**How is C# different from C?**

| S.NO | C | C# |
| --- | --- | --- |
| 1. | C language supports procedural programming. | Whereas C# supports object oriented programming. |
| 2. | C language supports pointers. | Whereas in C#, pointers are used only in unsafe mode. |
| 3. | In C language, garbage collection is not. | While in C#, garbage collection is managed by [Common Language Runtime (CLR)](https://www.geeksforgeeks.org/common-language-runtime-clr-in-c-sharp/). |
| 4. | C language can be executed cross-platform. | Whereas [.NET Framework](https://www.geeksforgeeks.org/introduction-to-net-framework/) is required to execute C# language. |
| 5. | By using C language we can achieve low level of abstraction. | Whereas by using the C# we can achieve a high degree of abstraction. |
| 6. | C language is more on functions. | While C# is more on design. |
| 7. | C language gives top notch performance. | While C# gives standard performance. |
| 8. | There are 32 total keywords used in the C language. | While a total of 86 keywords are used in C#. |
| 9. | C language is mainly used in commercial industries and engineering. | Whereas C# is used for software formation and other networking related objective. |

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

1. Single line

//This is a single line comment.

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

/\*This is a multiple line comment.

1. XML Comments (///).

/// summary;

/// Set error message for multilingual language.

/// summary

### Can multiple catch blocks be executed?

No, Multiple catch blocks of similar type 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.

### 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.

### What is an object?

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.

### 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.

### 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.

### 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.

### 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.

### 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.

### 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.

### 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.

### 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!!";

### 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.

### 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.

### 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

### How can we sort the elements of the Array in descending order?

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

### 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.

### 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.

### 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.

### 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.

### List down the commonly used types of exceptions in .net

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

### 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.

### 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.

### 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

### 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.

### 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.

### 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.

### 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.

### 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.

### 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.

### 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)

{

}

### How we can create an array with non-default values?

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

### 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.

### What’s a multicast delegate?

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

### 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

### 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.

### 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.

### 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();

}

### 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.

### Is C# code is managed or unmanaged code?

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

### 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.

### 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();

}

}

}

### What is Common Language Runtime (CLR)?

CLR handles program execution for various languages including C#. The architecture of CLR handles memory management, garbage collection, security handling, and looks like:

### What is garbage collection in C#?

Garbage collection is the process of freeing up memory that is captured by unwanted objects. When you create a class object, automatically some memory space is allocated to the object in the heap memory. Now, after you perform all the actions on the object, the memory space occupied by the object becomes waste. It is necessary to free up memory. Garbage collection happens in three cases:

* If the occupied memory by the objects exceeds the pre-set threshold value.
* If the garbage collection method is called
* If your system has low physical memory

### What are the types of classes in C#?

Class is an entity that encapsulates all the properties of its objects and instances as a single unit. C# has four types of such classes:

* **Static class:** Static class, defined by the keyword ‘static’ does not allow inheritance. Therefore, you cannot create an object for a static class.

***Sample code:***

**static** **class** **classname**

{

//static data members

//static methods

}

* **Partial class:** Partial class, defined by the keyword ‘partial’ allows its members to partially divide or share source (.cs) files.
* **Abstract class:** Abstract classes are classes that cannot be instantiated where you cannot create objects. Abstract classes work on the OOPS concept of abstraction. Abstraction helps to extract essential details and hide the unessential ones.
* **Sealed class:** Sealed classes are classes that cannot be inherited. Use the keyword sealed to restrict access to users to inherit that class.

**sealed** **class** **InterviewBit**

{

// data members

// methods

.

.

.

}

### What is a managed and unmanaged code?

**Managed code** lets you run the code on a managed CLR runtime environment in the .NET framework.   
Managed code runs on the managed runtime environment than the operating system itself.   
Benefits: Provides various services like a garbage collector, exception handling, etc.   
  
