Universidade Federal de Ouro Preto - UFOP
Departamento de Computação - DECOM
Comissão da Disciplina Programação de Computadores I – CDPCI
Programação de Computadores I – BCC701
www.decom.ufop.br/red

Aula Teórica 12

Material Didático Proposto

Conteúdos da Aula

- Vetores
- Algumas Funções Aplicadas a Vetores
- Exercícios

Vetores

Problema

Suponha que você deseje armazenar a nota da primeira prova de programação, de cada um dos 400 alunos de BCC701.

Seu programa teria o seguinte código para leitura de dados:

```
Nota_001 = input("DIGITE A NOTA: ");
Nota_002 = input("DIGITE A NOTA: ");
Nota_003 = input("DIGITE A NOTA: ");
Nota_004 = input("DIGITE A NOTA: ");
Nota_005 = input("DIGITE A NOTA: ");
Nota_006 = input("DIGITE A NOTA: ");
Nota_007 = input("DIGITE A NOTA: ");
• • •
Nota_400 = input("DIGITE A NOTA: ");
```

Problema

Para a impressão de dados:

```
printf("NOTA DO ALUNO: %g", Nota_001);
printf("NOTA DO ALUNO: %g", Nota_002);
printf("NOTA DO ALUNO: %g", Nota_003);
printf("NOTA DO ALUNO: %g", Nota_004);
printf("NOTA DO ALUNO: %g", Nota_005);
printf("NOTA DO ALUNO: %g", Nota_006);
printf("NOTA DO ALUNO: %g", Nota_007);
• • •
printf("NOTA DO ALUNO: %g", Nota_400);
```

Com esta codificação:

- **É necessário um nome diferente de variável para cada nota de aluno.**
- **O programador deve se lembrar que a variável cujo nome é Nota_003 representa a nota do aluno, por exemplo, BART. Logo, quando o programador manipula as notas ele deve se lembrar da associação que fez entre aluno e nome de variável.**
- **Se o programa manipulasse 1000 alunos, seriam necessárias 1000 variáveis com nomes diferentes.**

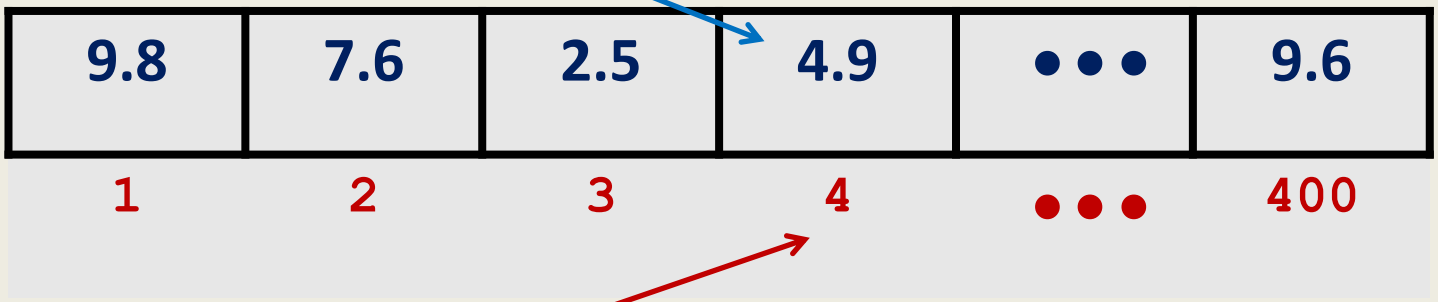
Uma solução:

- Seria interessante se pudéssemos armazenar todos os dados referentes as notas dos alunos em um "conjunto".
- Assim, todos os dados teriam um único nome de variável (o nome do conjunto) na memória.
- Para identificar cada elemento do conjunto individualmente, associaríamos ao nome do aluno um índice (1 até 400).
- Você já faz este tipo de associação em Geometria, para o terno ordenado (x, y, z), onde, o índice de x é 1, de y é 2, e de z é 3.

Uma solução:

- Em programação este conjunto de dados recebe o nome de VETOR, e no nosso exemplo teríamos um vetor com o nome Notas_BCC701 ilustrado por:

notas dos alunos



índices do vetor

Tipo de Dados Vetor

- Um vetor possui um nome, como uma variável comum, para armazenar dados na memória.
- Todos os elementos do vetor são do mesmo tipo (inteiro, real, string, booleano, etc.). Os índices são sempre inteiros.
- Os elementos de um vetor são unicamente identificados pelos seus respectivos índices.

