EJERCICIOS

Ejercicio 1

El objetivo del ejercicio es realizar la suma y resta de unos vectores cuya longitud y valores los introducirá el usuario a través del teclado.

Tendremos 4 arrays, Z, K, J y H, que seguirán la siguiente fórmula H = K – Z + J.

Comenzaremos por crear nuestro paquete en Visual Studio creando un nuevo proyecto de Aplicación de consola con C#. Llamaremos al proyecto **Ejercicio1** y elegiremos como ubicación del proyecto nuestra carpeta **Tema8**.

Al crear nuestro proyecto nos encontraremos con una pantalla como esta

```
using System;

namespace Ejercicio1
{
    class Program
    {
        static void Main(string[] args)
        {
            Console.WriteLine("Hello World!");
        }
    }
}
```

Cambiaremos el nombre de la clase a **RestarArray** y crearemos los 4 arrays necesarios para el ejercicio.

```
class RestarArray {
  int[] Z;
  int[] K;
  int[] J;
  int[] H;
}
```

Crearemos un método **Cargar(),** donde pediremos al usuario que introduzca la longitud de los arrays, crearemos una variable de tipo string, le diremos al método que lea el string y haremos una conversión de string a int en una variable e inicializaremos los arrays en función de la longitud.

```
public void Cargar(){
    Console.Write("Inserte la longitud de los arrays a restar y sumar: ");
    string linea;
    linea = Console.ReadLine();
    int n = int.Parse(linea);
    Z = new int[n];
    K = new int[n];
    J = new int[n];
    H = new int[n];
```

Crearemos una línea por consola que nos diga que estamos introduciendo los valores del primer array.

Crearemos un bucle ***for*** que recorrerá el array e irá metiendo los valores del usuario en función de la longitud antes especificada.

```
Console.WriteLine("Introduciendo los valores en el array Z");
for (int i = 0; i < Z.Length; i++){
    Console.Write("Introduzca componente[" + (i + 1) + "]: ");
    linea = Console.ReadLine();
    Z[i] = int.Parse(linea);
}
```

Repetiremos el mismo proceso del primer array con el segundo array pero cambiando la variable de dentro del bucle para que al recorrerlo no coja los valores del primer bucle.

```
Console.WriteLine("Introduciendo valores en el array K");
for (int j = 0; j < K.Length;j++){
    Console.Write("Introduzca componenete[" + (j + 1) + "]: ");
    linea = Console.ReadLine();
    K[j] = int.Parse(linea);
}
```

Seguiremos con la introducción de valores en los arrays con el 3er array que tenemos.

```
Console.WriteLine("Introduciendo valores en el array J");
for (int z = 0; z < J.Length; z++){
    Console.Write("Introduzca componente[" + (z + 1) + "]: ");
    linea = Console.ReadLine();
    J[z] = int.Parse(linea);
}
```

Y por último, crearemos un último bucle que rellenará el último array pero utilizando los valores de los otros arrays y realizando las operaciones que hayamos especificado.

```
for (int i = 0; i < Z.Length; i++){
    H[i] = K[i] - Z[i] + J[i];
}
```

Una vez creado el método Cargar(), crearemos el método **Visualizar**(), que usaremos para que se vea por pantalla el resultado final de las operaciones.

Para ello escribiremos por consola una línea donde irá el resultado, luego crearemos un bucle para que pase cada uno de los valores correspondientes a cada operación.

```
public void Visualizar() {
  Console.WriteLine("La suma y resta de los arrays es : ");

  for (int i = 0; i < Z.Length; i++) {
    Console.Write("[" + H[i] + "]");
  }

  Console.ReadLine();
}
```

Y por último, crearemos el método Main donde instanciaremos un objeto del tipo RestarArray() y llamaremos a los métodos creados.

```
static void Main(string[]args){
    RestarArray c = new RestarArray();
    c.Cargar();
    c.Visualizar();
}
```

Para poder ejecutar el programa que hemos escrito será necesario pulsar el botón *Iniciar* arriba a la izquierda en Visual Studio:

```
Inserte la longitud de los arrays a restar y sumar: 2
Introduciendo los valores en el array Z
Introduzca componente[1]: 1
Introduzca componente[2]: 2
Introduciendo valores en el array K
Introduzca componenete[1]: 3
Introduzca componenete[2]: 4
Introduciendo valores en el array J
Introduzca componente[1]: 2
Introduzca componente[2]: 1
La suma y resta de los arrays es :
[4][3]
```

Ejercicio 2

El objetivo del ejercicio es realizar la suma y resta de unos vectores cuya longitud y valores los introducirá el usuario a través de teclado.

El ejercicio seguirá la siguiente fórmula: C = A + B

Comenzaremos el ejercicio como el anterior, creando un proyecto nuevo.

Una vez creado el paquete nuevo y se nos haya creado la estructura base de una clase de C#, cambiaremos el nombre de la clase a **SumaMatriz**.

```
class SumaMatriz {

}
```

Una vez cambiado el nombre de la clase, procederemos a crear tres matrices de nombres A, B y C.

```
private int[,] MatrizA;
private int[,] MatrizB;
private int[,] MatrizC;
```

Ahora crearemos un método **Cargar()** que le pedirá al usuario por teclado todos los valores que poseen las matrices.

Comenzaremos por declarar las matrices creadas con anterioridad.

```
MatrizA = new int[10, 10];
MatrizB = new int[10, 10];
MatrizC = new int[10, 10];
```

A continuación crearemos una instrucción que nos diga por pantalla que vamos a introducir los valores de la primera matriz.

```
Console.WriteLine("Introduciendo datos en la matriz A");
```

Pasaremos ahora a crear los bucles ***for*** anidados para poder introducir los datos en la matriz dado que tiene dos dimensiones.

```
for (int i = 0; i <= 3; i++){
    for (int j = 0; j <= 3; j++){
        Console.Write("Introduzca posición[" + i + "," + j + "]: ");
        string linea;
        linea = Console.ReadLine();
        MatrizA[i, j] = int.Parse(linea);
    }
}
```

A continuación haremos lo mismo pero para la matriz B.

```
Console.WriteLine("Introduciendo datos en la matriz B");
for (int i = 0; i <= 3; i++){
    for (int j = 0; j <= 3; j++){
        Console.Write("Introduzca posición[" + i + "," + j + "]: ");
        string linea;
        linea = Console.ReadLine();
        MatrizB[i, j] = int.Parse(linea);
    }
}
```

Para finalizar el método Cargar(), crearemos los bucles necesarios para introducir los valores que hemos recibido del usuario para la matriz A y la matriz B, en la matriz C.

```
for (int i = 0; i <= 3; i++){
    for (int j = 0; j <= 3; j++){
        MatrizC[i, j] = MatrizA[i, j] + MatrizB[i, j];
    }
}
```

Ahora crearemos un método **Visualizar**() que nos muestre por pantalla todos los datos anteriormente introducidos.

```
public void Visualizar(){
    Console.WriteLine("La suma de la matrizA y MatrizB es: ");
    for (int i = 0; i <= 3; i++){
        Console.Write("\n");
        for (int j = 0; j <= 3; j++){
            Console.Write(MatrizC[i, j] + " ");
        }
    }
    Console.ReadKey();
}
```

Y en última instancia crearemos el método **Main**() donde crearemos un objeto de la clase SumaMatriz y llamaremos a los métodos Cargar() y Visualizar().

```
static void Main(string[] args){
    SumaMatriz sm = new SumaMatriz();
    sm.Cargar();
    sm.Visualizar();
}
```

El resultado de la ejecución es el siguiente:

```
Introduciendo datos en la matriz A
Introduzca posición[0,0]: 12
Introduzca posición[0,1]: 2
Introduzca posición[0,2]: 3
Introduzca posición[0,3]: 4
Introduzca posición[1,0]: 5
Introduzca posición[1,1]: 6
Introduzca posición[1,2]: 7
Introduzca posición[1,3]: 8
Introduzca posición[2,0]: 8
Introduzca posición[2,1]: 9
Introduzca posición[2,2]: 2
Introduzca posición[2,3]: 3
Introduzca posición[3,0]: 4
Introduzca posición[3,1]: 5
Introduzca posición[3,2]: 6
Introduzca posición[3,3]: 6
Introduciendo datos en la matriz B
Introduzca posición[0,0]: 77
Introduzca posición[0,1]: 8
Introduzca posición[0,2]: 8
Introduzca posición[0,3]: 8
Introduzca posición[1,0]: 8
Introduzca posición[1,1]: 8
Introduzca posición[1,2]: 8
Introduzca posición[1,3]: 8
Introduzca posición[2,0]: 8
Introduzca posición[2,1]: 8
Introduzca posición[2,2]: 8
Introduzca posición[2,3]: 8
Introduzca posición[3,0]: 8
Introduzca posición[3,1]: 8
Introduzca posición[3,2]: 8
Introduzca posición[3,3]: 8
La suma de la matrizA y MatrizB es:

89 10 11 12
13 14 15 16
16 17 10 11
12 13 14 14 8
8
```

Ejercicio 3

Crearemos un nuevo proyecto llamado **Personas_T8** que contendrá una clase que llamaremos **Persona**.

En esta clase incluiremos tres atributos, dos de tipo **string**: **Nombre** y **Apellido**, y un atributo de tipo **int**: **Edad**. Los atributos de esta clase serán públicos. También incluiremos un método constructor al que le pasaremos 3 parámetros e instanciaremos nuestros atributos a estos parámetros.

Sobreescribiremos el método **ToString**() para que nos devuelva el nombre de la persona, el apellido y su edad.

El código es el siguiente:

```
public class Persona
{
    public string Nombre;
    public string Apellido;
    public int Edad;

    6 referencias
    public Persona(string nombre, string apellido, int edad)
    {
        this.Nombre = nombre;
        this.Apellido = apellido;
        this.Edad = edad;
    }

    0 referencias
    public override string ToString()
    {
        return Nombre + " " + Apellido + " " + Edad;
    }
```

Una vez creada esta clase, en el **Main** de la clase **Program**, nuestra clase principal, crearemos 5 instancias de la clase **Persona**, con los siguientes parámetros:

- José Sanz 34
- Andrés Ruiz 40
- Sara Martínez 60
- Sonia López 37
- Pepe Vázquez 50

```
Persona p1 = new Persona("José", "Sanz", 34);
Persona p2 = new Persona("Andres", "Ruiz", 40);
Persona p3 = new Persona("Sara", "Martinez", 60);
Persona p4 = new Persona("Sonia", "López", 37);
Persona p5 = new Persona("Pepe", "Vazquez", 50);
Console.WriteLine(p1);
Console.WriteLine(p2);
Console.WriteLine(p3);
Console.WriteLine(p4);
Console.WriteLine(p5);
Console.ReadKey();
```

