**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) List some of the advantages of C#?**

Below are the advantages of C# -

* Easy to learn
* Object oriented
* Component oriented
* Part of .NET framework

**4) What are IDE’s provided by Microsoft for C# development?**

Below are the IDE’s used for C# development –

* Visual Studio Express (VCE)
* Visual Studio (VS)
* Visual Web Developer

**5) Explain the types of comments in C#?**

Below are the types of comments in C# -

* Single Line Comment Eg : //
* Multiline Comments Eg: /\* \*/
* XML Comments Eg : ///

**6) 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.

**7) Give an example of using sealed class in C#?**

Below is the sample code of sealed class in C# -

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() { }  
}

If any class inherits from class “B” then method – “First” will not be overridable as this method is sealed in class B.

**Must Read** - [.Net Framework Interview Questions and Answers](http://a4academics.com/interview-questions/52-dot-net-interview-questions/421-net-interview-questions-and-answers-on-net-framework)

**8) List out the 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.

**9) Why to use “using” in C#?**

“Using” statement calls – “dispose” method internally, whenever any exception occurred in any method call and in “Using” statement objects are read only and cannot be reassignable or modifiable.

**10) Explain namespaces in C#?**

Namespaces are containers for the classes. We will use namespaces for grouping the related classes in C#. “Using” keyword can be used for using the namespace in other namespace.

**11) Why to use keyword “const” in C#? Give an example.**

“Const” keyword is used for making an entity constant. We can’t reassign the value to constant.

Eg: const string \_name = "Test";

**12) 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.

**Must Read** - [ADO.Net Interview Questions and Answers](http://a4academics.com/interview-questions/52-dot-net-interview-questions/488-ado-net-interview-questions-and-answers) 

**13) 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.

**14) What is the difference between “dispose” and “finalize” variables in C#?**

* Dispose - This method uses interface – “IDisposable” interface and it will free up both managed and unmanaged codes like – database connection, files etc.
* Finalize - This method is called internally unlike Dispose method which is called explicitly. It is called by garbage collector and can’t be called from the code.

**15) How the exception handling is done in C#?**

In C# there is a “try… catch” block to handle the error.

**16) 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.

**17) Why to use “finally” block in C#?**

“Finally” block will be executed irrespective of exception. So while executing the code in try block when exception is occurred, control is returned to catch block and at last “finally” block will be executed. So closing connection to database / releasing the file handlers can be kept in “finally” block.

**18) 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.

**19) What is the difference between “throw ex” and “throw” methods in C#?**

* “throw ex” will replace the stack trace of the exception with stack trace info of re throw point.
* “throw” will preserve the original stack trace info.

**20) Can we have only “try” block without “catch” block in C#?**

Yes we can have only try block without catch block but we have to have finally block.

**21) List out two different types of errors in C#?**

Below are the types of errors in C# -

* Compile Time Error
* Run Time Error

**22) Do we get error while executing “finally” block in C#?**

Yes. We may get error in finally block.

**23) Mention the assembly name where System namespace lies in C#?**

Assembly Name – mscorlib.dll

**24) What are the differences between static, public and void in C#?**

* Static classes/methods/variables are accessible throughout the application without creating instance. Compiler will store the method address as an entry point.
* Public methods or variables are accessible throughout the application.
* Void is used for the methods to indicate it will not return any value.

**Don't Miss** - [Database Interview Questions and Answers](http://a4academics.com/interview-questions/53-database-and-sql/411-sql-interview-questions-and-answers-database)

**25) What is the difference between “out” and “ref” parameters in C#?**

“out” parameter can be passed to a method and it need not be initialized where as “ref” parameter has to be initialized before it is used.

**26) Explain Jagged Arrays in C#?**

If the elements of an array is an array then it’s called as jagged array. The elements can be of different sizes and dimensions.

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

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

**28) What are value types in C#?**

Below are the list of value types in C# -

* decimal
* int
* byte
* enum
* double
* long
* float

**Must Read** - [SQL Query Interview Questions and Answers](http://a4academics.com/interview-questions/53-database-and-sql/397-top-100-database-sql-interview-questions-and-answers-examples-queries)

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

Below are the list of reference types in C# -

* class
* string
* interface
* object

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

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

**31) Explain access modifier – “protected internal” in C#?**

“protected internal” can be accessed in the same assembly and the child classes can also access these methods.

**32) 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.

**33) What you mean by inner exception in C#?**

Inner exception is a property of exception class which will give you a brief insight of the exception i.e, parent exception and child exception details.

**34) 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"

**35) What is the difference between “StringBuilder” and “String” in C#?**

* StringBuilder is mutable, which means once object for stringbuilder is created, it later be modified either using Append, Remove or Replace.
* String is immutable and it means we cannot modify the string object and will always create new object in memory of string type.

**36) What is the difference between methods – “System.Array.Clone()” and “System.Array.CopyTo()” in C#?**

* “CopyTo()” method can be used to copy the elements of one array to other.
* “Clone()” method is used to create a new array to contain all the elements which are in the original array.

