4. CAPÍTULO 4 – COMANDOS DE SELEÇÃO

TERMOS CHAVE		CONTEÚDO
comandos de seleção	verificação de erros	
comandos de desvio	comandos aninhados	
condição	if-else em cascata	
ação	funções "is"	
variável temporária		

Nos *scripts* e funções que vimos até agora, todas as instruções (comandos) foram executadas em sequência. Isso nem sempre é desejável e, neste capítulo, veremos como fazer escolhas sobre se as instruções são executadas ou não e como escolher entre as instruções. As instruções que realizam isso são chamadas de instruções (ou comandos) de seleção ou desvios.

O software MATLAB® tem duas instruções básicas que nos permitem fazer escolhas: a instrução **if** e a instrução **switch**. A instrução **if** tem as cláusulas opcionais **else** e **elseif** para desvios. A instrução **if** usa expressões que são logicamente **true** (***verdadeiras***) ou **false** (***falsas***). Essas expressões usam operadores relacionais e lógicos. O MATLAB também possui uma função **menu** que apresenta opções para o usuário; isso será apresentado no final deste capítulo.

4.1 A INTRUÇÃO IF

A instrução **if** escolhe se outra instrução, ou grupo de instruções, é executada ou não. A forma geral da instrução **if** é:

```
if condição
    ação
end
```

Uma condição é uma expressão relacional que é conceitual ou logicamente ***verdadeira*** ou ***falsa***. A ação é uma instrução ou um grupo de instruções que serão executadas se a condição for verdadeira. Quando a instrução **if** é executada, primeiro a condição é avaliada. Se o valor da condição for **verdadeiro**, a ação será executada; E se não for, a ação não será executada. A ação pode conter qualquer número de instruções até a palavra reservada **end**; a ação é naturalmente delimitada pelas palavras reservadas **if** e **end** (Observe que isso é diferente do **end** que é usado como um índice em um vetor ou matriz). A ação geralmente é recuada (deslocada) para torná-la mais fácil de ver.

Por exemplo, a seguinte instrução **if** verifica se o valor de uma variável é negativa. Se for, o valor é alterado para zero; caso contrário, nada muda.

```
if num < 0
    num = 0
end
```

Instruções **if** podem ser inseridas na Janela de Comandos, embora elas geralmente fazem mais sentido em *scripts* ou funções. Na Janela de Comandos, a linha **if** poderia ser inserida, seguida da ***tecla Enter***, da ação, da ***tecla Enter*** e, finalmente, end e ***Enter***. Os resultados seguirão imediatamente. Por exemplo, a instrução **if** anterior é mostrada aqui duas vezes.

```
>> num = -4;
>> if num < 0
```

```
        num = 0
    end
num =
    0
>> num = 5;
>> if num < 0
        num = 0
    end
>>
```

Observe que a saída da atribuição não é suprimida, portanto, o resultado da ação será mostrada se a ação for executada. Na primeira vez, o valor do variável é negativa, então a ação é executada e a variável é modificada, mas, no segundo caso, a variável é positiva, então a ação é ignorada.

Isso pode ser usado, por exemplo, para garantir que a função de raiz quadrada não seja usada com um número negativo. O *script* a seguir solicita ao usuário um número e imprime sua raiz quadrada. Se o usuário digitar um número negativo, a instrução **if** a muda para zero antes de calcular a raiz quadrada.

exemplosqrtif.m

```
% Solicita ao usuário um número e imprime sua raiz quadrada
num = input('Por favor, digite um número: ');
% Se o usuário inseriu um número negativo, altere-o
if num < 0
    num = 0;
end
fprintf ('A raiz quadrada de %.1f é %.1f\n', num, sqrt(num))
```

Aqui estão dois exemplos de execução deste *script*:

```
>> exemplosqrtif
Por favor, digite um número: -4.2
A raiz quadrada de 0.0 é 0.0

>> exemplosqrtif
Por favor, digite um número: 1.44
O 'A raiz quadrada de 1.4 é 1.2
```

Observe que no *script* a saída da instrução de atribuição é suprimida. Nesse caso, a ação da instrução **if** era uma única instrução de atribuição. A ação pode ser qualquer número de instruções válidas. Por exemplo, podemos desejar imprimir uma mensagem para o usuário para dizer que o número digitado estava sendo alterado. Além disso, em vez de alterá-lo para zero, usaremos o valor absoluto do número negativo inserido pelo usuário.

exemplosqrtif2.m

```
% Solicita ao usuário um número e imprime sua raiz quadrada

num = input ('Por favor, digite um número: ');

% Se o usuário digitou um número negativo, informe
% ao usuário e altere-o
if num < 0
    disp('OK, vamos usar o valor absoluto')
    num = abs(num);
end
fprintf('A raiz quadrada de %.1f é %.1f\n', num, sqrt(num))
```

```
>> exemplosqrtif2
Por favor, digite um número: -25
OK, vamos usar o valor absoluto
A raiz quadrada de 25.0 é 5.0
```

PRÁTICA 4.1

Escreva uma instrução **if** que imprima "Ei, você recebe horas extras!" Se o valor da variável $horas$ for maior que 40. Teste a instrução **if** para valores de horas menores, iguais e maiores que 40. Será mais fácil fazer isso na janela de comando ou em um *script*?

PERGUNTA RÁPIDA!

Suponha que queremos criar um vetor de valores inteiros crescentes de $meumin$ para $meumax$. Vamos escrever uma função $criavet$ que recebe dois argumentos de entrada, $meumin$ e $meumax$, e retorna um vetor com valores de $meumin$ para $meumax$ em passos de um. Primeiro, nos certificamos de que o valor de $meumin$ seja menor que o valor de $meumax$. Caso contrário, precisaríamos trocar seus valores antes de criar o vetor. Como poderíamos conseguir isso?

