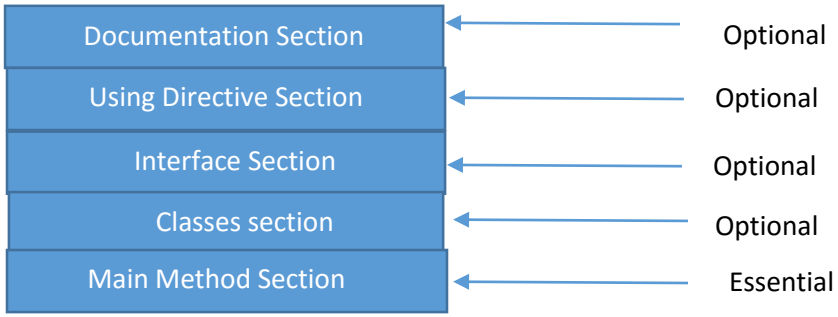
PROGRAM STRUCTURE



An executable c# program may contain a number of coding blocks. The documentation section consists of a set of comments giving name of the program, the author, date and other details, which the programmer may like to use at a later storage.

The using directive section will include all those namespaces that contain classes required by the application. Using directives tell the compiler to look in the namespace specified for these unresolved classes.

An interface is similar to a class but contain only abstract members. Interfaces are used when we want to implement the concept of multiple inheritance in a program.

A c# program may contain multiple class definition. Classes are the primary and essential elements of a C# program. These classes are used to map the objects of real world problems. The number of the classes depends on the complexity of the problem.

Since every c# application program requires a **Main** method as the starting point, the class containing the main is the essential part of the program. A simple C# program may contain only this part. The main method create objects of various classes and establishes communication between them. On reaching the end of Main, the program terminates and the control passes back to the operating system.

COMMAND LINE ARGUMENTS

Command line arguments are parameters supplied to the main method at the time of invoking it for execution. The below program accept name from the command line and writes it to the console.

```
/*this program uses command line argument as input*/
using System;
class sample
{
        public static void Main(string [  ] args )
       {
              Console.Write("welcome to ");
               Console.Write(" " +args [ 0 ]);
             Console.WriteLine(" " +args[ 1] ):;
         }
 }
```

Main method is declared with a parameter **args.** The parameter is declared as an array of strings known as string objects. Any arguments provided in the command line at the time of execution are passed to

the array **args** as its element. We can access the array elements by using subscript like **args** [0], **args**[1] and so on. For example consider the command line.

sample c sharp

This command line contain two arguments which are assigned to the array as follows. Note that c# subscript begin with zero and not one.

C → args[0]
Sharp → args[1]

We will get the following output.

Welcome to c Sharp

Notice the use of a new output method

Console.Write( )

Its only difference from the **WriteLine( )** method is that **Write( )** does not causes a line break and therefore the next output statement is printed on the same line.

COMPILE TIME ERRORS

A program is never totally error-free

- Types of errors:
  - Syntax Errors
  - Logic Errors

Syntax errors will be caught by the compiler. Logical errors should be eliminated by testing the program logic carefully. When the compiler cannot interpret what we are attempting to convey through our code the result is syntax error.

- Example:

```
//Program with syntax errors

using Systom;

class SampleTen
{
        public static void main()
        {
        Console.Write("Hello");
        }
}
```

Program when compiled,will display the following output:

Errors.cs(2.7): error cs0234: The type or namespace name 'Systom' does not exists in the class or namespace

The error message contains the following information:

1. Name of the file being compiled(Errors.cs)
2. Line number & column position of the error(2.7)
3. Error code as defined by the compiler (cs0234)
4. Short description of error