**37) 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.

**Also Read** - [ASP.Net Interview Questions and Answers](http://a4academics.com/interview-questions/52-dot-net-interview-questions/706-asp-net)

**38) Explain circular reference in C#?**

This is a situation where in, multiple resources are dependent on each other and this causes a lock condition and this makes the resource to be unused.

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

Below are some of the exceptions in C# -

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

**40) 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.

**41) Explain object pool in C#?**

Object pool is used to track the objects which are being used in the code. So object pool reduces the object creation overhead.

**Also Read** - [ASP.Net MVC Interview Questions and Answers](http://a4academics.com/interview-questions/52-dot-net-interview-questions/713-asp-net-mvc)

**42) 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.

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

Below are the uses of delegates in C# -

* Single Delegate
* Multicast Delegate
* Generic Delegate

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

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

* Func
* Action
* Predicate

**45) 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.

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

Yes. We can use delegates for asynchronous method calls.

**47) 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

**48) What is Nullable Types in C#?**

Variable types does not hold null values so to hold the null values we have to use nullable types. So nullable types can have values either null or other values as well.

Eg: Int? mynullablevar = null;

**49) Why to use “Nullable Coalescing Operator” (??) in C#?**

Nullable Coalescing Operator can be used with reference types and nullable value types. So if the first operand of the expression is null then the value of second operand is assigned to the variable. For example,

double? myFirstno = null;  
double mySecno;  
mySecno = myFirstno ?? 10.11;

**50) What is the difference between “as” and “is” operators in C#?**

* “as” operator is used for casting object to type or class.
* “is” operator is used for checking the object with type and this will return a Boolean value.

51) 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);

**52) What is the difference between CType and Directcast in C#?**

* CType is used for conversion between type and the expression.
* Directcast is used for converting the object type which requires run time type to be the same as specified type.

**53) Is C# code is unmanaged or managed code?**

C# code is managed code because the compiler – CLR will compile the code to Intermediate Language.

**54) Why to use lock statement in C#?**

Lock will make sure one thread will not intercept the other thread which is running the part of code. So lock statement will make the thread wait, block till the object is being released.

**55) Explain Hashtable in C#?**

It is used to store the key/value pairs based on hash code of the key. Key will be used to access the element in the collection. For example,

Hashtable myHashtbl = new Hashtable();  
myHashtbl.Add("1", "TestValue1");  
myHashtbl.Add("2", "TestValue2");

**56) How to check whether hash table contains specific key in C#?**

Method – “ContainsKey” can be used to check the key in hash table. Below is the sample code for the same –

Eg: myHashtbl.ContainsKey("1");

**57) 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};

**58) Which are the loop types available in C#?**

Below are the loop types in C# -

For  
While  
Do.. While

**59) What is the difference between “continue” and “break” statements in C#?**

* “continue” statement is used to pass the control to next iteration. This statement can be used with – “while”, “for”, “foreach” loops.
* “break” statement is used to exit the loop.

**60) Write a sample code to write the contents to text file in C#?**

Below is the sample code to write the contents to text file –

Using System.IO;  
File.WriteAllText(”mytextfilePath”, “MyTestContent”);

**61) 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;

**62) 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.

**63) Explain Anonymous type in C#?**

This is being added in C# 3.0 version. This feature enables us to create an object at compile time. Below is the sample code for the same –

Var myTestCategory = new { CategoryId = 1, CategoryName = “Category1”};

**64) Name the compiler of C#?**

C# Compiler is – CSC.

**65) Explain the types of unit test cases?**

Below are the list of unit test case types –

* Positive Test cases
* Negative Test cases
* Exception Test cases

**66) Explain Copy constructor in C#?**

If the constructor contains the same class in the constructor parameter then it is called as copy constructor.

class MyClass  
{  
 public string prop1, prop2;  
 public MyClass(string a, string b)  
 {  
 prop1 = a;  
 prop2 = b;  
 }  
   
 public MyClass(MyClass myobj) // Copy Constructor  
 {  
 prop1 = myobj.prop1;  
 prop2 = myobj.prop2;  
 }  
}

67) 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;  
 }  
}

**68) Which string method is used for concatenation of two strings in c#?**

“Concat” method of String class is used to concatenate two strings. For example,

string.Concat(firstStr, secStr)

**69) 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;  
 }  
 }  
}

**70) What are the collection types can be used in C#?**

Below are the collection types in C# -

* ArrayList
* Stack
* Queue
* SortedList
* HashTable
* Bit Array

**71) Explain Attributes in C#?**

* Attributes are used to convey the info for runtime about the behavior of elements like – “methods”, “classes”, “enums” etc.
* Attributes can be used to add metadata like – comments, classes, compiler instruction etc.

