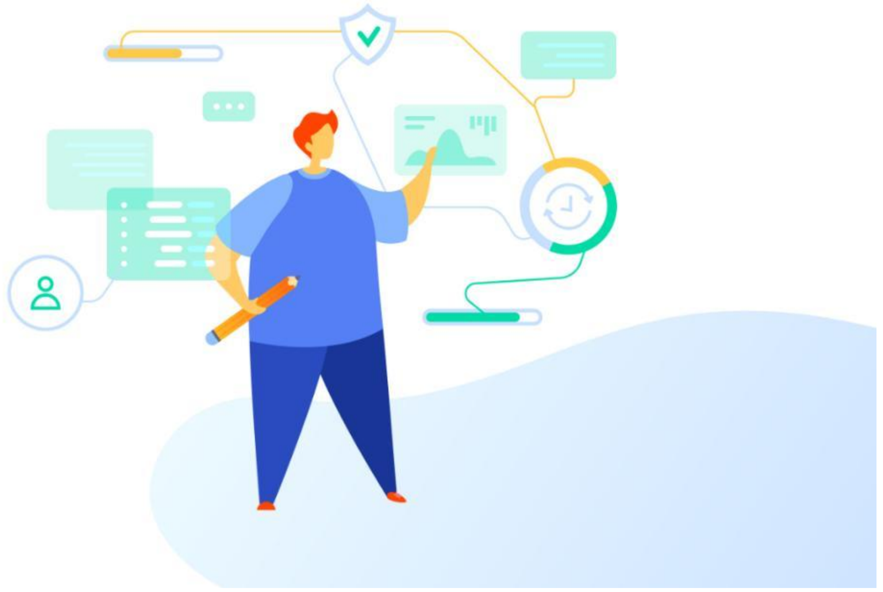
C#

Aitrich

Introduction

- Microsoft developed C#.
- Anders Hejlsberg, the principal architect of C#.
- C# will be familiar to those, who have programmed in C, C++, or Java.
- C# is standardized by ECMA.

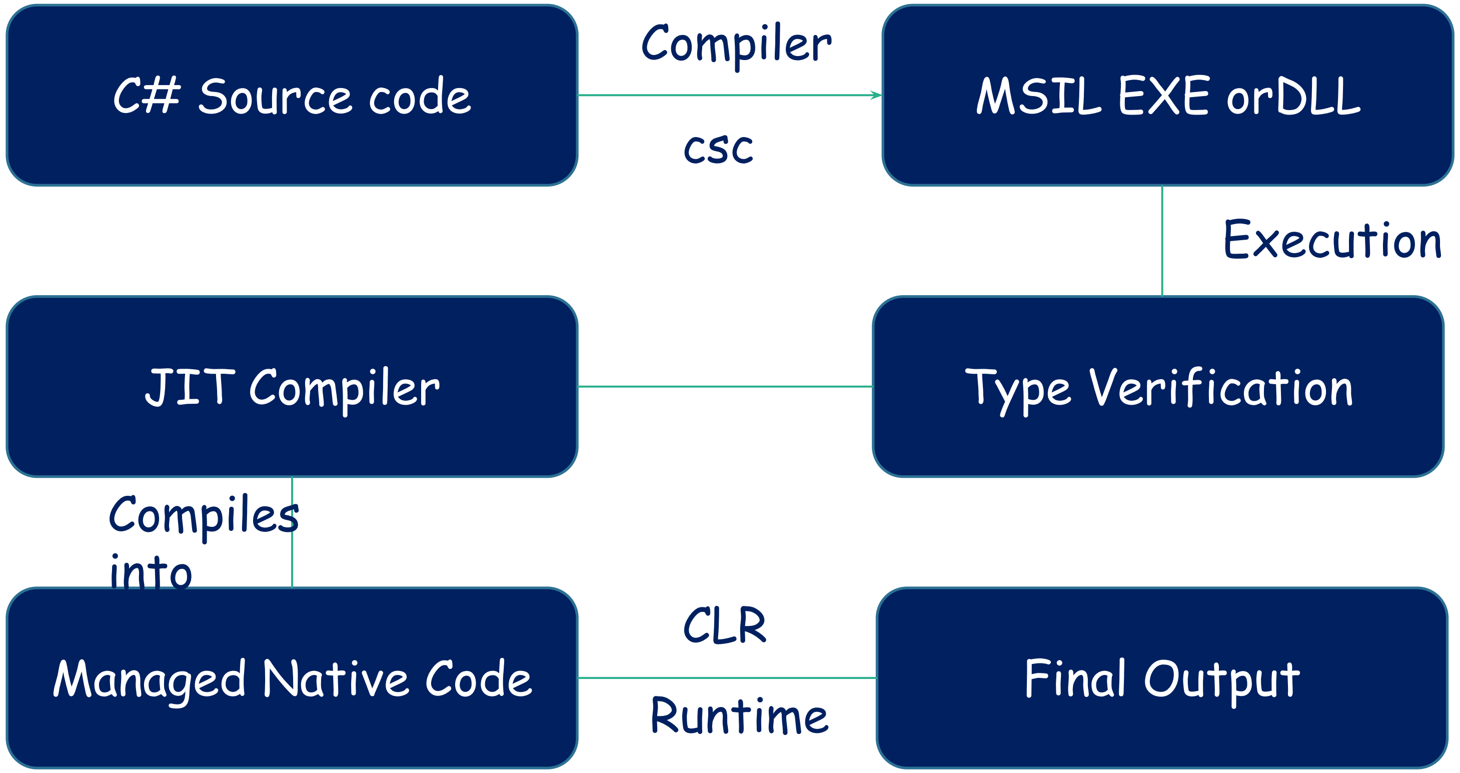


Aitrich

Overview

- Introduction to C#
- What is C Sharp ?
- Working of C Sharp
- Features of C Sharp
- Basic Programming Structure
- Namespaces
- Data Types
- Variables
- Expressions and Operators
- Control Statement- Select
- Control Statement- Loop
- Array , Enums

Working of C#


C# Source code
Compiler
csc
MSIL EXE orDLL
Execution
Type Verification
JIT Compiler
Compiles into
Managed Native Code
CLR
Runtime
Final Output

Compilation of C sharp

It can be compiled with the following command line:

```
csc Welcome.cs
```

This produces a file named Welcome.exe, which can thenbe executed.

Aitrich

What is c#

- The C# programming language (pronounced "C-Sharp") .
- C Sharp is a powerful object-oriented programming languages.
- Microsoft developed C#, a new programming language based on the C and C++ languages.
- C# is a case sensitive language.
- C# support component-oriented programming.

Aitrich

C# Language Features

•Many of the features in C# language are preexisted in various languages such as C++, Java, Pascal, and Visual Basic.

- Object Oriented.
- Simple and Flexible.
- Automatic Memory Management.
- Cross Platform Interoperability.

C# Editors and IDEs

Notepad

Visual Studio .NET

Application Types In C#

Class Library:- Creates a project for creating classes that can be used in other applications.

Console Application:- Creates a Visual C# application with a command-line interface.

Asp.Net Web Application:- Creates a Visual C# application with a Web user interface.

Asp.Net Web Service:- Creates an XML Web service with Visual C# that other applications can access.

Mobile Application:- Creates an application for mobile devices.

Programming Structure

```
using System;
class JobPortal
{
        static void Main()
        {
         Console.WriteLine("Welcome to the Job Portal");
        }
}
```

Programming Structure

The- source code for a C# program is typically stored in one text files with a file extension of .cs,

say in welcome.cs.

