IDE : Integrated Development Environment

DotNet Framework

MS DOS > Operating System, It is character based

Windows Operating System , it is Graphical based (GUI)

Programming Languages : C , C++ , Java , Pascal, Fortran, COBOL : CUI

C++ > Visual C++

Basic > Visual Basic

Foxpro > Visual Foxpro

DotNet Framework : It allows multiple languages to be used together

C++ > Complier > Binary Form

Java > Interpreter > Binary Form

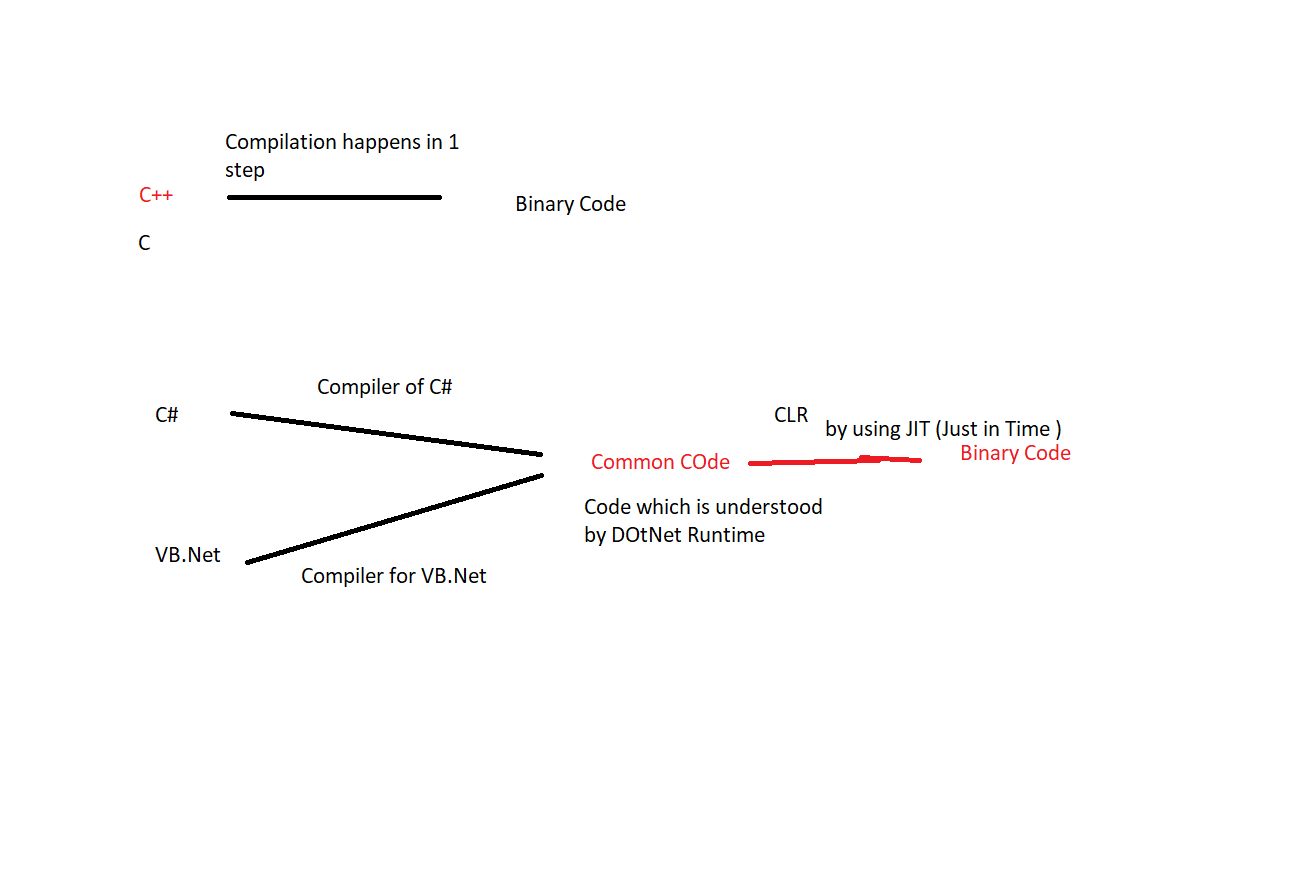
Compilation is done in 1 step from C , C++ , VBasic, Java to Binary Form

