# Write a Program to demonstrate the difference between Console.ReadLine() and Console.Read(), Console.Write() and Console.WriteLine() in C#.

**Code:**

using System;

namespace lab1

{

internal class Program

{

static void Main(string[] args)

{

Console.WriteLine("-----Console.ReadLine----");

Console.Write("Enter Your Name: ");

string name = Console.ReadLine(); // Reads entire line

Console.WriteLine($"Your name is {name}.");

Console.WriteLine("-----Console.Read--------");

Console.Write("Enter a character:");

char charac = (char)Console.Read(); // Reads only one character

Console.WriteLine($"You entered {charac}.");

Console.WriteLine("-----Console.Write-----");

Console.Write("This is ");

Console.Write("a single ");

Console.Write("line."); // Outputs without newline

Console.WriteLine();

Console.WriteLine("-----Console.WriteLine---");

Console.WriteLine("This is");

Console.WriteLine("multiple");

Console.WriteLine("lines."); // Outputs with newline each time

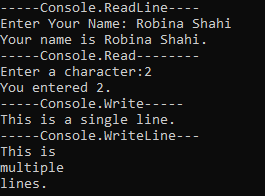
Console.ReadKey();

}

}

}

**Output:**



# 2.Write a program to add number x, y and z. One Function to add x and y another function to add all three Numbers. Print on Screen As given output.

Code:

using System;

namespace lab2

{

internal class Program

{

static void Main(string[] args)

{

Console.WriteLine($"The Sum of 5 and 6 = {Sum(5, 6)}");

Console.WriteLine();

Console.WriteLine($"The Sum of 5,6 and 7 = {Sum(5,6,7)}");

Console.ReadKey();

}

static int Sum(int x,int y)

{

return x + y;

}

static int Sum(int x,int y,int z)

{

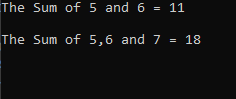
return x + y + z;

}

}

}

Output:



# 3.Write a Program to implement the concept of default constructor, parameterized Constructor.

Code:

Program.cs

using lab3.Classes;

using System;

namespace lab3

{

internal class Program

{

static void Main(string[] args)

{

Wish Wish = new Wish();

Console.WriteLine("-----Default Constructor----------");

Wish.Happy();

Console.WriteLine();

Console.WriteLine("-----Parameterized Constructor----");

Wish Wish2 = new Wish("Happy Anniversy","Honey");

Wish2.Happy();

Console.ReadKey();

}

}

}

Wish.cs

using System;

namespace lab3.Classes

{

public class Wish

{

public string wish;

public string name;

public Wish()

{

wish = "Happy Birthday";

name= "Robina";

}

public Wish(string wish,string Name)

{

this.wish = wish;

this.name = Name;

}

public void Happy()

{

Console.WriteLine($"{wish} {name}!");

}

}}

Output:



# 4. Write the concept of multilevel inheritance and multiple inheritance in C#.

Code:

Multilevel inheritance:

using System;

namespace lab4

{

public class Animal

{

public void Eat()

{

Console.WriteLine("Animal is Eating");

}

}

public class Dog:Animal

{

public void Bark()

{

Console.WriteLine("Dog is barking.");

}

}

public class Doberman:Dog

{

public void Type()

{

Console.WriteLine("Doberman is a security Type Dog.");

}

}

public class Program

{

static void Main(string[] args)

{

Doberman doberman = new Doberman();

doberman.Eat();

doberman.Bark();

doberman.Type();

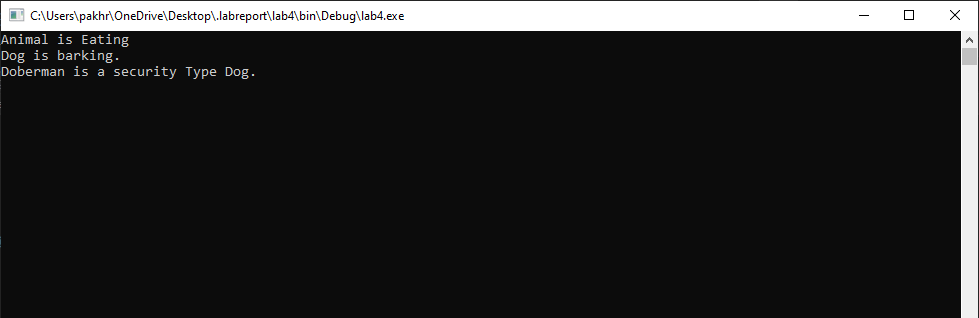
Console.ReadKey();

}

}

}

Output:



Multiple Inheritance:

using System;

namespace lab4.\_1

{

public interface Addition

{

double add(double a, double b);

}

public interface Substraction

{

double sub(double a, double b);

}

public interface Multiply

{

double mul(double a, double b);

}

public interface Division

{

double div(double a, double b);

}

public class Calculation : Addition, Substraction, Division, Multiply

{

public double add(double a, double b)

{

return a + b;

}

public double sub(double a, double b)

{

return a - b;

}

public double mul(double a, double b)

{

return a \* b;

}

public double div(double a, double b)

{

return a / b;

}

public class Program

{

static void Main(string[] args)

{

Calculation calc = new Calculation();

Console.WriteLine($"Addition:{calc.add(5, 6)}");

Console.WriteLine($"Substraction:{calc.sub(10,5)}");

Console.WriteLine($"Multiplication:{calc.mul(5,5)}");

Console.WriteLine($"Division:{calc.div(10,5)}");

Console.ReadKey();

}

}

}

}

Output:

