|  |
| --- |
| .NET |
| V.V.P CE Sem 6 .net |
| Abhay Katharotiya |

|  |
| --- |
| 160470107028  V.V.P CE Sem 6 |

Contents

[AIM : Introduction to C# 3](#_Toc1388258)

[AIM: Inheritance 11](#_Toc1388260)

[Program 1 11](#_Toc1388261)

[Program 2 13](#_Toc1388262)

[Program 3 14](#_Toc1388263)

[Program 4 15](#_Toc1388264)

[AIM: Method & constructor overloading 17](#_Toc1388266)

[Program 1 17](#_Toc1388267)

[Program 2 20](#_Toc1388268)

[AIM: Reflection API 22](#_Toc1388269)

[Program 1 22](#_Toc1388270)

# AIM : Introduction to C#

Variables:

Initialization

Scope

Constant

Predefined Data Types

Value Types

Reference TYpes

Flow Control

Conditional Statements(if, switch)

Loop(for, while, dowhile, foreach)

Jump(goto, break, continue, return)

Eumerations

Passing Arguments

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace aim

{

class Program

{

static int newint=100;

public enum TimeOfDay

{

Morning = 0,

Afternoon = 1,

Evening = 2

}

public static void Main(string[] args)

{

Console.WriteLine("\n integer types");

sbyte sb = 10;

short s = 33;

int i = 10;

long l = 33L;

byte b = 22;

ushort us = 33;

uint ul = 33u;

ulong ulo = 33ul;

Console.WriteLine("{0},{1},{2},{3},{4},{5},{6},{7}", sb, s, i, l, b, us, ul, ulo);

float f = 1.122345656767f;

double d = 12.1234455657878797;

Console.Write("\nFloat and Double:\n");

Console.WriteLine("{0} and \n{1}", f, d);

decimal dec=111.666666666666666666666M;

Console.WriteLine("decimal:\n{0} ",dec);

Console.WriteLine("\nBoolean:");

bool boolean =true;

Console.WriteLine("Status: " + boolean);

// Console.ReadLine();

char character ='d';

Console.WriteLine(character);

character = '\0';

Console.WriteLine("Now null: " + character);

object o1 = "Hi, I am ALICE";

object o2 = 15.3454365;

string strObj = o1 as string;

Console.WriteLine(strObj);

Console.WriteLine(o1.GetHashCode() + " " + o1.GetType());

Console.WriteLine(o2.GetHashCode() + " " + o2.GetType());

Console.WriteLine(o1.Equals(o2));

string s1, s2;

s1 = "this is string";

s2 = s1;

Console.WriteLine("S1 is: {0} and s2 is {1}", s1, s2);

s2 = "other string";

Console.WriteLine("S1 is: {0} and s2 is {1}", s1, s2);

s1 = "c:C:\\Users\\Dell\\source\\repos\\aim";

Console.WriteLine(s1);

s1 = @"c:C:\Users\Dell\source\repos\aim\aim";

Console.WriteLine(s1);

s1 = @"We can also write

like this";

Console.WriteLine(s1);

bool isZero;

Console.WriteLine("\nFlow Control: (if)\ni is " + i);

if (i == 10)

{

isZero = true;

Console.WriteLine("i is Zero {0}",isZero);

}

else

{

isZero = false;

Console.WriteLine("i is Non - zero");

}

int integerA = 1;

Console.WriteLine("\nSwitch:");

switch (integerA)

{

case 1:

Console.WriteLine("integerA = 1");

break;

case 2:

Console.WriteLine("integerA = 2");

//goto case 3;

break;

case 3:

Console.WriteLine("integerA = 3");

break;

default:

Console.WriteLine("integerA is not 1, 2, or 3");

break;

}

WriteGreeting(TimeOfDay.Morning);

Console.WriteLine("Argument is: {0}",args[1]);

void WriteGreeting(TimeOfDay timeOfDay)

{

switch (timeOfDay)

{

case TimeOfDay.Morning:

Console.WriteLine("Good morning!");

break;

case TimeOfDay.Afternoon:

Console.WriteLine("Good afternoon!");

break;

case TimeOfDay.Evening:

Console.WriteLine("Good evening!");

break;

default:

Console.WriteLine("Hello!");

break;

}

}

Console.WriteLine("Scope of Variables.\n1:");

int newint=0;

int j;

for (/\*int\*/ j = 0; j < 2; j++) //removing comment from for loop will raise error

{

//int j;

//uncomment above line to error "A local variable named 'j' cannot be declared in this

//scope because it would give a different meaning to 'j', which is already

//used in a 'parent or current' scope to denote something else"

Console.Write("{0} {1}\n", newint, Program.newint);

}

Console.WriteLine("2:");

for (int k = 0; k < 3; k++)

{

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

}//Scope of k ends here

Console.Write("\n");

//Console.Write(k);

//uncomment above line to see error "The name 'k' does not exist in the current context"

for (int k = 3; k > 0; k--)

{

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

}//scope of k ends here again

Console.WriteLine("Constants");

const int valConst = 100; // This value cannot be changed.

Console.WriteLine("{0} is constant value", valConst);

//valConst = 45;

//uncomment above line to see error "The left-hand side of an assignment must be a variable, property or indexer"

//const only allow constant variables into the expression

const int valConst2 = valConst + 9 /\* + j\*/;

//remove comments from the above line to see error "The expression being assigned to 'valConst2' must be constant"

Console.WriteLine("Another Constant: {0}", valConst2);

Console.WriteLine("\nPredefined Data Types\n\nValue Types and Reference Types");

//Value Types

int vali = 2, valj = vali;

Console.WriteLine("vali is: {0} and valj is: {1}", vali, valj);

valj = 90;

Console.WriteLine("vali is: {0} and valj is: {1}", vali, valj);

//Referece Types

Vector x, y;

x = new Vector();

x.value = 3;

y = x;

Console.WriteLine("x is: {0} and y is:{1}", x.value, y.value);

y.value = 234;

Console.WriteLine("x is: {0} and y is:{1}", x.value, y.value);

//If a variable is a reference, it is possible to indicate that it does not refer to any object by setting its value to null:

y = null;

//Console.Write("Value for y is: " + y.value);

//uncomment above line to see runtime exception "System.NullReferenceException: Object reference not set to an instance of an object."

//CTS

}

public class Vector

{

public int value;

}

}

}

# 

# AIM: Inheritance

## Program 1

Perform following programs in c#.

1. Write console based program in code behind language VB or C# to print following pattern.

@ @ @ @ @

@ @ @ @

@ @ @

@ @

@

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace pattern1

{

class Program

{

static void Main(string[] args)

{

for(int i=5;i>0;i--)

{

for (int j = i; j > 0; j--)

{

Console.Write("@");

}

Console.WriteLine(" ");

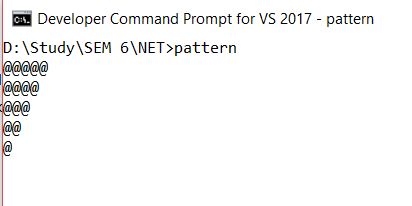
}

Console.ReadKey();

}

}

}



## Program 2

2.Write console based program in code behind language VB or C# to print following pattern.

1

1 2

1 2 3

1 2 3 4

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace pattern2

{

class Program

{

static void Main(string[] args)

{

for(int i=1;i<=5;i++)

{

for(int j=i;j>0;j--)

{

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

}

Console.WriteLine("");

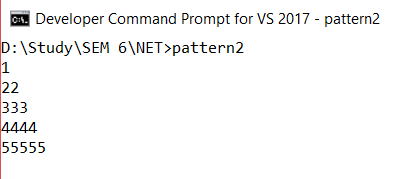
}

Console.ReadKey();

}

}

}



## Program 3

3. Write C# code to prompt a user to input his/her name and country name and then the output will be shown as an example below:

Hello Ram from country India

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace country

{

class Program

{

static void Main(string[] args)

{

string name;

string country;

Console.WriteLine("enter your name:");

name=Console.ReadLine();

Console.WriteLine("enter your country:");

country = Console.ReadLine();

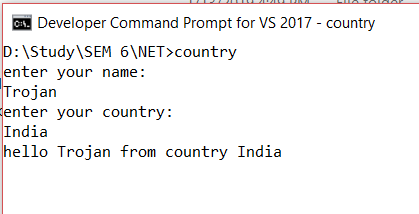
Console.WriteLine("hello {0} from country {1}",name,country);

Console.ReadKey();

}

}

}



## Program 4

4. What is inheritance? Create C# console application to define Car class and derive Maruti and Mahindra from it to demonstrate inheritance.