Tipo de Dados Vetor

- Um vetor lembra um conjunto usado na matemática. A diferença aqui é que os elementos do vetor estão organizados em posições consecutivas e ordenadas na memória. A ordem é determinada pelos índices inteiros: 1, 2, 3, ...; não pelos dados do vetor.
- Um vetor representa um conjunto ordenado, pelo índice, de valores homogêneos (do mesmo tipo). Por esta razão um vetor é denominado uma Estrutura de Dados Homogênea.

Vetor Linha

- Os elementos são separados pelo espaço em branco ou pela vírgula.

```
--> massa = [ 20.6 45 13.8 ]
 massa  =
    20.6    45.    13.8
--> peso = [ 23, 44, 78 ]
 peso  =
    23.    44.    78.
--> massa(1)
 ans  =       20.6
-->peso(3)
 ans  =       78.
```

Vetor

```
--> peso(1)^2
 ans  =       529.

--> massa(1) * sqrt(massa(2)) - peso(2)^2
 ans  =       - 1797.811

--> massa(1)=0;

--> massa(2)= massa(2) + massa(3);

--> massa
 massa  =       0.    58.8    13.8
```

Vetor

```
--> massa
 massa  =       0.     58.8     13.8
--> peso
 peso  =       23.     44.     78.
--> massa(1) = massa(1) + peso(1)
 massa  =       23.     58.8     13.8
--> peso(2) = massa(1) + massa(3)
 peso  =       23.     36.8     78.
--> massa
 massa  =       23.     58.8     13.8
-->peso
 peso  =       23.     36.8     78.
```

Vetor Coluna

- Os elementos são separados por ponto e vírgula.

```
v = [11; 22; 33; 44 ]

  v  =

     11.
     22.
     33.
     44.
```

Vetor

```
--> v(2) = v(2) * 100;


--> v
 v  =

    11.
    2200.
    33.
    44.
```

Vetor Coluna

- Os elementos também podem ser definidos por atribuição;

```
--> nota(1) = 6.8;
--> nota(2) = 7.6;
--> nota(3) = 8.8;
--> nota(4) = 9.5;
--> nota
 nota  =
    6.8
    7.6
    8.8
    9.5
```

Vetor - Definição

- **Definindo por faixas de elementos**

Vetor = <valorInicial> : <incremento> : <valorFinal>

```
--> V1 = 1:5; // o incremento 1 é opcional
--> V2 = 2:0.2:3;
--> V1
 V1  =
    1.    2.    3.    4.    5.

--> V2
 V2  =
    2.    2.2    2.4    2.6    2.8    3.
```

Vetor - Definição

- Vetor de 1’s
- Todos os elementos assumirão valor inicial 1

Vetor = ones(<linhas>, <colunas>)

- Vetor: nome da variável do tipo vetor;
- ones: função que retorna uma matriz* com valores 1;
- <linhas>: número de linhas;
- <colunas>: número de colunas;

- * Matriz é objeto de estudos do próximo tópico abordado na disciplina;
- Para construir um vetor, o número de linhas OU o número de colunas deve ser igual a um.

Vetor

- **Vetor de 1's**
  - **Vetor linha com cinco colunas:**
    - **c = ones(1, 5)**

      c = 1. 1. 1. 1. 1.
  - **Vetor coluna com cinco linhas:**
    - **c = ones(5, 1)**

      c = 1.
      1.
      1.
      1.
      1.

Vetor - Definição

- **Vetor de 0's**
- Todos os elementos assumirão valor inicial **0**

Vetor = zeros(<linhas>, <colunas>)

  - **Vetor**: nome da variável do tipo vetor;
  - **zeros**: função que retorna uma **matriz*** com valores 0;
  - **<linhas>**: número de linhas;
  - **<colunas>**: número de colunas;

  - * **Matriz** é objeto de estudos do próximo tópico abordado na disciplina;
  - Para construir um **vetor**, o número de linhas OU o número de colunas deve ser igual a **um**.

- **Vetor de 0's**

  - **Vetor linha com cinco colunas**:
    - c = zeros(**1**, 5)

      c = 0. 0. 0. 0. 0.

  - **Vetor coluna com cinco linhas**:
    - c = zeros (5, **1**)

      c = 0.
      0.
      0.
      0.
      0.

