4.4.6. Otras manipulaciones de cadenas

Ya hemos comentado que las cadenas en C# son inmutables, no se pueden modificar. Pero sí podemos realizar ciertas operaciones sobre ellas para obtener una nueva cadena. Por ejemplo:

- ToUpper() convierte a mayúsculas: nombreCorrecto = nombre.ToUpper();
- ToLower() convierte a minúsculas: password2 = password.ToLower();
- Insert(int posición, string subcadena): Insertar una subcadena en una cierta posición de la cadena inicial: nombreFormal = nombre.Insert(0,"Don");
- Remove(int posición, int cantidad): Elimina una cantidad de caracteres en cierta posición: apellidos = nombreCompleto.Remove(0,6);
- Replace(string textoASustituir, string cadenaSustituta): Sustituye una cadena (todas las veces que aparezca) por otra: nombreCorregido = nombre.Replace("Pepe", "Jose");

Un programa que probara todas estas posibilidades podría ser así:

```
/*---------------------------*/
/*  Ejemplo en C# nº 43:     */
/*  ejemplo43.cs             */
/*                           */
/*  Cadenas de texto (2)     */
/*                           */
/*  Introduccion a C#,       */
/*    Nacho Cabanes          */
/*---------------------------*/

using System;

public class Ejemplo43
{

  public static void Main()
  {

    string ejemplo = "Hola, que tal estas";

    Console.WriteLine("El texto es: {0}",
      ejemplo);

    Console.WriteLine("La primera letra es: {0}",
      ejemplo[0]);

    Console.WriteLine("Las tres primeras letras son: {0}",
      ejemplo.Substring(0,3));

    Console.WriteLine("La longitud del texto es: {0}",
      ejemplo.Length);

    Console.WriteLine("La posicion de \"que\" es: {0}",
      ejemplo.IndexOf("que"));

    Console.WriteLine("La ultima A esta en la posicion: {0}",
```

```
      ejemplo.LastIndexOf("a"));

    Console.WriteLine("En mayúsculas: {0}",
      ejemplo.ToUpper());

    Console.WriteLine("En minúsculas: {0}",
      ejemplo.ToLower());

    Console.WriteLine("Si insertamos \", tio\": {0}",
      ejemplo.Insert(4,", tio"));

    Console.WriteLine("Si borramos las 6 primeras letras: {0}",
      ejemplo.Remove(0, 6));

    Console.WriteLine("Si cambiamos ESTAS por ESTAMOS: {0}",
      ejemplo.Replace("estas", "estamos"));

  }
}
```

Y su resultado sería

```
El texto es: Hola, que tal estas
La primera letra es: H
Las tres primeras letras son: Hol
La longitud del texto es: 19
La posicion de "que" es: 6
La ultima A esta en la posicion: 17
En mayúsculas: HOLA, QUE TAL ESTAS
En minúsculas: hola, que tal estas
Si insertamos ", tio": Hola, tio, que tal estas
Si borramos las 6 primeras letras: que tal estas
Si cambiamos ESTAS por ESTAMOS: Hola, que tal estamos
```

Ejercicios propuestos:

- **(4.4.6.1)** Una variante del ejercicio 4.4.5.2, que no distinga entre mayúsculas y minúsculas a la hora de buscar.
- **(4.4.6.2)** Un programa que pida al usuario una frase y elimine todos los espacios redundantes que contenga (debe quedar sólo un espacio entre cada palabra y la siguiente).