Primary Elements

- a namespace declaration.
- a class.
- a Main method.
- a program statement.
- a comments

Data types

❑C# supports mainly two kinds of types.

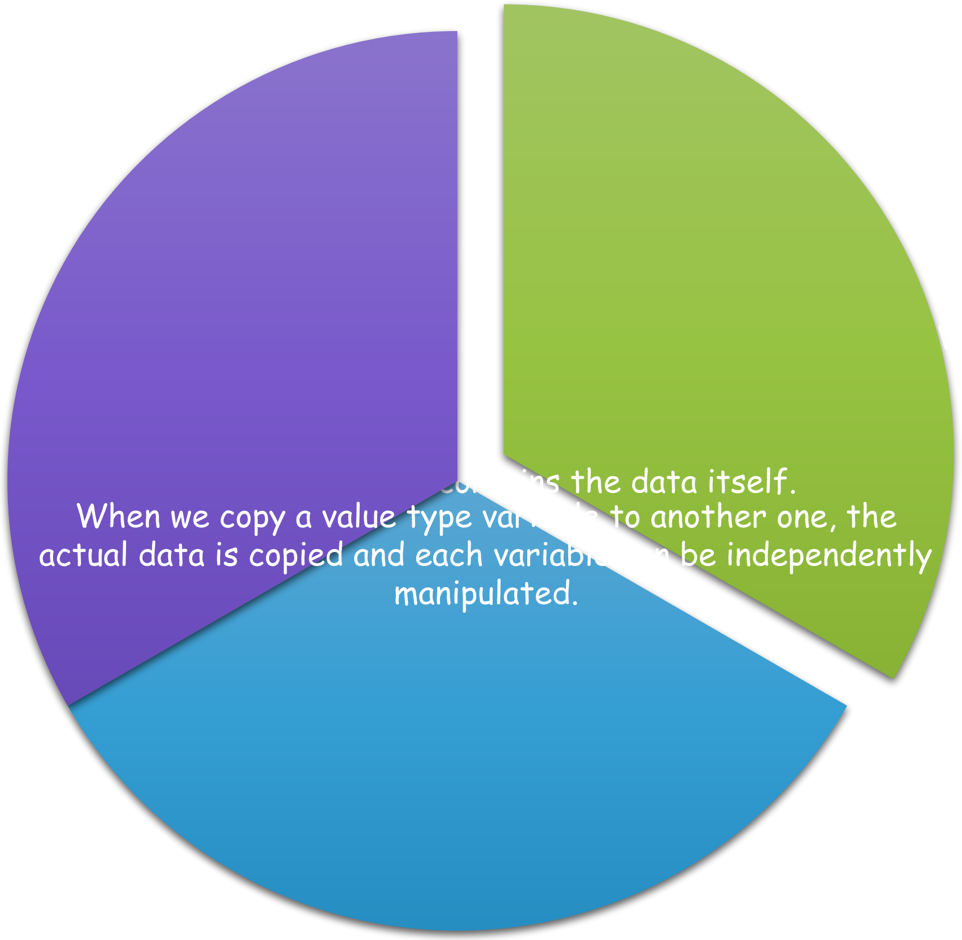
✔value types .
✔Reference types.

❑value types are – char, int, structures, enums .

❑Reference types are – class, interface, delegate, arrays.

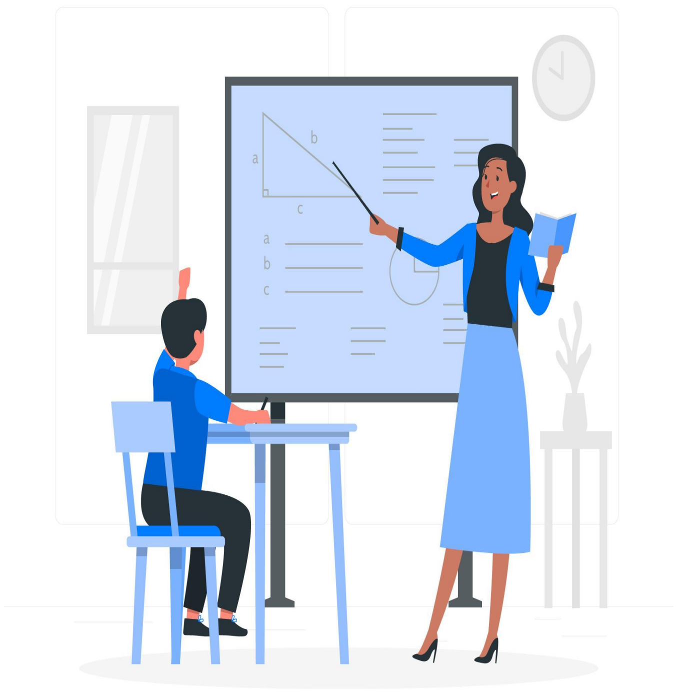
Aitrich

Value type



contains the data itself.

When we copy a value type variable to another one, the actual data is copied and each variable can be independently manipulated.



Reference types

A reference type is a reference to an instance type.

Reference type variable contains the address of memory location where data is actually stored

A null value is assigned to a reference type by default.

Thank You

Variables

- ❑ "Variables" are simply storage locations for data.
- ❑ You can place data into them and retrieve their contents as part of a C# expression.
- ❑ The interpretation of the data in a variable is controlled through "Types".
- ❑ In C#, you declare a variable in this format:
- ❑ [modifiers] datatype identifier;

```
public static string role = admin;
```

Aitrich

Control Flow Statements

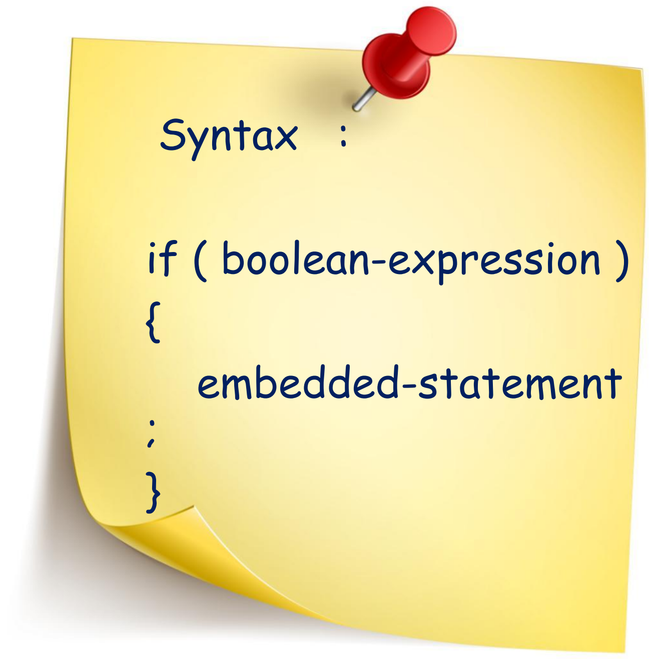
- ❑ Control flow and program logic are of the most important parts of a programming language.

- ❑ Selection statements select one of a number of possible statements for execution based on the value of some expression.

