Comments are statements which are ignored by compiler

// Single Line Comment

/\*

Multiple Line Comments

\*/

The shortcut key for creating comments is

Ctrl K + C

The shortcut key for uncommenting statements is

Ctrl K + U

using System;

class Program

{

static void Main()

{

Console.WriteLine("Hello");

}

}

class Program // Minimum we need one class

{

static void Main() // Main is entry & exit point of th eprogrem

{

Console.WriteLine("Hello");

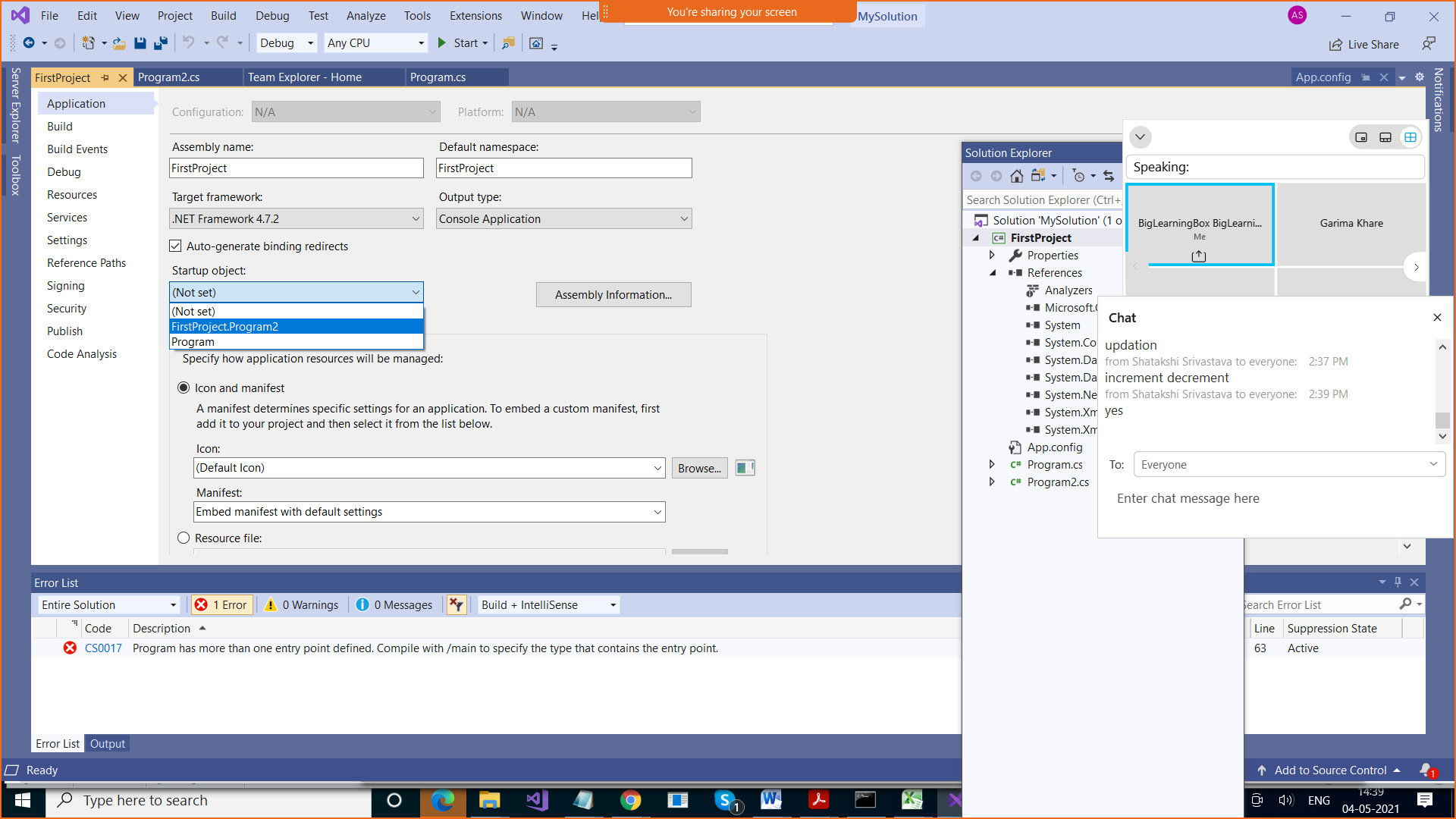
}

}

// Console is class , WriteLine is Method

Every Loop has 3 statements

1. Initialization
2. Condition / Termination Point
3. Increment / Decrement Statement



Print from 1 to 10

(do – while loop)

using System;

namespace FirstProject

{

class Program2

{

static void Main()

{

// Print numbers from 1 to 10

int n = 1;

do

{

Console.WriteLine(n);

n++;

} while (n <= 10);

}

}

}

(while loop)

using System;

namespace FirstProject

{

class Program2

{

static void Main()

{

// Print numbers from 1 to 10

Console.WriteLine("Do while loop");

int n = 100;

do

{

Console.WriteLine(n);

n++;

} while (n <= 10);

Console.WriteLine("While loop");

n = 100;

while(n<=10)

{

Console.WriteLine(n);

n++;

}

}

}

}

using System;

namespace FirstProject

{

class Program2

{

static void Main()

{

// Print numbers from 1 to 10

Console.WriteLine("Do while loop");

int n = 100;

do

{

Console.WriteLine(n);

n++;

} while (n <= 10);

Console.WriteLine("While loop");

n = 100;

while(n<=10)

{

Console.WriteLine(n);

n++;

}

Console.WriteLine("For loop");

n = 1;

for (; n <= 10;)

{

Console.WriteLine(n);

n++;

}

}

}

}

Break & Continue

Break > It is used to exit from a loop based on some condition

Break statement is used in

1. Loops
2. Switch

Continue > It is used to skip the remaining statements of the loop. It takes you to the next iteration/pass of the loop

Calculate sum of 10 numbers

using System;

namespace FirstProject

{

class Program3

{

static void Main()

{

SByte n=1, num;

int sum = 0;

while(n<=10)

{

Console.WriteLine("Enter number");

num = Convert.ToSByte(Console.ReadLine());

sum = sum + num;

n++;

}

Console.WriteLine("Sum is " + sum);

}

}

}

Calculate sum of 10 positive numbers

using System;

namespace FirstProject

{

class Program3

{

static void Main()

{

SByte n=1, num;

int sum = 0;

while(n<=10)

{

n++;

Console.WriteLine("Enter number");

num = Convert.ToSByte(Console.ReadLine());

if (num < 0) continue;

sum = sum + num;

}

Console.WriteLine("Sum is " + sum);

}

}

}

Calculate sum of 10 positive numbers & stop when user enters 0

using System;

namespace FirstProject

{

class Program3

{

static void Main()

{

SByte n=1, num;

int sum = 0;

while(n<=10)

{

n++;

Console.WriteLine("Enter number");

num = Convert.ToSByte(Console.ReadLine());

if (num < 0) continue;

if (num == 0) break;

sum = sum + num;

}

Console.WriteLine("Sum is " + sum);

}

}

}

Switch

using System; // namespace

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace FirstProject // project name

{

class Program

{

static void Main(string[] args)

{

// Add 2 numbers

double num1;

double num2;

int ch;

Console.WriteLine("Enter Value of num1");

num1 = Convert.ToDouble(Console.ReadLine());

Console.WriteLine(num1);

Console.WriteLine("Enter Value of num2");

num2 = Convert.ToDouble(Console.ReadLine());

Console.WriteLine("Enter Choice");

ch = Convert.ToByte(Console.ReadLine());

//Console.WriteLine("Sum of num1 & num2 is "

// + (num1+num2));

//// Console.WriteLine("Sum of %d & %d is %d "

// ,num1, num2 , (num1 + num2));

switch (ch)

{

case 1:

{

Console.WriteLine("Sum of {0} and {1} is {2}"

, num1, num2, (num1 + num2));

break;

}

case 2:

{

Console.WriteLine("Difference of {0} and {1} is {2}"

, num1, num2, (num1 - num2));

break;

}

case 3:

{

Console.WriteLine("Product of {0} and {1} is {2}"

, num1, num2, (num1 \* num2));

break;

}

case 4:

{

Console.WriteLine("Remainder of {0} and {1} is {2}"

, num1, num2, (num1 % num2));

break;

}

case 5:

{

Console.WriteLine("Quotient of {0} and {1} is {2}"

, num1, num2, (num1 / num2));

break;

