Universidade Positivo Disciplina: Programação de Computadores Professor: Marcelo Fassina	aula **04**

Conteúdo
- Vetores e matrizes – estruturas de dados homogêneas.

Competências
- (5) Utilizar estruturas homogêneas de armazenamento com uma ou mais dimensões na construção de sistemas computacionais simples.

1 Estruturas de dados homogêneas

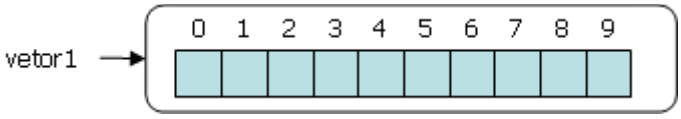
Representam variáveis que têm a capacidade de armazenar mais de um valor, de um determinado tipo, sendo que estes valores podem ser acessados por um índice. Estas estruturas também são conhecidas como variáveis indexadas, variáveis compostas, arranjos, vetores, matrizes ou arrays.

As estruturas mais usadas contêm 1 ou 2 dimensões, sendo aqui tratadas como vetores (arrays unidimensionais) e matrizes (arrays bidimensionais).

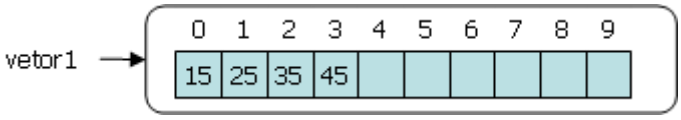
1.1 Vetores

Um vetor representa uma coleção unidimensional de dados do mesmo tipo. Deve-se, portanto, definir o nome da coleção – ou seja, o nome do vetor – e o tamanho máximo de elementos que esta coleção deve possuir – ou seja, a capacidade do vetor.

Um vetor de 10 números inteiros, referenciado pela variável vetor1, tem a seguinte representação:



Neste caso, a variável vetor1 tem capacidade máxima de armazenamento de 10 elementos, que podem ser acessados individualmente pelos seus índices respectivos (de 0 a 9). **Deve-se considerar que o primeiro índice de um vetor é o índice 0**. Supondo que os valores 15, 25, 35 e 45 são armazenados nas 4 primeiras posições de vetor1, tem-se a seguinte representação:



Em linguagem C, a estrutura genérica de declaração e acesso a um vetor é a seguinte:

Declaração:
```
tipo nome_do_vetor[capacidade];
```

Acesso para inserção de valor:
```
nome_do_vetor[índice] = valor;
```

Acesso para recuperação de valor:
```
variável = nome_do_vetor[índice];
```

Desta forma, para declarar o vetor1 ilustrado anteriormente, usa-se:

```
int vetor1[10];
```

Para inserir o valor 15 na posição 0 do vetor1 e recuperar o valor da posição 2 e armazena-lo em uma variável a, usa-se:

```
vetor1[0] = 15;
 a = vetor1[2];
```

Obs.: Em geral, programas que manipulam vetores utilizam uma estrutura de repetição (WHILE, DO, FOR) para percorrer as posições do vetor.

Exemplo1: Escreva um programa que leia 5 números inteiros, armazene-os em um vetor e depois mostre os números na tela.

PORTUGOL

```
ALGORITMO exemplo1
VAR
   INTEIRO: vetor[5], cont;
INÍCIO
   PARA(cont ← 0; cont <= 4; cont ← cont + 1) FAÇA
         ESCREVA("Informe número: ");
         LEIA(vetor[cont]);
   FIM_PARA

   PARA(cont ← 0; cont <= 4; cont ← cont + 1) FAÇA
         ESCREVA(vetor[cont]);
   FIM_PARA
FIM
```

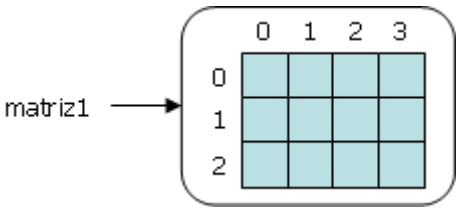
LINGUAGEM C

```
#include <stdio.h>

int main() {

   int vetor[5], i;

   for(i = 0; i <= 4; i++){
         printf("Informe o número: ");
         scanf("%d", &vetor[i]);
   }

   for(i = 0; i <= 4; i++){
         printf("\nNúmero: %d", vetor[i]);
   }

   return (0);
}
```

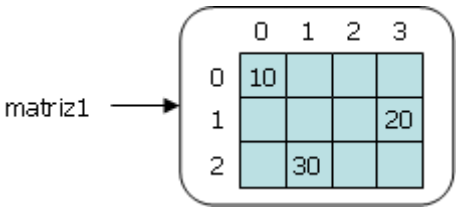
1.2 Matrizes

Uma matriz representa uma coleção bidimensional de dados do mesmo tipo. Deve-se portanto definir o nome da coleção – ou seja, o nome da matriz, o número de linhas da matriz e o número de colunas da matriz. Esta estrutura fornece uma representação tabular dos dados.