  - The if statement
  - The if-else Statement
  - The if-else if-else Statement
  - The Nested if-else Statement
  - The switch statement

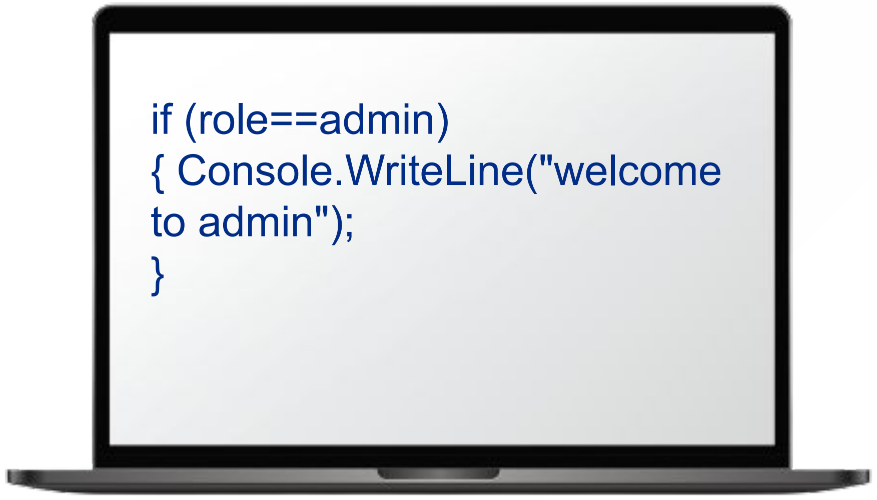
If Statement

Use the if statement to specify a block of C# code to be executed if a condition is True.



Syntax :

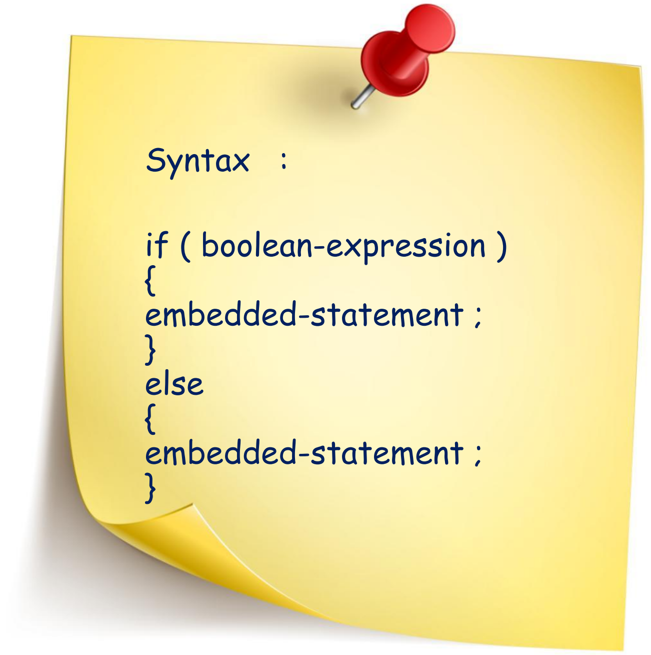
```
if ( boolean-expression )
{
    embedded-statement
;
}
```



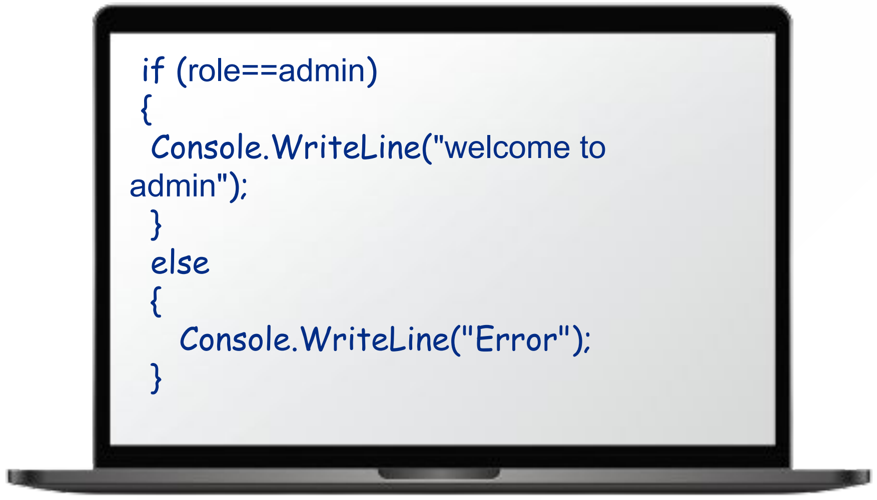
```
if (role==admin)
{ Console.WriteLine("welcome
to admin");
}
```

If/else Statement

Use the else statement to specify a block of code to be executed if the condition is False.



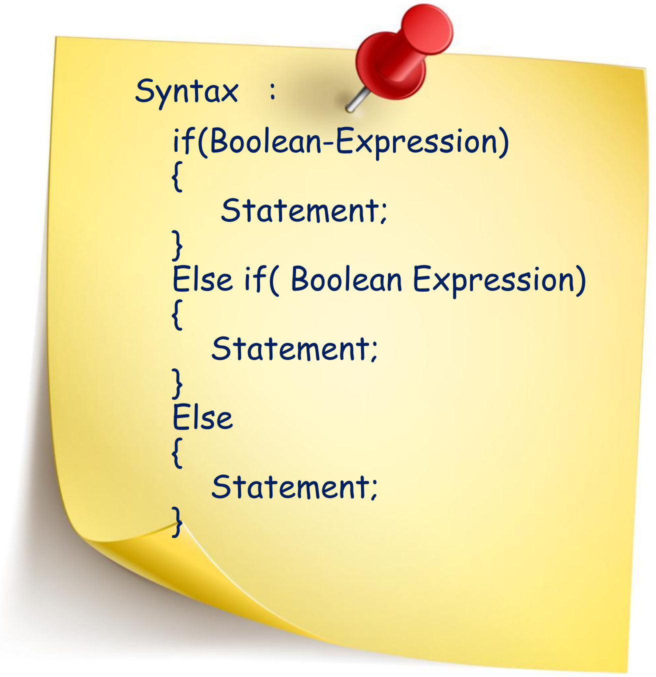
```
Syntax   :

if ( boolean-expression )
{
embedded-statement ;
}
else
{
embedded-statement ;
}
```



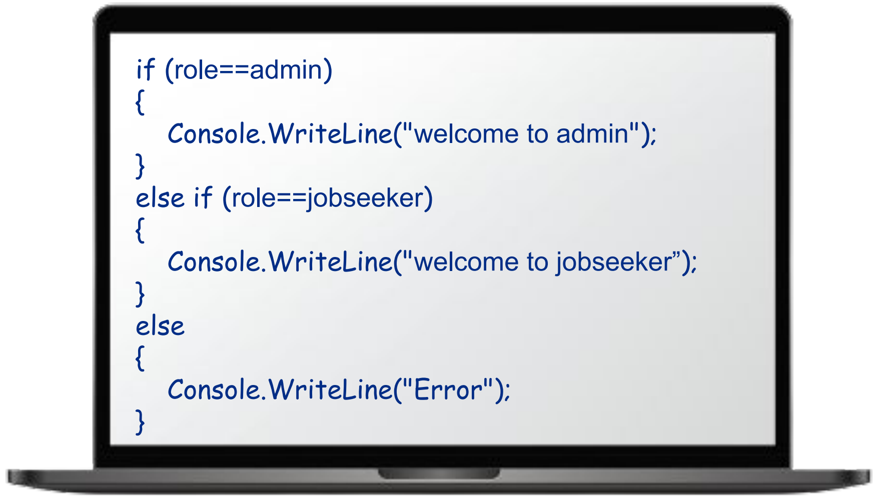
```
 if (role==admin)
 {
  Console.WriteLine("welcome to
admin");
  }
  else
  {
     Console.WriteLine("Error");
  }
```

if/else if/else Statement

Use the else if statement to specify a new condition if the first condition is False.



