

Aula Teórica 06

Material Didático Proposto

Conteúdos da Aula

- **Instrução de Repetição**
- **Laços Aninhados**
- **Exercícios**

Instrução de Repetição

- Para permitir que uma operação seja executada repetidas vezes utiliza-se comandos de repetição;
- Uma estrutura deste tipo também é chamada de laço (do inglês loop);
- No Scilab, são definidos dois comandos de repetição:
 1. Laço controlado por contador (for);
 2. Laço controlado logicamente (while).

Em um laço controlado por contador, os comandos (corpo do laço) são repetidos um número predeterminado de vezes.

Já em um laço controlado logicamente, os comandos (corpo do laço) são repetidos enquanto uma expressão lógica for verdadeira.

Denomina-se iteração a repetição de um conjunto de comandos:

- Cada execução do corpo do laço, juntamente com a condição de terminação do laço, é uma iteração.

Instrução de Repetição – Sintaxe para o **for**

O comando **for** pode ser definido da seguinte forma:

```
for variável = <inicial>:<passo>:<final>  
    <conjunto de comandos>  
end
```

- **<conjunto de comandos>** é o conjunto de instruções a serem executadas, é denominado corpo do laço.
- **variável = <inicial>:<passo>:<final>** é a declaração da variável contadora em conjunto com a definição dos valores inicial, final e o passo do laço; ao final de cada iteração a variável será incrementada pelo valor do passo.
- **for** e **end** são palavras reservadas da linguagem.

Elabore um programa que gere e imprima os números Naturais até um dado número k:

```
k = input("Digite o valor limite");  
for nat = 0:k  
    printf("%g    ", nat)  
end
```

Elabore um programa para calcular a soma dos números naturais até um dado número k :

```
k = input("Digite o valor limite");  
soma = 0;  
for nat = 0:k    // nat = 1:k  
    soma = soma + nat;  
end  
printf("A soma dos naturais até %g é  
      igual a %g", k, soma);
```


Instrução de Repetição - Sintaxe para o **while**

- O comando **while** é um laço controlado logicamente;
- O laço **while** é definido da seguinte forma:

```
while <expressão lógica>  
    <conjunto de comandos>  
end
```

- **<conjunto de comandos>** é o conjunto de instruções a serem executadas, é denominado corpo do laço;
- **<expressão lógica>** é a expressão que define quando os comandos deverão ser executados;
- **while** e **end** são palavras reservadas da linguagem.

Elabore um programa que gere e imprima os números Naturais até um dado número k:

```
k = input("Digite o valor limite");  
nat = 0; // inicialização fora do laço  
while nat <= k  
    printf("%g    ", nat)  
    nat = nat + 1; // incremento dentro do laço  
end
```

Elabore um programa para calcular a soma dos números naturais até um dado número k:

```
k = input("Digite o valor limite");  
nat = 0;  
soma = 0;  
while nat <= k  
    soma = soma + nat;  
    nat = nat + 1;  
end  
printf("A soma dos naturais até %g é igual a  
%g", k, soma);
```

Ler uma sequência de números positivos e calcular a sua média. O fim dos dados será indicado pelo número -1 (flag), que não deve ser considerado pertencente ao conjunto.

```
soma = 0;
cont = 0;
num = input("Digite o primeiro número");
while num <> -1
    soma = soma + num;
    cont = cont + 1;
    num = input("Digite outro número");
end
media = soma/cont;
printf("Média dos números= %g",media);
```

**Pode-se codificar o exemplo 3
utilizando o comando for?**

- **Elabore um programa que calcule e imprima o valor de S:**

$$S = \frac{1}{1} + \frac{3}{2} + \frac{5}{3} + \frac{7}{4} + \dots + \frac{99}{50}$$

- **Dica, encontre o padrão entre o numerador e o denominador:**

$$\text{Numerador} = 2 * \text{Denominador} - 1$$

Instrução de Repetição – Exemplo 1

```
s = 0;
for d = 1:50
    s = s + (2 * d - 1) / d;
end
printf("Valor de S = %g\n", s);
// ----- ou -----
s = 0; d = 1; // inicialização fora do laço
while d <= 50
    s = s + (2 * d - 1) / d;
    d = d + 1; // última instrução
end
printf("Valor de S = %g\n", s);
```