Uma matriz 3x4 de números inteiros, referenciado pela variável matriz1, tem a seguinte representação:



Neste caso, a matriz é vista como uma tabela que possui 3 linhas e 4 colunas, com capacidade para armazenar até 12 elementos. **Deve-se considerar que o índice da primeira linha é 0 e o da primeira coluna é 0. Ou seja, o primeiro elemento da matriz está na posição (0,0)**. Supondo que os valores 10, 20 e 30 são armazenados respectivamente nas posições (0,0), (1, 3) e (2,1), tem-se a seguinte representação:



Em linguagem C, a estrutura genérica de declaração e acesso a uma matriz é a seguinte:

Declaração:
```
tipo nome_do_matriz[num_de_linhas][num_de_colunas];
```

Acesso para inserção de valor:
```
nome_da_matriz[índice_linha][índice_coluna] = valor;
```

Acesso para recuperação de valor:
```
variável = nome_da_matriz[índice_da_linha][índice_da_coluna];
```

Desta forma, para declarar a matriz1 ilustrada anteriormente, usa-se:

```
int matriz1[3, 4];
```

Para inserir o valor 10 na posição (0,0) da matriz1 e recuperar o valor da posição (1, 3) e armazená-lo em uma variável a, usa-se:

```
matriz1[0][0] = 10;
a = matriz1[1][3];
```

Obs.: Em geral, programas que manipulam matrizes utilizam duas estruturas de repetição (WHILE, DO, FOR) aninhadas para percorrer as linhas e as colunas da matriz.

Exemplo 2: Escreva um programa que leia 10 números inteiros, armazene-os em uma matriz de 2 linhas e 5 colunas (5 números por linha) e depois mostre os números na tela.

PORTUGOL

```
ALGORIMO exemplo2
VAR
   INTEIRO: matriz[2][5], linha, coluna;
INÍCIO
   PARA(linha ← 0; linha <= 1; linha ← linha + 1) FAÇA
         PARA(coluna ← 0; coluna <= 4; coluna ← coluna + 1) FAÇA
               ESCREVA("Informe o número: ");
               LEIA(matriz[linha][coluna]);
         FIM_PARA
   FIM_PARA

   PARA(linha ← 0; linha <= 1; linha ← linha + 1) FAÇA
         PARA(coluna ← 0; coluna <= 4; coluna ← coluna + 1) FAÇA
               ESCREVA(matriz[linha][coluna]);
         FIM_PARA
   FIM_PARA
FIM
```

LINGUAGEM C

```
#include <stdio.h>

int main() {

   int matriz[2][5], i, j;

   for(i = 0; i <= 1; i++){
         for(j = 0; j <= 4; j++){
               printf("Informe o número: ");
               scanf("%d", &matriz[i][j]);
         }
   }

   for(i = 0; i <= 1; i++){
         for(j = 0; j <= 4; j++){
               printf("\nNúmero: %d", matriz[i][j]);
         }
   }

   return (0);
}
```

EXERCÍCIOS - VETORES

1. Escreva um programa que armazene 10 números inteiros em um vetor e depois mostre os números armazenados na ordem inversa em que foram digitados.

2. Escreva um programa que armazene 10 números inteiros em um vetor. Depois, leia um número inteiro e informe quantas vezes ele aparece no vetor.

3. Escreva um programa que armazene 10 números inteiros em um vetor. Depois, apresente o maior valor armazenado no vetor.

4. Escreva um programa que leia 10 números inteiros e armazene-os em um vetor chamado numeros. Depois, percorra o vetor numeros procurando por números pares. Os números pares encontrados deverão ser armazenados em um vetor chamado pares.

5. Escreva um programa que armazene 10 notas em um vetor e depois calcule a média delas. Mostre quantas notas estão abaixo da média e quantas notas estão acima da média.

6. Escreva um programa que armazene 10 números inteiros em um vetor. Depois alimente dois vetores: um contendo os números positivos e outro contendo os números negativos do vetor inicial.

7. Escreva um programa que armazene 10 números inteiros em um vetor. Depois, atribua o valor 0 para todos os elementos do vetor que possuam valores negativos.

8. Escreva um programa que armazene 10 números inteiros em 2 vetores de tamanho 5. Depois, crie um vetor de 10 elementos que contenha nas posições pares os elementos do primeiro vetor e nas posições ímpares os elementos do segundo vetor.

9. Escreva um programa que armazene 10 números inteiros em vetor e, em um segundo vetor, armazene o quadrado desses números. Por fim, apresente os valores dos dois vetores.

10. Escreva um programa que armazene 10 números inteiros em um vetor. Em seguida, troque o 1° elemento com o último, o 2° elemento com o penúltimo, e assim por diante. Mostre na tela o vetor modificado.

11. Escreva um programa que leia 10 números inteiros e armazene-os em um vetor. Depois leia um número e, se este número estiver no vetor, retire-o, reorganizando os elementos para ocupar o espaço dele. Ao final, mostre os elementos do vetor resultante.

12. Escreva um programa que armazene 10 números inteiros em 2 vetores de tamanho 5. Ao final, o programa deve mostrar os números comuns aos dois vetores.