Compilation in Dotnet is done in 2 steps

C# > MSIL > Binary Form

Visual Basic > MSIL >Binary Form

MSIL > Microsoft Intermediate Language



Components in DotNet Framework

1. CLR > Common Language Runtime

It performs

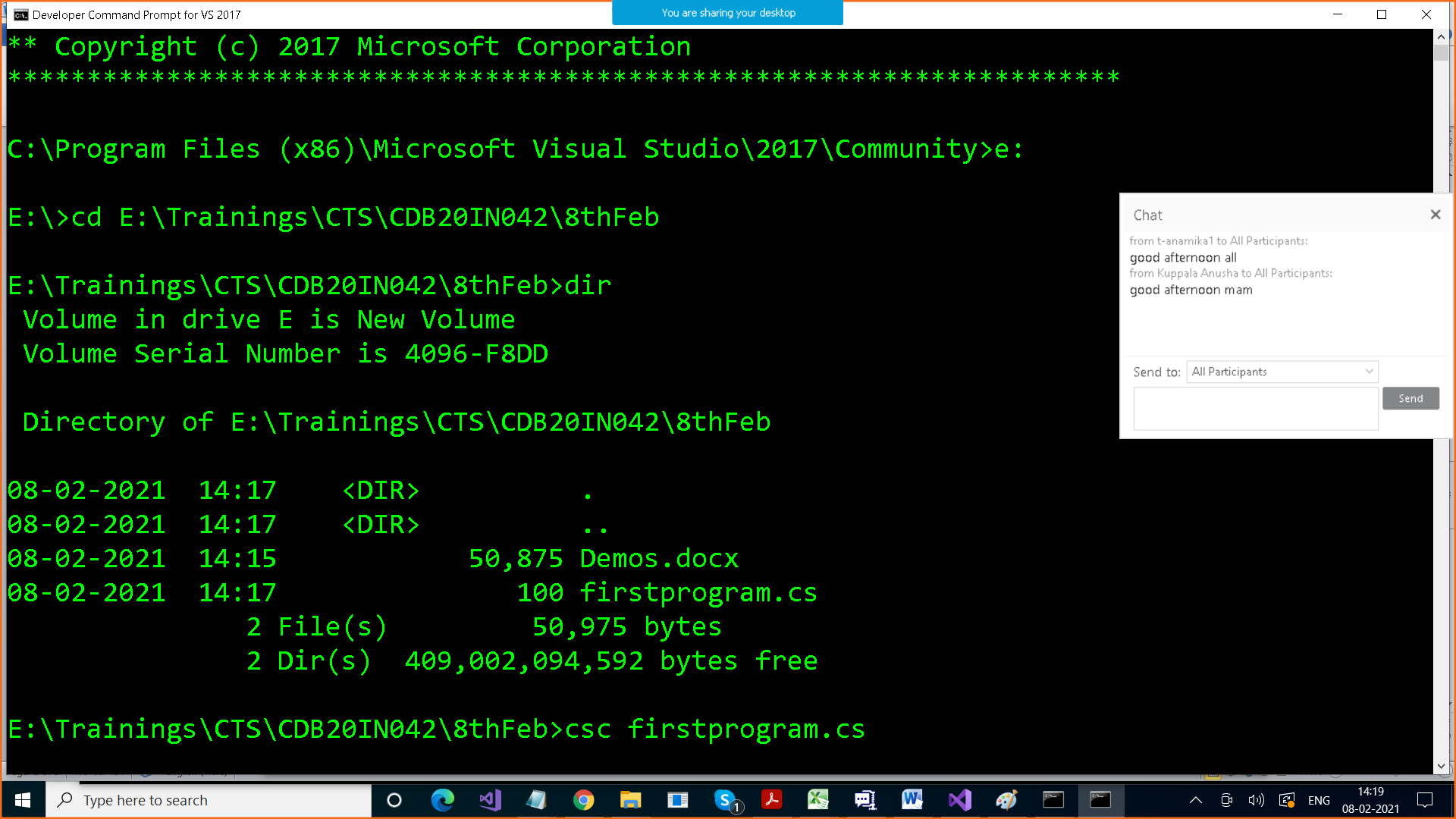
* Memory Allocation
* Garbage Collection
* Code Compilation