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

Below are the predefined attributes in C# -

* Conditional
* Obsolete
* Attribute Usage

**73) 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.

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

Below are the states of thread –

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

**75) 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

**76) What is a class ?**  
  
A class is the generic definition of what an object is. A Class describes all the attributes of the object, as well as the methods that implement the behavior of the member object. In other words, class is a template of an object. For ease of understanding a class, we will look at an example. In the class Employee given below, Name and Salary are the attributes of the class Person. The Setter and Getter methods are used to store and fetch data from the variable.

public class Employee

{

private String name;

private String Salary;

public String getName()

{

return name;

}

public void setName(String name)

{

this.name = name;   
}

public String getSalary ()

{

return Salary;

}

public void setSalary (String Salary)

{

this. Salary = Salary;

}

}

**77) What is an Object?**

An object is an instance of a class. It contains real values instead of variables. For example, let us create an instance of the class Employee called “John”.

Employee John= new Employee();

Now we can access all the methods in the class “Employee” via object “John” as shown below.

John.setName(“XYZ”);

**78) 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.

**79) Explain Static Members in C# ?**

If an attribute's value had to be same across all the instances of the same class, the static keyword is used. For example, if the Minimum salary should be set for all employees in the employee class, use the following code.

private static double MinSalary = 30000;

To access a private or public attribute or method in a class, at first an object of the class should be created. Then by using the object instance of that class, attributes or methods can be accessed. To access a static variable, we don't want to create an instance of the class containing the static variable. We can directly refer that static variable as shown below.

double var = Employee.MinSalary ;

**80) What is Reference Type in C# ?**

Let us explain this with the help of an example. In the code given below,

Employee emp1;

Employee emp2 = new Employee();

emp1 = emp2;

Here emp2 has an object instance of Employee Class. But emp1 object is set as emp2. What this means is that the object emp2 is referred in emp1, rather than copying emp2 instance into emp1. When a change is made in emp2 object, corresponding changes can be seen in emp1 object.

**81) Define Property in C# ?**

Properties are a type of class member, that are exposed to the outside world as a pair of Methods. For example, for the static field Minsalary, we will Create a property as shown below.

private double minimumSalary;

public static double MinSalary

{

get

{

return minimumSalary;

}

set

{

minimumSalary = value;

}

}

So when we execute the following lines code

double minSal = Employee.MinSalary;

get Method will get triggered and value in minimumSalary field will be returned. When we execute,

Employee. MinSalary = 3000;

set Method will get triggered and value will be stored in minimumSalary field.

**82) Explain Overloading in C# ?**

When methods are created with the same name, but with different signature its called overloading. For example, WriteLine method in console class is an example of overloading. In the first instance, it takes one variable. In the second instance, “WriteLine” method takes two variable.

Console.WriteLine(x);

Console.WriteLine("The message is {0}", Message);

Different types of overloading in C# are

* Constructor overloading
* Function overloading
* Operator overloading

**83) What is Constructor Overloading in C# .net ?**

In Constructor overloading, n number of constructors can be created for the same class. But the signatures of each constructor should vary. For example

public class Employee

{

public Employee()

{ }

public Employee(String Name)

{ }

}

**84) What is Function Overloading in C# .net ?**

In Function overloading, n number of functions can be created for the same class. But the signatures of each function should vary. For example

public class Employee

{

public void Employee()

{ }

public void Employee(String Name)

{ }

}

**85) What is Operator Overloading in C# .net ?**

We had seen function overloading in the previous example. For operator Overloading, we will have a look at the example given below. We had defined a class rectangle with two operator overloading methods.

class Rectangle

{

private int Height;

private int Width;

public Rectangle(int w,int h)

{

Width=w;

Height=h;

}

public static bool operator >(Rectangle a,Rectangle b)

{

return a.Height > b.Height ;

}

public static bool operator <(Rectangle a,Rectangle b)

{

return a.Height < b.Height ;

}

}

Let us call the operator overloaded functions from the method given below. When first if condition is triggered, the first overloaded function in the rectangle class will be triggered. When second if condition is triggered, the second overloaded function in the rectangle class will be triggered.

public static void Main()

{

Rectangle obj1 =new Rectangle();

Rectangle obj2 =new Rectangle();

if(obj1 > obj2)

{

Console.WriteLine("Rectangle1 is greater than Rectangle2");

}

if(obj1 < obj2)

{

Console.WriteLine("Rectangle1 is less than Rectangle2");

}

}

**86) What is Data Encapsulation ?**

Data Encapsulation is defined as the process of hiding the important fields from the end user. In the above example, we had used getters and setters to set value for MinSalary. The idea behind this is that, private field “minimumSalary” is an important part of our classes. So if we give a third party code to have complete control over the field without any validation, it can adversely affect the functionality. This is inline with the OOPS Concept that an external user should know about the what an object does. How it does it, should be decided by the program. So if a user set a negative value for MinSalary, we can put a validation in the set method to avoid negative values as shown below

set

{

if(value > 0)

{

minSalary = value;

}

}

**87) Explain Inheritance in C# ?**

In object-oriented programming (OOP), inheritance is a way to reuse code of existing objects. In inheritance, there will be two classes - base class and derived classes. A class can inherit attributes and methods from existing class called base class or parent class. The class which inherits from a base class is called derived classes or child class. For more clarity on this topic, let us have a look at 2 classes shown below. Here Class Car is Base Class and Class Ford is derived class.

class Car

{

public Car()

{

Console.WriteLine("Base Class Car");

}

public 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 $");

}

}

When we execute following lines of code ,

Ford CarFord = new Ford();

CarFord.DriveType();

CarFord.Price();

Output Generated is as given below.

Base Class Car

Derived Class Ford

Right Hand Drive

Ford Price : 100K $

What this means is that, all the methods and attributes of Base Class car are available in Derived Class Ford. When an object of class Ford is created, constructors of the Base and Derived class get invoked. Even though there is no method called DriveType() in Class Ford, we are able to invoke the method because of inheriting Base Class methods to derived class.

**88) Can Multiple Inheritance implemented in C# ?**

In C#, derived classes can inherit from one base class only. If you want to inherit from multiple base classes, use interface.

**89) What is Polymorphism in C# ?**

The ability of a programming language to process objects in different ways depending on their data type or class is known as Polymorphism. There are two types of polymorphism

* Compile time polymorphism. Best example is Overloading
* Runtime polymorphism. Best example is Overriding

**90) 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

**91) What is overriding in c# ?**

To override a base class method which is defined as virtual, Override keyword is used. In the above example, method DriveType is overridden in the derived class.

**92) What is Method Hiding in C# ?**

If the derived class doesn't want to use methods in the base class, derived class can implement it's own version of the same method with same signature. For example, in the classes given below, DriveType() is implemented in the derived class with same signature. This is called Method Hiding.

class Car

{

public void DriveType()

{

Console.WriteLine("Right Hand Drive");

}

}

class Ford : Car

{

public void DriveType()

{

Console.WriteLine("Right Hand ");

}

}

**93) 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.

**94) 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 ");

}

}

**95) What is an Interface in C# ?**

An interface is similar to a class with method signatures. There wont be any implementation of the methods in an Interface. Classes which implement interface should have an implementation of methods defined in the abstract class.

**96) What is a Constructor in C# ?**

Constructor is a special method that get invoked/called automatically, whenever an object of a given class gets instantiated. In our class car, constructor is defined as shown below

public Car()

{

Console.WriteLine("Base Class Car");

}

When ever an instance of class car is created from the same class or its derived class(Except Few Scenarios), Constructor get called and sequence of code written in the constructor get executed.

interface Breaks

{

void BreakType();

}

interface Wheels

{

void WheelType();

}

class Ford : Breaks, Wheels

{

public Ford()

{

Console.WriteLine("Derived Class Ford");

}

public void Price()

{

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

}

public void BreakType()

{

Console.WriteLine("Power Break");

}

public void WheelType()

{

Console.WriteLine("Bridgestone");

}

}

**97) What is a Destructor in C# ?**

Destructor is a special method that get invoked/called automatically whenever an object of a given class gets destroyed. Main idea behind using destructor is to free the memory used by the object.

<http://a4academics.com/interview-questions/52-dot-net-interview-questions/417-c-oops-interview-questions-and-answers?showal>

**1. What is C#?**

C# is an object-oriented, type-safe, and managed language that is compiled by .Net framework to generate Microsoft Intermediate Language.

**2. Explain types of comment in C# with examples**

Single line

Example:

//This is a single line comment

ii. Multiple line (/\* \*/)

Example:

/\*This is a multiple line comment

We are in line 2

Last line of comment\*/

iii. XML Comments (///).

Eg:

/// summary;

/// Set error message for multilingual language.

/// summary

**3. Can multiple catch blocks be executed?**

No, Multiple catch blocks can't be executed. Once the proper catch code executed, the control is transferred to the finally block, and then the code that follows the finally block gets executed.

**4. What is the difference between public, static, and void?**

Public declared variables or methods are accessible anywhere in the application. Static declared variables or methods are globally accessible without creating an instance of the class. Static member are by default not globally accessible it depends upon the type of access modified used. The compiler stores the address of the method as the entry point and uses this information to begin execution before any objects are created. And Void is a type modifier that states that the method or variable does not return any value.

**5. What is an object?**

[](https://www.guru99.com/images/1/101818_0801_Top50CInter1.jpg)

An object is an instance of a class through which we access the methods of that class. "New" keyword is used to create an object. A class that creates an object in memory will contain the information about the methods, variables, and behavior of that class.

**6. Define Constructors**

A constructor is a member function in a class that has the same name as its class. The constructor is automatically invoked whenever an object class is created. It constructs the values of data members while initializing the class.

**7. What is Jagged Arrays?**

The Array which has elements of type array is called jagged Array. The elements can be of different dimensions and sizes. We can also call jagged Array as an Array of arrays.