- **Agora vamos mudar o problema anterior para:**

$$S = \frac{1}{1} + \frac{5}{3} + \dots + \frac{97}{49}$$

- **O padrão entre o numerador e o denominador é o mesmo, mas agora o denominador varia de forma diferente.**

Instrução de Repetição – Exemplo 2

```

s = 0;
for d = 1:50
    if (modulo(d, 2) == 1) then
        s = s + (2 * d - 1) / d;
    end
end
printf("Valor de S = %g\n", s);
// ----- ou -----
s = 0; d = 1; // inicialização fora do laço
while d <= 50
    if (modulo(d, 2) == 1) then
        s = s + (2 * d - 1) / d;
    end
    d = d + 1; // última instrução
end
printf("Valor de S = %g\n", s);
  
```

Instrução de Repetição – Exemplo 2 – Outra Solução

```

s = 0;
for d = 1:2:50
    s = s + (2 * d - 1) / d;
end
printf("Valor de S = %g\n", s);
// ----- ou -----
s = 0; d = 1; // inicialização fora do laço
while d <= 50
    s = s + (2 * d - 1) / d;
    d = d + 2; // última instrução
               // incremento de 2
end
printf("Valor de S = %g\n", s);
  
```

Os valores assumidos pela variável contadora não precisam ser inteiros, por exemplo:

```
for x = 0 : 0.3 : 0.7  
    printf("\nX = %g", x) ;  
end
```

Este programa resultará em:

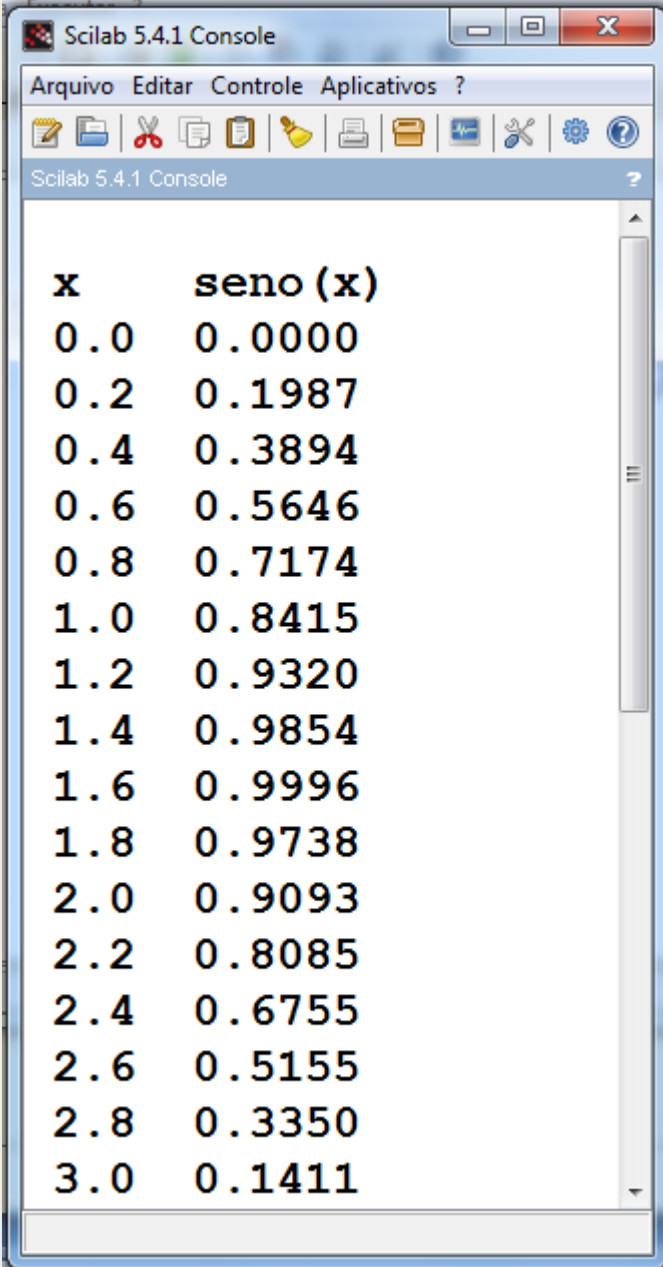
```
X = 0  
X = 0.3  
X = 0.6
```

Tabela de senos

Elabore um programa que calcule e Imprima uma tabela de senos, conforme a tabela apresentada.

