Linguagens de Programação I

Tema # 7

Vetores ou Matrizes Unidimensionais
Matrices Multidimencionais

Susana M Iglesias

INTRODUÇÃO

- Um vetor geralmente é associado a uma lista ou conjunto de elementos similares,
- Exemplos:
  - Os números inteiros de 1..10
  - As notas de uma turma em uma disciplina
  - Os resultados de uma enquête sobre a cantina da UESC
  - Preços de venda dos produtos de uma loja

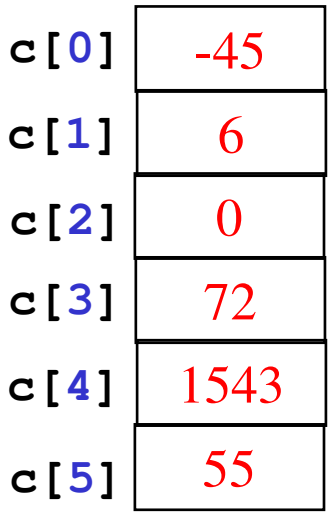
INTRODUÇÃO

- Vetores definição:
  1. itens de dados **do mesmo tipo**, relacionados entre si.
  2. coleção de variáveis **do mesmo tipo** que é referenciado por um **nome comum**.
- Os vetores são entidades estáticas, i.e. permanecem do mesmo tamanho ao longo da execução do programa,
- um vetor é um grupo de locais de memória relacionados pelo fato que tem o mesmo nome e o mesmo tipo,

INTRODUÇÃO

- os elementos do vetor ocupam posições contíguas na memória,
- o endereço mais baixo corresponde ao primeiro elemento e o mais alto ao último,

Representação de um vetor



`c` – nome do array
- valores do array
- subscritos do array

SUBSCRITOS

- para fazer referencia a um elemento no vetor precisamos o nome do vetor e a posição de aquele elemento no vetor,

```
c[0], a[21]
```

- a posição de um elemento no vetor é formalmente conhecida como subscrito,
- um subscrito deve ser um inteiro ou uma expressão inteira,
- para um vetor de *N* elementos os subscritos variam de 0 a *N-1*,

DECLARANDO VETORES

- Ao declarar um vetor reservamos espaço na memória para ele,
- portanto, devemos especificar o tipo de cada elemento e a quantidade apropriada de elementos,

```
tipo nome_do_array[tamanho]
```

- Exemplos:
  - `int c[6];`
  - `float b[100];`
  - `char s[25];`
- a linguagem C não incorpora verificação de subscritos, utilizar um valor do subscrito fora do vetor não gera nenhuma mensagem de erro.

EXEMPLO

```
#include <stdio.h>
#include <stdlib.h>

int main()
{
  int n[10], i;

  for(i=0; i<=9; i++)
    n[i] = i;

  printf("%s%13s\n", "Elemento", "Valor");
  for(i=0; i<=9; i++)
    printf("%7d%13d\n",i,n[i]);

  system("PAUSE");
  return 0;
}
```

```
Elemento        Valor
      0            0
      1            1
      2            2
      3            3
      4            4
      5            5
      6            6
      7            7
      8            8
      9            9
Press any key to continue . . .
```

INICIALIZAÇÃO DE VETORES

- Ao igual que as variáveis os elementos de um vetor podem ser inicializados na declaração do vetor,
- para inicializar um vetor utilizamos um sinal de igual e uma lista de inicializadores separados por vírgulas, (entre chaves),

```
int n[5] = {1, 2, 3, 4, 5};
char pal[6] = {'b','r','a','s','i','l'};
```

INICIALIZAÇÃO DE VETORES

- se houver menos inicializadores que o número de elementos do vetor, os elementos restantes são inicializados com zero,

```
float num[10] = {1.0, 2.0, 3.0};
int n[5] = {};
```

- se houver mais inicializadores que termos no vetor teremos um erro de compilação,

```
int n[2] = {1, 2, 3};
```

- se o tamanho do vetor for omitido na declaração, o numero de elementos será igual ao número de inicializadores,

```
float num[] = {1.0, 2.0, 3.0};
```

INICIALIZAÇÃO DE VETORES

- Que acontece se utilizarmos os elementos de um vetor sem ter inicializado eles?