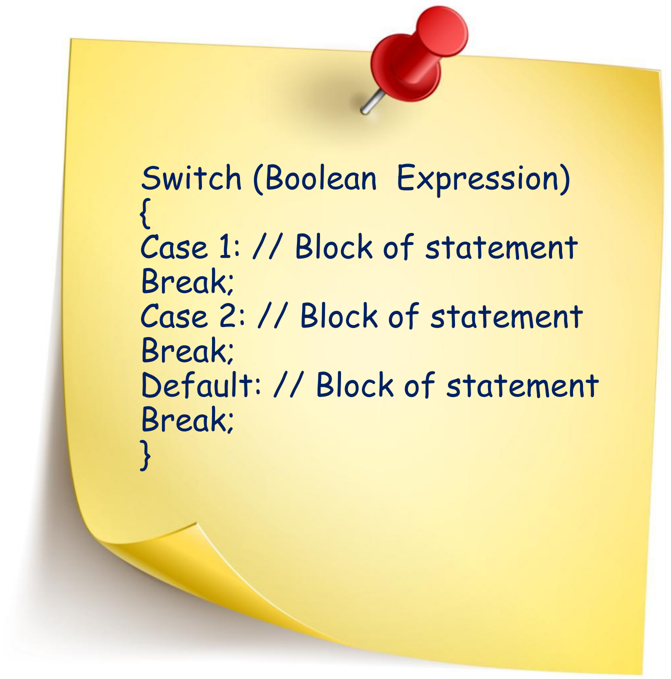
```
Syntax  :
    if(Boolean-Expression)
    {
         Statement;
    }
    Else if( Boolean Expression)
    {
        Statement;
    }
    Else
    {
        Statement;
    }
```



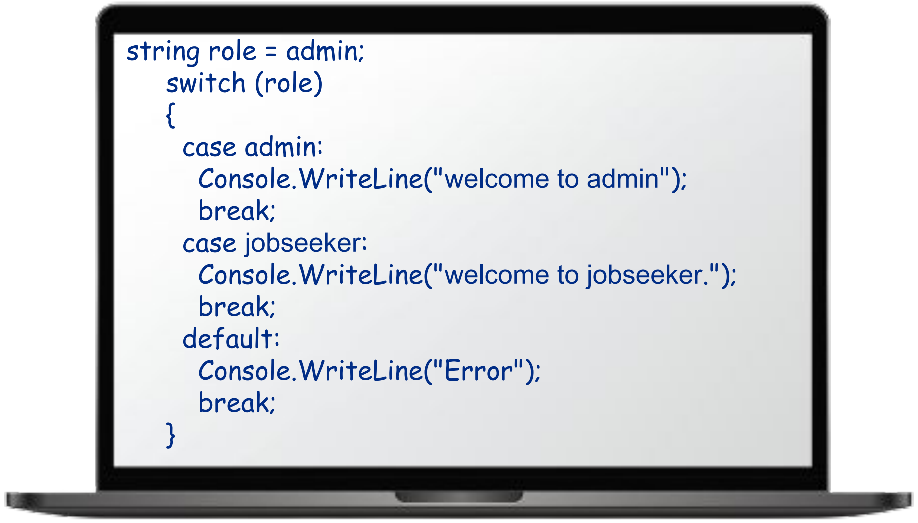
```
if (role==admin)
{
   Console.WriteLine("welcome to admin");
}
else if (role==jobseeker)
{
   Console.WriteLine("welcome to jobseeker");
}
else
{
   Console.WriteLine("Error");
}
```

Switch Statement

Use the switch statement to select one of many code blocks to be executed.



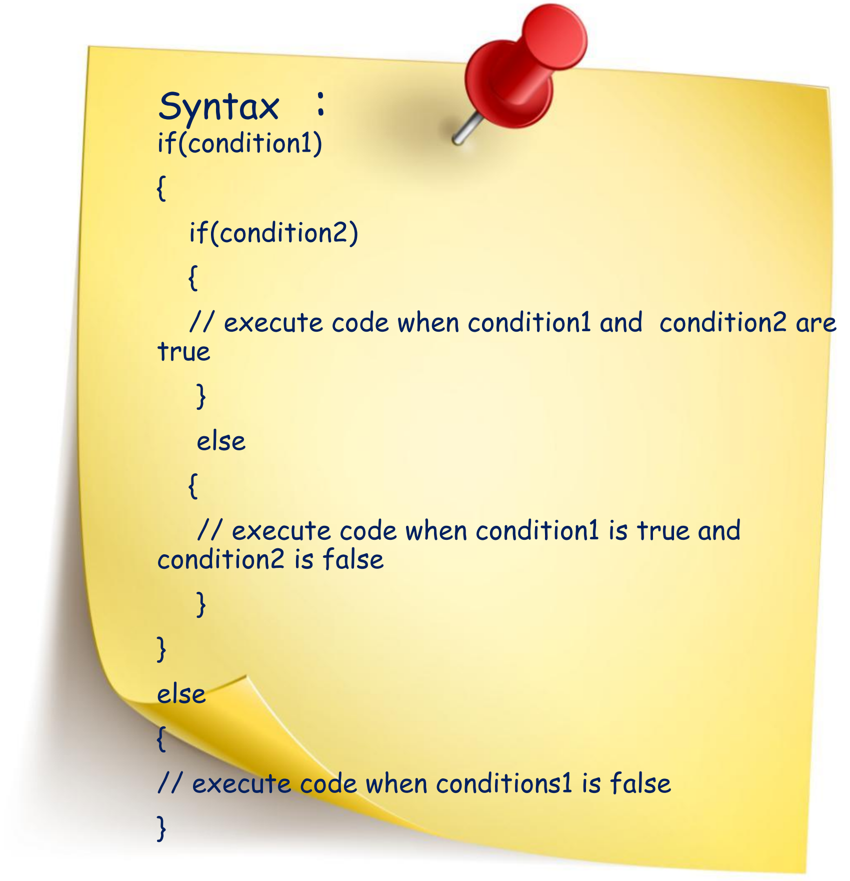
```
Switch (Boolean  Expression)
{
Case 1: // Block of statement
Break;
Case 2: // Block of statement
Break;
Default: // Block of statement
Break;
}
```



```
string role = admin;
   switch (role)
   {
    case admin:
      Console.WriteLine("welcome to admin");
      break;
    case jobseeker:
      Console.WriteLine("welcome to jobseeker.");
      break;
    default:
      Console.WriteLine("Error");
      break;
   }
```

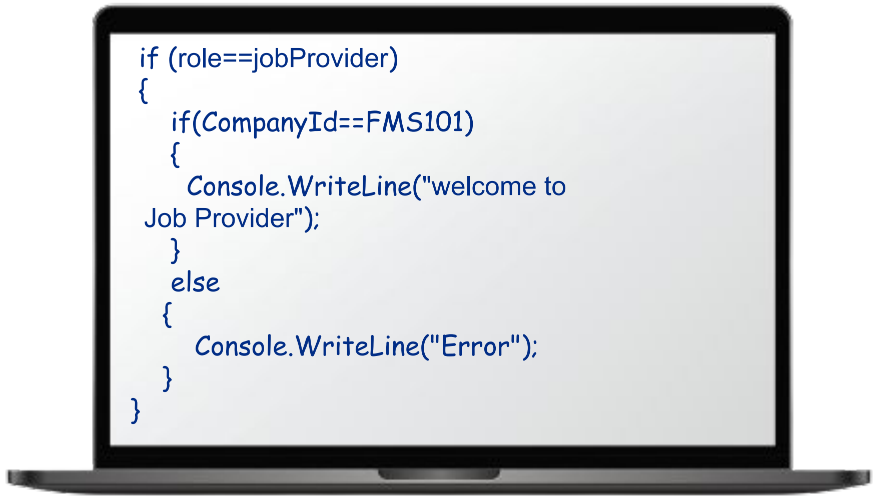
Nested If-else Statement

Nested IF functions, meaning one IF function inside of another, allows you to test multiple criteria and increases the number of possible outcomes.