}

default:

{

Console.WriteLine("Invalid choice");

break;

}

}

}

}

}

If(n > 2)

With switch , operator used is for equality

If(n> 2 && n<10)

Also we cannot use multiple conditions combined using && Operator

But we can use multiple conditions combined using || Operator

using System; // namespace

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace FirstProject // project name

{

class Program

{

static void Main(string[] args)

{

// Add 2 numbers

double num1;

double num2;

string ch;

Console.WriteLine("Enter Value of num1");

num1 = Convert.ToDouble(Console.ReadLine());

Console.WriteLine(num1);

Console.WriteLine("Enter Value of num2");

num2 = Convert.ToDouble(Console.ReadLine());

Console.WriteLine("Enter Choice");

ch = Console.ReadLine();

//Console.WriteLine("Sum of num1 & num2 is "

// + (num1+num2));

//// Console.WriteLine("Sum of %d & %d is %d "

// ,num1, num2 , (num1 + num2));

switch (ch)

{

case "1":

case "+":

{

Console.WriteLine("Sum of {0} and {1} is {2}"

, num1, num2, (num1 + num2));

break;

}

case "2":

case "-":

{

Console.WriteLine("Difference of {0} and {1} is {2}"

, num1, num2, (num1 - num2));

break;

}

case "3":

case "\*":

{

Console.WriteLine("Product of {0} and {1} is {2}"

, num1, num2, (num1 \* num2));

break;

}

case "4":

case "%":

{

Console.WriteLine("Remainder of {0} and {1} is {2}"

, num1, num2, (num1 % num2));

break;

}

case "5":

case "/":

{

Console.WriteLine("Quotient of {0} and {1} is {2}"

, num1, num2, (num1 / num2));

break;

}

default:

{

Console.WriteLine("Invalid choice");

break;

}

}

}

}

}

Functions :

Method Overloading

Named Parameter

Optional Parameters

Call By value

Call By reference

Params array (Not covered)

Why we need functions : For Modularity

Breaking a big program into smaller programs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace FirstProject

{

class Program4

{

static void Main()

{

int x, y;

Console.WriteLine("Enter Value of x");

x = Convert.ToInt16(Console.ReadLine());

Console.WriteLine("Enter Value of y");

y = Convert.ToInt16(Console.ReadLine());

AddNumbers(x, y);

SubtractNumbers(x, y);

MultiplyNumbers(x, y);

}

static void AddNumbers(int x, int y)

{

Console.WriteLine("Sum is " + (x+y));

}

static void SubtractNumbers(int x, int y)

{

Console.WriteLine("Difference is " + (x - y));

}

static void MultiplyNumbers(int x, int y)

{

Console.WriteLine("Product is " + (x \* y));

}

}

}

How do we define a function

Access\_specifier return\_type function\_name(parameters)

{

Statements;

//return

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace FirstProject

{

class Program4

{

static void Main()

{

int x, y;

Console.WriteLine("Enter Value of x");

x = Convert.ToInt16(Console.ReadLine());

Console.WriteLine("Enter Value of y");

y = Convert.ToInt16(Console.ReadLine());

int res;

res= AddNumbers(x, y);

Console.WriteLine("Sum is " + res);

Console.WriteLine("Difference is " + SubtractNumbers(x, y));

Console.WriteLine("Product is " + MultiplyNumbers(x, y));

}

static int AddNumbers(int x, int y)

{

return x + y;

}

static int SubtractNumbers(int x, int y)

{

int res = x-y;

return res;

}

static int MultiplyNumbers(int x, int y)

{

return(x \* y);

}

}

}

Function

When we call a function, we may or may not pass parameters, but when we parameters,

By default they are passed as value

When para are passed as value, the called function declares a local variable there

Call By Value : When the parameter is passed as value to the called method (default behaviour)

static void Main()

{

int x = 100;

int y = 200;

Console.WriteLine("Value of x and y inside Main function are {0} and {1} ", x, y);

**Change1(x, y);**

Console.WriteLine("Value of x and y after calling Change1 function are {0} and {1} ", x, y);

}

static void Change1(int x, int y)

{

// It creates local variables here

x = 1000;

y = 2000;

Console.WriteLine("Value of x and y inside Change1 function are {0} and {1} ", x, y);