Vetor – Acesso aos Elementos

- **Para acessar um elemento específico**:

  Vetor(<índice>)

  – Exemplo:

  V = [10, 20, **30,** 40, 50];

  disp(**V(3)**);

  – Resultado: 30

  – Pode ser aplicado tanto a vetor de coluna quanto de linha:
    - Para **vetor linha**, **<índice>** corresponde ao **número da coluna**;
    - Para **vetor coluna**, **<índice>** corresponde ao **número da linha**;

  – Pode ser usado para modificar o valor: **V(3) = 300**, modifica o valor do **índice 3** de **30** para **300**.

Vetor – Transposição de vetores

- **Operador apóstrofo (')**: Vetor'
  - Transforma um vetor coluna em um vetor linha, e vice-versa:

```
V = [10, 20, 30, 40, 50];
V = V';
disp(V);
```

  - Resultado:

```
10.
20.
30.
40.
50.
```

  - Também poderia ser feito: V = [10:10:50]', para obter o mesmo resultado anterior;

Vetor – Operações Binárias

- **Soma com escalar: V + <valor> *OU* <valor> + V:**
  - **Exemplo:**

```
clc;
V1 = [1 2 3 4 5];
V2 = V1 + 2;  // Ou V2 = 2 + V1; (causará o mesmo
              // efeito)
disp(V2);
```

  - **Resultado:**

```
 3.    4.    5.    6.    7.
```

Vetor – Operações Binárias

- **Soma de vetores: V1 + V2:**
  - **V1 e V2 devem ser da mesma dimensão;**
  - **Exemplo:**

```
clc;
V1 = [1 2 3 4 5];
V2 = [5 4 3 2 1];
Soma = V1 + V2;
disp(Soma);
```

  - **Resultado:**

```
6.    6.    6.    6.    6
```

Vetor – Operações Binárias

- **Subtração com escalar: Vetor - <valor> *OU***

  <valor> - Vetor:

  - **Exemplo:**

```
V1 = [1 2 3 4 5];
V2 = V1 - 1;
V3 = 1 - V1;
disp(V2);
disp(V3);
```

  - **Resultado:**

```
 0.    1.    2.    3.    4.
 0.  - 1.  - 2.  - 3.  - 4.
```

Vetor – Operações Binárias

- **Subtração de vetores: V1 - V2:**
  - V1 e V2 devem ser da mesma dimensão;
  - Exemplo:

```
V1 = [1 2 3 4 5];
V2 = [5 4 3 2 1];
Subtracao = V1 - V2;
disp(Subtracao);
```

  - Resultado:

```
- 4.  - 2.    0.    2.    4.
```

Vetor – Operações Binárias

- **Multiplicação por escalar: V * <valor> *OU* <valor> * V:**
  - **Exemplo:**

```
V1 = [1 2 3 4 5];
V2 = V1 * 2;  // Ou V2 = 2 * V1; (causará o mesmo
              // efeito)
disp(V2);
```

  - **Resultado:**

2. 4. 6. 8. 10.

Vetor – Operações Binárias

- **Multiplicação de vetores: V1 .* V2:**
  - **V1 e V2 devem ser da mesma dimensão;**
  - **Exemplo:**

```
V1 = [1 2 3 4 5];
V2 = [5 4 3 2 1];
Mult = V1 .* V2;
disp(Mult);
```

  - **Resultado:**

```
5.   8.   9.   8.   5.
```

Vetor – Operações Binárias

- **Produto interno: V1 * V2:**
  - V1 é um vetor linha e V2 é um vetor coluna;
  - O número de colunas de V1 deve ser igual ao número de linhas de V2;
  - Exemplo:

```
V1 = [1 2 3 4 5];  // Vetor linha
V2 = [5;4;3;2;1];  // Vetor coluna
Mult = V1 * V2;
disp(Mult);
```

  - Resultado:

    35.

