Aula prática 2

Variáveis, Atribuição, Entrada e Saída

Resumo

Nesta aula o aluno deverá desenvolver programas simples para resolver problemas de cálculo usando **variáveis** e o **comando de atribuição**, e os **comandos de entrada e saída** de dados para interação com o usuário.

Sumário

1 Variáveis

Variável é um local de armazenamento de dados na memória do computador. Podemos entendê-la como uma *caixinha* onde são colocados os dados processados em um programa.

$$x = 2.5 \qquad x \boxed{2.5}$$

Variáveis são identificadas por um **nome** que consiste em uma sequência de letras, dígitos decimais, e caracteres especiais (`%`, `_`, `#`, `!`, `$`, `?`), devendo começar com uma letra ou com o caracter `%`. Letras maiúsclas e minúsculas são consideradas diferentes. Somente os 24 primeiros caracters do nome de uma variável são significativos.

São nomes de variáveis válidos: `altura`, `peso1`, `quantidade_tomates`, `numeroAlunos`, `%pi`. Não são nomes de variáveis válidos: `10anos`, `_idade`, `media final`.

A operação de **atribuição** é utilizada para armazenar um valor em uma variável. Sua forma básica é:

$$\textit{variável} = \textit{expressão}$$

Quando executado, este comando avalia a expressão e armazena o seu valor na variável. O valor anteriormente armazenado na variável é perdido.

$$x = \underbrace{2 * x + 1}_{2*2.5+1} \qquad x \boxed{\cancel{2.5} \quad 6}$$

Para **acessar** o valor de uma variável basta usar o nome da variável em uma expressão.

2 Entrada de dados

`input(`*`mensagem`*`)`

Exibe *mensagem* (uma string ou cadeia de caracteres) na tela e espera uma entrada pelo teclado, e então retorna o valor numérico da expressão digitada pelo usuário.

`input(`*`mensagem`*`, "string")`

input(*mensagem*, "s")

Exibe *mensagem* (uma string ou cadeia de caracteres) na tela e espera uma entrada pelo teclado, e então retorna a linha digitada pelo usuário como uma string (cadeia de caracteres).

Exemplos:

```
-->input("digite um número: ")
digite um número: 135
 ans  =

    135.

-->input("digite o valor desejado: ")
digite o valor desejado: 2 + 3*4
 ans  =

    14.

-->input("Nome do cliente: ", "string")
Nome do cliente: Manoel da Silva
 ans  =

 Manoel da Silva

-->idade = input("idade do cliente: ")
idade do cliente: 17
 idade  =

    17.
```

3 Saída de dados

disp(x_1, ..., x_n)

Exibe os valores dos objetos $x_1, \ldots, x_n$ na tela, em ordem inversa.

printf(*formato*, a_1, ..., a_n)

Exibe na tela a string *formato*, inserindo os valores das expressões $a_1, \ldots, a_n$ nas posições indicadas pelos formatadores presentes em *formato*.

Alguns formatadores:

formatador	formatação
%s	uma string (cadeia de caracteres)
%i ou %d	um número inteiro em notação decimal
%f	um número real em notação decimal
%e ou %E	um número real em notação científica
%g ou %G	um número real no estilo de f, e ou E, o que for mais adequado
%%	insere um %

Logo após o caracter % de um formatador pode-se informar o tamanho mínimo do campo usado para inserir o valor. Por exemplo %12i insere um número inteiro usando um espaço mínimo de 12 caracteres.

Nos formatadores de números fracionários pode-se informar o número de casas decimais logo após o tamanho do campo. Por exemplo %10.5f insere um número fracionário usando um espaço mínimo de 10 caracteres (contando o ponto decimal), com 5 casas decimais.

Um - antes do tamanho do campo alinha à esquerda. O padrão é alinhar à direita.

Strings podem conter **sequências de escape**, usuais na represesntação de caracteres não gráficos:

`\n` nova linha: quando exibido na tela faz com que o cursor avance para a próxima linha.

`\t` tabulação horizontal: quando exibido na tela faz com que o cursor avance para a próxima parada de tabulador. Normalmente existe uma parada de tabulador a cada 8 colunas na tela.

`\\` o caracter \.

Exemplos:

```
-->printf('uma string: |%s|\n', 'Scilab');
uma string: |Scilab|

-->printf('um inteiro: |%d|\n', 10);
um inteiro: |10|

-->printf('um inteiro: |%4d|\n', 10);
um inteiro: |  10|

-->printf('um inteiro alinhado à esquerda: |%-4d|\n', 10);
um inteiro alinhado à esquerda: |10  |

-->printf('um float: |%d|\n', %pi);
um float: |3|

-->printf('um float em notação científica: |%3.2e|\n', %pi);
um float em notação científica: |3.14e+00|

-->printf('um float: |%3.2g|\n', %pi);
um float: |3.1|
```

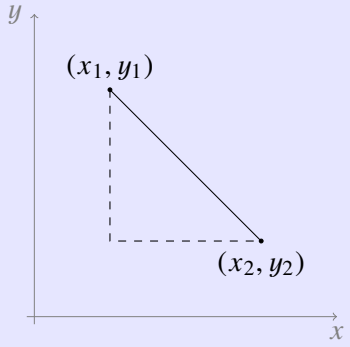
4 Usando Scilab para resolver problemas

Tarefa 1: Distância entre dois pontos no plano

A distância entre dois pontos (x_1, y_1) e (x_2, y_2) em um plano de coordenadas cartesianas é dada pela equação abaixo:

$$d = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$$

Veja também a figura a seguir.