Resposta

Para trocar valores, é necessária uma terceira variável – uma variável temporária. Por exemplo, digamos que temos duas variáveis, a e b, armazenando os valores:

```
a = 3;
b = 5;
```

Para trocar valores, não podemos simplesmente atribuir o valor de b para a, como segue:

```
a = b;
```

Se isso foi feito, então o valor de a (o 3) está perdido! Em vez disso, precisamos atribuir o valor do primeiro para uma variável temporária para que o valor não seja perdido. O algoritmo seria:

- atribuir o valor de a para $temp$

- atribuir o valor de b para a
- atribui o valor de $temp$ para b.

```
>> temp = a;
>> a = b
a =
    5
>> b = temp
b =
    3
```

Agora, para a função. Uma instrução **if** é usada para determinar se a troca é necessária ou não.

criaevet.m

```
function vetsaida = criaevet(meumin, meumax)
% criaevet cria um vetor que se repete de um
% do mínimo especificado para um máximo
% Formato da chamada: criaevet(mínimo, máximo)
% Retorna um vetor

% Se o "mínimo" não for menor que o "máximo",
% troca os valores usando uma variável temporária
if meumin > meumax
    temp = meumin;
    meumin = meumax;
    meumax = temp;
end

% Use o operador de dois pontos para criar o vetor
vetsaida = meumin:meumax;
end
```

Exemplos de chamar a função são:

```
>> createvec (4,6)
ans =
   4   5   6
>> createvec (7,3)
ans =
   3   4   5   6   7
```

4.1.1 Representando Verdadeiro e Falso Lógicos

Foi afirmado que as expressões conceitualmente verdadeiras têm o ***valor lógico*** de 1 e as expressões que são conceitualmente falsas têm o ***valor lógico*** de 0. Representar os conceitos ***verdadeiro lógico*** e ***falso*** no MATLAB é um pouco diferente: o conceito de falso é representado pelo valor de 0, mas o conceito de verdadeiro pode ser representado por qualquer valor diferente de zero (não apenas 1). Isso pode levar a algumas expressões ***lógicas*** estranhas. Por exemplo:

```
>> all(1:3)
ans =
   1
```

Além disso, considere a seguinte instrução:

```
>> if 5
       disp('Sim, isso é verdade!')
   end
Sim, isso é verdade!
```

Como 5 é um valor diferente de zero, a condição é verdadeira. Portanto, quando esta expressão lógica é avaliada, ela será verdadeira, então a função **disp** será executada e "Sim, isso é verdade" será exibido. Claro, isso é uma instrução bem bizarra e que esperamos que nunca seja encontrada!

No entanto, um simples erro em uma expressão pode levar a um resultado semelhante. Por exemplo, digamos que o usuário seja solicitado a escolher "S" ou "N" para uma pergunta sim/não.

```
letra = input('Escolha (S/N):', 's');
```

Em um *script*, poderíamos querer executar uma ação específica se o usuário respondesse com "S". A maioria dos scripts permite que o usuário insira letras minúsculas ou maiúsculas; por exemplo, "s" ou "S" para indicar "sim". A expressão adequada que retornaria verdadeiro se o valor da letra fosse "y" ou "Y" seria

```
letra == 's' || letra == 'S'
```

No entanto, se por engano isso foi escrito como:

```
letra == 's' || 'S'   % Nota: incorreto!!
```

esta expressão seria SEMPRE verdadeira, independentemente do valor da letra variável. Isso ocorre porque 'S' é um valor diferente de zero, portanto, é uma expressão verdadeira. A primeira parte da expressão pode ser falsa, mas como a segunda expressão é verdadeira, toda a expressão seria verdadeira, independentemente do valor da letra variável.

4.2 A INSTRUÇÃO IF-ELSE

A instrução **if** escolhe se uma ação é executada ou não. Escolher entre duas ações, ou escolher entre várias ações, é realizado usando as instruções **if-else**, **if-else** aninhadas e **switch**.

A instrução **if-else** é usada para escolher entre duas instruções ou conjuntos de instruções. A forma geral é:

```
if condição
    ação1
else
    ação2
end
```

Primeiro, a condição é avaliada. Se for ***verdadeira***, então o conjunto de instruções designadas como "ação1" é executado e esse é o final da instrução **if-else**. Se, em vez disso, a condição for ***falsa***, o segundo conjunto de instruções designadas como "ação2" será executado e esse será o final da instrução **if-else**. O primeiro conjunto de instruções ("ação1") é chamado de ação da cláusula **if**; é o que será executado se a expressão for ***verdadeira***. O segundo conjunto de instruções ("ação2") é chamado de ação da cláusula **else**; é o que será executado se a expressão for ***falsa***. Uma dessas ações, e apenas uma, será executada – a qual depende do valor da condição.

Por exemplo, para determinar e imprimir se um número aleatório no intervalo de 0 a 1 é menor que 0.5, uma instrução **if-else** poderia ser usada:

```
if rand < 0.5
    disp('Foi menor que 0.5!')
else
    disp('Não foi menor que 0.5!')
end
```

PRÁTICA 4.2

Escreva um *script imprimeseno* que:

- solicitará ao usuário um ângulo
- solicitará ao usuário (r)adianos ou (g)raus, com radianos como padrão
- se o usuário digitar 'g', a função **sind** será usada para obter o seno do ângulo em graus; caso contrário, a função **sin** será usada – a função seno a ser usada será baseada unicamente em se o usuário inseriu um 'g' ou não (um 'g' significa graus, então **sind** é usada; caso contrário, para qualquer outro caractere, o padrão de radianos é assumido, então a **sin** é usada)
- imprimirá o resultado.

Aqui estão alguns exemplos de execução do script:

```
>> imprimeseno
Digite o ângulo: 45
(r)adianos (o padrão) ou (g)raus: g
O seno é 0.71

>> imprimeseno
Digite o ângulo: pi
(r)adianos (o padrão) ou (g)raus: r
O seno é 0.00
```

Uma aplicação de uma instrução **if-else** é verificar se há erros nas entradas para um *script* (isso é chamado de verificação de erros). Por exemplo, um *script* anterior solicitou ao usuário um *raio* e usou isso para calcular a área de um círculo. No entanto, ele não verificou se o raio era válido (por exemplo, um número positivo). Aqui está um *script* modificado que verifica o raio:

checaraio.m

```
% Este script calcula a área de um círculo
% Verifica erros do raio do usuário
raio = input('Por favor, digite o raio: ');
if raio < = 0
    fprintf('Desculpe; %.2f não é um raio válido\n', raio)
else
    area = calcarea(raio);
    fprintf('Para um círculo com um raio de %.2f,', raio)
    fprintf('a área é %.2f\n', area)
end
```

Exemplos de execução desse *script* quando o usuário digita raios inválidos e, em seguida, válidos, são mostrados da seguinte maneira:

```
>> checaraio
Por favor, insira o raio: -4
Desculpe; -4.00 não é um raio válido

>> checaraio
Por favor, insira o raio: 5.5
Para um círculo com um raio de 5.50, a área é 95.03
```

A instrução **if-else** neste exemplo escolhe entre duas ações: imprimir uma mensagem de erro ou usar o raio para calcular a área e depois imprimir o resultado. Observe que a ação da cláusula **if** é uma instrução única, enquanto a ação da cláusula **else** é um grupo de três instruções.

