Práctica: Creación y uso de tablas

Objetivos

Al final de esta práctica, usted será capaz de:

- Crear y utilizar tablas de tipos valor.
- Pasar argumentos a **Main**.
- Crear y utilizar tablas de tamaño calculado.
- Utilizar tablas de distintos rangos.

Requisitos previos

Antes de realizar la práctica debe estar familiarizado con los siguientes temas:

- Uso de instrucciones de programación en C#.
- Creación y uso de métodos en C#.

Ejercicio 1
Uso de una tabla de tipos valor

En este ejercicio escribirá un programa que espera el nombre de un archivo de texto como argumento para **Main** y resume sus contenidos. El programa lee los contenidos del archivo de texto en una tabla de caracteres. A continuación recorre toda la tabla, contando el número de vocales y consonantes. Finalmente, el programa imprime en la consola el número total de caracteres, vocales, consonantes y nuevas líneas.

Cómo capturar el nombre del archivo como parámetro del método Main

1. Abra el proyecto FileDetails.sln. Este proyecto está en la carpeta Starter\FileDetails dentro del fichero lab06.zip.
2. Añada una tabla de cadenas de caracteres llamada **args** como parámetro del método **Main** de la clase **FileDetails**. Esta tabla contendrá todos los argumentos de línea de comandos cuando se ejecute el programa. Así es como el runtime pasa esos argumentos a **Main**. En este ejercicio, el argumento de línea de comandos que se pase a **Main** será el nombre de un archivo de texto.
3. Añada a **Main** una instrucción que escriba la longitud de **args** en la consola. Esta instrucción comprobará que la longitud de **args** es cero si el runtime no pasa ningún argumento de línea de comandos a **Main**.
4. Añada a **Main** una instrucción **foreach** que escriba en la consola cada cadena de caracteres de **args**. Esta instrucción comprobará que **Main** recibe del runtime los argumentos de línea de comandos.

   El código completo debería ser como éste:

```
static void Main(string[ ] args)
{
    Console.WriteLine(args.Length);
    foreach (string arg in args) {
        Console.WriteLine(arg);
    }
}
```

5. Compile el programa FileDetails.cs y corrija los posibles errores. Ejecute el programa desde la línea de comandos sin utilizar ningún argumento. Compruebe que la longitud de **args** es cero.

> Consejo Para ejecutar el programa desde la línea de comandos, abra la ventana Command y vaya a la carpeta bin\Debug dentro de la carpeta el proyecto, donde estará el archivo ejecutable.

6. Ejecute el programa desde la línea de comandos con el nombre del archivo Solution\FileDetails\FileDetails.cs dentro del fichero lab06.zip. Compruebe que el runtime pasa el nombre del archivo a **Main**.
7. Pruebe el programa con distintos argumentos de línea de comandos y compruebe que todos los argumentos se escriben en la consola. Marque como comentarios las instrucciones que escriben en la consola.
8. Añada a **Main** una instrucción que declare una variable **string** llamada (*fileName) nombreArchivo* e inicialícela con `args[0]`.

Cómo leer del archivo de texto a una tabla

9. Quite las marcas de comentarios del código de declaración e inicialización de FileStream y StreamReader.
10. Determine la longitud del archivo de texto.

Consejo Para encontrar una propiedad adecuada de la clase **Stream**, busque "clase Stream" en los documentos de ayuda del SDK de Microsoft .NET Framework.

---

11. Añada a **Main** una instrucción que declare una variable de cadena de caracteres llamada *contenidos (contents)*. Inicialice *contenidos* con una nueva tabla cuya longitud sea igual a la longitud del archivo de texto, que acaba de calcular.
12. Añada a **Main** una instrucción **for**. El cuerpo de la instrucción **for** leerá un solo carácter de *reader* y lo añadirá a *contenidos*.

Consejo Use el método **Read**, que no recibe ningún parámetro y devuelve un **int**. Convierta el resultado en **char** antes de almacenarlo en la tabla.