PONTEIROS vs VETORES

- Na linguagem C vetores e ponteiros estão intimamente relacionados,

```
int a[10]
```

- `a`, é um ponteiro inteiro que aponta ao endereço base do vetor,
- `a[0]`, referencia o conteúdo do endereço base do vetor,

PONTEIROS x VETORES

- no lugar de vetores podemos utilizar ponteiros e alocação dinâmica de memória, este enfoque tem maior complexidade do ponto de vista computacional e uma sobrecarga ao executar o programa,
- a quantidade de memória para um vetor é reservada pelo compilador no ato de compilação do programa,

PONTEIROS x VETORES

- se a quantidade de elementos do vetor não for conhecida, o maior número de elementos possíveis deverá ser alocado, geralmente esta estratégia produz um uso ineficiente da memória,
- em tais casos existe a alternativa de utilizar ponteiros e alocação dinâmica de memória,

ARMAZENAMENTO - VETORES

- a quantidade memória para armazenar uma matriz é alocada durante a compilação,
- a quantidade de memória alocada dependera do número de elementos e o tipo de dado dos elementos,
- podemos mostrar o endereço de uma matriz imprimindo seu nome,
- podemos mostrar a quantidade de memória ocupada pela matriz utilizando a função **`sizeof(nome_da_matriz)`**

ARMAZENAMENTO - VETORES

- Exemplo:

```
int c[10];

printf("Endereço do array %p\n", c);

printf("Memória alocada em bytes: %d\n", sizeof(c));
```

EXEMPLO

```
#include <stdio.h>
#include <stdlib.h>
#define N 5

int main()
{
  int i, n[N] = {32, 27, 64, 18, 95}, soma = 0;

  for(i=0; i<N; i++)
    soma += n[i];

  printf("%s%13s\n", "Elemento", "Valor");
  for(i=0; i<N; i++)
    printf("%7d%13d\n", i, n[i]);
  printf("Soma: %d\n", soma);

  system("PAUSE");
  return 0;}
```

```
Elemento        Valor
      0           32
      1           27
      2           64
      3           18
      4           95
Soma: 236
Press any key to continue . . .
```

EXEMPLO

Foi perguntado a 20 alunos o nível de qualidade da comida na cantina da UESC, em uma escala de 0 a 5 (0 significa horrorosa e 5 significa excelente). Escreva um programa que resuma os resultados da pesquisa (mostre a quantidade de vezes que aparece cada nota), imprima um histograma com os resultados.

```
#define TAM_R 20
#define TAM_F 6

int main()
{
  int i, j;
  int respostas[TAM_R] = {3, 4, 5, 2, 1, 1, 2, 0, 3, 4,
                          5, 2, 2, 4, 4, 3, 1, 2, 2, 3};
  int frequencias[TAM_F] = {0};

  for(i=0; i<TAM_R; i++)
    ++frequencias[respostas[i]];
  printf("%s%13s%12s\n", "Nivel", "Frequencia", "Histograma");
  for(i=0; i<TAM_F; i++){
    printf("%5d%13d    ", i, frequencias[i]);
    for(j=0; j<frequencias[i]; j++)
      printf("*");
    printf("\n");
  }

  system("PAUSE");
  return 0;
}
```

```
Nivel    Frequencia  Histograma
    0             1    *
    1             3    ***
    2             6    ******
    3             4    ****
    4             4    ****
    5             2    **
```