Otra posibilidad interesante, aunque un poco más avanzada, es la de **descomponer una cadena** en trozos, que estén separados por una serie de delimitadores (por ejemplo, espacios o comas). Para ello se puede usar **Split**, que crea un array a partir de los fragmentos de la cadena, así:

```
/*--------------------------*/
/*  Ejemplo en C# nº 43b:   */
/*  ejemplo43b.cs           */
/*                          */
```

```
/*  Cadenas de texto (2b)    */
/*                           */
/*  Introduccion a C#,       */
/*    Nacho Cabanes          */
/*--------------------------*/

using System;

public class Ejemplo43b
{

        public static void Main()
        {

                string ejemplo = "uno,dos.tres,cuatro";
                char [] delimitadores = {',', '.'};
                int i;

                string [] ejemploPartido = ejemplo.Split(delimitadores);

                for (i=0; i<ejemploPartido.Length; i++)
                        Console.WriteLine("Fragmento {0}= {1}",
                            i, ejemploPartido[i]);

        }
}
```

Que mostraría en pantalla lo siguiente:

```
Fragmento 0= uno
Fragmento 1= dos
Fragmento 2= tres
Fragmento 3= cuatro
```

Ejercicios propuestos:

- **(4.4.6.3)** Un programa que pida al usuario una frase y muestre sus palabras en orden inverso.
- **(4.4.6.4)** Un programa que pida al usuario varios números separados por espacios y muestre su suma.

4.4.7. Comparación de cadenas

Sabemos comprobar si una cadena tiene exactamente un cierto valor, con el operador de igualdad (==), pero no sabemos comparar qué cadena es "mayor" que otra, algo que es necesario si queremos ordenar textos. El operador "mayor que" (>) que usamos con los números no se puede aplicar directamente a las cadenas. En su lugar, debemos usar "CompareTo", que devolverá un número mayor que 0 si la nuestra cadena es mayor que la que indicamos como parámetro (o un número negativo si nuestra cadena es menor, o 0 si son iguales):

```
if (frase.CompareTo("hola") > 0)
        Console.WriteLine("Es mayor que hola");
```

También podemos comparar sin distinguir entre mayúsculas y minúsculas, usando String.Compare, al que indicamos las dos cadenas y un tercer dato "true" cuando queramos ignorar esa distinción:

```
if (String.Compare(frase, "hola", true) > 0)
       Console.WriteLine("Es mayor que hola (mays o mins)");
```

Un programa completo de prueba podría ser así:

```
/*--------------------------*/
/*  Ejemplo en C# nº 43c:   */
/*  ejemplo43c.cs           */
/*                          */
/*  Cadenas de texto (2c)   */
/*                          */
/*  Introduccion a C#,      */
/*    Nacho Cabanes         */
/*--------------------------*/

using System;

public class Ejemplo43c
{

       public static void Main()
       {

              string frase;

              Console.WriteLine("Escriba una palabra");
              frase = Console.ReadLine();

              // Compruebo si es exactamente hola
              if (frase == "hola")
                     Console.WriteLine("Ha escrito hola");

              // Compruebo si es mayor o menor
              if (frase.CompareTo("hola") > 0)
                     Console.WriteLine("Es mayor que hola");
              else if (frase.CompareTo("hola") < 0)
                     Console.WriteLine("Es menor que hola");

              // Comparo sin distinguir mayúsculas ni minúsculas
              bool ignorarMays = true;
              if (String.Compare(frase, "hola", ignorarMays) > 0)
                     Console.WriteLine("Es mayor que hola (mays o mins)");
              else if (String.Compare(frase, "hola", ignorarMays) < 0)
                     Console.WriteLine("Es menor que hola (mays o mins)");
              else
                     Console.WriteLine("Es hola (mays o mins)");

       }
}
```

Si tecleamos una palabra como "gol", que comienza por G, que alfabéticamente está antes de la H de "hola", se nos dirá que esa palabra es menor:

```
Escriba una palabra
gol
Es menor que hola
Es menor que hola (mays o mins)
```

Si escribimos "hOLa", que coincide con "hola" salvo por las mayúsculas, una comparación normal nos dirá que es mayor (las mayúsculas se consideran "mayores" que las minúsculas), y una comparación sin considerar mayúsculas o minúsculas nos dirá que coinciden:

```
Escriba una palabra
hOLa
Es mayor que hola
Es hola (mays o mins)
```

Ejercicios propuestos:

- **(4.4.7.1)** Un programa que pida al usuario cinco frases, las guarde en un array y muestre la "mayor" de ellas (la que aparecería en último lugar en un diccionario).

