Number of occurrences of 2 as a digit in numbers from 0 to n

Count the number of 2s as digit in all numbers from 0 to n.

**Examples :**

Input : 22

Output : 6

Explanation: Total 2s that appear as digit

from 0 to 22 are (2, 12, 20,

21, 22);

Input : 100

Output : 20

Explanation: total 2's comes between 0 to 100

are (2, 12, 20, 21, 22..29, 32, 42, 52, 62, 72,

82, 92);

Answer :

// Counts the number of '2' digits

// between 0 and n

static int numberOf2sinRange(int n)

{

    // Initialize result

    int count = 0;

    // Count 2's in every number

    // from 2 to n

    for (int i = 2; i <= n; i++)

    count += number0f2s(i);

    return count;

}

static int number0f2s(int n)

{

    int count = 0;

    while (n > 0)

    {

        if (n % 10 == 2)

            count++;

    n = n / 10;

    }

    return count;

}

// Driver code

public static void Main()

{

    Console.Write(numberOf2sinRange(22));

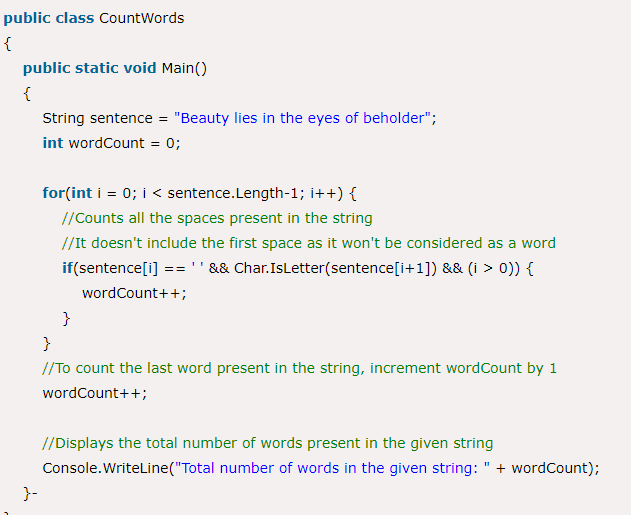
    Console.WriteLine();

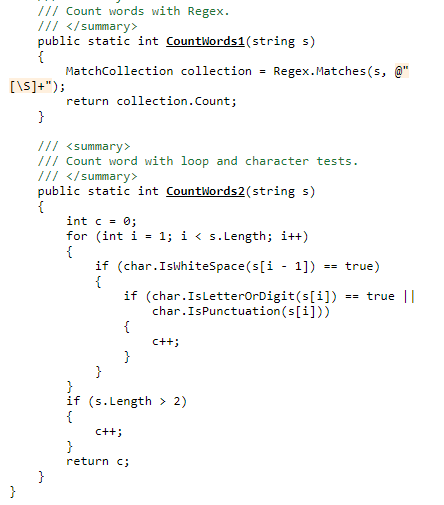
    Console.Write(numberOf2sinRange(100));

}

}

Program to count the total number of words in a string





SOLID principles

* S: Single Responsibility Principle (SRP)
* O: Open closed Principle (OSP)
* L: Liskov substitution Principle (LSP)
* I: Interface Segregation Principle (ISP)
* D: Dependency Inversion Principle (DIP

S: Single Responsibility Principle (SRP)

This means that every class, or similar structure, in your code should have only one job to do. Everything in that class should be related to a single purpose. Ex: The SendEmail and ValidateEmail methods have nothing to do within the UserService class.

O: Open/Closed Principle

The Open/closed Principle says "A software module/class is open for extension and closed for modification".

 Here "Open for extension" means, we need to design our module/class in such a way that the new functionality can be added only when new requirements are generated. "Closed for modification" means we have already developed a class and it has gone through unit testing. We should then not alter it until we find bugs. As it says, a class should be open for extensions, we can use inheritance to do this.

L: Liskov Substitution Principle

This principle is just an extension of the Open Closed Principle and it means that we must ensure that new derived classes extend the base classes without changing their behavior.Child calss should be substutable for parent ex brand car shoulbe substutiable for car feature class

I: Interface Segregation Principle (ISP)/Multiple interface inheritance

The Interface Segregation Principle states "that clients should not be forced to implement interfaces they don't use. Instead of one fat interface, many small interfaces are preferred based on groups of methods, each one serving one submodule.".