Ahora crearemos un array de objetos Persona, en el que incluiremos las 5 personas que hemos creado anteriormente. Además en la clase principal crearemos un método que nos devuelva un objeto de tipo **Persona** y reciba como parámetros un array de objetos de tipo **Persona** y busquemos entre estos la persona con mayor edad, es decir, debemos recorrer el array y buscar la persona con mayor edad, mostrarla por pantalla y devolver el objeto correspondiente. Este método se llamará **PersonasArray**().

```
Persona[] ArrayPersonas = new Persona[5];
ArrayPersonas[0] = p1;
ArrayPersonas[1] = p2;
ArrayPersonas[2] = p3;
ArrayPersonas[3] = p4;
ArrayPersonas[4] = p5;
Console.WriteLine("La persona mayor es:");
Console.WriteLine(PersonasArray(ArrayPersonas));
Console.ReadKey();
```

A continuación crearemos dos métodos en la clase **Persona** que nos permitan crear personas aleatorias seleccionando **nombres** y **apellidos** que guardaremos dentro de dos listas. Primero crearemos el método **NombreRandom**, dentro de este crearemos una variable llamada listadoNombres con los siguientes nombres:

Pepe, Paco, Jose, Juan, Andrés, Raúl, Pablo, Carlos

Crearemos un número aleatorio mediante: new Random().Next(7) para obtener un número entre 0 y 6. Este número decidirá cuál será el escogido de la lista.

Random().Next() ejecutado dentro de un método en sucesivas llamadas devuelve siempre el mismo número. Para solucionar esto deberemos crear una variable global de clase del siguiente modo:

```
public static Random numAleatorio=new Random();
```

En el método deberemos llamar al método Next[1] del objeto **numAleatorio**

Recorreremos el array y devolveremos el string correspondiente.

```
public static string NombreRandom()
{
    List<string> listadoNombres = new List<string>() { "Pepe", "Paco", "Jose",
        "Juan", "Andres", "Raul", "Pablo", "Carlos" };
    int numeroAleatorio = numAleatorio.Next(8);
    string nombreElegido = listadoNombres[numeroAleatorio];
    return nombreElegido;
}
```

Realizaremos lo mismo para el apellido, incluyendo estos apellidos en la lista:

Ruiz, López, Sáez, Martínez, Navarro, Fernández, Núñez, Pérez

[1] **Next** devuelve un número aleatorio no negativo que es menor que el valor máximo especificado.

El funcionamiento del método debe ser el mismo, creando un número aleatorio y devolviendo el apellido elegido.

```
public static string ApellidoRandom()
{
    List<string> listadoApellidos = new List<string>() { "Ruiz", "López", "Sáez",
        "Martínez", "Navarro", "Fernández", "Núñez", "Pérez" };
    int numeroAleatorio=numAleatorio.Next(8);
    string apellidoElegido = listadoApellidos[numeroAleatorio];
    return apellidoElegido;
}
```

Para concluir nuestra implementación para obtener personas aleatorias, crearemos un nuevo constructor dentro de la clase **Persona** que no tome ningún parámetro e inicializaremos los atributos de **Persona** llamando a los métodos que seleccionan aleatoriamente N**ombreRandom** y **ApellidoRandom** y al campo **Edad** le daremos el valor de un número aleatorio que tenga como valor máximo 100, utilizando el método Random.Next() que ya hemos utilizado anteriormente.

Tenemos dos formas:

```
public Persona()
{
    Nombre = NombreRandom();
    Apellido = ApellidoRandom();
    Edad = numAleatorio.Next(100);
}
```

O

```
public Persona():this(NombreRandom(), ApellidoRandom(), numAleatorio.Next(100))
{

}
```

Para aprovechar los cambios realizados en la clase **Persona**, nos dirigimos a la clase **Program** y crearemos 5 personas más utilizando este nuevo constructor y añadiremos estas nuevas personas a nuestro array de personas.

```
Persona per1 = new Persona();
Persona per2 = new Persona();
Persona per3 = new Persona();
Persona per4 = new Persona();
Persona per5 = new Persona();
Array.Resize(ref ArrayPersonas, ArrayPersonas.Length + 5);
ArrayPersonas[5] = per1;
ArrayPersonas[6] = per2;
ArrayPersonas[7] = per3;
ArrayPersonas[8] = per4;
ArrayPersonas[9] = per5;
Console.WriteLine("Personas Aleatorias:");
Console.WriteLine(ArrayPersonas[5]);
Console.WriteLine(ArrayPersonas[6]);
Console.WriteLine(ArrayPersonas[7]);
Console.WriteLine(ArrayPersonas[8]);
Console.WriteLine(ArrayPersonas[9]);
Console.WriteLine("La persona mayor es:");
Console.WriteLine(PersonasArray(ArrayPersonas));
Console.ReadKey();
```

Como hemos visto en el tema, es posible crear un **Diccionario** que almacene pares **<clave, valor>**. En nuestro caso crearemos un diccionario que almacene pares **<Persona, int>**. Vamos a utilizar este diccionario para almacenar los hijos que tienen estas personas.

Dentro de nuestra clase **Program** crearemos un método CrearDiccionaro para crear el nuevo diccionario. Recorreremos el array de personas y añadiremos al diccionario sólo las personas que sean mayores de 18 años. Para almacenar el número de hijos crearemos un nuevo Random cuyo valor máximo sea 4.

