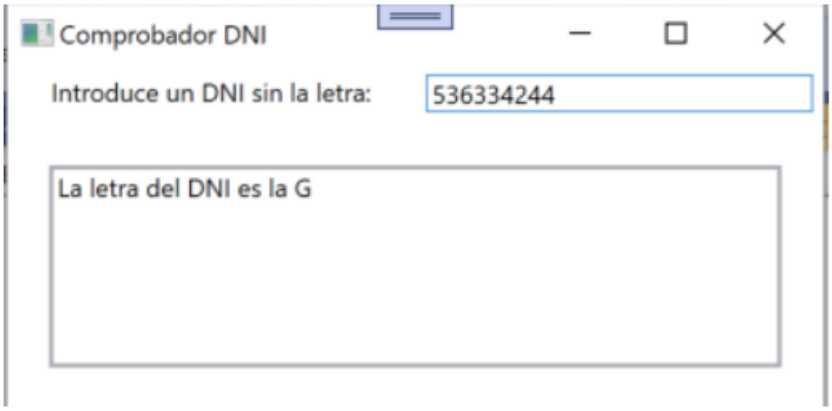
Práctica10. Ejercicios completos C# WPF

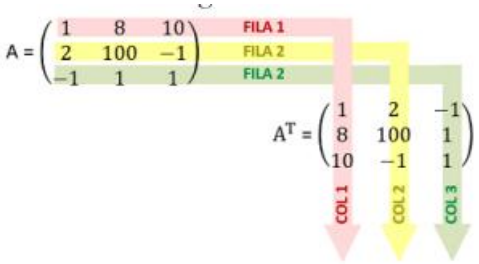
Ejercicio 1. TeLlamoEnLaSiesta S.L. es una empresa que intenta captar clientes para nuevas altas de telefonía. Esta empresa acaba de hacernos un encargo: han detectado que sus operadores, en muchos casos, cometen errores tipográficos a la hora de almacenar el DNI de un nuevo cliente, lo que provoca que el alta no termine de completarse por datos incorrectos. Esto supone grandes pérdidas para la empresa. Con el fin de reducir estos costes, nos solicitan que le diseñemos un programa que, dado un DNI sin la letra, nos devuelva la letra que corresponde. De esta forma el operador, si se equivoca en algún dígito, podrá comprobar que la letra no coincide con la que indica el cliente y tendrá la oportunidad de corregirlo . Por lo tanto, el programa a realizar recibirá por teclado un número de 8 dígitos, y devolverá la letra que corresponde a esos dígitos. Esa letra se calcula conforme al siguiente algoritmo:

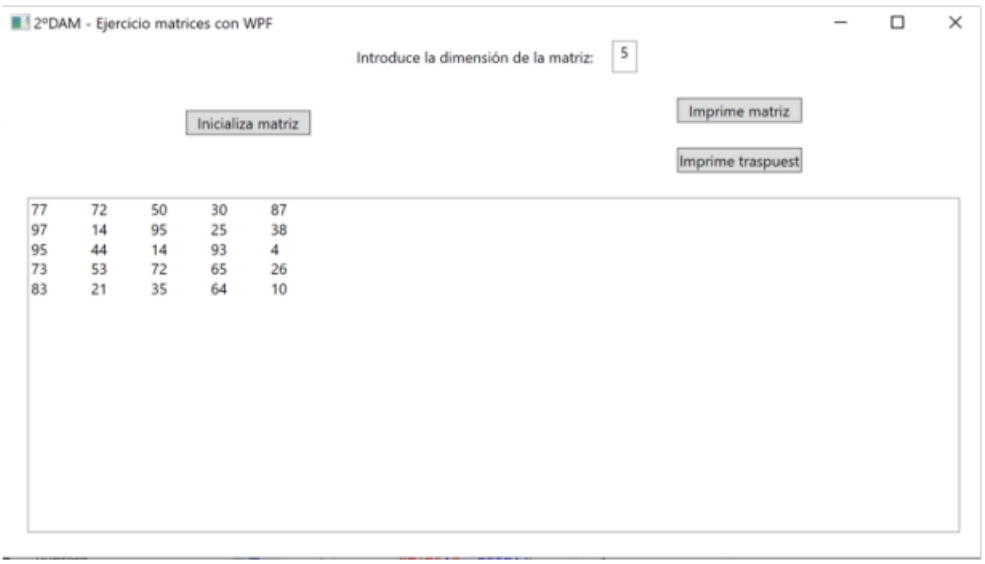
0	1	2	3	4	5	6	7	8	9	10	11
T	R	W	A	G	M	Y	F	P	D	X	B



Ejercicio 2. El profesor de matemáticas del IES Francisco Rodríguez Marín nos pide ayuda para que le realicemos un programa que automáticamente genere matrices bidimensionales. El usuario de la aplicación introducirá por teclado el número de filas/columnas (igual número de filas que de columnas). El programa generará la matriz con números aleatorios entre 0 y 99. Por ejemplo, si el usuario introduce el número 5, podemos ver una matriz con 5 columas y 5 filas. El programa mostrará por pantalla la matriz generada. Y además mostrará la matriz traspuesta.