## .D: Dependency Inversion Principle

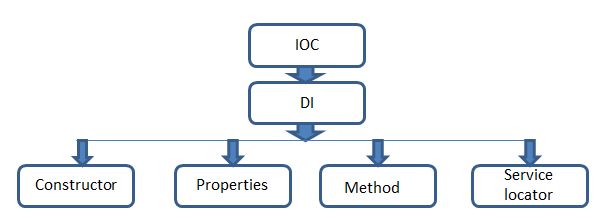
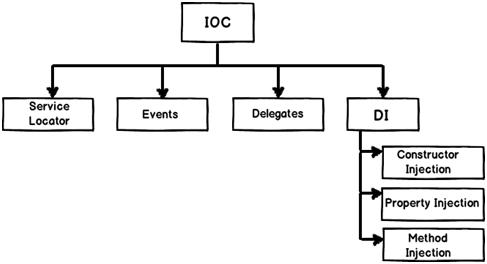
The Dependency Inversion Principle (DIP) states that high-level modules/classes should not depend on low-level modules/classes. Both should depend upon abstractions. Secondly, abstractions should not depend upon details. Details should depend upon abstractions.

**IOC & Dependency Injection**

IOC can be done using **Dependency Injection** (DI). It explains how to inject the concrete implementation into a class that is using abstraction, in other words an interface inside. The main idea of dependency injection is to reduce the coupling between classes and move the binding of abstraction and concrete implementation out of the dependent class.

In Simple words, DI is how one object know about other dependent object which is abstracted.

There are mainly 4 ways of achieving the Dependency Injection.



**Injection via Constructor,** **Property,** **Method**

**Advantages of implementing this principle**

* It helps in class decoupling.
* Due to decoupling, the reusability of the code is increased.
* Improved code maintainability and testing.

***Inversion of control (IOC) talks about who is going to initiate the call where as the Dependency Injection (DI) talks about how one object acquires dependency on other object through abstraction.***

For example, say your application has a text editor component and you want to provide spell checking. Your standard code would look something like this:

public class TextEditor {

private SpellChecker checker;

public TextEditor() {

this.checker = new SpellChecker();

}

}

What we've done here creates a dependency between the TextEditor and the SpellChecker. In an IoC scenario we would instead do something like this:

public class TextEditor {

private IocSpellChecker checker;

public TextEditor(IocSpellChecker checker) {

this.checker = checker;

}

}

In the first code example we are instantiating SpellChecker (this.checker = new SpellChecker();), which means the TextEditor class directly depends on the SpellChecker class.

In the second code example we are creating an abstraction by having the SpellChecker dependency class in TextEditor's constructor signature (not initializing dependency in class). This allows us to call the dependency then pass it to the TextEditor class like so:

SpellChecker sc = new SpellChecker(); // dependency

TextEditor textEditor = new TextEditor(sc);

Now the client creating the TextEditor class has control over which SpellChecker implementation to use because we're injecting the dependency into the TextEditor signature.

Service Locator (SL)

Service Locator is a software design pattern that also allows us to develop loosely coupled code. It implements the DIP principle and easier to use with an existing codebase as it makes the overall design looser without forcing changes to the public interface.

The Service Locator pattern introduces a locator object that objects are used to resolve dependencies means it allows you to "resolve" a dependency within a class.

Find closest number in array

<https://www.geeksforgeeks.org/find-closest-number-array/>

Given an array of sorted integers. We need to find the closest value to the given number. Array may contain duplicate values and negative numbers.

**Examples:**

Input : arr[] = {1, 2, 4, 5, 6, 6, 8, 9}

Target number = 11

Output : 9

9 is closest to 11 in given array

Input :arr[] = {2, 5, 6, 7, 8, 8, 9};