Recuerda crear un campo global de clase igual que en la clase Persona.

```
public static Random numAleatorio = new Random();
```

En el método también recorreremos el diccionario y mostraremos cada una de las personas en el siguiente formato:

```
Persona: Nº hijos:
```

El método **CrearDiccionario** queda como se muestra a continuación:

```
public static OrderedDictionary CrearDiccionario(Persona[] ArrayPersonas)
{ // Crea el diccionario con las personas mayores de 18 años
  // y el número de hijos que tiene cada una.
    OrderedDictionary dicOrd = new OrderedDictionary();
    int randomHijos;
    int edadPersona = 0;
    for (int i = 0; i < ArrayPersonas.Length; i++)
    {
        Persona personaDevuelta = new Persona();
        edadPersona = ArrayPersonas[i].Edad;
        if (edadPersona >= 18)
        {
            randomHijos = numAleatorio.Next(5);
            dicOrd.Add(ArrayPersonas[i], randomHijos);
            // Muestra los valores que se insertan en el diccionario
            Console.WriteLine("Persona: {0} Nº hijos: {1} ",
                ArrayPersonas[i], randomHijos);
        }
    }
    return dicOrd;
}
```

Llamaremos a este método desde Main():

```
OrderedDictionary dicPersonas = new OrderedDictionary();
dicPersonas = CrearDiccionario(ArrayPersonas);
```

Después desde Main() recorremos el diccionario y mostraremos cada una de las personas en el siguiente formato:

```
José Sanz 34,0
```
Nombre Apellido Edad, Nº hijos

Se realiza una llamada al método **ToString**() que sobreescribimos en la clase **Persona** y le añadimos a esto el número de hijos.

```
Console.WriteLine("Recorremos el diccionario y presentamos " +
    "las Personas Aleatorias >= 18 y el número de hijos aleatorios(0-4)-EXTRACCIÓN DEL MÉTODO");
foreach (DictionaryEntry entrada in (dicPersonas))
{
    Console.WriteLine("{0},{1}", entrada.Key, entrada.Value);
}
Console.ReadKey();
```

A continuación crearemos un método llamado **EliminaHijos0** que devuelva un diccionario y reciba un diccionario como parámetro. En él volveremos a recorrer el diccionario que hemos creado anteriormente y, en este caso, eliminaremos las entradas de nuestro diccionario que no tengan hijos, es decir, todas aquellas cuyo valor de hijos sea 0 serán eliminadas de nuestro diccionario.

Para ello recorremos el diccionario que nos pasan como parámetro y guardamos en un segundo diccionario las personas que sí tienen hijos.

```
public static OrderedDictionary EliminaHijos0(OrderedDictionary dicc)
{
    //Diccionario para almacenar el número de hijos de las personas
    // mayores de edad
    OrderedDictionary dicConHijos = new OrderedDictionary();
    foreach (DictionaryEntry entrada in dicc)
    {
        int valorhijos = int.Parse(string.Format("{0}", entrada.Value));
        if (valorhijos != 0)
        {
            dicConHijos.Add(entrada.Key, entrada.Value);
        }
    }
    return dicConHijos;
}
```

Además, desde Main llamamos al método anterior y recorremos el diccionario devuelto por el mismo tras eliminar las entradas con hijos=0.

```
Console.WriteLine("Recorremos el diccionario devuelto por el método tras eliminar las entradas con hijo=0");
OrderedDictionary dicConHijos = new OrderedDictionary();
dicConHijos = EliminaHijos0(dicPersonas);
foreach (DictionaryEntry entrada in dicConHijos)
{
    Console.WriteLine("{0}, {1}", entrada.Key, entrada.Value);
}
Console.ReadKey();
```

Por último, vamos a crear un array bidimensional en el que cada subarray (cada fila de la matriz) contenga algo relacionado con las personas que tenemos creadas en el **dicConHijos**. Así, cada fila de la matriz va a tener en su primer elemento la persona que está almacenada en **dicConHijos**. Los siguientes elementos del subarray van a ser los hijos, esos hijos serán tantos como venga indicado en el elemento **Value** del diccionario. Los nombres de los hijos se crearán de forma aleatoria mediante el constructor sin parámetros de la clase **Persona**.

- Comenzaremos creando un array bidimensional de tipo **Persona** cuyo nombre será **arrayConHijos** y su tamaño será: **[dicConHijos.Count, 5]**, además deberemos crear un **contador** de tipo **int** que inicializaremos a **0** y servirá para contar el número de personas del diccionario.

```
Persona[,] arrayConHijos = new Persona[dicConHijos.Count, 5];
int contador = 0; // Lleva la cuenta del número de pesonas del diccionario
```

A continuación recorreremos el diccionario mediante un bucle **foreach**. Dentro de él crearemos el objeto **Persona** padre, que será el primero de cada fila de la matriz y los objetos **Persona** hijo, que serán los hijos creados de forma aleatoria del primero. El **padre** se obtendrá a partir de la clave del diccionario **dicConHijos** y el número de hijos se obtiene del valor del diccionario. En la posición **[contador, 0]** siempre se encontrará el padre.

Además dentro del bucle crearemos otro bucle **for** que recorrerá la matriz desde **i=1** hasta **i<=valor+1** e incrementaremos **i**.

En este bucle crearemos una nueva **Persona** hijo dentro de cada posición **[contador, i]**.

Además incrementaremos **contador** al final del bucle **foreach**.

El código es el siguiente:

```
foreach (DictionaryEntry entrada in dicConHijos)
{
    // Obtengo del diccionaro el objeto Persona llamado padre
    Persona padre = (Persona)entrada.Key;
    // Introduzco el padre en la matriz
    arrayConHijos[contador, 0] = padre;
    Console.WriteLine("arrayConHijos({0}, {1})={2}", contador, 0, padre);

    // Introduzco los hijos en la matriz

    // Número de hijos de la persona
    //int entradaInt = int.Parse(string.Format("{0}", entrada.Value));
   int entradaInt = (int)entrada.Value;

    for (int i = 1; i <= entradaInt; i++)
     {
       Persona hijo = new Persona();
       arrayConHijos[contador, i] = hijo;
       Console.WriteLine("arrayConHijos({0}, {1}) = {2}",
           contador, i, arrayConHijos[contador, i]);
     }
    Console.WriteLine();
    // Paso al siguiente elemento del diccionario que será la siguiente fila de la matriz
    contador++;
}
Console.ReadKey();
```

Ejercicio 4

Vamos a crear en este ejercicio un diccionario que contenga personajes de Star Wars.

