**1. What is C#?**

C# is an [object-oriented programming language](https://www.simplilearn.com/tutorials/java-tutorial/oops-interview-questions) compiled by the .Net framework to generate Microsoft Intermediate Language.

**2. What is an object?**

An object is a class instance that can be used to access class methods. The "New" keyword can be used to construct an object.

**3. Define Constructors.**

A [constructor](https://www.simplilearn.com/tutorials/c-sharp-tutorial/c-sharp-constructor) is a member function with the same name as its class. The constructor is automatically invoked when an object is created.

**4. What is the difference between out and ref parameters?**

When an argument is passed as a ref, it must be initialized before it can be passed to the method. An out parameter, on the other hand, need not to be initialized before passing to a method.

**5. What is the benefit of ‘using’ statement in C#?**

The ‘using’ statement can be used in order to obtain a resource for processing and manage resources automatically, ensuring they are properly disposed of once they are no longer needed.

**6. Differentiate between Break and Continue Statement.**

Continue statement - Used in jumping over a particular iteration and getting into the next iteration of the[loop.](https://www.simplilearn.com/tutorials/asp-dot-net-tutorial/for-each-loop)

Break statement - Used to skip the next statements of the current iteration and come out of the loop.

**7. Discuss the various methods to pass parameters in a method.**

The various methods of passing parameters in a method include -

* **Value parameters**: The formal value copies and stores the value of the actual argument, which enables the manipulation of the formal parameter without affecting the value of the actual parameter.
* **Reference parameters**: The memory address of the actual parameter is stored in the formal argument, which means any change to the formal parameter would reflect on the actual argument too.
* **Output parameters**: Lets the method return more than one value.

**8. Name all the C# access modifiers.**

The C# access modifiers are -

* **Private Access Modifier** - A private attribute or method is one that can only be accessed from within the class.
* **Protected Access Modifier** - When a user declares a method or attribute as protected, it can only be accessed by members of that class and those who inherit it
* **Internal Access Modifier** - When a property or method is defined as internal, it can only be accessible from the current assembly point of that class.
* **Public Access Modifier** - When an attribute or method is declared public, it can be accessed from anywhere in the code.

**9. Why do we use C# language?**

Below are the reasons why we use the C# language -

* C# is a component-oriented language.

**Definition**: COP is a higher-level paradigm built on OOP principles, focusing on building software by assembling **reusable components**. A component is a self-contained, modular, and deployable piece of functionality with a clearly defined interface.

* C# is an object-oriented language.

**Definition**: OOP is a programming paradigm that organizes software design around **objects**—instances of classes that encapsulate data and behaviour.

* The C# language can be compiled on many platforms.
* The C# language follows a structured approach.
* The C# language produces really efficient and readable programmes.

**10. What is meant by Unmanaged or Managed Code?**

**Managed code** is the code that runs under the control of the **Common Language Runtime (CLR)** in the .NET environment. The CLR manages various aspects of code execution, such as memory allocation, garbage collection, security, and type safety.

**Unmanaged code** is the code that runs directly on the operating system, outside the control of the CLR. It relies on the programmer to manage memory and other resources manually.

**11. What is meant by an abstract class?**

An **abstract class** in C# is a class that **cannot be instantiated directly** and is designed to be a base class for other classes. It serves as a blueprint and can contain **abstract members** (methods, properties, events, or indexers without implementation) as well as fully implemented members. The purpose of an abstract class is to provide a common definition of a base class that multiple derived classes can share.