Syntax :

```
if(condition1)
{
    if(condition2)
    {
    // execute code when condition1 and  condition2 are
true
     }
     else
    {
     // execute code when condition1 is true and
condition2 is false
     }
}
else
{
// execute code when conditions1 is false
}
```



```
if (role==jobProvider)
{
    if(CompanyId==FMS101)
    {
      Console.WriteLine("welcome to
 Job Provider");
    }
    else
   {
      Console.WriteLine("Error");
   }
}
```

Control Statement- Loop

Iteration statements repeatedly execute an embedded statement.

- while-statement
- do-while statement
- for-statement
- foreach-statement

Aitrich

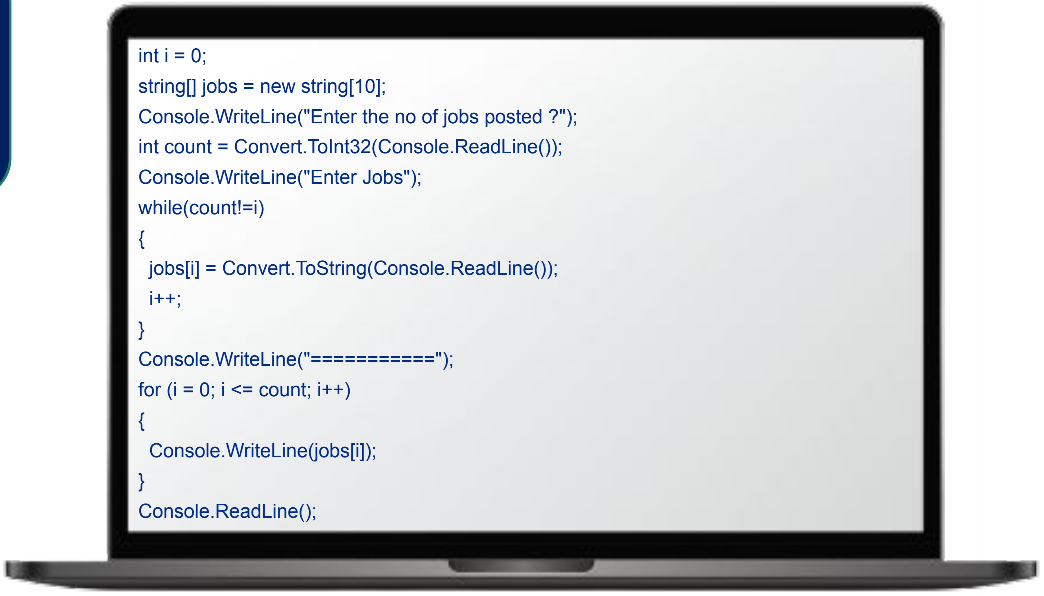
While-statement

The while statement conditionally executes an embedded statement zero or more times.

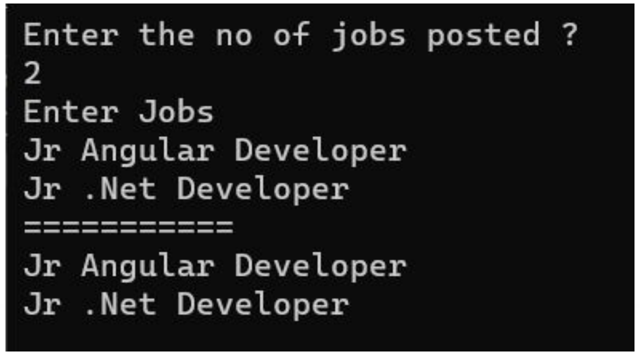
Syntax:

```
while( condition )
{
  embedded-statement;
}
```

Example



```
int i = 0;
string[] jobs = new string[10];
Console.WriteLine("Enter the no of jobs posted ?");
int count = Convert.ToInt32(Console.ReadLine());
Console.WriteLine("Enter Jobs");
while(count!=i)
{
 jobs[i] = Convert.ToString(Console.ReadLine());
 i++;
}
Console.WriteLine("===========");
for (i = 0; i <= count; i++)
{
 Console.WriteLine(jobs[i]);
}
Console.ReadLine();
```



```
Enter the no of jobs posted ?
2
Enter Jobs
Jr Angular Developer
Jr .Net Developer
===========
Jr Angular Developer
Jr .Net Developer
```

Aitrich

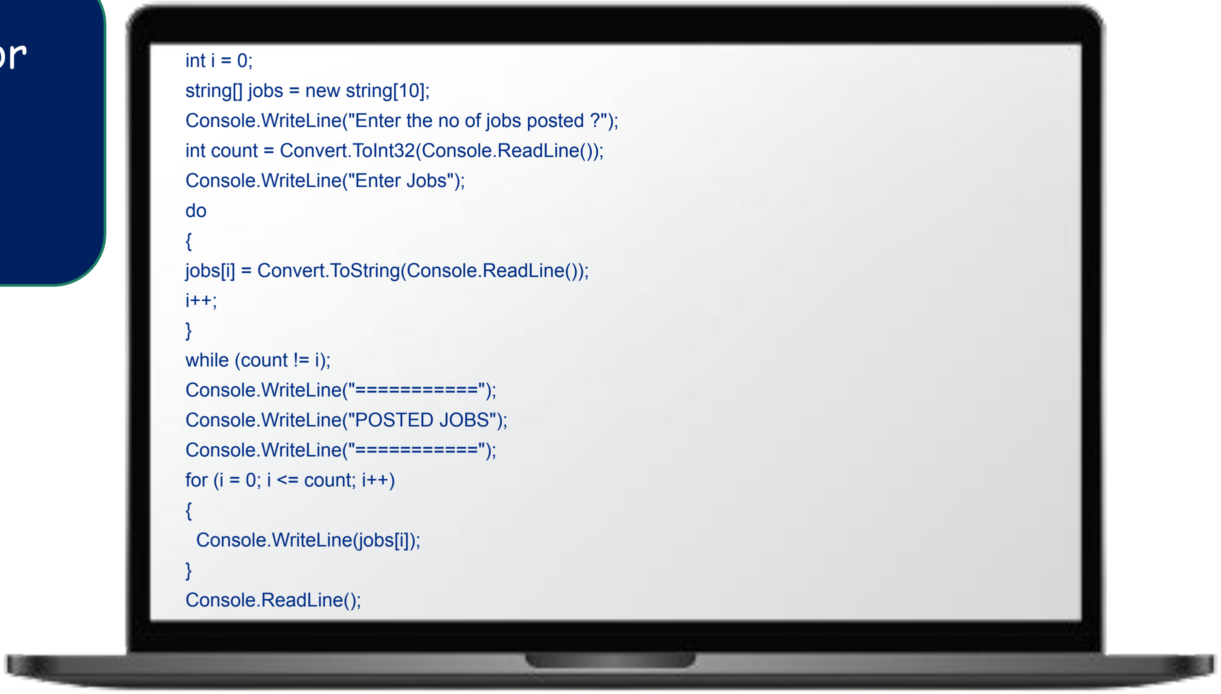
do-while Statement

This statement executes its embedded statements one or more times.
Unlike the while Statement ,a do-while loop is executed once before the conditional expression evaluated.

