**1) What is C-Sharp (C#)?**

C# is a type-safe, managed and object oriented language, which is compiled by .Net framework for generating intermediate language (IL).

**2) Explain the features of C#?**

Below are some of the features supported in C# -

* Constructors and Destructors
* Properties
* Passing Parameters
* Arrays
* Main
* XML Documentation and
* Indexers

**3) Explain sealed class in C#?**

Sealed class is used to prevent the class from being inherited from other classes. So “sealed” modifier also can be used with methods to avoid the methods to override in the child classes.

**Example**

class X {}

sealed class Y : X {}

Sealed methods –

class A

{

protected virtual void First() { }

protected virtual void Second() { }

}

class B : A

{

sealed protected override void First() {}

protected override void Second() { }

}

**4) Differences between Array and ArrayList in C#?**

* Array stores the values or elements of same data type but arraylist stores values of different datatypes.
* Arrays will use the fixed length but arraylist does not uses fixed length like array.

**5) What is the difference between “constant” and “readonly” variables in C#?**

* “Const” keyword is used for making an entity constant. We cannot modify the value later in the code. Value assigning is mandatory to constant variables.
* “readonly” variable value can be changed during runtime and value to readonly variables can be assigned in the constructor or at the time of declaration.

**6) Explain “static” keyword in C#?**

“Static” keyword can be used for declaring a static member. If the class is made static then all the members of the class are also made static. If the variable is made static then it will have a single instance and the value change is updated in this instance.

**7) Can we execute multiple catch blocks in C#?**

No. Once any exception is occurred it executes specific exception catch block and the control comes out.

**8) What is the difference between “finalize” and “finally” methods in C#?**

* Finalize – This method is used for garbage collection. So before destroying an object this method is called as part of clean up activity.
* Finally – This method is used for executing the code irrespective of exception occurred or not.

**9)** **Can we use “this” inside a static method in C#?**

No. We can’t use “this” in static method.

**10) What are reference types in C#?**

Below are the list of reference types in C# -

* class
* string
* interface
* object

**11)** **Can we override private virtual method in C#?**

No. We can’t override private virtual methods as it is not accessible outside the class.

**12) In try block if we add return statement whether finally block is executed in C#?**

Yes. Finally block will still be executed in presence of return statement in try block.

**13) Explain String Builder class in C#?**

This will represent the mutable string of characters and this class cannot be inherited. It allows us to Insert, Remove, Append and Replace the characters. “ToString()” method can be used for the final string obtained from StringBuilder. For example,

StringBuilder TestBuilder = new StringBuilder("Hello");

TestBuilder.Remove(2, 3); // result - "He"

TestBuilder.Insert(2, "lp"); // result - "Help"

TestBuilder.Replace('l', 'a'); // result - "Heap"

**14) How we can sort the array elements in descending order in C#?**

“Sort()” method is used with “Reverse()” to sort the array in descending order.

**15) List out some of the exceptions in C#?**

Below are some of the exceptions in C# -

* NullReferenceException
* ArgumentNullException
* DivideByZeroException
* IndexOutOfRangeException
* InvalidOperationException
* StackOverflowException etc.

**16) Explain Generics in C#?**

Generics in c# is used to make the code reusable and which intern decreases the code redundancy and increases the performance and type safety.   
Namespace – “System.Collections.Generic” is available in C# and this should be used over “System.Collections” types.

**17) What you mean by delegate in C#?**

Delegates are type safe pointers unlike function pointers as in C++. Delegate is used to represent the reference of the methods of some return type and parameters.

**18) What are the types of delegates in C#?**

Below are the uses of delegates in C# -

* Single Delegate
* Multicast Delegate
* Generic Delegate

**19) What are the three types of Generic delegates in C#?**

Below are the three types of generic delegates in C# -

* Func
* Action
* Predicate

**20) What are the differences between events and delegates in C#?**

Main difference between event and delegate is event will provide one more of encapsulation over delegates. So when you are using events destination will listen to it but delegates are naked, which works in subscriber/destination model.

**21) Can we use delegates for asynchronous method calls in C#?**

Yes. We can use delegates for asynchronous method calls.

**22) What are the uses of delegates in C#?**

Below are the list of uses of delegates in C# -

* Callback Mechanism
* Asynchronous Processing
* Abstract and Encapsulate method
* Multicasting

**22) Define Multicast Delegate in C#?**

A delegate with multiple handlers are called as multicast delegate. The example to demonstrate the same is given below

public delegate void CalculateMyNumbers(int x, int y);

int x = 6;

int y = 7;

CalculateMyNumbers addMyNumbers = new CalculateMyNumbers(FuncForAddingNumbers);

CalculateMyNumbers multiplyMyNumbers = new CalculateMyNumbers(FuncForMultiplyingNumbers);

CalculateMyNumbers multiCast = (CalculateMyNumbers)Delegate.Combine (addMyNumbers, multiplyMyNumbers);

multiCast.Invoke(a,b);

