1. **What is C#?**
   * Answer: C# is an object-oriented programming language developed by Microsoft. It runs on the .NET framework and is used for building various types of applications, such as web, desktop, and mobile applications.
2. **What are the different types of data types in C#?**
   * Answer: In C#, data types are categorized into two types:
     + **Value types**: int, char, float, double, bool, etc.
     + **Reference types**: string, arrays, classes, interfaces, etc.
3. **What is a namespace in C#?**
   * Answer: A namespace is used to organize code and prevent name conflicts by grouping classes, interfaces, and methods. It provides a way to declare a scope where identifiers (like class names) are defined.
4. **What is the difference between Array and ArrayList in C#?**
   * Answer:
     + **Array**: A fixed-size collection of items that must all be of the same type.
     + **ArrayList**: A dynamic-sized collection that can store elements of different types, but it is part of the non-generic collections.
5. **What is a delegate in C#?**
   * Answer: A delegate is a type-safe function pointer in C#. It holds a reference to a method and can be used to call the method. Delegates are essential for event handling and callback functions.
6. **Explain the concept of LINQ in C#.**
   * Answer: LINQ (Language Integrated Query) is a query syntax in C# used to query collections such as arrays, lists, XML, databases, etc. It provides a more readable and concise syntax for performing operations like filtering, sorting, and grouping data.
7. **What is async and await in C#?**
   * Answer: async and await are used for asynchronous programming in C#. The async keyword is used to declare a method as asynchronous, and the await keyword is used to wait for the completion of an asynchronous task.
8. **What is the difference between == and .Equals() in C#?**
   * Answer:
     + ==: Compares the reference or value depending on whether it is used with reference types or value types.
     + .Equals(): Compares the contents or values of objects.

**OOP (Object-Oriented Programming) Interview Questions**

1. **What is Object-Oriented Programming (OOP)?**
   * Answer: OOP is a programming paradigm that uses the concept of objects and classes. It aims to bind data and methods that operate on the data within the same structure (objects) to increase code reusability and scalability. The four pillars of OOP are encapsulation, inheritance, polymorphism, and abstraction.
2. **Explain the concept of Class and Object.**
   * Answer:
     + **Class**: A blueprint or template that defines the attributes (fields) and behaviors (methods) of an object.
     + **Object**: An instance of a class. It contains data and methods to manipulate that data.
3. **What is Encapsulation?**
   * Answer: Encapsulation is the process of wrapping data (variables) and code (methods) together into a single unit, called a class. It restricts direct access to some of the object’s components, which is achieved by using access modifiers like private, protected, and public.
4. **What is Inheritance in OOP?**
   * Answer: Inheritance allows a new class (child class) to inherit properties and methods from an existing class (parent class). It enables code reusability and a hierarchical classification of classes.
5. **What is Polymorphism? Explain its types.**
   * Answer: Polymorphism allows methods to perform different tasks based on the object that invokes them. It has two types:
     + **Compile-time polymorphism (Method Overloading)**: Multiple methods with the same name but different signatures.
     + **Run-time polymorphism (Method Overriding)**: A method in a derived class overrides a method in the base class with the same signature.
6. **What is Abstraction in OOP?**
   * Answer: Abstraction is the concept of hiding the implementation details and showing only the essential features of an object. It is achieved in C# using abstract classes and interfaces.
7. **What is an Interface? How is it different from an Abstract Class?**
   * Answer:
     + **Interface**: A contract that defines a set of methods and properties without providing their implementation. A class implementing the interface must provide the concrete implementation of its members.
     + **Abstract Class**: Can have both implemented and unimplemented (abstract) methods. It provides a base class for other classes to derive from.
     + **Key Difference**: Interfaces cannot have any implementation, while abstract classes can have both abstract and concrete methods.
8. **What is Constructor and what are its types in C#?**
   * Answer: A constructor is a special method of a class that is automatically invoked when an object is created. It is used to initialize the object’s state. Types of constructors in C#:
     + **Default Constructor**: Parameterless constructor that initializes fields with default values.
     + **Parameterized Constructor**: Takes parameters and initializes fields with specific values.
     + **Copy Constructor**: Initializes an object by copying another object.
     + **Static Constructor**: Initializes static fields of the class and is called only once when the class is first accessed.
9. **What is Method Overloading?**
   * Answer: Method overloading allows multiple methods in a class to have the same name but different parameter lists (i.e., different types or numbers of parameters). It is an example of compile-time polymorphism.
10. **What is Method Overriding?**

* Answer: Method overriding allows a subclass to provide a specific implementation of a method that is already defined in its base class. The method in the base class must be marked with the virtual keyword, and the overriding method in the derived class should be marked with the override keyword.

 **What is the difference between ref and out parameters in C#?**

* Answer: Both are used to pass arguments by reference, but:
  + ref: The parameter must be initialized before it is passed to a method.
  + out: The parameter does not need to be initialized before it is passed, but it must be initialized within the method.