4.4.8. Una cadena modificable: StringBuilder

Si tenemos la necesidad de poder modificar una cadena letra a letra, no podemos usar un "string" convencional, deberemos recurrir a un "StringBuilder", que sí lo permiten pero son algo más complejos de manejar: hay de reservarles espacio con "new" (igual que hacíamos en ciertas ocasiones con los Arrays), y se pueden convertir a una cadena "convencional" usando "ToString":

```
/*--------------------------*/
/*  Ejemplo en C# nº 44:    */
/*  ejemplo44.cs            */
/*                          */
/*  Cadenas modificables    */
/*  con "StringBuilder"     */
/*                          */
/*  Introduccion a C#,      */
/*    Nacho Cabanes         */
/*--------------------------*/

using System;
using System.Text; // Usaremos un System.Text.StringBuilder

public class Ejemplo44
{

  public static void Main()
  {
    StringBuilder cadenaModificable = new StringBuilder("Hola");
    cadenaModificable[0] = 'M';
    Console.WriteLine("Cadena modificada: {0}",
```

```
      cadenaModificable);

    string cadenaNormal;
    cadenaNormal = cadenaModificable.ToString();
    Console.WriteLine("Cadena normal a partir de ella: {0}",
      cadenaNormal);
  }
}
```

Ejercicios propuestos:

- **(4.4.8.1)** Un programa que pida una cadena al usuario y la modifique, de modo que las letras de las posiciones impares (primera, tercera, etc.) estén en minúsculas y las de las posiciones pares estén en mayúsculas, mostrando el resultado en pantalla. Por ejemplo, a partir de un nombre como "Nacho", la cadena resultante sería "nAcHo".
- **(4.4.8.2)** Un programa que pida tu nombre, tu día de nacimiento y tu mes de nacimiento y lo junte todo en una cadena, separando el nombre de la fecha por una coma y el día del mes por una barra inclinada, así: "Juan, nacido el 31/12".
- **(4.4.8.3)** Crear un juego del ahorcado, en el que un primer usuario introduzca la palabra a adivinar, se muestre esta programa oculta con guiones (-----) y el programa acepte las letras que introduzca el segundo usuario, cambiando los guiones por letras correctas cada vez que acierte (por ejemplo, a---a-t-). La partida terminará cuando se acierte la palabra por completo o el usuario agote sus 8 intentos.

4.4.9. Recorriendo con "foreach"

Existe una construcción parecida a "for", pensada para recorrer ciertas estructuras de datos, como los arrays (y otras que veremos más adelante).

Se usa con el formato "foreach (variable in ConjuntoDeValores)":

```
/*----------------------------*/
/*  Ejemplo en C# nº 45:      */
/*  ejemplo45.cs              */
/*                            */
/*  Ejemplo de "foreach"      */
/*                            */
/*  Introduccion a C#,        */
/*    Nacho Cabanes           */
/*----------------------------*/

using System;

public class Ejemplo45
{

  public static void Main()
  {
    int[] diasMes = {31, 28, 21};
    foreach(int dias in diasMes) {
      Console.WriteLine("Dias del mes: {0}", dias);
```

```
    }

    string[] nombres = {"Alberto", "Andrés", "Antonio"};
    foreach(string nombre in nombres) {
      Console.Write(" {0}", nombre);
    }
    Console.WriteLine();

    string saludo = "Hola";
    foreach(char letra in saludo) {
      Console.Write("{0}-", letra);
    }
    Console.WriteLine();

  }
}
```

Ejercicios propuestos:

- **(4.4.9.1)** Un programa que pida al usuario una frase y la descomponga en subcadenas separadas por espacios, usando "Split". Luego debe mostrar cada subcadena en una línea nueva, usando "foreach".
- **(4.4.9.2)** Un programa que pida al usuario varios números separados por espacios y muestre su suma (como el del ejercicio 4.4.6.4), pero empleando "foreach".