---

13. Añada a **Main** una instrucción **foreach** que escriba carácter por carácter toda la cadena de caracteres en la consola. Esta instrucción comprobará que el archivo de texto se ha leído correctamente en la tabla *contenidos*.

    El código completo debería ser como éste:

```
static void Main(string[ ] args)
{
    string fileName = args[0];
    FileStream stream = new FileStream(fileName,
   ➥                                FileMode.Open);
    StreamReader reader = new StreamReader(stream);
    int size = (int)stream.Length;
    char[ ] contents = new char[size];
    for (int i = 0; i < size; i++) {
        contents [i] = (char)reader.Read( );
    }
    foreach(char ch in contents) {
        Console.Write(ch);

    }
}
```

14. Compile el programa y corrija los posibles errores. Ejecute el programa con el nombre del archivo Solution\FileDetails\FileDetails.cs dentro del fichero lab06.zip como argumento de línea de comandos. Compruebe que los contenidos del archivo se escriben correctamente en la consola.
15. Marque como comentario la instrucción **foreach**.
16. Cierre el objeto **Reader** con una llamada al método **StreamReader** correspondiente.

Cómo clasificar y resumir los contenidos del archivo

17. Declare un nuevo método static llamado **Summarize** en la clase **FileDetails**. Este método no devolverá nada y esperará un parámetro de cadena de caracteres. Añada a **Main** una instrucción que haga una llamada al método **Summarize**, pasando *contenidos* como argumento.
18. Añada a **Summarize** una instrucción **foreach** que inspeccione cada carácter en el argumento de tabla. Cuente el número de caracteres de vocal, consonante y nueva línea, y almacene los resultados en variables distintas.

Consejo Para determinar si un carácter es una vocal, cree una cadena de caracteres con todas las vocales posibles y use sobre ella el método **IndexOf** para ver si el carácter existe en esa cadena de caracteres:

```
if ("AEIOUaeiou".IndexOf(myCharacter) != -1) {
   // myCharacter es una vocal
} else {
   // myCharacter no es una vocal
}
```

---

19. Escriba en la consola cuatro líneas que indiquen:
    - El número total de caracteres en el archivo.
    - El número total de vocales en el archivo.
    - El número total de consonantes en el archivo.
    - El número total de líneas en el archivo.

    El código completo debería ser como éste:

```
   static void Summarize (char[ ] contents)
   {
        int vocales = 0, consonantes = 0, lines = 0;
        foreach (char actual in contents) {
            if (Char.IsLetter(current)) {
                if ("AEIOUaeiou".IndexOf(current) != -1) {
                    vowels ++;
                } else {
                    consonants++;
                }
            }
            else if (actual == '\n') {
                lines++;
            }
        }
        Console.WriteLine("Núm total de caracteres : {0}",
➥contenidos.Length);
        Console.WriteLine("Núm total de vocales    : {0}",
➥vocales);
        Console.WriteLine("Núm total de consonantes: {0}",
➥consonantes);
        Console.WriteLine("Núm total de líneas     : {0}",
➥lineas);
   }
```

20. Compile el programa y corrija los posibles errores. Ejecute el programa desde la línea de comandos para resumir los contenidos del archivo Solution\FileDetails\FileDetails.cs dentro del fichero lab06.zip. El resultado correcto es el siguiente:
    - 1.353 caracteres
    - 247 vocales
    - 394 consonantes
    - 41 líneas

Ejercicio 2
Multiplicación de matrices

En este ejercicio escribirá un programa que usa tablas para multiplicar matrices. El programa leerá cuatro valores enteros de la consola y los pondrá en una matriz entera 2 x 2. A continuación leerá otros cuatro valores enteros de la consola y los pondrá en una segunda matriz entera 2 x 2. Luego multiplicará las dos matrices, almacenando el resultado en una tercera matriz entera 2 x 2. Finalmente, imprimirá la matriz resultante en la consola.