Mediante la clase genérica de diccionarios *Dictionary<TKey, TValue>*, recorreremos el diccionario e indicaremos qué personajes son Jedi y cuáles no.

Comenzaremos por crear un nuevo proyecto de tipo consola.

Una vez tengamos el proyecto creado, lo primero que tenemos que hacer es importar la librería *System.Collections.Generic* para poder usar el diccionario.

```
using System.Collection.Generic;
```

A continuación, crearemos el diccionario que tendrá la pareja clave/valor como string y boolean.

```
Dictionary<string, bool> starWars = new Dictionary<string, bool>();
```

Añadiremos unos cuantos elementos al diccionario mediante el método *Add* en el objeto diccionario que hemos creado antes.

```
starWars.Add("Luke Skywalker", true);
starWars.Add("Leia Organa", false);
starWars.Add("Han Solo", false);
starWars.Add("Yoda", true);
starWars.Add("Obi-Wan Kenobi", true);
starWars.Add("Chewbacca", false);
starWars.Add("Darth Vader", true);
starWars.Add("Kylo Ren", true);
```

Ahora crearemos el bucle que recorrerá el diccionario usando un *foreach* y definiremos la condición para que se nos muestre por consola si el elemento del diccionario es Jedi.

```
foreach(KeyValuePair<string, bool> pareja in starWars) {
  if(pareja.Value) {
    Console.WriteLine(pareja.Key + " es Jedi");
  } else {
    Console.WriteLine(pareja.Key + " no es Jedi");
  }
}
```

Cuando lo ejecutemos nos saldrá lo siguiente por pantalla:

```
Luke Skywalker es Jedi
Leia Organa no es Jedi
Han Solo no es Jedi
Yoda es Jedi
Obi-Wan Kenobi es Jedi
Chewbacca no es Jedi
Darth Vader es Jedi
Kylo Ren es Jedi
```

Ejercicio 5

En este ejercicio haremos algo similar al ejercicio anterior pero con *Queues*.

Introduciremos nombres en la lista y, posteriormente, mediante un bucle, los quitaremos de la lista.

Igual que en el ejercicio anterior exportaremos *System.Collections.Generic* para poder trabajar con la clase genérica de Queue.

```
using System.Collection.Generic
```

Ahora crearemos un Queue que se llamará **nombres** y que tendrá un parámetro de tipo String.

```
Queue<string> nombres = new Queue<string>();
```

Añadiremos varios elementos a la cola mediante el método *Enqueue* que vimos en la teoría.

```
nombres.Enqueue("Rick");
nombres.Enqueue("Morty");
nombres.Enqueue("Squanchy");
nombres.Enqueue("Summer");
nombres.Enqueue("Jerry");
nombres.Enqueue("Unity");
```

Crearemos un bucle *while* ahora para recorrer la *Queue* que tendrá como condición del bucle que la propiedad *Count* sea mayor que 0.

Dentro del bucle crearemos una variable **nombre** que haremos que quite los nombres de la cola.

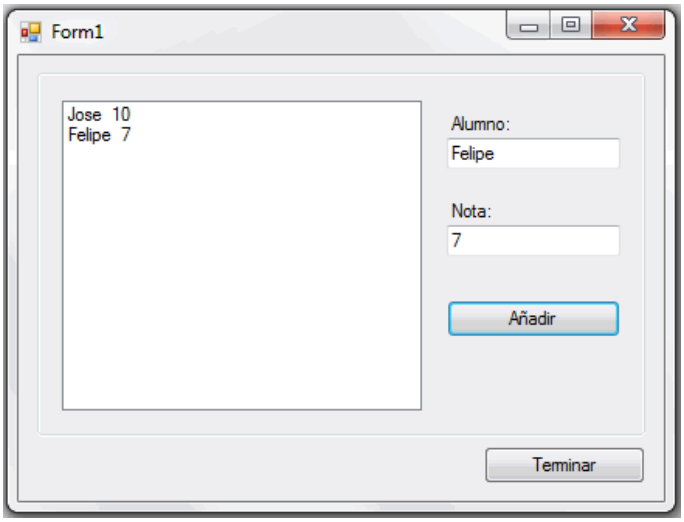
```
while (nombres.Count > 0) {
  string nombre = nombres.Dequeue();
  Console.WriteLine(nombre);
}
```

Ejercicio 6

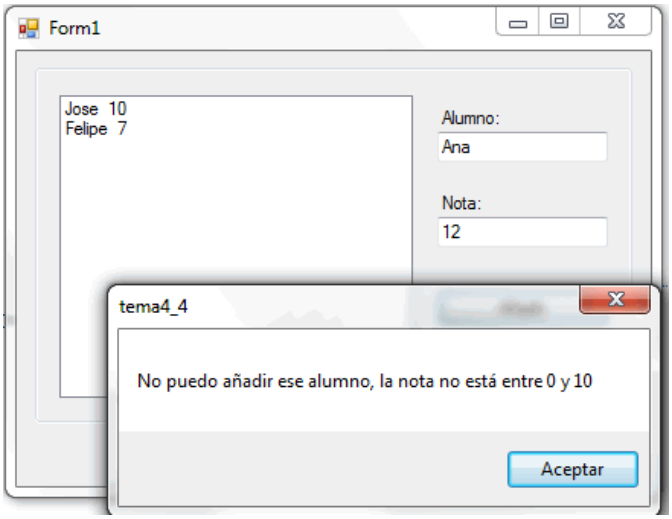
Crea una aplicación de Windows. Define una estructura:

Alumno
Nombre de tipo string
Nota de tipo real

Pon unos cuadros de texto para leer los valores e introduce la pareja de valores en un cuadro de lista sólo si los valores no están vacíos.