**Unmanaged code** is when the code doesn’t run on CLR, it is an unmanaged code that works outside the .NET framework.   
They don’t provide services of the high-level languages and therefore, run without them. Such an example is C++.

### 6. What is the difference between an abstract class and an interface?

Let’s dig into the differences between an abstract class and an interface:

* Abstract classes are classes that cannot be instantiated ie. that cannot create an object. The interface is like an abstract class because all the methods inside the interface are abstract methods.
* Surprisingly, abstract classes can have both abstract and non-abstract methods but all the methods of an interface are abstract methods.
* Since abstract classes can have both abstract and non-abstract methods, we need to use the Abstract keyword to declare abstract methods. But in the interface, there is no such need.

An abstract class has constructors while an interface encompasses none.

Ex.

**Abstract class:**

**public** **abstract** **class** **Shape**{

**public** **abstract** **void** **draw**();

}

**Interface:**

**public** **interface** **Paintable**{

**void** **paint**();

}

### What are extension methods in C#?

Extension methods help to add new methods to the existing ones. The methods that are added are static. At times, when you want to add methods to an existing class but don’t perceive the right to modify that class or don’t hold the rights, you can create a new static class containing the new methods. Once the extended methods are declared, bind this class with the existing one and see the methods will be added to the existing one.

// C# program to illustrate the concept

// of the extension methods

**using** System;

**namespace** **ExtensionMethod** {

**static** **class** **NewMethodClass** {

// Method 4

**public** **static** **void** **M4**(**this** Scaler s)

{

Console.WriteLine("Method Name: M4");

}

// Method 5

**public** **static** **void** **M5**(**this** Scaler s, string str)

{

Console.WriteLine(str);

}

}

// Now we create a new class in which

// Scaler class access all the five methods

**public** **class** **IB** {

// Main Method

**public** **static** **void** **Main**(string[] args)

{

Scaler s = **new** Scaler();

s.M1();

s.M2();

s.M3();

s.M4();

s.M5("Method Name: M5");

}

}

}

**Output:**

Method Name: M1

Method Name: M2

Method Name: M3

Method Name: M4

Method Name: M5

### 9. What are Generics in C#?

In C# collections, defining any kind of object is termed okay which compromises C#’s basic rule of type-safety. Therefore, generics were included to type-safe the code by allowing re-use of the data processing algorithms. Generics in C# mean not linked to any specific data type. Generics reduce the load of using boxing, unboxing, and typecasting objects. Generics are always defined inside angular brackets <>. To create a generic class, this syntax is used:

GenericList<float> list1 = new GenericList<float>();

GenericList<Features> list2 = new GenericList<Features>();

GenericList<Struct> list3 = new GenericList<Struct>();

Here, GenericList<float> is a generic class. In each of these instances of GenericList<T>, every occurrence of T in the class is substituted at run time with the type argument. By substituting the T, we have created three different type-safe using the same class.

### 10. What is the difference between an Array and ArrayList in C#?

An array is a collection of similar variables clubbed together under one common name. While ArrayList is a collection of objects that can be indexed individually. With ArrayList you can access a number of features like dynamic memory allocation, adding, searching, and sorting items in the ArrayList.

* When declaring an array the size of the items is fixed therefore, the memory allocation is fixed. But with ArrayList, it can be increased or decreased dynamically.
* Array belongs to system.array namespace while ArrayList belongs to the system.collection namespace.
* All items in an array are of the same datatype while all the items in an ArrayList can be of the same or different data types.
* While arrays cannot accept null, ArrayList can accept null values.

**For ex.:**

// C# program to illustrate the ArrayList

**using** System;

**using** System.Collections;

**class** **IB** {

// Main Method

**public** **static** **void** **Main**(string[] args)

{

// Create a list of strings

ArrayList al = **new** ArrayList();

al.Add("Bruno");

al.Add("Husky");

al.Add(10);

al.Add(10.10);

// Iterate list element using foreach loop

**foreach**(**var** names **in** al)

{

Console.WriteLine(names);

}

}

}

### What is Boxing and Unboxing in C#?

The two functions are used for typecasting the data types:

**Boxing:** Boxing converts value type (int, char, etc.) to reference type (object) which is an implicit conversion process using object value.