Target number = 4

Output : 5

## C#

|  |
| --- |
| // C# program to find element  // closet to given target.  using System;    class GFG  {        // Returns element closest      // to target in arr[]      public static int findClosest(int []arr,                                    int target)      {          int n = arr.Length;            // Corner cases          if (target <= arr[0])              return arr[0];          if (target >= arr[n - 1])              return arr[n - 1];            // Doing binary search          int i = 0, j = n, mid = 0;          while (i < j)          {              mid = (i + j) / 2;                if (arr[mid] == target)                  return arr[mid];                /\* If target is less              than array element,              then search in left \*/              if (target < arr[mid])              {                    // If target is greater                  // than previous to mid,                  // return closest of two                  if (mid > 0 && target > arr[mid - 1])                      return getClosest(arr[mid - 1],                                   arr[mid], target);                    /\* Repeat for left half \*/                  j = mid;              }                // If target is              // greater than mid              else              {                  if (mid < n-1 && target < arr[mid + 1])                      return getClosest(arr[mid],                           arr[mid + 1], target);                  i = mid + 1; // update i              }          }            // Only single element          // left after search          return arr[mid];      }        // Method to compare which one      // is the more close We find the      // closest by taking the difference      // between the target and both      // values. It assumes that val2 is      // greater than val1 and target      // lies between these two.      public static int getClosest(int val1, int val2,                                   int target)      {          if (target - val1 >= val2 - target)              return val2;          else              return val1;      }        // Driver code      public static void Main()      {          int []arr = {1, 2, 4, 5,                       6, 6, 8, 9};          int target = 11;          Console.WriteLine(findClosest(arr, target));      }  }  What is anonymous type in C#?  In C#, an anonymous type is a type (class) without any name that can contain public read-only properties only. It cannot contain other members, such as fields, methods, events, etc. You create an anonymous type using the *new* operator with an [object initializer](https://www.tutorialsteacher.com/csharp/csharp-object-initializer) syntax. The [implicitly typed variable- var](https://www.tutorialsteacher.com/csharp/csharp-var-implicit-typed-local-variable) is used to hold the reference of anonymous types.  var student = new { Id = 1, FirstName = "James", LastName = "Bond" };  var students = from s in studentList  select new { Id = s.StudentID, Name = s.StudentName };  **Anonymous types** provide a convenient way to encapsulate a set of read-only properties into a single object without having to explicitly define a **type** first. The **type** name is generated by the compiler and is not available at the source code level. The **type** of each property is inferred by the compiler.  What are lambda expressions in C#?  A **lambda expression** is a convenient way of defining an anonymous (unnamed) function that can be passed around as a variable or as a parameter to a method call. Many LINQ methods take a function (called a delegate) as a parameter. Lambda expressions are how anonymous functions are created. Lambda expressions are anonymous functions that contain expressions or sequence of operators. All lambda expressions use the lambda operator =>, that can be read as “goes to” or “becomes”. The left side of the lambda operator specifies the input parameters and the right side holds an expression or a code block that works with the entry parameters Expression Lambdas Parameter => expressionParameter-list => expressionCount => count + 2;Sum => sum + 2;n => n % 2 == 0  The lambda operator => divides a lambda expression into two parts. The left side is the input parameter and the right side is the lambda body  list.FindAll(x => (x % 2) == 0)  dogs.Select(x => **new** { Age = x.Age, FirstLetter = x.Name[0] });  dogs.OrderByDescending(x => x.Age);  <https://www.c-sharpcorner.com/UploadFile/dacca2/lambda-expression-in-15-minutes/>  What are the three types of Generic delegates in C#?  three types of generic delegates in C# - 1) Func 2) Action 3) Predicate  <https://www.c-sharpcorner.com/article/delegates-action-vs-func-vs-predicate2/>  Action is a delegate, it can be used to point a method that has no return type. (i.e. return type will be void.)   1. **static** **void** Main(**string**[] args) 2. { 3. Action<**string**> log = **new** Action<**string**>(LogInfo); 4. log.Invoke("Hi ALL"); 5. Console.ReadLine(); 7. } 9. **static** **void** LogInfo(**string** message) 10. { 11. Console.WriteLine(message); 12. }   Func    Func is a delegate, we can define type(s) of input params, and at the end, we can write the output param type.   1. **static** **void** Main(**string**[] args) 2. { 3. Func<**int**, **int**, **int**> addFunc = **new** Func<**int**, **int**, **int**>(Add); 4. **int** result = addFunc(3, 4); 5. Console.WriteLine(result); 6. Console.ReadLine(); 8. } 9. **static** **int** Add(**int** a, **int** b) 10. { 11. **return** a + b; 12. }   Predicate    Predicate will always return bool, which accepts any type of parameter as its input.   1. **static** **void** Main(**string**[] args) 2. { 3. Predicate<**int**> IsEven = **new** Predicate<**int**>(IsEvenNumber); 4. Console.WriteLine(IsEven(10)); 5. Console.WriteLine(IsEven(1567)); 6. Console.ReadLine(); 8. } 9. **static** **bool** IsEvenNumber(**int** number) 10. { 11. **return** number % 2 == 0; 12. }  What is IEnumerable in C#?   IEnumerable in C# is an interface that defines one method, GetEnumerator which returns an IEnumerator interface. This allows readonly access to a collection then a collection that implements IEnumerable can be used with a for-each statement.   Key Points  1. IEnumerable interface contains the System.Collections.Generic namespace. 2. IEnumerable interface is a generic interface which allows looping over generic or non-generic lists. 3. IEnumerable interface also works with linq query expression. 4. IEnumerable interface Returns an enumerator that iterates through the collection.   <https://www.c-sharpcorner.com/UploadFile/0c1bb2/ienumerable-interface-in-C-Sharp/>  **C# yield Example:** Use the yield keyword to implement IEnumerable. Return elements that are used in foreach.  The function returns an object that implements the IEnumerable<object> interface. If a calling function starts foreaching over this object, the function is called again until it "yields". This is syntactic sugar introduced in **C# 2.0**. In earlier versions you had to create your own IEnumerable and IEnumerator objects to do stuff like this.  The easiest way understand code like this is to type-in an example, set some breakpoints and see what happens. Try stepping through this example:  public void Consumer()  {  foreach(int i in Integers())  {  Console.WriteLine(i.ToString());  }  }  public IEnumerable<int> Integers()  {  yield return 1;  yield return 2;  yield return 4;  yield return 8;  yield return 16;  yield return 16777216;  }  When you step through the example, you'll find the first call to Integers() returns 1. The second call returns 2 and the line yield return 1 is not executed again.  Here is a real-life example:  public IEnumerable<T> Read<T>(string sql, Func<IDataReader, T> make, params object[] parms)  {  using (var connection = CreateConnection())  {  using (var command = CreateCommand(CommandType.Text, sql, connection, parms))  {  command.CommandTimeout = dataBaseSettings.ReadCommandTimeout;  using (var reader = command.ExecuteReader())  {  while (reader.Read())  {  yield return make(reader);  }  }  }  }  } |

# Managed code and unmanaged code in .NET

The code, which is developed in .NET framework, is known as managed code. This code is directly executed by CLR with help of managed code execution. Any language that is written in .NET Framework is managed code.

