3 Controle de fluxo

Neste capítulo você vai aprender como controlar o fluxo de ações do seu programa utilizando perguntas (condições) ou combinações de perguntas que podem ter respostas lógicas do tipo "verdadeiro" ou "falso". Você vai ver como utilizar instruções para criar desvios, bifurcações e múltiplos caminhos alternativos de processamento. Também vai aprender como fazer com que certas ações sejam realizadas um número predefinido de vezes ou enquanto condições bem definidas forem satisfeitas ou não.

Existem essencialmente 5 estruturas utilizadas para controlar o fluxo de um programa:

1. `if (expressão) { instruções } else { instruções }`, conhecida resumidamente como *if/else*, que executa um ou outro grupo de instruções dependendo do valor de uma expressão, que pode ser `true` (verdadeira) ou `false` (falsa).

2. `switch (expressão) case { rótulo: { instruções }; rótulo: { instruções }; ... }`, conhecida resumidamente como *switch/case*", que possibilita a execução de um ou mais grupos de instruções quando uma expressão dada resulta idêntica a um rótulo.

3. `for (inicialização; teste; incremento) { instruções }`, denominada usualmente como *laço for*, que em geral é utilizada para executar um grupo de instruções um número bem definido de vezes.

4. `while (condição) { instruções }`, conhecida como *laço while*, que executa um grupo de instruções enquanto uma condição for verdadeira.

5. `do { instruções } while (condição)`, conhecida como *laço do*, que executa um grupo de instruções pelo menos uma vez e repete sua execução enquanto a condição for verdadeira.

A seguir, uma análise mais detalhada de cada uma delas.

`if (`*`expressão`*`) { instruções } else { instruções }`

O exemplo a seguir pede ao usuário que digite dois números que compõem um ponto (*x*,*y*) no plano cartesiano, calcula a sua distância à origem e diz ao usuário se o ponto está dentro de um círculo de raio unitário. Note o uso do `parseFloat()` para converter o valor digitado em um número real.

Listagem:

```
<script>
var x = parseFloat(prompt("Digite um número entre 0 e 1: "));
var y = parseFloat(prompt("Digite outro número entre 0 e 1: "));
var r = Math.sqrt(x*x+y*y);
if (r<1) {
   var msg = "O ponto (" + x + "," + y + ") está DENTRO do círculo";
   alert(msg);
   }
else {
   var msg = "O ponto (" + x + "," + y + ") está FORA do círculo";
   alert(msg);
   }
</script>
```

O exemplo acima assume que se o raio calculado a partir dos valores digitados para *x* e *y* for *exatamente* igual a 1, o ponto está *fora* do círculo. Troque o operador menor ("<") pelo operador menor ou igual ("<=") para fazer com que o script considere que, neste caso, o ponto esteja dentro da circunferência.

O script abaixo faz exatamente a mesma coisa, mas simplifica o bloco de instruções dentro das chaves. Note que quando há apenas uma instrução em cada bloco, o uso das chaves passa a ser opcional.

Listagem:

```
<script>
var x = parseFloat(prompt("Digite um número entre 0 e 1: "));
var y = parseFloat(prompt("Digite outro número entre 0 e 1: "));
var r = Math.sqrt(x*x+y*y);
if (r<1)
   alert("O ponto (" + x + "," + y + ") está DENTRO do círculo");
else
   alert("O ponto (" + x + "," + y + ") está FORA do círculo");
</script>
```

Na estrutura `if/else`, o uso do `else { ... }` é opcional. O exemplo a seguir pede ao usuário que digite um número e verifica se este é negativo. Se for, avisa ao usuário que ele digitou um número negativo e prossegue calculando a raiz quadrada (`Math.sqrt(arg)`) de seu módulo (`Math.abs(arg)`). Note que a função que calcula a raiz e mostra o resultado é sempre executada, independentemente do resultado do teste.

Listagem:

```
<script>
var x = parseFloat(prompt("Entre com um número: " ));
if (x<0) alert("Você digitou um número negativo!");
alert(Math.sqrt(Math.abs(x)));
</script>
```

Sua vez.. (3-1)

Modifique o exemplo anterior de modo que continue avisando que o usuário digitou um número negativo quando for o caso, mas que só calcule a raiz quadrada de números maiores ou iguais a zero.

