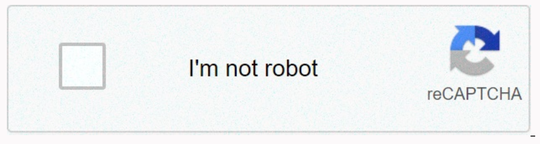

Continue

Array questions and answers in c programming

Often asked basic C # Interview questions about programming and coding: C # is a programmable language that has grown rapidly and is also widely used. It's in high demand, flexibility and multi-platform support as well. It is not only used for windows but many other operating systems. Therefore, it is very important to have a strong understanding of the language to land in any job in the software testing industry. Below enlisted is not only a set of the most frequently asked questions of C # but also some very important topics to be understood to stand out from the crowd of population C #. As C# is a broad topic, to easily solve all the concepts, I've divided this topic into three sections as mentioned below: Questions about basic conceptsQuestions on Arrays and StringsAdvanced ConceptsThis articles include a set of top 50 C #interview questions and answers covering nearly all of its important topics in simple terms, to help you prepare for your interview. What is the most popular C #Interview Questions and AnswersBasic ConceptsQ #1) What is an audience and a class? A: Class is a packing of properties and methods used to represent a real-time real-time domain. It is a data structure that brings all versions together in a single unit. The object is defined as an express of a class. Technically, it is only a block of memory that is spokesman that can be stored in the form of variables, arrays or a collection. Question #2) What are the basic OOP concepts? A: The four basic concepts of Object-Oriented Programming are:Encapsulation: Here, the internal performance of an object is hidden from the view outside the definition of the object. The necessary information can only be accessed while the rest of the data deployment is hidden. Abstraction: It's a process that identifies an object's critical behavior and data and removes unsu related details. Inheritance: It is the ability to create new classes from one class to another. It is done by accessing, modifying and expanding the behavior of objects in the parent class. Polysyming: Name means, one name, multiple forms. It is achieved by having multiple methods of having the same name but different implementations. Question #3) What is managed and un managed code? A: Code management is a code made by CLR (Common Language Runtime) i.e. all application code is based on the .Net platform. It is considered manageable because of the .Net framework that internally uses garbage collectors to clear unused memory. Un managed code is any code made by application run time of any framework other than .Net. Application run time takes care of memory, security, and other performance activities. Question #4) What is the interface? A: The interface is a class with no deployments. The only thing it contains is a statement about methods, attributes, and events. Q#5) What are the different types of classes in C #? Lyrics: The different class types in C# are: Class part: It allows its members it shared with many of .cs friends. It is expressed by the keyword Partial.Sealed class: It is an un inheritable class. To access the members of a closed layer, we need to create objects of the class. It is expressed by the keyword Sealed.Abstract class: It is a class where the object cannot be initialed. Classes can only be inherited. It should contain at least one method. It is expressed by abstract keywords. Static layer: It is a layer that does not allow inheritance. The members of the class were also static. It is expressed by static keywords. This keyword gives the compiler to check for any random cases of static layers. Q #6) Explain compiling the code in C #. Answer: The compiled code in C # consists of four steps: Compile the source code into code management by C#compiler.Combining new code is created into different ass. Loading general language run time (CLR). Performed assembly by CLR. Q#7) What is the difference between a class and a Struct? Answer: Given below what is the difference between a class and a Struct: Q #8) what is the difference between virtual methods and abstract methods? Answer: The Virtual Method must always have a default implementation. However, it can be overwritten in the derivative class, although it is not required. It can be overwritten using keyword override. An abstract method does not have an implementation. It's in the abstract layer. It is imperative that the derived classes perform abstract methods. A keyword override is not required here although it can be used. Q #9) Explain the name space in C #. A: They are used to organize large code projects. The system is the most widely used name space in C#. We can create our own name space and also use a name space in another space, called Nested Name Space. They are expressed by the name space keyword. Q#10) What is the ending used in C#? A: Using keywords that signes that a specific name space is being used by the program. For example, by using SystemHere, the system is a name space. The class console is defined in the System. So we can use console.writeline (....) or readline in our program. Q#11) Abstraction.Answer explanation: Abstraction is one of the OOP concepts. It is used to show only the necessary features of the class and hide unnecessary information. Let us take an example of a car: A driver should know the details about the car such as color, name, mirror, steering wheel, gears, brakes, etc. What he does not need to know is an internal engine, exhaust system. So Abstraction helps in knowing what is needed and hides internal details from the outside world. Hiding internal information can be achieved by declaration of parameters such as Privacy using private keywords. Q#12) Polysoprual explanation? Answer: Programming, figure means the same method but different implementations. It is of 2 types, Compile-time and Runtime.Compile-time polytasm achieved by the operator Polysyming is achieved by overwriting. Inheritance and virtual functionality are used during polysyming runs. For example, if a class has a Void Add() method, polyps are achieved by overloading the method, that is, void Add(int a, int b), void Add(int add) are all overload methods. Q#13) How is exception processing done in C#? Answer: Handling exceptions made using four keywords in C #:try: Contains a block of code for which an exception will be checked.catch: It is a program that catches an exception with the help of handler.finally exception: It is a block of code written to make regardless of whether an exception is caught or not. Throw: Throw an exception when a problem occurs. Q#14) What are grades C# I/O? What are the commonly used I/O classes? Answer: C# has System.IO name spaces, including classes used to perform various activities on files such as creating, deleting, opening, closing, etc. Some commonly used I/O classes are:File – Helps manipulate files. StreamWriter – Used to write characters into streams. StreamReader – Used to read characters into streams. StringWriter - Used to read a string buffer. StringReader – Used to write a string buffer. Paths – Used to perform activities related to path information. Question #15) What is a StreamReader/StreamWriter class? A: StreamReader and StreamWriter are the most named System.IO. They are used when we want to read or write charact90, data-based readers, respectively. Some members of StreamReader are: Close(), Read(), Readline(). StreamWriter members are: Close(), Write(), Writeline(). Class Program1 { using(StreamReader sr = new StreamReader(C:\ReadMe.txt) { //----------code to read----------/// } using(StreamWriter sw = new StreamWriter(C:\ReadMe.txt)) { //----------code to write ----------// } } } Q #16) Deor in C#? Answer: Destructor is used to clean memory and free up resources. But in this #C is done by garbage collectors on their own. System.GC.Collect() is called internally for cleaning. But sometimes it may be necessary to carry out destructors manually. Example:~Car() { Console.writeline(....); } Q#17) What is an abstract class? Answer: An Abstract class is a class that is indicated by abstract keywords and can only be used as a Base class. This class should always be inherited. An express of the class itself cannot be created. If we do not want any program to create an object of a class, then such classes can be made abstract. Any method in the abstract class has no implementation in the same class. But they must be done in the children's class. For example, abstract layer AB1 { Public Void Add(); } ChildClass : AB1 { childClass cs = new childClass (); int Sum = cs. Add(); } All methods in an abstract layer are underground virtual methods. So Virtual keys should not be used with any method in the abstract layer. Q #18) What is boxing and and Convert a value type to a reference type called Boxing.For example, int Value1 -= 10; ————Boxing ————// canned object Value = Value1; Explicit conversion of the same type of reference (generated by boxing) back to the type of value known as Unboxing.For example, //————UnBoxing————// int UnBoxing = int (boxedValue); Question #19) What's the difference between Continue and Break Statement? Answer: Breaks the loop breaking statement. It makes control of the program to exit the loop. Further claims make control of the program to get rid of only the current only time. It doesn't break the loop. Q#20) What is the difference between final and complete blocks? Answer: Finally the block is called after performing the try and catching the block. It is used to handle exceptions. Regardless of whether an exception is captured or not, this block of code will be enforced. Typically, this block will have a code.finalize cleaning method called just before garbage collection. It is used to perform un managerless code cleanup operations. It is automatically called when a certain case is not called afterwards. What are Arrays And StringsQ #21) Arrays? Provides syntax for a single and multi-dimensional array? A: An array is used to store multiple variables of the same type. It is a set of variables stored in a adjacent memory location. Example: double number = new double number[10]; int[] score = new int[4] {25,24,23,25}; A single dimensional array is a linear array where variables are stored in a single row. The example above is a single dimensional array. Arrays can be more than one size. Multi-dimensional arrays are also known as rectangular arrays. For example, the new int[,] = int[3.2] { {1.2} , {2,3},{3.4} }; Q #22) What is jagged array? Answer: A jagged array is an array with elements that are arrays. It is also called array array. It could be a single dimensions.int many jagged albums. Q #23) Name some array.Answer properties: Array properties include:Length: Get the total number of elements in an array. IsFixedSize: Shows whether the array is fixed in size or not. IsReadOnly: Indicates whether the array is read-only or not. Q #24) What is the Array Class? A: An Array class is the base class for all arrays. It offers many properties and methods. It's in the nameless system. Q #25) What is a series? What are the properties of a String Class? Answer: A string is a set of char objects. We can also declare string variables in the name c#.string = C# Questions; A string layer in C# represents a string. The attributes of the string class are:Chars get the Char object in the current String.Length gets the number of objects in the current String.Q #26) What is an Escape Sequence? Name some strings that exit the string in C#. A: An Exit string is shown with an