Vetor – Operações Binárias

- **Divisão por escalar: V / <valor> *OU* <valor> \ V:**
  - **Exemplo:**

```
V1 = [1 2 3 4 5];
V2 = V1 / 2;  // Ou V2 = 2 \ V1; (causará o mesmo
              // efeito)
disp(V2);
```

  - **Resultado:**

```
0.5    1.    1.5    2.    2.5
```

- **Divisão de vetores à direita: V1 ./ V2:**
  - **V1 e V2 devem ser da mesma dimensão;**
  - **Exemplo:**

```
clc;
V1 = [1 2 3 4 5];
V2 = [5 4 3 2 1];
Div = V1 ./ V2;
disp(Div);
```

  - **Resultado:**

```
  0.2   0.5   1.   2.   5.
```

Vetor – Operações Binárias

- **Divisão de vetores à esquerda: V1 .\ V2:**
  - **V1 e V2 devem ser da mesma dimensão;**
  - **Exemplo:**

```
clc;
V1 = [1 2 3 4 5];
V2 = [5 4 3 2 1];
Div = V1 .\ V2;
disp(Div);
```

  - **Resultado:**

```
    5.    2.    1.    0.5    0.2
```

Algumas Funções Aplicadas a Vetores

Algumas Funções Aplicadas a Vetores

- **Dimensão de vetores;**
- **Somatório;**
- **Somatório cumulativo;**
- **Produtório;**
- **Produtório cumulativo;**
- **Elementos únicos;**
- **União;**
- **Interseção;**
- **Busca (pesquisa);**
- **Ordenação;**
- **Plotando gráficos.**

- **Dimensão de vetores**

[resultado] = length(<Vetor>)

- **Retorna a quantidade de elementos do vetor, muito útil para construir laços de repetição para percorrer os elementos do vetor:**

```
V = [10, 20, 30, 40, 50];
n = length(V);
disp(n);
```

- **Resultado:**

5.

- **Somatório**

[resultado] = sum(<Vetor>)

- **Retorna o somatório de todos os elementos do vetor:**

```
V = [10, 20, 30, 40, 50];
somatorio = sum(V);
disp(somatorio);
```

- **Resultado:**

150.

- **Perceba que o resultado é um valor numérico.**

- **Somatório cumulativo**

[resultado] = cumsum(<Vetor>)

- **Retorna o somatório de todos os elementos do vetor, de forma acumulativa a cada linha/coluna:**

```
V = [10, 20, 30, 40, 50];
somatorio = cumsum(V);
disp(somatorio);
```

- **Resultado:**

```
10.   30.   60.   100.   150.
```

- **Perceba que o resultado é um vetor.**

- **Produtório**

[resultado] = prod(<Vetor>)

- **Retorna o produtório de todos os elementos do vetor:**

```
V = [10, 20, 30, 40, 50];
produtorio = prod(V);
disp(produtorio);
```

- **Resultado:**

12000000.

- **Perceba que o resultado é um valor numérico.**

- **Produtório cumulativo**

  [resultado] = cumprod(<Vetor>)

- **Retorna o produtório de todos os elementos do vetor, de forma acumulativa a cada linha/coluna:**

```
V = [10, 20, 30, 40, 50];
produtorio = cumprod(V);
disp(produtorio);
```

- **Resultado:**

```
10.   200.   6000.   240000.   12000000.
```

- **Perceba que o resultado é um vetor.**

- **Elementos únicos**

[resultado [, k]] = unique(<Vetor>)

• Retorna um vetor ordenado contendo os elementos únicos de um vetor, adicionalmente retorna um vetor com os índices dos elementos no vetor de entrada:

```
V= [60, 30, 40, 50, 20, 20, 30, 10, 70, 80];
[unicos, indices] = unique(V);
disp(unicos);
disp(indices);
```

• Resultado:

```
 10.   20.   30.   40.   50.   60.   70.   80.
  8.    5.    2.    3.    4.    1.    9.   10.
```

← Elementos únicos de V

← Índices dos elementos em V

- **União**

[resultado [, kA, kB]] = union(<Vetor A>, <Vetor B>)

- **Retorna um vetor ordenado contendo a união entre os elementos de dois vetores, adicionalmente retorna vetores com os índices dos elementos em cada vetor de entrada:**

```
V1 = [60, 30, 40, 50, 20];
V2 = [20, 30, 10, 70, 80];
[uniao, indicesA, indicesB] = union(V1, V2);
disp(uniao);   disp(indicesA);   disp(indicesB);
```

- **Resultado:**

```
 10.   20.   30.   40.   50.   60.   70.   80.
```
← Elementos únicos de V1 ∪ V2

```
   5.    2.    3.    4.    1.
```
← Índices dos elementos em V1

```
   3.    4.    5.
```
← Índices dos elementos em V2

- **Interseção**

  [resultado [, kA, kB]] = intersect(<Vetor A>, <Vetor B>)

