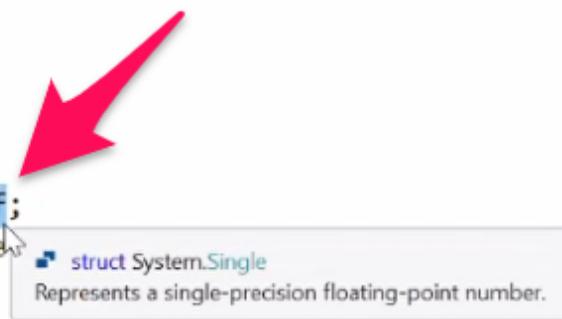


C#-3

```
5  using System.Threading.Tasks;
6
7  namespace ConsoleApp
8  {
9      class IntTypeDemo
10     {
11         static void Main()
12         {
13             float f = 100.98f;
14             Console.WriteLine("A = " + f);
15
16             /*
17             int a; //decl
18             a = 10; //init
19             Console.WriteLine("A = " + a);
20             a = 'C';
21             Console.WriteLine("A = " + (char)a);
22             */
23         }
24     }
}
```

this is called literals



but if we write double in place of float then there will be no error



```
namespace ConsoleApp
{
    0 references
    class IntTypeDemo
    {
        0 references
        static void Main()
        {
            //float f = 100.98f;
            double f = 100.98;
            Console.WriteLine("F = " + f);

            /*
            int a; //decl
            a = 10; //init
            Console.WriteLine("A = " + a);
            a = 'C';
            Console.WriteLine("A = " + (char)a);
            */
        }
    }
}
```

No issues found

because by default all your floating value will be treated as double

**NameSpace, Class Name, MethodName,
Interface should follow PascalCasing**

almost every thing follow PascalCasing apart from local Variable Name

localVariableName should follow camelCasing

C# Keywords

- Keywords are reserved identifiers

```
abstract, base, bool, default, if, finally
```

- Do not use keywords as variable names

- Results in a compile-time error

- Avoid using keywords by changing their case sensitivity

```
int INT; // Poor style
```

```
5  [using System.Threading.Tasks;
6
7  namespace ConsoleApp
8  {
9      0 references
10     class IntTypeDemo
11     {
12         0 references
13         static void Main()
14         {
15             string firstName = "Shashi"; // here f is small
16             string FirstName = "Kanth"; //here F is capital
```

so my C# will not complaint here because these are 2 different variable even tho they both have the first name but in first case f is small and in 2nd F is capital , so this is case sensitive

Declaring Local Variables

- Usually declared by data type and variable name:

```
int itemCount;
```

- Possible to declare multiple variables in one declaration:

```
int itemCount, employeeNumber;
```

--OR--

```
int itemCount,  
employeeNumber;
```

Assigning Values to Variables

- Assign values to variables that are already

```
int employeeNumber;
```

```
employeeNumber = 23;
```

```
int employeeNumber = 23;
```

- Initialize a variable when you declare it:

```
char middleInitial = 'J';
```

- You can also initialize character values:

Common Operators

Common Operators	Example
• Equality operators	<code>== !=</code>
• Relational operators	<code>< > <= >= is</code>
• Conditional operators	<code>&& ?:</code>
• Increment operator	<code>++</code>
• Decrement operator	<code>--</code>
• Arithmetic operators	<code>+ - * / %</code>
• Assignment operators	<code>= *= /= %= += -= <<= >>= &= ^= =</code>

Increment and Decrement

- Changing a value by one is very common

```
itemCount += 1;  
itemCount -= 1;
```

-
- ```
itemCount++;
itemCount--;
```

```
++itemCount;
--itemCount;
```

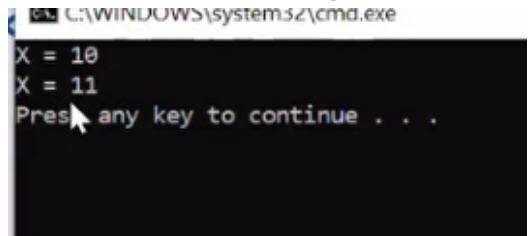
- This shorthand exists in two forms

You should be little carefull about incrementing and decrementing

```
namespace ConsoleApp
{
 0 references
 class IncDemo
 {
 0 references
 static void Main()
 {
 int x = 10;
 Console.WriteLine("X = " + x++);
 Console.WriteLine("X = " + x);
 }
 }
}
```

I

What is the output of this code



```
X = 10
X = 11
Press any key to continue . . .
```

because here `++` is a post increment so after printing it will increment

```
2 using System.Collections.Generic;
3 using System.Linq;
4 using System.Text;
5 using System.Threading.Tasks;
6
7 namespace ConsoleApp
8 {
9 class IncDemo
10 {
11 static void Main()
12 {
13 int x = 10;
14 Console.WriteLine("X = " + ++x);
15 Console.WriteLine("X = " + x); I
16 }
17 }
18 }
19
```

it will print me 11 11 because it is a pre increment

```
3 using System.Linq;
4 using System.Text;
5 using System.Threading.Tasks;
6
7 namespace ConsoleApp
8 {
9 0 references
10 class IncDemo
11 {
12 0 references
13 static void Main()
14 {
15 int y = 10;
16 int x = 10;
17 y = y++ + ++x;
18
19 }
20 }
21
22 }
23
24 No issues found
25
```

Package Manager Console Error List Immediate Window

here what will be the output

```
emc C:\WINDOWS\system32\cmd.exe
ons
X = 11
Y = 21
Press any key to continue . . .
```



## Operator Precedence

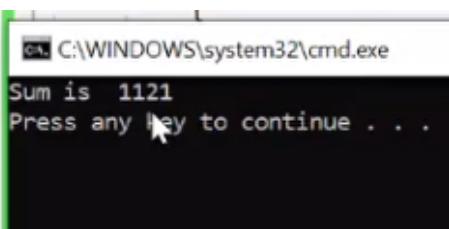
- Operator Precedence and Associativity
    - Except for assignment operators, all binary operators are left-associative
    - Assignment operators and conditional operators are right-associative
- 

```
int y = 10;
int x = 10;
y = y++ + ++x;

//Console.WriteLine("X = " + x);
//Console.WriteLine("Y = " + y);

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

here it will concatenate it so output is



but i want to say baba add this baba don't concatenate do i'll put them in brackets

X ConsoleApp

ConsoleApp.IncDemo Main()

```
int y = 10;
int x = 10;
y = y++ + ++x;

//Console.WriteLine("X = " + x);
//Console.WriteLine("Y = " + y);

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

 }

}
```

X File Edit View Git Project

C:\WINDOWS\system32\cmd.exe

```
Sum is 32
Press any key to continue . . .
```

```
ConsoleApp.IncDemo Main()
```

```
{
 int y = 5; [
 int x = 2;
 //y = y++ + ++x;

 int res = x + y * 8 / 2 - 4;

 //Console.WriteLine("X = " + x);
 //Console.WriteLine("Y = " + y);