OPERAÇÕES ARITMETICAS

- Adição e Subtração de vetores:

$$\vec{c} = \vec{a} \pm \vec{b}$$

$$c[i] = a[i] + b[i]$$

- os vetores a e b devem ter a mesma dimensão,
- o vetor resultante c tem a mesma dimensão que a e b
- Norma do vetor: $\left\|\vec{c}\right\| = \sqrt{\sum_{i=0}^{N-1} \left(c[i]\right)^2}$

CATEGORIAS ESTATISTICAS

- Media

$$\overline{c} = \frac{\sum_{i=0}^{N-1} c[i]}{N}$$

- Moda: é o elemento que mais se repete entre os elementos do vetor.
- Maximo
- Mínimo
- Mediana: é o elemento "central" de um vetor ordenado (N – par ou N – ímpar)

BUSCA E ORDENAÇÃO

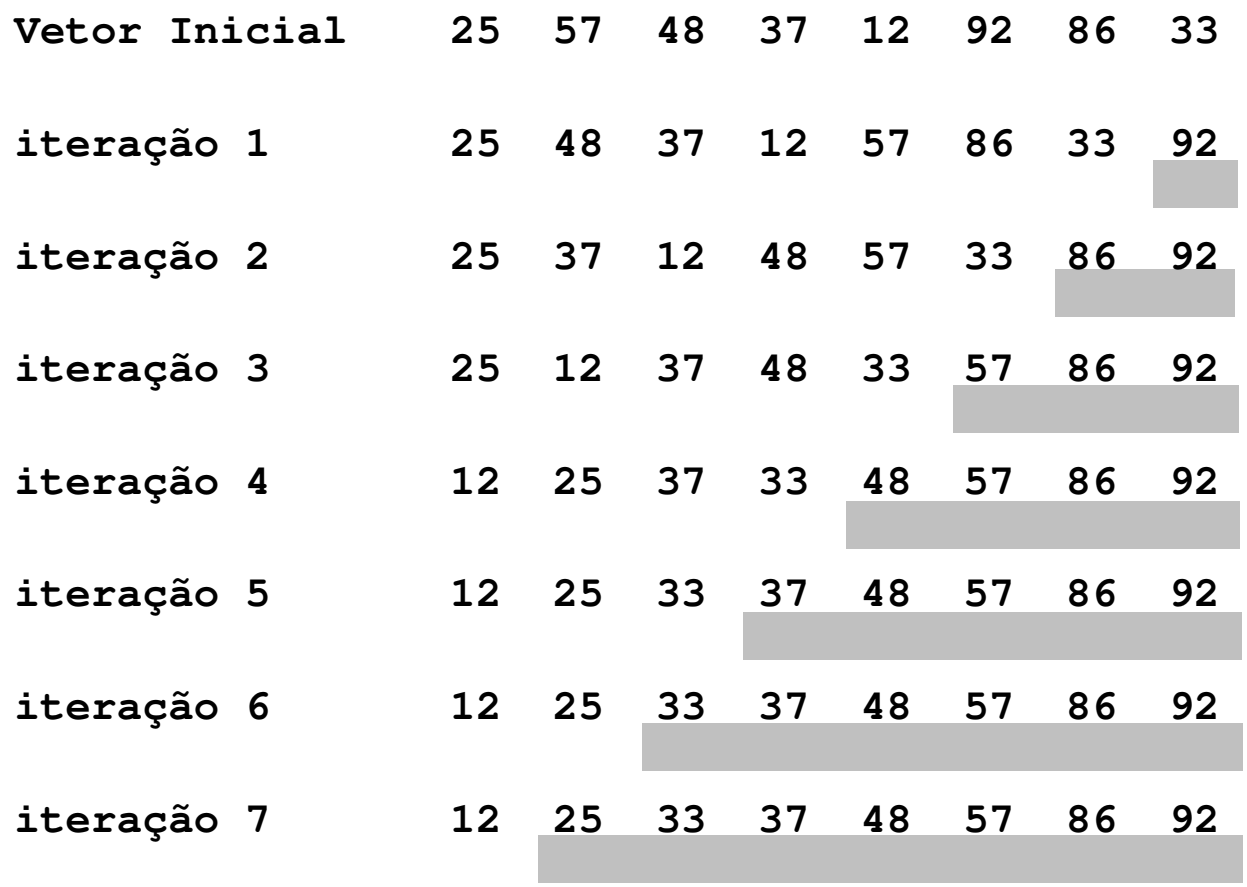
- A operação de busca consiste em informar se um determinado valor existe ou não em um conjunto de elementos (vetor).
- A operação de ordenar (classificar) um vetor consiste em organizar seus elementos de forma que