1. CTS(Common Type System)

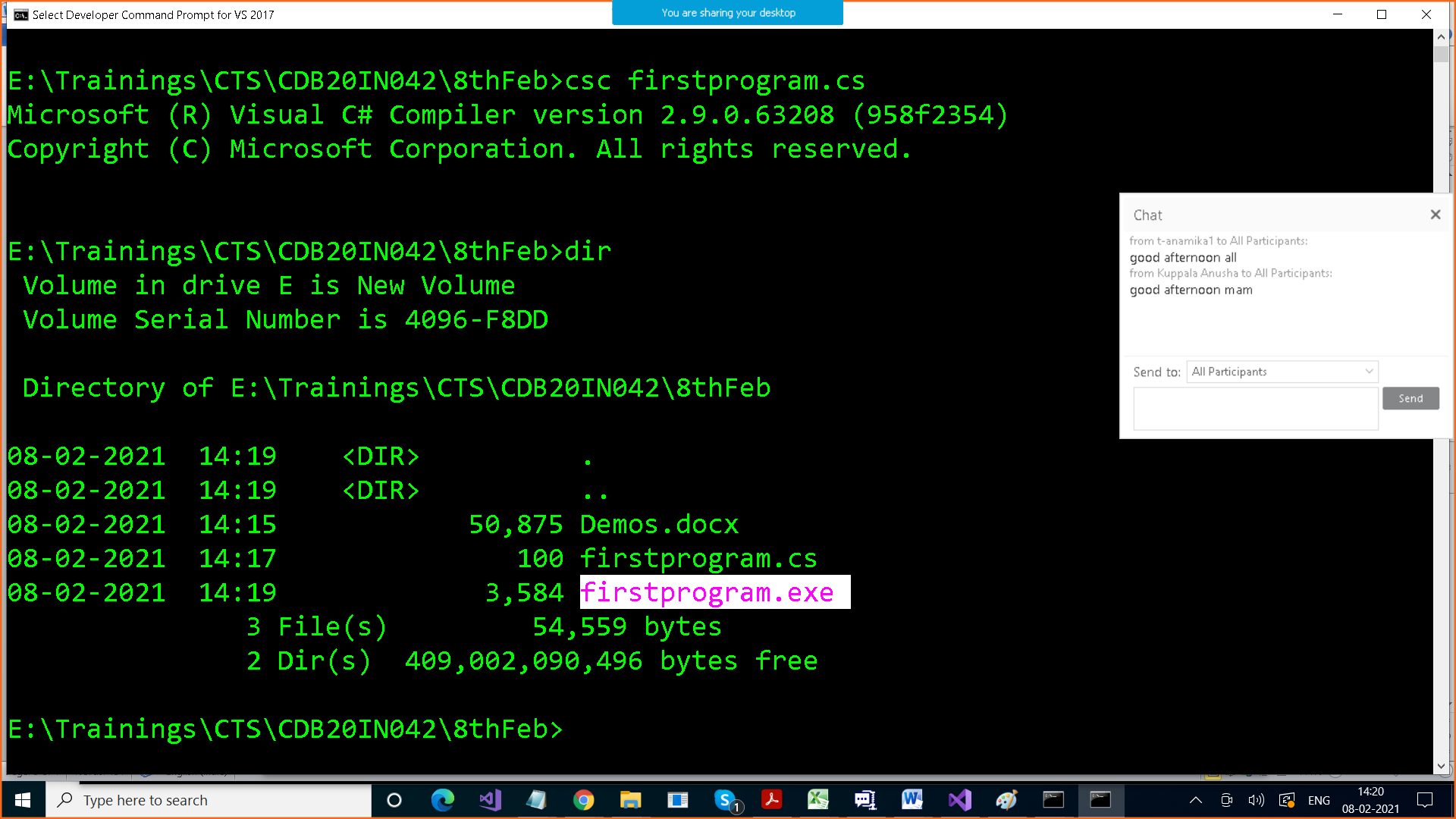
C# - Int > System.Integer

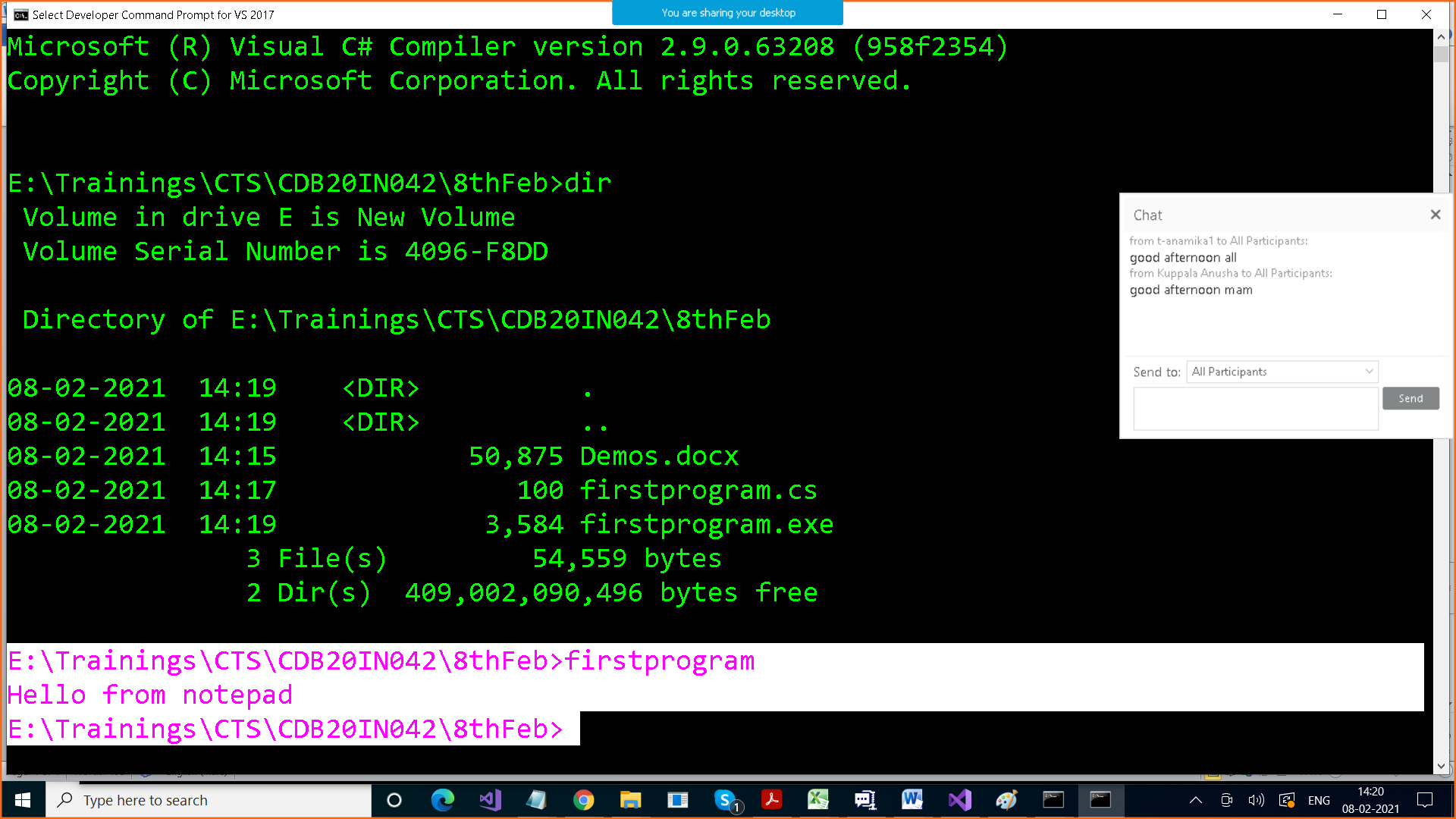
VB.Net integer

1. CLS(Common Language Specification) (Guidelines using which this conversion happens)



We get after compiling exe also



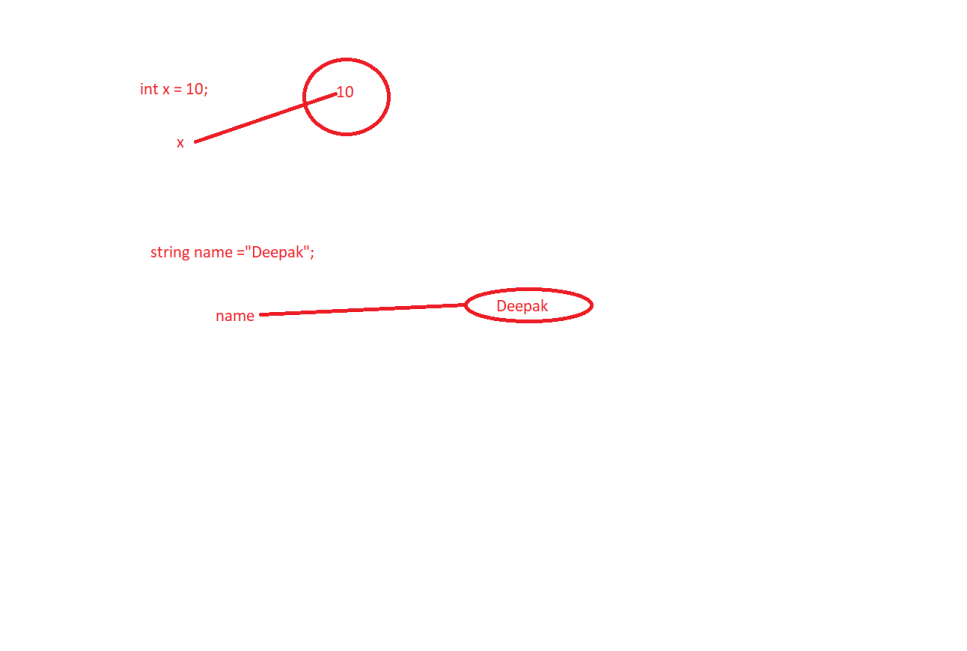


Variables : They are named memory where we store some value

Variables are of 2 types

