Q1. What Is ‘this’ Pointer?

‘this’ pointer refers to the current object of a class.

Q2. Difference Between Const, Readonly And Static Readonly In C# ?

* Constant variables are declared and initialized at compile time. The value can’t be changed after words.
* Readonly variables will be initialized only from the non-static constructor of the same class.
* Static Readonly variables will be initialized only from the static constructor of the same class.

Q3. Is It Possible For A Class To Inherit The Constructor Of Its Base Class?

No. A class cannot inherit the constructor of its base class.

Q4. Is It Possible To Execute Two Catch Blocks?

No. Whenever, an exception occurs in your program, the correct catch block is executed and the control goes to the finally block.

Q5. You Declare An Overridden Method To Be Static If The Original Method Is Not Static?

No. Two virtual methods must have the same signature.

Q6. What’s The Difference Between The ‘ref’ And ‘out’ Keywords?

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, and it will be initialized inside the method.

Q7. Can You Inherit Private Members Of A Class?

No, we cannot inherit private members of a class because private members are accessible only to that class and not outside that class.

Q8. Can You Allow A Class To Be Inherited, But Prevent A Method From Being Overridden In C#?

Yes. By Declaring the class public and making the method sealed.

Q9. Can You Prevent A Class From Overriding?

Yes. You can prevent a class from overriding in C# by using the sealed keyword; and in VB by using NotInheritable keyword.

Q10. Why To Use “using” Statement In C#?

The using block is used to obtain a resource and use it and then automatically dispose off when the execution of block completed.

Q11. Can ‘this’ Be Used Within A Static Method?

No.

Q1. What’s The Role Of The Datareader Class In Ado.net Connections?

It returns a read-only dataset from the data source when the command is executed.

Q2. When Do We Generally Use Destructors To Release Resources?

If the application uses unmanaged resources such as windows, files, and network connections, we use  
destructors to release resources.

Q3. How Do I Get Deterministic Finalization In C#?

In a garbage collected environment, it’s impossible to get true determinism. However, a design pattern that we recommend is implementing IDisposable on any class that contains a critical resource. Whenever this class is consumed, it may be placed in a using statement, as shown in the following example:  
using(FileStream myFile = File.Open(@”c:temptest.txt”, FileMode.Open))  
{  
int fileOffset = 0;  
while(fileOffset < myFile.Length)  
{  
Console.Write((char)myFile.ReadByte());  
fileOffset++;  
}  
}  
When myFile leaves the lexical scope of the using, its dispose method will be called.

Q4. How Do You Determine Whether A String Represents A Numeric Value?

To determine whether a String represents a numeric value use TryParse method as shown in the example below. If the string contains nonnumeric characters or the numeric value is too large or too small for the particular type you have specified, TryParse returns false and sets the out parameter to zero. Otherwise, it returns true and sets the out parameter to the numeric value of the string.

string str = “One”;  
int i = 0;  
if(int.TryParse(str,out i))  
{  
Console.WriteLine(“Yes string contains Integer and it is ” + i);  
}  
else  
{  
Console.WriteLine(“string does not contain Integer”);  
}

Q5. A Class Inherits From 2 Interfaces And Both The Interfaces Have The Same Method Name As Shown Below. How Should The Class Implement The Drive Method For Both Car And Bus Interface?

namespace Interfaces  
{  
interface Car  
{  
void Drive();  
}  
interface Bus  
{  
void Drive();  
}  
class Demo : Car,Bus  
{  
//How to implement the Drive() Method inherited from Bus and Car  
}  
}

To implement the Drive() method use the fully qualified name as shown in the example below. To call the respective interface drive method type cast the demo object to the respective interface and then call the drive method.

using System;  
namespace Interfaces  
{  
interface Car  
{  
void Drive();  
}  
interface Bus  
{  
void Drive();  
}  
class Demo : Car,Bus  
{  
void Car.Drive()  
{  
Console.WriteLine(“Drive Car”);  
}  
void Bus.Drive()  
{  
Console.WriteLine(“Drive Bus”);  
}  
static void Main()  
{  
Demo DemoObject = new Demo();  
((Car)DemoObject).Drive();  
((Bus)DemoObject).Drive();  
}  
}  
}

Q6. What Are Constants In C#?

Constants in C# are immutable values which are known at compile time and do not change for the life of the program. Constants are declared using the const keyword. Constants must be initialized as they are declared. You cannot assign a value to a constant after it isdeclared. An example is shown below.

using System;  
class Circle  
{  
public const double PI = 3.14;  
public Circle()  
{  
//Error : You can only assign a value to a constant field at the time of declaration  
//PI = 3.15;  
}  
}  
class MainClass  
{  
public static void Main()  
{  
Console.WriteLine(Circle.PI);  
}  
}

Q7. Will The Following Code Compile And Run?

string str = null;  
Console.WriteLine(str.Length);  
The above code will compile, but at runtime System.NullReferenceException will be thrown

Q8. Are C# References The Same As C++ References?