O aninhamento de várias instruções `if ... else` pode ser utilizado para a tomada de decisões que dependem de decisões anteriores. O exemplo a seguir solicita que o usuário digite um par de coordenadas (*x*,*y*) em um dos quadrantes do plano cartesiano, sorteado aleatoriamente.

Listagem:

```
<script>
var num = Math.random();
if (num<0.25) quad = "PRIMEIRO";
else if (num<0.5) quad = "SEGUNDO";
else if (num<0.75) quad = "TERCEIRO";
else quad = "QUARTO";
var x = parseFloat(prompt("Digite x no " + quad + " quadrante: "));
var y = parseFloat(prompt("Digite y no " + quad + " quadrante: "));
</script>
```

Na primeira linha, a variável `num` recebe um número aleatório entre 0 e 1. Se o número sorteado estiver entre 0 e 0.25, a variável `quad` recebe o texto "PRIMEIRO"; se estiver entre 0.25 e 0.5, recebe o texto "SEGUNDO"; se estiver entre 0.5 e 0.75, recebe o texto "TERCEIRO"; e se estiver entre 0.75 e 1, recebe o texto "QUARTO". Definido o quadrante, o script pede ao usuário que digite valores para (*x*,*y*) no quadrante especificado na variável `quad`.

No exemplo, como cada bloco de instruções consiste sempre de apenas uma instrução foram omitidas todas as chaves delimitadoras e indentação opcionais. A versão abaixo faz exatamente a mesma coisa mas coloca chaves e indentações:

Listagem:

```
<script>
var num = Math.random();
if (num<0.25) {
   quad = "PRIMEIRO";
   }
else {
   if (num<0.5) {
      quad = "SEGUNDO";
```

```
      }
   else {
      if (num<0.75) {
         quad = "TERCEIRO";
      }
      else {
         quad = "QUARTO";
      }
   }
}
var x = parseFloat(prompt("Digite x no " + quad + " quadrante: "));
var y = parseFloat(prompt("Digite y no " + quad + " quadrante: "));
</script>
```

Sua vez... (3-2)

Utilize `if`'s aninhados para fazer com que no exemplo 3.2 o script tenha uma terceira opção, alertando o usuário que o ponto está SOBRE a circunferência quando o raio calculado for exatamente igual a 1.

As condições são construídas utilizando-se operadores de comparação, sendo os mais comuns o maior (`>`), menor (`<`), maior ou igual (`>=`), menor ou igual (`<=`), igual (`==`) e diferente (`!=`).

> Lembre-se (como visto no capítulo anterior) que o *teste* de igualdade é sempre feito com o uso de *dois* sinais de igual ("==") sucessivos. O uso de apenas um sinal de igual constitui uma instrução válida (atribuição), porém logicamente imprópria nessa situação.

Estas condições podem ser combinadas para produzir condições mais complexas utilizando-se os operadores "E" (`&&`) e "OU" (`||`). O exemplo abaixo solicita dois números e utiliza o operador "E" para verificar em que quadrante se encontram:

Listagem:

```
<script>
var x = parseFloat(prompt("x = "));
var y = parseFloat(prompt("y = "));
if ((x>0)&&(y>0)) alert("1o. quadrante");
if ((x<0)&&(y>0)) alert("2o. quadrante");
if ((x<0)&&(y<0)) alert("3o. quadrante");
if ((x>0)&&(y<0)) alert("4o. quadrante");
</script>
```

O exemplo a seguir verifica se o valor da variável `angulo` é menor que $\pi/2$ ou maior que $3\pi/2$:

Listagem:

```
<script>
var angulo = parseFloat(prompt("ângulo = "));
if ((angulo<Math.PI/2)||(angulo>3/2*Math.PI))
   alert("hemisfério direito");
else
   alert("hemisfério esquerdo");
</script>
```

