.NET CONSOLE APPLICATION IN C#

- In C#, Console.Write(), Console.WriteLine(), Console.Read(), and Console.ReadLine() are fundamental tools for interacting with the user through a console window (the black text-based window).

- They are part of the System namespace, which you typically include with using System.

1) Console.Write() and Console.WriteLine() (For Output):

- These methods are used to display text, numbers, or other information from your program onto the console screen. Think of it like your program "talking" to the user.

- **Console.Write()**: Puts the text on the screen and keeps the cursor on the **same line**. So, anything you print next will appear right after it.

- **Console.WriteLine():** Puts the text on the screen and then automatically moves the cursor to the **next line**. This is like pressing "Enter" after typing. It's used for displaying messages one after another, each on its own line.

2) Console.Read() and Console.ReadLine() (For Input):

These methods are used to get input from the user who is typing into the console. This is how your program "listens" to the user.

Console.Read():

- Reads only a **single character** that the user types.
- It returns the **numeric (ASCII/Unicode) value** of that character.
- It waits for the user to press any key.
- **Console.ReadLine():**
- Reads an **entire line of text** that the user types, until they press the Enter key.
- It returns the input as a **string** (a sequence of characters).

.NET CONSOLE APPLICATION IN C#

- This is the most common method for getting text-based input from a user.

1) Create a .NET Console Application that will Add two numbers.

```
static void Main(string[] args)
{
        Console.Write("Enter the first number: ");
        double num1 = double.Parse(Console.ReadLine());

          Console.Write("Enter the second number: ");
           double num2 = double.Parse(Console.ReadLine());

             double result = Add(num1, num2);

             Console.WriteLine($"Result: {num1} + {num2} = {result}");
             Console.ReadKey();
          }

          public static double Add(double a, double b)
          {
             return a + b;
          }
```

2) Create a .NET Console Application that will find the factorial of the given number.

```
static void Main(string[] args)
    {
       Console.WriteLine("--- Factorial Calculator ---");
       Console.Write("Enter a number: ");
       string input = Console.ReadLine();

       int number = int.Parse(input);

       long factorialResult = 1;

       if (number == 0)
       {
          factorialResult = 1;
       }
```

```
        else
        {
            for (int i = 1; i <= number; i++)
            {
                factorialResult *= i;
            }
        }

        Console.WriteLine($"The factorial of {number} is: {factorialResult}");

        Console.ReadKey();
    }
```

3) Create a .NET Console Application that will find the cube of the given number.

```
static void Main(string[] args)
    {
        Console.Write("Enter an integer: ");
        string input = Console.ReadLine();

        int number = int.Parse(input); // assumes valid integer
input
        int cube = number * number * number;

        Console.WriteLine("The cube of " + number + " is " + cube);
        Console.ReadKey(); // keeps the console window open
    }
```

.NET CONSOLE APPLICATION IN C#

4) Create a .NET Console Application that will handle run time exception.

```
using System;

class Program
{
   static void Main()
   {
      int a = 10;
      int b = 0;
      int result;

      try
      {
         result = a / b;
         Console.WriteLine("Result: " + result);
      }
      catch (DivideByZeroException)
      {
         Console.WriteLine("You can't divide by zero!");
      }

      Console.ReadKey();
   }
```

.NET CONSOLE APPLICATION IN C#

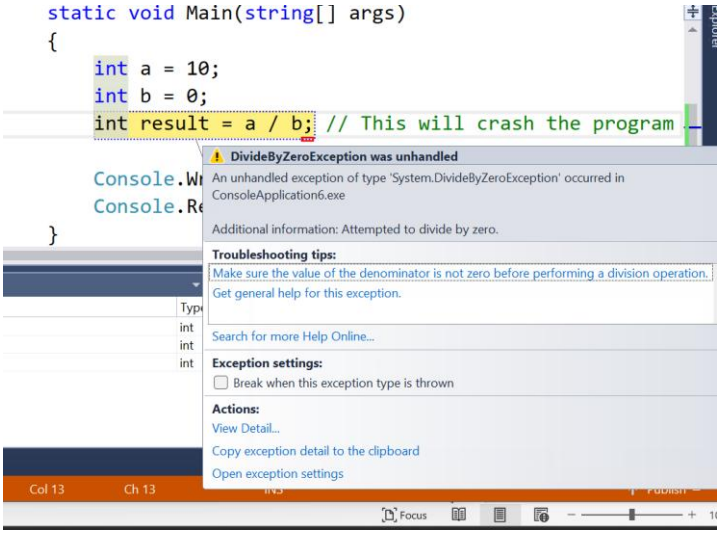
5) Create a .NET Console Application using UN structure Exception handling.

```
using System;

class Program
{
   static void Main()
   {
      int a = 10;
      int b = 0;
      int result = a / b; // This will crash the program

      Console.WriteLine("Result: " + result);
      Console.ReadKey();
   }
}
```