Not quite. The basic idea is the same, but one significant difference is that C# references can be null . So you cannot rely on a C# reference pointing to a valid object. In that respect a C# reference is more like a C++ pointer than a C++ reference. If you try to use a null reference, a NullReferenceException is thrown.

For example, look at the following method:  
void displayStringLength( string s )  
{  
Console.WriteLine( “String is length {0}”, s.Length );  
}  
The problem with this method is that it will throw a NullReferenceException if called like this:  
string s = null;  
displayStringLength( s );

Of course for some situations you may deem a NullReferenceException to be a perfectly acceptable outcome, but in this case it might be better to re-write the method like this:  
void displayStringLength( string s )  
{  
if( s == null )  
Console.WriteLine( “String is null” );  
else  
Console.WriteLine( “String is length {0}”, s.Length );  
}

Q9. What Are Three Test Cases You Should Go Through In Unit Testing?

Positive test cases (correct data, correct output), negative test cases (broken or missing data, proper handling), exception test cases (exceptions are thrown and caught properly).

Q10. Can You Instantiate A Struct Without Using A New Operator In C#?

Yes, you can instantiate a struct without using a new operator.

Q11. Is It True That All C# Types Derive From A Common Base Class?

Yes and no. All types can be treated as if they derive from object (System.Object), but in order to treat an instance of a value type (e.g. int, float) as object-derived, the instance must be converted to a reference type using a process called ‘boxing’. In theory a developer can forget about this and let the run-time worry about when the conversion is necessary, but in reality this implicit conversion can have side-effects that may trip up the unwary.

Q12. What Do You Know About .net Assemblies?

Assemblies are the smallest units of versioning and deployment in the .NET application. Assemblies are also the building blocks for programs such as Web services, Windows services, serviced components, and .NET remoting applications.

Q13. What Are Access Modifiers Used For?

Access Modifiers are used to control the accessibilty of types and members with in the types.

Q14. What Is The Difference Between A Struct And A Class In C#?

From language spec:  
The list of similarities between classes and structs is as follows. Longstructs can implement interfaces and can have the same kinds of members as classes. Structs differ from classes in several important ways; however, structs are value types rather than reference types, and inheritance is not supported for structs. Struct values are stored on the stack or in-line. Careful programmers can sometimes enhance performance through judicious use of structs. For example, the use of a struct rather than a class for a Point can make a large difference in the number of memory allocations performed at runtime. The program below creates and initializes an array of 100 points. With Point implemented as a class, 101 separate objects are instantiated-one for the array and one each for the 100 elements.

Q15. Is It Possible To Have Different Access Modifiers On The Get/set Methods Of A Property?

No. The access modifier on a property applies to both its get and set accessors. What you need to do if you want them to be different is make the property read-only (by only providing a get accessor) and create a private/internal set method that is separate from the property.

Q16. If You Define A User Defined Data Type By Using The Struct Keyword, Is It A Value Type Or Reference Type?

Value Type.

Q17. What Is The Difference Between String Keyword And System.string Class?

string keyword is an alias for Syste.String class. Therefore, System.String and string keyword are the same, and you can use whichever naming convention you prefer. The String class provides many methods for safely creating, manipulating, and comparing strings.

Q18. Can Structs In C# Have Destructors?

No, structs can have constructors but not destructors, only classes can have destructors.

Q19. If C# Destructors Are So Different To C++ Destructors, Why Did Ms Use The Same Syntax?

Presumably they wanted C++ programmers to feel at home. I think they made a mistake.

Q20. Explain What Is An Interface In C#?

An Interface in C# is created using the interface keyword. An example is shown below.

using System;  
namespace Interfaces  
{  
interface IBankCustomer  
{  
void DepositMoney();  
void WithdrawMoney();  
}  
public class Demo : IBankCustomer  
{  
public void DepositMoney()  
{  
Console.WriteLine(“Deposit Money”);  
}  
public void WithdrawMoney()  
{  
Console.WriteLine(“Withdraw Money”);  
}  
public static void Main()  
{  
Demo DemoObject = new Demo();  
DemoObject.DepositMoney();  
DemoObject.WithdrawMoney();  
}  
}  
}

In our example we created IBankCustomer interface. The interface declares 2 methods.  
@void DepositMoney();  
@void WithdrawMoney();

Notice that method declarations does not have access modifiers like public, private, etc. By default all interface members are public. It is a compile time error to use access modifiers on interface member declarations. Also notice that the interface methods have only declarations and not implementation. It is a compile time error to provide implementation for any interface member. In our example as the Demo class is inherited from the IBankCustomer interface, the Demo class has to provide the implementation for both the methods (WithdrawMoney() and DepositMoney()) that is inherited from the interface. If the class fails to provide implementation for any of the inherited interface member, a compile time error will be generated. Interfaces can consist of methods, properties, events, indexers, or any combination of those four member types. When a class or a struct inherits an interface, the class or struct must provide implementation for all of the members declared in the interface. The interface itself provides no functionality that a class or struct can inherit in the way that base class functionality can be inherited. However, if a base class implements an interface, the derived class inherits that implementation.

Q21. Explain The Three Services Model Commonly Know As A Three-tier Application.

Presentation (UI), Business (logic and underlying code) and Data (from storage or other sources).

Q22. Where Is The Output Of Textwritertracelistener Redirected?

To the Console or a text file depending on the parameter passed to the constructor.

Q23. What’s The .net Datatype That Allows The Retrieval Of Data By A Unique Key?

HashTable.

Q24. Difference Between A Sub And A Function.

A Sub does not return anything whereas a Function returns something.

-A Sub Procedure is a method will not return a value  
-A sub procedure will be defined with a “Sub” keyword

Sub ShowName(ByVal myName As String)  
Console.WriteLine(”My name is: ” & myName)  
End Sub

-A function is a method that will return value(s).  
-A function will be defined with a “Function” keyword

Function FindSum(ByVal num1 As Integer, ByVal num2 As Integer) As Integer  
Dim sum As Integer = num1 + num2  
Return sum  
End Function

Q25. Can You Declare An Override Method To Be Static If The Original Method Is Not Static?

No. The signature of the virtual method must remain the same. (Note: Only the keyword virtual is changed to keyword override)

Q26. What Does The Keyword “virtual” Declare For A Method Or Property?

The method or property can be overridden.

Q27. Give An Example To Show For Hiding Base Class Methods?

Use the new keyword to hide a base class method in the derived class as shown in the example below.  
using System;  
public class BaseClass  
{  
public virtual void Method()  
{  
Console.WriteLine(“I am a base class method.”);  
}  
}  
public class DerivedClass : BaseClass  
{  
public new void Method()  
{  
Console.WriteLine(“I am a child class method.”);  
}  
public static void Main()  
{  
DerivedClass DC = new DerivedClass();  
DC.Method();  
}  
}

Q28. Structs Are Not Reference Types. Can Structs Have Constructors?

Yes, even though Structs are not reference types, structs can have constructors.

Q29. Is There A Way Of Specifying Which Block Or Loop To Break Out Of When Working With Nested Loops?

The easiest way is to use goto:  
using System;  
class BreakExample  
{  
public static void Main(String[] args)  
{  
for(int i=0; i<3; i++)  
{  
Console.WriteLine(“Pass {0}: “, i);  
for( int j=0 ; j<100 ; j++ )  
{  
if ( j == 10) goto done;  
Console.WriteLine(“{0} “, j);  
}  
Console.WriteLine(“This will not print”);  
}  
done:  
Console.WriteLine(“Loops complete.”);  
}  
}

Q30. If A Child Class Instance Is Created, Which Class Constructor Is Called First – Base Class Or Child Class?

When an instance of a child class is created, the base class constructor is called before the child class constructor. An example is shown below.

using System;  
namespace TestConsole  
{  
class BaseClass  
{  
public BaseClass()  
{  
Console.WriteLine(“I am a base class constructor”);  
}  
}  
class ChildClass : BaseClass  
{  
public ChildClass()  
{  
Console.WriteLine(“I am a child class constructor”);  
}  
public static void Main()  
{  
ChildClass CC = new ChildClass();  
}  
}  
}

Q31. How Do I Create A Delegate/multicastdelegate?

C# requires only a single parameter for delegates: the method address. Unlike other languages, where the programmer must specify an object reference and the method to invoke, C# can infer both pieces of information by just specifying the method’s name. For example, let’s use System.Threading.ThreadStart: Foo MyFoo = new Foo(); ThreadStart del = new ThreadStart(MyFoo.Baz); This me that delegates can invoke static class methods and instance methods with the exact same syntax!

Q32. Does C# Support Multiple Inheritance?

No, use interfaces instead.

Q33. Will The Finally Block Get Executed If An Exception Has Not Occurred?

Yes. Finally block always get executed.

Q34. What Are Access Modifiers In C#?

In C# there are 5 different types of Access Modifiers.  
Public  
The public type or member can be accessed by any other code in the same assembly or another assembly that references it.

Private  
The type or member can only be accessed by code in the same class or struct.

Protected  
The type or member can only be accessed by code in the same class or struct, or in a derived class.

Internal  
The type or member can be accessed by any code in the same assembly, but not from another assembly.

Protected Internal  
The type or member can be accessed by any code in the same assembly, or by any derived class in another assembly.

Q35. Difference Between Imperative And Interrogative Code.