4.3 INSTRUÇÕES IF-ELSE ANINHADAS

A instrução **if-else** é usada para escolher entre duas ações. Para escolher entre mais de duas ações, as instruções **if-else** podem ser aninhadas, ou seja, uma instrução dentro de outra. Por exemplo, considere implementar a seguinte função matemática contínua $y = f(x)$:

```
y = 1 se x < -1
y = x^2 se -1 ≤ x ≤ 2
y = 4 se x > 2
```

O valor de y é baseado no valor de x, que pode estar em um dos três intervalos possíveis. Escolhendo qual intervalo poderia ser realizado com três instruções **if** separadas, como segue:

```
if x < -1
    y = 1;
end
if x >= -1 && x <= 2
    y = x^2;
end
if x > 2
    y = 4;
end
```

Observe que o && na expressão da segunda instrução **if** é necessário. Escrever a expressão como $-1 <= x <= 2$ seria incorreto; recordar do capítulo 1 que essa expressão seria sempre ***verdadeira***, independentemente do valor da variável x.

Como as três possibilidades são mutuamente exclusivas, o valor de y pode ser determinado usando três instruções **if** separadas. No entanto, esse código não é muito eficiente: todas as três expressões lógicas devem ser avaliadas, independentemente do intervalo em que x caia. Por exemplo, se x for menor que -1, a primeira expressão será verdadeira e 1 será atribuído a y. No entanto, as duas expressões nas próximas duas instruções **if** ainda são avaliadas. Em vez de escrevê-lo dessa maneira, as instruções podem ser aninhadas para que a instrução **if-else** inteira termine quando uma expressão for considerada verdadeira:

```
if x < -1
    y = 1;
else
    % Se estamos aqui, x deve ser >= -1
    % Use uma instrução if-else para escolher
    % entre os dois intervalos restantes
```

```
    if x <= 2
        y = x^2;
    else
        % Não precisa verificar
        % Se estamos aqui, x deve ser > 2
        y = 4;
    end
end
```

Usando um **if-else** aninhado para escolher entre as três possibilidades, nem todas as condições devem ser testadas como no exemplo anterior. Nesse caso, se x for menor que -1, a instrução para atribuir 1 a y será executada e a instrução **if-else** será concluída para que nenhuma outra condição seja testada. Se, no entanto, x não for menor que -1, a cláusula **else** será executada. Se a cláusula **else** for executada, então já sabemos que x é maior que ou igual a -1 assim esta parte não precise ser testada.

Em vez disso, existem apenas duas possibilidades restantes: x é menor ou igual a 2 ou é maior que 2. Uma instrução **if-else** é usada para escolher entre essas duas possibilidades. Então, a ação da cláusula **else** foi outra instrução **if-else**. Embora seja longo, todo o código acima é uma instrução **if-else**, uma instrução **if-else** aninhada. As ações são recuadas para mostrar a estrutura da instrução. Aninhando instruções **if-else** dessa maneira, podem ser usadas para escolher entre 3, 4, 5, 6, ... – as possibilidades são praticamente infinitas!

Este é na verdade um exemplo de um tipo particular de **if-else** aninhado chamado de instrução **if-else *em cascata***. Esse é um tipo de instrução **if-else** aninhada em que as condições e ações se formam em cascata em um padrão semelhante a uma escada.

Nem todas as instruções **if-else** aninhadas estão em cascata. Por exemplo, considere o seguinte (que pressupõe que uma variável x foi inicializada):

```
if x >= 0
    if x < 4
        disp('a')
    else
        disp('b')
    end
else
    disp('c')
end
```

4.3.1 A Cláusula elseif

O CONCEITO DE PROGRAMAÇÃO

Em algumas linguagens de programação, escolher entre várias opções significa usar instruções **if-else** aninhadas. No entanto, o MATLAB tem outro método para realizar isso usando a cláusula **elseif**.

O MÉTODO EFICIENTE

Para escolher entre mais de duas ações, a cláusula **elseif** é usada. Por exemplo, se houver n opções (onde $n > 3$ neste exemplo), o seguinte forma geral será usada:

```
if condição1
    ação1
elseif condição2
    ação2
elseif condição3
    ação3
% etc: pode haver muitos deles
else
    açãon% a enésima ação
end
```

As ações das cláusulas **if**, **elseif** e **else** são naturalmente delimitadas pelas palavras reservadas **if**, **elseif**, **else** e **end**.

Por exemplo, o exemplo anterior poderia ser escrito usando a cláusula **elseif**, em vez de aninhar instruções **if-else**:

```
if x < -1
    y = 1;
elseif x <= 2
    y = x^2;
else
    y = 4;
end
```

Note que neste exemplo só precisamos de um **end**. Portanto, há três maneiras de realizar a tarefa original: usar três instruções **if** separadas, usando instruções **if-else** aninhadas e usar uma instrução **if** com cláusulas **elseif**, que é a mais simples.

Isso pode ser implementado em uma função que recebe um valor de x e retorna o valor correspondente de y:

calcy.m

```
function y = calcy(x)
% calcy calcula y como uma função de x
% Formato da chamada: calcy(x)
% y = 1, se x <-1
% y = x^2, se -1 <= x <= 2
% y = 4, se x > 2

if x < -1
    y = 1;
elseif x <= 2
    y = x^2;
else
    y = 4;
end
end
```

```
>> x = 1.1;
>> y = calcy(x)
y =
   1.2100
```

PERGUNTA RÁPIDA!

Como você poderia escrever uma função para determinar se um argumento de entrada é um escalar, um vetor ou uma matriz?

Resposta

Para fazer isso, a função **size** pode ser usada para encontrar as dimensões do argumento de entrada. Se o número de linhas e colunas for igual a 1, o argumento de entrada será escalar. Se, no entanto, apenas uma dimensão é 1, o argumento de entrada é um vetor (um vetor de linha ou coluna). Se nenhuma das dimensões for 1, o argumento de entrada será uma matriz. Essas três opções podem ser testadas usando uma instrução **if-else** aninhada. Neste exemplo, a palavra "escalar", "vetor" ou "matriz" é retornada da função.

findargtype.m

```
function tiposaida = desctipoarg(argentrada)
% desctipoarg descobre se o argumento de entrada
% é escalar, vetor ou matriz
% Formato da chamada: desctipoarg(ArgumentoDeEntrada)
% Retorna uma string

[l c] = length(argentrada);
if l == 1 && c == 1
    tiposaida = 'escalar';
elseif l == 1 k c == 1
    tiposaida = 'vetor';
else
    tiposaida = 'matriz';
end
end
```

Observe que não há necessidade de verificar o último caso: se o argumento de entrada não for escalar ou vetor, ele deve ser uma matriz!

Exemplos de chamada desta função são:

```
>> desctipoarg(33)
ans =
escalar

>> disp(desctipoarg(2:5))
vetor

>> desctipoarg(zeros(2, 3))
ans =
matriz
```

PRÁTICA 4.3

Modifique a função $desctipoarg$ para retornar 'escalar', 'vetor de linha', 'vetor de coluna' ou 'matriz', dependendo do argumento de entrada.

