C# is a simple, modern and a general-purpose Object-oriented programming language developed by Microsoft within its .NET framework.

# Beginner's Level

## Q1. List down the differences between Public, Static and void keywords?

**Ans-**

| Keyword | Description |
|---|---|
| Public | It is an access specifier which states that the method of a class can be accessed publicly |
| Static | It is a keyword used for declaring a member of a type, specific to that type |
| void | It states that the method does not return any value |

```
using System;
class test
{
static void main(String args[])// main is the method that gets executed first
{
Console.Writeline("Welcome to the world of C#");
}
}
```

## Q2. Define C# and list the features.

**Ans-** C# is an object-oriented, typed safe and manage language that is compiled by .NET framework and was developed by Microsoft back in 2000. The idea was that it will be used in the development of all kinds of software aiming at different platforms like Windows, Web, mobile using only one programming language. Fast forward to the present day it is one of the most popular programming languages worldwide, millions of developers use this language to build all kinds of software.

Some of the features of C#(sharp) are:

- Use of Constructors and Destructors
- Easy to grasp
- General Purpose and Object-Oriented
- Structured language
- Platform independent for compilation
- Part of .NET framework

Let's move on to the next question,

## Q3. List down the reason behind the usage of C# language.

**Ans-** There are several reasons for the usage of C# as a programming platform. Some of them are listed below.

- Easy to pickup
- Component oriented language
- Follows a Structured approach
- Produces readable and efficient programmes
- Once written can be compiled on different platforms
- Passing parameters is easy

## Q4. What are the advantages of using C#?

**Ans-** Following are the advantages of using C#:

- Easy to learn
- Object-Oriented language
- The syntax is easy to grasp
- Component oriented
- Part of the .NET framework

## Q5. What are the different types of comments in C#?

**Ans-** Below listed are the types of comments followed in C#:

### a. Single line comments

```
// hello, this is a single line comment
```

### b. Multiline comments

```
/* Hello this is a multiline comment
 last line of comment*/
```

**c. XML comments**

```
/// Hello this is XML comment
```

## Q6. Illustrate the process of code compilation in C#?

**Ans-** There exist four steps in the process of code compilation:

a. Compilation of Source code in managed code
b. Clubbing the newly created code into assembly
c. Loading the CLR (Common Language Runtime)
d. Execution of assembly through CLR

## Q7. List down the access modifiers available in C#.

**Ans-** Following are the access modifiers available for general use:

- **Public-** When an attribute or method is defined as public it can be accessed from any part of code.
- **Private-** A private attribute or method can be accessed from within the class itself.
- **Protected-** When a user defines a method or attribute as protected then it can be accessed only within that class and the one inheriting the class.
- **Internal-** When an attribute or method is defined as internal then it will be accessed from that class at the current assembly position.
- **Protected Internal-** When you define an attribute or method as protected internal, then it's access restricted to classes within the current project assembly or different types defined by that class.

Moving ahead in C# Interview questions, let's unfold some functionalities used in C#.

## Q8. List down the different IDE's Provided by Microsoft for C# Development.

**Ans-** Below listed are few IDE's for C# development:

- Visual Studio Express (VCE)
- Visual Studio (VS)
- Visual Web Developer
- MonoDevelop
- browxy

## Q9. Distinguish between Continue and Break Statement?

**Ans-** Using break statement you can 'jump out' of the loop whereas while making use of continue statement you can jump over an iteration and continue the loop execution.

**Example (Break Statement):**

```
using System;
using System.Collections;
using System.Linq;
using System.Text;

namespace break_example{
Class break_Stmnt{
public static void main(String args[]){
for(int i=0;i<=6;i++)
{
if(i==5)
{
break;
}
Console.Readline("The number is +i");
}
}
}
```

**Output:**

the number is 0;
the number is 1;
the number is 2;
the number is 3;
the number is 4;

**Example (Continue Statement):**

```
using System;
using System.Collections;
using System.Linq;
using System.Text;

namespace continue_example{
Class continue_Stmnt{
public static void main(String args[]){
for(int i=0;i<=6;i++)
{
if(i==5)
{
continue;
}
```

```
14Console.Readline("The number is +i");
15}
16}
17}
18
```

**Output:**

the number is 0;
the number is 1;
the number is 2;
the number is 3;
the number is 5;

### Q10. What are the different approaches of passing parameters to a method?

**Ans-** There are three ways of passing parameters to a method:

**Value Parameters-** Under this method the actual value of an argument is copied to the formal parameter of the function. In, this case the changes made into the formal parameter of the function have no effect on the actual value of the argument.

**Reference Parameters-** This method copies the argument referring to the memory location into the formal parameter. Meaning changes made to the parameter affect the argument.

**Output Parameters-** This method returns more than one value.

Let's move ahead in C# Interview Questions and see the next category.

## Intermediate level C# Interview Questions

### Q11. Distinguish between finally and finalize blocks?

**Ans-** finally block is called after the execution of *try* and catch *blocks,* It is used for exception handling whether or not the exception has been caught this block of code gets executed. Generally, this block of code has a cleaner code.

The finalize method is called just before the garbage collection. Main priorities are to perform clean up operation for unmanaged code, it is automatically invoked when an instance is not subsequently called.

### Q12. What is Managed or Unmanaged Code?

**Ans-** Managed code is one which is executed by CLR (Common Language Runtime) in simple terms it means all the application code is dependent on the .NET platform, considered as overseen in view of them. Unmanaged code is any code that is executed by runtime application of some other structure different from .NET platform. The runtime of application will deal with memory, security and other execution activities.

## Q13. What is an Object?

**Ans-** Object is an instance of a class, It is used for accessing the methods of a class. "New" keyword is used for the creation of an object, a class that creates an object in the memory location contains the information about methods, variables, and behavior of that class.

To know more about Object in Java, Python, and C++ you can go through the following blog:

- Java Object
- Python Object
- C++ Object

## Q14. What is a Class?

**Ans-** A class is a blueprint of an object. It defines different kinds of data and functionalities objects will have. A class enables you to create your own custom types by clubbing together variables of different types, methods, and events. In C# a class is defined by a class keyword.

To know more about Class in Java, Python, and C++ you can go through the following blog:

- Java Class
- Python Class
- C++ Class

## Q15. Define an abstract class?

**Ans-** It is a type of class whose objects can't be instantiated, It contains something like a single approach or technique and indicated by the keyword 'abstract'.

To know more about  abstract class in Java you can go through the following blog:

- Java abstract class

## Q16. Define sealed classes in C#?

**Ans-** You create sealed classes in situations when you want to restrict the class to be inherited. For doing this sealed modifiers are used. If you forcefully specify a sealed class as a base class then a compilation error occurs.

## Q17. Define a Partial class?

**Ans-** A partial class is the only one that essentially splits the definition of a class into multiple classes in either same source code files or multiple source code files. One can create a class definition in multiple files but that is compiled as one class at run-time and when an instance of this class is created, one can access all methods from every source file with the same object. It is indicated by the keyword 'partial'.

## Q18. List down the fundamental OOP concepts?

**Ans-** There are four fundamental OOP (Object Oriented Programming) concept they are listed as follows:

- **Inheritance-** Ever heard of this dialogue from relatives "you look exactly like your father/mother" the reason behind this is called 'inheritance'. From the Programming aspect, It generally means "inheriting or transfer of characteristics from parent to child class without any modification". The new class is called the **derived/child** class and the one from which it is derived is called a **parent/base** class.
- **Polymorphism-** You all must have used GPS for navigating the route, Isn't it amazing how many different routes you come across for the same destination depending on the traffic, from a programming point of view this is called 'polymorphism'. It is one such OOP methodology where one task can be performed in several different ways. To put it in simple words, *it is a property of an object which allows it to take multiple forms*.
- **Encapsulation-** In a raw form, encapsulation basically means binding up of data in a single class.A class shouldn't be directly accessed but be prefixed in an underscore.
- **Abstraction-** Suppose you booked a movie ticket from bookmyshow using net banking or any other process. You don't know the procedure of how the pin is generated or how the verification is done. This is called 'abstraction' from the programming aspect, it basically means you only show the implementation details of a particular process and hide the details from the user. It is used to simplify complex problems by modeling classes appropriate to the problem.An abstract class cannot be instantiated

which simply means you cannot create objects for this type of class. It can only be used for inheriting the functionalities.