$$c[0] \leq c[1] \leq \ldots \leq c[i] \leq c[i+1] \leq \ldots \leq c[N-1]$$

- Existem diversos algoritmos de ordenação um dos mais simples é o método de ordenação por bolhas.

CLASSIFICAÇÃO POR BOLHAS

- realizar passagens sucessivas no arquivo, em cada passagem se x[i] > x[i+1], permutar os elementos



CLASSIFICAÇÃO POR BOLHAS

```
#define N 8
int main()
{
  int x[N] = {25, 57, 48, 37, 12, 92, 86, 33};
  int i, j, k, aux;

  for(k=0;k<N;k++)
    printf("%4d", x[k]);
  printf("\n");

  for(i=0; i<N; i++){
    for(j=0; j<(N-i-1); j++)
      if (x[j]>x[j+1]){
        aux = x[j];
        x[j] = x[j+1];
        x[j+1] = aux;
      }
  for(k=0;k<N;k++)   printf("%4d", x[k]);
  printf("\n");
  }
return 0;}
```

FUNÇÕES - VETORES

- No exemplo anterior o vetor é impresso no começo do programa e cada vez que uma passagem de ordenação e terminada,
- Podemos simplificar o programa do exemplo anterior se introduzirmos uma função para imprimir o vetor,
- Ilustramos o uso de vetores como parâmetros de uma função através do seguinte exemplo.

FUNÇÕES - VETORES

```
#define N 8

void prn_vet(int [], int); /* prototipo da função */

int main()
{
  int x[N] = {25, 57, 48, 37, 12, 92, 86, 33};
  int i, j, k, aux;

  prn_vet(x, N); /* chamada a função */

  for(i=0; i<N; i++){
    for(j=0; j<(N-i-1); j++)
      if (x[j]>x[j+1]){
        aux = x[j];
        x[j] = x[j+1];
        x[j+1] = aux;
      }
  prn_vet(x, N);
  } system("PAUSE");
  return 0;}
```

```
/* definição da função */
void prn_vet(int v[], int n){
  int k;
  for(k=0;k<N;k++)
    printf("%4d", v[k]);
  printf("\n");
}
```

CONSIDERAÇÕES FINAIS DE VETORES

- Matrizes unidimensionais ou vetores são, essencialmente, listas de informações do mesmo tipo, que são armazenadas em posições contíguas na memória em ordem crescente.
- Os elementos do vetor estão relacionados por um índice ou subscrito, através dele acessamos os elementos do vetor.

MATRIZ - INTRODUÇÃO

- Os vetores ou matrizes em C podem ter vários subscritos,
- Uma matriz com mais de um subscrito é chamada multidimensional,
- Matrizes de dois subscritos ou bidimensionais são usualmente utilizados para representar tabelas de valores do mesmo tipo.

MATRIZ - INTRODUÇÃO

- Exemplos:
  - Notas de uma turma em uma disciplina considerando várias avaliações,
  - Resultados de um campeonato de futebol (equipes, rodadas, pontos)
  - Estoque de uma loja de roupa (tamanho, cores). Como considerar vários produtos?

