C# Basics

Boolean Operations

Boolean Operations

- Operations that combine and compare bools
  - ! The NOT Operator
  - && The AND Operator
  - || The OR Operator

Boolean Operations

- ! The NOT Operator
  - Pronounced either "not" or "bang"
  - Reverses value of the bool

```
Console.WriteLine ( !true );    // Outputs: false
Console.WriteLine ( !false );   // Outputs: true
Console.WriteLine ( !(!true) ); // Outputs: true (the double negative of true)
```

  - Also called the "logical negation operator"
    - This differentiates it from ~, the bitwise not operator

Boolean Operations

- && The AND Operator
  - Returns true only if both operands are true

```
Console.WriteLine ( false && false );    // false
Console.WriteLine ( false && true  );    // false
Console.WriteLine ( true  && false );    // false
Console.WriteLine ( true  && true  );    // true
```

Boolean Operations

- || The OR Operator
  - Returns true if either operand is true

```
Console.WriteLine ( false && false );    // false
Console.WriteLine ( false && true  );    // true
Console.WriteLine ( true  && false );    // true
Console.WriteLine ( true  && true  );    // true
```

  - | (the pipe) is Shift-Backslash
    - Just above the return or enter key on a US keyboard

Comparison Operators

- Allow the comparison of two values
- Return a bool (either true or false)

==	Is Equal To
!=	Not Equal To
>	Greater Than
<	Less Than
>=	Greater Than or Equal To
<=	Less Than or Equal To

COMPARISON BY VALUE OR REFERENCE

- Simple variables are compared by value
  - bool, int, float, char, string
- More complex variables are compared by reference
  - When variables are compared by reference, the comparison is not of their internal values but of whether they point to the same location in memory
  - C# classes you write
  - Detailed discussion will be in the future modules

Comparison Operators

- == Is Equal To
  - Returns true if the values or references compared are equivalent

```
Console.WriteLine ( 10 == 10 );          // Outputs: True
Console.WriteLine ( 20 == 10 );          // Outputs: False
Console.WriteLine ( 1.23f == 3.14f );    // Outputs: False
Console.WriteLine ( 1.23f == 1.23f );    // Outputs: True
Console.WriteLine ( 3.14f == Math.PI ); // Outputs: False
// Math.PI has more decimal places than 3.14f
```

  - Do NOT confuse == and =

==	The *comparison* operator
=	The *assignment* operator

Comparison Operators

- != Not Equal To
  - Returns true if the values or references compared are NOT equivalent

```
Console.WriteLine ( 10 != 10 );          // Outputs: False
Console.WriteLine ( 20 != 10 );          // Outputs: True
Console.WriteLine ( 1.23f != 3.14f );    // Outputs: True
Console.WriteLine ( 1.23f != 1.23f );    // Outputs: False
Console.WriteLine ( 3.14f != Math.PI ); // Outputs: True
```

Comparison Operators

- > Greater Than
  - Returns true if the first operand is greater than the second

```
Console.WriteLine ( 10 > 10 );          // Outputs: False
Console.WriteLine ( 20 > 10 );          // Outputs: True
Console.WriteLine ( 1.23f > 3.14f );    // Outputs: False
Console.WriteLine ( 1.23f > 1.23f );    // Outputs: False
Console.WriteLine ( 3.14f > 1.23f );    // Outputs: True
```

- < Less Than
  - Returns true if the first operand is less than the second

```
Console.WriteLine ( 10 < 10 );          // Outputs: True
Console.WriteLine ( 20 < 10 );          // Outputs: False
Console.WriteLine ( 1.23f < 3.14f );    // Outputs: True
Console.WriteLine ( 1.23f < 1.23f );    // Outputs: True
Console.WriteLine ( 3.14f < 1.23f );    // Outputs: False
```

Comparison Operators

- >= Greater Than or Equal To
  - True if the 1st operand is greater than or equal to the 2nd

```
Console.WriteLine ( 10 >= 10 );          // Outputs: True
Console.WriteLine ( 20 >= 10 );          // Outputs: True
Console.WriteLine ( 1.23f >= 3.14f );    // Outputs: False
Console.WriteLine ( 1.23f >= 1.23f );    // Outputs: True
Console.WriteLine ( 3.14f >= 1.23f );    // Outputs: True
```