*Example:*

int num = 23; // 23 will assigned to num

Object Obj = num; // Boxing

**Unboxing:** Unboxing converts reference type (object) to value type (int, char, etc.) using an explicit conversion process.

*Example:*

int num = 23; // value type is int and assigned value 23

Object Obj = num; // Boxing

int i = (int)Obj; // Unboxing

### 13. What are Properties in C#?

Properties in C# are public members of a class where they provide the ability to access private members of a class. The basic principle of encapsulation lets you hide some sensitive properties from the users by making the variables private. The private members are not accessible otherwise in a class. Therefore, by using properties in C# you can easily access the private members and set their values.   
  
The values can be easily assigned using get and set methods, also known as accessors. While the get method extracts the value, the set method assigns the value to the variables.

### 14. What are partial classes in C#?

Partial classes implement the functionality of a single class into multiple files. These multiple files are combined into one during compile time. The partial class can be created using the partial keyword.

**public** **partial** Clas\_name

{

// code

}

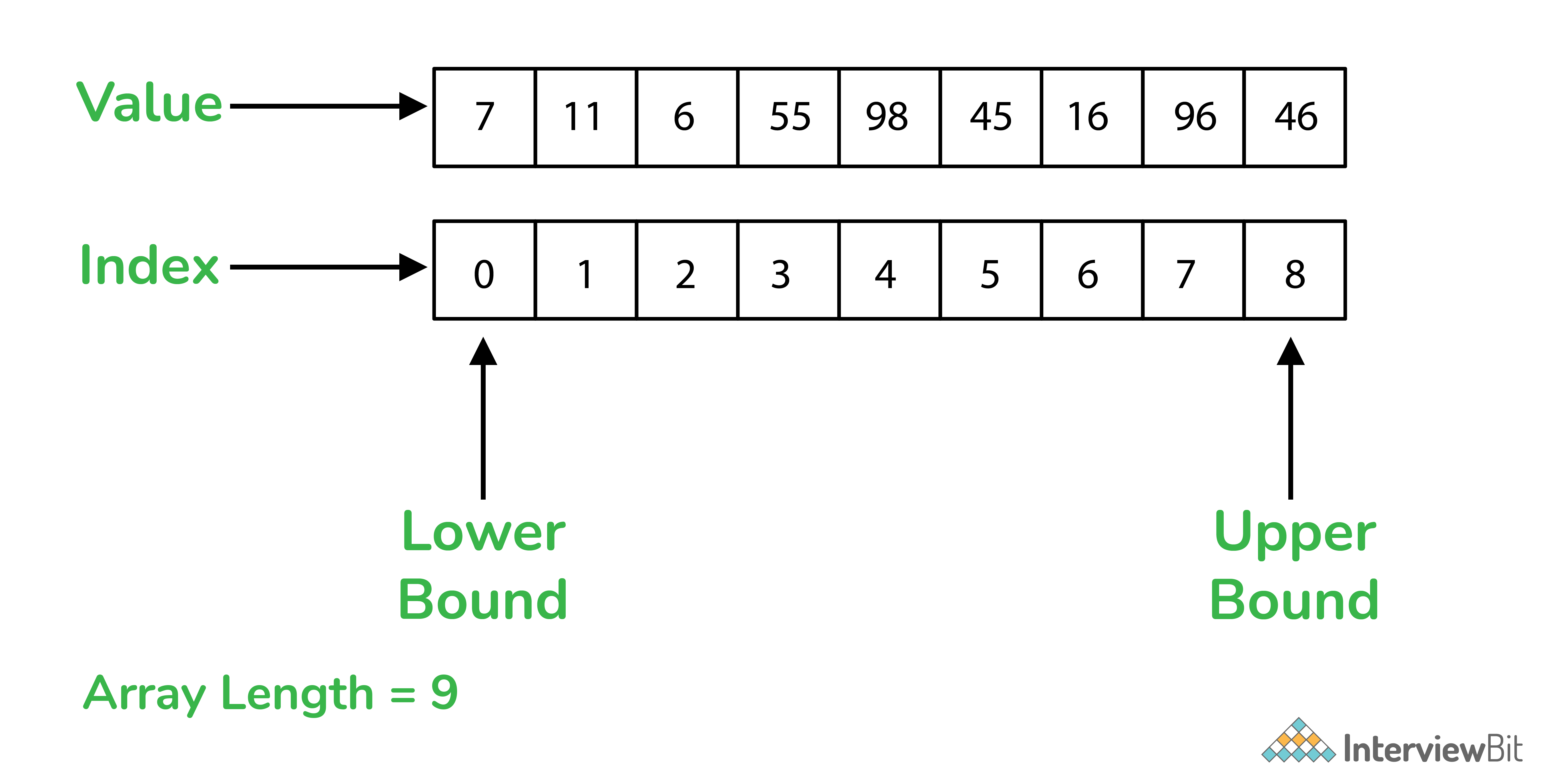
You can easily split the functionalities of methods, interfaces, or structures into multiple files. You can even add nested partial classes.

