Arithmetic operations in C# using switch cases and functions

```
// Arithmetic Operators in C#
using System;

public class MyClass{
   // It's the driver function
   static public void Main (string[] args) {
      Console.WriteLine("Enter any two positive integer numbers:\n");
      int p = Convert.ToInt32(Console.ReadLine());
      int q = Convert.ToInt32(Console.ReadLine());

      int sum, sub, mul, mod;
      float div;

      // It will perform all arithmetic operations
      sum = p + q;
      sub = p - q;
      mul = p * q;
      div = (float)p / q;
      mod = p % q;
            // It will print the final output
      Console.WriteLine("Addition of       "+p+" + "+q+" = "+sum);
      Console.WriteLine("Subtraction of    "+p+" - "+q+" = "+sub);
      Console.WriteLine("Multiplication of "+p+" * "+q+" = "+mul);
      Console.WriteLine("Division of       "+p+" / "+q+" = "+div);
      Console.WriteLine("Modulus of        "+p+" % "+q+" = "+mod);
   }
}
```

// Arithmetic Operators in C# using Switch Case

```
using System;

public class MyClass {
   // It's the driver function
   static public void Main (string[] args) {
      Console.WriteLine("Enter any two positive integer numbers:\n");
      int p = Convert.ToInt32(Console.ReadLine());
      int q = Convert.ToInt32(Console.ReadLine());

      // It will suggest choosing an option to make the operation
      Console.WriteLine("Input your choice to make an operation\n");
      Console.WriteLine("1 :: for Addition");
      Console.WriteLine("2 :: for Subtraction");
      Console.WriteLine("3 :: for Multiplication");
      Console.WriteLine("4 :: for Division");
      Console.WriteLine("5 :: for Modulus");

      Console.WriteLine("\nEnter your choice:\n");
      int choice = Convert.ToInt32(Console.ReadLine());

      // It will perform all arithmetic operations
      // According to user's choice & print the final output
      switch (choice) {
         case 1:
            Console.WriteLine("Addition of "+p+" + "+q+" = "+(p + q));
            break;
               case 2:
            Console.WriteLine("Subtraction of "+p+" - "+q+" = "+(p - q));
            break;
                     case 3:
            Console.WriteLine("Multiplication of "+p+" * "+q+" = "+(p * q));
            break;
                     case 4:
            Console.WriteLine("Division of "+p+" / "+q+" = "+(float)(p + q));
            break;
                     case 5:
            Console.WriteLine("Modulus of "+p+" % "+q+" = "+(p % q));
            break;
                     default:
            Console.WriteLine("Kindly input correct choice!");
            break;
      }
```

```
  }
}
```

// Arithmetic Operators in C# using Functions

```
using System;

public class MyClass {
  // Function to make an Addition
  public static int Add(int a, int b) {
    return a + b;
  }

  // Function to make Subtraction
  public static int Sub(int a, int b) {
    return a - b;
  }

  // Function to make Multiplication
  public static int Mul(int a, int b) {
    return a * b;
  }

  // Function to make Division
  public static float Div(int a, int b) {
    return (float)(a / b);
  }

  // Function to make Modulus
  public static int Mod(int a, int b) {
    return a % b;
  }

  // It's the driver function
  static public void Main (string[] args) {
    Console.WriteLine("Enter any two positive integer numbers:\n");
    int p = Convert.ToInt32(Console.ReadLine());
    int q = Convert.ToInt32(Console.ReadLine());
        // It will print the final output
    Console.WriteLine("Addition of       "+p+" + "+q+" = "+Add(p,q));
    Console.WriteLine("Subtraction of    "+p+" - "+q+" = "+Sub(p,q));
    Console.WriteLine("Multiplication of "+p+" * "+q+" = "+Mul(p,q));
    Console.WriteLine("Division of       "+p+" / "+q+" = "+Div(p,q));
    Console.WriteLine("Modulus of       "+p+" % "+q+" = "+Mod(p,q));
  }
}
```

Jagged Arrays

```
public class JaggedArrayTest
{
  public static void Main()
  {
    int[][] arr = new int[2][];// Declare the array
     arr[0] = new int[] { 11, 21, 56, 78 };// Initialize the array
    arr[1] = new int[] { 42, 61, 37, 41, 59, 63 };

    // Traverse array elements
    for (int i = 0; i < arr.Length; i++)
    {
      for (int j = 0; j < arr[i].Length; j++)
      {
        Console.Write(arr[i][j]+" ");
      }
      System.Console.WriteLine();
    }
  }
}
```