Muitos testes podem ser feitos na mesma expressão, desde que o resultado seja sempre algo que pode ser interpretado como `false` (falso) ou `true` (verdadeiro). No exemplo a seguir o usuário deve digitar valores para as coordenadas *x*, *y* e *z* que serão utilizadas para calcular a distância a uma carga elementar e o módulo do campo elétrico que ela produz naquele ponto. Se todos os valores forem nulos (`x = y = z = 0`), o valor não pode ser calculado pois haveria um zero no denominador:

Listagem:

```
<script>
var x = parseFloat(prompt("x = "));
var y = parseFloat(prompt("y = "));
var z = parseFloat(prompt("z = "));
if ((x==0)&&(y==0)&&(z==0))
   alert("x, y e z nulos! Operação inválida!");
else
   alert(9e9 * 1.6e-19 / (x*x+y*y+z*z));
</script>
```

Coloque sempre parênteses entorno de cada teste e em torno da expressão final. A não delimitação apropriada de cada condição pode levar a erros lógicos muito difíceis de detetar.

`switch (expressão) case { rotulo: { instruções } ... }`

O `switch ... case` permite a execução de um grupo de instruções quando o valor de uma expressão é igual ao rótulo do grupo de instruções, e é geralmente empregado em situações em que um grande número `if`'s aninhados e complexos poderiam ser utilizados.

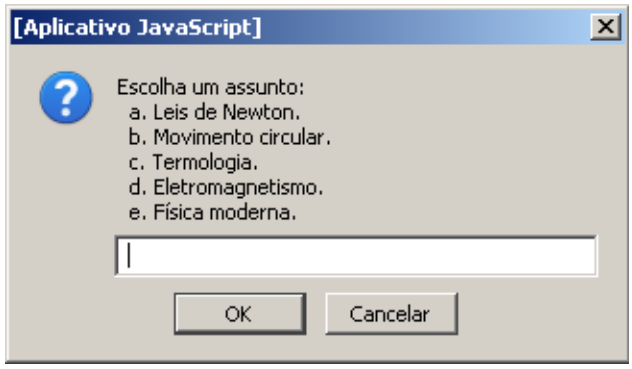
No exemplo a seguir uma questão é acumulada numa variável literal (`str`) que é utilizada para solicitar que o usuário faça uma escolha. Se a letra digitada estiver entre "a" e "e" frases específicas são apresentadas; para qualquer outra escolha será executado o `default` (padrão).

Listagem:

```
<script>
str = "";
str += "Escolha um assunto:\n";
str += "  a. Leis de Newton.\n";
str += "  b. Movimento circular.\n";
str += "  c. Termologia.\n"
str += "  d. Eletromagnetismo.\n"
str += "  e. Física moderna.\n"
```

```
var escolha = prompt(str,"");
switch (escolha) {
   case "a": alert("F = ma\n");
             break;
   case "b": alert("A cos(wt)\n");
             break;
   case "c": alert("Dilatação\n");
             break;
   case "d": alert("Lei de Faraday\n");
             break;
   case "e": alert("Efeito fotoelétrico\n");
             break;
   default : alert("Escolha inválida.\n");
}
</script>
```

Resultado:



Note novamente o uso de um literal inicialmente vazio (`str = ""`) onde vai sendo acumulada a questão utilizando o operador `"+="` (`x += y` é equivalente a `x = x + y`). Note também o uso do caracter de escape `\n` (*newline*), que significa uma quebra de linha dentro do texto para a caixa de alerta (o `<br>` funciona apenas dentro do documento HTML e a caixa de alerta não é um documento HTML).

Um elemento extremamente importante no `switch ... case` é a instrução `break`. Note que cada um dos blocos é terminado por ela pois aqui se deseja que uma letra defina apenas uma escolha. Se o `break` for removido das instruções, todos os blocos a partir do rótulo escolhido serão executados.

Sua vez... (3-3)

Como exercício, retire do script acima as instruções `break` e execute-o extensivamente, escolhendo todas as diferentes opções para observar os resultados. Experimente também colocar instruções `break` em um ou outro caso e observe o resultado.