4.5 Ejemplo completo

Vamos a hacer un ejemplo completo que use tablas ("arrays"), registros ("struct") y que además manipule cadenas.

La idea va a ser la siguiente: Crearemos un programa que pueda almacenar datos de hasta 1000 ficheros (archivos de ordenador). Para cada fichero, debe guardar los siguientes datos: Nombre del fichero, Tamaño (en KB, un número de 0 a 8.000.000.000). El programa mostrará un menú que permita al usuario las siguientes operaciones:

1- Añadir datos de un nuevo fichero
2- Mostrar los nombres de todos los ficheros almacenados
3- Mostrar ficheros que sean de más de un cierto tamaño (por ejemplo, 2000 KB).
4- Ver todos los datos de un cierto fichero (a partir de su nombre)
5- Salir de la aplicación (como no usamos ficheros, los datos se perderán).

No debería resultar difícil. Vamos a ver directamente una de las formas en que se podría plantear y luego comentaremos alguna de las mejoras que se podría (incluso se debería) hacer.

Una opción que podemos a tomar para resolver este problema es la de contar el número de fichas que tenemos almacenadas, y así podremos añadir de una en una. Si tenemos 0 fichas, deberemos almacenar la siguiente (la primera) en la posición 0; si tenemos dos fichas, serán la 0 y la 1, luego añadiremos en la posición 2; en general, si tenemos "n" fichas, añadiremos cada nueva ficha en la posición "n". Por otra parte, para revisar todas las fichas, recorreremos desde la posición 0 hasta la n-1, haciendo algo como

```
for (i=0; i<=n-1; i++) { ... más órdenes ...}
```

o bien algo como

```
for (i=0; i<n; i++) { ... más órdenes ...}
```

El resto del programa no es difícil: sabemos leer y comparar textos y números, comprobar varias opciones con "switch", etc. Aun así, haremos una última consideración: hemos limitado el número de fichas a 1000, así que, si nos piden añadir, deberíamos asegurarnos antes de que todavía tenemos hueco disponible.