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace practical2.\_3

{

class car

{

public void Method1()

{

Console.WriteLine("this is the method of car class");

}

}

class maruti:car

{

public void method2()

{

Console.WriteLine("this is the method of maruti");

Console.ReadKey();

}

}

class mahindra:car

{

public void method3()

{

Console.WriteLine("this is the method of mahindra");

}

}

class Program

{

static void Main(string[] args)

{

mahindra m = new mahindra();

maruti m1 = new maruti();

m.Method1();

m1.Method1();

Console.ReadKey();

}

}

}

# 

# AIM: Method & constructor overloading

## Program 1

Write a c# program to add two integers, two vectors and two metric using method overloading.

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

namespace p2

{

public class P3\_1

{

public int add(int a, int b) {

return a + b;

}

public static Vector add(Vector v1,Vector v2) { Vector v= new Vector();

v.a = v1.a + v2.a; v.b = v1.b + v2.b;

return v;

}

public static int[,] add(int[,] a, int[,] b) {

int[,] s = new int[2, 2];

for (int i = 0; i < 2; i++) {

for (int j = 0; j < 2; j++) {

s[i, j] = a[i, j] + b[i, j];

}

}

return s;

}

public static void Main(String[] ar) {

int n,n1, n2;

Vector v = new Vector();

Console.WriteLine("Enter Number 1:");

n1 = Convert.ToInt32(Console.ReadLine());

Console.WriteLine("Enter Number 2:");

n2 = Convert.ToInt32(Console.ReadLine());

n = n1 + n2;

Console.WriteLine("Addition of Number:{0}", n);

Console.WriteLine("Enter Vector 1:");

n1 = Convert.ToInt32(Console.ReadLine()); n2 = Convert.ToInt32(Console.ReadLine());

Vector v1 = new Vector(n1,n2);

Console.WriteLine("Enter Vector 2:");

n1 =Convert.ToInt32(Console.ReadLine()); n2 = Convert.ToInt32(Console.ReadLine()); Vector v2 = new Vector(n1,n2);

v = add(v1, v2);

Console.WriteLine("Addition of vector: <{0}, {1}>",v.a,v.b);

int[,] a = new int[,] { { 1, 2 }, { 3, 4 } };

int[,] b = new int[,] { { 5, 6 }, { 7, 8 } };

int[,] c = add(a, b); Console.WriteLine("Addition of two matrics:");

for (int z = 0; z < 2; z++) {

for (int m = 0; m < 2; m++) {

Console.WriteLine("Addition: "+ c[z, m]);

}

}

Console.ReadKey();

}

}

public class Vector {

public int a, b;

public Vector() { }

public Vector(int a, int b)

{

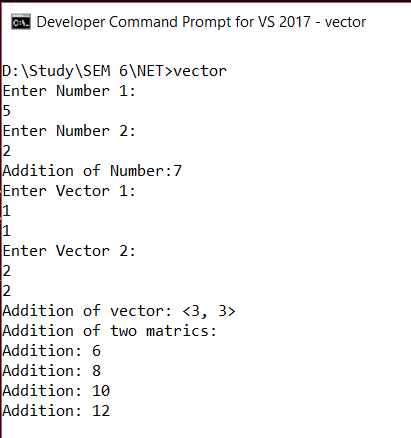
this.a = a;

this.b = b;

}

}

}



## Program 2

Write a c# program that create student object. Overload constructor to create new instant with following details.