`for (inicialização; teste; incremento) { instruções }`

O laço `for` é muito útil quando desejamos realizar cálculos um número bem definido de vezes. O exemplo a seguir calcula a posição de um objeto em MRUV para *t* indo de 0 a 10 em intervalos de 1 segundo:

Listagem:

```
<script>
var xo = 0;
var vo = 0;
var a  = 10;
for (var t=0; t<=10; t=t+1) {
   x = xo + vo*t + a*t*t/2;
   document.write("<i>t</i> = " + t.toFixed(2) + " ");
   document.write("<i>x</i> = " + x.toFixed(2) + "<br>");
}
</script>
```

Resultado:

t = 0.00 *x* = 0.00
t = 1.00 *x* = 5.00
t = 2.00 *x* = 20.00
t = 3.00 *x* = 45.00
t = 4.00 *x* = 80.00
t = 5.00 *x* = 125.00
t = 6.00 *x* = 180.00
t = 7.00 *x* = 245.00
t = 8.00 *x* = 320.00
t = 9.00 *x* = 405.00
t = 10.00 *x* = 500.00

Sua vez... (3-4)

Modifique o exemplo anterior de modo que calcule e imprima a posição e a velocidade a cada 0.5 segundos e inclua as unidades das grandezas:

t = 0.00 s, *x* = 0.00 m, *v* = 0.00 m/s.
t = 0.50 s, *x* = 1.25 m, *v* = 5.00 m/s.
t = 1.00 s, *x* = 5.00 m, *v* = 10.00 m/s.
t = 1.50 s, *x* = 11.25 m, *v* = 15.00 m/s.
...

Assim como no `if`, no laço `for` as chaves não são obrigatórias caso o bloco de instruções contenha apenas uma instrução.

Laços `for` podem ser aninhados, como no exemplo abaixo, que coloca nos elementos de uma matriz $n \times m$ os valores $a_{i,j} = i + j$.

Listagem:

```
<script>
var n = 3;
var m = 4;
for (var i=1; i<=n; i++) {
   for (var j=1; j<=m; j++) {
      document.write(i+j + " ");
   }
   document.write("<br>");
}
</script>
```

Resultado:

```
2 3 4 5
3 4 5 6
4 5 6 7
```

Note o uso das contrações `i++` e `j++`, que são equivalentes a `i = i + 1` e `j = j + 1`, respectivamente. Note também o uso de um espaço em branco (`" "`) na instrução `document.write`, utilizado para separar os elementos da matriz.

Sua vez... (3-5)

Modifique o exemplo anterior de modo que passe a imprimir a matriz transposta:

```
2 3 4
3 4 5
4 5 6
5 6 7
```

Apesar de serem em geral utilizados para executar um conjunto de instruções um número definido de vezes, é possível "escapar" do laço em qualquer iteração utilizando o comando `break`. No script a seguir o exemplo do MRUV é modificado para interromper o laço caso a posição do objeto seja maior que 100:

Listagem:

```
<script>
var xo = 0;
var vo = 0;
var a  = 10;
for (var t=0; t<=10; t=t+0.5) {
   x = xo + vo*t + a*t*t/2;
   if (x>100) break;
   document.write("<i>t</i> = " + t.toFixed(2) + " ");
```

```
   document.write("<i>x</i> = " + x.toFixed(2) + "<br>");
}
</script>
```

Resultado:

$t = 0.00\ x = 0.00$
$t = 0.50\ x = 1.25$
$t = 1.00\ x = 5.00$
$t = 1.50\ x = 11.25$
$t = 2.00\ x = 20.00$
$t = 2.50\ x = 31.25$
$t = 3.00\ x = 45.00$
$t = 3.50\ x = 61.25$
$t = 4.00\ x = 80.00$

O uso deste tipo de recurso em um laço `for` não é recomendável pois implica em um teste adicional que, como veremos a seguir, pode ser eficientemente substituído por outras estruturas de controle de fluxo.