La fórmula para multiplicar dos matrices A y B es la siguiente:

A1	A2	×	B1	B2	=	A1.B1 + A2.B3	A1.B2 + A2.B4
A3	A4		B3	B4		A3.B1 + A4.B3	A3.B2 + A4.B4

Cómo multiplicar dos matrices

Abra el proyecto MatrixMultiply.sln en la carpeta Starter\MatrixMultiply dentro del fichero lab06.zip.

En la clase **MatrixMultiply**, añada a **Main** una instrucción que declare una tabla 2 x 2 de **ints** y la llame **a**. La solución final para el programa leerá los valores de **a** desde la consola. Por el momento, inicialice **a** con los valores enteros de la siguiente tabla (esto permitirá comprobar que la multiplicación se realiza correctamente y que los cambios subsiguientes no hacen variar el comportamiento del programa):

$$\begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}$$

3. Añada a **Main** una instrucción que declare una tabla 2 x 2 de **ints** y la llame **b**. La solución final para el programa leerá los valores de **b** desde la consola. Por el momento, inicialice **b** con los valores enteros de la siguiente tabla:

$$\begin{pmatrix} 5 & 6 \\ 7 & 8 \end{pmatrix}$$

o Añada a **Main** una instrucción que declare una tabla 2 x 2 de **ints** y la llame **result**. Inicialice **result** utilizando las siguientes fórmulas:

a[0,0] * b[0,0] + a[0,1] * b[1,0]	a[0,0] * b[0,1] + a[0,1] * b[1,1]
a[1,0] * b[0,0] + a[1,1] * b[1,0]	a[1,0] * b[0,1] + a[1,1] * b[1,1]

o Añada a **Main** cuatro instrucciones que escriban en la consola los cuatro valores **int** de **result**. Estas instrucciones le ayudarán a comprobar que ha copiado las fórmulas correctamente.

El código completo debería ser como éste:

```
static void Main( )
{
      int[,] a = new int[2,2];
      a[0,0] = 1; a[0,1] = 2;
      a[1,0] = 3; a[1,1] = 4;

      int[,] b = new int[2,2];
      b[0,0] = 5; b[0,1] = 6;
      b[1,0] = 7; b[1,1] = 8;

      int[,] result = new int[2,2];
      result [0,0]=a[0,0]*b[0,0] + a[0,1]*b[1,0];
      result [0,1]=a[0,0]*b[0,1] + a[0,1]*b[1,1];
      result [1,0]=a[1,0]*b[0,0] + a[1,1]*b[1,0];
      result [1,1]=a[1,0]*b[0,1] + a[1,1]*b[1,1];

      Console.WriteLine(result [0,0]);
      Console.WriteLine(result [0,1]);
      Console.WriteLine(result [1,0]);
      Console.WriteLine(result [1,1]);
}
```

6. Compile el programa y corrija los posibles errores. Ejecute el programa y compruebe que los cuatro valores de **result** son los siguientes:

$$\begin{pmatrix} 19 & 22 \\ 43 & 50 \end{pmatrix}$$

Cómo mostrar el resultado usando un método con un parámetro de tabla

1. Declare un nuevo método static llamado **Output (Salida)** en la clase **MatrixMultiply**. Este método no devolverá nada y esperará como parámetro una tabla **int** de rango 2 llamada **result**.
2. Corte de **Main** las cuatro instrucciones que escriben en la consola los cuatro valores de **result** y péguelas en **Output**.
3. Añada a **Main** una instrucción que llame al método **Output**, pasando **result** como argumento (este código debe ir en el lugar que ocupaban las cuatro instrucciones cortadas en el paso anterior).