Managed code uses CLR which in turns looks after your applications by managing memory, handli **Unmanaged Code**

The code, which is developed outside .NET, Framework is known as unmanaged code.

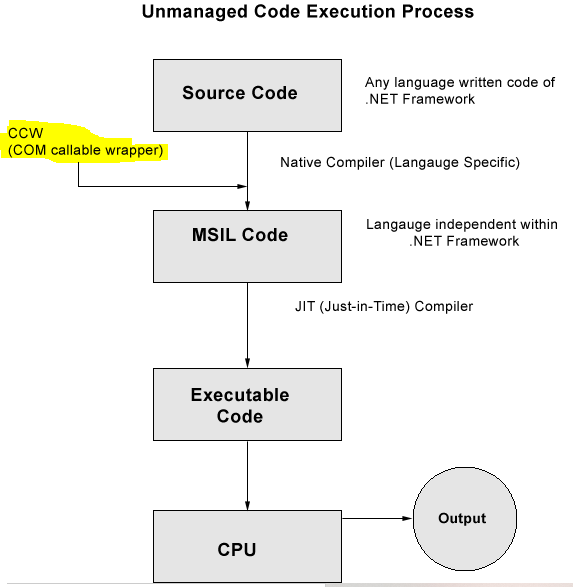
Applications that do not run under the control of the CLR are said to be unmanaged, and certain languages such as C++ can be used to write such applications, which, for example, access low - level functions of the operating system. Background compatibility with code of VB, ASP and COM are examples of unmanaged code.

Unmanaged code can be unmanaged source code and unmanaged compile code.

Unmanaged code is executed with help of wrapper classes.

Wrapper classes are of two types: **CCW (COM Callable Wrapper)** and **RCW (Runtime Callable Wrapper).**

Wrapper is used to cover difference with the help of CCW and RCW.



# Static Constructors

class SimpleClass

{

// Static variable that must be initialized at run time.

static readonly long baseline;

// Static constructor is called at most one time, before any

// instance constructor is invoked or member is accessed.

static SimpleClass()

{

baseline = DateTime.Now.Ticks;

}

# }

Static constructors have the following properties:

* A static constructor does not take access modifiers or have parameters.
* A class or struct can only have one static constructor.
* Static constructors cannot be inherited or overloaded.
* A static constructor cannot be called directly and is only meant to be called by the common language runtime (CLR). It is invoked automatically.
* The user has no control on when the static constructor is executed in the program.
* A static constructor is called automatically to initialize the [class](https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/keywords/class) before the first instance is created or any static members are referenced.

### Usage

* A typical use of static constructors is when the class is using a log file and the constructor is used to write entries to this file.
* Static constructors are also useful when creating wrapper classes for unmanaged code, when the constructor can call the LoadLibrary method.
* Static constructors are also a convenient place to enforce run-time checks on the type parameter that cannot be checked at compile time via constraints (Type parameter constraints).

<https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/classes-and-structs/static-constructors>

# Indexes in SQL Server

<https://www.c-sharpcorner.com/UploadFile/8af593/index-in-sql-server/>

SQL Indexes are used in relational databases to quickly retrieve data. They are similar to indexes at the end of the books whose purpose is to find a topic quickly. SQL provides Create Index, Alter Index, and Drop Index commands that are used to create a new index, update an existing index, and delete an index in SQL Server.

* Data is internally stored in a SQL Server database in “pages” where the size of each page is 8KB.
* A continuous 8 pages is called an “Extent”.
* When we create the table then one extent will be allocated for two tables and when that extent is computed it is filled with the data then another extent will be allocated and this extent may or may not be continuous to the first extent.

|  |  |
| --- | --- |
| Clustered Index | Non- Clustered Index |
| This will arrange the rows physically in the memory in sorted order | This will not arrange the rows physically in the memory in sorted order. |
| This will fast in searching for the range of values. | This will be fast in searching for the values that are not in the range. |
| Index for table. | You can create a maximum of 999 non clustered indexes for table. |
| Leaf node of 3 tier of clustered index contains, contains table data. | Leaf nodes of b-tree of non-clustered index contains pointers to get the contains pointers to get that contains two table data, and not the table data directly. |

<https://www.sqlshack.com/top-25-sql-interview-questions-and-answers-about-indexes/>

# What is Normalization? 1NF, 2NF, 3NF, BCNF Database Example

## What is Normalization?

**Normalization** is a database design technique that reduces data redundancy and eliminates undesirable characteristics like Insertion, Update and Deletion Anomalies. Normalization rules divides larger tables into smaller tables and links them using relationships. The purpose of Normalization in SQL is to eliminate redundant (repetitive) data and ensure data is stored logically.

