**LEVEL-1:EASY**

**C# .NET**

**80 INTERVIEW QUESTIONS**

**AND**

**ANSWERS**

**Done by Babika**

**KloudOne( 3rd Batch)**

**Q-1) What is the .NET framework?**

The .NET framework supports an object-oriented approach that is used for building applications on windows. It supports various languages like C#, VB, Cobol, Perl, .NET, etc. It has a wide variety of tools and functionalities like class, library and APIs that are used to build, deploy and run web services and different applications.

**Q-2) Mention what are the main components of .Net framework?**

The main components of .Net framework are:

* Common Language Runtime (CLR)
* .Net Framework Class Library (FCL)
* Application Domains
* Runtime Host
* Cross-Language Interoperability
* Side-by-Side Execution
* Profiling
* Dynamic Language Runtime (DLR)
* Common Type System(CTS)
* Metadata and Self-Describing Components
* .Net Framework Security
* Model View Presenter (MVP) Architecture

**Q-3) What do you know about CTS?**

CTS stands for Common Type System. It follows certain rules according to which a data type should be declared and used in the program code. CTS also describes the data types that are going to be used in the application. We can even make our own classes and functions following the rules in the CTS, it helps in calling the data type declared in one program language by other programming languages.

**Q-4) What is CLR?**

CLR stands for common language run-time, it is an important component of the .NET framework. We can use CLR as a building block of various applications and provides a secure execution environment for applications.

Whenever an application written in C# is compiled, the code is converted into an intermediate language. After this, the code is targeted to CLR which then performs several operations like memory management, security checks, loading assemblies, and thread management

**Q -5) What is CLS?**

CLS stands for Common Language Specification. With the rules mentioned under CLS, the developers are made to use the components that are inter-language compatible. They are reusable across all the .Net Compliant languages.

**Q -6) What is JIT?**

JIT stands for Just In Time. JIT is a compiler that converts Intermediate Language to a Native code. The code is converted into Native language during execution. Native code is nothing but hardware specifications that can be read by the CPU. The native

**Q-7) Why do we use Response.Output.Write()?**

Response.Output.Write() is used to get the formatted output.

**Q-8) Mention what is MSIL in .NET ?**

MSIL stands for Microsoft Intermediate Language.

During the compile time, the source code is converted into Microsoft Intermediate Language (MSIL) by compiler.MSIL is a CPU-independent set of instructions that can be efficiently converted to the native code.MSIL provides instructions for calling methods, initializing and storing values, operations such as memory handling, exception handling and so on. All .Net codes are first compiled to IL.

**Q-9) What is an IL?**

Intermediate Language is also known as MSIL (Microsoft Intermediate Language) or CIL (Common Intermediate Language). All .NET source code is compiled to IL. IL is then converted to machine code at the point where the software is installed, or at run-time by a Just-In-Time (JIT) compiler.

**Q-10) What are the Application of .NET Framework?**

* 1. Console Application
  2. Windows Application
  3. Web Application
  4. Mobile Application

**Q-11) Differentiate BCL and FCL?**

|  |  |
| --- | --- |
| **Base Class** | **Framework Class** |
| Bass Class Library(BCL) contains the basic and fundamental types of .net like types, collections, i.e, related functionality, security etc., | Framework Class Library(FCL) includes everything. It is BCL plus other libraries in .net other than BCL like windows form, asp.net, ado.net, etc.. |
| It is fixed or std library | It is run-time library |

**Q-12) What is Boxing and Unboxing?**

Boxing and Unboxing are the process that enables value types(for example, integers) to be treated as reference type (objects).The value is “boxed” inside an object and subsequently “unboxed” back to a value type.

Boxing is Implicit and Unboxing must be Explicit.

**Example:**

int i=166; //A value type

object box=i; //Boxing

int j=(int)box; //Unboxing

**Q -13) What is meant by Managed and Unmanaged code?**

**Managed code:** The code that is managed by the CLR is called Managed code. This code runs inside the CLR. Hence, it is necessary to install the .Net framework in order to execute the managed code. CLR manages the memory through garbage collection and also uses the other features like CAS and CTS for efficient management of the code.

**Unmanaged code:**  The code that does not depend on CLR for execution. It means it is developed by any other language independent of .Net framework. It uses its own runtime environment for compiling and execution.

**Q-14) What is Console application?**

A console application is an application that can be run in the command prompt in Windows. For any beginner on .Net, building a console application is ideally the first step, to begin with.

**Q-15) What is C#?**

C# is an object-oriented, type-safe, and managed language that is compiled by .Net framework to generate Microsoft Intermediate Language t and support all real-time application .C# is case sensitive Language.

**Q-16) Explain types of comment in C# with examples**

**1.Single line**(//)

**Example:** //This is a single line comment

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

**Example :** /\*This is a multiple line comment

Last line of comment\*

**3.XML Comments** (///).

**Example:** /// summary;

**Q-17) State the differences between C, C++,C# ?**

|  |  |  |
| --- | --- | --- |
| **C** | **C++** | **C#** |
| Procedural | Supports object –oriented | Supports object -oriented |
| Lowest level of abstraction | Low level of abstraction | High level of abstraction |
| Manual memory Management | Manual memory Management | Garbage collection |
| Very lightweight, compiled | Lightweight, compiled | Interpreted into bytecode, compiled by CLR, large binaries & overhead |
| Can code for any platform | Can code for any platform | Target toward Windows OS |

**Q-18) What is Namespace?**

Namespace contains , the group of classes and methods. It is designed for providing a way to keep one set of names separate from another. The class names declared in one namespace does not conflict with the same class names declared in another.

**Example:**

Using System;

Using System.IO;

Using System.Data;

**Q-19) Explain Data Types and Variables?**

**Data Types:** A type (or data type) defines the structure of data that will be stored in an object (variable) of that type. It describes about the type of data given by the user.

Variables: A variable is a named instance of that data type .Variables are used to store the data given by the user.

**Example:**

datatype variablename;

int studname;

**Q-20) Mention what is user-defined data type?**

A user-defined data type is a named data type created by the user. It can be a structured type which has a sequence of named attributes that each has a type, or it can be a distinct type sharing a common representation with some built-in data type. Based on this it can be categorized as,

* Distinct type
* Reference type
* Structured type

**Q-21) Distinguish between Continuous and Break Statement?**

|  |  |
| --- | --- |
| **Continuous Statement** | **Break Statement** |
| The Continuous Statement breaks one iteration (in the loop), if a specified condition occurs, and continues with the next iteration in the loop. | The Break Statement is used to jump out of a Loop. |
| **Example:**  For(int i=0;i<5;i++)  {  If(i==2)  {  continue;  }  Console.WriteLine(i);  } | **Example:**  For(int i=0;i<5;i++)  {  If(i==2)  {  break;  }  Console.WriteLine(i);  } |

**Q-22) Define Array?**

An array is an indexed collection of objects, all of the same type. C# arrays are somewhat different from array in C++ and others languages-because they are objects. This provides them with built-in support like useful methods and properties. There are three types of arrays:

* One-dimensional arrays
* Multidimensional arrays
* Jagged arrays

**Q-23) How to declare and initialize an array?**

C# provides native syntax for the **Declaration of array** object:

1. Int[] myIntArray;
2. Int[] myIntArray=new int[6];
3. myIntArray[2];//return the 3rd element

**Initialializing Array Elements:**

It initialize the contents of an array at the time it is instantiated by providing a list of values delimited by curly braces {}.

C# provides two different syntaxes to accomplish the same task:

1. int[] myIntArray=new int[6]{1,4,6,8,3,2}
2. int[] myIntArray={1,4,6,8,2}

**Q-24) Define Functions in C#?**

Function contains the collection of statement to perform any task and reusable of code. It defined ones and called in various part of applications. The two types of functions are pre-defined and user-defined. Advantage:

* Reduce the program length.
* Reduce the size of memory and .exe file.

**Q-25) what is data access modifier in .NET?**

Data access modifier in .NET provide a class, a function or a variable with accessibility.

The access modifier in .NET are five types:

* Public
* Private
* Protected
* Internal
* Protected Internal

**Q-26) What is Difference between call by value and call by reference?**

|  |  |
| --- | --- |
| **Call by value** | **call by reference** |
| It passes data to function | It passes address to the function |
| It is default parameter | Defined by using ref keyword |
| If you change the parameter value inside the function, it will not reflect outside of the function | If you change the parameter value inside the function, it will reflect outside of the function |

**Q-27) What is OOPS and its Advantages?**

OOP stands for Object-Oriented Programming. Procedural programming is about writing procedures or methods that perform operations on the data, while object-oriented programming is about creating objects that contain both data and methods.

**Advantages :**

* OOP is faster and easier to execute
* OOP provides a clear structure for the programs
* OOP helps to keep the C# code DRY "Don't Repeat Yourself", and makes the code easier to maintain, modify and debug
* OOP makes it possible to create full reusable applications with less code and shorter development time

**Q-28) Define Class?**

Class is a template, which contains a collection of member variable/attributes and member function.

**Syntax:**

class classname

{

//member variable

//member function

}

**Example:**

class student

{

int studname;//member variable

void details()//member function

{}

}

**Q-29) What is an object?**

An object is an instance of a class through which we access the methods of that class. "New" keyword is used to create an object. A class that creates an object in memory will contain the information about the methods, variables, and behavior of that class. Single class can create any number of object.

**Example:**

class hotel

{

string food = veg;

static void Main(string[] args)

{

food myobj = new food();

Console.WriteLine(myobj.food);

}

# Q-30) Define Nested Classes ?

A [class](https://www.geeksforgeeks.org/c-class-and-object/)is a user-defined blueprint or prototype from which objects are created. Basically, a class combines the fields and methods (member function which defines actions) into a single unit. In C#, a user is allowed to define a class within another class. Such types of classes are known as nested class.

**Syntax:**

class Outer\_class

{

// Code

class Inner\_class

{

// Code

}

}

**Q-31) Difference between class and interface?**

* A Class has both definition and an implementation whereas Interface only has a definition.
* A Class can be instantiated but an Interface cannot be instantiated You can create an instance of an Object that implements the Interface.
* A Class is a full body entity with members, methods along with there definition and implementation. An Interface is just a set of definition that you must implement in your Class inheriting that Interface.

**Q-32) What is the difference between ref and out parameters?**

|  |  |
| --- | --- |
| **Ref Parameter** | **Out Parameter** |
| The parameter or argument must be initialized first before it is passed to ref. | It is not compulsory to initialize a parameter or argument before it is passed to an out. |
| It is not required to assign or initialize the value of a parameter (which is passed by ref) before returning to the calling method. | A called method is required to assign or initialize a value of a parameter (which is passed to an out) before returning to the calling method. |
| Passing a parameter value by Ref is useful when the called method is also needed to modify the pass parameter. | Declaring a parameter to an out method is useful when multiple values need to be returned from a function or method. |
| When we use REF, data can be passed bi-directionally. | When we use OUT data is passed only in a unidirectional way (from the called method to the caller method). |
| It is not compulsory to initialize a parameter value before using it in a calling method. | A parameter value must be initialized within the calling method before its use. |

**Q-33) What are Pointers?**

Pointers are defined as a variable that contains the memory address of another variable. Pointers in C# are used whenever there is a statement that is unsafe and is marked by unsafe keyword. Those types of statements are not in control of garbage collectors and use pointer variables.

**Syntax:**

type  \*var name;  
 int\* a;

Here, \* is called a de-reference operator

a is the variable that contains the address of type int.

**Example:**

int \*p = & x;    // where &x is the memory address of x  
 Console.WriteLine((int)p) // displaying memory address  
 Console.WriteLine(\*p) // displaying value at memory address

**Q-34) Difference between Struct and Class?**

|  |  |
| --- | --- |
| **Struct** | **Class** |
| Struct are value types, allocated either on the stack or inline in containing types. | Classes are reference types, allocated on the heap and garbage-collected. |
| Allocations and de-allocations of value types are in general cheaper than allocations and de-allocations of reference types. | Assignments of large reference types are cheaper than assignments of large value types. |
| In Struct, each variable contains its own copy of the data (except in the case of the ref and out parameter variables), and an operation on one variable does not affect another variable. | In classes, two variables can contain the reference of the same object and any operation on one variable can affect another variable. |

**Q-35) What is enum?**

An enumeration is a set of named integer constants. An enumerated type is declared using the enum keyword.

C# enumerations are value data type. In other words, enumeration contains its own values and cannot inherit or cannot pass inheritance.

**Syntax:**

enum <enum\_name>

{

enumeration list

};

Where, The enum specifies the enumeration type name.

The enumeration list is a comma-separated list of identifiers.

**Q-36) Define Constructors?**

A constructor is a member function in a class that has the same name as its class. The constructor is automatically invoked whenever an object class is created. It constructs the values of data members while initializing the class.

Constructor doesn’t have returntype.

**Syntax:**

accesspecifier classname()

{}

**Types of Constructor :**

* Default Constructor
* Parameterized Constructor
* Copy Constructor
* Static Constructor

**Q-37) Explain Inheritance Concept?**

Inheritance is defined as to access the features from one class into another by using inheritance(i.e)reuse the entire class features in another class.

**The "inheritance concept" into two categories:**

* Derived Class (child) - the class that inherits from another class
* Base Class (parent) - the class being inherited from.

**Types of Inheritance:**

* Single Inheritance
* Multiple Inheritance
* Multilevel Inheritance
* Heriachial Inheritance
* Hybrid Inheritance

**Q-38) How do you inherit a class into other class in C#?**

Colon is used as inheritance operator in C#. Just place a colon and then the class name.

public class DerivedClass : BaseClass

**Q-39) Define Polymorphism?**

Polymorphism means "many forms", and it occurs when we have many classes that are related to each other by inheritance. **Polymorphism** uses those methods to perform different tasks. This allows us to perform a single action in different ways.

**Two types of Polymorphism:**

1. Overloading
2. Overriding

**Q-40) Explain overloading and overriding?**

**Method overloading** is creating multiple methods with the same name with unique signatures in the same class. When we compile, the compiler uses overload resolution to determine the specific method to be invoke.

Overloading can identified by **3** different features:

1. Number of Parameters
2. Types of Parameters
3. Sequence of Parameters

**Method overriding** is a features that allows you to invoke functions(that have the same signatures) that belong to different classes in the same hierarchy of inheritance using the base class reference.

**Q-41) Define Data Encapsulation?**

The wrapping up of datas combined and defined in the style logical unit called class that features is called Data Encapsulation.

**Example:**

class student

{

//student attributes are graped

}

**Q-42) Define Data Abstraction?**

The class contains multiple functionality but it won’t show all the features to other class. It provides only relavant information. This is called Data Abstraction.

**Example:**

Consider a real-life scenario of withdrawing money from ATM. The user only knows that in ATM machine first enter ATM card, then enter the pin code of ATM card, and then enter the amount which he\she wants to withdraw and at last, he\she gets their money. The user doesn’t know about the inner mechanism of the ATM or the implementation of withdrawing money etc. The user just simply know how to operate the ATM machine, this is called Abstraction.

**Q-43) Define Interfaces?**

An Interfaces  is a completely "abstract class", which can only contain abstract methods and properties (with empty bodies).

**Example:**

interface book

{

void booknumber();//interface method(does not have body)

}

**Q-44) Differentiate Interface and Abstract Class?**

|  |  |
| --- | --- |
| **Interface** | **Abstract** **Class** |
| Interface support Multiple Inheritance | Abstract Class does not support Multiple Inheritance |
| Interface doesn’t contains Data Member | Abstract Class contains Data Member |
| Interface doesn’t contains Constructor | Abstract Class contains Constructor |
| An Interface cannot have access modifiers by default, everything is assumed as public | An Abstract Class can contains access modifiers for the subs. functions, properties |
| Member of Interface cannot be Static | Only complete member of Abstract Class can be Static |

**Q-45) Define Collections?**

Collection classes are specialized classes for data storage and retrieval. These classes provide support for stacks, queues, lists, and hash tables. Most collection classes implement the same interfaces.

Collection classes serve various purposes, such as allocating memory dynamically to elements and accessing a list of items on the basis of an index etc. These classes create collections of objects of the Object class, which is the base class for all data types in C#.

The following are the various commonly used classes of the **System.Collection** namespace.

* ArrayList
* Hashtable
* SortedList
* Stack
* Queue
* BitArray

**Q-46) What is the difference between Function and Stored procedure?**

**Stored Procedure:**

* A Stored Procedure is always used to perform a specific task.
* It can return zero, one or more value.
* It can have both input and output parameters.
* Exception handling can be done using a try-catch block.
* A function can be called from a Procedure.

**Functions:**

* Functions must return a single value.
* It can only have the input parameter.
* Exception handling cannot be done using a try-catch block.
* A Stored procedure cannot be called from a function.

**Q-47) What are Delegates?**

Delegates are same are function pointers in C++, but the only difference is that they are type safe, unlike function pointers. Delegates are required because they can be used to write much more generic type-safe functions.

**Types of Delegates:**

1. Singlecast Delegates
2. Multicast Delegates

**Q-48) How to use nullable types in .Net?**

Value types can take either their normal values or a null value. Such types are called nullable types.

**Example:**

Int? someID = null;

If(someID.HasVAlue)

{ }

**Q-49) What is serialization?**

When we want to transport an object through a network, then we have to convert the object into a stream of bytes. The process of converting an object into a stream of bytes is called Serialization. For an object to be serializable, it should implement ISerialize Interface. De-serialization is the reverse process of creating an object from a stream of bytes.

**Q-50) What is ASP.Net and its Advantages?**

ASP stands for Active Server Pages. ASP .Net is a part of .Net technology and it comprises of CLR too. It is an open-source server-side technology that enables the programmers to build powerful web services, websites and web applications.

Advantages:

* Highly Scalable
* Compiled Code
* User Authentication
* Language Support
* Third party control
* Configuration and Deployment are easy.
* Object and Page caching
* Strict coding requirements

**Q -51) Explain State Management in ASP .Net?**

State Management means maintaining the state of the object. The object here refers to a web page/control.

**The two types of State Management:**

1. **Client-Side –** Storing the information in the Page or Client’s System. They are reusable, simple objects.
2. **Server Side –** Storing the information on the Server. It is easier to maintain the information on the Server rather than depending on the client for preserving the state.

**Q-52) Differentiate globalization and localization?**

The globalization is a technique to identify the specific part of a Web application that is different for different languages and make separate that portion from the core of the Web application. The localization is a procedure of configuring a Web application to be supported for a specific language or locale.

**Q-53) What is ADO.NET?**

ADO.NET provides a bridge between the front end controls and the back end database. The ADO.NET objects encapsulate all the data access operations and the controls interact with these objects to display data, thus hiding the details of movement of data.

**The key features of ADO.NET:**

* Disconnected Data Architecture.
* Data cached in DataSet.
* Scalability
* Data transfer in XML Format.
* Strongly typed language.

**Q-54) What is MVC?**

The Model-View-Controller (MVC) is an architectural pattern that separates an application into three main logical components: the model, the view, and the controller. Each of these components are built to handle specific development aspects of an application. MVC is one of the most frequently used industry-standard web development framework to create scalable and extensible projects.

MVC Component : Model, View, Controllers

### Q-55) What are the built-in objects in ASP.NET?

The major built-in objects are given below:

* Application
* Session
* Context
* Request
* Response
* Server
* Trace

### Q-56) What is IIS and its usage?

IIS stands for Internet Information Services. It is created by Microsoft to provide Internet-based services to ASP.NET Web applications.

**Following are the main usage of IIS:**

* IIS is used to make your computer to work as a Web server and provides the functionality to develop and deploy Web applications on the server.
* IIS handles the request and response cycle on the Web server.
* IIS also offers the services of SMTP and FrontPage server extensions.
* The SMTP is used to send emails and use FrontPage server extensions to get the dynamic features of IIS, such as form handler.

**Q-57) What is HTTPhandler?**

HTTPHandlers are used by ASP.NET web application server to handle specific requests based on extensions. HTTPHandlers run as processes in response to a request made to the ASP.NET website. It is a class that implements the System.Web.IHttpHandler interface. Any class that implements the IHttpHandler interface can perform as a target for the incoming HTTP requests.

It is an extension based processor and responsible for fulfilling requests from a browser depending on file extensions. When HttpHandler receives any request from browser, it checks the extension to see if it can handle that request and performs some predefined steps to respond that request.

ASP.NET framework offers a few default HTTP handlers. The most common handler is an ASP.NET page handler that processes .aspx files.

**Q-58) What is shadowing?**

Shadowing is either through scope or through inheritance. Shadowing through inheritance is hiding a method of a base class and providing a new implementation for the same. This is the default when a derived class writes an implementation of a method of base class which is not declared as overridden in the base class. This also serves the purpose of protecting an implementation of a new method against subsequent addition of a method with the same name in the base class. ’shadows’ keyword is recommended although not necessary since it is the default.

**Q-59) What is ViewState?**

The ViewState is a feature used by ASP.NET Web page to store the value of a page and its controls just before posting the page. Once the page is posted, the first task by the page processing is to restore the ViewState to get the values of the controls.

### Q-60) What are the HTML server controls in ASP.NET?

* HTML server controls are just like HTML elements that we use on the HTML pages.
* HTML server controls are used to expose properties and events for use.
* To make these controls programmatically accessible, we specify that the HTML controls act as a server control by adding the runat="server" attribute.

### Q-61) What is a cookie?

A Cookie is a small piece of information which is stored at the client side and the default timeout for a cookie is 30 minutes.

**There are two types of cookie:**

* Session/Temporary Cookie: valid for a single session
* Persistent Cookie: valid for multiple session

**Q -62) What is an Assembly? What are the different types of Assemblies?**

An Assembly is a collection of logical units. Logical units refer to the types and resources which are required to build an application and deploy them using the .Net framework. The CLR uses this information for type implementations. Basically, Assembly is a collection of Exe and DLLs. It is portable and executable.

**The two types of Assemblies:**

1. Private Assembly, as the name itself suggests, it is accessible only to the application. It is installed in the installation directory of the application.
2. Shared Assembly can be shared by multiple applications. It is installed in the GAC.

**Q -63) Explain the different parts of an Assembly.**

The different parts of an Assembly includes:

* **Manifest –** It contains the information about the version of an assembly. It is also called as assembly metadata.
* **Type Metadata –** Binary information of the program.
* **MSIL –** Microsoft Intermediate Language code.
* **Resources –** List of related files.

**Q -64) What is an EXE and a DLL?**

Exe and DLLs are Assembly executable modules.

**Exe** is an executable file. This runs the application for which it is designed. An Exe is generated when we build an application. Hence, the assemblies are loaded directly when we run an Exe. However, an Exe cannot be shared with the other applications.

**DLL** stands for Dynamic Link Library. It is a library that consists of code that needs to be hidden. The code is encapsulated inside this library. An application can consist of many DLLs. These can be shared with the other applications as well.

Other applications which share this DLL need not worry about the code intricacies as long as it is able to call the function on this DLL.

**Q -65) What is Caching?**

Caching means storing data temporarily in the memory so that the application can access the data from the cache instead of looking for its original location. This increases the performance of the application and its speed. System.Runtime.Caching namespace is used for Caching information in .Net.

**Different types of Caching:**

* Page Caching
* Data Caching
* Fragment Caching

**Q -66) What is a Dataset?**

Dataset is a disconnected copy of data that gets populated in the client PC. It contains Datatables and Datarelations. Generally, DataAdapter is required for populating a Dataset with the data.

Since it is disconnected, the user no longer need to be connected to the database every time and data manipulations are performed without interacting with the data source. It is very fast and reliable as it resides in the local system.

**Two Objects in Dataset:**

* DataTableCollection
* DataRelationCollection

**Two Types of Dataset:**

* Typed Dataset
* Untyped Dataset

**Q 67) What is DataAdapter?**

DataAdapter helps in linking the database and connection object. DataAdapter provides the communication between Dataset and data sources.

Data moves ‘in and out’ of the Dataset through DataAdapter.

**Properties of DataAdapter that allow the user to control the database are:**

* Select Command
* Update Command
* Insert Command
* Delete Command

**Q -68) What are DataReaders?**

DataReader object is ‘stream-based’, ‘read-only’ and ‘forward-only’, which provides a connection based data access from a database. This contains a ‘Read ()’ method that retrieves the data stored in a data source.

A Connection Object has only one DataReader at a time. ‘Read ()’ method retrieves only one row at a time. That is data need not be completely read into the application before it is processed.

**Q-69) Explain Deadlock?**

A deadlock is a situation that arises when a process isn’t able to complete it’s execution because two or more than two processes are waiting for each other to finish. This usually occurs in multi-threading. In this, a shared resource is being held up by a process and another process is waiting for the first process to get over or release it, and the thread holding the locked item is waiting for another process to complete.

**Q-70) Illustrate Race Condition?**

A Race Condition occurs in a situation when two threads access the same resource and try to change it at the same time. The thread which accesses the resource first cannot be predicted. Let me take a small example where two threads X1 and X2 are trying to access the same shared resource called T. And if both threads try to write the value to T, then the last value written to T will be saved.