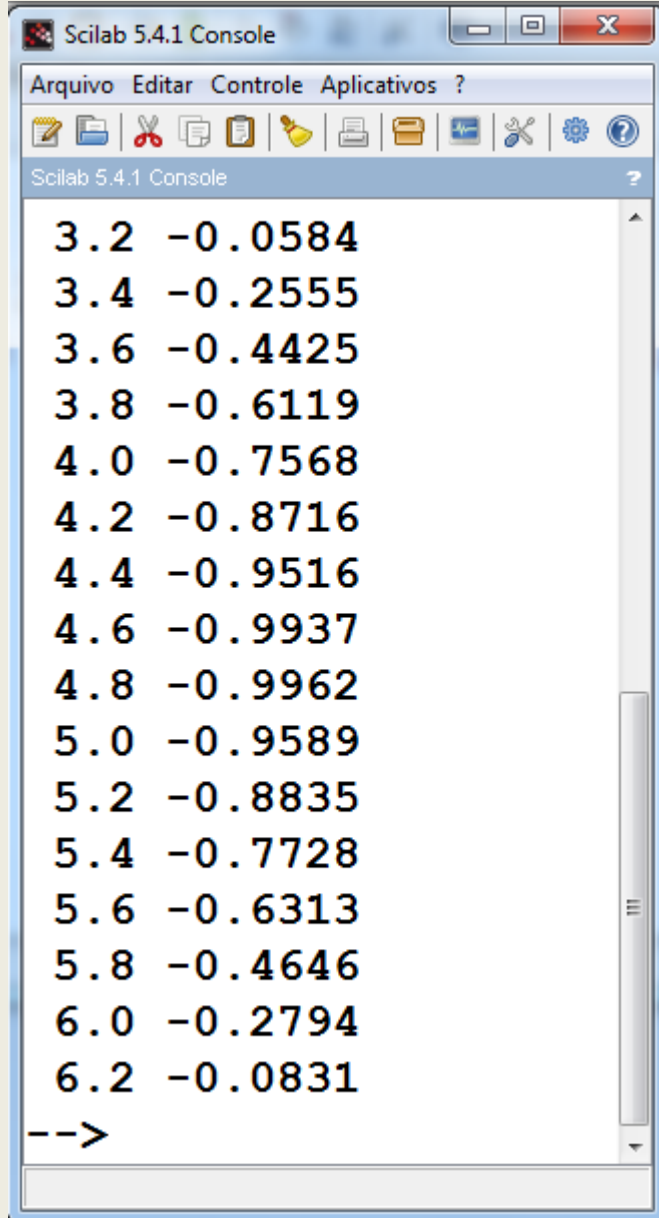
O critério de parada é $x = 2\pi$.

Instrução de Repetição – Exemplo 4



Scilab 5.4.1 Console

x	seno (x)
0.0	0.0000
0.2	0.1987
0.4	0.3894
0.6	0.5646
0.8	0.7174
1.0	0.8415
1.2	0.9320
1.4	0.9854
1.6	0.9996
1.8	0.9738
2.0	0.9093
2.2	0.8085
2.4	0.6755
2.6	0.5155
2.8	0.3350
3.0	0.1411



Scilab 5.4.1 Console

3.2	-0.0584
3.4	-0.2555
3.6	-0.4425
3.8	-0.6119
4.0	-0.7568
4.2	-0.8716
4.4	-0.9516
4.6	-0.9937
4.8	-0.9962
5.0	-0.9589
5.2	-0.8835
5.4	-0.7728
5.6	-0.6313
5.8	-0.4646
6.0	-0.2794
6.2	-0.0831

