TeenCoder™: Windows Programming
Second Edition Errata Sheet

Updated December 12th, 2011

This document lists the known typographical or other corrections to the *TeenCoder™: Windows Programming* Second Edition course.

- Early printings of the Student Textbook contain this line at the bottom of page 89 when discussing **for** loops:

  "If we didn't have the **break** statement our **for** loop would continue until the end and we'd actually find the last space instead (at index 7)."

  The line should specify index 8, which is the index of the second space, as follows:

  "If we didn't have the **break** statement our **for** loop would continue until the end and we'd actually find the last space instead (at index 8)."

- In first and second printings, the "Assigning Variables" discussion starting on page 69 does not give any examples of assignment to the **float** or **decimal** data types. However the student is asked to make assignments to these data types in the chapter activity on page 78.

  By default any literal real number you type into source code such as "1.234" is considered a **double** data type. So you can freely assign these literals to a **double** variable like this:

```
double myDouble = 1.234;
```

  However if you attempt to assign "1.234" directly to a **float** or **decimal** data type, you will receive an error because the compiler is treating "1.234" as a **double**. To let the compiler know specifically that you would like "1.234" to be a **float**, append an "F" to the literal like this:

```
float myFloat = 1.234F;
```

  Similarly, to assign the literal "1.234" to a **decimal** data type, append an "M" to the value like this:

```
decimal myDecimal = 1.234M;
```

  Equivalently, you could cast the original double to the target data type:

```
float myFloat = (float)1.234;
decimal myDecimal = (decimal)1.234;
```

- In first and second printings, the discussion about comparing reference variables on page 84 incorrectly states that the data in string reference variables cannot be compared with the comparison operators. You should not in general compare reference variables, but **strings** are a special case. Here is the re-written section:

"You do not want to use comparison operators to compare two reference variables! Reference variables only point to areas in memory that hold some data. Comparing reference variables will only let you know if the variables are pointing to the exact same copy of the data, but will not tell you if the data itself is the same in two different areas! So a comparison like this would not work as expected:

```
myReference == yourReference     // Not what you expect!
```

If your references pointed to two exact copies of the same data, the comparison operator "==" would still return **false** because the address of the data in memory is different for each reference! C# has a special rule for **string** references because they are so commonly used. Comparison of **string** references will actually compare the data in the strings, but it's still clearer to use the **string**.**Equals**() function as described earlier.

Do NOT use comparison operators to compare two references! Only for strings will the comparison operators work as expected, and it's generally clearer to use the string.Equals() function to compare strings.