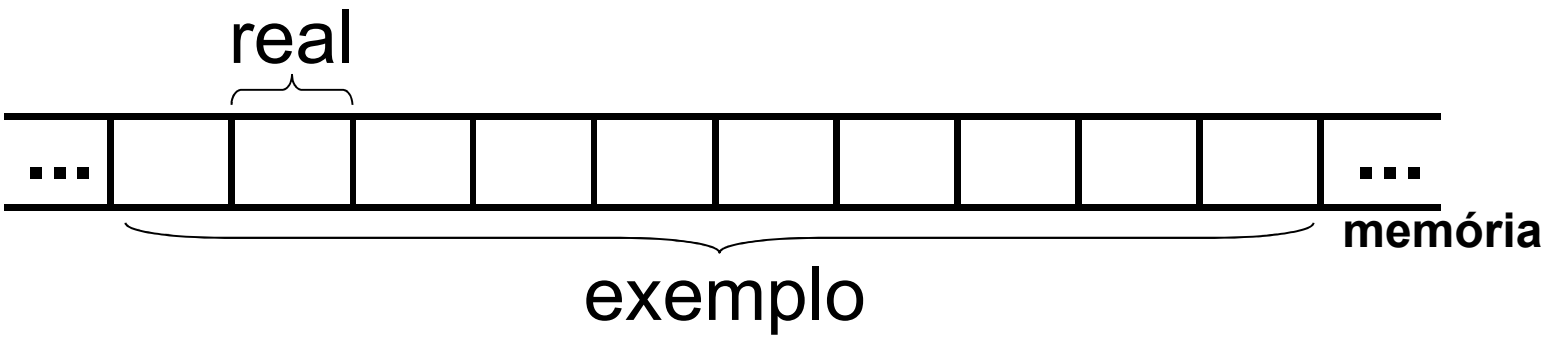
A Forma geral para se declarar um vetor unidimensional é:

nome_do_vetor : vetor [menor_indice.. maior_indice] de tipo_dos_elementos

Estruturas de dados homogêneas

Exemplo 1:

exemplo: vetor [1..10] de real



Estruturas de dados homogêneas

Em função de um vetor se tratar de um arranjo de elementos torna-se necessária uma forma de acessar individualmente cada elemento. A indexação possibilita tal acesso. A especificação do intervalo dos índices além de definir o número de elemento indica quais serão os valores dos índices utilizados para acessar cada elemento.

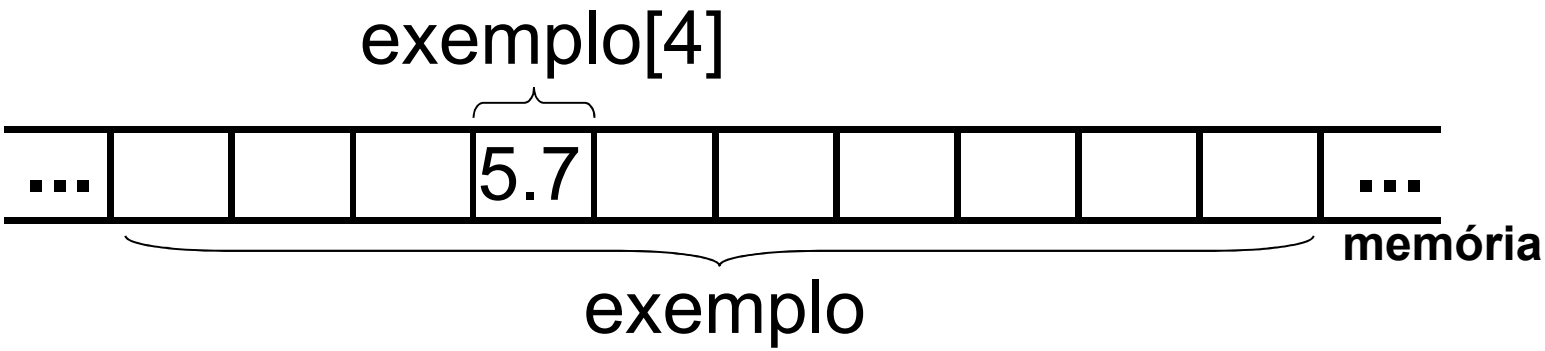
No exemplo anterior, os dados serão indexados de 1 a 10. Para acessá-los vamos escrever:

exemplo[**1**]
exemplo[**2**]
.
.
.
exemplo[**10**]

