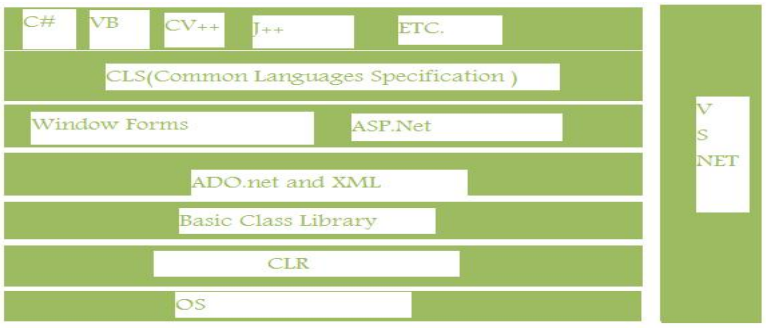
Q.2 a. What is C#? Discuss its features in brief.

Answer:

1. C# is a simple, modern, object oriented language derived from C++ and Java.
2. It aims to combine the high productivity of Visual Basic and the raw power of C++.
3. It is a part of Microsoft Visual Studio7.0.
4. Visual studio supports Vb, VC++,C++,Vbscript, Jscript. All of these languages provide access to the Microsft .NET platform.
5. .NET includes a Common Execution engine and a rich class library.
6. Microsofts JVM eqiv is Common language run time (CLR).
7. CLR accommadates more than one languages such as C#, VB.NET, Jscript, ASP.NET,C ++.
8. Source code --->Intermediate Language code(IL) ---> (JIT Compiler) Native code.
9. The classes and data types are common to all of the .NET languages.
10. We may develop Console application, Windows application, Web application using C#.
11. In C# microsoft has taken care of C++ problems such as Memory management, pointers etc.
12. It support garbage collection, automatic memory management and a lot.

Q.2b. Draw the .NET framework architecture & explain in detail.

Answer:



.Net is not an Operating System. It is a IDE. It provides some functionality for the programmers to build their solution in the most constructive and intelligent way ever. Just to tell you the truth, most of the codes in .Net environment resembles with the JAVA coding as if some people coming from Java would find it as to their native language.

.NET is a Framework that loads into your operating system, which you need not to load in the later versions of windows like Windows 2003 server or Just a new release of Windows Vista. As it is going to be added as a component in the next generation of windows.

Now, What is the .Net all about? Well, Its a framework that supports Common Language Runtime (CLR). As the name suggests, Common Language Runtime will be something that will run a native code for all the programming languages provided within the Architecture.

Q.3a. Write a program that employs nested *if...else* statements to determine that largest of three given numbers.

Answer: Page Number 101 of Text Book

Q.3b. What are increment and decrement operators in C#? Illustrate increment operator with a program.

Answer: Page Number 68-69 from Text Book

Q.4a. **What are methods? Write down the basic characteristics of methods?**

Answer:

A method is a code block containing a series of statements. Methods must be declared within a class or a structure. It is a good programming practice that methods do only one specific task. Methods bring modularity to programs. Proper use of methods bring the following advantages:

- Reducing duplication of code
- Decomposing complex problems into simpler pieces
- Improving clarity of the code
- Reuse of code
- Information hiding

Basic characteristics of methods are:

- Access level
- Return value type
- Method name
- Method parameters
- Parentheses
- Block of statements

Access level of methods is controlled with access modifiers. They set the visibility of methods. They determine who can call the method. Methods may return a value to the caller. In case our method returns a value, we provide its data type. If not, we use the void keyword to indicate, that our method does not return values. Method parameters are surrounded by parentheses, and separated by commas. Empty parentheses indicate that the method requires no parameters. The method block is surrounded with { } characters. The block contains one or more statements that are executed, when the method is invoked. It is legal to have an empty method block.

A **method signature** is a unique identification of a method for the C# compiler. The signature consists of a method name, and the type and kind (value, reference, or output) of each of its formal parameters. Method signature does not include the return type.

Q.4b. How to use Hash Table, Array List in c#? Explain with example.

Answer:

ArrayList:-Arraylist is a collection of objects(may be of different types).
Arraylist is very much similar to array but it can take values of different datatypes.
If you want to find something in a arraylist you have to go through each value in arraylist, theres no faster way out.ArrayList's size can be changed dynamically.

Items are added to the ArrayList with the Add() method.
example:-HOW TO ADD DATA IN ARRAYLIST:

```
using System.Collections;

class Program
{
   static void Main()
   {
      //
      // Create an ArrayList and add three elements.
      //
      ArrayList list = new ArrayList();
      list.Add("One");
      list.Add("Two");
      list.Add("Three");
   }
```

```
}
```

EXAMPLE-Adding one ArrayList to second one

There are different ways to add one ArrayList to another, but the best way is using AddRange. Internally, AddRange uses the Array.Copy or CopyTo methods, which have better performance than some loops.

=== Program that uses Add and AddRange ===

```
using System;
using System.Collections;

class Program
{
   static void Main()
   {
      //
      // Create an ArrayList with two values.
      //
      ArrayList list = new ArrayList();
      list.Add(15);
      list.Add(17);
      //
      // Second ArrayList.
      //
      ArrayList list2 = new ArrayList();
      list2.Add(110);
      list2.Add(113);
      //
      // Add second ArrayList to first.
      //
      list.AddRange(list2);
      //
      // Display the values.
      //
      foreach (int i in list)
      {
         Console.WriteLine(i);
      }
   }
}
```

=== Output of the program ===

```
15
17
```

110
113
The ArrayList class provides the Count property, which is a virtual property. When you use Count, no counting is actually done; instead, a cached field value is returned

HASH TABLE:-Hashtable is also collection which takes a key corresponding to each values.
If you want to find something in a hashtable you dont have to go through each value in hashtable, instead search for key values and is faster.The Hashtable lets you quickly get an object out of the collection by using it's key.
The Hashtable object contains items in key/value pairs. The keys are used as indexes. We can search value by using their corresponding key.
The data type of Hashtable is object and the default size of a Hashtable is 16.

Items are added to the Hashtable with the ADD()method.
Example-How to add data in hash table:--
Add method of Hashtable is used to add items to the hashtable. The method has index and value parameters. The following code adds three items to hashtable.

```
hshTable .Add("A1",  "Kamal");
hshTable .Add("A2",  "Aditya");
hshTable .Add("A3",  "Ashish");
```

Retrieving an Item Value from Hashtable

The following code returns the value of "Author1" key:

```
string name = hshTable["A1"].ToString();
```

Removing Items from a Hashtable

The Remove method removes an item from a Hashtable. The following code removes item with index "A1" from the hashtable:

```
hshTable.Remove("A1");
```

Looking through all Items of a Hashtable

The following code loops through all items of a hashtable and reads the values.

```
// Loop through all items of a Hashtable
IDictionaryEnumerator en = hshTable.GetEnumerator();
while (en.MoveNext())
{
```

```
        string str = en.Value.ToString();
    }
```

Q.5a. Write a program to reverse a string in C#.Net.

Answer:

```
using System;
    namespace Learn
{
        class Program
    {
            static void Main(string[] args)
        {
                string Str, Revstr = "";
                int Length;

                Console.Write("Enter A String : ");
               Str = Console.ReadLine();
                Length = Str.Length - 1;

                while (Length >= 0)
            {

                    Revstr = Revstr + Str[Length];
                    Length--;

            }

                Console.WriteLine("Reverse String Is {0}", Revstr);
                Console.ReadLine();

             }
        }
```

Q.5b. What is enumeration expression in C#? Illustrate through example.

Answer:

The elements of enumeration expressions evaluate the same as their underlying types. In addition to using normal operators, there are additional methods that can be performed with an `enum` type. An `Enum` class is used to obtain the majority of functionality shown in this section. Where the `Enum` class is being used, the capitalized `Enum` class name prefixes the method call.

The examples in this section refer to the following enum:

```
enum Weekday { Mon = 1, Tue, Wed, Thu, Fri, Sat = 10, Sun };
```

As a typed value, the enum must be assigned to a variable of its type. For example, the underlying representation of a Weekday enum may default to an integral value, but it's still a Weekday type. The following line shows the declaration and initialization of an enum variable:

```
Weekday w = Weekday.Mon;
```

During a Console.WriteLine() method call, enum values are printed with their names rather than their underlying integral values. Here's an example:

```
Console.WriteLine("WeekDay: {0}", w); // WeekDay: Mon
```

The Format() method returns the string representation of an enum value, as shown here:

```
Console.WriteLine("Format: {0}", w.Format()); // Format: Mon
```

To go in the opposite direction and convert a string to an enum, use the FromString() method. The arguments it accepts are the enum type, the string representation of the value to be converted, and a Boolean condition to verify case. The following example uses the typeof() operator to get the enum type. The string to be converted is Tue, and the method is case-sensitive.

```
Console.WriteLine("FromString: {0}",
  Enum.FromString(typeof(EnumTest.Weekday),
  "Tue", true));            // FromString: Tue
```

To get the name of an enum variable, use the GetName() method. The following example shows the GetName() method accepting the enum type and an instance of that enum type and returning its name as a string.

```
w = EnumTest.Weekday.Wed;
Console.WriteLine("GetName: {0}",
  Enum.GetName(typeof(EnumTest.Weekday), w)); // GetName: Wed
```

If there is a need to get the string representations of all the members of an enum, use the GetNames() method—plural of the previous method. The following example shows an array being filled with the names. The method call only needs the enum type.

```
string[] weekDays = new string[7];
weekDays = Enum.GetNames(typeof(EnumTest.Weekday));

Console.WriteLine("Day 1: {0}", weekDays[0]); // Day 1: Mon
Console.WriteLine("Day 2: {0}", weekDays[1]); // Day 2: Tue
Console.WriteLine("Day 3: {0}", weekDays[2]); // Day 3: Wed
```

```
Console.WriteLine("Day 4: {0}", weekDays[3]); // Day 4: Thu
Console.WriteLine("Day 5: {0}", weekDays[4]); // Day 5: Fri
Console.WriteLine("Day 6: {0}", weekDays[5]); // Day 6: Sat
Console.WriteLine("Day 7: {0}", weekDays[6]); // Day 7: Sun
```

A corresponding method to get the values of an enum is the GetValues() method. The following example shows the GetValues() method accepting an enum type and returning an array of objects. Notice that the array is of type objects. In C#, all types are also object types. Therefore, any type can be assigned to the object type.

```
object[] weekDayVals = new object[7];
weekDayVals = Enum.GetValues(typeof(EnumTest.Weekday));

Console.WriteLine("Day 1: {0}", weekDayVals[0]); // Day 1: Mon
Console.WriteLine("Day 2: {0}", weekDayVals[1]); // Day 2: Tue
Console.WriteLine("Day 3: {0}", weekDayVals[2]); // Day 3: Wed
Console.WriteLine("Day 4: {0}", weekDayVals[3]); // Day 4: Thu
Console.WriteLine("Day 5: {0}", weekDayVals[4]); // Day 5: Fri
Console.WriteLine("Day 6: {0}", weekDayVals[5]); // Day 6: Sat
Console.WriteLine("Day 7: {0}", weekDayVals[6]); // Day 7: Sun
```

To find out the underlying type of an enum, use the GetUnderlyingType() method. It accepts an enum type argument, and the return value is the integral type of the enum's underlying type. Here's an example:

```
Console.WriteLine("Underlying Type: {0}",
  Enum.GetUnderlyingType(
    typeof(EnumTest.Weekday))); // Underlying Type: Int32
```

When it's necessary to determine if an enum value is defined, use the IsDefined() method. It accepts an enum type and an enum value and returns a Boolean true if the value is defined in the enum. Otherwise, it returns false. Here's an example:

```
w = EnumTest.Weekday.Thu;
Console.WriteLine("Thu is Defined: {0}",
  Enum.IsDefined(typeof(EnumTest.Weekday), w));
    // Thu is Defined: True
```

To obtain an enum type that is set to a specific value, use the ToObject() method. The following example shows the method accepting an enum type and an integer, and returning an enum of the requested type with the value corresponding to the integer.

```
Console.WriteLine("Get Friday: {0}",
  Enum.ToObject(typeof(EnumTest.Weekday), 5));
  // Get Friday: Fri
```

Q.6a. Explain the three basic principles of OOPs.

Answer: Page Number 247 of the Text book

Q.6b. What is the difference between class & subclass?

Answer: Page Number 286 of the Text book

Q.6c. Write a program to illustrate a simple inheritance.

Answer: Page No 286 of the Text book

Q.8a. What is the need of operator overloading? Give examples of overloading unary operators, overloading binary operators, overloading comparison operators.

Answer:

Operator overloading is a concept of polymorphism where you can redefine operators like +, – , * etc with additional functionalities.

For instance we can redefine the + functionalities to add objects like obj1 + obj2. Below is simple code snippet which redefines + operator.

```
class SomeClass

{

private int someValue;

public SomeClass(int val)

{

someValue = val;

}

public static SomeClass operator +(SomeClass arg1, SomeClass arg2)
```

```
{

return new SomeClass(arg1.someValue + arg2.someValue);

}

}
```

You can now use the + operator to add objects of type someclass as shown in the below code snippet.

```
Obj = obj1 = obj2
```

Q.9b. What is thread? What does it do? What is the use of threading in C#?

Answer: Page Number 439 of text book

Text Book

Programming in C# -A primer, E Balagurusamy, II Edition, TMH, 2008