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

now what is the output  
so it will always give preferences



## Converting Data Types

- Implicit Data Type Conversion
- Explicit Data Type Conversion

Remember when ever you talk about conversion it should be of the same type , you cannot convert from one type to other type

## Implicit Data Type Conversion

- Converting lower range value into higher range value
- Example:

```
using System;
class Test
{
 static void Main()
 {
 int intValue = 123;
 long longValue = intValue;
 Console.WriteLine("(long) {0} = {1}", intValue, longValue);
 }
}
```

- Implicit conversions cannot fail
  - May lose precision, but not magnitude

i can dump every thing from 1bhk to 2 bhk but i can't dump every thing from 2bhk to 1bhk

```
namespace ConsoleApp
{
 0 references
 class ConvertType
 {
 0 references
 static void Main()
 {
 int a = 10; //1BHK
 float f = a; //2BHK
 Console.WriteLine("A = " + a + " F = " + f);
 Console.WriteLine("A = {0} F = {1}", a, f);
 }
 }
}
```

either you can use this syntax  
or you can write like this

```
[using System.Threading.Tasks;

namespace ConsoleApp
{
 class ConvertType
 {
 static void Main()
 {
 int a = 10; //1BHK
 float f = a; //2BHK
 double d = f; //3BHK
 //Console.WriteLine("A = " + a + " F = " + f);
 Console.WriteLine("A = {0} F = {1} D = {2}", a, f, d);
 }
 }
}
```

No issues found

i can pass float value to double

**this is called implicit conversion** i am going from lower to higher where the conversions are there but it is converting from integer to float type but it is implicit

now i am taking 3BHK content to 2 BHK can we do it directly ----- **No**

So, i need to filter i need to throw some of the unwanted things

```
namespace ConsoleApp
{
 0 references
 class ConvertType
 {
 0 references
 static void Main()
 {
 /*
 int a = 10; //1BHK
 float f = a; //2BHK
 double d = f; //3BHK
 */
 double d = 1000.233;
 float f = (float)d; here i am throwing some of the unwanted things

 //Console.WriteLine("A = " + a + " F = " + f);
 Console.WriteLine("A = {0} F = {1} D = {2}", a, f, d);
 }
 }
}
```

Now because of some reason i have to move from 2BHK to 1 BHK now i cannot directly move from 2bhk to 1bhk so i need to filter it

## this is called Explicit conversion

```
1 SOLUTION_Console
Select C:\WINDOWS\system32\cmd.exe
A = 1000 F = 1000.233 D = 1000.233
Press any key to continue . . .
```

in case of integer it is eliminating the complete decimal value

## Explicit Data Type Conversion

- Converting higher range value into lower range value
- Either, you have to use Cast operator to do that job
- Or, you have to methods of Convert class to do that job
- Example:

```
using System;
class Test
{
 static void Main()
 {
 long longValue = 1234567890;
 int intValue = (int) longValue;
 Console.WriteLine("(int) {0} = {1}", longValue, intValue);
 }
}
```

this is with the same type, in the same type we are using conversion this are called **Type Casting**

Now we are discussing **Parsing**

in parsing we are converting the whole type from one type to another type

## Using Convert Class

- Convert class contains methods to convert any data type into target data type
- Can be used for both Implicit and Explicit conversion
- Belongs to System namespace
- Example:

```
static void Main()
{
 Console.Write("Enter Your Identification Number: ");
 string strId = Console.ReadLine();
 int intId = convert.ToInt32(strId);
}
```

- User is asked to enter his/her id
- `ReadLine()` method returns id in string format
- `ToInt32` method of Convert class is being used to convert that string representation into integer representation

`ToInt32()` method converts any data type into '`int`' data type.

Similarly, all other methods like, `ToDecimal()`, `ToDouble()` etc are used to convert any data type into '`decimal`', '`double`' data type respectively

```
class Converttype
{
 0 references
 static void Main()
 {
 string no = "20";
 /*
 int a = 10; //1BHK
 float f = a; //2BHK
 double d = f; //3BHK
 */
 double d = 10002342333242342.23234234234;
 float f = (float)d;
 int a = (int)f;
 int res = a + (int)no; here this will not work because we are converting the whole type
 //Console.WriteLine("A = " + a + " F = " + f);
 Console.WriteLine("A = {0} F = {1} D = {2}", a, f, d);
 }
}
```

So if you want to do something like this we'll be using parsing here

```
{
 0 references
 static void Main()
{
 string no = "20";
 /*
 int a = 10; //1BHK
 float f = a; //2BHK
 double d = f; //3BHK
 */
 double d = 10002342333242342.23234234234;
 float f = (float)d;
 int a = (int)f;
 //int res = a + int.Parse(no); either you can use int.Parse
 int res = a + Convert.ToInt32(no); or you can write Convert.ToInt32()
 //Console.WriteLine("A = " + a + " F = " + f);
 Console.WriteLine("A = {0} F = {1} D = {2}", a, f, d);
 }
}
```

**Now you need to find the difference between parse and convert.to.....**

Whatever we take from Console.WriteLine() it always takes input as a string

```
7 namespace ConsoleApp
8 {
9 class ReadNumbers
10 {
11 static void Main()
12 {
13 int no1, no2;
14 Console.WriteLine("Enter no1:");
15 no1 = Convert.ToInt32(Console.ReadLine());
16 Console.WriteLine("Enter no2:");
17 no2 = Convert.ToInt32(Console.ReadLine());
18 Console.WriteLine("Sum of numbers is {0}", (no1+no2));
19 }
20 }
21 }
22 }
```

```
pleApp ConsoleApp.ReadNumbers Main()
```

```
7 -> namespace ConsoleApp
8 {
9 -> class ReadNumbers
10 {
11 -> static void Main()
12 {
13 int no1, no2;
14 Console.WriteLine("Enter no1:");
15 no1 = Convert.ToInt32(Console.ReadLine());
16 Console.WriteLine("Enter no2:");
17 no2 = Convert.ToInt32(Console.ReadLine());
18 Console.WriteLine("Sum of numbers is {0}" , (no1+no2));
19 //10 + 20 = 30 to print in this formate we write like this
20 int res = no1 + no2;
21 Console.WriteLine("{0} + {1} = {2}", no1, no2, res);
22 }
23 }
24 }
```

No issues found | Error List | Immediate Window



## Namespaces

- To provide conceptual grouping of your classes, we can provide namespaces.
- Similar to namespaces of C++.
- They can span across multiple files.
- Typically we use namespaces
  - To group our classes as per our logical groupings
  - To avoid naming conflicts.
- Units defined under namespaces are to be referred by namespace.UnitName.
- Example:
  - *System*;
  - *System.Collections*;
  - *System.IO*;

## Statements in C#

| Category                      | C# keywords                                      |
|-------------------------------|--------------------------------------------------|
| Selection statements          | if, else, switch, case                           |
| Iteration statements          | do, for, foreach, in, while                      |
| Jump statements               | break, continue, default, goto, return, yield    |
| Exception handling statements | throw, try-catch, try-finally, try-catch-finally |
| Checked and unchecked         | checked, unchecked                               |
| fixed Statement               | fixed                                            |
| lock Statement                | lock                                             |