PRÁTICA 4.4

Modifique o $desctipoarg$ da função original para usar três instruções **if** separadas, em vez de uma instrução **if-else** aninhada.

Outro exemplo demonstra a escolha de mais do que apenas algumas opções. A seguinte função recebe uma nota inteira de um exame, que deve estar no intervalo de 0 a 10. A função então retorna uma letra correspondente à nota, de acordo com o seguinte esquema: a 9 ou 10 é um 'A', um 8 é um 'B', um 7 é um 'C', um 6 é um 'D' e qualquer coisa abaixo disso é um 'F'. Como as possibilidades são mutuamente exclusivas, poderíamos implementar o esquema de classificação usando instruções **if** separadas. No entanto, é mais eficiente ter uma instrução **if-else** com várias cláusulas **elseif**. Além disso, a função retorna a letra 'X' se a nota do questionário não for válida. A função assume que a entrada é um inteiro.

notaletra.m

```
function nota = notaletra(exame)
% letranota retorna a letra correspondente
% ao argumento nota do exame – um inteiro
% Formato da chamada: notaletra(notaInteiro)
% Retorna um caractere

% Primeiro, verifica erros
if exame < 0 || exame > 10
    grau = 'X';

% Se até aqui é válido, então encontra a letra
% correspondente
elseif exame == 9 || questionário == 10
    nota = 'A';
elseif exame == 8
    nota = 'B';
elseif exame == 7
    nota = 'C';
elseif exame == 6
    nota = 'D';
else
    nota = 'F';
end
end
```

Três exemplos de chamada desta função são:

```
>> exame = 8;
>> conceito = letranota(exame)
conceito =
```

```
B

>> exame = 4;
>> letranota(exame)
ans =
F

>> ln = letranota(22)
ln =
X
```

Na parte desta instrução **if** que escolhe a letra apropriada da nota para retornar, todas as expressões ***lógicas*** estão testando , em sequência, o valor da variável $exame$ para ver se ele é igual aos vários valores possíveis (primeiro 9 ou 10, depois 8 , depois 7, etc.). Esta parte pode ser substituída por uma instrução **switch**.

4.4 A INSTRUÇÃO SWITCH

Uma instrução **switch** geralmente pode ser usada no lugar de uma instrução **if-else** aninhada ou **if** com muitas cláusulas **elseif**. As instruções **switch** são usadas quando uma expressão é testada para ver se é igual a um dos vários valores possíveis.

A forma geral da instrução **switch** é:

```
switch expressão_do_switch
    case expcaso1
        ação1
    case expcaso2
        ação2
    case expcaso3
        ação3
    % etc: pode haver muitos deles
    otherwise
        açãon
end
```

A instrução **switch** inicia com a palavra reservada **switch** e termina com a palavra reservada **end**. A $expressão\_do\_switch$ é comparada, em sequência, às expressões dos **case** ($expcaso1$, $expcaso2$, etc.). Se o valor da $expressão\_do\_switch$ corresponder a $expcaso1$, por exemplo, a $ação1$ será executada e a instrução **switch** será finalizada. Se o valor corresponder a $expcaso3$, então $ação3$ será executada e, em geral, se o valor corresponder a $expcasoi$, em que i pode ser qualquer inteiro de 1 a n, a $açãoi$ é executada. Se o valor da $expressão\_do\_switch$ não corresponder a nenhuma das expressões dos **case**, a ação após a palavra **otherwise** será executada (a enésima ação, $açãon$), se houver um **otherwise** (se não, nenhuma ação será executada). Não é necessário ter uma cláusula **otherwise**, embora seja frequentemente útil. A $expressão\_do\_switch$ deve ser um escalar ou uma *string*.

Para o exemplo anterior, a $expressão\_do\_switch$ pode ser usada da seguinte maneira:

notaletraswitch.m

```
function nota = notaletraswitch(exame)
% notaletraswitch retorna a letra correspondente
% ao argumento nota do exame – um inteiro,
% usando o switch
% Formato da chamada: notaletraswitch(integerQuiz)
% Retorna um caractere

% Primeiro, verifica erros
if exame < 0 || exame > 10
    nota = 'X';
else
    % Se até aqui é válido, então encontre a letra
    % correspondente usando um switch
    switch exame
        case 10
            nota = 'A';
        case 9
            nota = 'A';
        case 8
            nota = 'B';
        case 7
            nota = 'C';
        case 6
            nota = 'D';
        otherwise
        nota = 'F';
    end
end
end
```

Aqui estão dois exemplos de chamada **desta** função:

```
>> exame = 22;
>> ln = notaletraswitch(exame)
ln =
X

>> notaletraswitch(9)
ans =
A
```

Nota

Supõe-se que o usuário insira um valor inteiro. Se o usuário não o fizer, uma mensagem de erro será impressa ou um resultado incorreto será retornado. Os métodos para remediar isso serão discutidos no Capítulo 5.

Como a mesma ação de impressão 'A' é desejada para mais de uma nota, eles podem ser combinados da seguinte maneira:

```
switch exame
    case {10, 9}
        nota = 'A';
    case 8
        grau = 'B';
     % etc.
```

As chaves em torno das expressões 10 e 9 do **case** são necessárias. Neste exemplo, verificamos primeiro o erro, usando uma instrução **if-else**. Então, se a nota estava no intervalo válido, uma instrução **switch** foi usada para encontrar a letra da nota correspondente. Às vezes, a cláusula **otherwise** é usada para a mensagem de erro, em vez de usar primeiro uma instrução **if-else**. Por exemplo, se o usuário deve inserir apenas 1, 3 ou 5, o *script* pode ser organizado da seguinte maneira:

mensdeerro.m

```
% Exemplo de otherwise para mensagem de erro
escolha = input('Digite 1, 3 ou 5:');

switch escolha
    case 1
        disp('É um!!')
    case 3
        disp('É um três!!')
    case 5
        disp('É um cinco!!')
    otherwise
        disp('Siga as instruções na próxima vez!!')
end
```

Neste exemplo, as ações são tomadas se o usuário inserir corretamente uma das opções válidas. Se o usuário não inserir corretamente, a cláusula **otherwise** manipula a impressão de uma mensagem de erro. Observe o uso de duas aspas simples na seqüência para imprimir uma cotação.

```
>> mensdeerro
Digite 1, 3 ou 5: 4
Siga as instruções na próxima vez!!
```

Observe que a ordem das expressões de caso não importa, exceto que esta é a ordem na qual elas serão avaliadas.

4.5 A FUNÇÃO MENU

O MATLAB possui uma função interna chamada **menu** que exibire uma janela de figura com botões para as opções. A primeira *string* passada para a função **menu** é o cabeçalho (uma instrução), e o restante são rótulos que aparecem nos botões. A função retorna o número do botão que é pressionado. Por exemplo,

```
>> minhaescolha = menu('Escolha uma pizza:', 'Queijo', ...
'Cogumelo', 'Salsicha');
```

exibirá a Janela de Figura vista na Figura 4.1 e armazenará o resultado do botão do usuário na variável $minhaescolha$.

