1. Write a C# program to print hello world.

using System;

namespace MyFirstProgram

{

internal class Program

{

static void Main(string[] args)

{

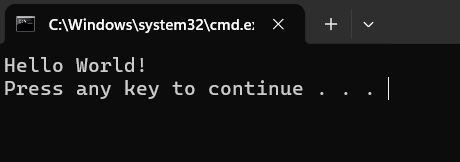
Console.WriteLine("Hello World!");

}

}

}

Output:



1. Write a program to demonstrate comments in C#.

using System;

namespace CommentExample

{

internal class Program

{

static void Main(string[] args)

{

Console.WriteLine("\tTab Use");

Console.WriteLine("this is a \nnewline example");

Console.WriteLine("BackSpace\be");

Console.WriteLine("This is an example of comments and escape sequence");

//This is a Single Line Comment

/\*

\* This

\* is

\* a

\* Multi

\* Line

\* Comment

\*/

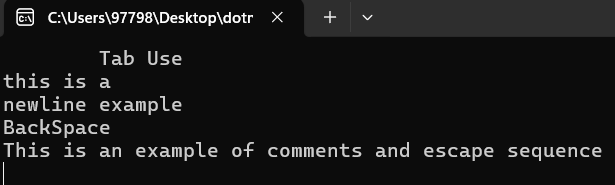
Console.ReadKey();

}

}

}

Output:



1. Write a program to demonstrate variables in C#.

using System;

namespace VariableExample

{

internal class Program

{

static void Main(string[] args)

{

int x; //declaration

x = 123; //initialization

int y = 321; // intialization + declartion

int z = x + y;

int age = 19; // Whole Integer

double weight = 55.5; // decimal number

bool happy = true; // true or false

char symbol = '@';

string name = "archana";

Console.WriteLine("value of x:" + x);

Console.WriteLine("value of y:" + y);

Console.WriteLine("value of z:" + z);

Console.WriteLine("your age is " + age);

Console.WriteLine("your height is " + weight + "kg");

Console.WriteLine("Are you happy? " + happy);

Console.WriteLine("Your symbol is " + symbol);

Console.WriteLine("Your name is " + name);

String userName = symbol + name + age;

Console.WriteLine("Your username is " + userName);

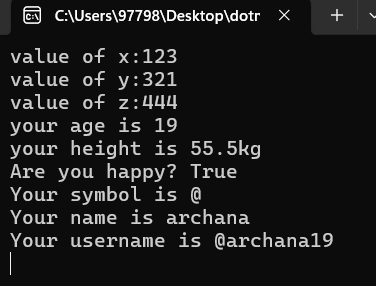
Console.ReadKey();

}

}

}

Output:



1. Write a program to demonstrate constant and typecasting in C#.

using System;

namespace Constant

{

internal class Program

{

static void Main(string[] args)

{

//constants = immutable values which are known at compile time

// and don't change for life of the program

//typecasting = Converting a value to a different data type

//Useful when we accept user input(string) Different data

//types can do different things

const double pi = 3.14159;

Console.WriteLine(pi);

double a = 3.14;

int b= Convert.ToInt32(a);

Console.WriteLine(b);

Console.WriteLine(b.GetType());

int c = 123;

double d = Convert.ToDouble(c) + 0.1;

Console.WriteLine(d);

Console.WriteLine(d.GetType());

int e = 321;

string f = Convert.ToString(e);

Console.WriteLine(f);

Console.WriteLine(f.GetType());

string g = "$";

char h = Convert.ToChar(g);

Console.WriteLine(h);

Console.WriteLine(h.GetType());

string i = "true";

bool j = Convert.ToBoolean(i);

Console.WriteLine(j);

Console.WriteLine(j.GetType());

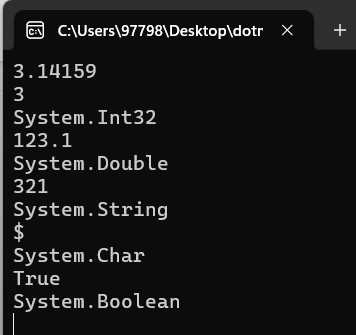
Console.ReadKey();

}

}

}

Output:



1. Write a program to demonstrate user input in C#.

using System;

namespace UserInput

{

internal class Program

{

static void Main(string[] args)

{

Console.WriteLine("What is your Name?");

String name = Console.ReadLine();

Console.WriteLine("What's your age?");

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

Console.WriteLine("Hello " + name + "!");

Console.WriteLine("You are " + age + " years old.");

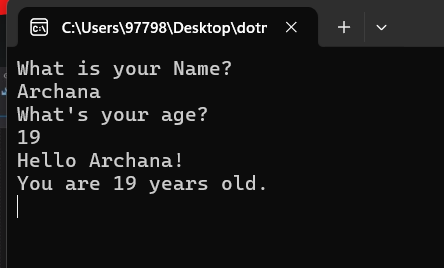
Console.ReadKey();

