2.

using System;

class Program

{

    static void Main()

    {

        // Print "Hello, World!" to the console

        Console.WriteLine("Hello, World!");

    }

}

3. TEMP

using System;

class Program

{

    static void Main()

    {

        Console.Write("Enter temperature in Fahrenheit: ");

        double fahrenheit = double.Parse(Console.ReadLine());

        double celsius = (fahrenheit - 32) \* 5 / 9;

        Console.WriteLine($"Temperature in Celsius: {celsius:F2}");

    }

}

4. PRIME NO

using System;

 class Program

 {

    static bool IsPrime(int number)

    {

         if (number < 2)

            return false;

        for (int i = 2; i <= Math.Sqrt(number); i++)

       {

           if (number % i == 0)

                return false;

       }

        return true;

    }

    static void Main()

    {

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

        int number = int.Parse(Console.ReadLine());

        if (IsPrime(number))

            Console.WriteLine($"{number} is a prime number.");

        else

            Console.WriteLine($"{number} is not a prime number.");

    }

 }

5. Palindrome

 using System;

 class Program

 {

     static bool IsPalindrome(string input)

     {

         // Remove any spaces and convert to lowercase

         input = input.Replace(" ", "").ToLower();

         int left = 0;

         int right = input.Length - 1;

         while (left < right)

         {

             if (input[left] != input[right])

                 return false;

             left++;

             right--;

         }

         return true;

     }

     static void Main()

     {

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

         string input = Console.ReadLine();

         if (IsPalindrome(input))

             Console.WriteLine($"{input} is a palindrome.");

         else

             Console.WriteLine($"{input} is not a palindrome.");

     }

 }

6. VOWELS

using System;

 class Program

 {

     static void Main()

     {

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

         string input = Console.ReadLine();

         string vowels = FindVowels(input);

         Console.WriteLine($"Vowels in the string: {vowels}");

     }

     static string FindVowels(string input)

     {

         string vowels = "";

         input = input.ToLower();

         foreach (char c in input)

         {

             if (IsVowel(c))

             {

                 if (!vowels.Contains(c.ToString()))

                     vowels += c;

             }

         }

         return vowels;

     }

     static bool IsVowel(char c)

     {

         char[] vowels = { 'a', 'e', 'i', 'o', 'u' };

         foreach (char vowel in vowels)

         {

             if (c == vowel)

                 return true;

         }

         return false;

     }

 }

7. If else

using System;

class Program

{

    static void Main()

    {

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

        int number = int.Parse(Console.ReadLine());

        // if-else ladder

        if (number == 0)

        {

            Console.WriteLine("Number is zero.");

        }

        else if (number > 0)

        {

            Console.WriteLine("Number is positive.");

        }

        else

        {

            Console.WriteLine("Number is negative.");

        }

    }

}

8. FIBONACCI SERIES

using System;

namespace Fibonacci

{

class Program

 {

     static void Main()

     {

         Console.Write("Enter the number of Fibonacci numbers to generate: ");

         int n = int.Parse(Console.ReadLine());

       Console.WriteLine("Fibonacci Series:");

       for (int i = 0; i < n; i++)

       {

            int fibonacciNumber = Fibonacci(i);

            Console.Write(fibonacciNumber + " ");

       }

    }

   static int Fibonacci(int n)

   {

       if (n <= 1)

         return n;

       int prev = 0;

        int current = 1;

        int fibonacciNumber = 0;

       for (int i = 2; i <= n; i++)

       {

           fibonacciNumber = prev + current;

           prev = current;

           current = fibonacciNumber;

       }

       return fibonacciNumber;

   }

}

}

9. FACTORIAL

using System;

class Program

{

    static void Main()

    {

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

        int number = int.Parse(Console.ReadLine());

        long factorial = CalculateFactorial(number);

        Console.WriteLine($"Factorial of {number} is {factorial}");