Database Normal Forms

<https://www.guru99.com/database-normalization.html>

<https://www.edureka.co/blog/normalization-in-sql/>

What is the opposite of 'Contains' in C#?

you can use not operator for check enter value is not in list .

1. if (!myList.Contains("name"))
2. {
3. myList.Add("name");
4. }

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

When we use “throw” , it will throw a new exception without carrying real error info. But when we use “throw ex”, object ex will have error info like ErrorMessage from which we can identity the real reason behind this exception.  
Another scenario, if we wish to show user-friendly error message for technical exception (like Database connect issue), we can change ErroMessage property to a user-friendly message and throw that exception as “throw ex” to client.

Can we use "this" command within a static method?

No, we can’t use “this” keyword inside a static method.

“this” refers to current instance of the class. But if we define a method as static , class instance will not have access to it, only CLR executes that block of code. Hence we can’t use “this” keyword inside static method.

What is IS and AS operator in c# .NET?

s Operator is used to Check the Compatibility of an Object with a given Type and it returns the result as a Boolean.

1. if(obj is int)
2. {
3. ...
4. }

As Operator is used for Casting of Object to a given Type or a Class. The as keyword is used to cast nullable types if the specified value is not an instance of the specified type.

1. AnyClass anyObject = obj as AnyClass;
2. What is Most Optimized Thread Safe Way To Lock Resources for Singleton Pattern?

Problem Statement is: the lock statement is used in case of implementing the Thread Safe Singleton Pattern.  
The question is, we need to lock a resource but using a lock(x){} statement is bit expensive as per the interviewer. What is the most efficient or optimum way to lock a resource without using lock statement?

Print 1-100 Numbers without using loop in Console Application ?

use recursive function exit when number reached to 100

1. List<int> list = Enumerable.Range(number, 100).ToList();
2. string s = string.Join(", ",list.Select(x =>x.ToString()));
3. Console.WriteLine(s);

Write a program to find an number in array of integers using Binary Search using c#

1. int[] arr = { 23, 56, 67, 78, 89, 90, 94, 95, 96, 97, 99 };
2. int key = 97;
3. int minNum = 0;
4. int maxNum = arr.Length - 1;
5. while (minNum <= maxNum)
6. {
7. int mid = (minNum + maxNum) / 2;
8. if (key == arr[mid])
9. {
10. ++mid;
11. break;
12. }
13. else if (key<arr[mid])
14. {
15. maxNum = mid - 1;
16. }
17. else
18. {
19. minNum = mid + 1;
20. }
21. }

Write a program for sorting an array of integers using c#

1. int[] numberArray = { 1, 3, 5, 67, 22, 90, 0, 6 };
2. for (int i = 0; i < numberArray.Length; i++)
3. {
4. for (int j = i; j < numberArray.Length; j++)
5. {
6. if (numberArray[i] < numberArray[j])
7. {
8. int temp = numberArray[j];
9. numberArray[j] = numberArray[i];
10. numberArray[i] = temp;
11. }
12. }
13. }
14. for (int i = 0; i < numberArray.Length; i++)
15. {
16. Console.WriteLine(numberArray[i]);
17. }
18. Console.Read();

Write a program for finding an missing number in between array of integer using c#

1. int[] array = { 1, 3, 6, 8, 14, 18 };
2. stringbuilder sbr = new stringbuilder();
3. for (int i = 0; i < array.length - 1; i++)
4. {
5. for (int j = array[i]; j < array[i + 1]; j++)
6. {
7. var temp = j + 1;
8. if (temp < array[i + 1])
9. {
10. sbr.append(temp);
11. }
12. }
13. }
14. console.writeline(sbr.tostring());
15. console.read();

///----

nt[] array = { 1, 3, 6, 8, 14, 18 };  
StringBuilder sbr = new StringBuilder();

1. for (int i = 0; i <array.Length-1; i++)
2. {
3. if (array[i] < array[i + 1])
4. {
5. int startval = array[i];
6. int endval = array[i + 1];
7. for (int j = startval+1; j < endval; j++)
8. {
9. sbr.Append( j+ " ");
10. }
11. }
12. }
13. Console.WriteLine(sbr.ToString());
14. Console.ReadKey();