Existem três botões, cujos valores equivalentes são 1, 2 e 3. Por exemplo, se o usuário apertar o botão 'Salsicha', $minhaescolha$ a terá o valor 3:

```
>> minhaescolha
minhaescolha =
    3
```

Tenha em atenção que as cadeias 'Queijo', 'Cogumelo' e 'Salsicha' são apenas rótulos nos botões. O valor real do botão pressionado neste exemplo seria 1, 2 ou 3, então é isso que seria armazenado na variável $minhaescolha$.

Um *script* que usa essa função **menu**, usa uma instrução **if-else** ou uma instrução **switch** para executar uma ação apropriada com base no botão pressionado. Por exemplo, o *script* a seguir simplesmente imprime qual pizza pedir, usando uma instrução **switch**.

pedepizza.m

```
% Este script pede ao usuário o tipo de pizza
% e imprime o tipo a pedir, usando um switch

minhaescolha = menu('Escolha uma pizza', 'Queijo', ...
'Shroom', 'Salsicha');
switch minhaescolha
    case 1
        disp('Pedir uma pizza de queijo')
    case 2
        disp('Pedir uma pizza de cogumelos')
    case 3
        disp('Pedir uma pizza de salsicha')
    otherwise
        disp('Não queremos pizza hoje')
end
```

Este é um exemplo de como executar este script e clicar no botão 'Salsicha':

```
>> pedepizza
Pedir uma pizza de salsicha
```

FIGURA 4.1 Janela de figura menu

PERGUNTA RÁPIDA!

Como a ação **otherwise** poderia ser executada nesta instrução **switch**?

Resposta

Se o usuário clicar no 'X' vermelho no topo da caixa de menu para fechá-la, em vez declicar em um dos três botões, o valor retornado da função de menu será 0, o que fará com que a cláusula **otherwise** seja executada. Isso também poderia ter sido foi realizado usando uma expressão de **case** 0 em vez de **otherwise**.

Em vez de usar uma instrução **switch** nesse *script*, um método alternativo seria usar uma instrução **if-else** com cláusulas **elseif**.

pedepizzaifelse.m

```
% Este script pede ao usuário um tipo de pizza
% e imprime que tipo para encomendar usando if-else
meupedido = menu('Escolha uma pizza', 'Queijo', ...
'Cogumelo', 'Salsicha');
if meupedido == 1
    disp('Pedir uma pizza de queijo')
elseif meupedido == 2
    disp('Pedir uma pizza de cogumelos')
elseif meupedido == 3
    disp('Pedir uma pizza de salsicha')
else
    disp(Não queremos pizza hoje')
end
```

PRÁTICA 4.5

Escreva uma função que receberá um número como um argumento de entrada. Ele usará a função **menu** para exibir 'Escolher uma função' e terá os botões 'fix', 'floor' e 'abs'. Usando uma instrução **switch**, a função calculará e retornará a função solicitada (por exemplo, se 'abs' for escolhido, a função retornará o valor absoluto do argumento de entrada). Escolha uma quarta função para retornar se o usuário clicar no 'X' vermelho em vez de apertar um botão.

4.6 AS FUNÇÕES "IS" EM MATLAB

Existem muitas funções embutidas no MATLAB que testam se algo é verdadeiro ou não; essas funções têm nomes que começam com a palavra "is". Por exemplo, já vimos o uso da função **isequal** para comparar igualdade matrizes. Como outro exemplo, a função chamada **isletter** retorna 1 ***lógico*** se o argumento de caractere for uma letra do alfabeto ou 0 se não for:

```
>> isletter('h')
ans =
   1
```

```
>> isletter('4')
ans =
   0
```

A função **isletter** retornará ***verdadeiro*** ou ***falso lógicos*** para que possa ser usada em uma condição em uma instrução **if**. Por exemplo, aqui está o código que solicita ao usuário um caractere e, em seguida, imprime se é ou não uma letra:

```
meucarac = input('Por favor, insira um caractere: ', 's');
if isletter(meucarac)
    disp('É uma letra')
else
    disp('Não é uma letra')
end
```

Quando usado em uma instrução **if**, não é necessário testar o valor para ver se o resultado de **isletter** é igual a 1 ou 0; isso é redundante. Em outras palavras, na condição da instrução **if**,

```
isletter(meucarac)
```

e

```
isletter(meucarac) == 1
```

produziriam os mesmos resultados.

PERGUNTA RÁPIDA!

Como podemos escrever nosso própria função $minhaisletter$ de para obter o mesmo resultado que o $isletter$?

Resposta

A função compararia a posição do caractere na codificação de caracteres.

minhaisletter.m

```
function saidalogica = minhaisletter(caracEntrada)
% minhaisletter retorna verdadeiro se o argumento de entrada
% é uma letra do alfabeto ou falso se não for
% Formato da chamada: minhaisletter(caractereEntrada)
% Retorna lógico 1 ou 0

saidalogica = caracEntrada >= 'a' && caracEntrada <= 'z'...
         || caracEntrada >= 'A' && caracEntrada <= 'Z';
end
```

Observe que é necessário verificar as letras minúsculas e maiúsculas.

A função **isempty** retorna ***verdadeiro lógico*** se uma variável estiver vazia, ***falso lógico*** se tiver um valor ou uma mensagem de erro se a variável não existir. Portanto, ele pode ser usado para determinar se uma variável possui um valor ainda ou não. Por exemplo,

```
>> clear
>> isempty(vetv)
Udefined function or 'vetv'.

>> vetv = [];
>> isempty(vetv)
ans =
   1

>> vetv = [vetv 5];
>> isempty(vetv)
ans =
   0
```

A função **isempty** também determinará se uma variável *string* está vazia ou não. Por exemplo, isso pode ser usado para determinar se o usuário inseriu uma *string* em uma função **input**:

```
>> cadeia = input('Por favor digite uma string: ', 's');
Por favor, insira uma string:
>> isempty(cadeia)
ans =
   1
```

PRÁTICA 4.6

Solicite ao usuário uma *string* e, em seguida, imprima a *string* que o usuário inseriu ou uma mensagem de erro se o usuário não tiver inserido nada.

A função **iskeyword** determinará se uma *string* é ou não o nome de uma palavra-chave em MATLAB e, portanto, algo que não pode ser usado como um nome de identificador. Por si só (sem argumentos), retornará a lista de todas as palavras-chave. Observe que os nomes de funções como "sin" não são palavras-chave, portanto, seus valores podem ser sobrescritos se usados como um nome de identificador.

```
>> iskeyword('sin')
ans =
   0

>> iskeyword('switch')
ans =
   1

>> iskeyword
ans =
   'brak'
   'case'
   'catch'
   % etc.
```

Existem muitas outras funções "is"; a lista completa pode ser encontrada no Navegador de Ajuda.