    }

    static long CalculateFactorial(int number)

    {

        if (number < 0)

            throw new ArgumentException("Factorial is not defined for negative numbers.");

        if (number == 0 || number == 1)

            return 1;

        long factorial = 1;

        for (int i = 2; i <= number; i++)

        {

            factorial \*= i;

        }

        return factorial;

    }

}

10. MENU DRIVEN

using System;

class Program

{

    static void Main()

    {

        while (true)

        {

            Console.WriteLine("Menu:");

            Console.WriteLine("1. Convert binary to decimal");

            Console.WriteLine("2. Convert decimal to binary");

            Console.WriteLine("3. Convert hexadecimal to octal");

            Console.WriteLine("4. Convert octal to decimal");

            Console.WriteLine("5. Exit");

            Console.Write("Enter your choice: ");

            int choice = int.Parse(Console.ReadLine());

            switch (choice)

            {

                case 1:

                    ConvertBinaryToDecimal();

                    break;

                case 2:

                    ConvertDecimalToBinary();

                    break;

                case 3:

                    ConvertHexadecimalToOctal();

                    break;

                case 4:

                    ConvertOctalToDecimal();

                    break;

                case 5:

                    Environment.Exit(0);

                    break;

                default:

                    Console.WriteLine("Invalid choice. Please try again.");

                    break;

            }

            Console.WriteLine();

        }

    }

    static void ConvertBinaryToDecimal()

    {

        Console.Write("Enter a binary number: ");

        string binary = Console.ReadLine();

        int decimalNumber = Convert.ToInt32(binary, 2);

        Console.WriteLine($"Decimal equivalent: {decimalNumber}");

    }

    static void ConvertDecimalToBinary()

    {

        Console.Write("Enter a decimal number: ");

        int decimalNumber = int.Parse(Console.ReadLine());

        string binary = Convert.ToString(decimalNumber, 2);

        Console.WriteLine($"Binary equivalent: {binary}");

    }

    static void ConvertHexadecimalToOctal()

    {

        Console.Write("Enter a hexadecimal number: ");

        string hexadecimal = Console.ReadLine();

        string binary = Convert.ToString(Convert.ToInt32(hexadecimal, 16), 2);

        string octal = Convert.ToString(Convert.ToInt32(binary, 2), 8);

        Console.WriteLine($"Octal equivalent: {octal}");

    }

    static void ConvertOctalToDecimal()

    {

        Console.Write("Enter an octal number: ");

        string octal = Console.ReadLine();

        int decimalNumber = Convert.ToInt32(octal, 8);

        Console.WriteLine($"Decimal equivalent: {decimalNumber}");

    }

}

1. Multiply 2 values by parametized constructor
2. using System;
3. class Multiplier
4. {
5. private int value1;
6. private int value2;
7. // Parameterized constructor
8. public Multiplier(int v1, int v2)
9. {
10. value1 = v1;
11. value2 = v2;
12. }
13. public int Multiply()
14. {
15. return value1 \* value2;
16. }
17. }
18. class Program
19. {
20. static void Main()
21. {
22. Console.Write("Enter the first value: ");
23. int val1 = int.Parse(Console.ReadLine());
24. Console.Write("Enter the second value: ");
25. int val2 = int.Parse(Console.ReadLine());
26. // Create an instance of Multiplier with the given values
27. Multiplier multiplier = new Multiplier(val1, val2);
28. // Call the Multiply method and display the result
29. int result = multiplier.Multiply();
30. Console.WriteLine("Multiplication Result: " + result);
31. }
32. }

2. Operator overloading

using System;

public class ComplexNumber

{

    public double Real { get; set; }

    public double Imaginary { get; set; }

    public ComplexNumber(double real, double imaginary)

    {

        Real = real;

        Imaginary = imaginary;

    }

    public static ComplexNumber operator +(ComplexNumber num1, ComplexNumber num2)