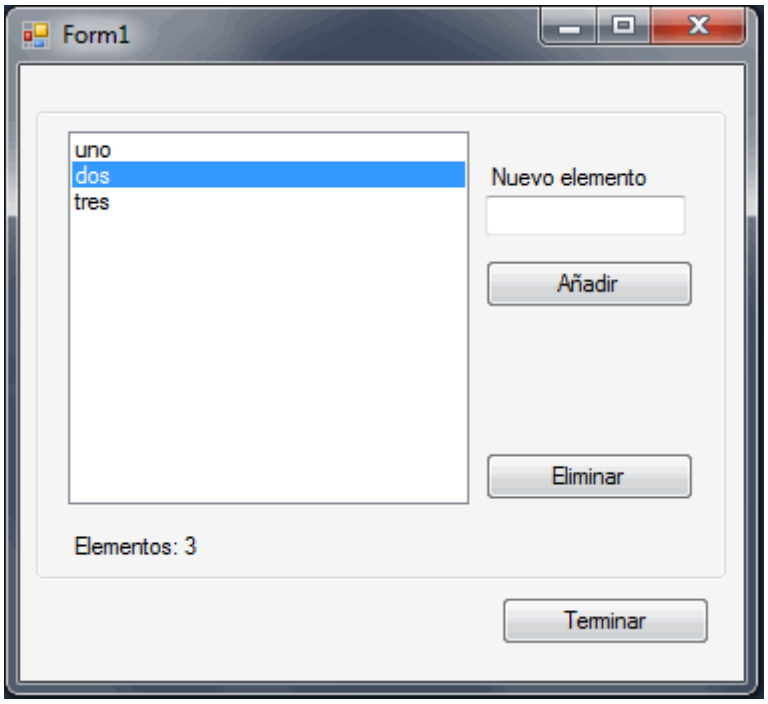
Además debemos comprobar el valor leído para que la nota esté entre los valores 0-10:



Nota: Utiliza MessageBox para escribir un mensaje y una función de conversión para el valor leído en el cuadro de texto de la nota.

Ejercicio 7

Crea una aplicación Windows con un cuadro de texto, un cuadro de lista, un botón y un label que indique cuántos elementos hay en la lista.

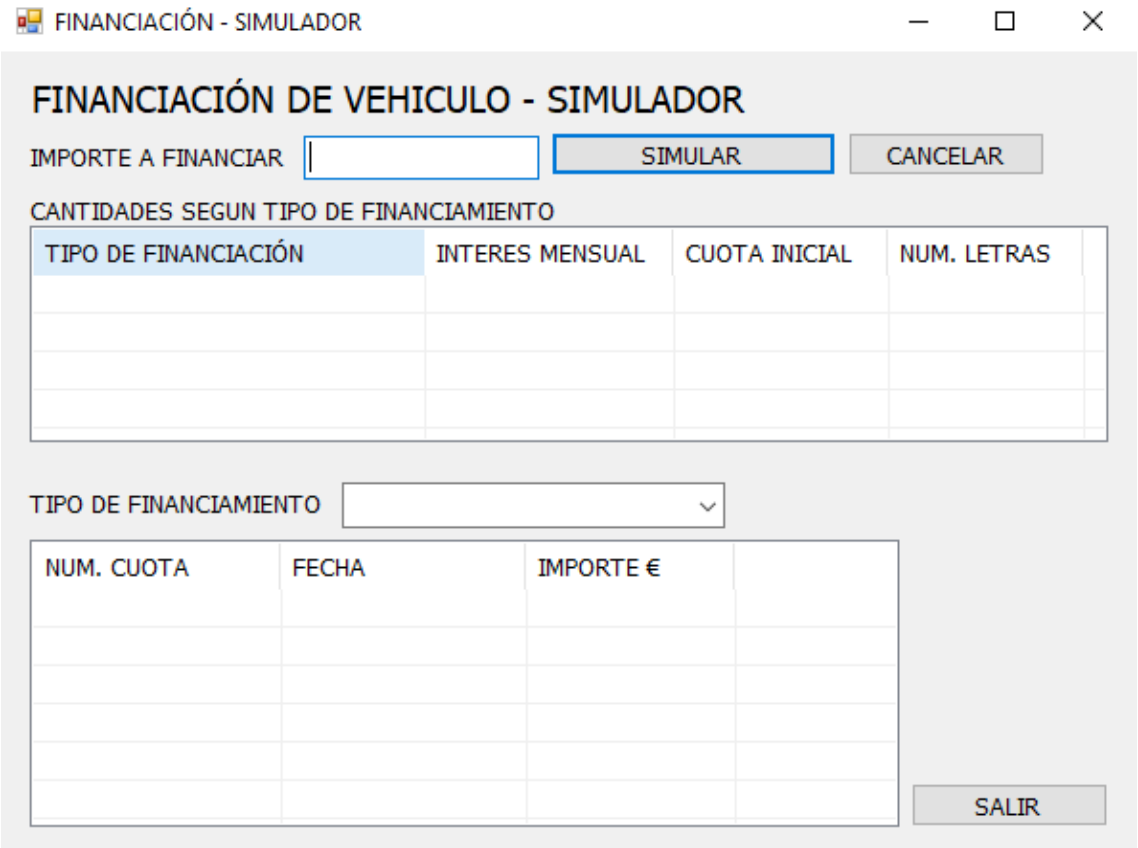


Escribe métodos independientes para insertar y eliminar elementos del cuadro de lista. Comprueba que sólo se inserten elementos si hay algo en el cuadro de texto y que se eliminen elementos sólo si hay alguno seleccionado.