Explorar Outros Recursos Interessantes

- Existem muitas outras funções "is". Quanto mais conceitos forem abordados no livro, mais e mais dessas funções serão introduzidas. Outros que você pode querer explorar agora incluem **isvarname** e funções que lhe dirão se um argumento é um tipo particular ou não (**ischar**, **isfloat**, **isinteger**, **islogical**, **isnumeric**, **isstr**, **isreal**).
- Existem funções "is" para determinar o tipo de uma matriz: **isvector**, **isrow**, **iscolumn**.
- As funções **try/catch** são um tipo particular de **if-else** usado para localizar e evitar possíveis erros. Eles podem ser um pouco complicados de entender neste momento, mas mantenha-os em mente para o futuro! ■

Resumo

Armadilhas Comuns

- Usar = em vez de == para igualdade nas condições.
- Colocar um espaço na palavra chave **elseif**.
- Não usar apóstrofos ao comparar uma variável *string* com uma *string*, tal como

  ```
  letra == y
  ```

  ao invés de

  ```
  letra == 'y'
  ```

- Não escrever inteiramente uma ***expressão lógica***. Um exemplo é digitar

  ```
  raio || altura <= 0
  ```

  ao invés de

  ```
  raio <= 0 || altura <= 0
  ```

  ou digitar

  ```
  letra == '' y '||' Y '
  ```

  ao invés de

  ```
  letra == 'y' || letra == 'Y'
  ```

  Note que estas instruções são logicamente incorretas, mas não resultariam em mensagens de erro. Note também que a expressão "`letra == 'y' || 'Y'`" sempre será ***verdadeira***, independentemente do valor da letra variável, já que 'Y' é um valor diferente de zero e, portanto, uma expressão ***verdadeira***.

- Condições escritas que são mais complicadas do que o necessário, como

  ```
  if (x <5) == 1
  ```

  em vez de apenas

  ```
  if (x < 5)
  ```

  (O "== 1" é redundante.)

- Usando uma instrução **if** em vez de uma instrução **if-else** para verificação de erros; por exemplo,

```
se ocorrer erro
    imprimir mensagem de erro
fim
continuar o resto do código
```

ao invés de

```
se ocorrer erro
    imprimir mensagem de erro
else
    continuar o resto do código
fim
```

No primeiro exemplo, a mensagem de erro seria impressa, mas o programa continuaria assim mesmo.

Diretrizes de Estilo de Programação

- Use o recuo para mostrar a estrutura de um *script* ou função. Em particular, as ações em uma instrução **if** devem ser recuadas.

- Quando a cláusula **else** não é necessária, use uma instrução **if** em vez de uma instrução **if-else**. O seguinte é um exemplo:

```
if unit == 'i'
    len = len * 2.54;
else
    len = len; % isso não faz nada, então pule isso!
end
```

Em vez disso, basta usar:

```
if unit == 'i'
    len = len * 2.54;
end
```

- Não coloque condições desnecessárias em cláusulas **else** ou **elseif**. Por exemplo, o seguinte imprime uma coisa se o valor de um número variável for igual a 5 e outra coisa, se não for.

```
if numero == 5
    disp('É um 5')
elseif numero ~= 5
    disp('não é um 5')
end
```

A segunda condição, no entanto, não é necessária. Ou o valor é 5 ou não, então apenas o **else** lidaria com isso:

```
if numero == 5
    disp('É um 5')
else
    disp('não é um 5')
end
```

- Ao usar a função de **menu**, certifique-se de que o programa manipule a situação quando o usuário clicar no 'X' vermelho na caixa de menu, em vez de pressionar um dos botões. ■

Palavras Reservadas do MATLAB	
if	else
switch	elseif
case	otherwise

Funções e Comandos do MATLAB	
menu	Isletter
isempty	iskeyword

Exercícios

1. Escreva um *script* que testa se o usuário pode seguir as instruções. Ele solicita que o usuário insira um "x". Se o usuário inserir algo diferente de um "x", ele imprime uma mensagem de erro; caso contrário, o *script* não faz nada.

2. Escreva uma função $proximahora$ que receba um argumento inteiro, que é uma hora do dia, e retorna a próxima hora. Isso pressupõe um relógio de 12 horas; Assim, por exemplo, a próxima hora após 12 seria 1. Aqui estão dois exemplos de chamar essa função.

```
>> fprintf('A próxima hora será %d.\n', (3))
A próxima hora será 4.
>> fprintf('A próxima hora será %d.\n', proximahora(12))
A próxima hora será 1.
```

3. Escreva um *script* para calcular o volume de uma pirâmide, que é $1/3 * base * altura$, onde a $base$ é $comprimento$ * $largura$. Solicite que o usuário insira valores para o $comprimento$, a $largura$ e a $altura$ e, em seguida, calcule o volume da pirâmide. Quando o usuário inserir cada valor, ele ou ela também será solicitado a informar "p" para polegadas ou "c" para centímetros. (Observe que 2.54 cm = 1 polegada.) O *script* deve imprimir o volume em polegadas cúbicas com três casas decimais. Como exemplo, o formato de saída será:

```
Este programa irá calcular o volume de uma pirâmide.
Digite o comprimento da base: 50
Este está em p ou c? p
Digite a largura da base: 6
Esta está em p ou c? c
Digite a altura: 4
Isso é eu ou c? p
O volume da pirâmide é xxx.xxx polegadas cúbicas.
```

4. As leituras da pressão arterial sistólica e diastólica são encontradas quando o coração está bombeando e o coração está em repouso, respectivamente. Um experimento biomédico está sendo conduzido apenas em participantes cuja pressão arterial é ótima. Isso é definido como uma pressão arterial sistólica menor que 120 e uma pressão arterial diastólica menor que 80. Escreva um *script* que indique as pressões arteriais sistólica e

diastólica de uma pessoa e imprima se essa pessoa é ou não uma candidata ao experimento.

5. O Teorema de Pitágoras afirma que, para um triângulo retângulo, a relação entre o comprimento da hipotenusa c e os comprimentos dos outros lados a e b é dada por:

$$c^2 = a^2 + b^2$$

Escreva um script que solicitará ao usuário os comprimentos a e c, chame uma função $encontrab$ para calcular e retornar o comprimento de b e imprima o resultado. Observe que qualquer valor de a ou c que seja menor ou igual a zero não faria sentido, portanto, o *script* deve imprimir uma mensagem de erro se o usuário inserir qualquer valor inválido. Aqui está a função $encontrab$:

encontrab.m

```
function b = findb(a, c)
% Calcula b a partir de a e c
b = sqrt(c^2 - a^2);
end
```

6. A excentricidade de uma elipse é definida como $\sqrt{1-\left(\frac{b}{a}\right)^2}$

onde a é o semi-eixo maior e b é o semi-eixo menor da elipse. Um *script* solicita ao usuário os valores de a e b. Como a divisão por 0 não é possível, o *script* imprime uma mensagem de erro se o valor de a for 0 (ignora qualquer outro erro, no entanto). Se a não for 0, o *script* chama uma função para calcular e retorna a excentricidade e, em seguida, o script imprime o resultado. Escreva o *script* e a função.

