Practical-1

Nandinee bhatt

Contents

Practical-1

AIM:

Introduction to c#:

-Variables:
    Initialization
    Scope
    Constant
-Predefined Data Types
    Value Types
    Reference TYpes
-Flow Control
    Conditional Statements(if, switch)
    Loop(for, while, dowhile, foreach)
    Jump(goto, break, continue, return)
-Eumerations
-Passing Arguments

```
usingSystem;
namespaceP1
{
    classMyFirstClass
    {
        publicstaticvoidMain()
        {
                Console.WriteLine("HiAll");
                Console.ReadKey();
                return;
                }
        }
}
```

2.constantvariable

```
usingSystem;
namespaceCant
{
    publicclassCant
    {
        publicstaticvoidMain()
        {
                inta;
                a=99;
                Console.WriteLine("Valueis:{0}",a);

                Console.ReadKey();
        }
    }
}
```

3.scopeofvariable

```
usingSystem;
namespaceP1
{
    classScope1
    {
        publicstaticvoidMain()
        {
                for(inti=0;i<5;i++)
                {
                        Console.WriteLine(i);
                }

                //igoesoutofScopehere

                for(inti=4;i>=0;i--)
                {
                        Console.WriteLine(i);
                }
        }
    }
}
```

4.scopeofvariable

```
usingSystem;
namespaceP1
{
    classScope2
    {
        publicstaticvoidMain()
        {
                intj;
                for(inti=0;i<15;i++)
                {
                        intj;
                        Console.WriteLine(i);
                }
        }
    }
}
```

5.

```
usingSystem;
namespaceP1
{
    publicclassScope{
    staticintj=430;
    publicstaticvoidMain()
    {
        intj=900;
        Console.WriteLine(Scope.j);
    }
}
```

6.consatntvariable

```
usingSystem;
namespaceP1
{
    publicclassConst
    {
        publicstaticvoidMain()
        {
                constdoublebonusPercent=0.51;
                intsal=3000;
                intbonus=(int)(sal*bonusPercent);
                Console.WriteLine(bonus);
        }
    }
}
```

7.

```
usingSystem;
namespaceP1
```

```
{
    publicclassVector
    {
        publicintvalue;
    }
    publicclassDataTypes
    {
        publicstaticvoidMain()
        {
                inti;
                intj;


                i=77;
                j=i;

                Console.WriteLine("iis{0}andjis{1}",i,j);
                j=20;
                Console.WriteLine("iis{0}andjis{1}",i,j);

                Vectorx,y;
                x=newVector();
                x.value=33;
                y=x;
                Console.WriteLine("xis{0}andyis{1}",x.value,y.value);
                y.value=24;
                Console.WriteLine("xis{0}andyis{1}",x.value,y.value);

        }
    }
}
```

8.integersignedorunsignedvariables

```
usingSystem;
namespaceP1
{
      classIntType
      {
           publicstaticvoidMain()
           {
                    //SignedVariables
                    sbytesb=33;
                    shorts=33;
                    inti=33;
                    longl=33L;

                    //UnsignedVariables
                    byteb=33;
                    ushortus=33;
                    uintui=33U;
                    ulongul=33UL;
                    us=(ushort)ul;

                    Console.WriteLine("{0}{1}{2}{3}{4}{5}{6}{7}",
sb,s,i,l,b,us,ui,ul);

           }
      }
}
```

9.floatingvariables

```
usingSystem;
namespaceP1
```

```
{
      publicclassFloatting
      {
           publicstaticvoidMain()
           {
                    floatf=0.123456789F;
                    doubled=0.112233445566778899;
                    decimaldec=11223344.111222333444555666777888999M;
                    f=(float)d;
                    Console.WriteLine("fis{0}anddis{1}anddecis{2}",f,d,dec);
           }
      }
}
```

10.boolean

```
usingSystem;
namespaceP1
{
      publicclassBoolean
      {
           publicstaticvoidMain()
           {
                    boolstatus=true;
                    Console.WriteLine(status);
           }
      }
}
```

11.charcter

```
usingSystem;

namespaceP1
{
      publicclassChar
      {
           publicstaticvoidMain()
           {
                    charc='a';
                    Console.WriteLine(\a);
           }
      }
}
```