 **What is the nullable type in C#?**

* Answer: Nullable types allow value types (like int, float, etc.) to hold null values. It is declared using ?, e.g., int? or Nullable<int>.

 **What is a static class in C#?**

* Answer: A static class cannot be instantiated. It can only contain static members (fields, properties, methods) and is used when a class only serves as a container for static methods or constants.

 **What is the difference between const and readonly in C#?**

* Answer:
  + const: The value must be assigned at compile time and cannot be changed afterward.
  + readonly: The value can be assigned either at the time of declaration or in the constructor and cannot be changed afterward.

 **What is the purpose of the using statement in C#?**

* Answer: The using statement ensures that IDisposable objects are disposed of properly after their use, typically for managing unmanaged resources like file streams or database connections.

 **What is a sealed class in C#?**

* Answer: A sealed class cannot be inherited by any other class. It is used to prevent further inheritance, improving security or performance.

 **What is the difference between String and StringBuilder in C#?**

* Answer:
  + String: Immutable, meaning any operation on a string creates a new instance.
  + StringBuilder: Mutable, allowing for more efficient modification of strings, especially in loops.

 **What is boxing and unboxing in C#?**

* Answer:
  + **Boxing**: Converting a value type to a reference type (e.g., an int to object).
  + **Unboxing**: Converting a reference type back to a value type.

 **What is extension method in C#?**

* Answer: Extension methods allow you to add new methods to existing types without modifying their source code, typically by declaring static methods in static classes.

 **Explain the difference between throw and throw ex in C#.**

* Answer:
  + throw: Rethrows the original exception while preserving the stack trace.
  + throw ex: Throws the exception with a new stack trace, losing the original information.

 **What is try-catch-finally in C#?**

* Answer: It is a construct used for exception handling.
  + try: Contains code that may throw exceptions.
  + catch: Catches and handles the exception.
  + finally: Always executed, used for cleaning up resources.

 **What is the difference between Abstract Class and Interface in C#?**

* Answer:
  + **Abstract Class**: Can have both abstract and concrete methods. Supports fields and access modifiers.
  + **Interface**: Only contains method declarations. All members are public by default, and no fields are allowed.

 **What are indexers in C#?**

* Answer: Indexers allow objects to be indexed like arrays. They are defined using the this keyword and provide access to elements in an object.

 **What is a Lambda Expression in C#?**

* Answer: A lambda expression is an anonymous function that can contain expressions or statements. It is used in LINQ queries and as a shorthand for defining methods.

 **What is the async keyword in C#?**

* Answer: The async keyword is used to define methods that run asynchronously, meaning they do not block the main thread while waiting for long-running tasks.

 **What are properties in C#?**

* Answer: Properties in C# are special methods that provide access to private fields. They use get and set accessors to encapsulate data access and modification.

 **What is the purpose of the lock statement in C#?**

* Answer: The lock statement ensures that a block of code runs only on one thread at a time, preventing race conditions in multi-threaded environments.

 **What is the difference between Task and Thread in C#?**

* Answer:
  + **Thread**: Represents a low-level unit of execution that directly works with the operating system.
  + **Task**: A higher-level abstraction for concurrent operations, built on top of threads, commonly used with async programming.

 **What is Type Casting in C#?**

* Answer: Type casting is the process of converting one type to another. There are two types of casting:
  + **Implicit Casting**: Done automatically (e.g., int to float).
  + **Explicit Casting**: Requires a cast operator (e.g., float to int).

 **What is a Partial Class in C#?**

* Answer: A partial class allows a class to be divided into multiple files. It is often used to separate auto-generated code from developer-written code.

 **What are access modifiers in C#?**

* Answer: Access modifiers define the scope of class members. Common ones are:
  + public: Accessible from anywhere.
  + private: Accessible only within the class.
  + protected: Accessible within the class and derived classes.
  + internal: Accessible within the same assembly.
  + protected internal: Accessible within the same assembly or derived classes.

 **What is the difference between override and new keywords in C#?**

* Answer:
  + override: Used to extend or modify the abstract or virtual method in the base class.
  + new: Hides the base class method and creates a new method with the same name in the derived class.

 **What is multiple inheritance? Does C# support it?**

* Answer: Multiple inheritance refers to a class inheriting from more than one class. C# does not support multiple inheritance with classes but allows it through interfaces.

 **What is an abstract method in C#?**

* Answer: An abstract method is declared in an abstract class without any implementation. Derived classes must implement this method.

 **What is composition in OOP?**

* Answer: Composition is a design principle where one class contains another class as a part of its state. It represents a "has-a" relationship between objects.

 **What is the virtual keyword in C#?**

* Answer: The virtual keyword is used to modify a method or property in a base class that allows it to be overridden in derived classes.

 **What is dependency injection in C#?**

* Answer: Dependency Injection (DI) is a design pattern used to inject dependencies (services) into a class rather than allowing the class to create them. This promotes loose coupling and easier testing.