1. Value Type
2. Reference Type

Value Type

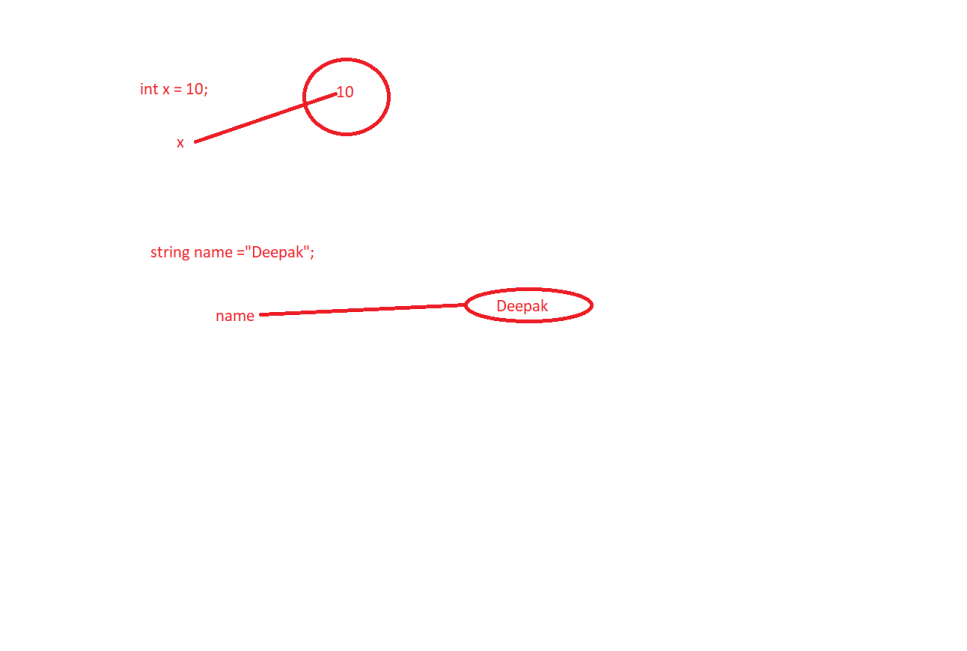


Variables of how many types

* Value Type : which stores the value . int , float , Boolean, char, struct, enum
* Reference Type : which stores the reference of the variable , string , class

When we declare varibale , they get memory and are stored somewhere in memory

Value type varibales are stored in stack

Reference type variables are stored in heap

Value type variables are stored in stack Reference type variables are stored in heap

|  |
| --- |
|  |
|  |
|  |
|  |
|  |
| + |
| 40 |
| 10  Deepak |
| 90 |

String name = “Deepak”

Int x =10;

|  |
| --- |
|  |
|  |
|  |
|  |
|  |
| + |
| 20 |
| 5 |

Garbage Collector runs on reference type variables

Methods : Modularity

using System;

class Prog

{

static void Add(int x, int y)