## If..Else Statement

- Syntax:

```
if (Boolean-expression)
 first-embedded-statement
else
 second-embedded-statement
```

- No implicit conversion from int to bool

```
int x;
...
if (x) ... // Must be if (x != 0) in C#
if (x = 0) ... // Must be if (x == 0) in C#
```

```

if(cond) ← if this condition is true
{
 stat; it will execute this statement
}
 and comes out and do nothing
```

## 2. If...Else

```

if(cond) ← if this condition is true it will execute
{
 ↑
 stat1; this statement and never execute else
}
else
{
 stat2;
}
 but if this condition is false then it
 will execute this statement inside
 else
```

### 3. if...elseif...else

```
if(cond1) ← if this condition is true it executes
{
 stat1; this statements and never executes this elseif and else statements
}
else if(cond2)
{
 stat2;
}
else but if this 2 conditions are false then the condition inside else will be executed
{
 statn;
}
```

but if condition1 is false then it will check for condition2 if this condition is true  
then it execute this statement and never execute statement inside else

#### 4. Nested if

```

if(cond1) ← when ever you are using nested if
{ then here cond1 should have association with
 if(cond1.1)
 {
 stat1.1;
 }
 else
 {
 stat1.2;
 }
}
else
{
 statn;
}
```

Enter login id:

Enter password:

When user clicks on this login if Loginid or password is blank i should get a message

Enter login id:

Login id cannot be blank

Enter password:

if user enter something in login id but pass is blank so i should get err msg like this

Enter login id:

Enter password:

Password cannot be blank

So to achieve this which if should i use  
senario like this dosen't need nested if  
So, my simplest solution is

```
if(login == '')
{
 print("Login cannot be blank");
}
if(password == '')
{
 print("Password cannot be blank");
}
```

I Have a 3rd scenario we have a banking application which you are developing in that we have a loan section and the form is been pass to the manager, and me being a manager i have to approve a loan so

- 1. the person should have acc open in our bank if not we'll say first open the acc then come back then we'll process it

but if the person already has the acc then we will check for other document

So, How do you control this

**Here we use nested if because the 2nd condition is dependent on 1st condition**

#### 4. Nested if

```

if(cond1) ← if he has acc then i will check
{
 if(cond1.1) ← weather he is eligible or not
 {
 stat1.1;
 }
 else
 {
 stat1.2;
 }
}
else ← if he dont have the account
{
 statn; i'll tell him go get the acc opened
}
```

biggest of 2 number using if else

The screenshot shows the Microsoft Visual Studio IDE interface. The title bar displays three tabs: 'IfDemo1.cs', 'ConsoleApp\*', and 'ConsoleApp'. The current file is 'IfDemo1.cs', which contains the following C# code:

```
4 using System.Text;
5 using System.Threading.Tasks;
6
7 namespace ConsoleApp
8 {
9 class IfDemo1
10 {
11 static void Main()
12 {
13 int no1 = 10, no2 = 20;
14 if(no1 > no2)
15 {
16 Console.WriteLine("{0} is bigger", no1);
17 }
18 else
19 {
20 Console.WriteLine("{0} is bigger", no2);
21 }
22 }
23 }
}
```

The code uses the 'Console.WriteLine' method to output the result of the comparison. A green vertical bar on the left margin indicates the current line of code being executed or selected.

we always have to use different methods for input process and output

The screenshot shows the Microsoft Visual Studio IDE interface. The title bar displays "ConsoleApp.IfDemo1". The main window shows the following C# code:

```
{
 0 references
 class IfDemo1
 {
 0 references
 static void Main()
 {
 int no1 = 10, no2 = 20;
 FindBiggest(no1, no2);
 }

 1 reference
 private static void FindBiggest(int no1, int no2)
 {
 if (no1 > no2)
 {
 Console.WriteLine("{0} is bigger", no1);
 }
 else
 {
 Console.WriteLine("{0} is bigger", no2);
 }
 }
 }
}
```

The status bar at the bottom left indicates "No issues found".

now can we optimize this code can we do this without else

```
1 reference
private static void FindBiggest(int no1, int no2)
{
 int big = no2;
 if (no1 > no2)
 {
 big = no1;
 }
 Console.WriteLine("{0} is bigger", big);
}
```

performance wise it is better but you can see we are taking a additional variable here  
Remember if is always a costly process so try don't to use if else as much as possible it is always a dangerous process

Now if it is just a matter of finding the biggest so you can always use a ternary operator

```
1 reference
private static void FindBiggest(int no1, int no2)
{
 int big = (no1 > no2 ? no1 : no2);
 //int big = no2;
 //if (no1 > no2)
 //{
 // big = no1;
 //}
 Console.WriteLine("{0} is bigger", big);
}
```



## Assignment :-Find Biggest of 3 Number using ternary operator

## Cascading if Statements

```
Console.WriteLine("Enter Color Choice: ");
string choice = Console.ReadLine();
if (choice == "Black")
 Console.WriteLine("Choice is Black..");
else if (choice == "Blue")
 Console.WriteLine("Choice is Blue..");
else if (choice == "Red")
 Console.WriteLine("Choice is Red..");
else
 Console.WriteLine("No Color choice..");
```

what will be the output ? will it print Hi or Hello

```
IfDemo2.cs* X ConsoleApp*
ConsoleApp
ConsoleApp.IfDemo2
3 using System.Linq;
4 using System.Text;
5 using System.Threading.Tasks;
6
7 namespace ConsoleApp
8 {
9 class IfDemo2
10 {
11 static void Main()
12 {
13 int x = 0;
14 if (x)
15 {
16 Console.WriteLine("Hi");
17 }
18 else
19 {
20 Console.WriteLine("Hello");
21 }
22 }
}
```

there is a compilation error

The screenshot shows the Microsoft Visual Studio IDE interface. The title bar displays "ConsoleApp.IfDemo2". The code editor contains the following C# code:

```
0 references
class IfDemo2
{
 0 references
 static void Main()
 {
 int x = 0;
 if (x)
 {
 // (local variable) int x
 }
 else
 {
 Console.WriteLine("Hello");
 }
 }
}
```

A tooltip is displayed over the opening brace of the if block, indicating a compilation error:

CS0029: Cannot implicitly convert type 'int' to 'bool'

I **C# or in java you cannot do this** because if will always check for boolean value **in C or C++ this will work**

```
ConsoleApp.IfDemo2
0 references
class IfDemo2
{
 0 references
 static void Main()
 {
 int x = 0;
 if (true) ← i can say if true this is possible
 {
 Console.WriteLine("Hi");
 }
 else
 {
 Console.WriteLine("Hello");
 }
 }
}
```

