4. Arrays, estructuras y cadenas de texto

4.1. Conceptos básicos sobre arrays o tablas

4.1.1. Definición de un array y acceso a los datos

Una tabla, vector, matriz o **array** (que algunos autores traducen por "**arreglo**") es un conjunto de elementos, todos los cuales son del mismo tipo. Estos elementos tendrán todos el mismo nombre, y ocuparán un espacio contiguo en la memoria.

Por ejemplo, si queremos definir un grupo de números enteros, el tipo de datos que usaremos para declararlo será "int[]". Si sabemos desde el principio cuantos datos tenemos (por ejemplo 4), les reservaremos espacio con "= new int[4]", así

```
int[] ejemplo = new int[4];
```

Podemos acceder a cada uno de los valores individuales indicando su nombre (ejemplo) y el número de elemento que nos interesa, pero con una precaución: se empieza a numerar desde 0, así que en el caso anterior tendríamos 4 elementos, que serían ejemplo[0], ejemplo[1], ejemplo[2], ejemplo[3].

Como ejemplo, vamos a definir un grupo de 5 números enteros y hallar su suma:

```
/*---------------------------*/
/*  Ejemplo en C# nº 33:     */
/*  ejemplo33.cs             */
/*                           */
/*  Primer ejemplo de tablas */
/*                           */
/*  Introduccion a C#,       */
/*    Nacho Cabanes          */
/*---------------------------*/

using System;

public class Ejemplo33
{
  public static void Main()
  {

    int[] numero = new int[5];           /* Un array de 5 números enteros */
    int suma;                /* Un entero que será la suma */

    numero[0] = 200;       /* Les damos valores */
    numero[1] = 150;
    numero[2] = 100;
    numero[3] = -50;
    numero[4] = 300;
    suma = numero[0] +     /* Y hallamos la suma */
        numero[1] + numero[2] + numero[3] + numero[4];
    Console.WriteLine("Su suma es {0}", suma);
    /* Nota: esta es la forma más ineficiente e incómoda */
    /* Ya lo iremos mejorando */
  }
```

```
}
```

Ejercicios propuestos:

- **(4.1.1.1)** Un programa que pida al usuario 4 números, los memorice (utilizando una tabla), calcule su media aritmética y después muestre en pantalla la media y los datos tecleados.
- **(4.1.1.2)** Un programa que pida al usuario 5 números reales (pista: necesitarás un array de "float") y luego los muestre en el orden contrario al que se introdujeron.
- **(4.1.1.3)** Un programa que pida al usuario 4 números enteros y calcule (y muestre) cuál es el mayor de ellos. Nota: para calcular el mayor valor de un array, hay que cada uno de los valores que tiene almacenados con el que hasta ese momento es el máximo. El valor inicial de este máximo no debería ser cero (porque fallaría si todos los números son negativos), sino el primer elemento del array.

4.1.2. Valor inicial de un array

Al igual que ocurría con las variables "normales", podemos dar valor a los elementos de una tabla al principio del programa. Será más cómodo que dar los valores uno por uno, como hemos hecho antes, pero sólo se podrá hacer si conocemos todos los valores. En este caso, los indicaremos todos entre llaves, separados por comas:

```
/*---------------------------*/
/*  Ejemplo en C# nº 34:     */
/*  ejemplo34.cs             */
/*                           */
/*  Segundo ejemplo de       */
/*  tablas                   */
/*                           */
/*  Introduccion a C#,       */
/*    Nacho Cabanes          */
/*---------------------------*/

using System;

public class Ejemplo34
{
  public static void Main()
  {

    int[] numero =            /* Un array de 5 números enteros */
      {200, 150, 100, -50, 300};
    int suma;                 /* Un entero que será la suma */

    suma = numero[0] +    /* Y hallamos la suma */
        numero[1] + numero[2] + numero[3] + numero[4];
    Console.WriteLine("Su suma es {0}", suma);
    /* Nota: esta forma es algo menos engorrosa, pero todavía no */
    /* está bien hecho.  Lo seguiremos mejorando */
  }
}
```

Ejercicios propuestos:

- **(4.1.2.1)** Un programa que almacene en una tabla el número de días que tiene cada mes (supondremos que es un año no bisiesto), pida al usuario que le indique un mes (1=enero, 12=diciembre) y muestre en pantalla el número de días que tiene ese mes.