To know more about OOPs in Java, Python, and C++ you can go through the following blogs:

- Java OOPs
- Python OOPs
- C++ OOPs

## Q19. Explain the process of inheriting a class into another class?

**Ans-** Colon is used as an inheritance operator in C#. Place the colon and the class name.

```
public class Derivedclass: childclass
```

## Q20. Define method overloading in C#?

**Ans-** Method overloading essentially is creating multiple methods with the same name and unique signatures with the same class. When you compile, the compiler makes use of overload resolution to determine the specific method which will be invoked.

## Q21. List the differences between method overriding and method overloading?

**Ans-** Under Method overriding the definition of the derived class is changed which in turn changes the method behavior. Method overloading is simply creating a method with the same name under the same class under different signatures.

## Q22. Explain StreamReader/StreamWriter class?

**Ans-** Streamreader and StreamWriter are classes of namespace.System.IO. Used when we want to read or write charact90, Reader based data, respectively.

members of StreamReader are: **close()**, **Read()**, **Readline()**. members of Streamwriter are: **close()**, **write()**, **writeline()**.

**Example program:**

```
Class Testprogram
{
using(StreamReader sr = new StreamReader("C:Readme.txt")
```

```
{
// Any code to read//
}
using(StreamWriter sr = ndew StreamWriter("C:Readme.txt")
{
// Any code to write//
}
}
```

## Q23. What is an Interface?

**Ans-** An Interface is basically a class with no implementation. It contains only the declaration of properties, attributes, and events.

## Q24. Distinguish between a class and struct?

**Ans-** Given below are the differences between a class and struct:

| Class | Struct |
|---|---|
| It supports inheritance | It does not support inheritance |
| It is Pass by reference | It is pass by value |
| Members are private by default | Members are public by default |
| Considered good for larger complex objects | Considered good for small models |

## Q25. List the difference between the Virtual method and the Abstract method?

**Ans-** A **Virtual method** must always have a default implementation. However, it can be overridden in a derived class by the keyword *override.*

An **Abstract method** doesn't have any implementation. It resides in the abstract class, and also it is mandatory that the derived class must implement abstract class. Use of override keyword is not necessary.

## Q26. Illustrate Namespaces in C#?

**Ans-** Namespaces are used for organizing large code projects. "System" is one of the most popular and widely used namespaces in C#. One can create their own namespace and use one into another called nesting.

## Q27. Define using statement in C#?

**Ans-** "Using" keyword simply denotes that the particular namespace is being used by the program. For ex- *using System*, Here System is a namespace. The class

console is defined under system, so we can use the Console.Writeline(...) or Readline in our program.

## Q28. Define an Escape Sequence, Name few strings in Escape Sequence?

**Ans-** An Escape Sequence is denoted by a backslash (). The backslash merely indicates that the character it is following should be interpreted literally or is a special character. An escape sequence is a single character.

Few escape sequences are as follows:

n – newline character

b – backspace

– backlash

‘ – Single quote

” – Double quote

## Q29. Define Boxing and Unboxing in C#?

**Ans-** Converting a value type to the reference type is called boxing.

**Example:**

```
int value = 10
//-------Boxing--------//
object boxedvalue = value1;
```

Explicit conversion of the same reference type i.e. converting it back to the value type is called Unboxing.

**Example:**

```
//----------------Unboxing-------// int UnBoxing = int(boxedvalue);
```

To know more about Boxing and Unboxing in Java you can go refer this article- Boxing in Java.

**Q30. Define an array?**

**Ans-** An array is used to store multiple variables of the same type. Collection of variables stored in a contiguous memory location.

**Example:**
double numbers = new double[10];
int[] Score = new int[4] {25,24,23,22,21,20}

Above given example above is of a Single Dimensional array. It is a linear array where values are stored in a single row. A multidimensional array is the one having more than one dimension. A good example would be of a rectangle.

**Example:**

int[,] numbers = new int[4,3]{ {1,2} {2,3} {3,4} {4,5} };

To know more about arrays in Java, Python , and C++ you can go through the following blogs:

- Python array
- Java array
- C++ array

## Q31. Define a Jagged Array in C#?

**Ans-** A Jagged array is referred to as an "array of arrays". It is an array whose elements are arrays, the element of the same can be of different dimensions and sizes. The length of each array index can differ.

**Example:**

int[][] jagArray = new int[5][];

## Q32. Distinguish between Array and Arraylist in C#?

**Ans-**

| Array | ArrayList |
|---|---|
| Array uses the vector array for storing the elements | ArrayList uses the LinkedList to store the elements |
| Size of the Array must be defined until redim used(vb) | There's no need for specifying storage size. |
| An array is a specific data type storage | ArrayList can store everything as an object |
| Typecasting is not necessary | Typecasting is necessary |
| There is no RunTime exception | There is a RunTime error exception |

| | |
|---|---|
| Elements can't be inserted or deleted in between | Elements can be inserted or deleted in between |

### Q33. Define Collections?

**Ans-** A collection essentially works like a container for instances of other classes. Every class implements Collection Interface.

To know more about collections in Java, check out the following blog:

- Java Collections

### Q34. Write a short note on Interface?

**Ans-** An Interface is a class with no implementation. It consists of a declaration of methods, Parameters, and values.

To know more about Interface in Java, check out the following blog:

- Java Interface

Moving on in C# Interview Question let's see the next category.

## Advance Level

### Q35. Illustrate Serialization?

**Ans-** A process that involves converting some code into its binary format is known as a serialization in C#. In doing so it gives the flexibility where code can be stored easily and or written to a disk or some other storage device. Serialization is used when there is a strict need for not losing the original code.

Serialization in C# is of three types:

**Binary Serialization-** It is fast and demands less space, converts any code into its binary format. Serialize and restore public and non-public properties.

**SOAP-** Generates a complete SOAP compliant envelope used by any system by its ability to understand SOAP. Classes under this type of Serialization reside in *System.Runtime.Serialization.*

**XML Serialization-** It serializes all the public properties to XML document. Readability being one factor, an XML document can also be manipulated in various ways. Classes under this type reside in *System.sml.Serialization.*

To know more about Serialization in Java, check out the following blog:

- Java Serialization

## Q36. Define Parsing? Explain how to Parse a DateTime String?

**Ans-** Parsing is a method of converting a string into another data type.

**Example:**

```
string text = "200";
int num = int.Parse(text);
```

In the above-given example, 200 is an integer. So, the Parse method converts the string 200 into its own base type, i.e. int.
To parse a DateTime String let's see a small example.

**Example:**

```
string dateTime = "Aug 26,2019"; Datetime parsedvalue = Datetime.Parse(dateTime);
```

## Q37. Define Delegate?

**Ans-** Delegate is a variable that holds the reference to a method. It is a function pointer of the reference type. All Delegates are derived from the *System.Delegate* namespace. Both Delegate and the method that it refers to can have the same signature. Let's see a small example.

**Example:**

```

public Delegate int myDel(int number); //declaring a delegate
public class Program
{
public int SubstractNumbers(int a) //Class Program has the method same signature as delegate
{
int difference = a - 10;
return difference;
}
public void start()
{
myDel DelegateExample = SubstractNumbers;
}
}
```

## Q38. Distinguish between System.String and System.Text.StringBuilder classes?

**Ans –** System.String is immutable. Whenever you modify the value of a string variable then a new memory is allocated to the new value and previous memory

allocation is released. System.Text.StringBuilder is mutable i.e. supports the fact that a variety of operations can be performed without allocating a seperate memory location for the already modified string.

**Q39. Illustrate the differences between the System.Array.CopyTo() and System.Array.Clone()?**

**Ans –** Using the Clone() method, a new array object is created containing all elements of the original array. Using the CopyTo() method all the elements of the existing array gets copied into another existing array. Both use a shallow copy method.

## Q40. Write the Syntax for catching an exception in C#?

**Ans –** To catch an Exception we make use of try-catch blocks. The catch block has a parameter of the system.Exception type.

**Example:**

```
try
{
GetAllData();
}
catch(Exception ex){
}
```

## Q41. Explain about generics in C#.NET?

**Ans-** Generics are used to make reusable code classes which decrease the code redundancy,

- Increase type safety, performance, and optimization
- Using Generics one can do a variety of things like create collections
- To create Generic collection, System.namespace
- The generic namespace should be used inspite of classes such as ArrayList in the System
- Generics instigates the usage of a parameterized type

## Q42. List down the differences between “dispose” and “finalize” methods in C#.

**Ans-** Dispose() is called when we want an object to release unmanaged resources with them. Whereas, Finalize() is used for the same purpose but it doesn’t assure garbage collection of an object.

