8.1 Introduction

In this chapter we look at two data-oriented features of the C# language. The first is the struct construct, used for representing data-only structures. The second is enumerations, which are useful for defining specific sets of values that may have a specific order associated with them.

8.2 Structs in C#

In C#, as well as being able to define classes (reference types), it is also possible to define value types (called structs). A value type is a data structure that is defined on the stack (in a similar manner to an int or float), whereas reference types are declared on the heap. In turn, variables that hold objects hold a reference to them, whereas variables that hold value types hold the value directly.

In general you would normally use structs (value types) for elements that are essentially just data, whereas virtually everything else would be defined via classes. Structs also have the advantage of being more efficient than classes (as they are on the stack and accessed directly rather than via a reference).

However, structs have a number of limitations on their behaviour. Firstly, you cannot use them in inheritance. That is, a struct cannot extend any class or struct, nor can any class extend a struct. Structs behave as though they extend the class Object; however, in practice they do not (although they do automatically extend the class System.ValueType, but this is hidden from you).

The syntax for a struct is:

```
access-modifier struct nameOfStruct {
    ...
    struct details ...
}
```
For example, the following listing defines a struct for holding addresses:

```csharp
using System;

public struct Address {
    public int number;
    public string street;
    public string town;
    public string postCode;

    public Address(int n, string s, string t, string p) {
        number = n;
        street = s;
        town = t;
        postCode = p;
    }

    public override string ToString() {
        string result = "";
        result += number + ", " + street + ", " + town + ", " + postCode;
        return result;
    }
}

public class Test {
    public static void Main() {
        Address a = new Address(10, "High Street", "Bath", "BA1 3ER");
        Console.WriteLine(a);
    }
}
```

The result of compiling and running this program is:

```
C:\c#>Address
10, High Street, Bath, BA1 3ER
```

This program defines a struct that holds an `int` (for the house number) and three strings for the street name, the town and the postcode. It also defines a `ToString` method so that it can be printed in a tidy manner. The struct also defines a single constructor that takes an initial value for the four variables.

Structs are created using the keyword `new`, as for classes. Thus the `Main` method creates a new struct and then prints out the result.