4.1.3. Recorriendo los elementos de una tabla

Es de esperar que exista una forma más cómoda de acceder a varios elementos de un array, sin tener siempre que repetirlos todos, como hemos hecho en

```
suma = numero[0] + numero[1] + numero[2] + numero[3] + numero[4];
```

El "truco" consistirá en emplear cualquiera de las estructuras repetitivas que ya hemos visto (while, do..while, for), por ejemplo así:

```
    suma = 0;            /* Valor inicial de la suma */
    for (i=0; i<=4; i++) /* Y hallamos la suma repetitiva */
      suma += numero[i];
```

El fuente completo podría ser así:

```
/*--------------------------*/
/*  Ejemplo en C# nº 35:    */
/*  ejemplo35.cs            */
/*                          */
/*  Tercer ejemplo de       */
/*  tablas                  */
/*                          */
/*  Introduccion a C#,      */
/*    Nacho Cabanes         */
/*--------------------------*/

using System;

public class Ejemplo35
{
  public static void Main()
  {

    int[] numero =           /* Un array de 5 números enteros */
      {200, 150, 100, -50, 300};
    int suma;                 /* Un entero que será la suma */
    int i;                    /* Para recorrer los elementos */

    suma = 0;            /* Valor inicial de la suma */
    for (i=0; i<=4; i++) /* Y hallamos la suma repetitiva */
      suma += numero[i];

      Console.WriteLine("Su suma es {0}", suma);
  }
}
```

En este caso, que sólo sumábamos 5 números, no hemos escrito mucho menos, pero si trabajásemos con 100, 500 o 1000 números, la ganancia en comodidad sí que está clara.

Ejercicios propuestos:

- **(4.1.3.1)** Un programa que almacene en una tabla el número de días que tiene cada mes (de un año no bisiesto), pida al usuario que le indique un mes (ej. 2 para febrero) y un día (ej. el día 15) y diga qué número de día es dentro del año (por ejemplo, el 15 de febrero sería el día número 46, el 31 de diciembre sería el día 365).

4.1.4. Datos repetitivos introducidos por el usuario

Si queremos que sea el usuario el que introduzca datos a un array, usaríamos otra estructura repetitiva ("for", por ejemplo) para pedírselos:

```
/*--------------------------*/
/*  Ejemplo en C# nº 36:    */
/*  ejemplo36.cs            */
/*                          */
/*  Cuarto ejemplo de       */
/*  tablas                  */
/*                          */
/*  Introduccion a C#,      */
/*    Nacho Cabanes         */
/*--------------------------*/

using System;

public class Ejemplo36
{
  public static void Main()
  {

    int[] numero = new int[5]; /* Un array de 5 números enteros */
    int suma;                  /* Un entero que será la suma */
    int i;                     /* Para recorrer los elementos */

    for (i=0; i<=4; i++)       /* Pedimos los datos */
    {
      Console.Write("Introduce el dato numero {0}: ", i+1);
      numero[i] = Convert.ToInt32(Console.ReadLine());
    }

    suma = 0;             /* Valor inicial de la suma */
    for (i=0; i<=4; i++) /* Y hallamos la suma repetitiva */
      suma += numero[i];

      Console.WriteLine("Su suma es {0}", suma);
  }
}
```

Ejercicios propuestos:

- **(4.1.4.1)** A partir del ejercicio 4.1.3.1, crear otro que pida al usuario que le indique la fecha, detallando el día (1 al 31) y el mes (1=enero, 12=diciembre), como respuesta muestre en pantalla el número de días que quedan hasta final de año.
- **(4.1.4.2)** Crear un programa que pida al usuario 10 números en coma flotante (pista: necesitarás un array de "float") y luego los muestre en orden inverso (del último que se ha introducido al primero que se introdujo).
- **(4.1.4.3)** Un programa que pida al usuario 10 números y luego calcule y muestre cuál es el mayor de todos ellos.
- **(4.1.4.4)** Un programa que pida al usuario 10 números, calcule su media y luego muestre los que están por encima de la media.
- **(4.1.4.5)** Un programa que pida 10 nombres y los memorice (pista: esta vez se trata de un array de "string"). Después deberá pedir que se teclee un nombre y dirá si se encuentra o no entre los 10 que se han tecleado antes. Volverá a pedir otro nombre y a decir si se encuentra entre ellos, y así sucesivamente hasta que se teclee "fin".
- **(4.1.4.6)** Un programa que prepare espacio para un máximo de 100 nombres. El usuario deberá ir introduciendo un nombre cada vez, hasta que se pulse Intro sin teclear nada, momento en el que dejarán de pedirse más nombres y se mostrará en pantalla la lista de los nombres que se han introducido.
- **(4.1.4.7)** Un programa que reserve espacio para un vector de 3 componentes, pida al usuario valores para dichas componentes (por ejemplo [2, -5, 7]) y muestre su módulo (raíz cuadrada de la suma de sus componentes al cuadrado).
- **(4.1.4.8)** Un programa que reserve espacio para dos vectores de 3 componentes, pida al usuario sus valores y calcule la suma de ambos vectores (su primera componente será x1+y1, la segunda será x2+y2 y así sucesivamente).
- **(4.1.4.9)** Un programa que reserve espacio para dos vectores de 3 componentes, pida al usuario sus valores y calcule su producto escalar (x1·y1+ x2·y2+x3·y3).

4.1.5. Operaciones habituales: buscar, añadir, insertar, borrar

Algunas operaciones con datos pertenecientes a un array son especialmente frecuentes: buscar si existe un cierto dato, añadir un dato al final de los existentes, insertar un dato entre dos que ya hay, borrar uno de los datos almacenados, etc. Por eso, vamos a ver las pautas básicas para realizar estas operaciones, y un fuente de ejemplo.

- Para ver si un dato existe, habrá que recorrer todo el array, comparando el valor almacenado con el dato que se busca. Puede interesarnos simplemente saber si está o no (con lo que se podría interrumpir la búsqueda en cuanto aparezca una primera vez) o ver en qué posiciones se encuentra (para lo que habría que recorrer todo el array). Si el array estuviera ordenado, se podría buscar de una forma más rápida, pero la veremos más adelante.
- Para poder añadir un dato al final de los ya existentes, necesitamos que el array no esté completamente lleno, y llevar un contador de cuántas posiciones hay ocupadas, para poder guardar el dato en la primera posición libre.
- Para insertar un dato en una cierta posición, los que quedén detrás deberán desplazarse "a la derecha" para dejarle hueco. Este movimiento debe empezar desde el final para que cada dato que se mueve no destruya el que estaba a continuación de él.

También habrá que actualizar el contador, para indicar que queda una posición libre menos.

- Si se quiere borrar el dato que hay en una cierta posición, los que estaban a continuación deberán desplazarse "a la izquierda" para que no haya huecos. Como en el caso anterior, habrá que actualizar el contador, pero ahora para indicar que queda una posición libre meás.

Vamos a verlo con un ejemplo:

```
/*---------------------------*/
/*  Ejemplo en C# nº 36b:    */
/*  ejemplo36b.cs            */
/*                           */
/*  Añadir y borrar en un    */
/*  array                    */
/*                           */
/*  Introduccion a C#,       */
/*    Nacho Cabanes          */
/*---------------------------*/

using System;

public class Ejemplo36b
{
  public static void Main()
  {

    int[] dato = {10, 15, 12, 0, 0};

    int capacidad = 5;          // Capacidad maxima del array
    int cantidad = 3;           // Número real de datos guardados

    int i;                      // Para recorrer los elementos

    // Mostramos el array
    for (i=0; i<cantidad; i++)
      Console.Write("{0} ",dato[i]);
    Console.WriteLine();

    // Buscamos el dato "15"
    for (i=0; i<cantidad; i++)
      if (dato[i] == 15)
        Console.WriteLine("15 encontrado en la posición {0} ", i+1);

    // Añadimos un dato al final
    Console.WriteLine("Añadiendo 6 al final");
    if (cantidad < capacidad)
    {
      dato[cantidad] = 6;
      cantidad++;
    }

    // Y volvemos a mostrar el array
    for (i=0; i<cantidad; i++)
      Console.Write("{0} ",dato[i]);
    Console.WriteLine();
```

```
    // Borramos el segundo dato
    Console.WriteLine("Borrando el segundo dato");
    int posicionBorrar = 1;
    for (i=posicionBorrar; i<cantidad-1; i++)
      dato[i] = dato[i+1];
    cantidad--;

    // Y volvemos a mostrar el array
    for (i=0; i<cantidad; i++)
        Console.Write("{0} ",dato[i]);
    Console.WriteLine();

    // Insertamos 30 en la tercera posición
    if (cantidad < capacidad)
    {
      Console.WriteLine("Insertando 30 en la posición 3");
      int posicionInsertar = 2;
      for (i=cantidad; i>posicionInsertar; i--)
        dato[i] = dato[i-1];
      dato[posicionInsertar] = 30;
      cantidad++;
    }

    // Y volvemos a mostrar el array
    for (i=0; i<cantidad; i++)
      Console.Write("{0} ",dato[i]);
    Console.WriteLine();

  }
}
```

que tendría como resultado:

```
10 15 12
15 encontrado en la posición 2
Añadiendo 6 al final
10 15 12 6
Borrando el segundo dato
10 12 6
Insertando 30 en la posición 3
10 12 30 6
```

Este programa "no dice nada" cuando no se encuentra el dato que se está buscando. Se puede mejorar usando una variable "booleana" que nos sirva de testigo, de forma que al final nos avise si el dato no existía (no sirve emplear un "else", porque en cada pasada del bucle "for" no sabemos si el dato no existe, sólo que no está en la posición actual).

Ejercicios propuestos:

- **(4.1.5.1)** Amplía el ejemplo anterior (36b) para que avise si el dato buscado no aparece.
- **(4.1.5.2)** Un programa que prepare espacio para un máximo de 10 nombres. Deberá mostrar al usuario un menú que le permita realizar las siguientes operaciones:
    - Añadir un dato al final de los ya existentes.
    - Insertar un dato en una cierta posición (como ya se ha comentado, los que quedén detrás deberán desplazarse "a la derecha" para dejarle hueco; por

ejemplo, si el array contiene "hola", "adios" y se pide insertar "bien" en la segunda posición, el array pasará a contener "hola", "bien", "adios".

- Borrar el dato que hay en una cierta posición (como se ha visto, lo que estaban detrás deberán desplazarse "a la izquierda" para que no haya huecos; por ejemplo, si el array contiene "hola", "bien", "adios" y se pide borrar el dato de la segunda posición, el array pasará a contener "hola", "adios"
- Mostrar los datos que contiene el array.
- Salir del programa.

4.2. Tablas bidimensionales

Podemos declarar tablas de **dos o más dimensiones.** Por ejemplo, si queremos guardar datos de dos grupos de alumnos, cada uno de los cuales tiene 20 alumnos, tenemos dos opciones:

- Podemos usar `int datosAlumnos[40]` y entonces debemos recordar que los 20 primeros datos corresponden realmente a un grupo de alumnos y los 20 siguientes a otro grupo. Es "demasiado artesanal", así que no daremos más detalles.
- O bien podemos emplear `int datosAlumnos[2,20]` y entonces sabemos que los datos de la forma `datosAlumnos[0,i]` son los del primer grupo, y los `datosAlumnos[1,i]` son los del segundo.
- Una alternativa, que puede sonar más familiar a quien ya haya programado en C es emplear `int datosAlumnos[2][20]` pero en C# esto no tiene exactamente el mismo significado que [2,20], sino que se trata de dos arrays, cuyos elementos a su vez son arrays de 20 elementos. De hecho, podrían ser incluso dos arrays de distinto tamaño, como veremos en el segundo ejemplo.

En cualquier caso, si queremos indicar valores iniciales, lo haremos entre llaves, igual que si fuera una tabla de una única dimensión.

Vamos a ver un primer ejemplo del uso con arrays de la forma [2,20], lo que podríamos llamar el "estilo Pascal", en el usemos tanto arrays con valores prefijados, como arrays para los que reservemos espacio con "new" y a los que demos valores más tarde:

```
/*--------------------------*/
/*  Ejemplo en C# nº 37:    */
/*  ejemplo37.cs            */
/*                          */
/*  Array de dos dimensiones */
/*  al estilo Pascal        */
/*                          */
/*  Introduccion a C#,      */
/*    Nacho Cabanes         */
/*--------------------------*/

using System;

public class Ejemplo37
{
  public static void Main()
```

```
  {

    int[,] notas1 = new int[2,2]; // 2 bloques de 2 datos
    notas1[0,0] = 1;
    notas1[0,1] = 2;
    notas1[1,0] = 3;
    notas1[1,1] = 4;

   int[,] notas2 = // 2 bloques de 10 datos
    {
      {1, 2, 3, 4, 5, 6, 7, 8, 9, 10},
      {11, 12, 13, 14, 15, 16, 17, 18, 19, 20}
    };

    Console.WriteLine("La nota1 del segundo alumno del grupo 1 es {0}",
      notas1[0,1]);
    Console.WriteLine("La nota2 del tercer alumno del grupo 1 es {0}",
      notas2[0,2]);
  }
}
```

Este tipo de tablas de varias dimensiones son las que se usan también para guardar matrices, cuando se trata de resolver problemas matemáticos más complejos que los que hemos visto hasta ahora.

La otra forma de tener arrays multidimensionales son los "arrays de arrays", que, como ya hemos comentado, y como veremos en este ejemplo, pueden tener elementos de distinto tamaño. En ese caso nos puede interesar saber su longitud, para lo que podemos usar "a.Length":

```
/*--------------------------*/
/*  Ejemplo en C# nº 38:    */
/*  ejemplo38.cs            */
/*                          */
/*  Array de dos dimensiones */
/*  al estilo C... o casi   */
/*                          */
/*  Introduccion a C#,      */
/*    Nacho Cabanes         */
/*--------------------------*/

using System;

public class Ejemplo38
{
  public static void Main()
  {

    int[][] notas;         // Array de dos dimensiones
    notas = new int[3][];  // Seran 3 bloques de datos
    notas[0] = new int[10];  // 10 notas en un grupo
    notas[1] = new int[15];  // 15 notas en otro grupo
    notas[2] = new int[12];  // 12 notas en el ultimo

    // Damos valores de ejemplo
    for (int i=0;i<notas.Length;i++)
```

```
    {
      for (int j=0;j<notas[i].Length;j++)
      {
        notas[i][j] = i + j;
      }
    }

    // Y mostramos esos valores
    for (int i=0;i<notas.Length;i++)
    {
      for (int j=0;j<notas[i].Length;j++)
      {
        Console.Write(" {0}", notas[i][j]);
      }
      Console.WriteLine();
    }

  }
}
```

La salida de este programa sería

0 1 2 3 4 5 6 7 8 9
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
2 3 4 5 6 7 8 9 10 11 12 13

Ejercicios propuestos:

- **(4.2.1)** Un programa que pida al usuario dos bloques de 10 números enteros (usando un array de dos dimensiones). Después deberá mostrar el mayor dato que se ha introducido en cada uno de ellos.
- **(4.2.2)** Un programa que pida al usuario dos bloques de 6 cadenas de texto. Después pedirá al usuario una nueva cadena y comprobará si aparece en alguno de los dos bloques de información anteriores.
- Si has estudiado álgebra de matrices:
  - **(4.2.3)** Un programa que calcule el determinante de una matriz de 2x2.
  - **(4.2.4)** Un programa que calcule el determinante de una matriz de 3x3.
  - **(4.2.5)** Un programa que calcule si las filas de una matriz son linealmente dependientes.
  - **(4.2.6)** Un programa que use matrices para resolver un sistema de ecuaciones lineales mediante el método de Gauss.

4.3. Estructuras o registros

4.3.1. Definición y acceso a los datos

Un **registro** es una agrupación de datos, llamados **campos**, los cuales no necesariamente son del mismo tipo. Se definen con la palabra "**struct**".

En C# (al contrario que en C), primero deberemos declarar cual va a ser la estructura de nuestro registro, lo que no se puede hacer dentro de "Main". Más adelante, ya dentro de "Main", podremos declarar variables de ese nuevo tipo.

```
  public static void Main()
  {
    tipoPersona persona;

    persona.nombre = "Juan";
    persona.inicial = 'J';
    persona.diaDeNacimiento.dia = 15;
    persona.diaDeNacimiento.mes = 9;
    persona.nota = 7.5f;
    Console.WriteLine("{0} nació en el mes {1}",
      persona.nombre, persona.diaDeNacimiento.mes);
    }
}
```

Ejercicios propuestos:

- **(4.3.3.1)** Ampliar el programa 4.3.2.1, para que el campo "duración" se almacene como minutos y segundos, usando un "struct" anidado que contenga a su vez estos dos campos.

