Steven Myers

30 October 2016

CSCI-A290 Windows Programming

1. **Variable**

**Links:** <https://msdn.microsoft.com/en-us/library/ms173104.aspx>, <https://www.tutorialspoint.com/csharp/csharp_variables.htm>, <http://csharp.net-tutorials.com/basics/variables/>

**Precis:**

”There are several basic value types provided in C#” including integral, floating point, decimal, Boolean, and nullable types. ­The correct syntax for declaring variables in C# is “<data\_type> <variable\_list>;”. You can then set a variable name equal to a value. You can also set the visibility of variables with “private” to clarify that the variable may only be visible within the scope of the current class.

**Summary:**

I think the most important thing about the articles regarding variables is that you should be careful about distinguishing between private and public variables. If you have information that shouldn’t be accessible to other parts of the program, you should use private variables when possible.

2. **Type**

**Links:** <https://msdn.microsoft.com/en-us/library/ya5y69ds.aspx>, <https://www.tutorialspoint.com/csharp/csharp_type_conversion.htm>,

<https://www.tutorialspoint.com/csharp/csharp_data_types.htm>

**Precis:**

“Variables in C# are categorized” into the following 3 type categories: value, reference, and pointer types. When you declare a variable a certain type, “the system allocates memory to store the value.” The following types are the ones available to C#: (bool, byte, sbyte, char, decimal, double, float, int, uint, long, ulong, object, short, ushort, string). You can convert data types to other data types. There are two forms of type conversion – “implicit type conversion” and “explicit type conversion.” This is also known as “type casting.” Most data types have built in conversion methods, like “ToBoolean, ToByte, toString.”

**Summary:**

I think understanding type conversions is very important, since there will almost always be a scenario where you need to at least convert a number to a string, or a string to a number. For example, if we are trying to calculate and display a date given a string of the current date, it’s more than likely we’re going to need to parse that string and convert it to an integer expression. It’s also incredibly important to understand what operations you can perform on certain data types. For example, you can’t do mathematical addition on two strings.

3. **Procedure**

**Links:** <http://www.c-sharpcorner.com/uploadfile/skumaar_mca/differences-between-procedures-and-functions/>, <https://msdn.microsoft.com/en-us/library/d7125bke.aspx>, <http://csharp-station.com/Tutorial/AdoDotNet/Lesson07>

**Precis: \*All search results for c# and procedures pointed at ADO.NET/SQL statements, so that’s the definition of procedure I am writing for.**

Procedures in C# most commonly refer to stored procedures, for SQL statements. “A stored procedure[s] is a pre-defined, reusable routine that is stored in a database.” This helps optimize performance for SQL Server queries, since these procedures are predefined and compiled on the server. That way, these queries aren’t built at runtime (dynamically) and are precompiled. Stored procedures are an “important part of the ADO.NET libraries”. Some differences between ‘methods’ and ‘procedures’ are that stored procedures are “used to perform specific tasks” but functions/methods are often used “for computing values.” Stored procedures “may or may not return values” but a function should return a value.

**Summary:**

The article from c-sharpcorner.com seemed to accurately depict the differences between normal functions and procedures when writing SQL commands. Procedures will always yield a large dataset of several rows of data, where as functions will simply narrow down values to be used in procedures. This makes sense because we want to use our predefined, compiled queries (faster and optimized for grabbing the data we want) with parameters generated by functions. I think the most important thing I’ve learned from these articles is that “procedures” have the strict connotation of SQL stored procedures.

4. **Method**

**Links:** <https://msdn.microsoft.com/en-us/library/ms173114.aspx>, <https://www.tutorialspoint.com/csharp/csharp_methods.htm>, <http://csharp-station.com/Tutorial/CSharp/Lesson05>

**Precis:**

“A method is a code block that contains a series of statements.” Every instruction in C# is executed by a method, and the main method of every C# program is called by the CLR when a program is started. Every method is declared within a class or struct with “access levels,” “optional modifiers”, the “return value”, “name of the method”, and “any method parameters.” This constitutes the signature of a method. An example is “public void foo(int someParam).” In order to call a method from non-static scope, you must create a new instance of the class where the method is stored. If we have a class Foobar, and it owns a method “doWork”, we must declare a Foobar object and instantiate with the new keyword. You can then call doWork from the newly instantiated Foobar object. “When a method is static, there are no instances of that method and you can invoke only that one definition of the static method.”

**Summary:**

I think the most important thing about methods is understanding the concept that every program has a main method, and that’s where the program is driven. This is where you instantiate classes and call other methods. In addition, I think it’s important to understand when to create static functions and what they do. Static functions should only be created whenever you have an operation to calculate some value that is independent of any fields/properties of a particular class. For example, if we defined a function to find the distance between two Cartesian points, we could make it a static method so that it could be called in other programs freely.

5. **Event**

**Links:** <https://www.tutorialspoint.com/csharp/csharp_events.htm>, <https://msdn.microsoft.com/en-us/library/aa645739(v=vs.71).aspx>, <https://www.intertech.com/Blog/c-sharp-tutorial-understanding-c-events/>

**Precis:**

“An event in C# is a way for a class to provide notifications to clients of that class when something happens to an object.” The most common place for use of events is graphical user interfaces for reporting mouse and keyboard events. “However, they need not be used only for graphical user interfaces.” Events are just simply signals to state changes and can be used all different kinds of applications. In order to declare an event, you must use delegates which are “objects that encapsulate methods to be called anonymously.” We declare delegates in a method modifier, like “public delegate string …”. The event itself is called using “the event keyword” like so “event SomeDelegate myEventName.”

**Summary:**

As someone who is somewhat familiar with asynchronous programming in JavaScript, the event/delegate system appears to be .NET’s solution to async programming by partitioning it off with its own distinct keywords. It’s the idea that events can occur throughout the program, and we need to teach our program how to deal with this events or “handle them.” It’s useful because we can plan proactive methods to handle user input.