Structure and Enumeration

C# allows us to define our own complex value types based on these simple data types.

These are two sorts of value types we can define in C#:

- Structure
- Enumerations

Value type variables store their data on the stack and therefore the structure and enumeration are stored on the stack.

Structure

A structure in C# provides a unique way of packing together data of different types.

Advantages:

• A structure provides a unique way of packing together data of different types.

• It creates a template that can be used to define its data properties.

• Once the structure type has been defined we can create variables of that type.

• They are instantly and automatically deallocated once they go out of the scope.

Defining struct:

struct keyword is used to create structure. Structure creates a template that may be used to define its data properties.

Syntax:

```
struct  struct_name
{
Data-member-1;
Data-member-1;
-------
-------
}
```

Example:

```
struct student
{
    public  string name;
    public  int  RollNumber;
    public double  totalmks;
}
```

These members are called as member or field or elements.

Once the structure has been defined, we can create variables of that type using declarations that are similar to the build-in type declaration.

Declaring Structure Variable

```
Student s1;
```

• Variable s1 is of type student.

• By using s1 we can access members of the struct.

For example: s1.name;

Assigning Values to Members

• S1.name="riya"

• S1.RollNumber=123

• S1.totalmks=500

• Members of the struct are accessed using dot operator along with struct variable.

• By default members of struct are private so cannot be accessed outside . (using dot operator)

• Member not declared public cannot be accessed using dot operator

Copying Struct:

Student s2;

s2=s1;

All the values will be copied to s2.

We can also instantiate a struct using the new keyword.

For example,

```
Structure  s1 = new Structure ();
```

Here, this line calls the parameterless constructor of the struct and initializes all the members with default values.

Example:

```
using System;
struct Item
{
      public string name;
      public int code;
      public double price;
}
class StructTest
{
      public static void Main( )
      {
            Item fan;     //create an item
            //Assign values to members
            fan.name  = "Bajaj";
            fan.code  =  123;
            fan.price =  1576.50;
            //Display item details
            Console.WriteLine("Fan name: " + fan.name);
            Console.WriteLine("Fan code: " + fan.code);
            Console.WriteLine("Fan cost: " + fan.price);
      }
}
```

Output

```
Fan name     :  Bajaj
Fan code     :  123
Fan cost     :  1576.5
```

Struct with methods

• We can assign value to the members using constructors

• A constructor is the method that is used to set the values of data members at the time of declaration.

Example:

```
struct Number
  {
      int num;                      //data member
     public Number(int val)    //constructor
      {
        num=val;
       }
 }
```

• The constructor method name is same as structure name and declared as public. The constructor is invoked as follows:

Number n1=new Number(100);

• This statement creates a structure object n1 and assigns the value 100 to its data member num.

• C# does not support default (parameterless) constructors.

• Structure is not permitted to declare destructor.

Example for Structure with method

```
using System;
struct Rectangle
{
    int a, b;
    public Rectangle ( int x, int y )   //constructor
    {
        a =  x;
        b = y;
    }
    public int Area( )                  //a method
    {
        return ( a * b);
    }
    public void Display ( )             //another method
    {
        Console.WriteLine("Area = " + Area( ) );
    }
}
class TestRectangle
{
    public static void Main( )
    {
        Rectangle rect = new Rectangle ( 10, 20 );
        rect.Display ( );   // invoking Display ( ) method
    }
}
```

Output

```
Area = 200
```

Nested Structure

C# permits declarations declaration of structs nested inside other structs.

```
struct student
   {
      public string name;
      public int rollno;
    }
 struct marks

    {
      public int mk1;
      public int mk2;


public student S;

    }
  class Program
   {
      static void Main(string[] args)
      {
         marks M;
         S.name = "Raju";
         S.rollno = 2345;
         S.mk1 = 65;
         S.mk2 = 78;

     Console.WriteLine("Name of the Student : " + S.name);
    Console.WriteLine("Register Number of the Student : " + S.rollno);

    Console.WriteLine("OOPS in C# : " + M.S.mk1);
    Console.WriteLine("Computer Network : " + M.S.mk2);

    Console.ReadKey();

      }
   }
}
```

Output

```
Name of the Student : Raju
Register Number of the Student : 2345
OOPS in C# : 65
Computer Network : 78
```

Difference between Classes and Structs

Structure	Class
• Structure is a user-defined data type that combines logically related data items of different data types like char, float, int, etc., together.	• Class is a blueprint or a set of instructions to build a specific type of object.
• Structure can be declared using the struct keyword.	• It can be declared using the class keyword.
• It is a value type data type.	• It is a reference type data type.
• You cannot inherit it from other structures or classes.	• You can inherit it from other structures or classes.
• It can instantiate objects with or without using a new keyword.	• It can instantiate an object using a new keyword.
• You cannot change the default constructor of structure.	• You can change class default constructor.
• Structure cannot have a destructor.	• Class can have a destructor.
• In structure, all the value types are allocated on stack.	• In class, all value types are allocated on heap.
• It is used in small programs.	• It is used in large programs.
• Member functions cannot be abstract or virtual.	• Member functions can be abstract or virtual.
• Structure instances are called 'structure variables.'	• Class instances are called objects.
• It cannot have null values.	• It can have null values.
• Structure member variables cannot be initiated directly.	• Class member variables can be initiated directly.
• If you have not declared any access specifier, then the members of the structure are public.	• If you have not declared any access specifier, then the members of class are private.

Enumeration

- **Enumeration (or enum)** is a value data type in C#.
- It is mainly used to assign the names or string values to integral constants, that make a program easy to read and maintain.
- The **enum** automatically enumerates a list of words by assigning them values 0,1,2 and so on.
- An explicit cast is required to convert from enum type to an integral type.

```
using System;

namespace EnumApplication
{
  class EnumProgram
   {
    enum Days
     {
           Sun,
           Mon,
           Tue,
           Wed,
           Thu,
           Fri,
          Sat
       };

    static void Main(string[] args)
    {
      int WeekdayStart = (int)Days.Mon;
      int WeekdayEnd = (int)Days.Fri;

      Console.WriteLine("Monday: {0}", WeekdayStart);
      Console.WriteLine("Friday: {0}", WeekdayEnd);
      Console.ReadKey();
    }
  }
}
```

Output

```
Monday: 1
Friday: 5
```