7. A área A de um losango é definida como $A = \frac{d_1 d_2}{2}$, onde d_1 e d_2 são os comprimentos das duas diagonais. Escreva um *script* $losango$ de que primeiro peça ao usuário os comprimentos das duas diagonais. Se um deles for um número negativo ou zero, o *script* imprime uma mensagem de erro. Caso contrário, se ambos forem positivos, ele chama uma função $arealosango$ para retornar a área do losango e imprime o resultado. Escreva a função também! Os comprimentos das diagonais, que você pode assumir que são em centímetros, são passados para a função $arealosango$.

8. Simplifique este comando:

```
if número > 100
    numero = 100;
else
    numero = número;
end
```

9. Simplifique este comando:

```
if val >= 10
    disp('Olá')
elseif val < 10
    disp('oi')
end
```

10. Escreva uma função $criavetMParaN$ que criará e retornará um vetor de inteiros de m para n (onde m é o primeiro argumento de entrada e n é o segundo), independentemente

de m ser menor que n ou maior que n. Se m é igual a n, o "vetor" será apenas 1×1 ou um escalar.

11. A equação de continuidade na dinâmica dos fluidos para o fluxo contínuo de fluido através de um tubo de vapor equivale ao produto da densidade, velocidade e área em dois pontos que possuem áreas de seção transversal variáveis. Para o fluxo incompressível, as densidades são constantes, portanto a equação é $A_1V_1 = A_1V_2$. Se as áreas e V_1 são conhecidas, V_2 pode ser encontrado como $\frac{A_1}{A_2} V_1$. Portanto, se a velocidade no segundo ponto aumenta ou diminui, depende das áreas nos dois pontos. Escreva um *script* que indique ao usuário as duas áreas em centímetros quadrados e imprima se a velocidade no segundo ponto aumentará, diminuirá ou permanecerá igual ao primeiro ponto.

12. Em química, o pH de uma solução aquosa é uma medida da sua acidez. A escala de pH varia de 0 a 14, inclusive. Diz-se que uma solução com um pH de 7 é neutra, uma solução com um pH superior a 7 é básica e uma solução com um pH inferior a 7 é ácida. Escreva um *script* que indique ao usuário o pH de uma solução e imprima se ele é neutro, básico ou ácido. Se o usuário inserir um pH inválido, uma mensagem de erro será impressa.

13. Escreva uma função $flipvet$ que receberá um argumento de entrada. Se o argumento de entrada for um vetor de linha, a função inverterá a ordem e retornará um novo vetor de linha. Se o argumento de entrada for um vetor de coluna, a função inverterá a ordem e retornará um novo vetor de coluna. Se o argumento de entrada for uma matriz ou um escalar, a função retornará o argumento de entrada inalterado.

14. Em um *script*, o usuário deve inserir um 'y' ou 'n' em resposta a um *prompt*. A entrada do usuário é lida em uma variável de caractere chamada "$letra$". O *script* irá imprimir "OK", continue" se o usuário inserir um 'y' ou 'Y', ou imprimirá "OK, pare" se o usuário inserir um 'n' ou 'N' ou “Erro” se o o usuário insere qualquer outra coisa. Coloque primeiro este comando no *script*:

    ```
    letra = input('Digite sua resposta:', 's');
    ```

    Escreva o *script* usando uma única instrução **if-else** aninhada (cláusula **elseif** é permitida).

15. Escreva o *script* do exercício anterior usando uma instrução **switch**.

16. Na aerodinâmica, o Número de Mach é uma quantidade crítica. É definido como a relação entre a velocidade de um objeto (por exemplo, uma aeronave) e a velocidade do som. Se o Número de Mach for menor que 1, o fluxo é subsônico; se o número Mach for igual a 1, o fluxo é transônico; se o número Mach for maior que 1, o fluxo é supersônico. Escreva um *script* que indique ao usuário a velocidade de uma aeronave e a velocidade do som na altitude atual da aeronave e imprima se a condição é subsônica, transônica ou supersônica.

17. Escreva um *script* que indique ao usuário uma temperatura em graus Celsius e, em seguida, um "F" para Fahrenheit ou "K" para Kelvin. O *script* imprimirá a temperatura correspondente na escala especificada pelo usuário. Por exemplo, a saída pode ser assim:

    ```
    Digite a temperatura em graus C: 29.3
    Você quer converter para Kelvin (K) ou Fahrenheit (F)? F
    A temperatura em graus F é 84.7
    ```

    O formato da saída deve ser exatamente conforme especificado acima. As conversões são:

$$F = \frac{9}{5}C + 32$$

$$K = C + 273.15$$

18. Escreva um *script* que gere um inteiro aleatório e imprima se o inteiro aleatório é um número par ou ímpar (Dica: um número par é divisível por 2, enquanto um número ímpar não é; portanto, verifique o restante depois de dividir por 2).

19. Os aminoácidos são os compostos fundamentais que compõem as proteínas. Vírus, como influenza e HIV, têm genomas cujo codificação das proteínas têm resultados patogênicos. Os corpos humanos reconhecem proteínas estranhas, ligando-as a outras moléculas, para que possam ser reconhecidas e mortas. Escreva um *script* que determine se um aminoácido vai se ligar a uma certa molécula. Sabe-se que a primeira região $(1-5)$ da molécula se liga fortemente aos aminoácidos A, C, I, L, Y e E, e fracamente a W, S, M, G e K. Sabe-se também que a segunda região $(6-10)$ liga-se fortemente a H, D, W, K, L e A e fracamente a I, E, P, C e T. O *script* solicita ao usuário duas coisas: o número da região e um caractere para o aminoácido. O *script* deve então determinar se o aminoácido e a molécula resultam em uma ligação "forte" ou "fraca".

20. Na dinâmica de fluidos, o número de Reynolds Re é um número adimensional usado para determinar a natureza de um fluxo de fluido. Para um fluxo interno (por exemplo, fluxo de água através de um tubo), o fluxo pode ser categorizado da seguinte forma:

Re ≤ 2300	Região Laminar
2300 < Re ≤ 4000	Região de Transição
Re > 4000	Região Turbulenta

    Escreva um *script* que solicitará ao usuário o número de Reynolds de um fluxo e imprimirá a região em que o fluxo está. Seria uma boa ideia escrever as instruções de seleção usando o **switch**? Por que sim ou por que não?