- **Retorna um vetor ordenado contendo os elementos em comum de dois vetores, adicionalmente retorna vetores com os índices dos elementos em cada vetor de entrada:**

```
V1 = [60, 30, 40, 50, 20];
V2 = [20, 30, 10, 70, 80];
[intersecao, indicesA, indicesB] = intersect(V1, V2);
disp(intersecao);   disp(indicesA);   disp(indicesB);
```

- **Resultado:**

```
20.   30.        ← Elementos de V1 ∩ V2
   5.    2.      ← Índices dos elementos em V1
   1.    2.      ← Índices dos elementos em V2
```

Algumas Funções Aplicadas a Vetores

- **Busca** (pesquisa);

[índices] = find(<condição>[, <nmax>])

- **Retorna um vetor ordenado contendo os índices dos elementos de um vetor que atendem à condição de entrada (o número de índices é limitado a nmax, o valor -1 (padrão) indica "todos"):**

```
V = [60, 30, 40, 50, 20, 20, 30, 10, 70, 80];
encontrados1 = find(V > 50);
encontrados2 = find(V == 30);
encontrados3 = find(V == 30 | V == 20);
disp(encontrados1);   disp(encontrados2);   disp(encontrados3);
```

- **Resultado:**

```
1.   9.   10.      ← Elementos de V maiores de 50
2.   7.           ← Elementos de V iguais a 30
2.   5.   6.   7. ← Elementos de V iguais a 30 OU iguais a 20
```

- **Ordenação**

  [resultado, indices] = gsort(<Vetor>[, tipo[, direção]])

- **Exemplo:**

```
V = [60, 30, 40, 50, 20, 20, 30, 10, 70, 80];
[ordenado1, indice1] = gsort(V);
[ordenado2, indice2] = gsort(V, 'g', 'i');
disp(ordenado1);      // Ordenação padrão
disp(indice1);        // Índices dos elementos da ordenação padrão
disp(ordenado2);      // Ordenação de forma crescente
disp(indice2);        // Índices dos elementos da ordenação crescente
```

- **Resultado:**

```
 80.   70.   60.   50.   40.   30.   30.   20.   20.   10.
 10.   9.   1.   4.   3.   2.   7.   5.   6.   8.
 10.   20.   20.   30.   30.   40.   50.   60.   70.   80.
 8.   5.   6.   2.   7.   3.   4.   1.   9.   10.
```

- **Ordenação**

  [resultado, indices] = gsort(<Vetor>[, tipo, direção])

- **Retorna um vetor ordenado contendo os elementos de um vetor de entrada, adicionalmente retorna um vetor com os índices dos elementos no vetor de entrada;**
  - **Utiliza o algoritmo “quick sort”;**
- **tipo: usado para definir o tipo de ordenação, no caso de vetores, recomenda-se utilizar sempre o valor ‘g’ (valor padrão), que significa ordenar todos os elementos;**
- **direção : usado para definir a direção de ordenação:**
  - **‘i’: para ordem crescente;**
  - **‘d’: para ordem decrescente (padrão);**

- **Plotando gráficos**

  plot2d(<Vetor X>, <Vetor Y>)