Write a program for character sorting from string using c#

1. string str;
2. char[] arr1;
3. char ch;
4. int i, j, l;
5. Console.Write("\n\nSort a string array in ascending order :\n");
6. Console.Write("--------------------------------------------\n");
7. Console.Write("Input the string : ");
8. str = Console.ReadLine();
9. l = str.Length;
10. arr1 = str.ToCharArray(0, l);
11. for (i = 1; i < l; i++)
12. for (j = 0; j < l - i; j++)
13. if (arr1[j] > arr1[j + 1])
14. {
15. ch = arr1[j];
16. arr1[j] = arr1[j + 1];
17. arr1[j + 1] = ch;
18. }
19. Console.Write("After sorting the string appears like : \n");
20. foreach (char c in arr1)
21. {
22. ch = c;
23. Console.Write("{0} ", ch);
24. }
25. Console.WriteLine("\n");
26. Jul, 202014
27. **Static** a modifier used in C#. It can be used for variable, methods, classes, constructor etc. Creating static classes means that the class can only contain static properties such as static variables and static methods. We cannot create instances of static classes, it simply means that it doesn’t matter how many objects of the static class are created only one copy of the static memeber will exist.

How can you return more than one value from a method?

There are multiple ways to do it.1) by using struct/Class as return type 2) by passing parameters by ref, using out params, using tuple,…

|  |  |
| --- | --- |
| Abstract Class | Interface |
| It contains both declaration and definition part. | It contains only a declaration part. |
| Multiple inheritance is not achieved by abstract class. | Multiple inheritance is achieved by interface. |
| It contain [constructor](https://www.geeksforgeeks.org/c-sharp-constructors/). | It does not contain [constructor](https://www.geeksforgeeks.org/c-sharp-constructors/). |
| It can contain static members. | It does not contain static members. |
| It can contain different types of access modifiers like public, private, protected etc. | It only contains public access modifier because everything in the interface is public. |
| The performance of an abstract class is fast. | The performance of interface is slow because it requires time to search actual method in the corresponding class. |
| It is used to implement the core identity of class. | It is used to implement peripheral abilities of class. |
| A class can only use one abstract class. | A class can use multiple interface. |
| If many implementations are of the same kind and use common behavior, then it is superior to use abstract class. | If many implementations only share methods, then it is superior to use Interface. |
| Abstract class can contain methods, fields, constants, etc. | Interface can only contain methods . |
| It can be fully, partially or not implemented. | It should be fully implemented. |

C# is strong type ? C# is mostly strong type because it determines the error at compile time but some exceptions like dynamic keyword has been introduced in c# which work at run time. so it's up to us to decide even most are strong type dynamic keyword is exception.

Can we inherit an abstract class from another abstract class? public abstract class ABS1  
{  
public abstract void Read();  
}  
public abstract class ABS2:ABS1  
{  
public abstract void Write();  
}  
public class AbsChild : ABS2  
{  
public override void Read()  
{  
throw new NotImplementedException();  
}

What is the role of [Flag] attribute in enums?

The [Flags] attribute should be used whenever the enumerable represents a collection of possible values, rather than a single value. Such collections are often used with bitwise operators, for example:

[Flags]

public enum MyColors

{

Yellow = 1,

Green = 2,

Red = 4,

Blue = 8

}

var allowedColors = MyColor.Red | MyColor.Green | MyColor.Blue;

What is short-circuiting?

net run time tries to find the requested result as earliest and returns it back the caller. This is called short-circuiting.

Ex:-

if(statement 1){  
return 1;  
}  
else if( statement 2){  
return 2;  
}  
else{  
return 3;  
}

as soon as any statement expression is true , runtime returns that value as response and hence short-circuiting the request pipeline.

what is basic diff. b/w tostring() and convert.tostring()?

tostring() : returns a null reference exception convert.tostring() : returns an empty string and does not throw an exception

What will be the return type if we use .Any() in entity framework. if we will use .any in Entity Code it will always return Bool results.  
i.e It My Be **TRUE** or My Be **False**

In which of the following collections is the Input/Output index-based?

1. Stack
2. Queue
3. BitArray
4. ArrayList
5. HashTable

3&4 BitArray and ArrayList

What will be the output of the C#.NET code snippet given below?

1. string str = "Hello World!";
2. Console.WriteLine(String.Compare(str, "Hello World?").GetType());

System.Int32

What happens when the following elements get inserted in an ArrayList?

1. al.Add(1);
2. al.Add('1');
3. al.Add("1");

a) Compilation Error  
b) Code works perfectly - Prints 1 ‘1’ “1” if an Console.Write statement is written.  
c) System.IndexOutOfRangeException  
d) System.InvalidCastException  
e) System.StackOverflowException  
f) System.IO.IOException

Unlike arrays, Arraylist can have objects of different types. So it will work. b.

What will be the output of the following C# code

1. byte num = 100;
2. dynamic val = num;
3. //What will be displayed as output for-
4. Console.Writeline(val.GetType());
5. //and
6. val+=100;
7. Console.Writeline(val.GetType());

Is it possible to store mixed datatypes such as int, string, float, char, all in one array? we can use dynamic array ex- dynamic[] arr = {"abc",11,11.23 };We can do that but , Array List is the correct

In an ArrayList, if an item is added using Insert method like below- al.Insert(2, "item") What will happen to the existing item at index 2?