Practical-2

AIM:

GTU Programs

1)WriteconsolebasedprogramincodebehindlanguageVBorC#toprintfollowingpattern.

```
@@@@@
 @@@@
@@@
@@
@
```

```
usingSystem;
namespacePattern
{
        classPatternExample
        {
        publicstaticvoidMain()
        {
                inti,j=5;
                for(;j>0;j--)
                {
                for(i=j;i>0;i--)
                        Console.Write("@");
                Console.WriteLine();
                }
        }
        }
}
```

2)WriteconsolebasedprogramincodebehindlanguageVBorC#toprintfollowingpattern.

```
1
12
123
1234
```

```
usingSystem;
namespacePattern
{
        classpatternExample
        {
        publicstaticvoidMain()
        {
                inti,j;
                for(j=1;j<5;j++)
                {
                for(i=1;i<=j;i++)
                        Console.Write(i+"");
                Console.WriteLine();
                }
        }
        }
}
```

3.WriteC#codetopromptausertoinputhis/hernameandcountrynameandthentheoutputwillbeshownas an examplebelow:
HelloRamfromcountryIndia

```
usingSystem;
publicclassuserdata
{
	publicstaticvoidMain()
	{
	stringname,country;
	Console.Write("EnterYourName:");
	name=Console.ReadLine();
	Console.Write("EnterYourCountry:");
	country=Console.ReadLine();
	Console.WriteLine("Hello"+name+"fromcountry"+country);
	}
}
```

4.Whatisinheritance?CreateC#consoleapplicationtodefineCarclassandderiveMarutiandMahindraf romit todemonstrateinheritance.

```
usingSystem;

publicclassCar
    {
        protectedstringname;
        publicCar(stringname)
        {
            this.name=name;
        }
    publicCar()
        {
        }
        publicvirtualstringName
        {
            get{returnname;}
            set
            {
                if(value.Length>3)
                    name=value;
                 else
                    name="Unknown";
            }
        }
    }

publicclassMaruti:Car
{
    publicMaruti(stringname):base(name)
    {
    }
    publicoverridestringName
        {
            get{returnname;}
            set
            {
                if(value.Length>3)
                    name=value+"-Maruti";
                 else
                    name="Unknown";
            }
        }
    public boolhaveAGS;
}

publicclassMahindra:Car
```

```
publicMahindra(stringname):base(name)
        {
        }
        publicMahindra(){}
        publicoverridestringName
             {
                      get{returnname;}
                      set
                      {
                               if(value.Length>3)
                                        name=value+"-Mahindra";
                                else
                                        name="Unknown";
                      }
             }
  }
  publicclassProgram
  {
        publicstaticvoidMain()
        {
             Maruticar1=newMaruti("Swift");
             car1.haveAGS=true; car1.Name="Swift";
             Console.WriteLine("DetailsCar1:{0}and
  {1}",car1.Name,car1.haveAGS==true?"HaveAGS":"notHaveAGS");
             Mahindracar2=newMahindra();
             car2.Name="XUV500";
             Console.WriteLine("Car2:{0}",car2.Name);
        }
  }
```

PRACTICAL-3

AIM:

Method & constructor overloading

Program 1:

Write a c# program to add two integers, two vectors and two metric using method overloading.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
namespace p3
{
   public class Add
    {
      public void add()
       {
          int[,] m1 = new int[50, 50];
          int[,] m2 = new int[50, 50];
          int[,] m3 = new int[50, 50];
          Console.WriteLine("enter size of array:");
          int size = Convert.ToInt32(Console.ReadLine());
          Console.WriteLine("enter first array:");
          for (int i = 0; i < size; i++)
          {
             for (int j = 0; j < size; j++)
             {
                m1[i, j] = Convert.ToInt32(Console.ReadLine())
          }
          }
          Console.WriteLine("enter second array:");
          for (int i = 0; i < size; i++)
          {
             for (int j = 0; j < size; j++)
             {
                m2[i, j] = Convert.ToInt32(Console.ReadLine());
             }
          }

          for (int i = 0; i < size; i++)
          {
             for (int j = 0; j < size; j++)
             {
                m3[i, j] = m1[i, j] + m2[i, j];
             }
          }
          Console.WriteLine("addition array:");
          for (int i = 0; i < size; i++)
          {
             Console.Write("\n");
             for (int j = 0; j < size; j++)
             {
                Console.Write("{0}\t", m3[i, j]);
             }
```