}

Call By Reference : When we pass reference / address of the variable to the called method

static void Main()

{

int x = 100;

int y = 200;

Console.WriteLine("Value of x and y inside Main function are {0} and {1} ", x, y);

**Change2(ref x, ref y);**

Console.WriteLine("Value of x and y after calling Change2 function are {0} and {1} ", x, y);

}

static void Change2(ref int x, ref int y)

{

// It creates local variables here

x = 3000;

y = 4000;

Console.WriteLine("Value of x and y inside Change2 function are {0} and {1} ", x, y);

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace FirstProject

{

class Program5

{

static void Main()

{

int x = 100;

int y = 200;

Console.WriteLine("Value of x and y inside Main function are {0} and {1} ", x, y);

Change1(x, y);

Console.WriteLine("Value of x and y after calling Change1 function are {0} and {1} ", x, y);

Change2(ref x, ref y);

Console.WriteLine("Value of x and y after calling Change2 function are {0} and {1} ", x, y);

}

static void Change1(int x, int y)

{

// It creates local variables here

x = 1000;

y = 2000;

Console.WriteLine("Value of x and y inside Change1 function are {0} and {1} ", x, y);

}

static void Change2(ref int x, ref int y)

{

// It creates local variables here

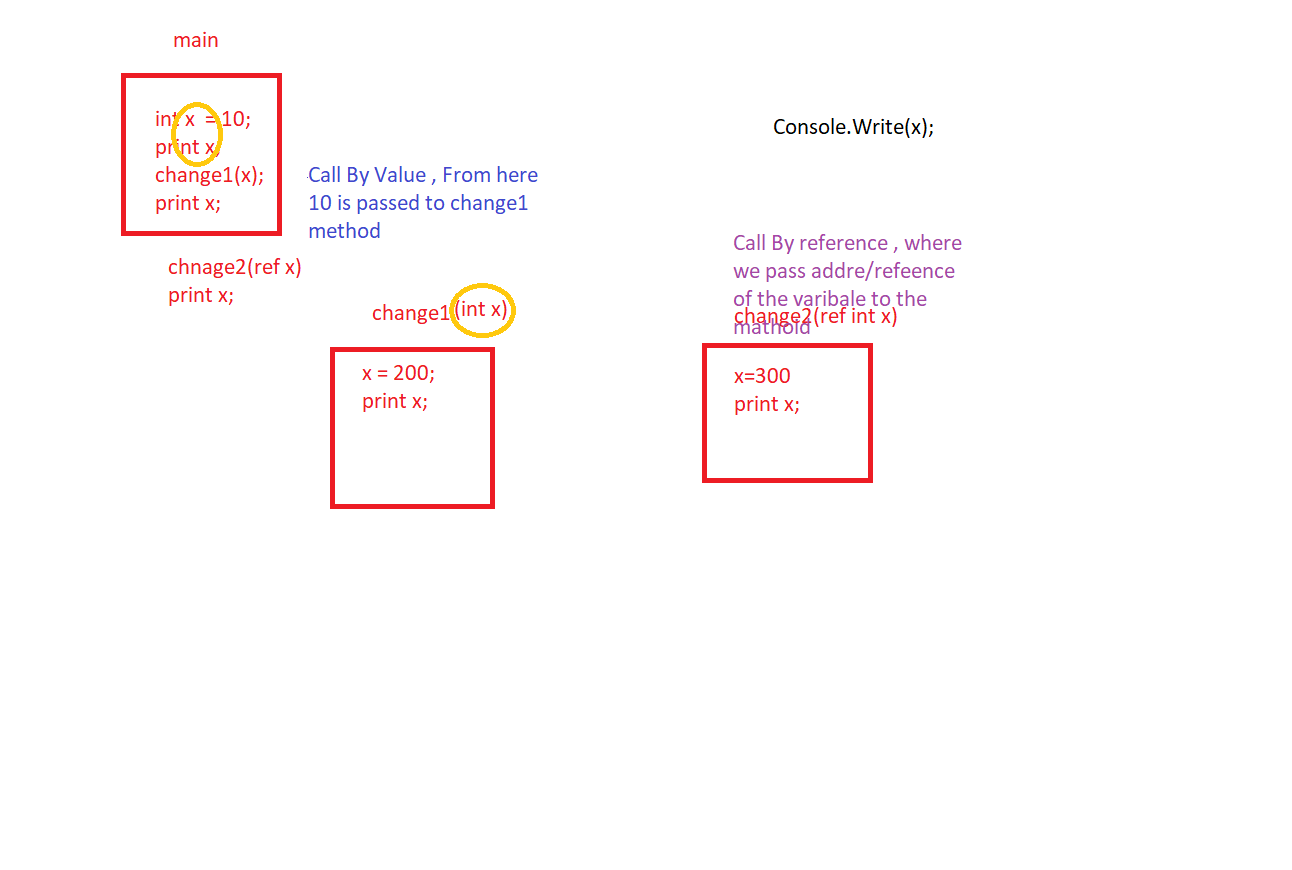
x = 3000;

