1. Write a C# Program to demonstrate Arrays ( 2D and Jagged)

1)a)2-D Array:

using System;

class TwoD

{

static void Main()

{

int t, i;

int[,] table = new int[3, 4];

for (t = 0; t < 3; ++t)

{

for (i = 0; i < 4; ++i)

{

table[t, i] = (t \* 4) + i + 1;

Console.Write(table[t, i] + " ");

}

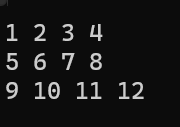
Console.WriteLine();

}

}

}

Output:



1)b)Jagged Array:

using System;

class Jagged

{

static void Main()

{

int[][] jagged = new int[3][];

jagged[0] = new int[4];

jagged[1] = new int[3];

jagged[2] = new int[5];

int i;

for (i = 0; i < 4; i++)

jagged[0][i] = i;

for (i = 0; i < 3; i++)

jagged[1][i] = i;

for (i = 0; i < 5; i++)

jagged[2][i] = i;

for (i = 0; i < 4; i++)

Console.Write(jagged[0][i] + " ");

Console.WriteLine();

for (i = 0; i < 3; i++)

Console.Write(jagged[1][i] + " ");

Console.WriteLine();

for (i = 0; i < 5; i++)

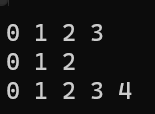
Console.Write(jagged[2][i] + " ");

Console.WriteLine();

}

}

Output:



2. Design a class to demonstrate String class methods.

using System;

using System.Globalization;

class StrOps

{

static void Main()

{

string str1 = "When it comes to .NET programming, C# is #1.";

string str2 = "When it comes to .NET programming, C# is #1.";

string str3 = "C# strings are powerful.";

string strUp, strLow;

Console.WriteLine("string 1: " + str1);

Console.WriteLine("Length of string 1: " + str1.Length);

Console.WriteLine("string 2: " + str2);

Console.WriteLine("Length of string 2: " + str2.Length);

Console.WriteLine("string 3: " + str3);

Console.WriteLine("Length of string 3: " + str3.Length);

strLow = str1.ToLower(CultureInfo.CurrentCulture);

strUp = str1.ToUpper(CultureInfo.CurrentCulture);

Console.WriteLine("Lowercase version of str1:\n " +strLow);

Console.WriteLine("Uppercase version of str1:\n " +strUp);

if (str1 == str2)

Console.WriteLine("str1 == str2");

else

Console.WriteLine("str1 != str2");

if (str1 == str3)

Console.WriteLine("str1 == str3");

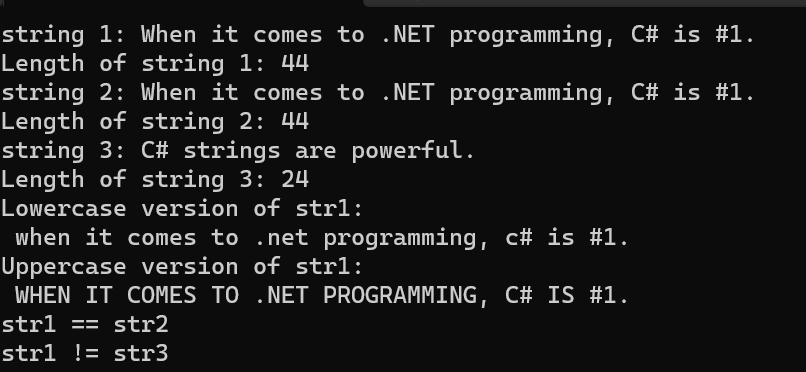
else

Console.WriteLine("str1 != str3");

}

}

Output:



3) Write a C# program to demonstrate ref, out & variable number of arguments.

3)a)ref

// Use ref to pass a value type by reference.

using System;

class RefTest

{

// This method changes its argument. Notice the use of ref.

public void Sqr(ref int i)

{

i = i \* i;

}

}

class RefDemo

{

static void Main()

{

RefTest ob = new RefTest();

int a = 10;

Console.WriteLine("a before call: " + a);

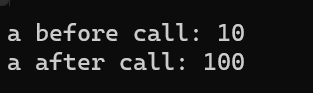
ob.Sqr(ref a); // notice the use of ref

Console.WriteLine("a after call: " + a);

}

}

Output:



3)b)out

// Use out.

using System;

class Decompose

{

/\* Decompose a floating-point value into its

integer and fractional parts. \*/

public int GetParts(double n, out double frac)

{

int whole;

whole = (int)n;

frac = n - whole; // pass fractional part back through frac

return whole; // return integer portion

}

}

class UseOut

{

static void Main()

{

Decompose ob = new Decompose();

int i;

double f;

i = ob.GetParts(10.125, out f);

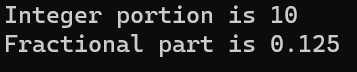
Console.WriteLine("Integer portion is " + i);

Console.WriteLine("Fractional part is " + f);

}

}

Output:



3)C) variable number of arguments.