1. Name

2. Name, Enrollment

3. Name, Enrollment, Branch

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Reflection;

namespace p3a1

{

class Program

{

public int ID { get; set; }

public string Name { get; set; }

String name, branch;

int enrol;

public Program(String name)

{

this.name = name;

Console.WriteLine("constructor 1:" + name);

}

public Program(String name, int enrol)

{

this.name = name;

this.enrol = enrol;

Console.WriteLine("constructor 2:" + name + " " + enrol);

}

public Program(String name, int enrol, String branch)

{

this.name = name;

this.enrol = enrol;

this.branch = branch;

Console.WriteLine("constructor 3:" + name + " " + enrol + " " + branch);

}

static void Main(string[] args)

{

Program p1 = new Program("Dhyey");

Program p2 = new Program("Dhyey ", 60);

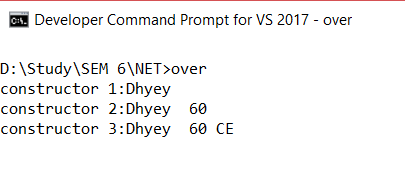
Program p3 = new Program("Dhyey ", 60, "CE");

Console.ReadLine();

}

}

}



# AIM: Reflection API

## Program 1

Create a c# program to find Methods, Properties and Constructors from class of running program.

|  |
| --- |
| using System; |
|  |

|  |
| --- |
| using System.Reflection; |
|  |

|  |
| --- |
| namespace ReflectionExample |
|  |

|  |
| --- |
| { |
|  |

|  |
| --- |
| class MainClass |
|  |

|  |
| --- |
| { |
|  |

|  |
| --- |
| static void Main() |
|  |

|  |
| --- |
| { |
|  |

|  |
| --- |
| Type T = Type.GetType("ReflectionExample.Customer"); |
|  |

|  |
| --- |
| MethodInfo[] methods = T.GetMethods(); |
|  |

|  |
| --- |
| foreach (MethodInfo method in methods) |
|  |

|  |
| --- |
| { |
|  |

|  |
| --- |
| Console.WriteLine(method.ReturnType + " " + method.Name); |
|  |

|  |
| --- |
| } |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| PropertyInfo[] properties = T.GetProperties(); |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| Console.WriteLine("\nProperties"); |
|  |

|  |
| --- |
| foreach (PropertyInfo property in properties) |
|  |

|  |
| --- |
| { |
|  |

|  |
| --- |
| Console.WriteLine(property.PropertyType+" "+ property.Name); |
|  |

|  |
| --- |
| } |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| Console.WriteLine("\nConstructors"); |
|  |

|  |
| --- |
| ConstructorInfo[] constructors = T.GetConstructors(); |
|  |

|  |
| --- |
| foreach (ConstructorInfo constructor in constructors) |
|  |

|  |
| --- |
| { |
|  |

|  |
| --- |
| Console.WriteLine(constructor.ToString()); |
|  |

|  |
| --- |
| } |
|  |

|  |
| --- |
| } |
|  |

|  |
| --- |
| } |
|  |

|  |
| --- |
| class Customer |
|  |

|  |
| --- |
| { |
|  |

|  |
| --- |
| public int ID { get; set; } |
|  |

|  |
| --- |
| public string Name { get; set; } |
|  |

|  |
| --- |
| public Customer(int ID, string Name) |
|  |

|  |
| --- |
| { |
|  |

|  |
| --- |
| this.ID = ID; |
|  |

|  |
| --- |
| this.Name = Name; |
|  |

|  |
| --- |
| } |
|  |

|  |
| --- |
| public Customer() |
|  |

|  |
| --- |
| { |
|  |

|  |
| --- |
| this.ID = -1; |
|  |

|  |
| --- |
| this.Name = string.Empty; |
|  |

|  |
| --- |
| } |
|  |

|  |
| --- |
| public void printID() |
|  |

|  |
| --- |
| { |
|  |

|  |
| --- |
| Console.WriteLine("ID is: {0}", this.ID); |
|  |

|  |
| --- |
| } |
|  |

|  |
| --- |
| public void printName() |
|  |

|  |
| --- |
| { |
|  |

|  |
| --- |
| Console.WriteLine("Name is: {0}", this.Name); |
|  |

|  |
| --- |
| } |
|  |

|  |
| --- |
| } |
|  |

}