- <= Less Than or Equal To
  - True if the 1st operand is less than or equal to the 2nd

```
Console.WriteLine ( 10 <= 10 );          // Outputs: True
Console.WriteLine ( 20 <= 10 );          // Outputs: False
Console.WriteLine ( 1.23f <= 3.14f );    // Outputs: True
Console.WriteLine ( 1.23f <= 1.23f );    // Outputs: True
Console.WriteLine ( 3.14f <= 1.23f );    // Outputs: False
```

Conditional Statements

- Control Flow Within Your Programs

  `if`

  `if / else`

  `if / else if / else`

  `switch`

- Can be combined with Boolean operations
- Make use of *braces* { }

Conditional Statements

- If - Performs code within braces if the argument within parentheses is true

```
if (true) {
    print( "This line will print." );
}

if (false) {
    print( "This line will NOT print." );
}

// The output of this code will be:
//    This line will print.
```

- All the code within the braces of the if statement executes

Conditional Statements

- Combining if statements with boolean operations

```
bool night = true;
bool fullMoon = false;

if (night) {
        Console.WriteLine ( "It's night." );
}

if (!fullMoon) {
        Console.WriteLine t( "The moon is not full." );
}


if (night && fullMoon) {
        Console.WriteLine ( "Beware werewolves!!!" );
}


if (night && !fullMoon) {
        Console.WriteLine ( "No werewolves tonight. (Whew!)" );
}

// The output of this code will be:
//     It's night.
//     The moon is not full.
//     No werewolves tonight. (Whew!)
```

Conditional Statements

- Combining if statements with comparison operators

```
if (10 == 10 ) {
 Console.WriteLine( "10 is equal to 10." );
}

if ( 10 > 20 ) {
 Console.WriteLine( "10 is greater than 20." );
}

if ( 1.23f <= 3.14f ) {
 Console.WriteLine( "1.23 is less than or equal to 3.14." );
}

if ( 1.23f >= 1.23f ) {
 Console.WriteLine( "1.23 is greater than or equal to 1.23." );
}

if ( 3.14f != Math.PI ) {
 Console.WriteLine( "3.14 is not equal to "+Math.PI+"." );
    // + can be used to concatenate strings with other data types.
    // When this happens, the other data is converted to a string.
}
```

- Don't accidentally use = in an if statement!!!

Conditional Statements

- if / else
  - Performs one action if true, and another if false

```
bool night = false;

if (night) {
    print( "It's night." );
} else {
    print( "What are you worried about?" );
}

// The output of this code will be:
//     What are you worried about?
```

Conditional Statements

- if / else if / else
  - Possible to chain several else if clauses

```
bool night = true;
bool fullMoon = true;

if (!night) {                // Condition 1 (false)
 Console.WriteLine("It's daytime. What are you worried about?");
} else if (fullMoon) {       // Condition 2 (true)
 Console.WriteLine( "Beware werewolves!!!" );
} else {                     // Condition 3 (not checked)
 Console.WriteLine( "It's night, but the moon is not full." );
}

// The output of this code will be:
//     Beware werewolves!!!
```

Conditional Statements

- Nested if statements

```
bool night = true;
bool fullMoon = false;

if (!night) {
   Console.WriteLine( "It’s daytime. Why are you worried about?" );
} else {
    if (fullMoon) {
           Console.WriteLine( "Beware werewolves!!!" );
    } else {
           Console.WriteLine( "It's night, but the moon isn't full." );
    }
}

// The output of this code will be:
//     It's night, but the moon isn't full.
```

Conditional Statements

- Switch: alternative to several if statements
  - Can only compare for equality
  - Can only compare against a single variable against literals

```
int num = 3;
switch (num) {  // The variable in parentheses is being compared
case (0):  // Each case is a literal that is compared against num
    Console.WriteLine( "The number is zero." );
     break;  // Each case must end with a break statement.
case (1):
    Console.WriteLine( "The number is one." );
     break;
case (2):
    Console.WriteLine( "The number is two." );
     break;
default:  // If none of the other cases are true, default will happen
    Console.WriteLine( "The number is more than a couple." );
     break;
}  // The switch statement ends with a closing brace.


// The output of this code is: The number is more than a couple.
```

Conditional Statements

- Switch can "fall through" to other cases

```
int num = 3;
switch (num) {
case (0):
    print( "The number is zero." );
    break;
case (1):
    print( "The number is one." );
    break;
case (2):
    print( "The number is a couple." );
    break;
case (3):                          // case (3) falls through to case (4)
case (4):                          // case (4) falls through to case (5)
case (5):
    print( "The number is a few." );
    break;
default:
    print( "The number is more than a few." );
    break;
}

// The output of this code is: The number is a few.
```