    {

        double realPart = num1.Real + num2.Real;

        double imaginaryPart = num1.Imaginary + num2.Imaginary;

        return new ComplexNumber(realPart, imaginaryPart);

    }

    public override string ToString()

    {

        return Real + " + " + Imaginary + "i";       //operator overloaded

    }

}

public class Program

{

    public static void Main(string[] args)

    {

        ComplexNumber num1 = new ComplexNumber(2, 3);

        ComplexNumber num2 = new ComplexNumber(4, 5);

        ComplexNumber sum = num1 + num2;

        Console.WriteLine("Number 1: " + num1);

        Console.WriteLine("Number 2: " + num2);

        Console.WriteLine("Sum: " + sum);

    }

}

3. Function overloading

using System;

public class Calculator

{

    public int Add(int num1, int num2)

    {

        return num1 + num2;

    }

    public double Add(double num1, double num2)

    {

        return num1 + num2;

    }

}

public class Program

{

    public static void Main(string[] args)

    {

        Calculator calculator = new Calculator();

        int result1 = calculator.Add(5, 10);

        double result2 = calculator.Add(6.25, 3.75);

        Console.WriteLine("Result 1: " + result1);

        Console.WriteLine("Result 2: " + result2);

    }

}

4. Multiple inheritance

using System;

public interface IShape

{

    void Draw();

}

public interface IColor

{

    void FillColor();

}

public class Rectangle : IShape, IColor

{

    public void Draw()

    {

        Console.WriteLine("Drawing a rectangle.");

    }

    public void FillColor()

    {

        Console.WriteLine("Filling the rectangle with color.");

    }

}

public class Circle : IShape, IColor

{

    public void Draw()

    {

        Console.WriteLine("Drawing a circle.");

    }

    public void FillColor()

    {

        Console.WriteLine("Filling the circle with color.");

    }

}

public class Program

{

    public static void Main(string[] args)

    {

        Rectangle rectangle = new Rectangle();

        rectangle.Draw();

        rectangle.FillColor();

        Circle circle = new Circle();

        circle.Draw();

        circle.FillColor();

    }

}

5. Multilevel inheritance

using System;

public class Animal

{

    public void eat()

    {

        Console.WriteLine("Was eating");

    }

}

    public class Cow:Animal

    {

        public void moo()

        {

            Console.WriteLine("Was mooing");

        }

    }

        public class Calf: Cow

        {

            public void laugh()

            {

                Console.WriteLine("Was laughing");

            }

        }

            class TestInheritance

            {

                public static void Main(string[] args)

                {

                    Calf d1=new Calf();

                    d1.eat();

                    d1.moo();

                    d1.laugh();

                }

            }

6. Hybrid inheritance

using System;

public interface IShape

{

    void Draw();

}

public class Shape

{

    public void Display()

    {

        Console.WriteLine("Displaying the shape.");

    }

}

public class Rectangle : Shape, IShape

{

    public void Draw()

    {

        Console.WriteLine("Drawing a rectangle.");

    }

}

public class Circle : Shape, IShape

{

    public void Draw()

    {

        Console.WriteLine("Drawing a circle.");

    }

}

public class Program

{

    public static void Main(string[] args)

    {

        Rectangle rectangle = new Rectangle();

        rectangle.Display();

        rectangle.Draw();

        Circle circle = new Circle();

        circle.Display();

        circle.Draw();

    }

}

7. Exception handling

using System;

public class Program

{

    public static void Main(string[] args)

    {

        try

        {

            int numerator = 10;

            int denominator = 0;

            int result = numerator / denominator;

            Console.WriteLine("Result: " + result); // This line won't be executed

        }

        catch (DivideByZeroException ex)

        {

            Console.WriteLine("An error occurred: " + ex.Message);

        }

        finally

        {

            Console.WriteLine("Exception handling is complete.");

        }

    }

}

8. Multiple catch

using System;

public class Program

{

    public static void Main(string[] args)