Con todo esto, nuestro fuente quedaría así:

```
/*--------------------------*/
/*  Ejemplo en C# nº 46:    */
/*  ejemplo46.cs            */
/*                          */
/*  Tabla con muchos struct */
/*  y menu para manejarla   */
/*                          */
/*  Introduccion a C#,      */
/*    Nacho Cabanes         */
/*--------------------------*/

using System;

public class Ejemplo46
{

  struct tipoFicha {
     public string nombreFich;   /* Nombre del fichero */
     public long tamanyo;        /* El tamaño en KB */
  };

  public static void Main()
  {
    tipoFicha[] fichas   /* Los datos en si */
       = new tipoFicha[1000];
    int numeroFichas=0;  /* Número de fichas que ya tenemos */
    int i;               /* Para bucles */
    int opcion;          /* La opcion del menu que elija el usuario */
    string textoBuscar;  /* Para cuando preguntemos al usuario */
    long tamanyoBuscar;  /* Para buscar por tamaño */


    do {
      /* Menu principal */
      Console.WriteLine();
      Console.WriteLine("Escoja una opción:");
      Console.WriteLine("1.- Añadir datos de un nuevo fichero");
      Console.WriteLine("2.- Mostrar los nombres de todos los ficheros");
      Console.WriteLine("3.- Mostrar ficheros que sean de mas de un cierto tamaño");
      Console.WriteLine("4.- Ver datos de un fichero");
      Console.WriteLine("5.- Salir");

      opcion = Convert.ToInt32( Console.ReadLine() );
```

```
        /* Hacemos una cosa u otra según la opción escogida */
        switch(opcion){
           case 1: /* Añadir un dato nuevo */
             if (numeroFichas < 1000) {  /* Si queda hueco */
               Console.WriteLine("Introduce el nombre del fichero: ");
               fichas[numeroFichas].nombreFich = Console.ReadLine();
               Console.WriteLine("Introduce el tamaño en KB: ");
               fichas[numeroFichas].tamanyo  = Convert.ToInt32( Console.ReadLine() );
               /* Y ya tenemos una ficha más */
               numeroFichas++;
             } else   /* Si no hay hueco para más fichas, avisamos */
               Console.WriteLine("Máximo de fichas alcanzado (1000)!");
             break;
           case 2: /* Mostrar todos */
             for (i=0; i<numeroFichas; i++)
                Console.WriteLine("Nombre: {0}; Tamaño: {1} KB",
                  fichas[i].nombreFich, fichas[i].tamanyo);
             break;
           case 3: /* Mostrar según el tamaño */
             Console.WriteLine("¿A partir de que tamaño quieres que te muestre?");
             tamanyoBuscar = Convert.ToInt64( Console.ReadLine() );
             for (i=0; i<numeroFichas; i++)
               if (fichas[i].tamanyo >= tamanyoBuscar)
                  Console.WriteLine("Nombre: {0}; Tamaño: {1} KB",
                    fichas[i].nombreFich, fichas[i].tamanyo);
             break;
           case 4: /* Ver todos los datos (pocos) de un fichero */
             Console.WriteLine("¿De qué fichero quieres ver todos los datos?");
             textoBuscar = Console.ReadLine();
             for (i=0; i<numeroFichas; i++)
               if ( fichas[i].nombreFich == textoBuscar )
                  Console.WriteLine("Nombre: {0}; Tamaño: {1} KB",
                    fichas[i].nombreFich, fichas[i].tamanyo);
             break;
           case 5: /* Salir: avisamos de que salimos */
             Console.WriteLine("Fin del programa");
             break;
           default: /* Otra opcion: no válida */
             Console.WriteLine("Opción desconocida!");
             break;
           }
        } while (opcion != 5);  /* Si la opcion es 5, terminamos */

     }
}
```

Funciona, y hace todo lo que tiene que hacer, pero es mejorable. Por supuesto, en un caso real es habitual que cada ficha tenga que guardar más información que sólo esos dos apartados de ejemplo que hemos previsto esta vez. Si nos muestra todos los datos en pantalla y se trata de muchos datos, puede ocurrir que aparezcan en pantalla tan rápido que no nos dé tiempo a leerlos, así que sería deseable que parase cuando se llenase la pantalla de información (por ejemplo, una pausa tras mostrar cada 25 datos). Por supuesto, se nos pueden ocurrir muchas más preguntas que hacerle sobre nuestros datos. Y además, cuando salgamos del programa se borrarán todos los datos que habíamos tecleado, pero eso es lo único "casi inevitable", porque aún no sabemos manejar ficheros.

```
public class Ejemplo
{
    ...

    public  LanzarJuego ()    {
      Juego j = new Juego();
      j.ComienzoPartida ();
      ...
```

Ejercicio propuesto:

- **(7.1.1)** Crea una nueva versión del ejercicio 5.2.3, en la que métodos y variables no sean "static".

7.2. Arrays de objetos

Es muy frecuente que no nos baste con tener un objeto de cada clase, sino que necesitemos manipular varios objetos pertenecientes a la misma clase. En ese caso, podríamos almacenar todos ellos en un "array". Al declararlo, deberemos reservar memoria primero para el array, y luego para cada uno de los elementos. Por ejemplo, podríamos tener un array de 5 perros, que crearíamos de esta forma:

```
Perro[] misPerros = new Perro[5];
for (byte i = 0; i < 5; i ++)
  misPerros[i] = new Perro();
```