**8. What is the difference between ref & out parameters?**

An argument passed as ref must be initialized before passing to the method whereas out parameter needs not to be initialized before passing to a method.

**9. What is the use of 'using' statement in C#?**

The 'using' block is used to obtain a resource and process it and then automatically dispose of when the execution of the block completed.

**10. What is serialization?**

When we want to transport an object through a network, then we have to convert the object into a stream of bytes. The process of converting an object into a stream of bytes is called Serialization. For an object to be serializable, it should implement ISerialize Interface. De-serialization is the reverse process of creating an object from a stream of bytes.

**11. Can we use "this" command within a static method?**

We can't use 'This' in a static method because we can only use static variables/methods in a static method.

**12. What is the difference between constants and read-only?**

Constant variables are declared and initialized at compile time. The value can't be changed afterward. Read-only is used only when we want to assign the value at run time.

**13. What is an interface class? Give one example of it**

An Interface is an abstract class which has only public abstract methods, and the methods only have the declaration and not the definition. These abstract methods must be implemented in the inherited classes.

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace DemoApplication

{

interface Guru99Interface

{

void SetTutorial(int pID, string pName);

String GetTutorial();

}

class Guru99Tutorial : Guru99Interface

{

protected int TutorialID;

protected string TutorialName;

public void SetTutorial(int pID, string pName)

{

TutorialID = pID;

TutorialName = pName;

}

public String GetTutorial()

{

return TutorialName;

}

static void Main(string[] args)

{

Guru99Tutorial pTutor = new Guru99Tutorial();

pTutor.SetTutorial(1,".Net by Guru99");

Console.WriteLine(pTutor.GetTutorial());

Console.ReadKey();

}

}

}

**14. What are value types and reference types?**

A value type holds a data value within its own memory space. Example

int a = 30;

Reference type stores the address of the Object where the value is being stored. It is a pointer to another memory location.

string b = "Hello Guru99!!";

**15. What are Custom Control and User Control?**

Custom Controls are controls generated as compiled code (Dlls), those are easier to use and can be added to toolbox. Developers can drag and drop controls to their web forms. Attributes can, at design time. We can easily add custom controls to Multiple Applications (If Shared Dlls). So, If they are private, then we can copy to dll to bin directory of web application and then add reference and can use them.

User Controls are very much similar to ASP include files, and are easy to create. User controls can't be placed in the toolbox and dragged - dropped from it. They have their design and code-behind. The file extension for user controls is ascx.

**16. What are sealed classes in C#?**

We create sealed classes when we want to restrict the class to be inherited. Sealed modifier used to prevent derivation from a class. If we forcefully specify a sealed class as base class, then a compile-time error occurs.

**17. What is method overloading?**

Method overloading is creating multiple methods with the same name with unique signatures in the same class. When we compile, the compiler uses overload resolution to determine the specific method to be invoke.

**18. What is the difference between Array and Arraylist?**

In an array, we can have items of the same type only. The size of the array is fixed when compared. To an arraylist is similar to an array, but it doesn't have a fixed size.

**19. Can a private virtual method can be overridden?**

No, because they are not accessible outside the class.

**20. Describe the accessibility modifier "protected internal".**

Protected Internal variables/methods are accessible within the same assembly and also from the classes that are derived from this parent class.

**21. What are the differences between System.String and System.Text.StringBuilder classes?**

System.String is immutable. When we modify the value of a string variable, then a new memory is allocated to the new value and the previous memory allocation released. System.StringBuilder was designed to have a concept of a mutable string where a variety of operations can be performed without allocation separate memory location for the modified string.

**22. What's the difference between the System.Array.CopyTo() and System.Array.Clone() ?**

Using Clone() method, we creates a new array object containing all the elements in the original Array and using CopyTo() method. All the elements of existing array copies into another existing array. Both methods perform a shallow copy.

**23. How can we sort the elements of the Array in descending order?**

Using Sort() methods followed by Reverse() method.

**24. Write down the C# syntax to catch an exception**

To catch an exception, we use try-catch blocks. Catch block can have a parameter of system.Exception type.

Eg:

try {

GetAllData();

}

catch (Exception ex) {

}

In the above example, we can omit the parameter from catch statement.

**25. What's the difference between an interface and abstract class?**

Interfaces have all the methods having only declaration but no definition. In an abstract class, we can have some concrete methods. In an interface class, all the methods are public. An abstract class may have private methods.

**26. What is the difference between Finalize() and Dispose() methods?**

Dispose() is called when we want for an object to release any unmanaged resources with them. On the other hand, Finalize() is used for the same purpose, but it doesn't assure the garbage collection of an object.

**27. What are circular references?**

Circular reference is situation in which two or more resources are interdependent on each other causes the lock condition and make the resources unusable.