### ConsoleApp.IfDemo2

```
0 references
class IfDemo2
{
 0 references
 static void Main()
 {
 int x = 0;
 if (!true)  i can say if !true this is possible
 {
 Console.WriteLine("Hi");
 }
 else
 {
 Console.WriteLine("Hello");
 }
 }
}
```

i can say if false that is also possible but i cannot say if 0 or 1 that is not possible

```
ConsoleApp.IfDemo2
0 references
class IfDemo2
{
 0 references
 static void Main()
 {
 bool isTrue = true;
 if (isTrue)
 {
 Console.WriteLine("Hi");
 }
 else
 {
 Console.WriteLine("Hello");
 }
 }
}
```

we can write like this

```
0 references
class IfDemo2
{
 0 references
 static void Main()
 {
 bool isTrue = true;
 if (isTrue == true)
 {
 Console.WriteLine("Hi");
 }
 else
 {
 Console.WriteLine("Hello");
 }
 }
}
```

we can also write like this  
but internally my compiler treat this as 1st one,  
so we don't write like this  
we write like the 1st one

```
0 references
class IfDemo2
{
 0 references
 static void Main()
 {
 bool isTrue = false;
 if (!isTrue)
 {
 Console.WriteLine("Hi");
 }
 else
 {
 Console.WriteLine("Hello");
 }
 }
}
```

Now what will be the output -----Hi

0 references

```
class IfDemo2
```

```
{
```

0 references

```
 static void Main()
```

```
{
```

```
 bool isTrue = false;
```

```
 if (!isTrue)
```

```
{
```

```
 Console.WriteLine("Hi");
```

```
 isTrue = !true;
```

```
}
```

```
 else
```

```
{
```

```
 Console.WriteLine("Hello");
```

```
}
```

```
 Console.WriteLine("Value of isTrue is {0}", isTrue);
```

```
}
```

```
}
```

I

```
0 references
class IfDemo2
{
 0 references
 static void Main()
 {
 bool isTrue = true;
 if (isTrue)
 {
 Console.WriteLine("Hi");
 }
 else
 {
 Console.WriteLine("Hello");
 }
 isTrue = !isTrue;
 Console.WriteLine("Value of isTrue is {0}", isTrue);
 }
}
```

Write program for this if day == sunday then color = white , if day == mon then color =yellow

.....

IfDemo3.cs X ConsoleApp\*

ConsoleApp

```
9 class IfDemo3
10 {
11 static void Main()
12 {
13 string dow = "Sun";
14 //sun = white
15 //mon = yellow
16 //tue = red
17 //wed = orange
18 //thur = pink
19 //fri = black
20 //sat = ur wish
21 }
22 }
23 }
```

```
string dow = "Sun";
//sun = white
//mon = yellow
//tue = red
//wed = orange
//thur = pink
//fri = black
//sat = ur wish
if(dow == "Sun")
{
 Console.WriteLine("White");
}
else if(dow == "Mon")
{
 Console.WriteLine("Yellow");
}
else if(dow == "Tue")
{
 Console.WriteLine("Red");
}
}
```

WE can keep writing like this elseif elseif .....  
.....

**But when you have the same variable but with different value**

it is the same you but you behave differently

when you sitting in the class u act like a student  
when you being with your mom u act like a Son

```
string dow = "Sun";
//sun = white
//mon = yellow
//tue = red
//wed = orange
//thur = pink
//fri = black
//sat = ur wish
if(dow == "Sun")
{
 Console.WriteLine("White");
}
else if(dow == "Mon")
{
 Console.WriteLine("Yellow");
}
else if(dow == "Tue")
{
 Console.WriteLine("Red");
}
```

when it is equal to  
when you have a variable and you checking for  
equality

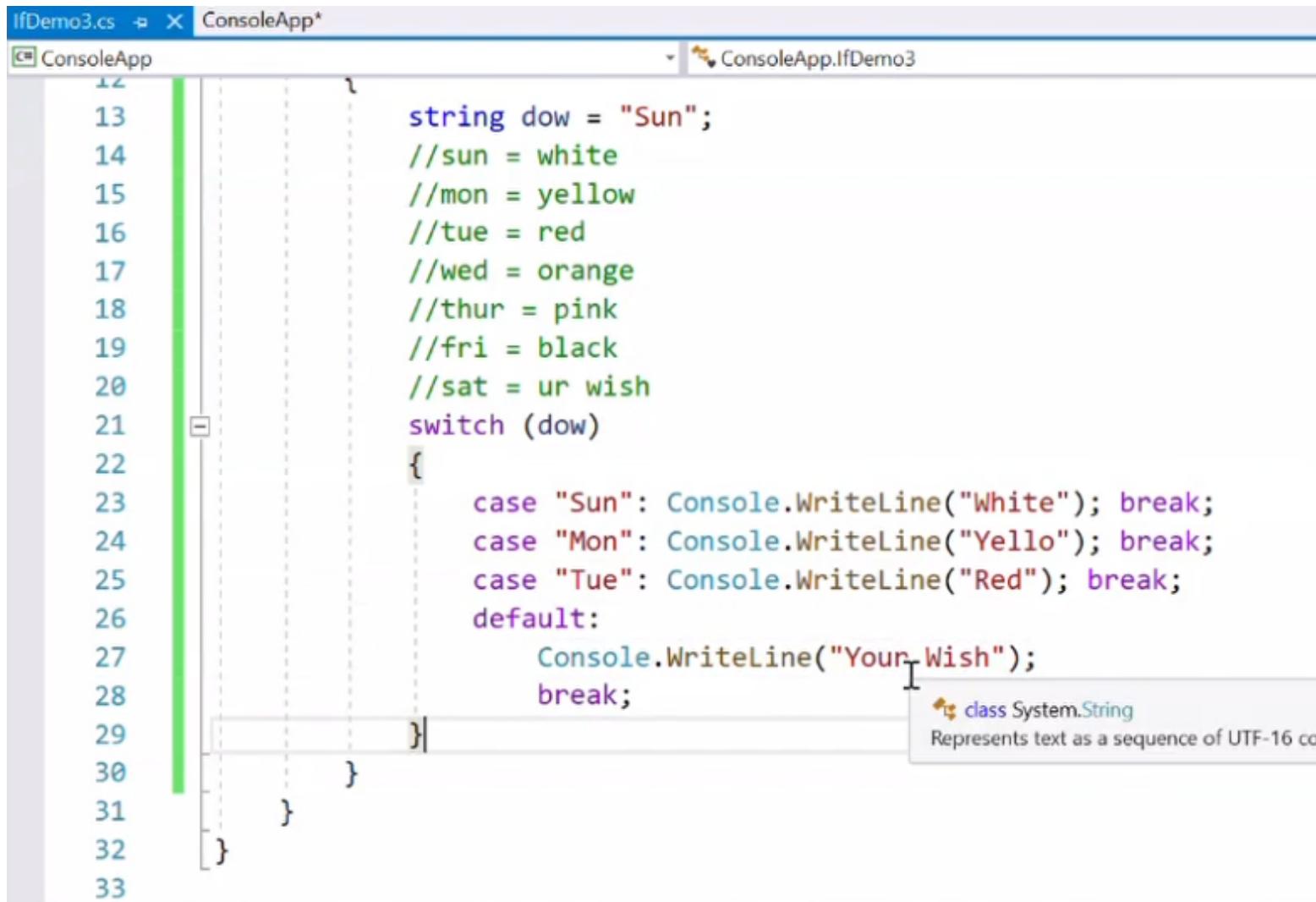
SO you should NOT use if

It is always a Performance head  
it will take a lot of time  
to render /Process