Output: -



.NET CONSOLE APPLICATION IN C#

6) Create a .NET Console Application that will match the given string using ternary operator.

```
using System;

class Program
{
   static void Main()
   {
      string str1 = "hello";
      string str2 = "hello";

      Console.WriteLine(str1 == str2 ? "Match" : "No Match");
   }
}
```

7) Create a .NET Console Application that will Join the given string.

```
using System;

class Program
{
   static void Main()
   {
      Console.Write("Enter first string: ");
      string str1 = Console.ReadLine();

      Console.Write("Enter second string: ");
      string str2 = Console.ReadLine();

      string result = str1 + str2;

      Console.WriteLine("Concatenated string: " + result);
   }
}
```

.NET CONSOLE APPLICATION IN C#

8) Create a .NET Console Application that uses Interface.

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

namespace ConsoleApplication18
{
  public interface IStudent
  {
    string Name { get; set; }
    double TotalMarks { get; }
    void DisplayStudentInfo();
  }

  public class RegularStudent : IStudent
  {
    public string Name { get; set; }
    private double mark1, mark2, mark3;

    public double TotalMarks
    {
      get { return mark1 + mark2 + mark3; }
    }

    public RegularStudent(string name, double m1, double m2, double m3)
    {
      Name = name; mark1 = m1; mark2 = m2; mark3 = m3;
    }

    public void DisplayStudentInfo()
    {
      Console.WriteLine($"{Name}: Total Marks = {TotalMarks}");
    }
  }

    class Program
  {
    static void Main(string[] args)
    {
      Console.WriteLine("--- Student Interface Demo ---");

      List<IStudent> students = new List<IStudent>();
      students.Add(new RegularStudent("TYBCA 1", 85, 90, 78));
      students.Add(new RegularStudent("TYBCA 2", 72, 65, 80));

      foreach (IStudent student in students)
      {
        student.DisplayStudentInfo();
      }
      Console.ReadKey();
    }
  }
}
```

.NET CONSOLE APPLICATION IN C#

9) Create a .NET Console Application that will use throw keyword in exception handling.

```
using System;

public class ExceptionDemo
{
  public static void CheckAge(int age)
  {
    if (age < 0)
    {
      throw new ArgumentException("Age cannot be negative.");
    }
    Console.WriteLine($"Age: {age}");
  }

  public static void Main(string [] args)
  {
    try
    {
      Console.Write("Enter age: ");
      int ageInput = int.Parse(Console.ReadLine());
      CheckAge(ageInput);
    }
    catch (ArgumentException ex)
    {
      Console.WriteLine($"Error: {ex.Message}");
    }
    catch (FormatException)
    {
      Console.WriteLine("Error: Invalid number.");
    }
    finally
    {
      Console.WriteLine("Done.");
    }
    Console.ReadKey();
  }
}
```

.NET CONSOLE APPLICATION IN C#

10) Create a .NET Console Application that will use of Error object in exception handling.

```
using System;

public class ExceptionDemo
{
   public static void Main(string[] args)
   {
      try
      {
         Console.Write("Enter age: ");
         int ageInput = int.Parse(Console.ReadLine());

       if (ageInput < 0)
         {
            throw new ArgumentException("Age cannot be negative.");
         }
         Console.WriteLine($"Age: {ageInput}");
      }
      catch (Exception ex) // Catching the general 'Exception' object
      {
         Console.WriteLine($"Error: {ex.Message}");
      }
      finally
      {
         Console.WriteLine("Done.");
      }
      Console.ReadKey();
   }
}
```