It will shift to the next index of that ArrayList.

Please follow the given example below

ArrayList a1 = new ArrayList();  
a1.Add(“abc”);  
a1.Add(“aa”);  
a1.Add(“baba”);

//before inserting new element  
Console.WriteLine(a1[2]);

//adding new element at 2nd index  
a1.Insert(2, “item”);

//after added  
Console.WriteLine(a1[2]);  
Console.WriteLine(a1[3]);  
//“baba” is shifted at index 3 now

Console.ReadLine();

Can DateTime be compared to null? Why or why not?

DateTime CAN be compared to null; It cannot hold null value, thus the comparison will always be false. DateTime is a "Value Type". Basically a "value type" can't set to NULL. But by making them to "Nullable" type, We can set to null. Yes, If declare DateTime variable as nullable like DateTime? then we can compare it with null.

Write the name of the three ways to pass a parameter to a method in C#.

1Value Parameters 2. Reference Parameters 3. Output Parameters

Can we inherit Static Class in C# ?

static classes are sealed classes , they can not be inherit

Can an abstract class have static methods? Yes, abstract class can have Static Methods. The reason for this is Static methods do not work on the instance of the class, they are directly associated with the class itself. So if you write a static method in the class and compile it, and when you try to view the IL, it will be same as any other class accessing the static member.

Can I Create The Constructor Of Static Class

We can create static constructor without any access modifier. We can not create parameter constructor for static class. Please try this public static class MyStaticClass{public static int i;static MyStaticClass(){i = 10;}}

Why private constructor is used ? . Private Constructor: Private Constructors are used to restrict the instantiation of object using 'new' operator.  2. It is commonly used in classes that contain static members only.  3. This type of constructors is mainly used for creating singleton object.  4. If you don't want the class to be inherited we declare its constructor private.  5. We can't initialize the class outside the class or the instance of class can't be created outside if its constructor is declared private. 6.  We have to take help of nested class (Inner Class) or static method to initialize a class having private constructor.

Difference between IEnumerable and Enumerable

IEnumerable is an interface. It is part of system.collection namespace and it has one method, GetEnumerator. This is the method the foreach loop uses loop through a collection and returns an IEnumerator Interface.An Enumerable class is part of Linq library and is a bunch of extension methods written for the IEnumerable interface. It has many extension methods like First, FirstorDefault, Any, Last, LastorDefault, Max, Sum, Average etc..

Can we inherit enum in C#?No, not possible. Enums cannot inherit from other enums. In fact all enums must actually inherit from System.Enum . C# allows syntax to change the underlying representation of the enum values which looks like inheritance, but in actuality they still inherit from System.enum

Can we use await in catch and finally blocks?

Yes, we can use it, from C# 6.0 onward. C# 6 was introduced with Visual Studio 2015. It supports using await in catch and finally blocks.

What is Null Design Patter in C#

Can you declare variable in interfaceWhen you declare a variable, it will give "Interface can not contain a field" which means you can;t declare a variable in an interface. So answer is No, we Can't.

What is the difference between sealed class and abstract class? 1)Sealed class cannot be inherited by a normal class. 1)Abstract class must be inherited by a class.2)Instance must be used for Sealed class for accessing its public methods. 2)Instance cannot be created for Abstract class and it should be inherited for accessing its abstract methods.3)Sealed class methods cannot be override. 3)Abstract class methods can be override.

what is virtual constructor? Constructors cannot be virtual. Declaring a constructor as a virtual function is a syntax error.

Is 'string' a value type or Reference type? string is reference type and default is sealed class. its does not have default allocation size. string is reference type,all primitive data types except general data type(string &object ) are value types.class, interface and delegates are reference type,in which the memory will store in heap memory.structures and enums are value type in which memory will store in stack memory

Tell me use cases of Array and Linked list? Size of array is fixed. size of linked list is dynami

What are the fundamental differences between value types and reference types? C# divides types into two categories - value types and reference types. Most of the intrinsic types (e.g. int, char) are value types. Structs are also value types. Reference types include classes, arrays and strings. The basic idea is straightforward - an instance of a value type represents the actual data, whereas an instance of a reference type represents a pointer or reference to the data.The most confusing aspect of this for C++ developers is that C# has predetermined which types are represented as values, and which are represented as references. A C++ developer expects to take responsibility for this decision.For example, in C++ we can do this:int x1 = 3; // x1 is a value on the stack int \*x2 = new int(3) // x2 is a pointer to a value on the heapbut in C# there is no control:int x1 = 3; // x1 is a value on the stack int x2 = new int(); x2 = 3; // x2 is also a value on the stack!