**28. What are generics in C#.NET?**

Generics are used to make reusable code classes to decrease the code redundancy, increase type safety, and performance. Using generics, we can create collection classes. To create generic collection, System.Collections.Generic namespace should be used instead of classes such as ArrayList in the System.Collections namespace. Generics promotes the usage of parameterized types.

**29. What is an object pool in .NET?**

An object pool is a container having objects ready to be used. It tracks the object that is currently in use, total number of objects in the pool. This reduces the overhead of creating and re-creating objects.

**30. List down the commonly used types of exceptions in .net**

ArgumentException, ArgumentNullException , ArgumentOutOfRangeException, ArithmeticException, DivideByZeroException ,OverflowException , IndexOutOfRangeException ,InvalidCastException ,InvalidOperationException , IOEndOfStreamException , NullReferenceException , OutOfMemoryException , StackOverflowException etc.

**31. What are Custom Exceptions?**

Sometimes there are some errors that need to be handled as per user requirements. Custom exceptions are used for them and are used defined exceptions.

**32. What are delegates?**

Delegates are same are function pointers in C++, but the only difference is that they are type safe, unlike function pointers. Delegates are required because they can be used to write much more generic type-safe functions.

**33. How do you inherit a class into other class in C#?**

Colon is used as inheritance operator in C#. Just place a colon and then the class name.

public class DerivedClass : BaseClass

**34. What is the base class in .net from which all the classes are derived from?**

System.Object

**35. What is the difference between method overriding and method overloading?**

In method overriding, we change the method definition in the derived class that changes the method behavior. Method overloading is creating a method with the same name within the same class having different signatures.

**36. What are the different ways a method can be overloaded?**

Methods can be overloaded using different data types for a parameter, different order of parameters, and different number of parameters.

**37. Why can't you specify the accessibility modifier for methods inside the interface?**

In an interface, we have virtual methods that do not have method definition. All the methods are there to be overridden in the derived class. That's why they all are public.

**38. How can we set the class to be inherited, but prevent the method from being over-ridden?**

Declare the class as public and make the method sealed to prevent it from being overridden.

**39. What happens if the inherited interfaces have conflicting method names?**

Implement is up to you as the method is inside your own class. There might be a problem when the methods from different interfaces expect different data, but as far as compiler cares you're okay.

**40. What is the difference between a Struct and a Class?**

Structs are value-type variables, and classes are reference types. Structs stored on the Stack causes additional overhead but faster retrieval. Structs cannot be inherited.

**41. How to use nullable types in .Net?**

Value types can take either their normal values or a null value. Such types are called nullable types.

Int? someID = null;

If(someID.HasVAlue)

{

}

**42. How we can create an array with non-default values?**

We can create an array with non-default values using Enumerable.Repeat.

**43. What is difference between "is" and "as" operators in c#?**

"is" operator is used to check the compatibility of an object with a given type, and it returns the result as Boolean.

"as" operator is used for casting of an object to a type or a class.

**44. What's a multicast delegate?**

A delegate having multiple handlers assigned to it is called multicast delegate. Each handler is assigned to a method.

**45. What are indexers in C# .NET?**

Indexers are known as smart arrays in C#. It allows the instances of a class to be indexed in the same way as an array.

Eg:

public int this[int index] // Indexer declaration

**46. What is difference between the "throw" and "throw ex" in .NET?**

"Throw" statement preserves original error stack whereas "throw ex" have the stack trace from their throw point. It is always advised to use "throw" because it provides more accurate error information.

**47. What are C# attributes and its significance?**

C# provides developers a way to define declarative tags on certain entities, eg. Class, method, etc. are called attributes. The attribute's information can be retrieved at runtime using Reflection.

**48. How to implement a singleton design pattern in C#?**

In a singleton pattern, a class can only have one instance and provides an access point to it globally.

Eg:

Public sealed class Singleton

{

Private static readonly Singleton \_instance = new Singleton();

}

**49. What is the difference between directcast and ctype?**

DirectCast is used to convert the type of object that requires the run-time type to be the same as the specified type in DirectCast.

Ctype is used for conversion where the conversion is defined between the expression and the type.

**50. Is C# code is managed or unmanaged code?**

C# is managed code because Common language runtime can compile C# code to Intermediate language.

**51. What is Console application?**

A console application is an application that can be run in the command prompt in Windows. For any beginner on .Net, building a console application is ideally the first step, to begin with.