## Q43. Define C# I/O classes? List the commonly used classes?

**Ans-** C# consists of System.IO namespace which has classes that compute and perform various operations on files like creating, deleting, opening and closing etc.

Few commonly used I/O classes are listed below:

**File-** Helps in manipulating file.
**StreamWriter-** Generally used for writting characters to a stream.
**StreamReader-** Generally used for reading characters to a stream.
**StringWriter-** Used for writing a string buffer.
**Stringreader-** Used for reading a string buffer.

## Q44. Define thread? Explain about Multithreading?

**Ans-** Thread is a set of instructions that when executed enables the program to perform concurrent processing. Concurrent processing helps in doing more than one process at a time. By default, C# consists of only thread. Other threads can be created to execute the code in parallel with original thread.

Thread follows a life cycle where it starts whenever a thread is created and gets terminated immediately after the execution. The namespace it follows is *System.Threading* which has to be included to create threads and use its members.

Threads are created by extending the thread class. Start() method marks the beginning of the thread execution.

```
1//callThread is our target method//
2ThreadStart methodThread = new ThreadStart(CallThread);
3Thread childThread = new Thread(methodThread);
4childThread.start();
```

C# can also execute more than proceeses/task at a time which is done by handling different processes at different time labeled as “multithreading”. Several operations of the same are listed below:

- **Start**
- **Sleep**
- **Abort**
- **Suspend**
- **Resume**
- **Join**

## Q45. What are Events?

**Ans-** Events in C# follow a concept where it consists of a Publisher, Subscriber, Notification and a handler. You can think of an event as nothing but an encapsulated delegate.

**Example:**

```
public Delegate void TestEvent();
public TestEvent TestEvent1;
```

## Q46. Explain Synchronous and Asynchronous Operations?

**Ans-** Synchronization is a way of creating a thread-safe code where only a single thread will access the code in a given time. A synchronous call waits for completion of method and then continous the program flow. Synchronous programming adversely affects the UI operations that normally happens when user tries to perform time-consuming operations since only one thread is used.

In Asynchronous operation, the method call immediately returns allowing the program to perform other operations while the method called completes its share of work in certain circumstances.

## Q47. Explain Async and Await?

**Ans-** Async and Await keywords are mostly used for creating asynchronous methods in C#. Usage of the same is shown through an example:

```

public async Task>CalculateCount()
{
await Task.Delay(2000);
return 1;
}
public async task mytestmethod()
{
Task> count = CalculateCount();
int result = await count;
}
```

In the above-given code async keyword is used for method declaration. The Count is of a task type int which calls the method CalculateCount(). CalculateCount() starts execution and calculates something.

## Q48. Explain Deadlock?

**Ans-** A deadlock is a situation that arises when a process isn't able to complete it's execution because two or more than two processes are waiting for each other to

finish. This usually occurs in multi-threading. In this, a shared resource is being held up by a process and another process is waiting for the first process to get over or release it, and the thread holding the locked item is waiting for another process to complete.

## Q49. Illustrate Race Condition?

**Ans-** A Race Condition occurs in a situation when two threads access the same resource and try to change it at the same time. The thread which accesses the resource first cannot be predicted. Let me take a small example where two threads X1 and X2 are trying to access the same shared resource called T. And if both threads try to write the value to T, then the last value written to T will be saved.

## Q50. What is Thread Pooling?

**Ans-** A Thread pool is a collection of threads that perform tasks without disturbing the primary thread. Once the task is completed by a thread it returns to the primary thread.

This brings us to the end of this article on C# Interview Questions. I hope it helped in adding up to your knowledge. Wishing you all the best for your interview. Happy learning.

In case you are looking for the most common Technical Interview Questions, read along:

- OOPs Interview Questions
- MVC Interview Questions
- SQL Interview Questions
- Java Interview Questions
- Python Interview Questions

*Check out the online training by Edureka, a trusted online learning company with a network of more than 250,000 satisfied learners spread across the globe. We are here to help you with every step on your journey through our online certification training.*

Got a question for us? Please mention it in the comments section of this “C# Interview Questions” article and we will get back to you as soon as possible.