// Use regular parameter with a params parameter.

using System;

class MyClass

{

public void ShowArgs(string msg, params int[] nums)

{

Console.Write(msg + ": ");

foreach (int i in nums)

Console.Write(i + " ");

Console.WriteLine();

}

}

class ParamsDemo2

{

static void Main()

{

MyClass ob = new MyClass();

ob.ShowArgs("Here are some integers",

1, 2, 3, 4, 5);

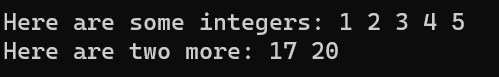
ob.ShowArgs("Here are two more",

17, 20);

}

}

Output:



4. Implement a class hierarchy with abstract classes, virtual methods & overriding.

4)a) abstract classes

using System;

public abstract class Vehicle

{

public abstract void display();

}

public class Bus : Vehicle

{

public override void display()

{

Console.WriteLine("Bus");

}

}

public class Car : Vehicle

{

public override void display()

{

Console.WriteLine("Car");

}

}

public class Motorcycle : Vehicle

{

public override void display()

{

Console.WriteLine("Motorcycle");

}

}

public class MyClass

{

public static void Main()

{

Vehicle v;

v = new Bus();

v.display();

v = new Car();

v.display();

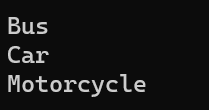
v = new Motorcycle();

v.display();

}

}

Output:



4)b) virtual methods

// Demonstrate a virtual method.

using System;

class Base

{

// Create virtual method in the base class.

public virtual void Who()

{

Console.WriteLine("Who() in Base");

}

}

class Derived1 : Base

{

// Override Who() in a derived class.

public override void Who()

{

Console.WriteLine("Who() in Derived1");

}

}

class Derived2 : Base

{

// Override Who() again in another derived class.

public override void Who()

{

Console.WriteLine("Who() in Derived2");

}

}

class OverrideDemo

{

static void Main()

{

Base baseOb = new Base();

Derived1 dOb1 = new Derived1();

Derived2 dOb2 = new Derived2();

Base baseRef; // a base class reference

baseRef = baseOb;

baseRef.Who();

baseRef = dOb1;

baseRef.Who();

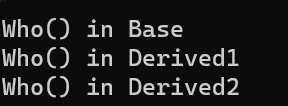
baseRef = dOb2;

baseRef.Who();

}

}

Output:



4)c)overriding.

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

namespace Hello\_Word

{

class baseClass

{

public virtual void Greetings()

{

Console.WriteLine("baseClass Saying Hello!");

}

}

class subClass : baseClass

{

public override void Greetings()

{

base.Greetings();

Console.WriteLine("subClass Saying Hello!");

}

}

class Program

{

static void Main(string[] args)

{

baseClass obj1 = new subClass();

obj1.Greetings();

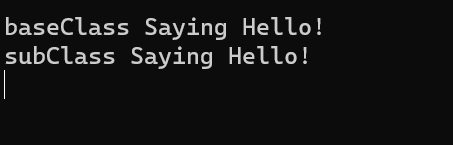
Console.ReadLine();

}

}

}

Output:



5. Write a C# program to demonstrate interfaces.

using System;

public interface ISeries

{

int Next

{

get;

set;

}

}

class ByTwos : ISeries

{

int val;

public ByTwos()

{

val = 0;

}

public int Next

{

get

{

val += 2;

return val;

}

set

{

val = value;

}

}

}

class SeriesDemo3

{

static void Main()

{

ByTwos ob = new ByTwos();

for (int i = 0; i < 5; i++)

Console.WriteLine("Next value is " + ob.Next);

Console.WriteLine("\nStarting at 21");

ob.Next = 21;

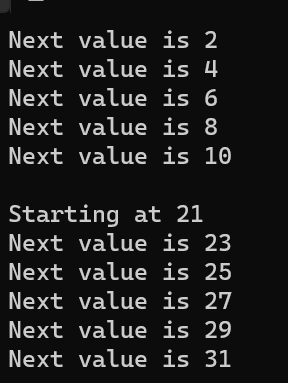
for (int i = 0; i < 5; i++)

Console.WriteLine("Next value is " + ob.Next);

}

}

Output:



6. Write a C# program to demonstrate exceptions.

using System;

class ExcDemo3

{

static void Main()

{

int[] numer = { 4, 8, 16, 32, 64, 128 };

int[] denom = { 2, 0, 4, 4, 0, 8 };

for (int i = 0; i < numer.Length; i++)

{

try

{

Console.WriteLine(numer[i] + " / " +

denom[i] + " is " +

numer[i] / denom[i]);