### 15. What is the difference between late binding and early binding in C#?

Late binding and early binding are examples of one of the primary concepts of OOPS: **Polymorphism.**  
  
*For ex:* one function calculateBill() will calculate bills of premium customers, basic customers, and semi-premium customers based on their policies differently. The calculation for all the customer objects is done differently using the same function which is called polymorphism.   
  
When an object is assigned to an object variable in C#, the .NET framework performs the binding.   
  
When the binding function happens at compile-time, it is called early binding. It investigates and checks the methods and properties of the static objects. With early binding, the number of run-time errors decreases substantially and it executes pretty quickly.   
  
But when the binding happens at runtime, it is called late binding. Late binding happens when the objects are dynamic (decided based on the data they hold) at run-time. It is slower as it looks through during run-time.

### 16. What are the Arrays in C#?

When a group of similar elements is clubbed together under one name, they are called arrays.   
  
For ex. An array of tea Atea[4]: [green tea, chamomile tea, black tea, lemon tea]. The length of the array defines how many elements are present in the array.   
  
In C#, the memory allocations for the elements of the array happen dynamically.  This is how values are stored in an array sequentially.

Arrays in C#

**A few pointers for arrays in C#:**

* The memory allocation is DYNAMIC.
* Arrays in C# are treated as objects.
* The length of the array is easy to find by detecting the number of members in the array.
* The members in the array are ordered and begin with the index value=0.
* The array types are reference types derived from the base array type.

Syntax: < Data Type > [ ] < Name\_Array >

### 17. What are Indexers in C#?

Indexers are called smart arrays that allow access to a member variable. Indexers allow member variables using the features of an array. They are created using the Indexer keyword. Indexers are not static members.

**For ex.** Here the indexer is defined the same way.

<**return** type> **this**[<parameter type> index]

{

**get**{

// return the value from the specified index of an internal collection

}

**set**{

// set values at the specified index in an internal collection

}

}

### 18. Difference between the Equality Operator (==) and Equals() Method in C#?

Although both are used to compare two objects by value, still they both are used differently.

**For ex.:**

int x = 10;

int y = 10;

Console.WriteLine( x == y);

Console.WriteLine(x.Equals(y));

Output:

True

True

**Equality operator (==**) is a reference type which means that if equality operator is used, it will return true only if both the references point to the same object.    
  
**Equals() method:** Equals method is used to compare the values carried by the objects. int x=10, int y=10. If x==y is compared then, the values carried by x and y are compared which is equal and therefore they return true.   
  
