CHAPTER 7

Conditionals

Conditional statements are used to execute different code blocks based on different conditions.

If statement

The if statement will only execute if the condition inside the parentheses is evaluated to true. The condition can include any of the comparison and logical operators.

```
int x = new System.Random().Next(3); // gives 0, 1 or 2

if (x < 1) {
  System.Console.Write(x + " < 1");
}
```

To test for other conditions, the if statement can be extended by any number of else if clauses. Each additional condition will only be tested if all previous conditions are false.

```
else if (x > 1) {
  System.Console.Write(x + " > 1");
}
```

The if statement can have one else clause at the end, which will execute if all previous conditions are false.

```
else {
  System.Console.Write(x + " == 1");
}
```

As for the curly brackets, they can be left out if only a single statement needs to be executed conditionally.

```
if (x < 1)
  System.Console.Write(x + " < 1");
else if (x > 1)
  System.Console.Write(x + " > 1");
else
  System.Console.Write(x + " == 1");
```

Switch statement

The switch statement checks for equality between either an integer or a string and a series of case labels, and then passes execution to the matching case. The statement can contain any number of case clauses and may end with a default label for handling all other cases.

```
int x = new System.Random().Next(3); // gives 0, 1 or 2

switch (x)
{
  case 0: System.Console.Write(x + " is 0"); break;
  case 1: System.Console.Write(x + " is 1"); break;
  default:System.Console.Write(x + " is 2"); break;
}
```

Note that the statements after each case label are not surrounded by curly brackets. Instead, the statements end with the `break` keyword to break out of the switch. Unlike many other languages, case clauses in C# must end with a jump statement, such as break. This means that the `break` keyword cannot be left out to allow the execution to fall-through to the next label. The reason for this is that unintentional fall-throughs is a common programming error.

Goto statement

To cause a fall-through to occur, this behavior has to be explicitly specified using the goto jump statement followed by a case label. This will cause the execution to jump to that label.

```
case 0: goto case 1;
```

Goto may be used outside of switches to jump to a label within the same method's scope. Control may then be transferred out of a nested scope, but not into a nested scope. However, using goto in this manner is discouraged since it can become difficult to follow the flow of execution.

```
goto myLabel;
// ...
myLabel:
```

Ternary operator

In addition to the if and switch statements there is the ternary operator (`?:`). This operator can replace a single if/else clause that assigns a value to a specific variable. The operator takes three expressions. If the first one is evaluated to true then the second expression is returned, and if it is false, the third one is returned.

```
// Value between 0.0 and 1.0
double x = new System.Random().NextDouble();

x = (x < 0.5) ? 0 : 1; // ternary operator (?:)
```