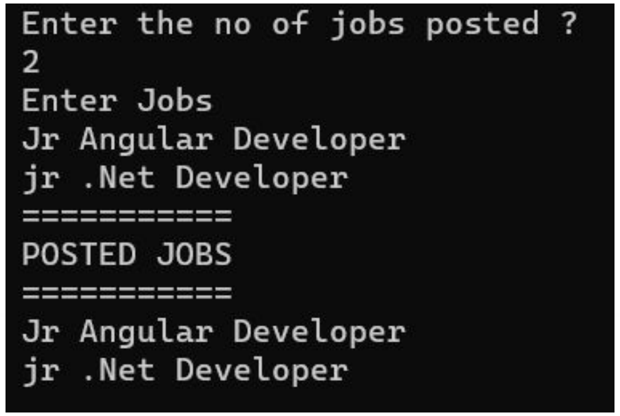
Syntax:

```
do
{
   //code to be executed
}
 while   ( condition )
```

Example



```
int i = 0;
string[] jobs = new string[10];
Console.WriteLine("Enter the no of jobs posted ?");
int count = Convert.ToInt32(Console.ReadLine());
Console.WriteLine("Enter Jobs");
do
{
jobs[i] = Convert.ToString(Console.ReadLine());
i++;
}
while (count != i);
Console.WriteLine("===========");
Console.WriteLine("POSTED JOBS");
Console.WriteLine("===========");
for (i = 0; i <= count; i++)
{
  Console.WriteLine(jobs[i]);
}
Console.ReadLine();
```



```
Enter the no of jobs posted ?
2
Enter Jobs
Jr Angular Developer
jr .Net Developer
===========
POSTED JOBS
===========
Jr Angular Developer
jr .Net Developer
```

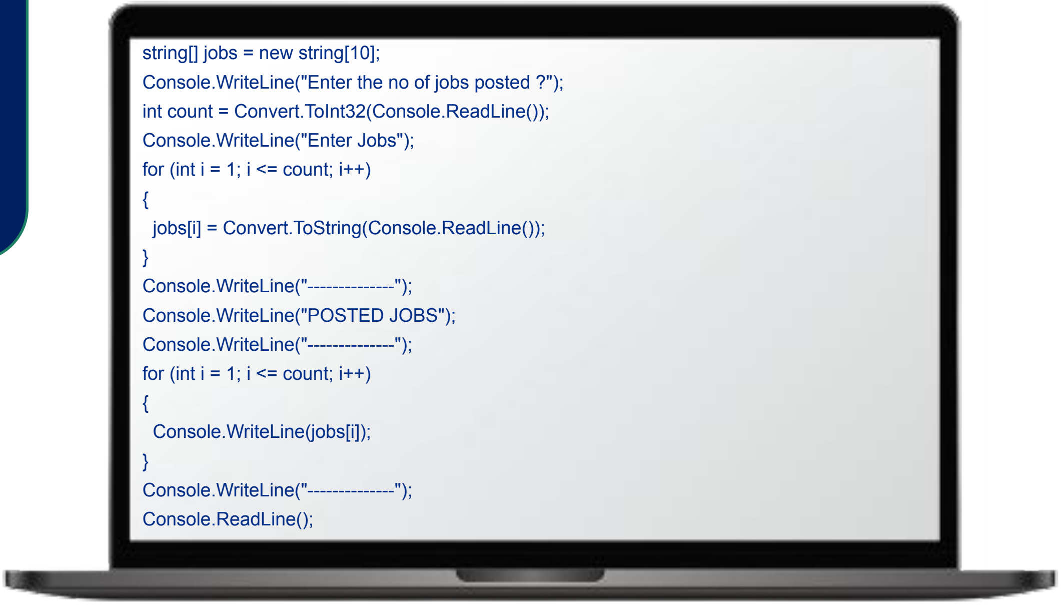
The for statement

This statement begins with the for keyword and is followed by parentheses.
The parentheses contain an initializer, a condition, and an iterator statement, all separated by semicolons.

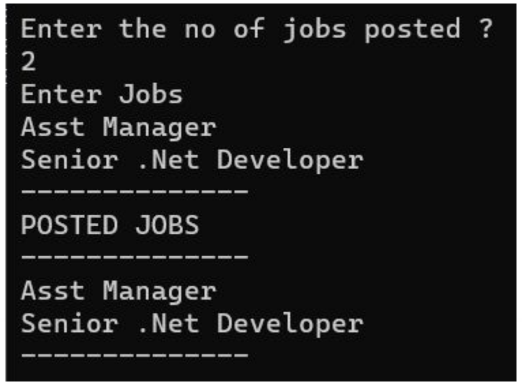
Syntax:

```
for ( Initialization  ; condition  ; Iterator )
{
Statement;
}
```

Example



```
string[] jobs = new string[10];
Console.WriteLine("Enter the no of jobs posted ?");
int count = Convert.ToInt32(Console.ReadLine());
Console.WriteLine("Enter Jobs");
for (int i = 1; i <= count; i++)
{
  jobs[i] = Convert.ToString(Console.ReadLine());
}
Console.WriteLine("--------------");
Console.WriteLine("POSTED JOBS");
Console.WriteLine("--------------");
for (int i = 1; i <= count; i++)
{
  Console.WriteLine(jobs[i]);
}
Console.WriteLine("--------------");
Console.ReadLine();
```



```
Enter the no of jobs posted ?
2
Enter Jobs
Asst Manager
Senior .Net Developer
--------------
POSTED JOBS
--------------
Asst Manager
Senior .Net Developer
--------------
```

Aitrich

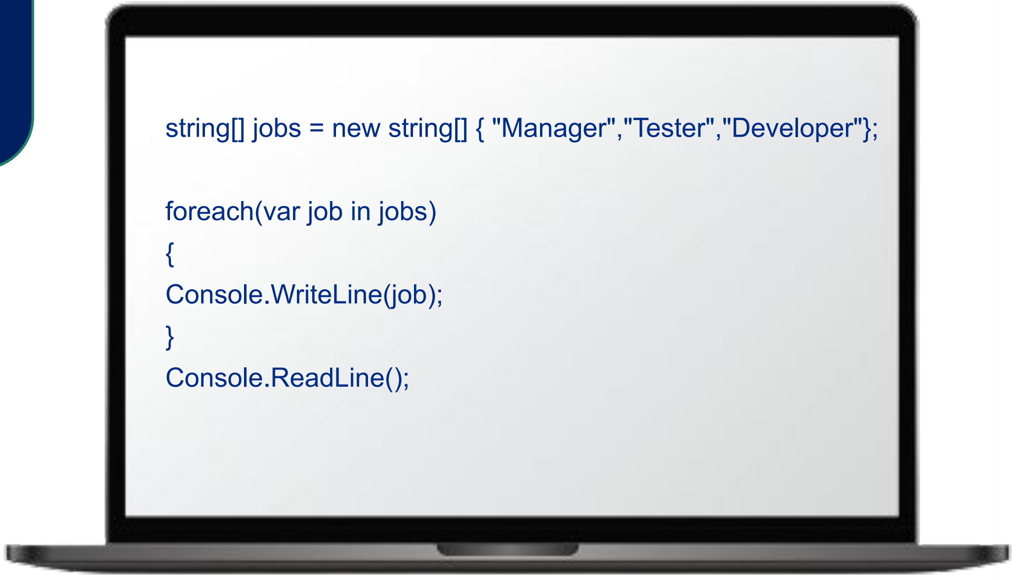
The foreach statement

The foreach statement enumerates the elements of a collection, executing an embedded statement for each element of the collection.