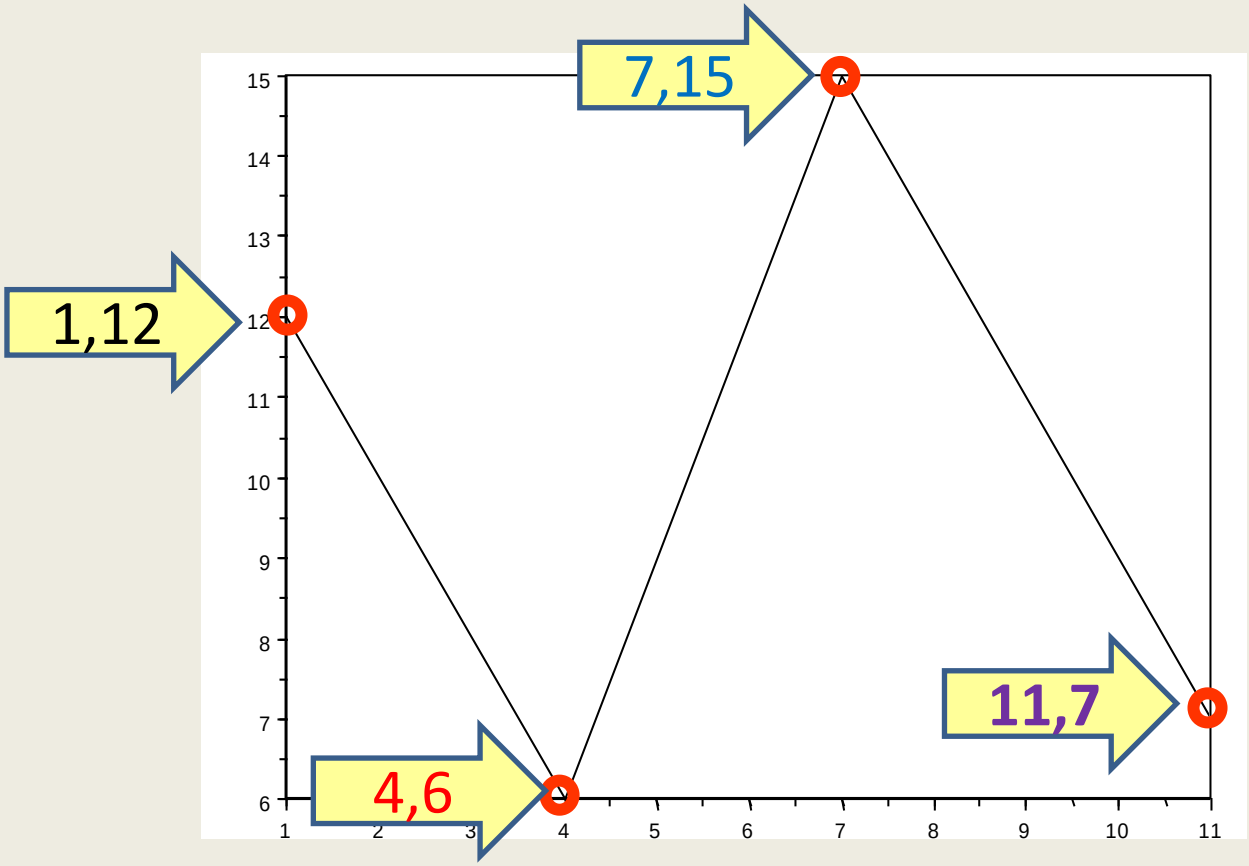
- **Plota um gráfico no plano cartesiano unindo seguimentos de reta formados pelas coordenadas X e Y dos vetores passados como parâmetro:**
  - **<Vetor X> possui as coordenas do eixo X;**
  - **<Vetor Y> possui as coordenadas do eixo Y;**
  - **Os dois vetores devem possuir a mesma dimensão (*n*);**
  - **Cada posição corresponde a uma coordenada: (x(1), y(1)), (x(2), y(2)), ..., (x(*n*), y(*n*)).**

Plotando gráficos

- Exemplo 1:

  --> x = [1 4 7 11]; y = [12 6 15 7];

  →plot2d(x,y)