    {

        try

        {

            int[] numbers = { 1, 2, 3 };

            int index = 4;

            int result = numbers[index];

            Console.WriteLine("Result: " + result); // This line won't be executed

        }

        catch (IndexOutOfRangeException ex)

        {

            Console.WriteLine("Index out of range exception: " + ex.Message);

        }

        catch (DivideByZeroException ex)

        {

            Console.WriteLine("Divide by zero exception: " + ex.Message);

        }

        catch (Exception ex)

        {

            Console.WriteLine("General exception: " + ex.Message);

        }

        finally

        {

            Console.WriteLine("Exception handling is complete.");

        }

    }

}

9. Nested catch

using System;

public class Program

{

    public static void Main(string[] args)

    {

        try

        {

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

            int number = Convert.ToInt32(Console.ReadLine());

            try

            {

                int result = 100 / number;

                Console.WriteLine("Result: " + result);

            }

            catch (DivideByZeroException ex)

            {

                Console.WriteLine("Divide by zero exception: " + ex.Message);

            }

            catch (Exception ex)

            {

                Console.WriteLine("Inner exception: " + ex.Message);

            }

        }

        catch (FormatException ex)

        {

            Console.WriteLine("Format exception: " + ex.Message);

        }

        catch (Exception ex)

        {

            Console.WriteLine("Outer exception: " + ex.Message);

        }

    }

}

10.Collections

using System;

using System.Collections;

public class Program

{

    public static void Main(string[] args)

    {

        // ArrayList

        ArrayList arrayList = new ArrayList();

        arrayList.Add("John");

        arrayList.Add("Alice");

        arrayList.Add("Bob");

        Console.WriteLine("ArrayList:");

        foreach (var item in arrayList)

        {

            Console.WriteLine(item);

        }

        Console.WriteLine();

        // List<T>

        List<string> list = new List<string>();

        list.Add("Mary");

        list.Add("Tom");

        list.Add("Jane");

        Console.WriteLine("List<T>:");

        foreach (var item in list)

        {

            Console.WriteLine(item);

        }

        Console.WriteLine();

        // Queue

        Queue queue = new Queue();

        queue.Enqueue("Apple");

        queue.Enqueue("Banana");

        queue.Enqueue("Orange");

        Console.WriteLine("Queue:");

        while (queue.Count > 0)

        {

            Console.WriteLine(queue.Dequeue());

        }

        Console.WriteLine();

        // Stack

        Stack stack = new Stack();

        stack.Push("Red");

        stack.Push("Green");

        stack.Push("Blue");

        Console.WriteLine("Stack:");

        while (stack.Count > 0)

        {

            Console.WriteLine(stack.Pop());

        }

        Console.WriteLine();

        // Dictionary<TKey, TValue>

        Dictionary<int, string> dictionary = new Dictionary<int, string>();

        dictionary.Add(1, "One");

        dictionary.Add(2, "Two");

        dictionary.Add(3, "Three");

        Console.WriteLine("Dictionary<TKey, TValue>:");

        foreach (var item in dictionary)

        {

            Console.WriteLine(item.Key + " - " + item.Value);

        }

        Console.WriteLine();

        // HashSet<T>

        HashSet<int> hashSet = new HashSet<int>();

        hashSet.Add(10);

        hashSet.Add(20);

        hashSet.Add(30);

        Console.WriteLine("HashSet<T>:");

        foreach (var item in hashSet)

        {

            Console.WriteLine(item);

        }

        Console.WriteLine();

        // LinkedList<T>

        LinkedList<string> linkedList = new LinkedList<string>();

        linkedList.AddLast("Car");

        linkedList.AddLast("Bus");

        linkedList.AddLast("Train");

        Console.WriteLine("LinkedList<T>:");

        foreach (var item in linkedList)

        {

            Console.WriteLine(item);

        }

    }

}