**52. Give an example of removing an element from the queue**

The dequeue method is used to remove an element from the queue.

using System;

using System.Collections;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace DemoApplication

{

class Program

{

static void Main(string[] args)

{

Queue qt = new Queue();

qt.Enqueue(1);

qt.Enqueue(2);

qt.Enqueue(3);

foreach (Object obj in qt)

{

Console.WriteLine(obj);

}

Console.WriteLine(); Console.WriteLine();

Console.WriteLine("The number of elements in the Queue " + qt.Count);

Console.WriteLine("Does the Queue contain " + qt.Contains(3));

Console.ReadKey();

}

}

}

|  |  |  |
| --- | --- | --- |
| **Question** | **Answer** |  |
| 1) What is stored in Heap? | reference type (reference type) | http://quizform.jotform.io/images/tick.png |
| 2) Select the control which can’t be placed in the toolbox | Custom Control (User control) | http://quizform.jotform.io/images/cross.png |
| 3) Select the class which is immutable | System.StringBuilder (System.String) | http://quizform.jotform.io/images/cross.png |
| 4) Select the method that doesn’t gaurantee the garbage collection of an object. | Finalize() (Finalize()) | http://quizform.jotform.io/images/tick.png |
| 5) Generics are used to make reusable code classes to | decrease the code redundancy (decrease the code redundancy) | http://quizform.jotform.io/images/tick.png |
| 6) Select the type of exceptions which is NOT used in .Net. | StackUnderflowException (StackUnderflowException) | http://quizform.jotform.io/images/tick.png |
| 7) What is used as inheritance operator in C#. | Colon (Colon) | http://quizform.jotform.io/images/tick.png |
| 8) Which is the base class in .NET from which all the classes are derived from? | System.Object (System.Object) | http://quizform.jotform.io/images/tick.png |
| 9) Can you specify accessibility modifier for methods inside an interface? | No (No) | http://quizform.jotform.io/images/tick.png |
| 10) Structs can be inherited? | No (No) | http://quizform.jotform.io/images/tick.png |
| 11) Select the operator used to check the compatibility of an object with a given type | "is" operater ("is" operater) | http://quizform.jotform.io/images/tick.png |
| 12) Select the operator which is used for casting of object to a type or a class. | "as" operater ("as" operater) | http://quizform.jotform.io/images/tick.png |
| 13) In c#, the attribute’s information for a Class can be retrieved at | Compile time (Runtime) | http://quizform.jotform.io/images/cross.png |
| 14) What is used in conversion where the conversion is defined between the expression and the type. | directcast (Ctype) | http://quizform.jotform.io/images/cross.png |
| 15) What is sealed Modifier | Prevents inheritance of a class (Prevents inheritance of a class) |  |

**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 types of comments in C#?**

Below are the types of comments in C# –

Single Line Comment Eg : //

Multiline Comments Eg: /\* \*/

XML Comments Eg : ///

**3.So what makes your code really object-oriented?**

In order to understand this, we must first analyze what benefits we can derive from OO. In order to do this, we must clearly understand its foundation. So given that C# is at its core object-oriented

**4.What is Cohesion?**

In OOPS we develop our code in modules. Each module has certain responsibilities. Cohesion shows how much a module responsibilities are strongly related.

Higher cohesion is always preferred. Higher cohesion benefits are:

1. Improves maintenance of modules
2. Increase reusability

**5.Why are strings in C# immutable?**

Immutable means string values cannot be changed once they have been created. Any modification to a string value results in a completely new string instance, thus an inefficient use of memory and extraneous garbage collection. The mutable System.Text.StringBuilder class should be used when string values will change.

**6.List out the differences between Array and Array List in C#?**

Array stores the values or elements of same data type but array list stores values of different data types.

Arrays will use the fixed length but array list does not uses fixed length like array.

**7.What are the fundamental principles of OO programming?**

As a developer, you might be tempted to answer that it comprises things like Encapsulation, Polymorphism, Abstraction, and Inheritance. Although this is true, it doesn’t really explain the fundamental core of what OO is and what its benefits are.

Principles are crucial but they are not the most important aspect of what OO actually is. What is really important is to understand in what grounds OO is built upon, or in other words, what are the foundations of OO programming.

The two most fundamental core concepts on which OO has been built upon in C# are *this* pointer and Dynamic Dispatch.

Obviously, there are principles like Encapsulation, Polymorphism, Abstraction, and Inheritance, but these are the consequence and not the generating force behind the OO paradigm in C#.

**8.What is coupling?**

OOPS Modules are dependent on each other. Coupling refers to level of dependency between two software modules.

Two modules are highly dependent on each other if you have changed in one module and for supporting that change every time you have to change in dependent module.

Loose Coupling is always preferred.

Inversion of Control and dependency injections are some techniques for getting loose coupling in modules.

**9. What is the execution entry point for a C# console application?**

The Main method.

**10. Why to use “using” in C#?**

“Using” statement calls – “dispose” method internally, whenever any exception occurred in any method call and in “Using” statement objects are read only and cannot be reassignable or modifiable.

**11. What is the this Pointer?**

The this pointer is silently passed with a call to an instance-level function, which then operates on an object (instance of a class).

Basically, this core mechanism makes it possible to bring operations close to data. It also eliminates the need to have global functions and it gives data structures the intrinsic ability to perform operations on its data.

**12.What is Abstraction?**

Abstraction is a technique of taking something specific and making it less specific.

In OOPS we achieve the abstraction by separating the implementation from interface. We take a implemented class and took only those method signatures and properties which are required by the class client. We put these method signatures and properties into interface or abstract class.

**13. How do you initiate a string without escaping each backslash?**

You put an @ sign in front of the double-quoted string.

String ex = @”This has a carriage return\r\n”

**14.Explain namespaces in C#?**

Namespaces are containers for the classes. We will use namespaces for grouping the related classes in C#. “Using” keyword can be used for using the namespace in other namespace.

**15. What is the OO fundamental idea using C# that allows a data structure to perform operations on its own data?**

What would your answer be? Pretty obvious. The humble this pointer.

Notice that despite this being a mind-bending idea, we can already start to appreciate the bigger picture for which C# was designed.

The this pointer is basically a way for a data structure (object) to be able to access methods that allow itself to perform operations on its own data. It is a way to manage state within a data structure.

Now let’s talk a bit about the other core concept that takes this to the next level.

**16.What is Encapsulation?**

In non object oriented languages, data and behaviors are not tied together. That means any function in the program can modify the data.

In Encapsulation, we bind the data and behaviors in one object. Only defined behaviors in a class can modify the data. We hide the state of an object by using properties and methods. Clients of the object only see these behaviors and by only these behaviors clients can modify the data.

We also protect the data by using access specifiers. We put the private / protected keywords before data to protect it from the outside world.

**17. What is the difference between a struct and a class?**

Structs cannot be inherited. Structs are passed by value and not by reference. Structs are stored on the stack not the heap. The result is better performance with Structs.

**18.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.

**19. What is a singleton?**

A singleton is a design pattern used when only one instance of an object is created and shared; that is, it only allows one instance of itself to be created. Any attempt to create another instance simply returns a reference to the first one. Singleton classes are created by defining all class constructors as private. In addition, a private static member is created as the same type of the class, along with a public static member that returns an instance of the class. Here is a basic example:

public class SingletonExample { private static SingletonExample \_Instance; private SingletonExample () { } public static SingletonExample Get Instance() {  if (\_Instance == null)  {    \_Instance = new SingletonExample ();   }   return \_Instance;  }}

**20.Why to use “finally” block in C#?**

“Finally” block will be executed irrespective of exception. So while executing the code in try block when exception is occurred, control is returned to catch block and at last “finally” block will be executed. So closing connection to database / releasing the file handlers can be kept in “finally” block.

**21.What is boxing?**

Boxing is the process of explicitly converting a value type into a corresponding reference type. Basically, this involves creating a new object on the heap and placing the value there. Reversing the process is just as easy with unboxing, which converts the value in an object reference on the heap into a corresponding value type on the stack. The unboxing process begins by verifying that the recipient value type is equivalent to the boxed type. If the operation is permitted, the value is copied to the stack

**22.Can we have only “try” block without “catch” block in C#?**

Yes we can have only try block without catch block.

**23. How to move to a State-related Codebase?**

So now that we’ve explored these concepts, let’s move to a state-related code base. So you might be asking yourself at this moment

**24.What is the difference between “out” and “ref” parameters in C#?**

“out” parameter can be passed to a method and it need not be initialized where as “ref” parameter has to be initialized before it is used.

**25.How are methods overloaded?**

Methods are overloaded via different signatures (number of parameters and types). Thus, you can overload a method by having different data types, different number of parameters, or a different order of parameters.

**26.Explain Jagged Arrays in C#?**

If the elements of an array is an array then it’s called as jagged array. The elements can be of different sizes and dimensions.

**27.How do you prevent a class from being inherited?**

The sealed keyword prohibits a class from being inherited.

**28.What you mean by inner exception in C#?**

Inner exception is a property of exception class which will give you a brief insight of the exception i.e, parent exception and child exception details.

**29.What is the GAC, and where is it located?**

The GAC is the Global Assembly Cache. Shared assemblies reside in the GAC; this allows applications to share assemblies instead of having the assembly distributed with each application. Versioning allows multiple assembly versions to exist in the GAC—applications can specify version numbers in the config file. The gacutil command line tool is used to manage the GAC.

**30**.**Explain circular reference in C#?**

This is a situation where in, multiple resources are dependent on each other and this causes a lock condition and this makes the resource to be unused.

<https://www.mytectra.com/interview-question/c-interview-questions-and-answers-for-5-years-experienced/>

class Program {

static String location;

static DateTime time;

static void Main() {

Console.WriteLine(location == null ? "location is null" : location);

Console.WriteLine(time == null ? "time is null" : time.ToString());

}

}

The output will be:

location is null

1/1/0001 12:00:00 AM

Although both variables are uninitialized, String is a reference type and DateTime is a value type. As a value type, an unitialized DateTime variable is set to a default value of midnight of 1/1/1 (yup, that’s the year 1 A.D.), *not* null.

Given an array of ints, write a C# method to total all the values that are even numbers.

static long TotalAllEvenNumbers(int[] intArray) {

return intArray.Where(i => i % 2 == 0).Sum(i => (long)i);

}

Difference between Thread.Sleep and Task.await function in c# ?