-->

```
clc; clear;
printf("\n x seno(x) ");
for x = 0 : 0.2 : 2 * %pi
    printf("\n %3.1f %7.4f", x, sin(x));
end
// ----- ou -----
x = 0; // inicialização fora do laço
while x <= 2 * %pi
    printf("\n %3.1f %7.4f", x, sin(x));
    x = x + 0.2; // última instrução
                // incremento de 0.2
end
```

- Observações:
 - Perceba que os valores da variável contadora podem ser definidos por expressões ($2 * \%pi$);
 - É possível formatar a saída dos valores no *printf* para obter uma tabela:
 - Neste exemplo:
 - **$\%3.1f$** indica um valor *float* (número fracionário) com um total de 3 caracteres, com 1 casa decimal;
 - **$\%7.4f$** indica um valor *float* com um total de 7 caracteres, com quatro casas decimais.

- **Agora vamos mudar novamente o problema do somatório:**

$$S = \frac{97}{49} + \dots + \frac{5}{3} + \frac{1}{1}$$

- **Agora houve uma inversão na sequência dos termos, o que fazer?**

Instrução de Repetição – Exemplo 5

```

s = 0;
for d = 49:-2:1 // decremento de 2 em d
    s = s + (2 * d - 1) / d;
end
printf("Valor de S = %g\n", s);
// ----- ou -----
s = 0; d = 49; // inicialização fora do laço
while d >= 1
    s = s + (2 * d - 1) / d;
    d = d - 2; // última instrução
               // decremento de 2
end
printf("Valor de S = %g\n", s);
  
```

- Quando usar o `for` ou o `while`?
- No exemplo 5 o uso do `for` é mais adequado.
- Mas, existem situações em que o comando `while` é mais adequado, ou, em que não é possível utilizar o comando `for`:
 - a) o número de repetições do laço é desconhecido;
 - b) são necessários testes lógicos que não usam somente o operador `<=` (usam os demais operadores relacionais e lógicos).
- A seguir, dois exemplos.

Instrução de Repetição - for ou while ?

Validação de Dados de Entrada

```
x = input("ENTRE COM O VALOR DE X : ");  
while (x == 0)  
    printf("X NÃO PODE SER NULO!\n");  
    x = input("ENTRE COM O VALOR DE X : ");  
end
```

Observações:

- Não se pode prever quantas vezes o usuário entrará com um valor incorreto (nulo);
- Não é possível utilizar o comando `for` neste caso.

Instrução de Repetição - for ou while ?

No Algoritmo de Euclides para o cálculo do Máximo Divisor Comum, não podemos prever os valores da variável contadora para a utilização do comando for:

```
x = input("x = ") ;  
y = input("y = ") ;  
xa = x;  
ya = y;  
while y <> 0  
    r = modulo(x, y) ;  
    x = y;  
    y = r;  
end  
printf("mdc(%d,%d) = %d", xa, ya, x)
```

Observações:

a) use o for sempre que possível, ele será mais seguro e eficiente;

b) cuidado ao utilizar o while, pois será possível que o laço

nunca termine (laço infinito), veja 2 exemplos:

```
x = 0;  
while x <= 10  
    printf("\nx = %g", x)  
end
```

O valor de x nunca será alterado. Logo, teremos um laço infinito.

```
x = 0;  
while x <= 10  
    printf("\nx = %g", x)  
    x = x - 0.2;  
end
```

O valor de x é iniciado com zero, sendo depois decrementado. O valor de x sempre será negativo. O programa nunca deixará o laço infinito.

Em algumas situações desejamos repetir um programa que acabamos de executar.

Então vamos até o Scinotes e executamos novamente o programa.

É possível executar quantas vezes quisermos um determinado programa, permanecendo no console do Scilab.