y = 4000;

Console.WriteLine("Value of x and y inside Change2 function are {0} and {1} ", x, y);

}

}

}

Call by Value : We pass value of the variable, so any changes made there are not reflected in the calling method

Call by Reference : We pass address of the variable, so any changes made there are reflected in the calling method

Method Overloading : Same method but different number & type of parameters, there is no need to remember so many method names

Method Overloading : Function names are kept same but their signature is different , number and type of parameters will differ ( Also known as Polymorphism) (One Name different forms)

Function return type is not considered in this.

Advantage : No need to remember different function names while calling them

Without method overloading

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace FirstProject

{

class Program6

{

static void add(int x, int y)

{

Console.WriteLine(x+y);

}

static int add2(int x, int y, int z)

{

Console.WriteLine(x + y + z);

}

static void add3(float x, float y)

{

Console.WriteLine(x + y);

}

static void add4(int x, int y, float z)

{

Console.WriteLine(x + y + z);

}

static void Main()

{

add(10, 20);

add2(10, 20, 40);

add3(10.9f, 90.8f);

add4(10, 90, 90.5f);

}

}

}

With Method overloading

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace FirstProject

{

class Program6

{

static void add(int x, int y)

{

Console.WriteLine(x+y);

}

static void add(int x, int y, int z)

{

Console.WriteLine(x + y);

}

static void add(float x, float y)

{

Console.WriteLine(x + y);

}

static void add(int x, int y, float z)

{

Console.WriteLine(x + y + z);

}

static void Main()

{

add(10, 20);

add(10, 20, 40);

add(10.9f, 90.8f);

add(10, 90, 90.5f);

}

}

}

Named Parameters : While calling a method , when we pass parameter’ values , we pass it alongwith parameter name

Advantage : We can change sequence

It makes program easy to understand

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace FirstProject

{

class Program7

{

static void DisplayDetails(int rn , string name,

string address, int marks)

{

Console.WriteLine("Roll No is " + rn);

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

Console.WriteLine("Address is " + address);

Console.WriteLine("Marks are " + marks);

}

static void Main()

{

DisplayDetails(1, "Delhi", "Deepak", 90);

// Named Paramters

DisplayDetails(rn: 1, marks: 90, address: "Delhi",

name: "Deepak");

}

}

}

Optional Parameters : While defining method, we give parameters some default values , so If we don’t provide value to those parameters, they will take that default value

We have to provide values from RIGHT TO LEFT

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace FirstProject

{

class Program7

{

static void DisplayDetails(int rn=9 ,

string name="XXXXXX"

,string address="Delhi", int marks=70)

{

Console.WriteLine("Roll No is " + rn);

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

Console.WriteLine("Address is " + address);

Console.WriteLine("Marks are " + marks);

}

static void Main()

{

// DisplayDetails(1, "Delhi", "Deepak", 90);

// Named Paramters

DisplayDetails(rn: 1,address: "Delhi",

name: "Deepak");

DisplayDetails(1, "Delhi", "Deepak", 80);

DisplayDetails();

}

}

}

Output Parameters

How many values we can return from a function by using return statement? **1**

But if we want to return more than one value , for that we can use output parameters

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace FirstProject

{

class Program8

{

static void Operations(int x, int y,

out int add , out int subtract,

out int product, out int div)

{

add = x + y;

subtract = x - y;

product = x \* y;

div = x / y;

}

static void Main()

{

int add, subtract, product, div;

Operations(20, 10, out add, out subtract,

out product, out div);

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

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

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

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

}

}

}