}

}

}

Output:



1. Write a program to demonstrate arithmetic operator in C#.

using System;

namespace ArithematicOperator

{

internal class Program

{

static void Main(string[] args)

{

int numofapple = 20;

numofapple = numofapple + 1;

Console.WriteLine("the number of apple:" + numofapple);

int numoforange = 30;

numoforange \*= 5;

Console.WriteLine("the number of orange:" + numoforange);

int numofgrapes = 20;

numofgrapes = numofgrapes--;

Console.WriteLine("the number of grapes:" + numofgrapes);

int numofbanana = 70;

numofbanana = numofbanana/ 2;

Console.WriteLine("the number of banana:" + numofbanana);

int numoffriends = 70;

numoffriends = numoffriends % 6;

Console.WriteLine("the number of friends:" + numoffriends);

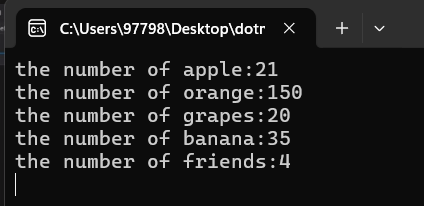
Console.ReadKey();

}

}

}

Output:



1. Write a program to demonstrate unary operator overloading in C#.

using System;

namespace unaryoperator

{

class Unaryoperator

{

public int number1, number2;

public Unaryoperator(int num1, int num2)

{

number1 = num1;

number2 = num2;

}

// function to perform operation

//by changing sign of integers

public static Unaryoperator operator -(Unaryoperator u1)

{

u1.number1 = -u1.number1;

u1.number2 = -u1.number2;

return u1;

}

public void Print()

{

Console.WriteLine("Number1 =" + number1);

Console.WriteLine("Number2 =" + number2);

} }

class Program

{

static void Main(string[] args)

{

// using overloaded - operator

// with the class object

Unaryoperator calc = new Unaryoperator(15, -25);

calc = -calc;

// To display the result

calc.Print();

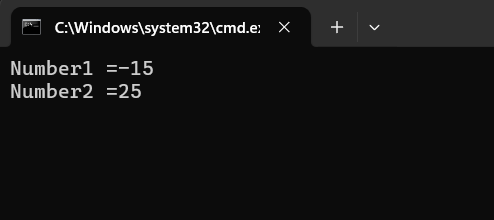
Console.ReadKey();

}

}

}

Output:



1. Write a program to demonstrate binary operator overloading in C#.

using System;

namespace binaryoperator

{

class Binaryoperator

{

public int number = 0;

public Binaryoperator()

{

}

public Binaryoperator(int n)

{

number = n;

}

public static Binaryoperator operator +(Binaryoperator bi1, Binaryoperator bi2)

{

Binaryoperator bi3 = new Binaryoperator(0);

bi3.number = bi1.number + bi2.number;

return bi3;

}

public void display()

{

Console.WriteLine("{0}", "number:" + number);

} }

internal class Program

{

static void Main(string[] args)

{

Binaryoperator num1 = new Binaryoperator(500);

Binaryoperator num2 = new Binaryoperator(450);

Binaryoperator num3 = new Binaryoperator();

num3 = num1 + num2;

num1.display();

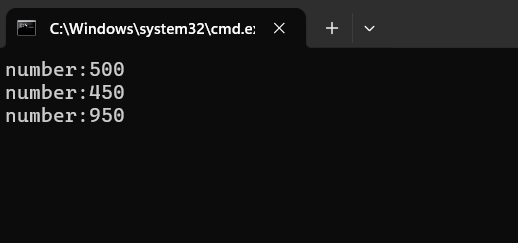
num2.display();

num3.display();

Console.ReadKey();

} } }

Output:



1. Write a program to implement delegate in C#.

using System;

namespace Delegate

{

class Program

{

//Step2: create delegate type

public delegate void DelegateDelegate();

static void Main(string[] args)

{

//Step3: create delegate instance

//Step4: point delegate to method

DelegateDelegate del = new DelegateDelegate(printString);

//Step5: call a method via delegate

del.Invoke();

Console.ReadKey();

}

//Step1: create a method to be callled via delegate

public static void printString()

{

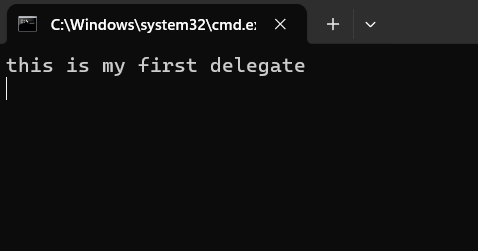
Console.WriteLine("this is my first delegate");

}

}

}

Output:



1. Write a program to implement Singlecast Delegate in C#.

using System;

namespace SingleDelegate

{

class Program

{

public static int square(int n)