{

Console.WriteLine("Result of sum is " + ( x+y));

}

static void Subtract(int x, int y)

{

Console.WriteLine("Result of subtraction is " + (x - y));

}

static void Product(int x, int y)

{

Console.WriteLine("Result of product is " + (x \* y));

}

static void Divide(int x, int y)

{

Console.WriteLine("Result of division is " + (x / y));

}

static void Main()

{

Add(10, 20);

Subtract(10, 20);

Product(10, 20);

Divide(20, 10);

}

}

Debugging > F10 , F11

F10 < Step Over

F11 < Step Into

Polymorphism > One name , different forms. It is achieved by Method Overloading and Method Overriding

Method Overloading > Overloading same function with different type and no of parameters

We keep function name same and change the parameters number , type , or sequence

Problems before Method Overlaoding

using System;

class Prog

{

static void Add1(int x, int y)

{

Console.WriteLine("Result of sum is " + ( x+y));

}

static void Add2(int x, int y , int a)

{

Console.WriteLine("Result of sum is " + (x + y + a));

}

static void Add3(float x, float y)

{

Console.WriteLine("Result of sum is " + (x + y ));

}

static void Add4(int x, float y)

{

Console.WriteLine("Result of sum is " + (x + y ));

}

static void Add5(string x, string y)

{

Console.WriteLine("Result of sum is " + (x + y));

}

static void Main()

{

Add1(10, 20);

}

}

-----

Solved by Method Overlaoding

using System;

class Prog

{

static void Add(int x, int y)

{

Console.WriteLine("Result of sum is " + ( x+y));

}

static void Add(int x, int y , int a)

{

Console.WriteLine("Result of sum is " + (x + y + a));

}

static void Add(float x, float y)

{

Console.WriteLine("Result of sum is " + (x + y ));

}

static void Add(int x, float y)

{

Console.WriteLine("Result of sum is " + (x + y ));

}

static void Add(float x, int y)

{

Console.WriteLine("Result of sum is " + (x + y));

}

//static int Add(float x, int y)

//{

// Console.WriteLine("Result of sum is " + (x + y));

//}

static void Add(string x, string y)

{

Console.WriteLine("Result of sum is " + (x + y));

}

static void Main()

{

Add(10, 20, 30);

Add("AAA", "BBB");

Add(10, 2.90.5);

}

}

How do a function return anything

By using return statement

using System;

class Prog

{

static int Add(int x, int y)

{

return x + y;

}

static int Subtract(int x, int y)

{

return x - y;

}

static int Product(int x, int y)

{

return x \* y;

}

static float Divide(int x, int y)

{

return (float)x / (float)y;

}

static void Add(int x, int y , int a)

{

Console.WriteLine("Result of sum is " + (x + y + a));

}

static void Add(float x, float y)

{

Console.WriteLine("Result of sum is " + (x + y ));

}

static void Add(int x, float y)

{

Console.WriteLine("Result of sum is " + (x + y ));

}

static void Add(float x, int y)

{

Console.WriteLine("Result of sum is " + (x + y));

}

//static int Add(float x, int y)

//{

// Console.WriteLine("Result of sum is " + (x + y));

//}

static void Add(string x, string y)

{

Console.WriteLine("Result of sum is " + (x + y));

}

static void Main()

{

int res= Add(10, 20);

Console.WriteLine("Result of sum is " + res);

res = Subtract(20, 10);

Console.WriteLine("Result of subtraction is " + res);

//Add(10, 20, 30);

//Add("AAA", "BBB");

//Add(10.9f, 2.90f);

}

}

Function can return how many values > Only 1 by using return statement

If we want to return more than one value from a method , **we use output parameters**

using System;

class Prog

{

static void DoAllOperations(int x, int y ,out int add,

out int subtract, out int product, out float div)

{

add = x + y;

subtract = x - y;

product = x \* y;

div = (float)x / (float)y;

}

static void Main()

{

int add, subtract, product;

float div;

DoAllOperations(20, 10, out add, out subtract, out product,

out div);

Console.WriteLine("Addition is " + add);

Console.WriteLine("Subtraction is " + subtract);

Console.WriteLine("Product is " + product);

Console.WriteLine("Divsion is " + div);

}

}