Seminario de Lenguajes (.NET)

Práctica 3

1. Ejecute y analice cuidadosamente el siguiente programa:

```
using System;
class Programa{
  static void Main(){
    for(int i = 0; i < Console.WindowHeight; i++ ){
      Console.CursorLeft = Console.WindowWidth - 1 - i;
      Console.CursorTop = i*2;
      Console.Write(i);
    }
    Console.CursorLeft=0;
    Console.CursorVisible = false;
    Console.WriteLine("Presione una tecla para finalizar");
    Console.ReadKey(true);
  }
}
```

   ¿Cuál es el efecto de cambiar el valor de las propiedades `WindowHeight`, `WindowWidth`, `CursorLeft`, `CursorTop` y `CursorVisible` de la clase `Console`?

2. Ejecute y analice cuidadosamente el siguiente programa:

```
using System;
class Programa{
  static void Main(){
    Console.CursorVisible=false;
    ConsoleKeyInfo k = Console.ReadKey(true);
    while(k.Key != ConsoleKey.End){
      Console.Clear();
      Console.Write("{0,-10} {1,-10} {2,-10}",k.Modifiers, k.Key, k.KeyChar);
      k = Console.ReadKey(true);
    }
  }
}
```

   Compruebe tipeando teclas y modificadores (shift, ctrl, alt) para apreciar de qué manera se puede acceder a esta información en el código del programa.

3. Escriba un programa donde se deba ingresar un texto por consola. A medida que se escribe el texto, el programa en vez de mostrar el texto tal y como se tipea lo va mostrando en mayúsculas. El programa solo debe imprimir las letras que se tipean ignorando otros caracteres. Al presionar la tecla Enter el programa finaliza mostrando el texto original que ingresó el usuario.

4. Escriba un programa que permita ingresar texto por pantalla. El texto, a medida que se tipea, es mostrado invertido y recostado a derecha. Cuando se presiona la tecla Enter, en vez de saltar a la siguiente linea, todo el texto es desplazado una linea hacia abajo y la nueva linea se tipea en la primer linea de la pantalla. El programa finaliza cuando se presiona la tecla escape o cuando se ingresaron 10 lineas.

5. Escriba un programa que acepte por linea de comando el nombre de un archivo a leer y otro a escribir. El programa lee el archivo correspondiente y escribe en el segundo archivo todas las palabras leídas ordenadas alfabéticamente.

6. Implemente un programa que permita al usuario ingresar números por la consola. Debe ingresarse de un número por línea finalizado el proceso cuando el usuario ingresa una línea vacía. A medida que se van ingresando los valores el sistema debe mostrar por la consola la suma acumulada de los mismos. Utilice la instrucción try/catch para validar que la entrada ingresada sea un número válido, en caso contrario advertir con un mensaje al usurio y proseguir con el ingreso de datos.

7. ¿Qué salida por la consola produce el siguiente código?

```
int x=0;
try{
  Console.WriteLine(1.0/x);
  Console.WriteLine(1/x);
  Console.WriteLine("Ok");
}catch{
}
```

¿Qué se puede inferir respecto de la excepción división por cero en relación al tipo de los operandos?

8. Compruebe el funcionamiento del siguiente programa:

```
using System;
class Ejercicio3 {
  static void Main(string[] args) {
    try{
      metodo1();
    }catch{
      Console.WriteLine("método 1 propagó una excepción no tratada");
    }
    Console.ReadLine();
  }
  static void metodo1() {
    byte b=255;
    try{
      b++;
    }finally{
      Console.WriteLine("bloque finally");
    }
  }
}
```

9. Implemente un programa calculadora que calcule una expresión ingresada por el usuario correspondiente a una operación binaria de las cuatro elementales (ejemplo “44.5/456”, “456*45”, “549-12”, “54+6”). Utilice try/catch para validar los números y controlar cualquier tipo de excepción que pudiese ocurrir en la evaluación de la expresión.

10. Consulte la documentación sobre la clase `BitArray`. Compile y ejecute el siguiente programa:

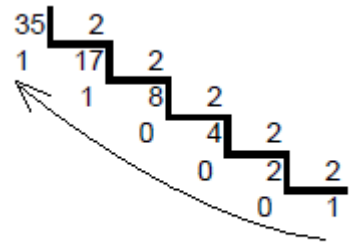
```
using System;
using System.Collections;
class Program
{
  public static void Main(string[] args){
    int[] vectorDeInt = new int[] {5,255};
    Console.WriteLine("BitArray construido a partir " +
                      "del vector de enteros {5,255}");
    BitArray ba1=new BitArray(vectorDeInt);
    for(int i=0;i<ba1.Count;i++){
       Console.WriteLine ("posicion {0,2} valor {1}", i, ba1[i]);
    }
    byte[] vectorDeByte = new byte[] {5,255};
    Console.WriteLine("BitArray construido a partir " +
                      "del vector de bytes {5,255}");
    BitArray ba2=new BitArray(vectorDeByte);
    for(int i=0;i<ba2.Count;i++){
       Console.WriteLine ("posicion {0,2} valor {1}", i, ba2[i]);
    }
    Console.ReadKey(true);
  }
}
```

¿Observó la diferencia en la creación de ambos objetos `Bitarray`? ¿Dónde se colocan los bits más significativos?