4.4. Cadenas de caracteres

4.4.1. Definición. Lectura desde teclado

Hemos visto cómo leer cadenas de caracteres (Console.ReadLine) y cómo mostrarlas en pantalla (Console.Write), así como la forma de darles un valor(=). También podemos comparar cual es su valor, usando ==, o formar una cadena a partir de otras si las unimos con el símbolo de la suma (+):

Así, un ejemplo que nos pidiese nuestro nombre y nos saludase usando todas estas posibilidades podría ser:

```
/*---------------------------*/
/*  Ejemplo en C# nº 42:     */
/*  ejemplo42.cs             */
/*                           */
/*  Cadenas de texto (1)     */
/*                           */
/*  Introduccion a C#,       */
/*    Nacho Cabanes          */
/*---------------------------*/

using System;

public class Ejemplo42
{

  public static void Main()
  {

    string saludo = "Hola";
    string segundoSaludo;
    string nombre, despedida;
```

```
    segundoSaludo = "Que tal?";
    Console.WriteLine("Dime tu nombre... ");
    nombre = Console.ReadLine();

    Console.WriteLine("{0} {1}", saludo, nombre);
    Console.WriteLine(segundoSaludo);

    if (nombre == "Alberto")
      Console.WriteLine("Dices que eres Alberto?");
    else
      Console.WriteLine("Así que no eres Alberto?");

    despedida = "Adios " + nombre + "!";
    Console.WriteLine(despedida);
  }
}
```

4.4.2. Cómo acceder a las letras que forman una cadena

Podemos leer una de las letras de una cadena, de igual forma que leemos los elementos de cualquier array: si la cadena se llama "texto", el primer elemento será texto[0], el segundo será texto[1] y así sucesivamente.

Eso sí, las cadenas en C# no se pueden modificar letra a letra: no podemos hacer texto[0]='a'. Para eso habrá que usar una construcción auxiliar, que veremos más adelante.

4.4.3. Longitud de la cadena.

Podemos saber cuantas letras forman una cadena con "cadena.Length". Esto permite que podamos recorrer la cadena letra por letra, usando construcciones como "for".

Ejercicios propuestos:

- **(4.4.3.1)** Un programa que te pida tu nombre y lo muestre en pantalla separando cada letra de la siguiente con un espacio. Por ejemplo, si tu nombre es "Juan", debería aparecer en pantalla "J u a n".
- **(4.4.3.2)** Un programa que pida una frase al usuario y la muestra en orden inverso (de la última letra a la primera).
- **(4.4.3.3)** Un programa capaz de sumar dos números enteros muy grandes (por ejemplo, de 30 cifras), que se deberán pedir como cadena de texto y analizar letra a letra.
- **(4.4.3.4)** Un programa capaz de multiplicar dos números enteros muy grandes (por ejemplo, de 30 cifras), que se deberán pedir como cadena de texto y analizar letra a letra.

4.4.4. Extraer una subcadena

Podemos extraer parte del contenido de una cadena con "Substring", que recibe dos parámetros: la posición a partir de la que queremos empezar y la cantidad de caracteres que queremos obtener. El resultado será otra cadena:

```
saludo = frase.Substring(0,4);
```

Podemos omitir el segundo número, y entonces se extraerá desde la posición indicada hasta el final de la cadena.

Ejercicios propuestos:

- **(4.4.4.1)** Un programa que te pida tu nombre y lo muestre en pantalla como un triángulo creciente. Por ejemplo, si tu nombre es "Juan", debería aparecer en pantalla:
  J
  Ju
  Jua
  Juan

4.4.5. Buscar en una cadena

Para ver si una cadena contiene un cierto texto, podemos usar IndexOf ("posición de"), que nos dice en qué posición se encuentra, siendo 0 la primera posición (o devuelve el valor -1, si no aparece):

```
if (nombre.IndexOf("Juan") >= 0) Console.Write("Bienvenido, Juan");
```

Podemos añadir un segundo parámetro opcional, que es la posición a partir de la que queremos buscar:

```
if (nombre.IndexOf("Juan", 5) >= 0) ...
```

La búsqueda termina al final de la cadena, salvo que indiquemos que termine antes con un tercer parámetro opcional:

```
if (nombre.IndexOf("Juan", 5, 15) >= 0) ...
```

De forma similar, LastIndexOf ("última posición de") indica la última aparición (es decir, busca de derecha a izquierda).

Si solamente queremos ver si aparece, pero no nos importa en qué posición está, nos bastará con usar "Contains":

```
if (nombre.Contains("Juan")) ...
```

Ejercicios propuestos:

- **(4.4.5.1)** Un programa que pida al usuario 10 frases, las guarde en un array, y luego le pregunte textos de forma repetitiva, e indique si cada uno de esos textos aparece como parte de alguno de los elementos del array. Terminará cuando el texto introducido sea "fin".
- **(4.4.5.2)** Crea una versión del ejercicio 4.4.5.1 en el que, en caso de que alguno de los textos aparezca como subcadena, se avise además si se encuentra exactamente al principio.