**Equality operator:** Compares by reference  
  
**Equals():** Compares by value

### 19. What are the different ways in which a method can be Overloaded in C#?

Overloading means when a method has the same name but carries different values to use in a different context. Only the main() method cannot be overloaded.

In order to overload methods in C#,

* Change the number of parameters in a method, or
* Change the order of parameters in a method, or
* Use different data types for parameters  
  In these ways, you can overload a method multiple times.

For ex.

**public** **class** **Area** {

**public** double **area**(double x) {

double area = x \* x;

**return** area;

}

**public** double **area**(double a, double b) {

double area = a \* b;

**return** area;

}

}

Here, the method Area is used twice. In the first declaration, one argument is used while in the second one, there were two arguments are used. Using different parameters in the same method, we were able to overload the method area().

### 20. What is Reflection in C#?

Reflection in C# extracts metadata from the datatypes during runtime.   
  
To add reflection in the .NET framework, simply use System.Refelction namespace in your program to retrieve the type which can be anything from:

* Assembly
* Module
* Enum
* MethodInfo
* ConstructorInfo
* MemberInfo
* ParameterInfo
* Type
* FieldInfo
* EventInfo
* PropertyInfo

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

The major difference between String and StringBuilder is that String objects are immutable while StringBuilder creates a mutable string of characters. StringBuilder will make the changes to the existing object rather than creating a new object.

StringBuilder simplifies the entire process of making changes to the existing string object. Since the String class is immutable, it is costlier to create a new object every time we need to make a change. So, the StringBuilder class comes into picture which can be evoked using the System.Text namespace.

In case, a string object will not change throughout the entire program, then use String class or else StringBuilder.

For ex:

string s = string.Empty;

**for** (i = 0; i < 1000; i++)

{

s += i.ToString() + " ";

}

Here, you’ll need to create 2001 objects out of which 2000 will be of no use.

The same can be applied using StringBuilder:

StringBuilder sb = **new** StringBuilder();

**for** (i = 0; i < 1000; i++)

{

sb.Append(i); sb.Append(' ');

}

By using StringBuilder here, you also de-stress the memory allocator.

### Write a program in C# Sharp to reverse a string?

**internal** **static** **void** **ReverseString**(string str)

{

char[] charArray = str.ToCharArray();

**for** (int i = 0, j = str.Length - 1; i < j; i++, j--)

{

charArray[i] = str[j];

charArray[j] = str[i];

}

string reversedstring = **new** string(charArray);

Console.WriteLine(reversedstring);

### 24. Write a program in C# Sharp to reverse the order of the given words?

**internal** **static** **void** **ReverseWordOrder**(string str)

{

int i;

StringBuilder reverseSentence = **new** StringBuilder();

int Start = str.Length - 1;

int End = str.Length - 1;

**while** (Start > 0)

{

**if** (str[Start] == ' ')

{

i = Start + 1;

**while** (i <= End)

{

reverseSentence.Append(str[i]);

i++;

}

reverseSentence.Append(' ');

End = Start - 1;

}

Start--;

}

**for** (i = 0; i <= End; i++)

{

reverseSentence.Append(str[i]);

}

Console.WriteLine(reverseSentence.ToString());

}

### 25. Write a program in C# Sharp to find if a given string is palindrome or not?

**internal** **static** **void** **chkPalindrome**(string str)

{

bool flag = false;

**for** (int i = 0, j = str.Length - 1; i < str.Length / 2; i++, j--)

{

**if** (str[i] != str[j])

{

flag = false;

**break**;

}

**else**

flag = true;

}

**if** (flag)

{

Console.WriteLine("Palindrome");

}

**else**

Console.WriteLine("Not Palindrome");