inverted slash (\). A reverse slash indicates that the character that follows it should be explained literally or that it is a special character. An escape sequence is as a single character. The string escape sequence is as follows: – Newline character\b - Backspace\\ – Backslash\' – Single quote\" – Double QuoteQ #27) What is a formal expression? Looking for a string using a formal expression? A: A formal expression is a template to match a set of inputs. Models can include operators, builders or literals characters. Regex is used to analyze strings and replace character strings. For example* matches the previous character not or multiple times. So a*b regex is equivalent to b, ab, aab, aaab and so on. Search for a string using Regex: Static void Main(string[] args) { string[] languages = { C#, Python, Java }; foreach(string s in languages) { if(System.Text.RegularExpressions.Regex.IsMatch(s,Python,)) { Console.WriteLine(Match found); } } } The example on python search versus set of inputs from the language array. It uses Regex.IsMatch to return the correct in case if the model is found in the input. A template can be any formal expression that represents the input we want to match. Q#28) What are the basic chain operations? Explain.Answer: Some of the basic string operations are:Connectors: Two strings can be joined either by using a System.String.Concat or by using + operator. Modification: Replace(a,b) is used to replace a string with another string. Trim() is used to cut strings at the end or at the top.Compare: System.StringComparison() is used to compare two strings, either as a word comparison or not in sensitive cases. Mostly take two parameters, the original string, and the string to be compared with. Search: StartWith, EndsWith method used to search for a specific string. Q #29) What is a syntax analysis? How do I erly analyze a date time series? A: The analysis of the syntax converts a string into a different type of data. Example: text string = 500; int num = int. Analysis of syntax (text); 500 is an indi number. Therefore, the Method of Syntax Analysis converts the 500 string into its own base type, i.e. int. Follow the same methods to convert a DateTime series. dateTime series = January 1, 2018; DateTime parsingValue = DateTime.Parse(dateTime); What are Advanced ConceptsQ #30) Delegates? Explain.Answer: A delegate is a variable that keeps reference to a method. Therefore it is a functional cursor or reference type. All delegates are derived from the name space system.delegate. Both the delegate and the method by which it refers may have the same signature. Declare a delegate: public delegate void AddNumbers(int n); After a delegate's statement, the object must be created by the delegate using the new keyword. AddNumbers an1 = AddNumbers(number) new; The delegates provide a kind of packaging to the reference method, which internally will be called when a delegate is called. public delegates int myDel (int number); public class Program { public int AddNumbers(int { int Sum = a + 10; return Sum; } public void Start() { myDel DelgateExample = AddNumbers; } } In the example above, we have a representative myDel that has an inso number value as The class program has a method of signature similar to the delegates, called AddNumbers(). If there is another method called Start() that creates an object of delegates, then the object can be assigned to AddNumbers because it has the same signature as that of the delegate. Q #31) What is an event? A: An event is a user's action that creates a notification for an app that the app must respond to. User actions can be mouse movement, keystrokes, and so on. Programming, a class that raises an event called a publishing house and a class that responds/receives an event called a subscription. The event should have at least one other subscription for which the event was never raised. Delegates used to declare Events.Public delegates void PrintNumbers(); PrintNumbers myEvent event; Question #32) How do I use delegates with events? A: Delegates are used to enhance events and handle them. Always a delegate needs to be declared first and then the facts are declared. Let's take a look at an example:Consider a class called Patient. Consider two other classes, Insurance, and Bank that require patient mortality information from the patient class. Here, Insurance and The Bank are re enrollees and the Patient class becomes a Publishing House. It causes death events and the other two classes will receive this event. consoleApp2 { public class Patient { public class void deathInfo();/Declaring a Delegate// public event deathInfo deathDate;//Declaring the event// public void Death() { deathDate(); } public class Insurance { Patient myPat = new Patient(); void GetDeathDetails() { //--------Do Something with the deathDate event------------// } void Main() { //-------- (Subscribe function GetDeathDetails----------// myPat.deathDate += GetDeathDetails; } public class Bank { Patient myPat = new Patient(); void GetPatInfo () { //-------- --------Do Something with the deathDate event------------// } void Main() { //--------Subscribe the function GetPatInfo ----------// myPat.death +Date= GetPatInfo; } } } } Q#33) What are the different types of delegates? Answer: Different types of delegates are:unique delegates: a delegate can call a single method. Multi-directional delegate play: a delegate can call many methods. + and – operators are used to register and cancel the corresponding subscription. Generic delegates: It does not require an expression of defined delegates. It's three categories, Action, Funcs and Predicate.Action- In the example above of delegates and events, we can replace the definition of delegates and events using action keywords. Delegate action defines a method that