`while (condição) { instruções }`

O laço `while` é utilizado quando um grupo de instruções deve ser executado um número não pré-definido de vezes, mas enquanto uma condição for verdadeira. O script a seguir calcula as posições de uma partícula em queda livre a partir de uma posição especificada enquanto sua coordenada *y* for maior do que zero:

Listagem:

```
<script>
var yo = parseFloat(prompt("Posição inicial: ","2"));
var t = 0;
var y = yo;
var vo = 0;
var a = -9.8;
var dt = 0.1;
while (y>0) {
   y = yo + vo*t + a*t*t/2;
   document.write("<i>t</i> = " + t.toFixed(2) + " ");
   document.write("<i>y</i> = " + y.toFixed(2) + "<br>");
   t = t + dt;
   }
</script>
```

Resultado:

$t = 0.00\ y = 2.00$
$t = 0.10\ y = 1.95$
$t = 0.20\ y = 1.80$
$t = 0.30\ y = 1.56$

$t = 0.40$ $y = 1.22$
$t = 0.50$ $y = 0.77$
$t = 0.60$ $y = 0.24$
$t = 0.70$ $y = -0.40$

Note que, como a condição de fim é testada *antes* do cálculo da coordenada, o script imprime o último valor calculado, que é menor que zero, algo que pode ser indesejável em certas situações. Pode também acontecer que o bloco nunca venha a ser executado caso a condição resulte em falsa logo no primeiro teste.

Sua vez... (3-6)

Modifique o exemplo anterior de modo que a partícula saia sempre do solo (y = 0 m), com uma velocidade positiva ($v_0 > 0$) informada pelo usuário, e que interrompa a execução logo após a partícula atingir a altura máxima da sua trajetória. *Dica:* uma das características do máximo da trajetória é a mudança do sinal da velocidade. Assim, se além da posição você calcular também a velocidade, poderá testar se essa é positiva ou negativa. Quando passar a ser negativa, significa que a partícula passou pelo máximo.

do { instruções } while (condição);

Muito semelhante ao laço `while`, o laço `do` também executa o bloco de instruções enquanto a condição for verdadeira, mas o teste é feito *depois* da execução do bloco. Isto significa que o bloco *sempre* será executado pelo menos uma vez. O script a seguir faz o mesmo que o anterior, mas com o laço `do`. Note que se o usuário digitasse um número negativo no exemplo anterior, nenhum resultado seria mostrado, enquanto neste seria mostrada a posição inicial em $t = 0$.

Listagem:

```
<script>
var yo = parseFloat(prompt("Posição inicial: ","2"));
var t = 0;
var y = yo;
var vo = 0;
var a = -9.8;
var dt = 0.1;
do {
   y = yo + vo*t + a*t*t/2;
   document.write("<i>t</i> = " + t.toFixed(2) + " ");
   document.write("<i>y</i> = " + y.toFixed(2) + "<br>");
   t = t + dt;
   } while (y>0);
</script>
```

Exercícios

1. Utilize laços `while`, `do ... while` e `for` para fazer scripts que calculem e mostrem os números inteiros entre 0 e 10, inclusive.

2. A equação de um oscilador harmônico amortecido com constante elástica k, massa m e constante de amortecimento b é:

$$x(t) = A\cos(2\pi f\, t)\, e^{-(b/2m)t}$$

   onde A é a amplitude e $f = [k/m - (b/2m)^2]^{1/2}$ a frequência. Utilize laços `while`, `do ... while` e `for` para fazer scripts que calculem a posição do oscilador em função do tempo para 20 pontos entre 0 e 2 segundos, em intervalos de 0.1 segundos, para A = 0.10 m, f = 2 Hz e $b/2m$ = 2.

3. Utilize laços `while` e `do ... while` para fazer um script que solicite ao usuário a altura em que um objeto é liberado em queda livre, a partir do repouso, e calcule posições sucessivas em intervalos de 0.1 s até que atinja o solo.

4. Faça um script que solicite ao usuário que escolha uma função dentre uma lista (seno, cosseno, tangente, raiz quadrada, exponencial, por exemplo) e a seguir um número que servirá de argumento. Utilize a estrutura `switch ... case` para realizar e apresentar o valor do cálculo da função para o argumento dado.

5. Refaça o exercício anterior utilizando `if ... else` aninhados.