}

catch (DivideByZeroException)

{

Console.WriteLine("Can't divide by Zero!");

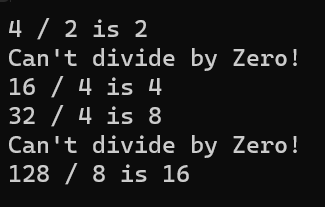
}

}

}

}

Output:



7. Design an ASP.NET application to demonstrate Web Controls.

Default.aspx

<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="Default.aspx.cs" Inherits="WEB\_CONTROLS.Default" %>

<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml">

<head runat="server">

<title>Web Controls Demo</title>

</head>

<body>

<form id="form1" runat="server">

<h2>Web Controls Demo</h2>

<asp:Label ID="lblName" runat="server" Text="Name:"></asp:Label>

<asp:TextBox ID="txtName" runat="server"></asp:TextBox>

<br /><br />

<asp:Label ID="lblEmail" runat="server" Text="Email:"></asp:Label>

<asp:TextBox ID="txtEmail" runat="server"></asp:TextBox>

<br /><br />

<asp:Button ID="btnSubmit" runat="server" Text="Submit" OnClick="btnSubmit\_Click" />

<hr />

<asp:Label ID="lblOutput" runat="server" Text=""></asp:Label>

</form>

</body>

</html>

Default.aspx.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.UI;

using System.Web.UI.WebControls;

namespace WEB\_CONTROLS

{

public partial class Default : System.Web.UI.Page

{

protected void Page\_Load(object sender, EventArgs e)

{

if (!IsPostBack)

{

// Initial page load logic goes here

}

}

protected void btnSubmit\_Click(object sender, EventArgs e)

{

string name = txtName.Text;

string email = txtEmail.Text;

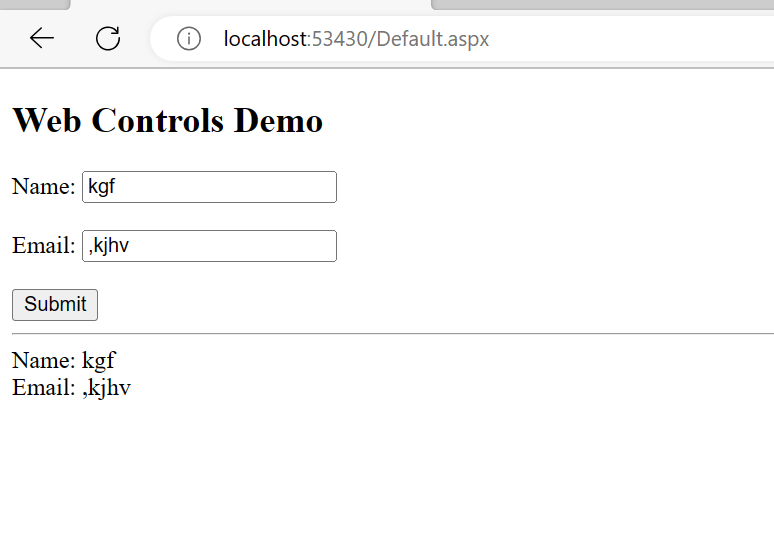
lblOutput.Text = $"Name: {name}<br />Email: {email}";

}

}

}

Output:



8. Design an ASP.NET application to demonstrate different List controls.

default.aspx

<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="default.aspx.cs" Inherits="list.\_default" %>

<!DOCTYPE html>

<html>

<head runat="server">

<title>CheckBoxTest</title>

</head>

<body>

<form runat="server">

<div>

Choose your favorite programming languages:<br /><br />

<asp:CheckBoxList ID="chklst" runat="server" /><br /><br />

<asp:Button ID="cmdOK" Text="OK" OnClick="cmdOK\_Click" runat="server" />

<br /><br />

<asp:Label ID="lblResult" runat="server" />

</div>

</form>

</body>

</html>

default.aspx.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.UI;

using System.Web.UI.WebControls;

namespace list

{

public partial class \_default : System.Web.UI.Page

{

protected void Page\_Load(object sender, EventArgs e)

{

if (!this.IsPostBack)

{

chklst.Items.Add("C");

chklst.Items.Add("C++");

chklst.Items.Add("C#");

chklst.Items.Add("Visual Basic 6.0");

chklst.Items.Add("VB.NET");

chklst.Items.Add("Pascal");

}

}

protected void cmdOK\_Click(object sender, EventArgs e)

{

lblResult.Text = "You chose:<b>";

foreach (ListItem lstItem in chklst.Items)

{

if (lstItem.Selected == true)

{

// Add text to label.

lblResult.Text += "<br />" + lstItem.Text;

}

}

lblResult.Text += "</b>";

}

}

}

Output:

