**Test de Auto Evaluación Capitulo 7:**

**1 ¿Qué son las colecciones?**

Las colecciones son estructuras de datos que nos permiten guardar en su interior cualquier tipo de información. Existen diferentes tipos de colecciones y la forma como se guarda, se accede y se elimina la información en cada una de ellas es distinta. En los arreglos nosotros teníamos que indicar la cantidad de elementos que el arreglo debía tener. En las colecciones esto no es necesario, ya que es posible agregar elementos dinámicamente.

**2 ¿Cómo funciona ArrayList?**

El ArrayList, que guarda la información como si fuera una lista. Y sobre esta lista es posible realizar diferentes actividades con los elementos almacenados. Entendemos al ArrayList como un arreglo que puede cambiar su tamaño según lo necesitemos.

**3 ¿Cómo introducimos un elemento en el ArrayList?**

Con el método Insert()

Ejemplo: AL.Insert(2, 5);

**4 ¿Cómo funciona el ciclo foreach?**

La principal característica del iterador es permitirnos realizat un recorrido por todos los elementos que existen en una estructura de datos. El recorrido lo hace de forma sequencial, uno por uno. Esto resulta muy útil cuando sabemos que debemos recorrer todos los elementos de la estructura, como un ArrayList, y hacer algo con ellos. En C# encontramos un iterador en el foreach. Con él es posible que recorramos los elementos, luego ejecutamos alguna acción con ellos y finalizamos cuando la colección ya no tiene más elementos que entregarnos.

**5 ¿Cómo funciona el Stack?**

Ahora empezaremos a conocer otro tipo de colección. A esta colección se la conoce como Stack o pila, nos permite guardar elementos y cambia su tamaño de forma dinámica, sin embargo, trabaja en forma diferente al arreglo y al ArrayList. El Stack es una estructura de tipo LIFO. LIFO es el acrónimo en inglés para Last-in-first-out, es decir el primero que entra, el último que sale. Para entender su funcionamiento podemos imaginar una pila de platos.

El primer plato que colocamos queda hasta la base, el siguiente se colocará encima y así sucesivamente. Como el primer plato queda hasta abajo, no lo podemos sacar directamente pues la pila se derrumbaría. Al sacar los platos, debemos tomar el que se encuentre hasta arriba de la pila primero y así continuar.

**6 ¿Qué hace la operación Push?**

El efecto de colocar nuevos elementos en la parte superior del Stack se conoce como Push.

**7 ¿Qué hace la operación Pop?**

Cuando tomamos un elemento de la parte superior del Stack se conoce como Pop.

**8 ¿Para qué sirve la propiedad Count?**

Esta propiedad nos regresa un valor entero con la cantidad de elementos del ArrayList.

**9 ¿Cómo funciona el Queue?**

La siguiente colección que aprenderemos se conoce como Queue, algunas veces también denominada cola. Al igual que en el Stack, nosotros ya no necesitamos programarla, pues C# nos provee una clase con toda la funcionalidad necesaria para poder utilizarla. La clase que debemos de utilizar para poder tener una cola es Queue. La forma como trabaja es diferente al Stack, ya que el Queue es una estructura de tipo FIFO.

**10 ¿Cómo funciona la operación Enqueue?**

Nosotros podemos adicionar elementos al Queue en cualquier momento que lo necesitemos. El tamaño del Queue se modificará dinámicamente por lo que no debemos de preocuparnos por él. Siempre que se adiciona un elemento, este elemento se coloca al final o en la parte baja del Queue. Para poder hacer esto debemos utilizar el método Enqueue (), el cual pertenece a la clase Queue de C#. Este método es muy sencillo, ya que solamente requiere de un parámetro. En el parámetro colocamos el elemento que deseamos añadir, este método no regresa ningún valor.

**11 ¿Cómo funciona la operación Dequeue?**

Para llevar a cabo la extracción tenemos que usar un método de la clase Queue que se llama Dequeue (). El método no necesita de ningún parámetro y regresa el elemento correspondiente. Es importante tener una variable que reciba al elemento o una expresión que haga uso de él.

**12 ¿Cómo funciona el Hashtable?**

Es una estructura de datos un poco compleja de implementar, pero afortunadamente C# nos provee de una clase que nos da toda la funcionalidad necesaria y podemos utilizarla tan fácilmente como las colecciones anteriores. El Hashtable es una colección indexada. Es decir que vamos a tener un índice y un valor referenciado a ese índice. Sin embargo, la indexación no se lleva a cabo como en el arreglo o el ArrayList. El lugar adentro del Hashtable donde se coloca el elemento va a depender de un valor conocido como key o llave. El valor contenido en key es usado para calcular la posición del elemento en el Hashtable. El elemento que vamos a colocar se conoce como value.

**Test De Autoevaluación Capitulo 8:**

**1 ¿Qué es una cadena?**

En los capítulos anteriores hemos trabajado con cadenas. Las utilizamos, por ejemplo, cuando necesitamos mandarle mensajes al usuario. Sin embargo, las cadenas tienen mucha más funcionalidad que eso. Si necesitamos guardar cualquier información de tipo alfanumérica las podemos utilizar. También son utilizadas para guardar nombres, direcciones, etcétera.

**2 ¿Qué clase se usa para obtener la fecha y hora de la computadora?**

Para utilizar la fecha y la hora, lo haremos mediante una clase conocida como **DateTime**. Esta clase provee todos los elementos necesarios para poder trabajar con la información relacionada con el tiempo, como son la fecha y las horas del día.

**3 ¿Cómo se les da el formato a la fecha y la hora?**

Podemos hacer uso del método Format() de String. Este método nos permite reemplazar los contenidos de la cadena por una cadena con un formato en particular.

**4 ¿Cómo se le da el formato a un valor numérico?**

Al igual que con la información de la fecha y hora, es posible darles formato a los

valores numéricos que deseemos mostrar. Para éstos también tendremos una serie

de especificadores que podemos utilizar.

**# Dígito**

**. Punto decimal**

**, Separador de miles**

**% Porcentaje**

**E0 Notación científica**

**; Separador de secciones**

**5 ¿Qué es la concatenación?**

La concatenación es unir dos o mas cadenas con el operador (+).

**6 ¿De qué forma nos ayuda StringBuilder?**

Para mejorar el uso de la memoria se puede utilizar **StringBuilder**, una clase que nos da funcionalidades para facilitar la concatenación.

**7 ¿De qué manera se lleva a cabo la comparación de cadenas?**

Nos permite construir cadenas de forma eficiente y a su vez podemos utilizarla en lugar de las concatenaciones si fuera necesario.

**8 ¿Cómo podemos obtener una parte de la cadena?**

Otra opción es la propiedad Chars, mediante la que podemos obtener o modificar un carácter en particular. La modificación se hace indicando el índice donde se encuentra ese carácter.

**9 ¿Cómo podemos saber si una palabra se encuentra adentro de una cadena?**

La comparación de cadenas es una herramienta muy importante para solucionar diversos problemas y lograr resultados que de otra forma serían muy complejos. La clase String nos da el método Compare(). Este método es estático, por lo que podemos hacer uso de éste sin necesidad de declarar un objeto de tipo String. El método necesitará dos parámetros, que son las cadenas que comparar. Los llamaremos en nuestro ejemplo Cadena1 y Cadena2. El método regresará luego del análisis, un valor entero y el valor de este entero será el que se encargue de indicarnos la relación que existe entre las cadenas que acabamos de comparar.

**10 ¿Cómo podemos justificar una cadena a la derecha?**

El método PadLeft() se encarga de esto. Requiere de un parámetro que es la cantidad de caracteres de la cadena final. Éste es un valor entero y representa los caracteres originales más los espacios en blanco. Regresa la cadena final justificada.

**11 ¿De qué forma podemos eliminar los espacios extras al final de la cadena?**

El método que podemos utilizar se conoce como Remove(). Este método está sobrecargado, pero veremos la versión que es más flexible. La eliminación de los caracteres puede hacerse en cualquier parte de la cadena, sólo debemos tener cuidado de no generar ningún error.

**12 ¿De qué forma podemos colocar una cadena totalmente en mayúscula?**

De forma similar, podemos pasar la cadena a mayúscula. La forma de hacer esto es con el método ToUpper(), que toma la cadena y pasa todas sus letras a mayúscula. El método no necesita ningún parámetro, ya que al igual que ToLower(), toma la información directamente de la cadena que lo invoca y regresa una cadena, que es la resultante con todas las letras en mayúscula.

**Test De Autoevaluación Capitulo 9:**

**1 ¿Qué es una estructura?**

Las estructuras son tipos definidos por el programador y son un conjunto de datos agrupados.

**2 ¿Cómo accedemos a un campo de la estructura?**

Luego que la estructura esta creada:

**Struct Contactos**

**{**

**String Nombre;**

**String Apellido;**

**};**

**Contactos c;**

**C.Nombre = “Nombre”;**

Para acceder a un campo de la estructura creamos una variable tipo la estructura y con un punto accedemos a ese campo X.

**3 ¿Cómo creamos un constructor para la estructura?**

Sigamos con el ejemplo de la pregunta anterior, crearemos el constructor de la estructura Contactos:

**Public Contactos(string nNombre)**

**{**

**Nombre = nNombre;**

**}**

**4 ¿Para qué sirve un constructor?**

El constructor no es otra cosa que un método que nos permitirá llevar a cabo la inicialización de los campos.

**5 ¿Cómo se usa la sobrecarga del constructor?**

Es simplemente crear otro constructor pero con parámetros diferentes, por ejemplo crearemos un constructor vacio:

Public Contactos(string nNombre, string nApellido)

{

Nombre = nNombre;

Apellido = nApellido;

}

**6 ¿Qué son las estructuras enlazadas?**

Es una estructura que tiene como campo una variable tipo otra estructura.

**7 ¿Cómo se define una variable de la estructura?**

Podemos definir una variable de la estructura de la forma tradicional, pero también podemos hacerlo por medio new. Con ésta última es posible colocar un constructor adentro de la estructura y facilitarnos la introducción de información. Si se define de forma tradicional deberemos inicializar cada campo de la estructura manualmente.

**8 ¿Cómo podemos usar el método ToString() en la estructura?**

La mejor forma de mostrar todos los datos contenidos en los campos sería mediante la conversión de la estructura a una cadena. Sabemos que existe un método llamado ToString() que hemos usado con las variables numéricas. Sin embargo, C# no se lo puede dar directamente a nuestra estructura porque no puede saber cuáles son los campos que contiene ni cómo los deseamos mostrar. Por esto, cae en nuestra responsabilidad programar el método ToString() de nuestra estructura.

**9 ¿Qué es una enumeración?**

Ya hemos aprendido las estructuras y ahora conoceremos otro nuevo tipo que puede ser definido por el programador. Este nuevo tipo se conoce como enumeraciones. Éstas son tipos que nosotros podemos definir, pero tendrán un número finito de valores posibles, es decir, que podemos definir el tipo y los valores a guardar.

**10 ¿Cómo se declara una enumeración?**

Para declarar una enumeración usamos el siguiente formato:

**enum nombre {valor1, valor2, …, valorn};**

**11 ¿Cómo creamos una variable de enumeración?**

Veamos cómo declarar una enumeración para nuestro ejemplo de la semana:

**enum semana {Lunes, Martes, Miercoles, Jueves, Viernes, Sabado, Domingo};**

**12 ¿De qué forma se le coloca un valor al elemento de la enumeración?**

La asignación del valor en esta variable se puede llevar a cabo en cualquier momento, dentro de nuestra aplicación, después de su declaración, como en cualquier otra variable, pero con uno de los valores contenidos en semana.

**miDia=semana.Viernes;**

**Test De Autoevaluación Capitulo 12:**

**1 ¿Qué es la depuración?**

La depuración de programas es el proceso de identificar y corregir errores de programación. En inglés se conoce como debugging, porque se asemeja a la eliminación de bichos (bugs), manera en que se conoce informalmente a los errores de programación.

**2 ¿Cuántos tipos de errores tenemos?**

Básicamente podemos tener dos clases de errores bien diferenciados: por un lado, los errores de compilación y por otro los errores en tiempo de ejecución.

**3 ¿Por qué algunos programas no pueden compilar?**

Los errores de compilación son aquellos que impiden que el programa logre compilarse y generalmente son más fáciles de resolver. Muchos de estos errores se deben a problemas de sintaxis, es decir, son responsabilidad del usuario.

**4 ¿Qué es un error de sintaxis?**

Estos se deben a la mala escritura del código, por ejemplo, cuando se escribe algo de forma incorrecta.

**5 ¿En dónde podemos ver los errores que tenemos?**

Los problemas de sintaxis son sencillos de corregir, se deben a que se ha escrito algo de forma incorrecta. Algunas veces los nombres de variables se han escrito erróneamente, o hemos olvidado colocar; al final de la sentencia. Una revisión rápida de la línea donde está el error nos permite encontrarlo, con la práctica también se reduce la incidencia de estos errores.

**6 ¿Qué son los errores en tiempo de ejecución?**

Los errores en tiempo de ejecución son aquellos que suceden cuando el programa se está ejecutando. Estos errores pueden deberse a problemas de lógica, es decir, que el algoritmo no fue diseñado correctamente.

**7 ¿Qué es un error de lógica?**

Los errores de lógica son los más difíciles de detectar. Cuando un programa no tiene errores de sintaxis ni errores de ejecución, pero, aun así, no funciona bien, esto es debido a la existencia de algún error lógico. De manera que, un error de lógica se produce cuando los resultados obtenidos no son los esperados.

**8 ¿Qué es una excepción?**

Una excepción es la indicación de que se produjo un error en el programa. Las excepciones, como su nombre lo indica, se producen cuando la ejecución de un método no termina correctamente, sino que termina de manera excepcional como consecuencia de una situación no esperada.

**9 ¿Qué es la administración de excepciones?**

No es posible evitar las excepciones completamente. Esto se debe a que no siempre tenemos control sobre la información o el manejo que va a tener el usuario en nuestro programa. Por ejemplo, en el programa anterior, la excepción se eliminó al dividir entre la variable que no tiene el valor de cero. Pero imaginemos que el usuario es el que debe colocar el valor del divisor.

En este caso no podemos forzar a una variable con un valor diferente de cero. En algunas ocasiones el usuario colocará un valor adecuado, pero también es posible que dé el valor de cero. En esos casos, el programa terminará por el error.

**10 ¿Cuáles son los bloques de código para administrar la excepción?**

Para llevar a cabo la administración de excepciones vamos a tener tres bloques de código conocidos como: try, catch y finally.

**11 ¿Qué es un punto de interrupción?**

Los puntos de interrupción detienen la ejecución del depurador para, por ejemplo, ver el estado de las variables de código o examinar la pila de llamadas. Constituyen una de las técnicas de depuración más importantes en los cuadros de herramientas de los desarrolladores.

**12 ¿Cómo podemos depurar paso a paso?**

Existen muchos métodos y sentencias que pueden generar excepciones. Pero no es bueno exagerar. Una gran cantidad de excepciones se evitan simplemente al usar correctamente los métodos y hacer buen uso de nuestras variables y sus valores. Solamente debemos implementar la administración de excepciones en las partes del programa que realmente lo necesiten.