Utiliza una variable global para almacenar el número de elementos de la lista y un método para escribir este número de elementos.

Ejercicio 8

Simulación de la financiación de un vehículo. Una empresa de venta de automóviles ofrece una serie de facilidades para la adquisición de nuevos vehículos. Necesitan una aplicación que permita introducir el importe real del vehículo que será financiado. La aplicación mostrará los tipos de financiación, el interés mensual, la cuota inicial y el número de letras de todos los tipos de financiación que ofrece la empresa. Luego, al seleccionar un tipo de financiación deberá mostrar el número de letras, las fechas de pago y las cuotas mensuales en caso de que el cliente acepte la financiación.



Hay que tener en cuenta los siguientes aspectos:

- Implementar un formulario que contemple los datos solicitados en la aplicación.
- Introducir el importe a financiar. Tened en cuenta que dicho valor genera todos los resultados en la aplicación.
- Los porcentajes, cuota inicial y número de letras se dan bajo la siguiente tabla.

Tipos de financiación	Interés mensual	Cuota inicial	Número de letras
Personal	5%	60%	2
Banco	10%	50%	4
Fondo Colectivo	15%	35%	6

- Mostrar todos los importes en euros.
- El botón *SIMULAR* es el encargado de mostrar información en el control ListView, como todos los tipos de financiación, los importes correspondientes al interés mensual, cuota inicial y número de letras según el tipo de financiación. Este botón tiene que estar asociado a la tecla <Enter>.

  El **interés mensual** se calcula mediante el porcentaje correspondiente aplicado al importe a financiar.

  La **cuota inicial** se calcula mediante la siguiente fórmula:

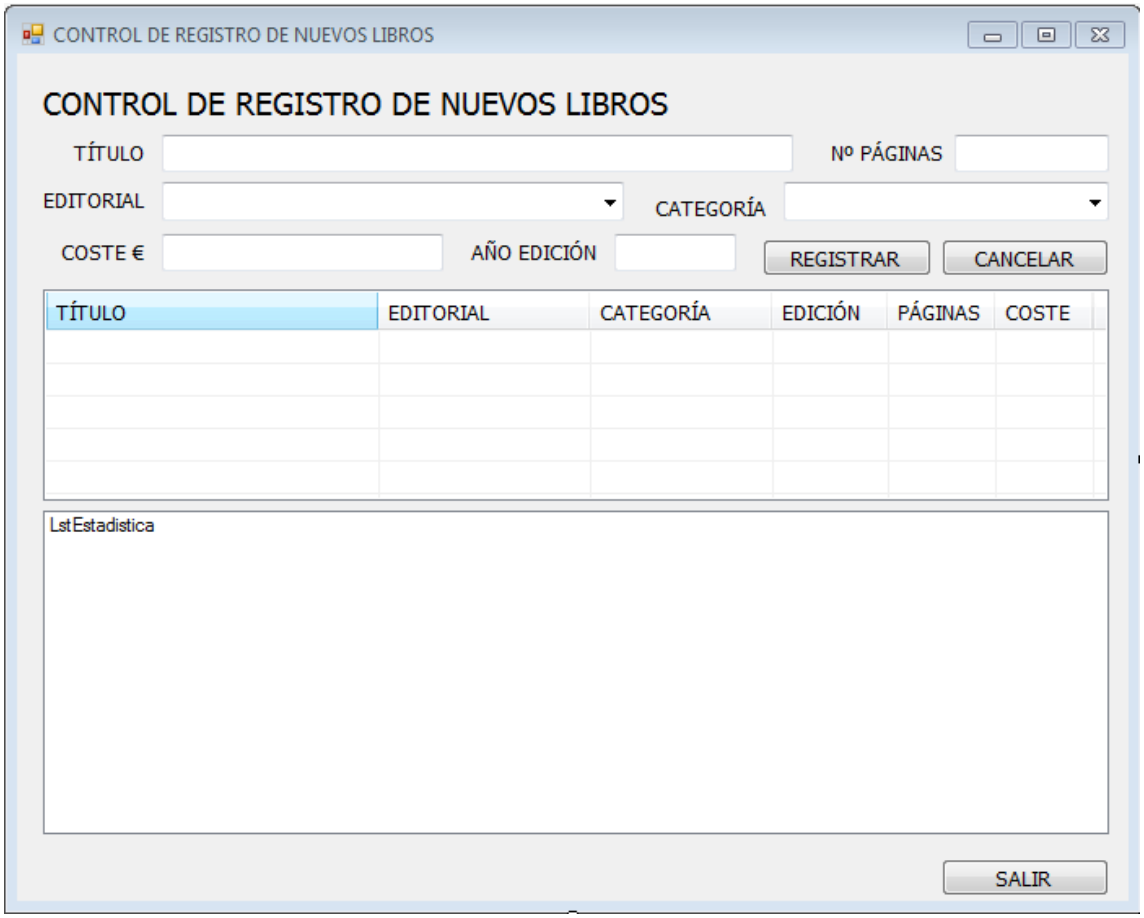
  (importe + interés)* porcentaje de cuota inicial expresado en la tabla anterior
- En un ComboBox hay que mostrar los tipos de financiación. Al seleccionar uno de ellos se debe mostrar en una lista el número de cuota, fecha e importe según el tipo de financiación elegido en el listado anterior.