Plotando gráficos

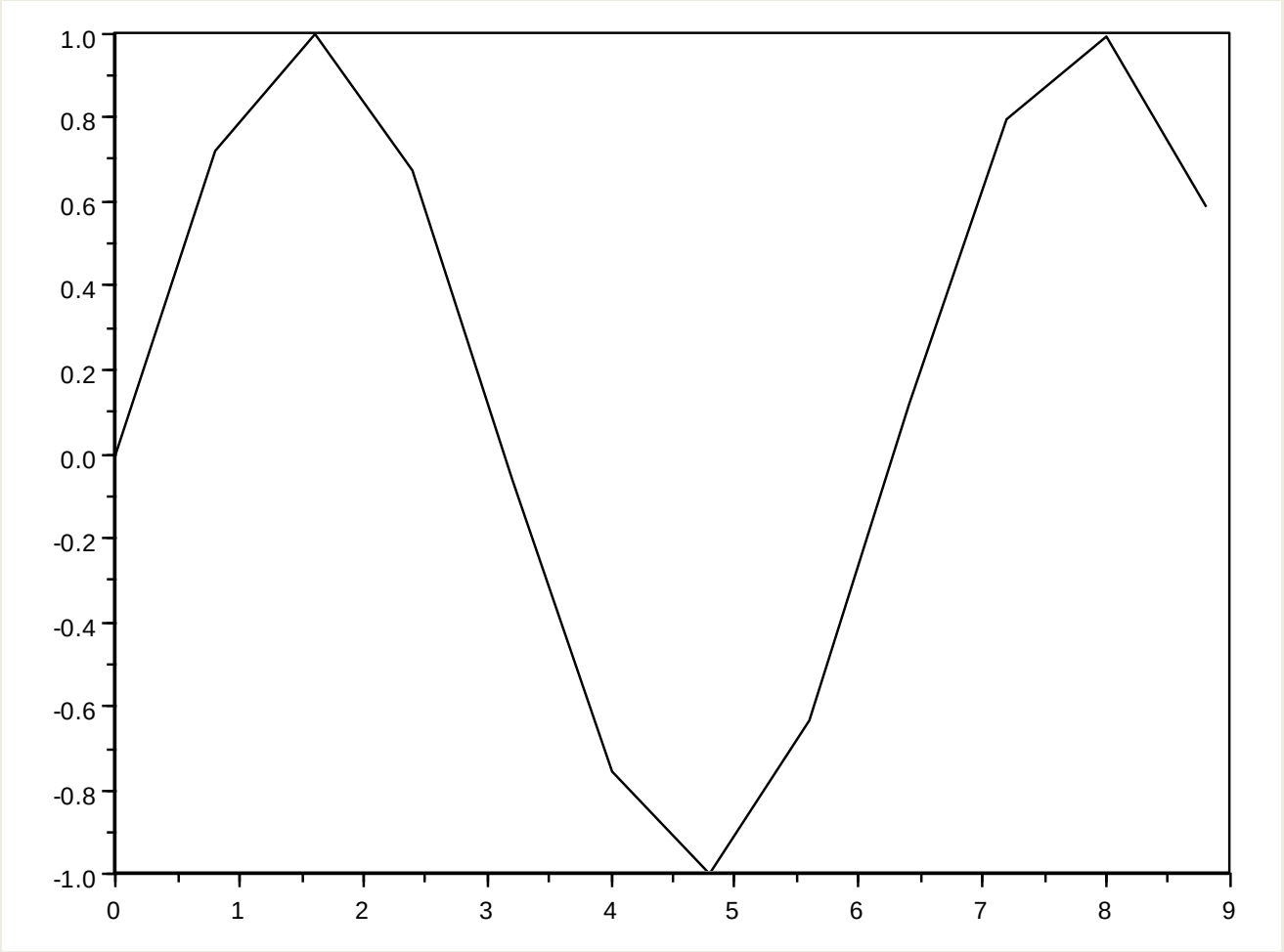
- Exemplo 2:

```
-->x = 0:0.8:3*%pi;
-->y = sin(x);
--> clf();
--> plot2d(x,y)
```

A função seno é aplicada a cada elemento do vetor (x), originando outro vetor (y) com os resultados individuais.

Por padrão, os gráficos plotados são sobrepostos. Para apagar os gráficos plotados anteriormente, utilize a função clf().

Observe no gráfico que o espaçamento de 0.8, a cada coordenada x, originou um gráfico da função seno "ruim".