**BASIC PROGRAMS:**

**Q-71) C# program to return Vowel of Not. Solution ?**

public bool CheckVowel(object sender,KeyEventArgs e)

{

if(e.KeyCode ==Keys.A||e.KeyCode ==Keys.E||e.KeyCode ==Keys.I||e.KeyCode ==Keys.O||e.KeyCode ==Keys.U)

{

return true;

}

else

return false;

}

**Q-72) C# Program to Even or Odd Solution?**

public string EvenOrOdd( int Num)

{

if(num%2 ==0)

retrun "evenNumber";

else

return "OddNumber"

}

**Q-73)C# program to find Prime or Not ?**

public string PrimeOrNot(int num)

{

int i = 0;

for (i = 3; i < num; i++)

{

if (num % i == 0)

{

Console.WriteLine("{0} is not a prime number", num);

break;

}

}

if (i == num)

{

Console.Writeline("{0} is a prime Number", num);

}

}

**Q-74) C# program to find**  **Fibonacci Series?**

public void FibonacciSeries(int number)

{

int f\_0 = 0;

int f\_1 = 1;

int sum;

Console.Write("{0} {1} ", f\_0, f\_1);

for (int i = 2; i < number; i++)

{

sum = f\_0 + f\_1;

f\_0 = f\_1;

f\_1 = sum;

int value = f\_0 + f\_1;

Console.Write("{0} ", value);

}

}

**Q-75) C# program to find Factorial number?**

public int Factorial(int number)

{

int value = 1;

for (int i = 1; i <= number; i++)

{

value = value \* i;

}

return value;

}

# Q-76) C# program to Find perfect numbers within a given range of number?

using System;

public class Exercise28

{

public static void Main()

{

int n,i,sum;

int mn,mx;

Console.Write("Input the starting range or number : ");

mn = Convert.ToInt32(Console.ReadLine());

Console.Write("Input the ending range of number : ");

mx = Convert.ToInt32(Console.ReadLine());

Console.Write("The Perfect numbers within the given range : ");

for(n=mn;n<=mx;n++)

{

i=1;

sum = 0;

while(i<n)

{

if(n%i==0)

sum=sum+i;

i++;

}

if(sum==n)

Console.Write("{0} ",n);

}

}

}

### Q-77) C# program to find sum of digits of a number using Recursion?

using System;

class program

{

public static void Main()

{

int num, result;

pro pg = new pro();

Console.WriteLine("Enter the Number : ");

num=int.Parse(Console.ReadLine());

result =pg.sum(num);

Console.WriteLine("Sum of Digits in {0} is {1}", num, result);

Console.ReadLine();

}

}

class pro

{

public int sum(int num)

{

if (num != 0)

{

return (num % 10 + sum(num / 10));

}

else

{

return 0;

}

}

}

**Q-78) C# program to find Amstrong number or not?**

Console.WriteLine("Enter the Number : ");

num=int.Parse(Console.ReadLine());

int n=num;

int sum=0;

while(no>0)

{

int rem=no%10;

sum=sum+(rem\*rem\*rem);

no=no/10;

}

If(sum==n)

{

Console.WriteLine(“Amstrong Number”);

}

else

{

Console.WriteLine(“Not an Amstrong Number”);

}

}

}

**Q-79) C# program to display the pattern like a pyramid containing an odd number of asterisks?**

using System;

public class Exercise20

{

public static void Main()

{

int i,j,n;

Console.Write("Input number of rows for this pattern :");

n= Convert.ToInt32(Console.ReadLine());

for(i=0;i<n;i++)

{

for(j=1;j<=n-i;j++)

Console.Write(" ");

for(j=1;j<=2\*i-1;j++)

Console.Write("\*");

Console.Write("\n");

}

}

}

**Output:**

Input number of rows for this pattern :4

\*

\*\*\*

\*\*\*\*\*

**Q-80)C# program to print pattern of 0's and 1's?**

using System; //include the namespace

namespace program

{

class ab

{

static void Main(String[] args)

{

int i,j,c=0;

for(i=0;i<4;i++) //loop for row

{

for(j=0;j<=i;j++) //loop for column

{

c++; //increment in count variable

if(c%2==0)

Console.Write(0);

else

Console.Write(1);

}

Console.WriteLine(); //for new line

}

}

}

}

**Output:**

1

01

010

1010