Estruturas de dados homogêneas

Exemplo:

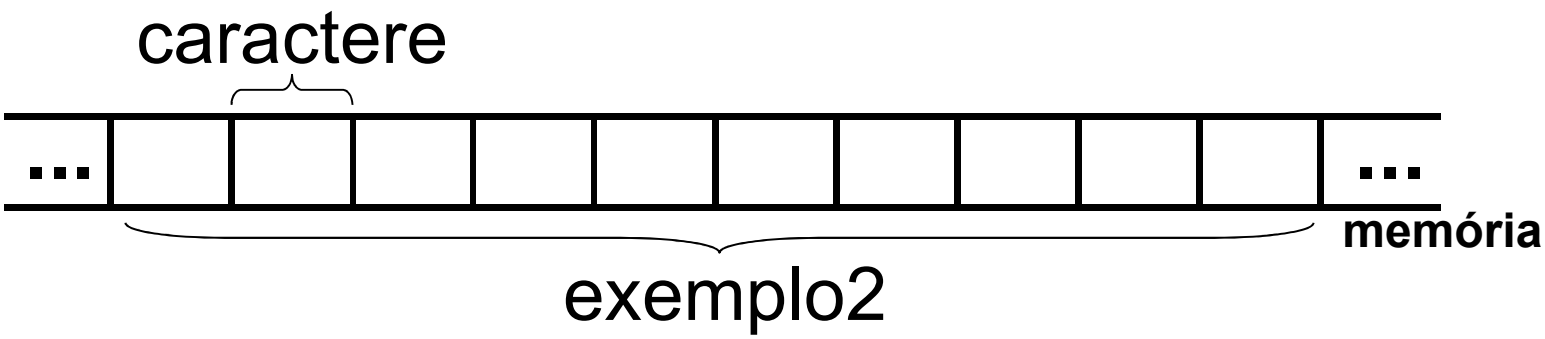
exemplo[4] <- 5.7



Estruturas de dados homogêneas

Exemplo 2:

exemplo2: vetor [4..13] de caractere



Estruturas de dados homogêneas

No último exemplo apresentado, os dados serão indexados de 4 a 13. Para acessá-los vamos escrever:

exemplo2[**4**]

exemplo2[**5**]

.

.

.

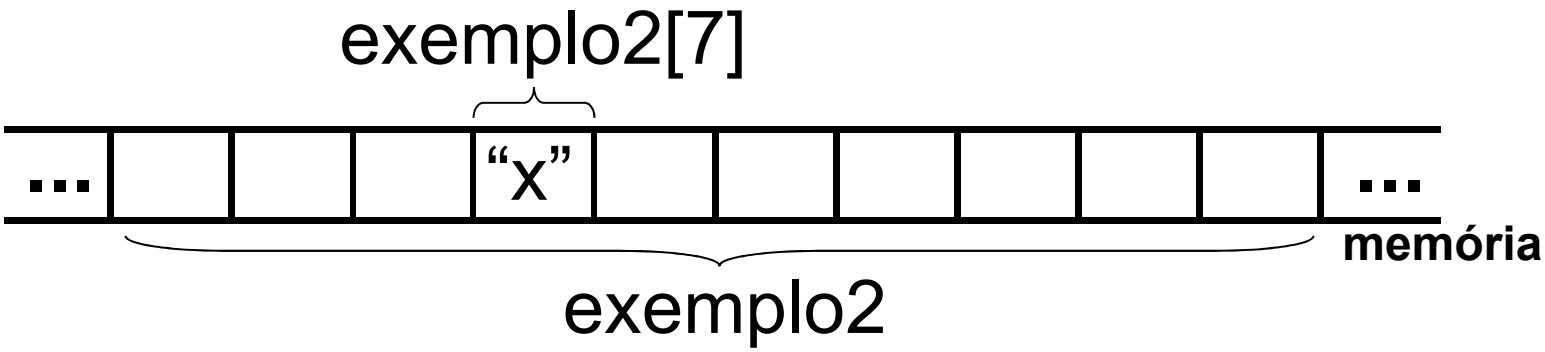
exemplo2[**13**]