Basta acrescentarmos ao código do nosso programa os códigos especificados no exemplo a seguir.

```
repetir= %t; // supõe que o usuário  
           // sempre repetirá a execução
```

```
while repetir
```

```
// Início do seu programa  
// Comandos do seu programa  
// Fim do seu programa
```

```
// Decisão sobre a repetição do programa  
decisao = input("Repetir? (s/n)", "string");  
repetir = decisao == 's' | decisao == 'S';  
end  
printf ("Término do programa.\n");
```

Laços Aninhados

Veja o seguinte desenho:

- **Repetição 1: temos oito repetições de linhas com o mesmo caractere ‘*’.**
- **Repetição 2: temos em cada linha, a repetição de n caracteres, sendo $1 \leq n \leq 8$. Assim, na linha 1 temos $n=1$, na linha 2 temos $n=2$, até a linha 8, onde temos $n=8$.**
- **Para obter o desenho temos a repetição 2 realizada dentro da repetição 1.**

*** Fazendo a Repetição 2:**

********* Agora, faremos a repetição do código acima 8 vezes, uma para cada linha.

```
// imprime uma linha com n '*'  
for j=1:n  
    printf("*");  
end
```

* **Fazendo a Repetição 1:**

**

*** **for n=1:8**

**** **// imprime uma linha com n '*'**

***** **for j=1:n**

***** **printf("*");**

***** **end**

***** **// muda a linha**

printf("\n");

end

Quando temos um laço dentro de outro temos laços aninhados:

```
for i = 1:n
    • • •
    for j = 1:m
        • • •
    end
    • • •
end
```

1. A execução começa no laço externo (azul);
2. Quando chegamos ao laço interno (vermelho), suas m interações são realizadas (j assume os valores de 1 a m);
3. Ao sair do laço mais interno, incrementa-se o contador do laço externo.
4. Se ocorrer a repetição do bloco do laço externo, o laço interno será executado novamente.

Faça um programa que imprima a tabela da tabuada de multiplicação:

	1	2	3	4	5	6	7	8	9	10
1	1	2	3	4	5	6	7	8	9	10
2	2	4	6	8	10	12	14	16	18	20
3	3	6	9	12	15	18	21	24	27	30
4	4	8	12	16	20	24	28	32	36	40
5	5	10	15	20	25	30	35	40	45	50
6	6	12	18	24	30	36	42	48	54	60
7	7	14	21	28	35	42	49	56	63	70
8	8	16	24	32	40	48	56	64	72	80
9	9	18	27	36	45	54	63	72	81	90
10	10	20	30	40	50	60	70	80	90	100

Exercício – Tabuada de Multiplicação

```
clc;

printf("\nTabuada de Multiplicação:\n\n");
printf("    |  1    2    3    4    5    6    7    8    9   10\n");
printf("-----\n");

for linha = 1 : 10
    printf("%2.0f |", linha);
    for coluna = 1 : 10
        printf("%3.0f ", linha * coluna);
    end
    printf("\n");
end
```

Exercícios

Um aluno foi ao supermercado e gastou X reais com as compras da semana.

Escreva um programa que tenha como entrada o valor X da compra. O programa deve determinar quantas notas de 50, de 10 e de 1 real são suficientes para o pagamento da compra.

Obs: O programa só deverá imprimir a quantidade de notas que forem maiores do que zero; e o valor da compra é um número inteiro.

Exercício – Compras no Supermercado

```
clc;  
ValorCompra = input("VALOR DA COMPRA: ");  
N50 = 0; N10 = 0;  
while (ValorCompra >= 50)  
    ValorCompra = ValorCompra - 50;  
    N50 = N50 + 1;  
end  
while (ValorCompra >= 10)  
    ValorCompra = ValorCompra - 10;  
    N10 = N10 + 1;  
end
```