Un fuente completo de ejemplo podría ser

```
/*--------------------------*/
/*  Ejemplo en C# nº 63:    */
/*  ejemplo63.cs            */
/*                          */
/*  Quinto ejemplo de clases */
/*  Array de objetos        */
/*                          */
/*  Introduccion a C#,      */
/*    Nacho Cabanes         */
/*--------------------------*/

using System;

public class Animal
{

  public Animal()
  {
    Console.WriteLine("Ha nacido un animal");
  }
}

// -----------------

public class Perro: Animal
```

```
{

  public Perro()
  {
    Console.WriteLine("Ha nacido un perro");
  }
}

// ------------------

public class Ejemplo63
{

  public static void Main()
  {
    Perro[] misPerros = new Perro[5];
    for (byte i = 0; i < 5; i ++)
      misPerros[i] = new Perro();
  }

}
```

y su salida en pantalla, parecida a la del ejemplo anterior, sería

```
Ha nacido un animal
Ha nacido un perro
Ha nacido un animal
Ha nacido un perro
Ha nacido un animal
Ha nacido un perro
Ha nacido un animal
Ha nacido un perro
Ha nacido un animal
Ha nacido un perro
```

Ejercicio propuesto:

- **(7.2.1)** Crea una versión ampliada del ejercicio 6.8.1, en la que no se cree un único objeto de cada clase, sino un array de tres objetos.

Además, existe una peculiaridad curiosa: podemos crear un array de "Animales", pero luego indicar que unos de ellos son perros, otros gatos, etc.,

```
Animal[] misAnimales = new Animal[3];

misAnimales[0] = new Perro();
misAnimales[1] = new Gato();
misAnimales[2] = new GatoSiames();
```

Un ejemplo más detallado:

```
/*---------------------------*/
/*  Ejemplo en C# nº 64:     */
/*  ejemplo64.cs             */
/*                           */
/*  Ejemplo de clases        */
/*  Array de objetos de      */
/*    varias subclases       */
/*                           */
/*  Introduccion a C#,       */
/*    Nacho Cabanes          */
/*---------------------------*/

using System;

public class Animal
{

  public Animal()
  {
    Console.WriteLine("Ha nacido un animal");
  }
}

// ------------------

public class Perro: Animal
{

  public Perro()
  {
    Console.WriteLine("Ha nacido un perro");
  }
}

// ------------------

public class Gato: Animal
{

  public Gato()
  {
    Console.WriteLine("Ha nacido un gato");
  }
}


// ------------------

public class GatoSiames: Gato
{

  public GatoSiames()
  {
    Console.WriteLine("Ha nacido un gato siamés");
  }
}

// ------------------
```

```
public class Ejemplo64
{

  public static void Main()
  {
    Animal[] misAnimales = new Animal[8];

    misAnimales[0] = new Perro();
    misAnimales[1] = new Gato();
    misAnimales[2] = new GatoSiames();

    for (byte i=3; i<7; i++)
      misAnimales[i] = new Perro();

    misAnimales[7] = new Animal();
  }

}
```

La salida de este programa sería:

```
Ha nacido un animal
Ha nacido un perro
Ha nacido un animal
Ha nacido un gato
Ha nacido un animal
Ha nacido un gato
Ha nacido un gato siamés
Ha nacido un animal
Ha nacido un perro
Ha nacido un animal
Ha nacido un perro
Ha nacido un animal
Ha nacido un perro
Ha nacido un animal
Ha nacido un perro
Ha nacido un animal
```

7.3. Funciones virtuales. La palabra "override"

En el ejemplo anterior hemos visto cómo crear un array de objetos, usando sólo la clase base, pero insertando realmente objetos de cada una de las clases derivadas que nos interesaba, y hemos visto que los constructores se llaman correctamente... pero con los métodos puede haber problemas.

Vamos a verlo con un ejemplo, que en vez de tener constructores va a tener un único método "Hablar", que se redefine en cada una de las clases hijas, y después comentaremos qué ocurre al ejecutarlo:

```
/*--------------------------*/
/*  Ejemplo en C# nº 65:    */
/*  ejemplo65.cs            */
```