Observação: Não é permitida a utilização de índices negativos!

Estruturas de dados homogêneas

Exemplo 2:

exemplo2[7] <- “x”



Estruturas de dados homogêneas

Exercício:

Construa um algoritmo que declare um vetor de inteiros com 12 elementos e o inicialize com números fornecidos pelo usuário, através da entrada padrão.

```
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:

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 através de uma pesquisa nos elementos do vetor, retorne na saída padrão os elementos de menor e maior valor, respectivamente.

UNIVASF

```
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:

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 posteriormente ordene seus elementos em ordem crescente. O algoritmo deve retornar na saída padrão o conteúdo do vetor após a ordenação.

UNIVASF

Classificação por Troca

Toda ordenação está baseada na permutação dos elementos do vetor; logo, sempre dependerá de trocas. São, no entanto, ditos processos por troca aqueles em que a operação de troca é dominante.

Analisaremos agora um método de classificação por troca, conhecido como classificação por troca simples, classificação por bolha ou bubble sort.

UNIVASF

Classificação por Troca - bubble sort

A idéia básica por trás do bubble sort é percorrer a lista seqüencialmente várias vezes. Cada passagem consistem em comparar cada elemento na lista com seu sucessor e trocar os dois elementos se eles não estiverem na ordem correta.

Para uma melhor compreensão examinaremos o seguinte exemplo:

25 57 48 37 12 92 86 33

var x: vetor [0..7] de inteiro

UNIVASF

Classificação por Troca - bubble sort

x = { 25, 57, 48, 37, 12, 92, 86, 33 }

As seguintes comparações são feitas na primeira passagem:

x[0] com x[1] (25 com 57) nenhuma troca

x[1] com x[2] (57 com 48) troca

x[2] com x[3] (57 com 37) troca

x[3] com x[4] (57 com 12) troca

x[4] com x[5] (57 com 92) nenhuma troca

x[5] com x[6] (92 com 86) troca

x[6] com x[7] (92 com 33) troca

Classificação por Troca - bubble sort

Conjunto completo de iterações:

iteração 0 (lista original)	25 57 48 37 12 92 86 33
iteração 1	25 48 37 12 57 86 33 92
iteração 2	25 37 12 48 57 33 86 92
iteração 3	25 12 37 48 33 57 86 92
iteração 4	12 25 37 33 48 57 86 92
iteração 5	12 25 33 37 48 57 86 92
iteração 6	12 25 33 37 48 57 86 92
iteração 7	12 25 33 37 48 57 86 92

```
algoritmo "exercício vetor"
var vet:vetor [1..12] de inteiro
    i , j, temp: inteiro
inicio
   para i de 1 ate 12 faca
      escreva ("Entre com vetor[",i,"]: ")
      leia (vet[i])
   fimpara
   para i de 1 ate 11 faca
      para j de 1 ate 11 faca
         se (vet[j]>vet[j+1]) entao
            temp <- vet[j]
            vet[j] <- vet[j+1]
            vet[j+1] <- temp
         fimse
      fimpara
   fimpara
```

```
para i de 1 ate 12 faca
      se (i=1) entao
         escreva ("Vetor = { ",vet[i])
      senao
         se (i=12) entao
            escreva (", ", vet[i] , " }")
         senao
            escreva (", ",vet[i])
         fimse
      fimse
   fimpara
fimprocedimento
```