Escreva um programa para calcular a distância entre dois pontos $(x1, y_1)$ e (x_2, y_2) especificados pelo usuário. Utilize boas práticas de programação em seu programa.

Use o seu programa para calcular a distância entre os pontos $(-3,2)$ e $(3,-6)$.

Exemplo de execução da aplicação

```
Cálculo da distância entre dois pontos
--------------------------------------
x1: -3
y1: 2
x2: 3
y2: -6

distância: 10
```

Solução:

```
clc;       // limpa a janela do console
clear;     // remove as variáveis do ambiente

// entrada de dados
printf("Cálculo da distância entre dois pontos\n");
printf("--------------------------------------\n");
x1 = input("x1: ");
y1 = input("y1: ");
x2 = input("x1: ");
y2 = input("y1: ");

// cácuculo
d = sqrt((x1-x2)^2 + (y1-y2)^2);

// saída de dados
printf("\n")
printf("distância: %g\n", d);
```

Tarefa 2: Energia armazenada em uma mola

A força requerida para comprimir uma mola linear é dada pela equação

$$F = kx$$

onde F é a força em N (newton), x é a compressão da mola em m (metro), e k é a constante da mola em N/m.

A energia potencial armazenada na mola comprimida é dada pela equação

$$E = \frac{1}{2}kx^2$$

onde E é a energia em J (joule).

Escreva um programa para calcular a *compressão* e a *energia potencial* armazenada de uma mola, dadas a constante da mola e a força usada para comprimi-la.

Exemplo de execução da aplicação

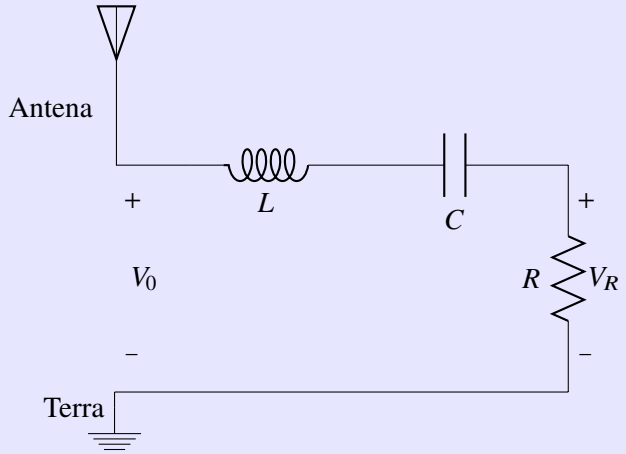
```
Cálculo da energia armazenada em uma mola
-----------------------------------------
constante da mola (N/m): 250
força na mola (N): 30

compressão da mola: 0.120000 m
energia armazenada na mola: 1.800000 J
```

Solução:

```
clc;          // limpa a janela do console
clear;        // remove as variáveis do ambiente

// entrada de dados
printf("Cálculo da energia armazenada em uma mola\n");
printf("-----------------------------------------\n");
k = input("constante da mola (N/m): ");
f = input("força na mola (N): ");

// cálculos
x = f / k;
energia = 0.5 * k * x^2;

// saída de dados
printf("\n")
printf("compressão da mola: %f m\n", x);
printf("energia armazenada na mola: %f J\n", energia);
```

Tarefa 3: Receptor de rádio

Uma versão simplificada da parte frontal de um receptor de rádio AM é apresentada na figura abaixo. Esse receptor é composto por um circuito que contém um resistor R, um capacitor C e um indutor L conectados em série. O circuito é conectado a uma antena externa e aterrado conforme mostra a figura.



O circuito permite que o rádio selecione uma estação específica dentre as que transmitem na faixa AM. Na frequência de resonância do circuito, essencialmente todo o sinal V_0 da antena vai até o resistor, que representa o resto do rádio. Em outras palavras, o rádio recebe seu sinal mais forte na frequência de ressonância. A frequência de ressonância do circuito indutor-capacitor é dada pela equação

$$f_0 = \frac{1}{2\pi\sqrt{LC}}$$

onde L é a indutância em H (henry) e C é a capcitância em F (farad).

Escreva um programa que calcule a frequência de ressonância desse aparelho de rádio, dados valores específicos para L e C (o usuário do programa informa estes dados).

Teste seu programa pelo cálculo da frequência do rádio quando $L = 0,25mH$ e $C = 0,10nF$.

Exemplo de execução da aplicação

```
Cálculo da frequência de ressonância
------------------------------------
digite a indutância em henry: 0.25/1000
digite a capacitância em farad: 0.10/10^9

frequência de ressonância: 1.00658e+06 Hz
```

Solução:

```
clc;          // limpa o console
clear;        // limpa as variáveis do ambiente

// entrada de dados
printf("Cálculo da frequência de ressonância\n");
printf("------------------------------------\n");
L = input("digite a indutância em henry: ");
C = input("digite a capacitância em farad: ");

// cálculo da frequência de ressonância
f0 = 1 / (2 * %pi * sqrt(L * C));

// saída de dados
printf("\n")
printf("frequência de ressonância: %g Hz\n", f0);
```