**So in such senario You should  
always use Switch**

**So, Switch is used when ever you have the same variable  
but may have different value in this case you can always go**

# with Switch



The screenshot shows a Microsoft Visual Studio interface with the title bar "ConsoleApp\*". The code editor displays a C# file named "IfDemo3.cs" under the project "ConsoleApp.IfDemo3". The code implements a switch statement based on the day of the week:

```
12
13 string dow = "Sun";
14 //sun = white
15 //mon = yellow
16 //tue = red
17 //wed = orange
18 //thur = pink
19 //fri = black
20 //sat = ur wish
21 switch (dow)
22 {
23 case "Sun": Console.WriteLine("White"); break;
24 case "Mon": Console.WriteLine("Yello"); break;
25 case "Tue": Console.WriteLine("Red"); break;
26 default:
27 Console.WriteLine("Your Wish");
28 break;
29 }
30 }
31 }
32 }
33 }
```

A tooltip for the word "String" is visible at the bottom right, indicating it is a class from the System namespace.

Can we do like this -----yes -----switch will support this

A screenshot of the Visual Studio IDE showing a C# code editor. The code is a simple console application named 'SwitchDemo'.

```
1 using System;
2 using System.Collections.Generic;
3 using System.Linq;
4 using System.Text;
5 using System.Threading.Tasks;
6
7 namespace ConsoleApp
8 {
9 class SwitchDemo
10 {
11 static void Main()
12 {
13 int no = 5;
14 switch (no)
15 {
16 case 1: Console.WriteLine("Its 1"); break;
17 case 1 + 1: Console.WriteLine("Its 2"); break;
18 case 2 + 2: Console.WriteLine("Its 4"); break;
19 default:
20 break;
21 }
22 }
23 }
```

The code uses a switch statement with integer cases. A yellow vertical bar highlights the opening brace of the Main() method. The status bar at the bottom shows "133 %", "No issues found", and a checkmark icon.

what switch will not support

The screenshot shows a Visual Studio code editor with the following C# code:

```
SwitchDemo.cs* X IfDemo3.cs ConsoleApp*
ConsoleApp
7 namespace ConsoleApp
8 {
9 class SwitchDemo
10 {
11 static void Main()
12 {
13 int no = 5;
14 switch (no)
15 {
16 case 1: Console.WriteLine("Its 1"); break;
17 case 1 + 1: Console.WriteLine("Its 2"); break;
18 case 2 + 2: Console.WriteLine("Its 4"); break;
19 case 4: Console.WriteLine("Its 4"); break;
20 }
21 }
22 }
23 }
24 }
25 }
26 }
```

A red arrow points from the text "so 4 is already there 2+2=4" to the line "case 4: Console.WriteLine("Its 4"); break;". A tooltip is displayed over the word "case" in the fourth case label, containing the following information:

- case 4: Console.WriteLine("Its 4"); break;
- d [ ]
- struct System.Int32
- Represents a 32-bit signed integer.

At the bottom of the tooltip, an error message is shown: "CS0152: The switch statement contains multiple cases with the label value '4'".

**you can see I am getting error here because the case value is already there so 4 is already there 2+2=4**

So, in the case you can always use + - \* / it is possible, so you can use expressions here you can use arithmetic operators here

```
namespace ConsoleApp
{
 0 references
 class SwitchDemo
 {
 0 references
 static void Main()
 {
 int no = 5;
 switch (no)
 {
 case 1: Console.WriteLine("Its 1"); break;
 case 1 + 1: Console.WriteLine("Its 2"); break;
 case 2 + 2: Console.WriteLine("Its 4"); break;
 case 1>3: Console.WriteLine("Its 4"); break;
 default:
 break;
 }
 }
 }
}
```

**but you cannot compare two values here  
in switch**

## The switch Statement

- Use switch statements for multiple case blocks
- Use break statements to ensure that no fall through occurs

```
switch (choice)
{
 case "Black":
 Console.WriteLine("Choice is Black..");
 break;
 case "Blue":
 Console.WriteLine("Choice is Blue..");
 break;
 case "Red":
 Console.WriteLine("Choice is Red..");
 default:
 Console.WriteLine("No Color choice..");
 break;
}
```

# LOOPS

## While

## The while Statement

- Execute embedded statements based on Boolean value
- Evaluate Boolean expression at beginning of loop
- Execute embedded statements while Boolean value Is True

```
int i = 0;
while (i < 10)
{
 Console.WriteLine(i);
 i++;
}
```