{

return n \*n;

}

public delegate int squaredelegate(int num);

static void Main(string[] args)

{

squaredelegate del = new squaredelegate(square);

int result = del.Invoke(5);

Console.WriteLine("square of 5 is:" + result);

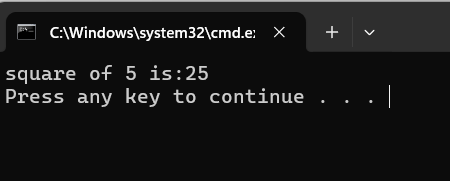
Console.ReadKey();

}

}

}

Output:



1. Write a program to implement Multicast Delegate in C#.

using System;

namespace MulticastDelegate

{

internal class Program

{

public static int square(int n)

{

return n \* n;

}

public static int cube(int n)

{

return n \* n \* n;

}

public delegate int multicastdelegate(int num);

static void Main(string[] args)

{

multicastdelegate del = new multicastdelegate(square);

del += square;

int squareresult = del(5);

Console.WriteLine("square of 5:" + squareresult);

del += cube;

int cuberesult = del(5);

Console.WriteLine("cube of 5:" + cuberesult);

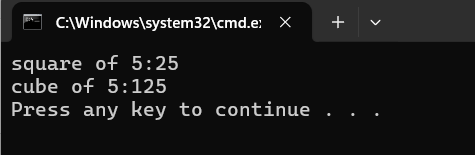
Console.ReadKey();

}

}

}

Output:



1. Write a program to implement event handler in c#.

using System;

namespace EventExample

{

public delegate void EventHandler();

internal class Program

{

public static event EventHandler add;

static void USA()

{

Console.WriteLine("This is American");

}

static void NEPAL()

{

Console.WriteLine("This is Nepali");

}

static void INDIA()

{

Console.WriteLine("This is Indiaan");

}

static void Main(string[] args)

{

add += new EventHandler(USA); // subscribe

add += new EventHandler(NEPAL);

add += new EventHandler(INDIA);

add.Invoke();

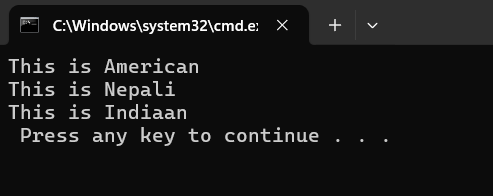
Console.ReadKey();

}

}

}

Output:



1. Write a C# program to print Fibonacci Series up to term ‘n’.

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace fibonacci\_serie

{

internal class Program

{

static void Main(string[] args)

{

Console.Write("Enter the number of terms for Fibonacci series: ");

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

Console.WriteLine("Fibonacci Series up to term {0}:", n);

PrintFibonacciSeries(n);

}

static void PrintFibonacciSeries(int n)

{

int firstTerm = 0, secondTerm = 1, nextTerm;

Console.Write("{0} {1} ", firstTerm, secondTerm);

for (int i = 3; i <= n; i++)

{

nextTerm = firstTerm + secondTerm;

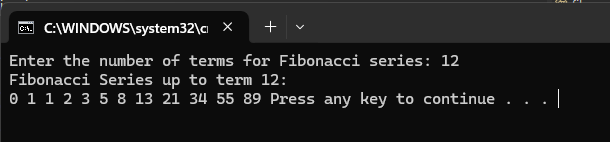
Console.Write("{0} ", nextTerm);

firstTerm = secondTerm;

secondTerm = nextTerm;

} } } }

Output:



1. Write a C# program to create generic class and generic method.

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace generic\_class\_nad\_method

{

using System;

// Generic class definition

public class MyGenericClass<T>

{

private T genericField;

// Constructor

public MyGenericClass(T value)

{

genericField = value;

}

// Generic method

public void DisplayGenericValue()

{

Console.WriteLine("Generic Field Value: " + genericField);

} }

class Program

{

static void Main(string[] args)

{

// Creating instances of MyGenericClass with different types

MyGenericClass<int> intObj = new MyGenericClass<int>(10);

MyGenericClass<string> stringObj = new MyGenericClass<string>("Hello, Generics!");

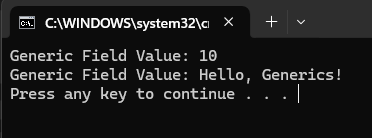
// Calling generic methods

intObj.DisplayGenericValue();

stringObj.DisplayGenericValue();

} } }

Output:



1. Write a a C# program to implement Throwing exceptions.

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace throwing\_exception

{

internal class Program

{

static void Check (int age)

{

if(age<18)

throw new ArithmeticException("Not Eligible to Vote!");

}

static void Main(string[] args)

{

try

{

Check(15);

}

catch(Exception ex)

{

Console.WriteLine(ex);

}

}

}

Output:

A screenshot of a phone

Description automatically generated