```
            Console.Write("\n");
        }
    }
    public int add(int a, int b)
    {
        return (a + b);
    }
  }
    public class Vector
    {
        public void add()
        {
            Console.WriteLine("enter first vector");
            int x = Convert.ToInt32(Console.ReadLine());
            int y = Convert.ToInt32(Console.ReadLine());
            int z = Convert.ToInt32(Console.ReadLine());
            Console.WriteLine("enter second vector");
            int x1 = Convert.ToInt32(Console.ReadLine());
            int y1 = Convert.ToInt32(Console.ReadLine());
            int z1 = Convert.ToInt32(Console.ReadLine());
            int x2 = x + x1;
            int y2 = y + y1;
            int z2 = z + z1;
            Console.WriteLine("<" + x2 + "," + y2 + "," + z2 + ">");

        }
    }
 class Program
  {
    static void Main(string[] args)
    {

        Add a1 = new Add();
        Vector v1 = new Vector();
        v1.add();
        a1.add();
        int res=a1.add(1, 2);
        Console.Write("method overloading for addtion{0}",res);
        Console.ReadLine();

    }
  }
}
```

Program 2

AIM:

Write a c# program that create student object. Overload constror to create new instant with following details.
1. Name
2. Name, Enrollment
3. Name, Enrollment, Branch

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Reflection;

namespace p3a1
{
    class Program
    {
        public int ID { get; set; }
        public string Name { get; set; }
        String name, branch;
        int enrol;
        public Program(String name)
        {
            this.name = name;
            Console.WriteLine("constructor 1:" + name);
        }
        public Program(String name, int enrol)
        {
            this.name = name;
            this.enrol = enrol;
            Console.WriteLine("constructor 2:" + name + " " + enrol);

        }
        public Program(String name, int enrol, String branch)
        {
            this.name = name;
            this.enrol = enrol;
            this.branch = branch;
            Console.WriteLine("constructor 3:" + name + " " + enrol + " " + branch);

        }
        static void Main(string[] args)
        {
 Program p1 = new Program("bob");
            Program p2 = new Program("bob", 1);
            Program p3 = new Program("bob", 1, "computer");
                        Console.ReadLine();

        }
    }
}
```

PRACTICAL-4

AIM: Reflection

Program 1:

Create a c# program to find Methods, Properties and Constructors from class of running program.

```
UsingSystem;
                    usingSystem.Reflection;
                    namespaceReflectionExample
                    {
                    classMainClass
                       {
                    staticvoidMain()
                          {
                    Type T = Type.GetType("ReflectionExample.Customer");
                    MethodInfo[] methods = T.GetMethods();
                    foreach (MethodInfo method in methods)
                            {
                               Console.WriteLine(method.ReturnType +" "+ method.Name);
                            }

                    PropertyInfo[] properties =  T.GetProperties();

                            Console.WriteLine("\nProperties");
                    foreach (PropertyInfo property in properties)
                            {
                               Console.WriteLine(property.PropertyType+" "+ property.Name);
                            }

                            Console.WriteLine("\nConstructors");
                    ConstructorInfo[] constructors = T.GetConstructors();
                    foreach (ConstructorInfo constructor in constructors)
                            {
                               Console.WriteLine(constructor.ToString());
                            }
                          }
                       }
                    classCustomer
                       {
                    publicint ID { get; set; }
publicstring Name { get; set; }
publicCustomer(int ID, string Name)
        {
this.ID = ID;
this.Name = Name;
        }
publicCustomer()
        {
this.ID =-1;
this.Name = string.Empty;
        }
publicvoidprintID()
        {
            Console.WriteLine("ID is: {0}", this.ID);
        }
publicvoidprintName()
        {
            Console.WriteLine("Name is: {0}", this.Name);
        }
    }
}
```