11. ¿Cómo escribiría la representación binaria de un `char` utilizando `BitArray`? Recuerde que puede hacer casting para transformar un `char` en un `byte`, por ejemplo `(byte)'A'` devuelve un byte con el valor 65.

12. Escriba un programa que solicite al usuario el ingreso de dos palabras de la misma longitud por consola, realice la operación lógica `AND` entre la representación binaria de ambos `strings` e imprima el resultado por pantalla (utilice la clase `BitArray`).

13. Utilice la clase `Stack` (pila) para implementar un programa que pase un número en base 10 a otra base realizando divisiones sucesivas. Por ejemplo para pasar 35 en base 10 a binario dividimos sucesivamente por dos hasta encontrar un cociente menor que el divisor, luego el resultado se obtiene leyendo de abajo hacia arriba el cociente de la última división seguida por todos los restos.



El resultado es 100011

14. Realice un análisis sintáctico para determinar si los paréntesis en una expresión aritmética están bien balanceados. Verifique que por cada paréntesis de apertura exista uno de cierre en la cadena de entrada. Utilice una pila para resolverlo. Los paréntesis de apertura de la entrada se almacenan en una pila hasta encontrar uno de cierre, realizándose entonces la extracción del último paréntesis de apertura almacenado. Si durante el proceso se lee un paréntesis de cierre y la pila está vacía, entonces la cadena es incorrecta. Al finalizar el análisis, la pila debe quedar vacía para que la cadena leída sea aceptada, de lo contrario la misma no es válida.

15. Utilice la clase `queue` (cola) para implementar un programa que realice el cifrado de un texto aplicando la técnica de clave repetitiva. La técnica de clave repetitiva consiste en desplazar un carácter una cantidad constante de acuerdo a la lista de valores que se encuentra en la clave. Por ejemplo: para la siguiente tabla de caracteres:

A	**B**	**C**	**D**	**E**	**F**	**G**	**H**	**I**	**J**	**K**	**L**	**M**	**N**	**Ñ**	**O**	**P**	**Q**	**R**	**S**	**T**	**U**	**V**	**W**	**X**	**Y**	**Z**	**sp**
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28

Si la clave es {5,3,9,7} y el mensaje a cifrar "`HOLA MUNDO`"
Se cifra de la siguiente manera:

H	**O**	**L**	**A**	**sp**	**M**	**U**	**N**	**D**	**O**	**← Mensaje original**
8	16	12	1	28	13	22	14	4	16	**← Código sin cifrar**
5	3	9	7	5	3	9	7	5	3	**← Clave repetida**
13	19	21	8	5	16	3	21	9	19	**← Código cifrado**
M	**R**	**T**	**H**	**E**	**O**	**C**	**T**	**I**	**R**	**← Mensaje original**

A cada carácter del mensaje original se le suma la cantidad indicada en la clave. En el caso que la suma fuese mayor que 28 se debe volver a contar desde el principio, Tenga en cuenta que para resolver este problema debe utilizar una cola que guarde la clave y que a medida que saque elementos de la cola los tiene que agregar nuevamente para poder utilizarla en forma repetitiva. Programe una rutina para cifrar y otra para descifrar.

16. Verifique si la sección opcional [:*formatString*] de formatos compuestos redondea o trunca en el caso de la impresión de números reales cuando se restringe la cantidad de decimales.

17. Implemente un método estático para imprimir por consola todos los elementos de una matriz (arreglo de dos dimensiones) pasada como parámetro. Debe imprimir todos los elementos de una fila en la misma línea en la consola.

```
static void imprimirMatriz(double[,] matriz)
```

Ayuda: Si `A` es un arreglo, `A.GetLength(i)` devuelve la longitud del arreglo en la dimensión `i`.

18. Idem al anterior pero ahora con un parámetro más que representa la plantilla de formato que debe aplicarse a los números al imprimirse. Observe que no hay inconveniente para implementar dos métodos con el mismo nombre siempre que NO lleven la misma cantidad de parámetros con los mismos tipos y en el mismo orden (sobrecarga de métodos).

```
static void imprimirMatriz(double[,] matriz, string formatString)
```

19. Implemente los métodos **`getDiagonalPrincipal`** y **`getDiagonalSecundaria`** que devuelva un vector con la diagonal correspondiente de la matriz pasada como parámetro. Si la matriz no es cuadrada devuelve null.

```
static double[] getDiagonalPrincipal(double[,] matriz)
static double[] getDiagonalSecundaria(double[,] matriz)
```

20. Implemente un método estático que devuelva un arreglo de arreglos con los mismos elementos que la matriz pasada como parámetro:

```
static double [][] getArregloDeArreglo(double [,] matriz)
```

21. Implemente los siguientes métodos estáticos que devuelven la suma, resta y multiplicación de matrices pasadas como parámetros. Para el caso de la suma y la resta, las matrices deben ser del mismo tamaño, en caso de no serlo devolver null. Para el caso de la multiplicación la cantidad de columnas de A debe ser igual a la cantidad de filas de B, en caso contrario devolver null.

```
static double[,] suma(double[,] A, double[,] B)
static double[,] resta(double[,] A, double[,] B)
static double[,] multiplicacion(double[,] A, double[,] B)
```