**Capítulo 1:**

1. ¿Cuál es la historia del desarrollo de las aplicaciones para Windows?  
   Años atrás la única manera de programar para el SO Windows era compilando en los lenguajes C o C++ y bajo una API denominada Win32, lo que hacia tediosa la programación porque todo era complejo y enorme, Microsoft diseño un conjuntos de clases que envolvían todo lo que contenía Win32 facilitando la programación este proyecto fue llamado MFC.
2. ¿Qué problemas ayuda a resolver .NET?   
   Es una solución a toda la problemática en torno al desarrollo de aplicaciones, brinda grandes beneficios no solamente al desarrollador, sino también al proceso de desarrollo. En primer lugar, .NET permite trabajar con código ya existente, podemos hacer uso de los componentes COM, e incluso, si lo necesitáramos usar el API de Windows. Cuando el programa .NET está listo es mucho más fácil de instalar en la computadora de los clientes, que las aplicaciones tradicionales ya que se tiene una integración fuerte entre los lenguajes. Un programador de C# puede entender fácilmente el código de un programador de Visual Basic .NET y ambos pueden programar en el lenguaje con el que se sienten más cómodos. Esto se debe a que todos los lenguajes que hacen uso de .NET comparten las librerías de .NET, por lo que no importa en qué lenguaje programemos, las reconocemos en cualquiera
3. ¿Qué es un assembly?  
   Un assembly no es más que lo que genera el . NET al momento de la compilación.
4. ¿Cuál es la definición de CIL y CLR?  
   CIL es el lenguaje intermedio en el que compila .NET y CLR es el encargado de leer el código generado por el compilador y empezar su ejecución.
5. Describa qué hace el CLR con el assembly.
6. ¿Cuál es el significado de Jitter?  
   Compilar Jit o Jitter este forma parte del runtime y es eficiente si el programa necesita volver a ejecutar un código ya que se ha compilador , El Jit en vez de volver a compilar el código, ejecuta lo ya compilado.
7. Mencione algunos compiladores de C# que podemos utilizar.  
   Visual Studio, ya sea la versión community, professional o Express.
8. ¿Qué es el CTS?  
   El Framework de .NET define por medio del CTS cómo van a funcionar los tipos en su entorno. Cualquier lenguaje que trabaje con .NET debe de usar los tipos tal y como se señalan en el CTS.
9. ¿Porqué .NET puede ser multiplataforma?  
   Este es un lenguaje intermedio que lee el runtime cuando se ejecuta la aplicación. El CLR compila el CIL para el microprocesador según va siendo necesario, el uso de un runtime le da a .NET la flexibilidad de ser multiplataforma. Todos los lenguajes
10. ¿Qué otro lenguaje que use .NET existe?  
    Visual Basic.Net y C#
11. ¿Cuál es la última versión de .NET?  
    Versión 4.7.03056.
12. ¿Qué otro lenguaje que use .NET existe?  
    Visual Basic.Net y C#

**Capítulo 2:**

1. ¿Qué es un algoritmo?  
   Son los pasos necesarios para la resolución de un problema.
2. ¿Qué características tienen los algoritmos?  
   Las características importantes de un algoritmo son: Debe ser finito , debe tener un comienzo y un final, debe ser secuencial y organizado entre otras.
3. ¿Qué es una sentencia?  
   Son instrucciones que el programa ejecuta.
4. ¿Cómo se finalizan las sentencias?  
   Las sentencias finalizan en punto y coma(;).
5. ¿Cuál es la diferencia entre Write() y WriteLine()?  
   El método Write() escribe la sentencia y la próxima sentencia aparece justamente al lado de lo anterior, mientras que el Metodo WriteLine() al final de la sentencia hace un salto de línea.
6. ¿Qué es una cadena?  
   es una colección de caracteres, es decir, letras, números y signos
7. ¿Qué es una variable?  
   Son las posiciones de memoria donde guardamos los datos de un programa.
8. ¿Cómo mostramos el valor de una variable?  
   Con los métodos Write() o WriteLine()
9. ¿Cómo le solicitamos un dato al usuario?  
   Con la sentencia Console.ReadLine() o sea el método ReadLine().
10. ¿Cómo se convierte una cadena a un valor numérico?  
    Con la sentencia Convert.To luego del “To” debemos poner al tipo de dato al que queremos convertir por ejemplo “ToSingle , ToDecimal, ToString ” entre otros.
11. ¿Qué son los operadores aritméticos?  
    = Asignación. Este operador ya es conocido por nosotros.   
    + Suma. Nos permite sumar los valores de las variables o los números  
     - Resta. Para restar los valores de las variables o los números.   
    \* Multiplicación. Multiplica los valores de las variables o los números.  
     / División. Divide los valores de las variables o los números.   
    % Módulo. Nos da el residuo de la división.
12. ¿De qué forma podemos agrupar operaciones aritméticas?  
    Utilizando los paréntesis “()”.
13. ¿Qué son las estructuras selectivas?  
    Son aquellas rutas que nos permiten elegir entre dos o más rutas de ejecución posibles.
14. ¿Cuándo se ejecuta la sentencia de if?  
    Cuando la condición dentro del paréntesis es verdadera.
15. ¿Qué colocamos entre paréntesis en if?  
    Una condición, si la misma se cumple se ejecuta el bloque de códigos o sentencias dentro de la condición.
16. ¿Qué es una expresión relacional?  
    Es una expresión utilizada para expresar la relación que existe entre dos valores.
17. ¿Cuáles son los operadores de una expresión relacional?  
    == Igualdad   
    != No igual   
    > Mayor que   
    < Menor que   
    >= Mayor que igual   
    <= Menor que igual
18. ¿A qué valores posibles puede evaluar una expresión relacional o lógica?  
    True o False , o sea verdadero o falso.
19. ¿Qué es una expresión lógica?  
    son  expresiones  pero  cuyo  resultado  es  un valor "lógico" (verdadero o falso).
20. ¿Cuáles son los operadores de las expresiones lógicas?  
    && y   
    || o   
    ! no
21. ¿Qué operador solamente necesita un operando?  
    El operador de la negación.
22. ¿Cómo funciona el switch?  
    es necesario colocar entre paréntesis la variable que utilizaremos para llevar a cabo las comparaciones. Luego tenemos que crear un bloque de código y colocar adentro de él los casos y el código a ejecutar para cada caso. Para indicar un caso, usamos case seguido del valor de comparación y dos puntos
23. ¿Qué es la variable de comparación y cómo se coloca?  
    La variable de comparación será la variable opción, ya que el valor de ésta será comparado para cada operación, se coloca dentro del paréntesis del Switch.
24. ¿Cómo definimos los casos y cómo usamos break?
25. ¿Qué es un ciclo?  
    Es una sentencia dentro del programa que se ejecuta varias veces hasta que una condición asignada a ese ciclo deje de cumplirse.
26. ¿Cuáles son las partes del ciclo for?  
    for( inicialización; condición; incremento)   
     código
27. ¿Cómo colocamos el valor inicial de conteo en un ciclo for?  
    El valor inicial se coloca en la inicializacion del ciclo
28. ¿Cómo colocamos el valor final de conteo en un ciclo for?  
    Se coloca en el campo llamado condición
29. ¿Cómo se lleva a cabo el incremento en un ciclo for?  
    El incremento sucede al final de cada vuelta del ciclo
30. ¿Cómo funciona el ciclo do while?  
    Permite que cierto código se repita mientras una condición se evalúe como verdadera. El valor de la evaluación dependerá del estatus del programa en un momento dado.
31. ¿Por qué el ciclo while se lleva a cabo al menos una vez?  
    Porque primero ejecuta el bloque de código y luego evalúa la condición al final del ciclo
32. ¿Se necesita punto y coma al finalizar el ciclo do while?
33. ¿Cómo funciona el ciclo while?  
    El ciclo While primero evalua la condicion y luego ejecuta el bloque de código.
34. ¿Cuántas veces se puede repetir el ciclo while?  
    0 o mas veces
35. ¿Qué tipo de condición podemos colocar en el ciclo while?  
    Una condicion o expresión booleana.
36. ¿Se coloca un bloque de código en el ciclo while?  
    Si.