C# Programming Homework 02

Chapter 02, C# Step by Step

Readings

Read chapter 2 in the *C# Step by Step* book.

Discussion Questions

Answer the discussion questions in writing for chapter 2.

1. What is a *local* variable? A variable that exists only in a method, function, or another small section/block of code.
2. What is a *statement*? A command that performs an action, such as calculating a value and storing the result or displaying a message to a user. An **expression** doesn't execute a command; it returns a mathematical value.
3. What i s an *identifier*? The name that you use to identify elements in your programs, such as namespaces, classes, methods, and variables.
4. What is a *keyword*? 77 identifiers the C# language reserves for its own use that you cannot reuse for your own purposes.
5. What is a *variable*? A variable is a named memory location that may hold a value.
6. How do you declare a variable?
   How do you assign a value to a variable? You declare the type and name of a variable in a declaration statement.
   Can you have a variable that does not have a value? No, it will contain a random value until you assign a value to it. You must assign a value to a variable before you can use it; otherwise, your program will not compile.
   Can you declare and initialize a variable in a single statement? Yes
7. What does Visual Studio do when you open a project (a `.csproj` file) without opening the solution (a `.sln` file)? If you open a project rather than a solution, Visual Studio 2017 automatically creates a new solution file for it.
   Why might this be a problem? It can result in you accidentally generating multiple solutions for the same project.
8. How are arithmetic operators and variable types related? The operators that you can use on a value depend on the value's type.
9. How do you turn an integer into a string? Use the ToString method of the variable object to generate a string version of the value of the variable.
10. How do you turn a string into an integer? By using the ~~showintvalue~~ int32.parse (int.parse) method.
11. What is the difference between precedence and associativity? Precedence governs the order in which an expression's operators are evaluated and has vertical order. In C#, the multiplicative operators (*, /, and %) have precedence over the additive operators (+ and –). Associativity is the direction (left or right horizontal order) in which the operands of an operator are evaluated.
12. What is the *definite assignment rule*? You must assign a value to a variable before you can use it; otherwise, your program will not compile.
13. How are t he prefix and postfix increment and decrement operators evaluated differently? When using a postfix the decrementing is done after the variable is evaluated. The way to remember which operand does what is to look at the order of the elements (the operand and the operator) in a prefix or postfix expression.
14. What is *string interpolation*? String interpolation concatenates strings and renders many uses of the + operator obsolete. It allows you to evaluate an expression in a string. It is more efficient than using the + operator and is also arguably more readable and less
    error-prone.
15. What does the *var* keyword do? The var keyword causes the compiler to deduce the type of the variables from the types of the expressions used to initialize them. This is "implicit typing".