REPRESENTAÇÃO DE UMA MATRIZ BIDIMENSIONAL

Matriz de 3 x 4

	Coluna 0	**Coluna 1**	**Coluna 2**
Linha 0	`a[0][0]`	`a[0][1]`	`a[0][2]`
Linha 1	`a[1][0]`	`a[1][1]`	`a[1][2]`
Linha 2	`a[2][0]`	`a[2][1]`	`a[2][2]`
Linha 3	`a[3][0]`	`a[3][1]`	`a[3][2]`

nome da matriz
subscrito da linha
subscrito da coluna

REPRESENTAÇÃO DE UMA MATRIZ BIDIMENSIONAL

Notas de LP1

	Avaliação 0	Avaliação 1	Avaliação 2
Aluno 0	5.3	8.3	7.1
Aluno 1	4.2	6.8	7.3
Aluno 2	6.0	5.4	1.0
Aluno 3	9.7	10.0	9.6
Aluno 4	2.1	5.8	7.9

DECLARAÇÃO DE UMA MATRIZ BIDIMENSIONAL

- Sintaxe:

```
tipo nome_matriz[num_linhas][num_colunas];
```

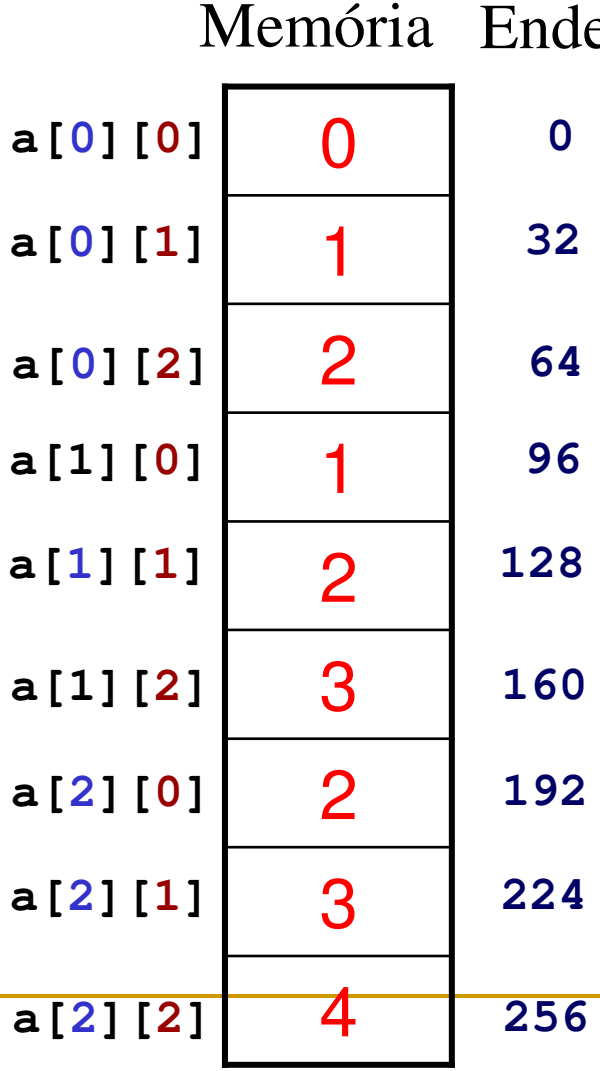
- Exemplos:

```
float n[5][3];
int a[4][4];
char ch[10][5];
```

- Ao igual que vetores matrizes podem ser inicializadas na declaração utilizando uma lista de inicializadores,
- Exemplos:

```
int n[3][3] = {{0,1,2},{1,2,3},{2,3,4}};
float a[4][4] = {{1,2,3,4},{5,6},{6}};
```