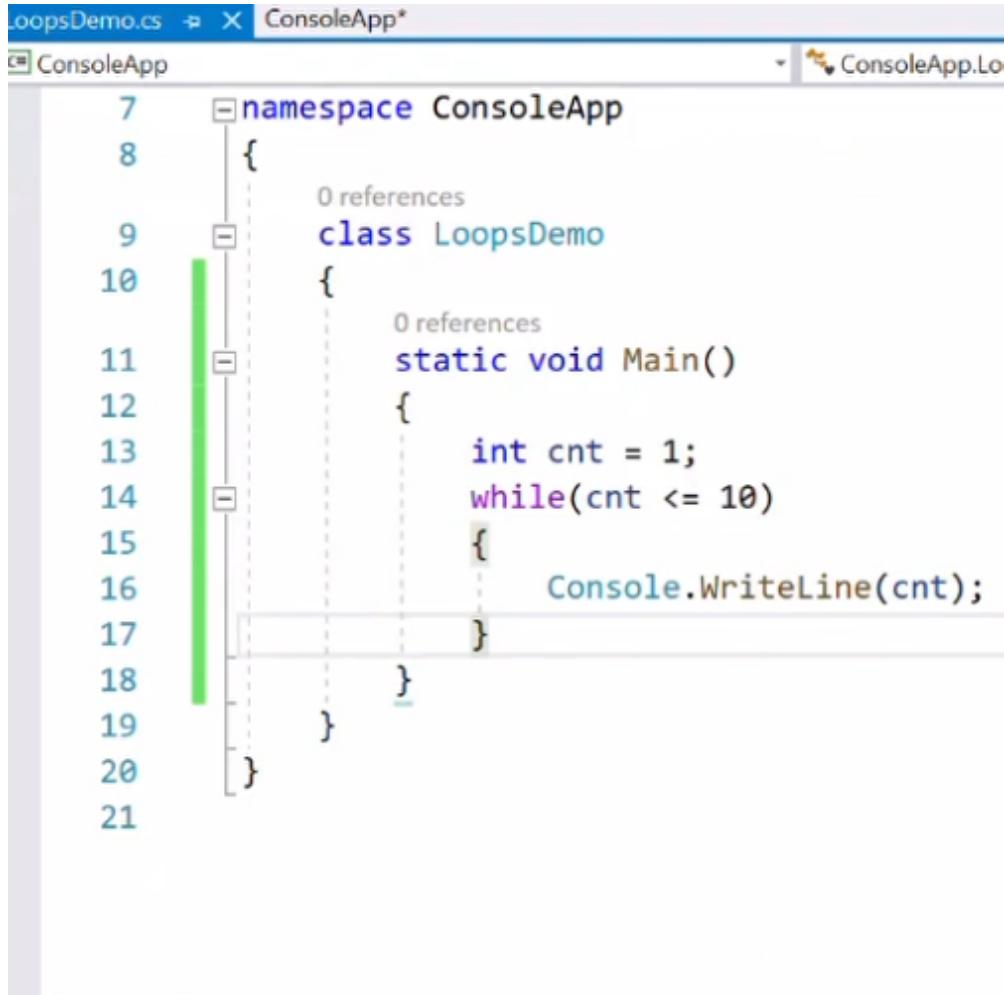
0 1 2 3 4 5 6 7 8 9

Loops are use to perform similar task again and again

**How many types of loop we have ?**

We have only 2 types of loops

1. **Entry Restricted**
2. **Exit Restricted-----even if the condition is false exist restricted will run atleast once**



```
LoopsDemo.cs X ConsoleApp*
ConsoleApp
ConsoleApp
7 namespace ConsoleApp
8 {
9 class LoopsDemo
10 {
11 static void Main()
12 {
13 int cnt = 1;
14 while(cnt <= 10)
15 {
16 Console.WriteLine(cnt);
17 }
18 }
19 }
20 }
21
```

What is the output of this loop ? --11111111111111111111111111111111.....infinite

because we are not incrementing the value of cnt here

The screenshot shows a code editor window with the file 'LoopsDemo.cs' open. The code is as follows:

```
7 namespace ConsoleApp
8 {
9 class LoopsDemo
10 {
11 static void Main()
12 {
13 int cnt = 1;
14 while(cnt <= 10)
15 {
16 Console.WriteLine(cnt++);
17 }
18 }
19 }
20 }
21
```

A green vertical bar highlights the entire code block. A red arrow points to the line 'Console.WriteLine(cnt++);'. The code is part of a project named 'ConsoleApp' with a solution named 'ConsoleApp.LoopsDemo'.

Now what is the output

```
C:\WINDOWS\system32\cmd.exe
1
2
3
4
5
6
7
8
9
10
Press any key to continue . . .
```

```
namespace ConsoleApp
{
 class LoopsDemo
 {
 static void Main()
 {
 int cnt = 1;
 while(cnt <= 10)
 {
 Console.WriteLine(++cnt);
 }
 }
 }
}
```

Now what will be the output?

```
Select C:\WINDOWS\system32\cmd.exe
2
3
4
5
6
7
8
9
10
11
Press any key to continue . . .
```

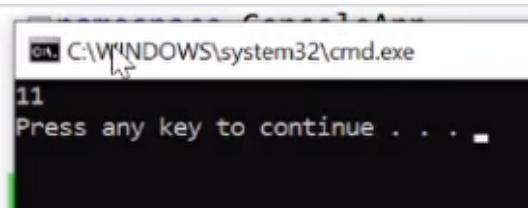
```
namespace ConsoleApp
{
 class LoopsDemo
 {
 static void Main()
 {
 int cnt = 10;
 while(cnt < 10)
 {
 Console.WriteLine(++cnt);
 }
 }
 }
}
```

Now we don't get any output because the condition is false  $10 = 10$  not  $10 < 10$

## do While

```
ConsoleApp.LoopsDem
namespace ConsoleApp
{
 class LoopsDemo
 {
 static void Main()
 {
 int cnt = 10;
 do
 {
 Console.WriteLine(++cnt);
 } while (cnt < 10);
 }
 }
}
```

Now what will be the output in case of do while



A screenshot of a Windows Command Prompt window titled "ConsoleApp". The window shows the path "C:\WINDOWS\system32\cmd.exe". The output of the program is displayed, starting with the number "11" on a new line, followed by the text "Press any key to continue . . .".

```
C:\WINDOWS\system32\cmd.exe
11
Press any key to continue . . .
```

**WhenEver you are working with loop there are 3 most important parts of the loop**

1. **Initialization**
2. **Condition Check**
3. **Increment or Decrement**

## **For Loop**

```
namespace ConsoleApp
{
 class LoopsDemo
 {
 static void Main()
 {
 int cnt = 10;
 //do
 //{
 // Console.WriteLine(++cnt);
 //} while (cnt < 10);
 for(cnt = 1; cnt <= 10; cnt++)
 {
 Console.WriteLine(cnt);
 }
 }
 }
}
```

```
ConsoleApp.LoopsDemo
namespace ConsoleApp
{
 class LoopsDemo
 {
 static void Main()
 {
 int cnt = 10;
 //do
 //{
 // Console.WriteLine(++cnt);
 //} while (cnt < 10);

 for(; cnt <= 10;)
 {
 Console.WriteLine(cnt++);
 }
 }
 }
}
```

will this work  
what will be the  
output

**o/p = 10**

**Where Can we Use While Loop and where we use For Loop  
or do-while Loop**

**Can't you do this using for----yes you can do this but you feel  
convinient /flexible with while loop**

**For-----when we go with numeric calculation , like to print all natural  
number from 1 to 100**

### **Numeric Manupulation we go with for**

I want to read the character from the user and if the character is equal to vowel (a,e,i,o,u) then i'll print the message saying this particular character is a vowel  
and if the user enter 0 then i have to exit the program

```
static void Main()
{
 int cnt = 1;
 Console.WriteLine("Enter characters:");
 do
 {
 char c = (char)Console.Read();
 if(c == '\0')
 {
 break;
 }
 switch (c)
 {
 case 'a':
 case 'e':
 case 'i':
 case 'o':
 case 'u': Console.WriteLine("Its ovel"); break;
 default:
 Console.WriteLine("Invalid character");
 break;
 }
 Console.WriteLine(++cnt);
 } while (1==2);
```

Now if i want to do something like this in for loop it will be really challenging for me that's a reason for string manupulation we'll go with while or do while

**Assignment :-take the number from user and convert that to words , for example if the number is 1264 then the output should be one two six four**

Tables.cs ✘ X ConsoleApp\*

ConsoleApp

ConsoleApp.Tables

```
2 using System.Collections.Generic;
3 using System.Linq;
4 using System.Text;
5 using System.Threading.Tasks;
6
7 namespace ConsoleApp
8 {
9 0 references
10 class Tables
11 {
12 0 references
13 static void Main()
14 {
15 for(var i=1; i<=5; i++)
16 {
17 for(var j=1; j<=10; j++)
18 {
19 Console.WriteLine("{0} x {1} = {2}", i, j, i*j);
20 }
21 }
22 }
23 }
24 }
```