21. Mudanças globais de temperatura resultaram em novos padrões de tempestades em muitas partes do mundo. O rastreamento das velocidades do vento e uma variedade de categorias de tempestades é importante para entender as ramificações dessas variações de temperatura. Programas que trabalham com dados de tempestade usarão instruções de seleção para determinar a gravidade das tempestades e também para tomar decisões com base nos dados.

    Se uma tempestade é uma depressão tropical, uma tempestade tropical ou um furacão é determinado pela velocidade média sustentada do vento. Em milhas por hora, uma tempestade é uma depressão tropical se os ventos forem menores que 38 mph. É uma tempestade tropical, se os ventos são entre 39 e 73 mph, e é um furacão, se as velocidades do vento são ≥ 74 mph. Escreva um *script* que indique ao usuário a velocidade do vento da tempestade e imprima o tipo de tempestade.

22. As nuvens são geralmente classificadas como de alto, médio ou baixo nível. A altura da nuvem é o fator determinante, mas os intervalos variam dependendo da temperatura. Por exemplo, em regiões tropicais, as classificações podem ser baseadas nos seguintes intervalos de altura (dados em pés):

Baixo	0 – 6500
Médio	6501 – 20000
Alto	> 20000

Escreva um *script* que indique ao usuário a altura da nuvem em pés e imprima a classificação.

23. A Escala de Vento de Beaufort é usada para caracterizar a força dos ventos. A escala usa valores inteiros e vai de uma força de 0, que é sem vento, até 12, que é um furacão. O *script* a seguir gera primeiro um valor de força aleatório. Em seguida, imprime uma mensagem sobre o tipo de vento que a força representa, usando uma instrução **switch**. Você deve reescrever esse comando **switch** como uma instrução **if-else** aninhada que realiza exatamente a mesma coisa. Você pode usar cláusulas **else** e/ou **elseif**.

```
forcavento = randi([0, 12]);
switch forcavento
    case 0
        disp('Não há vento')
    case {1, 2, 3, 4, 5, 6}
        disp('Há uma brisa')
    case {7, 8, 9}
        disp('Isto é um vendaval')
    case {10, 11}
        disp('É uma tempestade')
    case 12
        disp('Olá, furacão!')
end
```

24. Reescreva a instrução seguinte **if-else** aninhada como uma instrução **switch** que realiza exatamente o mesmo resultado para todos os valores possíveis. Suponha que val é uma variável inteira que foi inicializada e que “ok”, “xx”, “yy”, “tt” e “meio” são funções. Escreva a instrução **switch** da maneira mais sucinta.

```
if val > 5
    if val < 7
        ok(val)
    elseif val < 9
        xx(val)
    else
        yy(val)
    end
else
    if val < 3
        yy(val)
    elseif val == 3
        tt(val)
    else
        meio(val)
    end
end
```

25. Reescreva a seguinte instrução switch como uma instrução if-else aninhada (cláusulas elseif podem ser usadas). Suponha que haja uma letra variável e que ela tenha sido inicializada.

```
trocar carta
case 'x'
disp('Olá')
case {'y', 'Y'}
disp('sim')
```

```
case 'Q'
disp('Quit')
de outra forma
disp('Erro')
end
```

26. Reescreva a seguinte instrução **if-else** aninhada como uma instrução **switch** que realiza exatamente a mesma coisa. Suponha que num é uma variável inteira que foi inicializada e que existem funções f1, f2, f3 e f4. Não use nenhuma instrução **if** ou **if-else** nas ações da instrução **switch**, apenas chamadas para as quatro funções.

```
if num < -2 || num > 4
   f1(num)
else
    if num <= 2
        if num> = 0
            f2(num)
        else
            f3(num)
        end
    else
        f4(num)
    end
end
```

27. Escreva um *script* $menuArea$ de que imprimirá uma lista que consiste em "cilindro", "círculo" e "retângulo". Ele solicita que o usuário escolha um e, em seguida, solicita ao usuário as valores apropriados (por exemplo, o raio do círculo) e imprime sua área. Se o usuário inserir uma opção inválida, o *script* simplesmente imprime uma mensagem de erro. O *script* deve usar uma instrução **if-else** aninhada para realizar isso. Aqui estão dois exemplos de execução (as unidades são consideradas em centímetros).

```
>> menuArea
Menu
1. Cilindro
2. Círculo
3. Retângulo
Por favor, escolha uma opção: 2
Digite o raio do círculo: 4.1
A área é 52.81
>> menuArea
Menu
1. Cilindro
2. Círculo
3. Retângulo
Por favor, escolha uma opção: 3
Digite o comprimento do retângulo: 4
Digite a largura do retângulo: 6
A área é 24.00
```

28. Modifique o *script* $menuArea$ para usar uma instrução **switch** para decidir qual área será calculada.

29. Modifique o *script* $menuArea$ (qualquer versão) para usar a função interna **menu** em vez de imprimir as opções de menu.

30. Escreva um *script* que solicite ao usuário um valor de uma variável x. Em seguida, ele usa a função **menu** para apresentar escolhas entre "$sin(x)$", "$cos(x)$" e "$tan(x)$". O *script* irá imprimir qualquer função de x que o usuário escolher. Use uma instrução **if-else** para realizar isso.

31. Modifique o *script* anterior para usar uma instrução **switch**.

32. Escreva uma função que receberá um número como um argumento de entrada. Ele usará a função **menu** que exibirá "Escolher uma função:" e terá as opções "**ceil**", "**round**" e "**sign**". Usando uma instrução **switch**, a função calculará e retornará a função solicitada (por exemplo, se "**round**" for escolhido, a função retornará o valor arredondado do argumento de entrada).

33. Modifique a função no Exercício 32 para usar uma instrução **if-else** aninhada.

34. Escreva um *script* que irá solicitar ao usuário uma *string* e, em seguida, imprima se ela está vazia ou não.

35. Escreva uma função chamada $facamat$ que receberá dois vetores de linha como argumentos de entrada e, a partir deles, criará e retornará uma matriz com duas linhas. Você não pode assumir que o comprimento dos vetores é conhecido. Além disso, os vetores podem ter comprimentos diferentes. Se esse for o caso, primeiro adicione 0 ao final de um vetor para torná-lo igual ao outro. Por exemplo, uma chamada para a função poderia ser:

```
>> facamat(1:4, 2:7)
ans =
   1   2   3   4   0   0
   2   3   4   5   6   7
```

Referência

ATAWAY, S. MATLAB A Pratical Introduction to Programming and Program Solving. Butterworth-Heinemann/Elsevier, Waltham, MA, USA, Third Edition, 2013.