ARMAZENAMENTO NA MEMÓRIA DA MATRIZ BIDIMENSIONAL



- Mecanismo de busca de um elemento da matriz

EXEMPLO 1

```
#define lin 5
#define col 3

int main()
{
  float notas[lin][col] = {{5.3,  8.3, 7.1},{4.3,  6.8, 7.3},
                           {6.0,  5.4, 1.0},{9.7, 10.0, 9.6},
                           {2.1,  5.8, 7.9}};

  int i, j;

  printf("   ");
  for(j=0;j<col;j++)
    printf("%6d",j);
  printf("\n");
  for(i=0;i<lin;i++){
    printf("%3d",i);
    for(j=0;j<col;j++)
      printf("%6.1f", notas[i][j]);
    printf("\n");
  }
}
```

```
          0     1     2
  0     5.3   8.3   7.1
  1     4.3   6.8   7.3
  2     6.0   5.4   1.0
  3     9.7  10.0   9.6
  4     2.1   5.8   7.9
Press any key to continue . . .
```

EXEMPLO 2

Modifique o exercício anterior para criar um vetor com a nota final de cada um dos alunos, e outro vetor com a média da turma em cada avaliação.

```
#include <stdio.h>
#include <stdlib.h>

#define lin 5
#define col 3

int main()
{
  float notas[lin][col] = {{5.3,  8.3, 7.1},{4.3,  6.8, 7.3},
                           {6.0,  5.4, 1.0},{9.7, 10.0, 9.6},
                           {2.1,  5.8, 7.9}};
  float alunos[lin]={0}, aval[col]={0};
  int i, j;

  printf("   ");
  for(j=0;j<col;j++)
    printf("%6d",j);
  printf("\n");
```

EXEMPLO 2

```
  for(i=0;i<lin;i++){
    printf("%3d",i);
    for(j=0;j<col;j++){
      printf("%6.1f", notas[i][j]);
      alunos[i] += notas[i][j];
      aval[j] += notas[i][j];
    }
    printf("\n");
  }

  printf("\nNotas alunos:\n");
  for(i=0;i<lin;i++)
    printf("%d %6.1f\n", i, alunos[i]/col);

  printf("\nMedia das avaliacoes:\n");
  for(j=0;j<col;j++)
    printf("%d %6.1f\n", j, aval[j]/lin);

  system("PAUSE");
  return 0;
}
```

EXEMPLO 2

```
        0     1     2
  0   5.3   8.3   7.1
  1   4.3   6.8   7.3
  2   6.0   5.4   1.0
  3   9.7  10.0   9.6
  4   2.1   5.8   7.9

Notas alunos:
0    6.9
1    6.1
2    4.1
3    9.8
4    5.3

Media das avaliacoes:
0    5.5
1    7.3
2    6.6
Press any key to continue . . .
```

OPERAÇÕES COM MATRIZES

- Adição e Subtração:

$$\overline{\overline{C}}_{IxJ} = \overline{\overline{A}}_{IxJ} \pm \overline{\overline{B}}_{IxJ}$$

$$c[i][j] = a[i][j] \pm b[i][j]$$

- a operação só esta definida se as matrizes *A* e *B* tiverem o mesmo numero de linhas e colunas,
- a matriz *C* terá igual número de colunas que *A* e *B*

OPERAÇÕES COM MATRIZES

- Multiplicação de matriz por vetor:

$$\vec{v}_I = \overline{\overline{A}}_{IxJ}\vec{d}_J$$

$$v[i] = a[i][0]d[0] + a[i][1]d[1] + \ldots + a[i][N-1]d[N-1]$$

$$v[i] = \sum_{j=0}^{N-1} a[i][j]d[j]$$

- a operação só esta definida se o numero de colunas de *A* for igual ao numero de elementos de *d*,
- a quantidade de elementos de *v* será o número de linhas de *A*.

OPERAÇÕES COM MATRIZES

- Multiplicação de matriz por vetor:

$$\overline{\overline{C}}_{IxM} = \overline{\overline{A}}_{IxJ} \times \overline{\overline{B}}_{JxM}$$

$$c[i][j] = a[i][0]b[0][j] + a[i][1]b[1][j] + \ldots + a[i][N-1]b[N-1][j]$$

$$c[i][j] = \sum_{k=0}^{N-1} a[i][k]b[k][j]$$

- a operação só esta definida se o numero de colunas de *A* for igual ao numero de linhas de *B*.