133 %

No issues found

```
msc@msc-OptiPlex-5090:~/Desktop$
```

```
1 x 1 = 1
1 x 2 = 2
1 x 3 = 3
1 x 4 = 4
1 x 5 = 5
1 x 6 = 6
1 x 7 = 7
1 x 8 = 8
1 x 9 = 9
1 x 10 = 10
```

```

2 x 1 = 2
2 x 2 = 4
2 x 3 = 6
2 x 4 = 8
2 x 5 = 10
2 x 6 = 12
2 x 7 = 14
2 x 8 = 16
2 x 9 = 18
2 x 10 = 20
```

```

3 x 1 = 3
3 x 2 = 6
3 x 3 = 9
3 x 4 = 12
3 x 5 = 15
3 x 6 = 18
3 x 7 = 21
3 x 8 = 24
```

```
3 x 5 = 15
3 x 6 = 18
3 x 7 = 21
3 x 8 = 24
3 x 9 = 27
3 x 10 = 30

4 x 1 = 4
4 x 2 = 8
4 x 3 = 12
4 x 4 = 16
4 x 5 = 20
4 x 6 = 24
4 x 7 = 28
4 x 8 = 32
4 x 9 = 36
4 x 10 = 40

5 x 1 = 5
5 x 2 = 10
5 x 3 = 15
5 x 4 = 20
5 x 5 = 25
5 x 6 = 30
5 x 7 = 35
5 x 8 = 40
5 x 9 = 45
5 x 10 = 50

Press any key to continue . .
```

## GOTO demo

## The goto Statement

- Flow of control transferred to a labeled statement
- Can easily result in obscure “spaghetti” code

```
if (number % 2 == 0) goto Even;
Console.WriteLine("odd");
goto End;
Even:
Console.WriteLine("even");
End:;
```

GoToDemo.cs

ConsoleApp\*

ConsoleApp

```
class GoToDemo
{
 static void Main()
 {
 int x = 5;
 if(x == 5)
 {
 goto LessThan;
 }
 else
 {
 goto GreaterThan;
 }
 }

 LessThan: Console.WriteLine("This is 5");

 GreaterThan: Console.WriteLine("This is not equal to 5");
}
```

when ever you want to put lable you have to put  
lable name followed by : colon

133 % No issues found

how do you call the lable just write goto followed by lable name

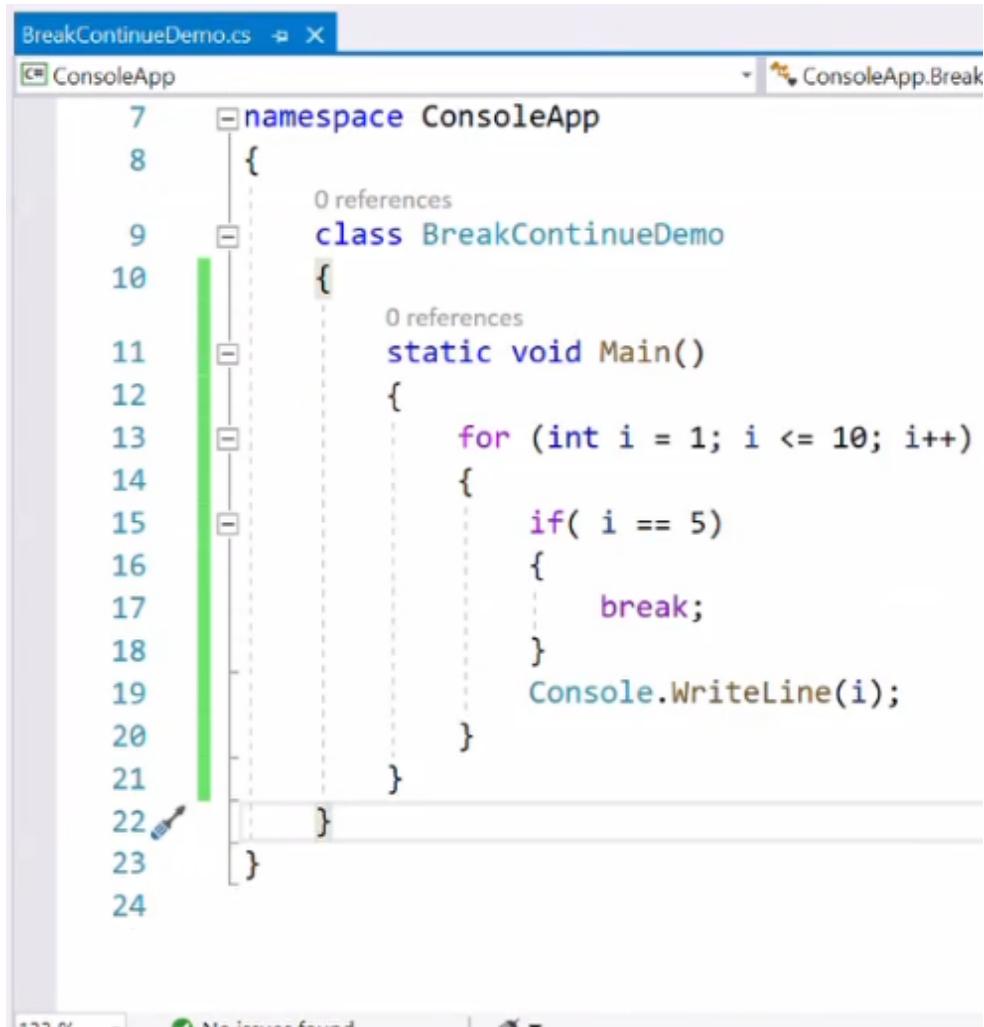
## The break and continue Statements

- The break statement jumps out of an iteration
- The continue statement jumps to the next iteration

```
int i = 0;
while (true)
{
 Console.WriteLine(i);
 i++;
 if (i < 10)
 continue;
 else
 break;
}
```

---

What will be the output ?



```
BreakContinueDemo.cs X
ConsoleApp
ConsoleApp.Break

7 namespace ConsoleApp
8 {
9 class BreakContinueDemo
10 {
11 static void Main()
12 {
13 for (int i = 1; i <= 10; i++)
14 {
15 if(i == 5)
16 {
17 break;
18 }
19 Console.WriteLine(i);
20 }
21 }
22 }
23 }
24
```

i got 1 2 3 4 because of the fact that i'll say break once the i value becomes 5 it will break this loop

```
1
2
3
4
Press any key to continue . . .
```

Now what will be the output ?

```
namespace ConsoleApp
{
 0 references
 class BreakContinueDemo
 {
 0 references
 static void Main()
 {
 for (int i = 1; i <= 10; i++)
 {
 if(i == 5)
 {
 continue;
 }
 Console.WriteLine(i);
 }
 }
 }
}
```

I will get 1 2 3 4 5will be skipped 6 7 8 9 10

```
1
2
3
4
6
7
8
9
10
Press any key to continue . . .
```

Now What will be the output ?

The screenshot shows a code editor window with the following C# code:

```
ConsoleApp.cs
ConsoleApp
namespace ConsoleApp
{
 class BreakContinueDemo
 {
 static void Main()
 {
 for (int i = 1; i <= 10; i++)
 {
 for(int j = 1; j<= 10; j++)
 {
 if(i == 5)
 {
 break;
 }
 Console.WriteLine(i);
 }
 Console.WriteLine("-----" + i + "-----");
 }
 }
 }
}
```

A yellow lightbulb icon is positioned next to the opening brace of the inner for loop on line 15. The code editor interface includes a status bar at the bottom with the text "133 %", a "No issues found" indicator, and a zoom control.