What’s the difference between the System.Array.CopyTo() and System.Array.Clone()?System.Array.CopyTo() is used when only we have source and destination array.This works in existing array. System.Array.Clone() is used when only we have source and not destination array.This create a new duplicate of array. The Clone() method returns a new array (a shallow copy) object containing all the elements in the original array. The CopyTo() method copies the elements into another existing array. Both perform a shallow copy. A shallow copy means the contents (each array element) contains references to the same object as the elements in the original array. A deep copy (which neither of these methods performs) would create a new instance of each element's object, resulting in a different, yet identacle object.

How do you mark a method obsolete? [Obsolete] public int Foo() {…} or [Obsolete(\”This is a message describing why this method is obsolete\”)] public int Foo() {…}

What is the difference between Finalize() and Dispose() methods? Dispose method will be used to free unmanaged resources like files, database connection etc. we need to write the code manually to call Dispose() method.It belongs IDisposable Interface.Finalize() : This method also free unmanaged resources like database connections, files etc…But Its automatically raised by Garbage Collector.This method belongs to object class. Finalize remove both live & dead objects But dispose can not

What is dynamic keyword ? dynamic is the run time bounded so it will decide the type of value on run time. its dynamic typed variable. Dynamic is a new type available in .NET to bypass compiler check during variable initialization and does it on run time.

What is the default access modifier for a Class in C#?Default access modifier of class is Internal. And private for class member.

What is the Use of a Private Constructor in C#?Private constructor are used in Singleton pattern. Private Constructor will restrict to creat the object of that class. it will leads to protect the data from that class with out accessing from another class.

C# 6.0 New Features

<https://www.codeproject.com/Articles/879887/Csharp-New-Features>

How can we pass parameters to Static Constructors? Static constructor never have parameter. static constructor is called when the execution of the class is started. It is the first member to be execute in the class. at this time how can we pass any parameters to it. if we create object of the class then it will call the non static constructor. so we can not pass any parameters into it at any cost.

LINQ – Lambda Expression vs Query Expression

A lambda expression is a convenient way of defining an anonymous (unnamed) function that can be passed around as a variable or as a parameter to a method call. Many LINQ methods take a function (called a delegate) as a parameter. ... The => operator is called the "lambda operator". What does => mean in C#?The => token is supported in two forms: as the lambda operator and as a separator of a member name and the member implementation in an expression body **definition**

As you’re probably aware of already, LINQ comes in two flavours – using Lambda expressions and using SQL-like query expressions:

|  |  |
| --- | --- |
|  | Func<int, bool> isEven = i => i % 2 == 0;  int[] ints = new int[] { 1, 2, 3, 4, 5, 6, 7, 8, 9 };    // using Query expression  var evensQuery = from i in ints where isEven(i) select i;  // using Lambda expression  var evensLambda = ints.Where(isEven); |

Both yields the same result because query expressions are translated into their lambda expressions before they’re compiled. So performance-wise, there’s no difference whatsoever between the two.

Which one you should use is mostly personal preference, many people prefer lambda expressions because they’re shorter and more concise, but personally I prefer the query syntax having worked extensively with SQL. With that said, it’s important to bear in mind that there are situations where one will be better suited than the other.

// using lambda expression

    var lambda = people.Join(pets,              // outer sequence

                             person => person,  // inner sequence key

                             pet => pet.Owner,  // outer sequence key

                             (person, pet) =>

                                 new { OwnerName = person.Name, Pet = pet.Name });

    // using query expression

    var query = from person in people

                join pet in pets on person equals pet.Owner

                select new { OwnerName = person.Name, Pet = pet.Name };

}

There are a number of methods that are only available with the Lambda expression, Single(), Take(), Skip(), First() just to name a few. Although you can mix and match the two by calling the Lambda-only methods at the end of the query:

Why to use “using” in C#? using keyword can use two places 1.) when adding a namespace 2.) dispose the object automatically when the class implement idisposable interface.

What is Nullable Types in C#? Value types can not store the null values,nullable feature added in c#2.0.now onwords we can able to add null values to value types. ex: int x=null; \\invalid int? x = null; \\ valid

IEnumerable Vs IQueryable? Ienumerable fetches the entire data from the database.It fetches the record one by one.It can Read , but cannot modify and update the data.It is the forward only data,we cannot insert the items at the middle.IQueryable : IQueryable fetches the Record based on filter wise. IQueryable is faster than IEnumerable. IEnumerable loads data in-memory and then apply filters to it one by one but IQueryable apply filters all at once and return the result.

Difference between background worker thread and foreground threads. Backgrounder worker as name states runs separate from main thread and does not hang the main thread so its a async operation and a foreground thread is runs parallel to main thread.in short we can run foreground thread and background worker at the same time together.

When and why you should use 1=1 in WHERE clause? it shows the all records from table. If you don't know  the list of conditions at compile time and it will built at run time, Then you can made a condition with “where 1=1”. and for other conditions that will affect run time, use

What’s a multicast delegate?