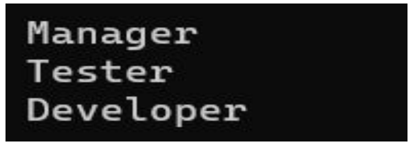
Syntax :

```
foreach(local-variable-type   identifier
in   expression   )
{
embedded-statement
}
```

Example



```
string[] jobs = new string[] { "Manager","Tester","Developer"};

foreach(var job in jobs)
{
Console.WriteLine(job);
}
Console.ReadLine();
```



```
Manager
Tester
Developer
```

Array

- An array is a data structure.
- The variables contained in an array, also called the elements of the array.
- The variables contained in an array all of the same type.
- Array types are reference types.
- An array index starts at zero.
- That means the first item of an array starts at the 0th position.
- The position of the last item on an array will be total number of items - 1.
- So if an array has 10 items, the last 10th item is at 9thposition.
- In C#, arrays can be declared as fixed length or dynamic.
- A fixed length array can store a predefined number of items.

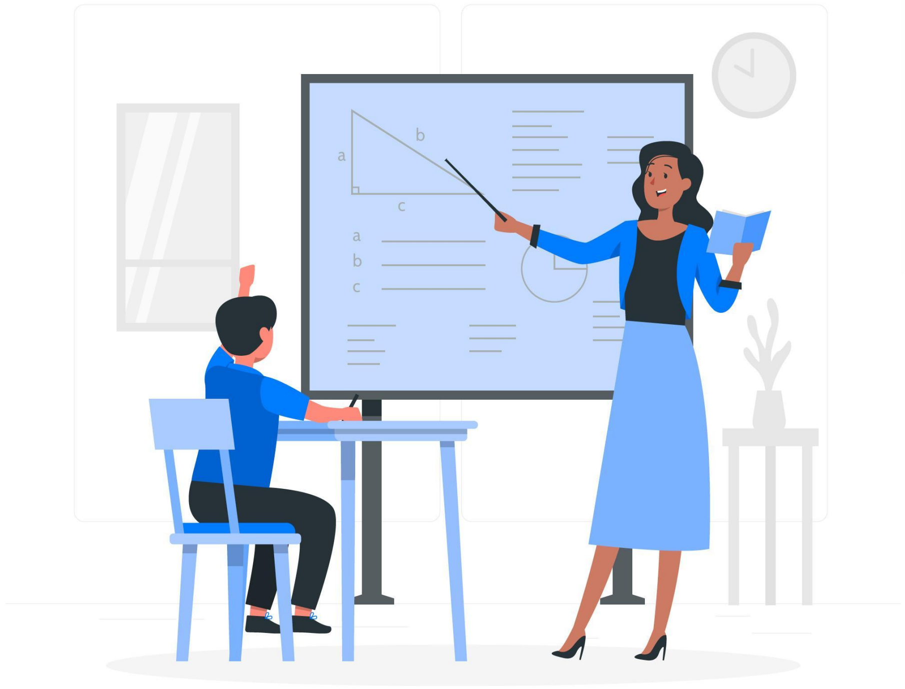
Array

A dynamic array does not have a predefined size.

The size of a dynamic array increases as you add new items to the array.

Defining arrays of different types.

```
double[] doubleArray = new double[5];
char[] charArray = new char[5];
bool[] boolArray = new bool[2];
string[] stringArray = new string[10];
```



Initializing Arrays.

Once an array is declared, the next step is to initialize an array.

The initialization process of an array includes adding actual data to the array.

First:

```
// Initialize a fixed array
int[] FixedArray = new int[3] {1, 3, 5};

// Initialize a dynamic array items during declaration

string[] jobs= new string[] { "Manager", "Developer",
"Accountant", "Marketting};
```

Array Types

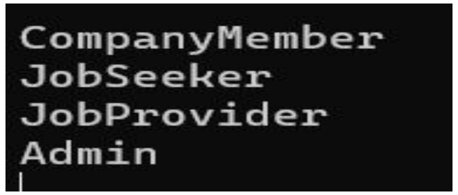
Arrays can be divided into the following categories.

✔Single-dimensional arrays
✔Multidimensional arrays or rectangular arrays
✔Jagged arrays

Single-dimensional arrays

Single-dimensional array is a collection of elements of the same type, arranged in a sequential manner. It is the most basic form of an array and can be declared and used as follows:

```
// Example: Creating an integer array with 4 elements
string[] roles= new string[4];

// Assigning values to array elements
roles[0] = "CompanyMember";
roles[1] = "JobSeeker";
roles[2] = "JobProvider";
roles[3] = "Admin";

// Iterating over array elements using a loop
for (int i = 0; i < roles.Length; i++)
{
    Console.WriteLine(roles[i]);
}
Console.ReadLine();
```

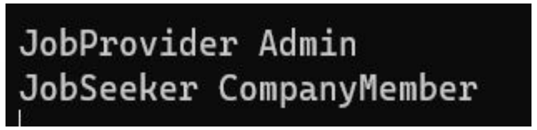


```
CompanyMember
JobSeeker
JobProvider
Admin
```

Multi-dimensional arrays

The multidimensional array is also known as rectangular arrays in C#. It can be two dimensional or three dimensional. The data is stored in tabular form (row * column) which is also known as matrix.

```
//declaration of 2D array
string[,] roles = new string[2, 2];

roles[0, 0] = "JobProvider";                //initialization
roles[0, 1] = "Admin";
roles[1, 0] = "JobSeeker";
roles[1, 1] = "CompanyMember";

//traversal
for (int i = 0; i < 2; i++)
{
    for (int j = 0; j <2; j++)
    {
      Console.Write(roles[i,j] + " ");
     }
     Console.WriteLine();          //new line at each row
}
Console.ReadLine();
```

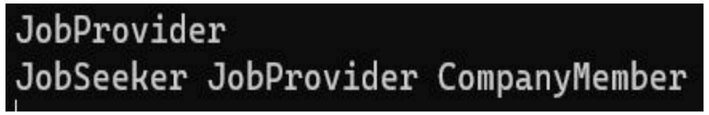


```
JobProvider Admin
JobSeeker CompanyMember
```

Jagged Arrays

In C#, jagged array is also known as "array of arrays" because its elements are arrays. The element size of jagged array can be different

```
// Declare the array
string[][] roles = new string[2][];

// Initialize the array
roles[0] = new string[] {"JobProvider" };
roles[1] = new string[] { "JobSeeker",
"JobProvider","CompanyMember" };

// Traverse array elements
for (int i = 0; i < roles.Length; i++)
{
   for (int j = 0; j < roles[i].Length; j++)
   {
      System.Console.Write(roles[i][j] + " ");
   }
   Console.WriteLine();
 }
Console.ReadLine();
```



```
JobProvider
JobSeeker JobProvider CompanyMember
```

Enum

- ❑ An enum type is a distinct value type with a set of named constants.
- ❑ The enum keyword is used to declare an enumeration.
- ❑ Enums are strongly typed constants.
- ❑ All member of enum are of enum type.
- ❑ Enums type can be integer (float, int, byte, double etc.).
- ❑ The default underlying type of the enumeration element is int.