1,2,3,4, ,6,7,8,9,10 print 10 10 times but 5 will not print

So, when you say break what will happen ----- **When ever you put break so break will always break the loop where it is currently present , if it is present in the inner loop it will**

## break the inner loop only same with continue

as we are saying break when  $i==5$  so it will keep breaking when  $i==5$  so 5 will not get printed

## Defining Methods

- Main is a method
  - Use the same syntax for defining your own

```
using System;

class ExampleClass
{
 static void ExampleMethod()
 {
 Console.WriteLine("Example method");
 }
 static void Main()
 {
 // ...
 }
}
```

Whenever you have any sorts of statements to execute push that to a particular method  
We always go with

# SRP -Single Responsibility Principle

## Assignment :-Read on SOLID principle

As per the solid principle your one method should not perform more than one task, because if you don't do this so maintenance becomes very difficult

1. maintainance becomes difficult
2. Debugging becomes difficult
3. Reading becomes difficult

Suppose 9 to 6 you are sitting for the classes ,imagine after 6 o'clock you have to goto shop and manage your shop

6:30 to 8 you will manage your shop and from 10 you'll go somewhere else and work will you be efficient, no doubt you will gone to work but will you be effecient -----No  
at any given point of time you should perform only one task , you should not perform more than one task

so any program should have a IPO---Input process output

Calc.cs\* ✘ X ConsoleApp<sup>4</sup>

ConsoleApp

```
7 namespace ConsoleApp
8 {
9 class Calc
10 {
11 static void Main()
12 {
13 //IPO - UI ↗
14 int no1, no2;
15 Console.WriteLine("Enter number 1:");
16 no1 = Convert.ToInt32(Console.ReadLine());
17 Console.WriteLine("Enter number 2:");
18 no2 = Convert.ToInt32(Console.ReadLine());
19 //SRP - Single Res Pri : SOLID
20 }
21 }
22 }
23 }
24 }
```

this is my UI code and my  
UI code should not merge  
with the process code

Calcs ✘ X ConsoleApp\*

ConsoleApp

ConsoleApp.Calc

```
13 //IPO - UI
14 int no1, no2;
15 string oper;
16 Console.WriteLine("Enter number 1:");
17 no1 = Convert.ToInt32(Console.ReadLine());
18 Console.WriteLine("Enter number 2:");
19 no2 = Convert.ToInt32(Console.ReadLine());
20 Console.WriteLine("Enter the operation(+,-,*,/):");
21 oper = Console.ReadLine();
22 //SRP - Single Res Pri : SOLID
23 int result = Sum(no1, no2);
24 switch (oper)
25 {
26 case "+": result = Sum(no1, no2); break;
27 case "-": result = Diff(no1, no2); break;
28 case "*": result = Multi(no1, no2); break;
29 case "/": result = Div(no1, no2); break;
30 default:
31 Console.WriteLine("Invalid operator");
32 break;
33 }
34 Dispaly(no1, no2, result, oper);
```

133 %

No issues found

```
1 reference
private static void Dispaly(int no1, int no2, int result, string oper)
{
 Console.WriteLine("{0} {1} {2} = {3}",no1, oper, no2, result);
}
```

```
1 reference
private static int Div(int no1, int no2)
{
 return no1 / no2;
}
```

```
1 reference
private static int Multi(int no1, int no2)
{
 return no1 * no2;
}
```

[?] (parameter) int no2

```
1 reference
private static int Diff(int no1, int no2)
{
 return no1 - no2;
}
```

1 found | ⌂ ▾

Look at the above code it is more maintainable more clean so u should always follow SRP

```
C:\WINDOWS\system32\cmd.exe
[1] 12
Enter number 1:
12
Enter number 2:
23
Enter the operation(+,-,*,/):
+
12 + 23 = 35
Press any key to continue . . .
```

## Calling Methods

- After you define a method, you can:
  - Call a method from within the same class
    - Use method's name followed by a parameter list in parentheses
  - Call a method that is in a different class
    - You must indicate to the compiler which class contains the method to call
    - The called method must be declared with the **public** keyword
  - Use nested calls
    - Methods can call methods, which can call other methods, and so on

## Using the return Statement

- Immediate return
- Return with a conditional statement

```
static void ExampleMethod()
{
 int numBeans;
 //...

 Console.WriteLine("Hello");
 if (numBeans < 10)
 return;
 Console.WriteLine("World");
}
```

## Using Local Variables

- Local variables
  - Created when method begins
  - Private to the method
  - Destroyed on exit
- Shared variables
  - Class variables are used for sharing
- Scope conflicts
  - Compiler will not warn if local and class names clash

Local variables are those variables which are created inside the method ,private to that particular method and it gets destroyed with that particular method

So whenever you create a particular method you can always even specify the scope just like your let and var

```
LocalDemo.cs* X ConsoleApp
ConsoleApp ConsoleApp.LocalDemo

4 using System.Text;
5 using System.Threading.Tasks;
6
7 namespace ConsoleApp
8 {
9 class LocalDemo
10 {
11 static void Main()
12 {
13 for (int i = 0; i < 10; i++)
14 {
15 Console.WriteLine(i);
16 }
17 Console.WriteLine("Value of I is " + i);
18 }
19 }
20 }
21
```

it will print 1 to 10



but look here  
i am having  
Error

ConsoleApp

ConsoleApp.LocalDemo

Main()

```
4 using System.Text;
5 using System.Threading.Tasks;
6
7 namespace ConsoleApp
8 {
9 class LocalDemo
10 {
11 static void Main()
12 {
13 for (int i = 0; i < 10; i++)
14 {
15 Console.WriteLine(i);
16 }
17 Console.WriteLine("Value of I is " + i);
18 }
19 }
20 }
```

it very clearly says Boss i don't have i



CS0103: The name 'i' does not exist in the current context  
Show potential fixes (Alt+Enter or Ctrl+.)

```
4 using System.Text;
5 using System.Threading.Tasks;
6
7 namespace ConsoleApp
8 {
9 0 references
10 class LocalDemo
11 {
12 0 references
13 static void Main()
14 {
15 for (int i = 0; i < 10; i++)
16 {
17 Console.WriteLine(i);
18 }
19 }
20 }
21 }
```

because i have declared  
i inside this block

just like your let it will only be accessible  
only in the block where it is declared

outside the block it cannot be accessible

So this are my BLOCK Variables

LocalDemo.cs | ConsoleApp

ConsoleApp

ConsoleApp.LocalDemo

```
4 using System.Text;
5 using System.Threading.Tasks;
6
7 namespace ConsoleApp
8 {
9 class LocalDemo
10 {
11 static void Main()
12 {
13 int i = 100;
14 for (int i = 0; i < 10; i++)
15 {
16 Console.WriteLine(i);
17 }
18 Console.WriteLine("Value of I is " + i);
19 }
20 }
21 }
22
```

133 %    X 1    0    ← → | ⌂ ▾

Output Package Manager Console Error List Immediate Window