**23)What is enum in C#?**

enum keyword is used for declaring an enumeration, which consists of named constants and it is called as enumerator lists. Enums are value types in C# and these can’t be inherited. Below is the sample code of using Enums

Eg: enum Fruits { Apple, Orange, Banana, WaterMelon};

**24)sample code to write the contents to text file in C#**

Using System.IO;

File.WriteAllText(”mytextfilePath”, “MyTestContent”);

**25) What you mean by boxing and unboxing in C#?**

Boxing – This is the process of converting from value type to reference type. For example,

int myvar = 10;

object myObj = myvar;

UnBoxing – It’s completely opposite to boxing. It’s the process of converting reference type to value type. For example,

int myvar2 = (int)myObj;

**26) Explain Partial Class in C#?**

Partial classes concept added in .Net Framework 2.0 and it allows us to split the business logic in multiple files with the same class name along with “partial” keyword.

**27) Explain Static constructor in C#**

If the constructor is declared as static then it will be invoked only once for all number of instances of a class. Static constructor will initialize the static fields of a class.

class MyClass

{

public string prop1, prop2;

public MyClass(string a, string b)

{

prop1 = a;

prop2 = b;

}

Static MyClass()

{

Console.WriteLine(“Static Constr Test”);

}

public MyClass(MyClass myobj) // Copy Constructor

{

prop1 = myobj.prop1;

prop2 = myobj.prop2;

}

}

**28) Explain Indexers in C#?**

Indexers are used for allowing the classes to be indexed like arrays. Indexers will resemble the property structure but only difference is indexer’s accessors will take parameters. For example,

class MyCollection<T>

{

private T[] myArr = new T[100];

public T this[int t]

{

get

{

return myArr[t];

}

set

{

myArr[t] = value;

}

}

}

**29) List out the pre defined attributes in C#?**

Below are the predefined attributes in C# -

* Conditional
* Obsolete
* Attribute Usage

**30) What is Thread in C#?**

Thread is an execution path of a program. Thread is used to define the different or unique flow of control. If our application involves some time consuming processes then it’s better to use Multithreading., which involves multiple threads.

**31) List out the states of a thread in C#?**

Below are the states of thread –

* Unstarted State
* Ready State
* Not Runnable State
* Dead State

**32) Explain the methods and properties of Thread class in C#?**

Below are the methods and properties of thread class –

* CurrentCulture
* CurrentThread
* CurrentContext
* IsAlive
* IsThreadPoolThread
* IsBackground
* Priority

**33) What are the Access Modifiers in C# ?**

Different Access Modifier are - Public, Private, Protected, Internal, Protected Internal

* Public – When a method or attribute is defined as Public, it can be accessed from any code in the project. For example, in the above Class “Employee” getName() and setName() are public.
* Private - When a method or attribute is defined as Private, It can be accessed by any code within the containing class only. For example, in the above Class “Employee” attributes name and salary can be accessed within the Class Employee Only. If an attribute or class is defined without access modifiers, it's default access modifier will be private.
* Protected - When attribute and methods are defined as protected, it can be accessed by any method in the inherited classes and any method within the same class. The protected access modifier cannot be applied to classes and interfaces. Methods and fields in a interface can't be declared protected.
* Internal – If an attribute or method is defined as Internal, access is restricted to classes within the current project assembly.
* Protected Internal – If an attribute or method is defined as Protected Internal, access is restricted to classes within the current project assembly and types derived from the containing class.

**34) Explain the use of Virtual Keyword in C# ?**

When we want to give permission to a derived class to override a method in base class, Virtual keyword is used. For example. lets us look at the classes Car and Ford as shown below.

class Car

{

public Car()

{

Console.WriteLine("Base Class Car");

}

public virtual void DriveType()

{

Console.WriteLine("Right Hand Drive");

}

}

class Ford : Car

{

public Ford()

{

Console.WriteLine("Derived Class Ford");

}

public void Price()

{

Console.WriteLine("Ford Price : 100K $");

}

public override void DriveType()

{

Console.WriteLine("Right Hand ");

}

}

When following lines of code get executed

Car CarFord = new Car();

CarFord.DriveType();

CarFord = new Ford();

CarFord.DriveType();

Output is as given below.

Base Class Car

Right Hand Drive

Base Class Car

Derived Class Ford

Right Hand

**35) What is Abstract Class in C#?**

If we don't want a class to be instantiated, define the class as abstract. An abstract class can have abstract and non abstract classes. If a method is defined as abstract, it must be implemented in derived class. For example, in the classes given below, method DriveType is defined as abstract.

abstract class Car

{

public Car()

{

Console.WriteLine("Base Class Car");

}

public abstract void DriveType();

}

class Ford : Car

{

public void DriveType()

{

Console.WriteLine("Right Hand ");

}

}

Method DriveType get implemented in derived class.

**36) What is Sealed Classes in c# ?**

If a class is defined as Sealed, it cannot be inherited in derived class. Example of a sealed class is given below.

public sealed class Car

{

public Car()

{

Console.WriteLine("Base Class Car");

}

public void DriveType()

{

Console.WriteLine("Right Hand ");

}

}