Enum

- ❑ By default, the first enumerator has the value 0, and the value of each successive enumerator is increased by 1.
- ❑ Enumerations (enums) make your code much more readable and understandable.
- ❑ Every enum type automatically derives from System.Enum

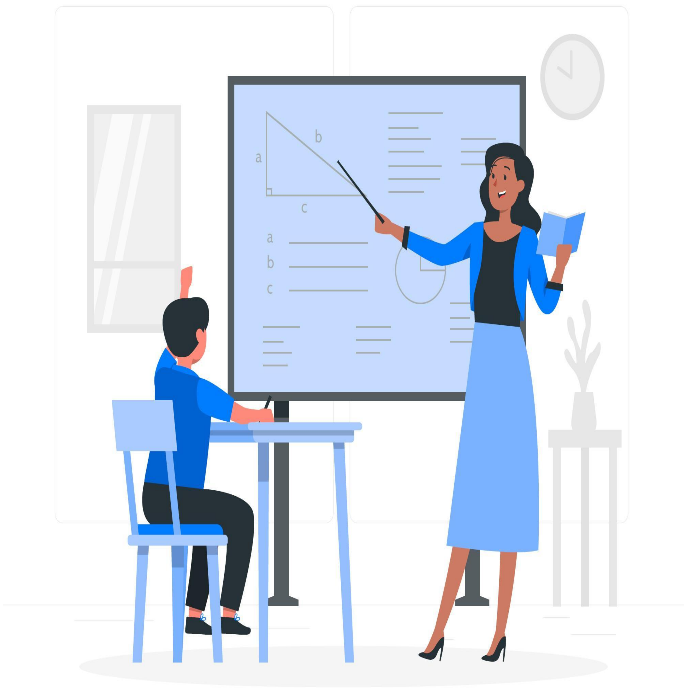
Syntax:

An enum is declared as follows:

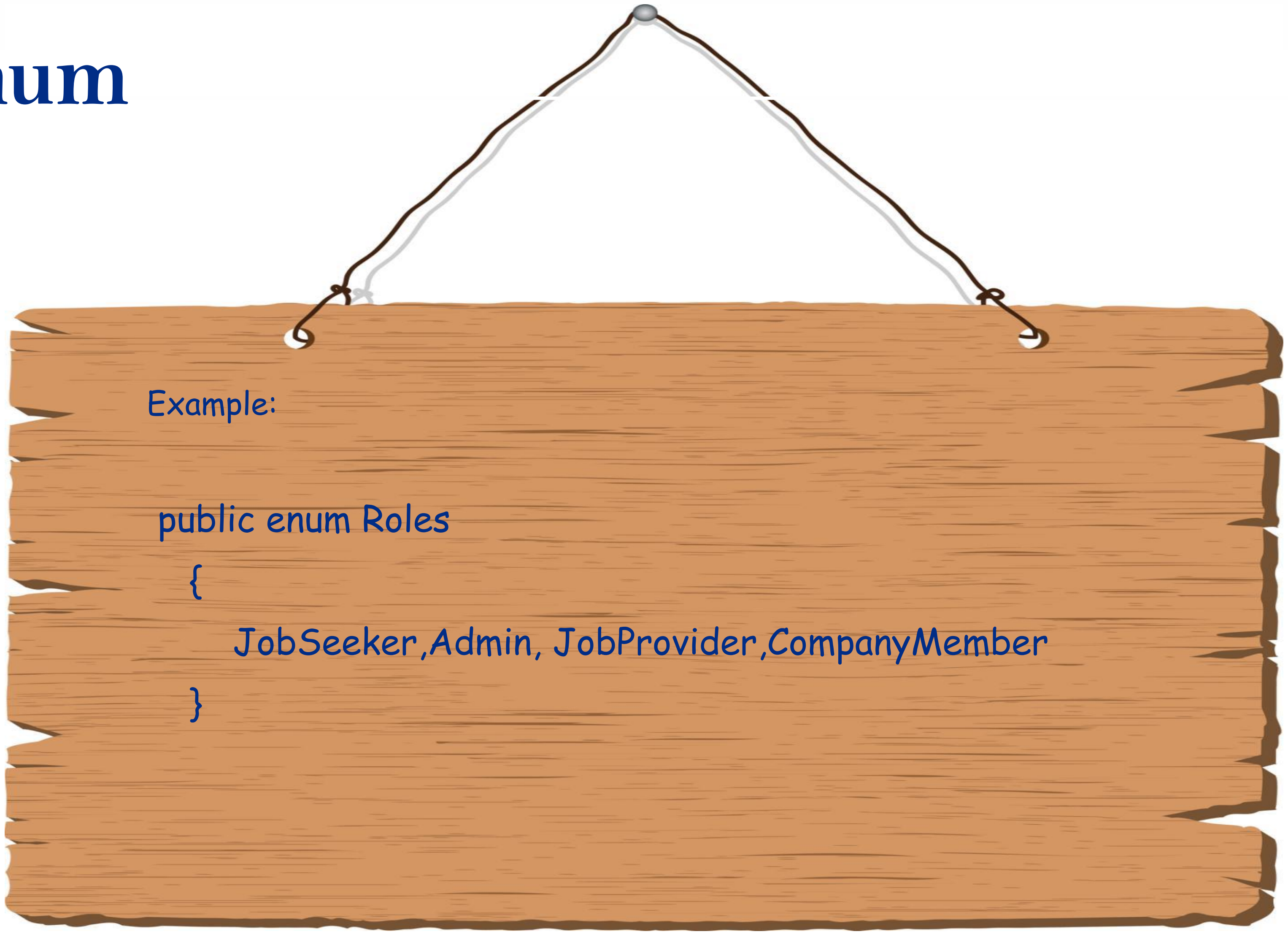
```
[modifiers] enum identifier
    {
        enumerator-list [,]
    }
```

Enum

- The attributes is optional and is used to hold additional declarative information.
- Modifiers are new, public, protected, internal and private.
- The keyword enum must be followed by an identifier that names the enum.
- The underlying type that specifies the storage allocated for each enumerator. It can be one of the integral types except char. The default is int..
- The enumerator-list contains the identifiers which are separated by commas



Enum



Example:

```
public enum Roles
   {
       JobSeeker,Admin, JobProvider,CompanyMember
   }
```

Type Conversions

Type conversion is converting one type of data to another type. It is also known as Type Casting.
In C# Type casting has two forms

1. Implicit type conversion
2. Explicit type conversion

Aitrich

Implicit type conversion

❑ conversions from smaller to larger integral types

```
int salary= 250000;
long sal= salary;
// implicit conversion from int type to long type
```

Explicit type conversion

- ❑ Explicit type conversion:- These conversions are done explicitly by users using the pre-defined functions.

```
long salary= 25000;
int sal= (int)salary;
// explicit conversion from long type to int type
```

Aitrich

Structures

A structure is a value type data type. It helps you to make a single variable hold related data of various data types. The struct keyword is used for creating a structure.

Structures are used to represent a record.

For example, declare the job structure

```
struct Books {
                    public string title;
                    public string Name;
                    public int Salary;
                  };
```

Thank You