can be called an estp but does not return a resultPublic delegates void deathInfo(); Public events deathInfo deathDate; Replace it with Action // Public Action Event DeathDate; The implicit action refers to a delegate. Func- A Func delegate defines a can be called an math and return results. Func <int, string,= bool=> myDel is the same as myDel bool delegates (int a, string b); Predicate- Define a method that can </int,> </int,> calls for 100 opponentss and always returns bool. Predicate&string> myDel is the same as myDel bool delegates (series s); Q#34) What does a multi-directional broadcasting representative mean? Answer: A delegate points to more than one method known as a multi-directional speech delegate. Multi-directional play is achieved using the + and +=. Consider examples from Q #32.There are two subscribers for deathEvent, GetPatInfo, and GetDeathDetails. And so we used += the algorithm. It means that whenever myDel is called, both subscribers are called. Delegates will be called in the order in which they are added. Q#35) Explain the publishing house and subscribers in Events.Answer: The publishing house is a class responsible for publishing a message of different types of other classes. What is the message, but the event as discussed in the above questions. From Example in Q&#32, Class Patient is the Publisher class. It is creating a deathEvent event, received by other classes. Subscribers capture the message of the kind that it cares about. Again, from the example of Q#32, Class Insurance and Bank are subscribers. They are interested in deathEvent events of the void type. Q#36) What are sync and asym sync operations? Answer: Syncing is a way to create a secure theme code where only one theme can access resources at any given time. Asym sync calls wait for the method to complete before continuing with the program flow. Synchronized programming adversely affects UI activities when users try to perform time-consuming activities because only one theme will be used. In asym sync operations, the call method will immediately return so that the program can perform other operations while the method is called completing its work in certain situations. In C#, the keywords Async and Await are used to achieve asym sync asym sync programming. See Q#43 for more details on synchronized programming. Q#37) What is reflected #là C? A: Reflection is the ability of a code to access the council's metadata during run time. A program reflects upon itself and uses metadata to notify the user or modify its behavior. Metadata refers to information about objects and methods. The System.Reflection name space contains methods and information management classes of all types of feeds and methods. It is mainly used for window applications, for example, to view the properties of a button in a window form. The MemberInfo object of class reflection is used to explore the attributes associated with a class. Reflections are performed in two steps, first we get the object type, and then we use the type to identify members as methods and attributes. To get the type of a class we just need to use,Type mytype = myClass.GetType(); Once we have class type, other information about the class is easily accessible. System.Reflection.MemberInfo Information = mytype. GetMethod(AddNumbers); Above the statement tries to find a method named AddNumbers in the myClass.Q class #38) What is Generic </string> </string> Generics or generic classes are used to create classes or objects without any particular type of data. The data type can be specified during run time, i.e. when it is used in the program. Example: So from the code above, we see 2 comparisons of the original methods, to compare strings and int. In the case of comparing other data type parameters, instead of creating multiple overload methods, we can create a common class and pass an alternate data type, i.e. T acts as a datatype until it is used specifically in the Main() method. Q #39) Explain Get and Set Accessor Properties? Answer: Get and Set is called Accessors. They are used by Properties. Assets provide a mechanism for reading and writing the value of a private sector. To access that private sector, these accessors are used. Get Property is used to return the value of a Set Property accessor property used to set the value. The use of receive and setup is as follows: Q #40) What is thread? What is Multithreading? Answer: A Thread is a set of instructions that can be made, which will allow our program to perform simultaneous processing. Simultaneous processing helps us to perform multiple operations at once. By default, C# has only one theme. But other themes can be created to make the code parallel to the original theme. Thread has a life cycle. It starts whenever a theme class is created and is terminated after implementation. System.Threading is the name space that needs to be included to create the thread and use its members. The theme was created by expanding the Thread Class. Start() method used to start implementing the thread.