**Output:**  
  
Input: Key Output: Not Palindrome  
Input: step on no pets Output: Palindrome

### 26. Write a C# program to find the substring from a given string.

**internal** **static** **void** **findallsubstring**(string str)

{

**for** (int i = 0; i < str.Length; ++i)

{

StringBuilder subString = **new** StringBuilder(str.Length - i);

**for** (int j = i; j < str.Length; ++j)

{

subString.Append(str[j]);

Console.Write(subString + " ");

}

}

}

### 27. Write a C# program to find if a positive integer is prime or not?

**static** **void** **Main**(string[] args)

{

**if** (FindPrime(47))

{

Console.WriteLine("Prime");

}

**else**

{

Console.WriteLine("Not Prime");

}

Console.ReadLine();

}

**internal** **static** bool **FindPrime**(int number)

{

**if** (number == 1) **return** false;

**if** (number == 2) **return** true;

**if** (number % 2 == 0) **return** false;

**var** squareRoot = (int)Math.Floor(Math.Sqrt(number));

**for** (int i = 3; i <= squareRoot; i += 2)

{

**if** (number % i == 0) **return** false;

}

**return** true;

}

## **C# MCQ**

1.

What is the need for ‘Conversion of data type’ in C#?

To store a value of one data type into a variable of another data type

To get desired data

To prevent situations of runtime error during change or conversion of data type

None of the mentioned

2.

What are the types of ‘Data Conversion’ in C#?

Implicit Conversion

Explicit Conversion

Implicit Conversion and Explicit Conversion

None of the mentioned

3.

What is the subset of ‘int’ datatype?

long, ulong, ushort

long, ulong, uint

long, float, double

long, float, ushort

4.

What will be the output of the following C# code?

**static** **void** **Main**(string[] args)

{

int a, b, c, x;

a = 90;

b = 15;

c = 3;

x = a - b / 3 + c \* 2 - 1;

Console.WriteLine(x);

Console.ReadLine();

}

90

92

89

88

5.

The correct way of incrementing the operators are:

d =+ 1

c += 1

b ++ 1

++ a ++

6.

Which reference modifier is used to define reference variables?

&

ref

#

$

7.

What will be the output of the following C# code?

**static** **void** **Main**(string[] args)

{

int []a = { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10};

func(**ref** a);

Console.ReadLine();

}

**static** **void** **func**(**ref** int[] x)

{

Console.Write(" numbers are : ");

**for** (int i = 0; i < x.Length; i++)

{

**if** (x[i] % 2 == 0)

{

x[i] = x[i] + 1;

Console.Write(x[i] +" ");

}

}

}

numbers are : 3 5 7 9 11

numbers are : 2 4 6 8 10

numbers are : 2 3 4 5 6

none of the mentioned

8.

What will be the output of the following code?

**static** **void** **Main**(string[] args)

{

int x = 4 ,b = 2;

x -= b/= x \* b;

Console.WriteLine(x + " " + b);

Console.ReadLine();

}

4 2

0 4

2 2

4 0

9.

What will be the output of the following C# expression?

int a+= (float) b/= (long)c

float

int

long

none of the mentioned

10.

Which of the following modifiers is used when an abstract method is redefined by a derived class?

Override

Overloads

Base

Virtual

11.

What is the process of defining a method in terms of itself, that is a method that calls itself?

Polymorphism

Abstraction

Encapsulation

Recursion

12.

What is the output of this code?

class maths

{

public int fun(int k, int y)

{

return k + y;

}

public int fun1(int t, float z)

{

return (t+(int)z);

}

}

class Program

{

static void Main(string[] args)

{

maths obj = new maths();

int i;

int b = 90;

int c = 100;

int d = 12;

float l = 14.78f;

i = obj.fun(b, c);

Console.WriteLine(i);

int j = (obj.fun1(d, l));

Console.WriteLine(j);

Console.ReadLine();

}

}

190, 26.78f

0, 26.78f

190, 26

190, 0

https://www.interviewbit.com/c-sharp-interview-questions/