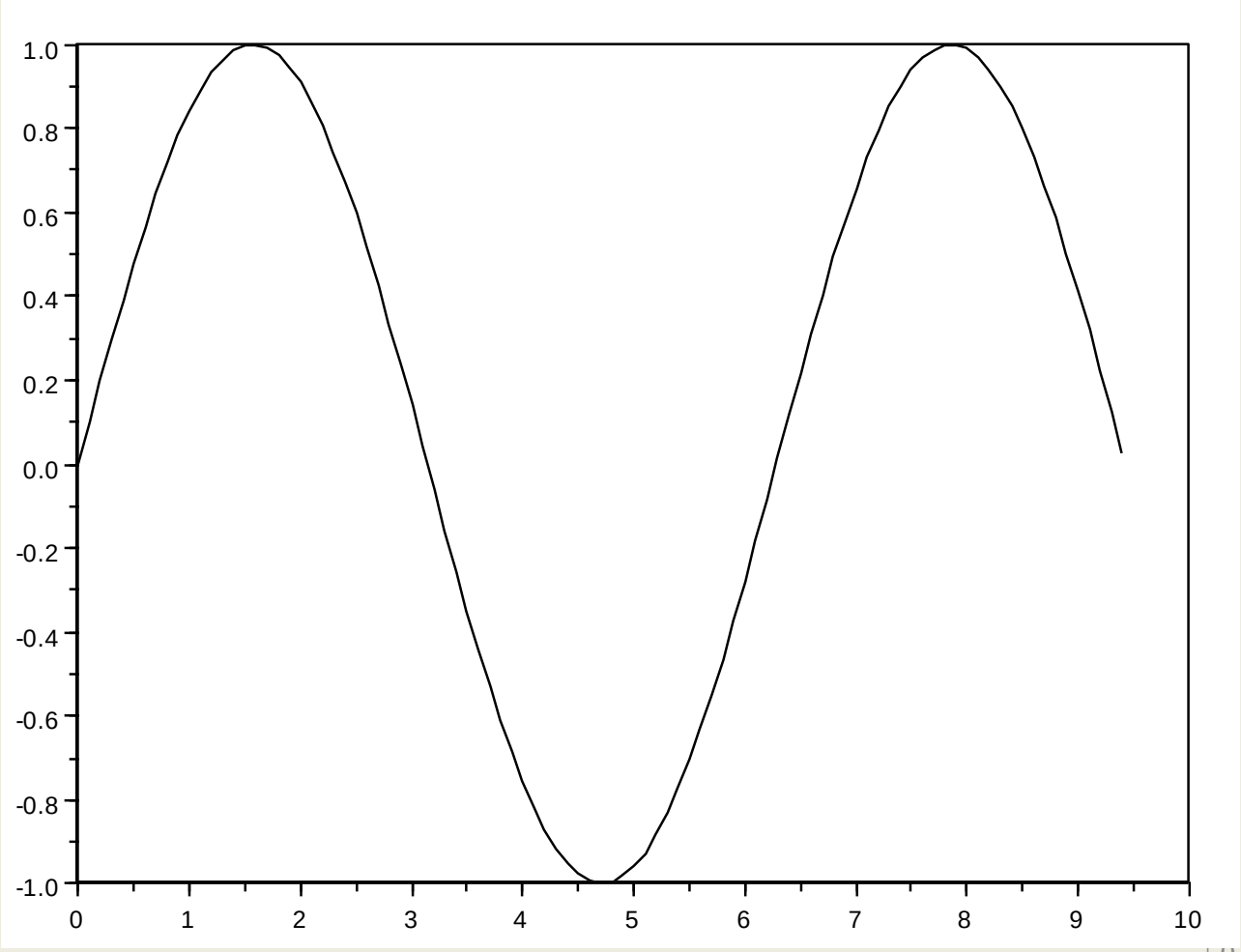


Plotando gráficos

- Exemplo 3:

```
--> x = 0:0.1:3*%pi;
--> y = sin(x);
--> clf();
--> plot2d(x,y)
```

Observe agora que, com um espaçamento de 0.1, o gráfico da função seno originado ficou bem melhor.



Exercícios

Exercícios propostos

1. Faça um programa que preencha um vetor de n elementos através de entradas do usuário. Após a definição dos elementos do vetor, calcule a média dos valores.

2. Faça um programa que preencha dois vetores de 10 elementos através de entradas do usuário. Após a definição dos dois vetores, construa um terceiro vetor onde cada elemento corresponde ao dobro da soma entre os elementos correspondentes dos outros dois vetores. Imprima o conteúdo do vetor calculado.

3. Faça um programa que preencha dois vetores de 10 elementos através de entradas do usuário. Após a definição dos dois vetores, construa um terceiro vetor (20 elementos) onde os elementos de *índice ímpar* recebem os valores do *primeiro vetor* e os elementos de *índice par* recebem os valores do *segundo vetor*. Imprima o conteúdo do vetor calculado.e

Exercícios propostos

4. Escreva um programa que preencha um vetor com entradas do usuário. Considere que o usuário definirá apenas valores numéricos positivos, e que, ao desejar encerrar a definição dos elementos ele digite um valor negativo. Após a entrada de todos os elementos do vetor, calcule e imprima o seu somatório, **sem a utilização da função sum**.

5. Escreva um programa semelhante ao anterior, que retorne e imprima o produtório cumulativo, **sem a utilização da função cumprod**.

6. Escreva um programa semelhante aos anteriores, mas que retorne e imprima um vetor contendo apenas os elementos únicos, **sem a utilização da função unique**. ***Dicas****: Com o vetor preenchido, percorra seus elementos inserindo os elementos únicos em um novo vetor. Um elemento único é aquele que ainda não se encontra no novo vetor. Para descobrir se um elemento já está inserido no novo vetor, utilize a função* ***find***.