//CallThread is the target method// ThreadStart methodThread = new ThreadStart(CallThread); Thread childThread = new Thread(methodThread); childThread.Start(); C# can perform multiple tasks at once. This is done by handling different processes by different themes. This is called MultiThreading.There are several threaded methods used to handle multi-threaded operations:Start, Sleep, Cancel, Suspend, Continue and Join.Most of these methods are self-solved. Q #41) Name some of the properties of the class thread. Answer: some of the attributes of the subject class are:IsAlive - which contains the correct value when a subject is Active.Name - can return the names of the themes. Additionally, a name can be set for the theme. Priority – returns the priority value of the task set by the operating system. IsBackground - be or set a value that indicates whether a theme should be a background process or foreground. ThreadState- describes the topic status. Q#42) What are the different states of a topic? Answer: The different state of a topic is: Unstarted - Thread is created. Run – Thread starts. WaitSleepJoin - Sleep call thread, calls waiting on an object and calls participate in a topic. Suspended - The subject has been suspended. Cancel - The theme is dead but does not change the stop state. Stopped – Stream stopped. Q #43) What is Async and Await? A: Asym sync and Pending keywords are used to create Methods in C.Asynchronous programming mean that the process runs independently of the main or other processes. Use Async and Await as shown below: Asym sync keyword is used for method declaration. The number is of a task of the int type that calls the CalculateCount() method. Calculatecount() begins to implement and calculate something. Independent work is done on my subject and then waiting for the counting statement to be reached. If Calculatecount is not complete, myMethod will return to its calling method, so the main theme is not blocked. If Calculatecount is complete, then we have the results available when the control reaches pending counting. So the next step will continue in the same topic. However, it is not the situation in the above case that the delay of 1 second is related. Q#44) What is a deadlock? Answer: A deadlock is a situation where a process cannot be completed because two or more processes are waiting for each other to finish. This usually occurs in multiple streams. Here a shared resource is being organized by a process and a process is waiting for the first process to release it and the subject holding the locked item is waiting for a process to complete. Consider the example below: Perform an objB access task and wait for 1 second. Meanwhile, PerformtaskB tries to access ObjA.After 1 second, PeformtaskA tries to access ObjA locked by PerformtaskB.PerformtaskB tries to access ObjB locked by PerformtaskA.This creates Deadlock.Q #45) Explains the key, screen and Mutex object in Threading.Answer: Keyword key ensures that only one thread can enter a specific part of the code at any time. In the example above, lock (ObjA) means that the lock is placed on ObjA until the process releases it, no other theme can access ObjA.Mutex just like a lock but it can work on multiple processes at once. WaitOne() is used for locking and ReleaseMutex() is used to issue keys. But Mutex is slower than lock because it takes time to get and release it. Monitor.Enter and Monitor.Exit perform internal keys. lock is a shortcut to Monitor. lock(objA) internal calls. Monitor.enter(ObjA); try { } Finally {Monitor.Exit(ObjA));} Q #46) What are racial conditions? Ans: Race conditions occur when two topics access the same resource and are trying to change it at the same time. Topics that will have access to the first resource cannot be predicted. If we have two themes, T1 and T2, and they are trying to access a shared resource called X. And if both themes try to write a value to X, the final written value for X will be saved. Q #47) What is Thread Pooling? Ans: Pool theme is a collection of themes. The theme can be used to perform tasks without disturbing the main theme. Once the topics complete the task, theme back to the pool. System.Threading.ThreadPool Space Name has classes that manage the topic in the pool and its it System.Threading.WaitCallback (SomeTask)); The line above queues a task. What is someTask method that should have a parameters of this type of Object.Q #48) Serialization? A: Serialization is a process of converting code to its binary format. Once it is converted into a byte, it can be easily stored and burned to a disc or any such storage device. Serializations are mostly useful when we don't want to lose the original form of the code and it can be taken at any time in the future. Any class marked with the [Serializable] property will be converted to its binary form. The reverse process of getting the C# code back from binary forms is called Deserialization.To Serialize an object we need the object to be serialized, a line that can contain serialized objects and the name space System.Runtime.Serialization may contain classes for serialization. Q #49) What are the types of Serialization? Answer: The different types of serialization are: SERIAL XML - It serializes all public properties into XML documents. Since the data is in XML format, it can be easily read and manipulated in different formats. Classes are located in System.sml.Serialization.SOAP - Classes residing in System.Runtime.Serialization. Similar to XML but creating a complete SOAP compliant envelope can be used by any system that understands soap. Binary Serialization - Allows any code to be converted to its binary form. Public and non-public properties can be posted and restored. It's faster and takes up less space. Q #50) What is an XSD file? A: The XSD file stands for XML Diagram Definition. It gives a structure for XML files. It means that it decides what xml elements should be and in order and what attributes should be present. If no XSD files are associated with XML, XML can have any tags, any properties, and any components. Xsd.exe tool converts files to XSD format. In the serialization of C#code, classes are converted to XSD-compatible formats by xsd.exe.ConclusionC # is growing rapidly day by day and it plays an important role in the industry. I checked the software. I am sure that this article will make your preparation for the interview much easier and give you a reasonable amount of knowledge of most #C topics. Hope you'll be ready to face any C#interview confidently!! confidence!!