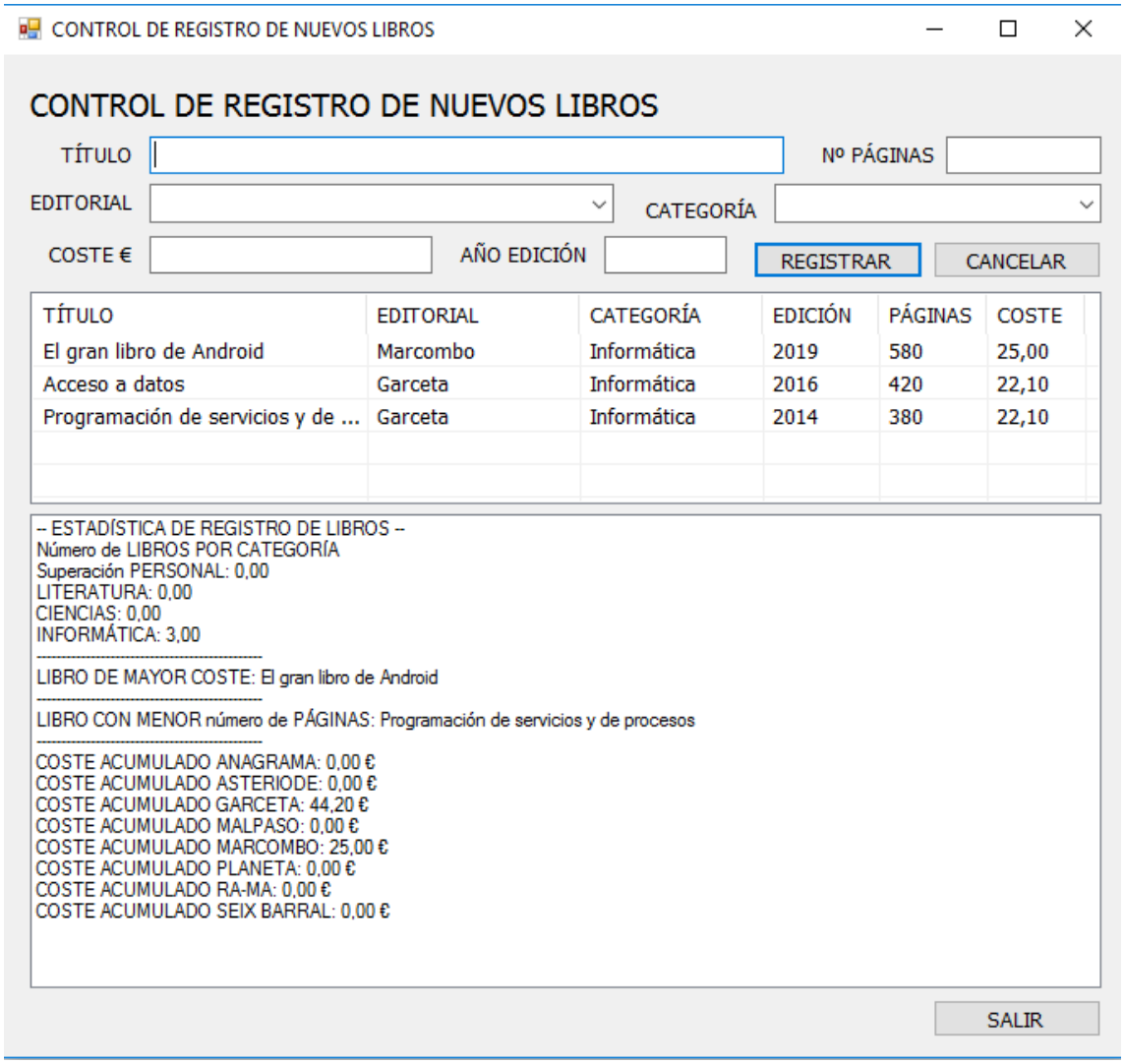
  El importe de la cuota mensual se calcula según la siguiente fórmula:

  (importe + interés mensual – cuota inicial)/ número de letras
- Las fechas se dan de acuerdo al número de cuotas que le corresponde al tipo de financiación. Además, debe mostrar las fechas próximas de pago usando la función *DateAdd*.
- Implementar la salida de la aplicación usando un cuadro de mensaje que se encuentre asociado al botón <Esc>

Ejercicio 9

Control de registro de nuevos libros. Un centro de formación adquiere libros nuevos y usados a fin de año, de diferentes entidades en forma de donación. En vista de esto necesitan una aplicación que permita controlar el registro de dichos libros como título, número de páginas, editorial, categoría, coste y año de edición del libro. Al final debe mostrar algunas características solicitadas por la administración, como el número total de libros adquiridos según la categoría, el título del libro cuyo coste sea mayor a los demás, el título del libro cuyo número de páginas es menor a los demás y, finalmente, el coste acumulado según la editorial.





Hay que tener en cuenta los siguientes aspectos:

- Implementar un formulario que contemple los datos solicitados en la aplicación.
- Mostrar todos los precios en euros.
- El botón *REGISTRAR* es el encargado de mostrar información en el control ListView, como título, editorial, categoría, año de edición, número de páginas y el coste del libro registrado, además de mostrar las estadísticas solicitadas para el caso. Este botón tiene que estar asociado a la tecla <Enter>.
- El botón *CANCELAR* es el encargado de limpiar el contenido de los controles, para un nuevo registro de datos del libro.
- Implementar la salida de la aplicación usando un cuadro de mensaje que se encuentre asociado al botón <Esc>