**argument promotion**

Implicit conversion of data types to match other data types in an expression, to match parameter types for a method, and so on.

**array**

A simple collection of data that is provided by most programming languages that provides direct access to data, however is limited to a fixed size.

**base case**

The known solution in a recursive function, e.g., for Factorial the base case is 0! = 1.

**casting**

Explicit conversion of a data type to another data type.

**collection**

A general term used to refer to a grouping of objects in a program, such as an array or the generic collections provided by Microsoft.Net.

**command line arguments**

The data/values that are written on the command line after the name of the program's executable file, which are provided in a C# program as a parameter to the Main method.

**constant**

A constant value is an element of data whose value never changes, e.g., pi (π).

**entry-point**

The location in your program where execution begins, i.e., the Main method for C#.

**general case**

The statement/s in a recursive method which reduce the problem one step closer to the base case and invoke the method again (the recursion call).

**generic collection**

A collection provided by the Microsoft.Net framework that exploits generics (hence generic collection) and various data structures and algorithms to provide storage for a collection.

**input parameters**

Variables or literals that are passed as input (for reading) to a method, that is pass-by-value.

**jagged array**

A multi-dimensional array where the number of elements is not constant, e.g., an array containing three rows of three, five, and four columns each.

**multi-dimensional array**

An array that consists of more than one-dimension, i.e., a two-dimensional array has rows and columns, a three-dimensional array has rows, columns, and planes, etc. Each dimension is constant for that array, i.e., each row has the same number of columns, and so on.

**named parameters**

Refers to the ability to specify which parameter is being specified explicitly in a method invocation, e.g., methodCall(paramOne: valueOne, paramThree: valueThree, paramTwo: valueTwo).

**one-dimensional array**

A simple array containing some number of elements in a sequence.

**optional parameters**

Parameters specified in a method's parameter list that may be omitted when invoking the method, whereby a default value will be used instead.

**output parameters**

Variables that are passed to a method to receive output/results from the method that are pass-by-reference.

**pass-by-reference**

A variable that is passed to a method whereby the variable itself can be used by the method, i.e., any changes made to the parameter's value by the method persist in the original variable after the method terminates.

**pass-by-value**

A literal or variable whose value is passed to a method as a copy, i.e., any changes made to the parameter's value by the method will not persist beyond the method's termination.

**recursion**

An alternative to iteration whereby a method includes an invocation to itself (method calls itself).

**reference parameters**

Variables that are passed as input (for reading) to a method, but can also optionally receive output/results form the method (pass-by-reference).

**scope**

A fundamental concept of programming defining the regions or sections of a program in which a name (variable name, method name, etc.) can be used without requiring some additional qualification.

**static members**

Members (variables, properties, and methods) that are defined in a class and are part of that class, as opposed to instance members which are also defined in a class but are part of the instances/objects created from that class.

**static methods**

A method that has been defined as a static member.

**static properties**

A property that has been defined as a static member.

**static variables**

A variable that has been defined as a static member.

**validity checking**

Generically refers to checking user input to make sure that it is valid. Validity can be simple such as making sure numeric data is entered and/or in the correct range, however for reasons of security validity can be much more complex.