There are imperative and interrogative functions. Imperative functions are the one which return a value while the interrogative functions do not return a value.

Q36. Is It Possible To Force Garbage Collector To Run?

Yes, it possible to force garbage collector to run by calling the Collect() method, but this is not considered a good practice because this might create a performance over head. Usually the programmer has no control over when the garbage collector runs. The garbage collector checks for objects that are no longer being used by the application. If it considers an object eligible for destruction, it calls the destructor(if there is one) and reclaims the memory used to store the object.

Q37. What Types Of Object Can I Throw As Exceptions?

Only instances of the System.Exception classes, or classes derived from System.Exception. This is in sharp contrast with C++ where instances of almost any type can be thrown.

Q38. What Is A Partial Class. Give An Example?

A partial class is a class whose definition is present in 2 or more files. Each source file contains a section of the class, and all parts are combined when the application is compiled. To split a class definition, use the partial keyword as shown in the example below. Student class is split into 2 parts. The first part defines the study() method and the second part defines the Play() method. When we compile this program both the parts will be combined and compiled. Note that both the parts uses partial keyword and public access modifier.

using System;  
namespace PartialClass  
{  
public partial class Student  
{  
public void Study()  
{  
Console.WriteLine(“I am studying”);  
}  
}  
public partial class Student  
{  
public void Play()  
{  
Console.WriteLine(“I am Playing”);  
}  
}  
public class Demo  
{  
public static void Main()  
{  
Student StudentObject = new Student();  
StudentObject.Study();  
StudentObject.Play();  
} }}

It is very important to keep the following points in mind when creating partial classes.  
@All the parts must use the partial keyword.  
@All the parts must be available at compile time to form the final class.  
@All the parts must have the same access modifiers – public, private, protected etc.  
@Any class members declared in a partial definition are available to all the other parts.  
@The final class is the combination of all the parts at compile time.

Q39. What Happens If A Static Constructor Throws An Exception?

If a static constructor throws an exception, the runtime will not invoke it a second time, and the type will remain uninitialized for the lifetime of the application domain in which your program is running.

Q40. What Is The Wildcard Character In Sql?

Let’s say you want to query database with LIKE for all employees whose name starts with La. The wildcard character is %, the proper query with LIKE would involve ‘La%’.

Q41. What Happens In Memory When You Box And Unbox A Value-type?

Boxing converts a value-type to a reference-type, thus storing the object on the heap. Unboxing converts a reference-type to a value-type, thus storing the value on the stack.

Q42. Describe The Accessibility Modifier Protected Internal?

It is available to derived classes and classes within the same Assembly (and naturally from the base class it is declared in).

Q43. Can You Declare A Field Readonly?

Yes, a field can be declared readonly. A read-only field can only be assigned a value during initialization or in a constructor. An example is shown below.

using System;  
class Area  
{  
public readonly double PI = 3.14;  
}  
class MainClass  
{  
public static void Main()  
{  
Area A = new Area();  
Console.WriteLine(A.PI);  
}  
}

Q44. Who Is A Protected Class-level Variable Available To?

It is available to any sub-class (a class inheriting this class).

Q45. Can You Mark Static Constructor With Access Modifiers?

No, we cannot use access modifiers on static constructor.

Q46. Is The Following Code Legal?

using System;  
namespace Demo  
{  
class Program  
{  
public static void Main()  
{  
}  
public void Sum(int FirstNumber, int SecondNumber)  
{  
int Result = FirstNumber + SecondNumber;  
}  
public int Sum(int FirstNumber, int SecondNumber)  
{  
int Result = FirstNumber + SecondNumber;  
}  
}  
}

No, The above code does not compile. You cannot overload a method based on the return type. To overload a method in C# either the number or type of parameters should be different. In general the return type of a method is not part of the signature of the method for the purposes of method overloading. However, it is part of the signature of the method when determining the compatibility between a delegate and the method that it points to.

Q47. What’s The C# Equivalent Of C++ Catch (…), Which Was A Catch-all Statement For Any Possible Exception?

A catch block that catches the exception of type System.Exception. You can also omit the parameter data type in this case and just write catch {}.

Q48. What Is Wrong With The Sample Program Below?

using System;  
class Area  
{  
public const double PI = 3.14;  
static Area()  
{  
Area.PI = 3.15;  
}  
}  
class MainClass  
{  
public static void Main()  
{  
Console.WriteLine(Area.PI);  
}  
}  
You cannot assign a value to the constant PI field.

Q49. Where’s Global Assembly Cache Located On The System?

Usually C:winntassembly or C:windowsassembly.

Q50. What Does Protected Internal Access Modifier Mean?

The protected internal access me protected OR internal, not protected AND internal. In simple terms, a protected internal member is accessible from any class in the same assembly, including derived classes. To limit accessibility to only derived classes in the same assembly, declare the class itself internal, and declare its members as protected.