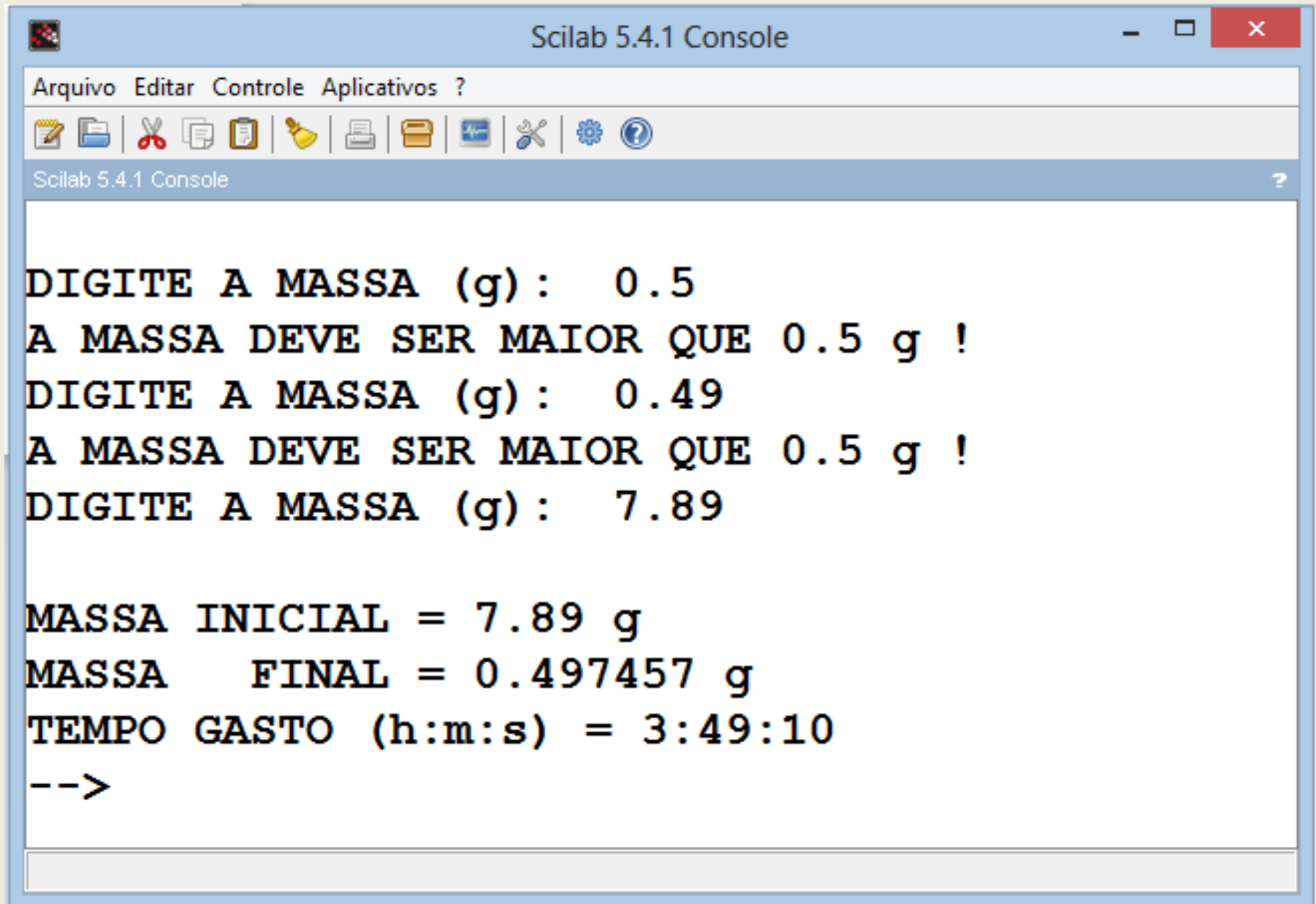
```
printf("O VALOR DA COMPRA SERÁ PAGO COM:\n") ;  
if (N50 > 0) then  
    printf("%g NOTA(S) DE CINQUENTA\n", N50) ;  
end  
if (N10 > 0) then  
    printf("%g NOTA(S) DE DEZ\n", N10) ;  
end  
if (ValorCompra > 0) then  
    printf("%g NOTA(S) DE UM\n", ValorCompra) ;  
end
```

Um determinado material radioativo perde 1% de sua massa a cada 50 segundos.

Codifique um programa Scilab que leia a massa inicial em gramas. A massa fornecida deve ser maior que 0,5 gramas e o programa repete a entrada até que uma massa com esta especificação seja fornecida.

O programa calcula e imprime o tempo necessário para que a massa se torne menor que 0,5 gramas.