NOTA. La matriz traspuesta se calcula de la siguiente forma:





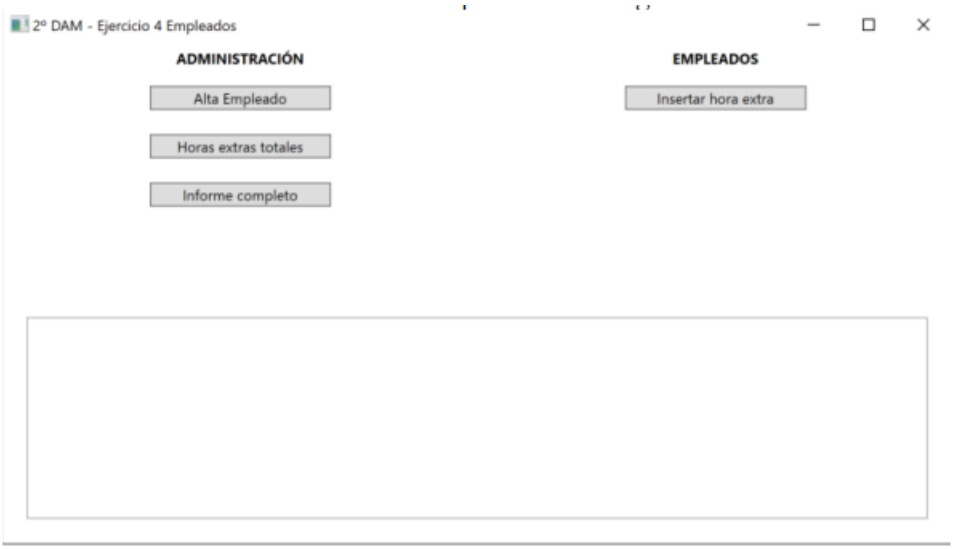
Ejercicio 3. Se nos encarga un programa para organizar tareas. Para ello, hemos analizado varias aplicaciones existentes en el mercado, pero todas ellas tienen algún coste o inconveniente. El programa deberá tener una forma parecida a esta:

La explicación de cada opción es la siguiente:

- Agregar nueva tarea: Debe agregar a la cola un nuevo elemento string con la descripción de la tarea a realizar.

- Mostrar siguiente: Debe mostrar un mensaje como el siguiente: “La siguiente tarea que tienes que realizar es XXX”, donde XXX se obtendrá de la cola, pero sin extraer el elemento de dicha cola.
- Realizar siguiente: En esta opción SÍ se extraerá la tarea de la cola.
- Mostrar nº de tareas: Mostrará el número de tareas pendientes por realizar.
- Mostrar pendientes: Hará un resumen por pantalla de las tareas que están encoladas pendientes de realizar.

Ejercicio 4. La empresa OsunaTech. S.L. contrata nuestros servicios para implantar un sistema de control de horas extras de sus empleados. Nuestra aplicación será instalada en un ordenador a la entrada de la oficina, por lo que deberá tener SIEMPRE visible un menú de opciones de la siguiente forma:



Como se ve en este menú, hay opciones disponibles para la administración y otras para los empleados. El acceso a las opciones de Administración debe estar protegido mediante contraseña (debes solicitar contraseña para poder acceder).

Debes modelar este problema mediante la clase Empleado. Cada objeto de la clase Empleado se almacenará en un array de máximo 15 posiciones, que tendrás que inicializar a “null” en la primera ejecución del programa. Se recomienda crear una variable para controlar el número de objetos insertados en ese array. La funcionalidad que debe implementar cada opción del menú es la siguiente:

- Alta Empleado: permite dar de alta a un nuevo empleado en el sistema. Para ello, solicitará su nombre y salario. Con estos datos, agregará el nuevo objeto al array o lista de empleados.
- Horas extras totales: mostrará por pantalla el número de horas extra en total que han realizado todos los empleados dados de alta en el sistema.
- Informe completo: mostrará por pantalla un informe completo de todos los empleados que debe incluir: nombre del empleado, salario, horas extras, salario total incluyendo extras.

- Insertar Hora Extra: pedirá al usuario su nombre, y a continuación el número de horas extras a insertar. Si el usuario no existe, indicará el error por pantalla al usuario.

Debes implementar en un fichero independiente la clase Empleado, en la que tendrás que crear lo siguiente:

- Tendrá los atributos(campos) “nombre”, “numHorasExtra” y “salarioSinExtras”.
- Constructor que asigne al objeto creado el nombre establecido (string) y que inicialice numHorasExtra a 0 y salarioSinExtras al salario que se le indique (el constructor llevará dos argumentos).
- Método HaceHoraExtra que recibirá como argumento el número de horas extra que hace el empleado en un determinado momento, y la sumará a “numHorasExtra”.
- Método CalculaSalarioTotal, que devolverá el salario mensual total, y que calculará con la siguiente fórmula: salario total = salarioSinExtras + 25*numHorasExtra.