OR

```
public class JaggedArrayTest
{
  public static void Main()
  {
    int[][] arr = new int[3][]{
    new int[] { 11, 21, 56, 78 },
    new int[] { 2, 5, 6, 7, 98, 5 },
    new int[] { 2, 5 }
    };
     // Traverse array elements
    for (int i = 0; i < arr.Length; i++)
    {
        for (int j = 0; j < arr[i].Length; j++)
        {
            System.Console.Write(arr[i][j]+" ");
        }
        System.Console.WriteLine();
    }
  }
}
```

```
    //Operator Overloading
using System;
class Complex
{
    private int x;
    private int y;
    public Complex()
    {
    }
    public Complex(int i, int j)
    {
        x = i;
        y = j;
    }
    public void ShowXY()
    {
        Console.WriteLine("{0} {1}", x, y);
    }
    public static Complex operator +(Complex c1, Complex c2)
    {
        Complex temp = new Complex();
        temp.x = c1.x + c2.x;
        temp.y = c1.y + c2.y;
        return temp;
    }
}
class MyComplex: Complex
{
    private double x;
    private double y;
    public MyComplex(double i, double j)
    {
        x = i;
        y = j;
    }
    public MyComplex()
    {
    }
    public new void ShowXY()
    {
        Console.WriteLine("{0} {1}", x, y);
    }
}
class MyClient
{
    public static void Main()
    {
        MyComplex mc1 = new MyComplex(1.5, 2.5);
        mc1.ShowXY();
        MyComplex mc2 = new MyComplex(3.5, 4.5);
        mc2.ShowXY();
        MyComplex mc3 = new MyComplex();
        //mc3 = mc1 + mc2;
        //mc3.ShowXY();
    }
}
```

//Method Overloading

```
using System;
namespace MethodOverloading
{
    class Program
    {
        static void Main(string[] args)
        {
            Program obj = new Program();
            obj.Method(); //Invoke the 1st Method
            obj.Method(10); //Invoke the 2nd Method
            obj.Method("Hello"); //Invoke the 3rd Method
            obj.Method(10, "Hello"); //Invoke the 4th Method
            obj.Method("Hello", 10); //Invoke the 5th Method

            Console.ReadKey();
        }

        public void Method()
        {
            Console.WriteLine("1st Method");
        }
        public void Method(int i)
        {
            Console.WriteLine("2nd Method");
        }
        public void Method(string s)
        {
            Console.WriteLine("3rd Method");
        }
        public void Method(int i, string s)
        {
            Console.WriteLine("4th Method");
        }
        public void Method(string s, int i)
        {
            Console.WriteLine("5th Method");
        }
    }
}
```

// Demo on Classes and objects

```
using System;
namespace ClassObjectsDemo
{
    class Program
    {
        static void Main(string[] args)
        {
            //Creating object
            Calculator calObject = new Calculator();
            //Accessing Calculator class member using Calculator class object
            int result = calObject.CalculateSum(10, 20);

            Console.WriteLine(result);
            Console.ReadKey();
        }
    }

    //Defining class or blueprint or template
    public class Calculator
    {
        public int CalculateSum(int no1, int no2)
        {
            return no1 + no2;
        }
    }
}
```

//Properties (EmpId and EmpName)

```
using System;
namespace PropertyDemo
{
    public class Employee
    {
        //Private Data Members
        private int EmpId;
        private string EmpName;

        //Public Properties
        public int EmpId
        {
            //The Set Accessor is used to set the EmpId private variable value
            set
            {
                EmpId = value;
            }
            //The Get Accessor is used to return the EmpId private variable value
            get
            {
                return EmpId;
            }
        }
        public string EmpName
        {
            //The Set Accessor is used to set the EmpName private variable value
            set
            {
                EmpName = value;
            }
            //The Get Accessor is used to return the EmpName private variable value
            get
            {
                Return EmpName;
            }
        }
    }
    class Program
    {
        static void Main(string[] args)
        {
            Employee employee = new Employee();
            //We cannot access the private data members
            //So, using public properties (SET Accessor) we are setting
            //the values of private data members
            employee.EmpId = 1000;
            employee.EmpName = "Kalpana";

            //Using public properties (Get Accessor) we are Getting
            //the values of private data members
            Console.WriteLine("Employee Details:");
            Console.WriteLine("Employee id:" + employee.EmpId);
            Console.WriteLine("Employee name:" + employee.EmpName);
            Console.ReadKey();
        }
    }
}
```