   El código completo debería ser como éste:

```
static void Output(int[,] result)
{
      Console.WriteLine(result [0,0]);
      Console.WriteLine(result [0,1]);
      Console.WriteLine(result [1,0]);
      Console.WriteLine(result [1,1]);
}
```

4. Compile el programa y corrija los posibles errores. Ejecute el programa y compruebe que los cuatro valores escritos en la consola siguen siendo:

$$\begin{pmatrix} 19 & 22 \\ 43 & 50 \end{pmatrix}$$

5. Cambie el método **Salida** para que utilice dos instrucciones **for** anidadas en lugar de cuatro instrucciones **WriteLine**. Use el valor literal 2 en las comprobaciones de límites de las tablas.

El código completo debería ser como éste:

```
static void Output(int[,] result)
{
    for (int r = 0; r < 2; r++) {
        for (int c = 0; c < 2; c++) {
            Console.Write("{0} ", result[r,c]);
        }
        Console.WriteLine( );
    }
}
```

6. Compile el programa y corrija los posibles errores. Ejecute el programa y compruebe que los cuatro valores escritos en la consola siguen siendo:

$$\begin{pmatrix} 19 & 22 \\ 43 & 50 \end{pmatrix}$$

7. Vuelva a modificar el método **Output** para hacerlo más genérico. Sustituya el valor literal 2 en las comprobaciones de límites de las tablas por llamadas al método **GetLength**.

El código completo debería ser como éste:

```
static void Output(int[,] result)
{
    for (int r = 0; r < result.GetLength(0); r++) {
        for (int c = 0; c < result.GetLength(1); c++) {
            Console.Write("{0} ", result [r,c]);
        }
        Console.WriteLine( );
    }
}
```

8. Compile el programa y corrija los posibles errores. Ejecute el programa y compruebe que los cuatro valores escritos en la consola siguen siendo:

$$\begin{pmatrix} 19 & 22 \\ 43 & 50 \end{pmatrix}$$

Cómo calcular el resultado en un método y devolverlo

21. Declare un nuevo método static llamado **Multiply** en la clase **MatrixMultiply**. Este método devolverá una tabla **int** de rango 2 y esperará como parámetros dos tablas **int** de rango 2 llamadas **a** y **b**.
22. Copie (sin cortar) de **Main** a **Multiply** la declaración e inicialización de **result**.
23. Añada a **Multiply** una instrucción **return** que devuelva **result**
24. Sustituya la inicialización de **result** en **Main** por una llamada a **Multiply**, pasando **a** y **b** como argumentos.

    El código completo debería ser como éste:

```
static int[,] Multiply(int[,] a, int [,] b)
{
    int[,] result = new int[2,2];
    result [0,0]=a[0,0]*b[0,0] + a[0,1]*b[1,0];
    result [0,1]=a[0,0]*b[0,1] + a[0,1]*b[1,1];
    result [1,0]=a[1,0]*b[0,0] + a[1,1]*b[1,0];
    result [1,1]=a[1,0]*b[0,1] + a[1,1]*b[1,1];
    return result;
}
```

5. Compile el programa y corrija los posibles errores. Ejecute el programa y compruebe que los cuatro valores escritos en la consola siguen siendo:

$$\begin{pmatrix} 19 & 22 \\ 43 & 50 \end{pmatrix}$$

Cómo calcular el resultado en un método usando instrucciones for

1. Sustituya la inicialización de **result** en **Multiply** por una tabla 2 x 2 de **ints** de nueva creación.
2. Añada a **Multiply** dos instrucciones **for** anidadas. Utilice un entero llamado *r* en la instrucción **for** más externa para recorrer todos los índices de la primera dimensión de **result**, y un entero llamado *c* en la instrucción **for** más interna para recorrer todos los índices de la segunda dimensión de **result**. Use el valor literal 2 para los límites de las tablas. El cuerpo de la instrucción **for** más interna tendrá que calcular el valor de `result [r,c]` con la siguiente fórmula:

```
result [r,c] = a[r,0] * b[0,c]
               + a[r,1] * b[1,c]
```

El código completo debería ser como éste:

```
static int[,] Multiply int[,] a, int [,] b)
{
    int[,] result = new int[2,2];
    for (int r = 0; r < 2; r++) {
       for (int c = 0; c < 2; c++) {
          result[r,c] += a[r,0] * b[0,c] + a[r,1] * b[1,c];
       }
    }
    return result;
}
```

3. Compile el programa y corrija los posibles errores. Ejecute el programa y compruebe que los cuatro valores escritos en la consola siguen siendo:

$$\begin{pmatrix} 19 & 22 \\ 43 & 50 \end{pmatrix}$$

Cómo introducir la primera matriz desde la consola

1. Sustituya la inicialización de **a** en **Main** por una nueva tabla 2 x 2 de **ints**.
2. Añada a **Main** instrucciones que pidan al usuario cuatro valores y los lean en **a** desde la consola. Tienen que ir antes de la llamada al método **Multiply**. Las instrucciones para leer *un* valor desde la consola son:

```
string s = Console.ReadLine( );
a[0,0] = int.Parse(s);
```

3. Compile el programa y corrija los posibles errores. Ejecute el programa, escribiendo en la consola los mismos cuatro valores para **a** (es decir, 1, 2, 3 y 4). Compruebe que los cuatro valores escritos en la consola siguen siendo:

$$\begin{pmatrix} 19 & 22 \\ 43 & 50 \end{pmatrix}$$

4. Declare un nuevo método static llamado **Input (Entrada)** en la clase **MatrixMultiply**. Este método no devolverá nada y esperará como parámetro una tabla **int** de rango 2 llamada **dst**.

5. Corte de **Main** las instrucciones que leen los cuatro valores en **a** y péguelas en **Input**. Añada a **Main** (antes de la llamada a **Multiply**) una instrucción que llame al método **Input**, pasando **a** como parámetro.
6. Compile el programa y corrija los posibles errores. Ejecute el programa, escribiendo en la consola los mismos cuatro valores para **a** (es decir, 1, 2, 3 y 4). Compruebe que los cuatro valores escritos en la consola siguen siendo:

$$\begin{pmatrix} 19 & 22 \\ 43 & 50 \end{pmatrix}$$

7. Cambie el método **Input** para que utilice dos instrucciones **for** anidadas. Use el valor literal 2 para los límites de las tablas. Incluya en el método **Input** una instrucción **Write** que pida los valores al usuario.

   El código completo debería ser como éste:

```
static void Input (int[,] dst)
{
    for (int r = 0; r < 2; r++) {
        for (int c = 0; c < 2; c++) {
            Console.Write(
➥"Escriba el valor de [{0},{1}] : ", r, c);
            string s = Console.ReadLine( );
            dst[r,c] = int.Parse(s);
        }
    }
    Console.WriteLine( );
}
```

8. Compile el programa y corrija los posibles errores. Ejecute el programa, escribiendo en la consola los mismos cuatro valores para **a** (es decir, 1, 2, 3 y 4). Compruebe que los cuatro valores escritos en la consola siguen siendo:

$$\begin{pmatrix} 19 & 22 \\ 43 & 50 \end{pmatrix}$$

Cómo introducir la segunda matriz desde la consola

1. Sustituya la inicialización de **b** en **Main** por una tabla 2 x 2 de **ints** de nueva creación cuyos cuatro valores tengan por defecto el valor cero.
2. Añada a **Main** una instrucción que lea valores en **b** desde la consola llamando al método **Input** y pasando **b** como argumento.
3. Compile el programa y corrija los posibles errores. Ejecute el programa, escribiendo los mismos cuatro valores para **a** (1, 2, 3 y 4) y los mismos cuatro valores para **b** (5, 6, 7 y 8). Compruebe que los cuatro valores escritos en la consola siguen siendo:

$$\begin{pmatrix} 19 & 22 \\ 43 & 50 \end{pmatrix}$$

4. Ejecute el programa con datos diferentes. Colabore con otro estudiante para ver si obtienen la misma respuesta para los mismos datos.