Exercício - Perda de Massa



The image shows a screenshot of the Scilab 5.4.1 Console window. The window has a title bar with the text 'Scilab 5.4.1 Console' and standard window controls (minimize, maximize, close). Below the title bar is a menu bar with the items 'Arquivo', 'Editar', 'Controle', 'Aplicativos', and '?'. Underneath the menu bar is a toolbar containing various icons for file operations (like open, save, print) and editing (like copy, paste, undo, redo). The main area of the window is a text editor displaying the following text in a monospaced font:

```
DIGITE A MASSA (g): 0.5
A MASSA DEVE SER MAIOR QUE 0.5 g !
DIGITE A MASSA (g): 0.49
A MASSA DEVE SER MAIOR QUE 0.5 g !
DIGITE A MASSA (g): 7.89

MASSA INICIAL = 7.89 g
MASSA FINAL = 0.497457 g
TEMPO GASTO (h:m:s) = 3:49:10
-->
```

At the bottom of the window, there is a small, empty rectangular box, likely for command input or output.

Exercício - Perda de Massa

```
1 clc; clear;
2 massaInicial = input("DIGITE A MASSA (g): ");
3 while massaInicial <= 0.5
4     printf("A MASSA DEVE SER MAIOR QUE 0.5 g !")
5     massaInicial = input("DIGITE A MASSA (g): ");
6 end
7 nroPerda = 0;
8 massaFinal = massaInicial;
9 while massaFinal > 0.5
10     massaFinal = massaFinal * 0.99;
11     nroPerda = nroPerda + 1;
12 end
13 tempo = nroPerda * 50;
14 segundos = modulo(tempo, 60);
15 minutos = modulo(int(tempo / 60), 60);
16 horas = int(tempo / 3600);
```

```
17 printf("\nMASSA INICIAL = %g.g", massaInicial);  
18 printf("\nMASSA . . . FINAL = %g.g", massaFinal);  
19 printf("\nTEMPO GASTO (h:m:s) = %g:%g:%g", . . .  
20 . . . . . horas, minutos, segundos);
```

Fazer um algoritmo para calcular a raiz quadrada (x) de um número positivo (y), usando o roteiro abaixo, baseado no método de aproximações sucessivas de Newton:

1) a primeira aproximação para a raiz quadrada de y é: $x_1 = y / 2$

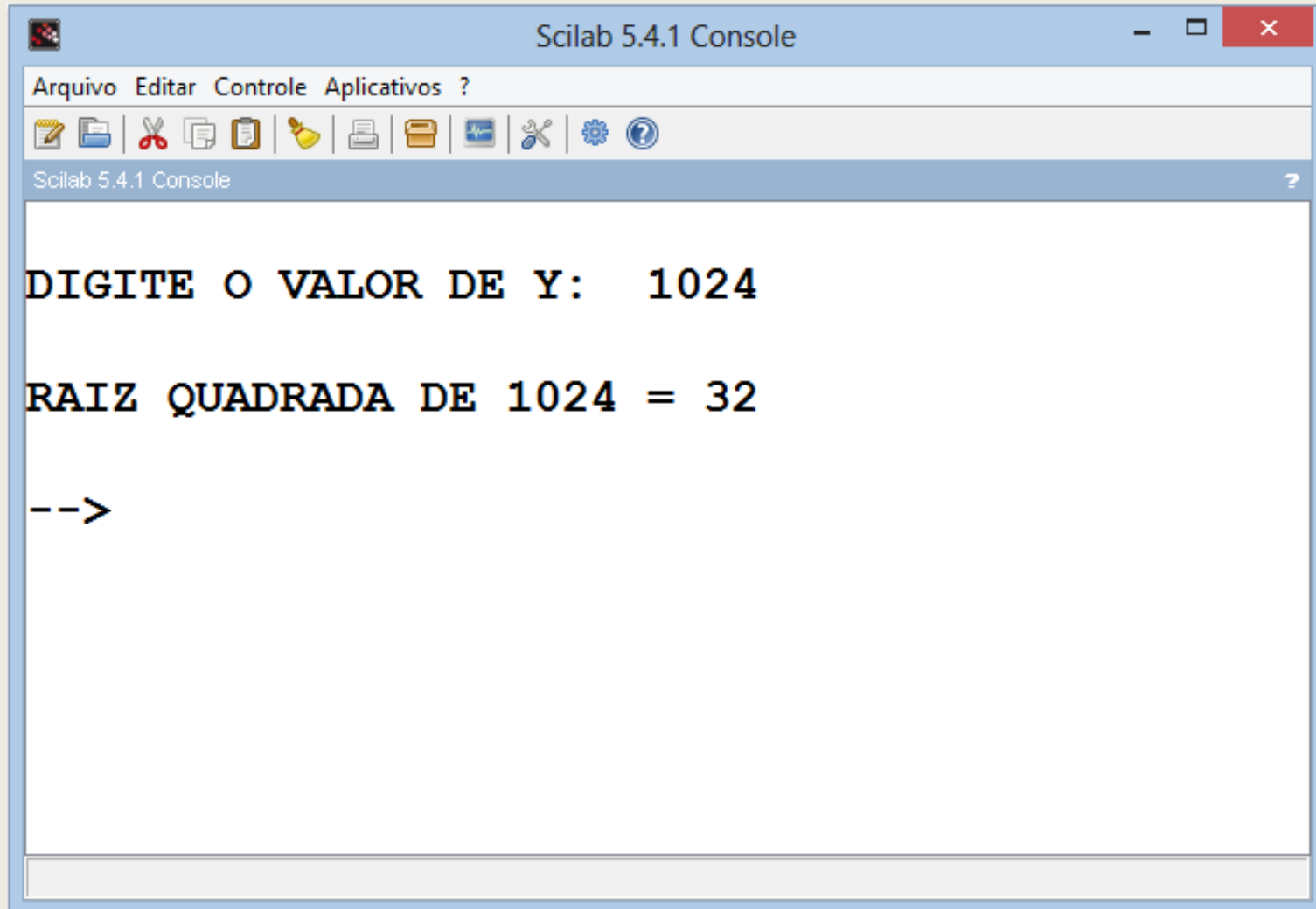
2) as sucessivas aproximações serão:

$$x_{n+1} = \frac{x_n^2 + y}{2x_n}$$

3) O laço dos cálculos das aproximações deverá terminar quando:

$$| x_i - x_{i-1} | < 0.0001$$

Exercício – Raiz Quadrada



The image shows a screenshot of the Scilab 5.4.1 Console window. The window has a title bar with the text "Scilab 5.4.1 Console" and standard window controls (minimize, maximize, close). Below the title bar is a menu bar with the items "Arquivo", "Editar", "Controle", "Aplicativos", and "?". Under the menu bar is a toolbar containing various icons for file operations (new, open, save, print, etc.) and editing (copy, paste, undo, redo, etc.). The main area of the window is a text editor with a white background and a blue border. It contains the following text: "DIGITE O VALOR DE Y: 1024", "RAIZ QUADRADA DE 1024 = 32", and "-->". The text is in a monospaced font. At the bottom of the window is a status bar.

```
DIGITE O VALOR DE Y: 1024  
RAIZ QUADRADA DE 1024 = 32  
-->
```


Exercício – Raiz Quadrada

A06_RaizNewton.sce

```
1 clc; clear;
2 y = input("DIGITE O VALOR DE Y: ");
3 x = y / 2;
4 xAnterior = 0;
5 while (abs(x - xAnterior) > 0.0001)
6     xAnterior = x;
7     x = (x*x + y) / (2*x);
8 end
9 printf("\nRAIZ QUADRADA DE %g = %g\n", y, x);
10
```

Implementar um algoritmo para calcular o seno(x).

O valor de x deverá ser digitado em radianos.

O valor do seno de x será calculado pela soma dos 100 primeiros termos da série a seguir:

$$\text{Seno}(x) = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \dots$$

Exercício – Série do Seno

A06_Seno.sce

```
1 clc; clear;
2 xGrau = input("DIGITE O VALOR DO ÂNGULO EM GRAUS: ");
3 x = xGrau * %pi / 180; // converte para radianos
4 seno = x; sinal = 1; fat = 1; num = x;
5 for i=2:100
6     sinal = sinal * -1;
7     num = num * x * x;
8     fat = fat * (2*i-1) * (2*i-2);
9     seno = seno + sinal * num / fat;
10 end
11 printf("sen(%g) = %g", xGrau, seno);
12
```