

EJERCICIOS MÓDULO 5

Objetivos

Al final de esta práctica, usted será capaz de:

- Crear y hacer llamadas a métodos con y sin parámetros.
- Utilizar distintos mecanismos para pasar parámetros.
- Crear variables referencia y pasarlas como parámetros de métodos.

Requisitos previos

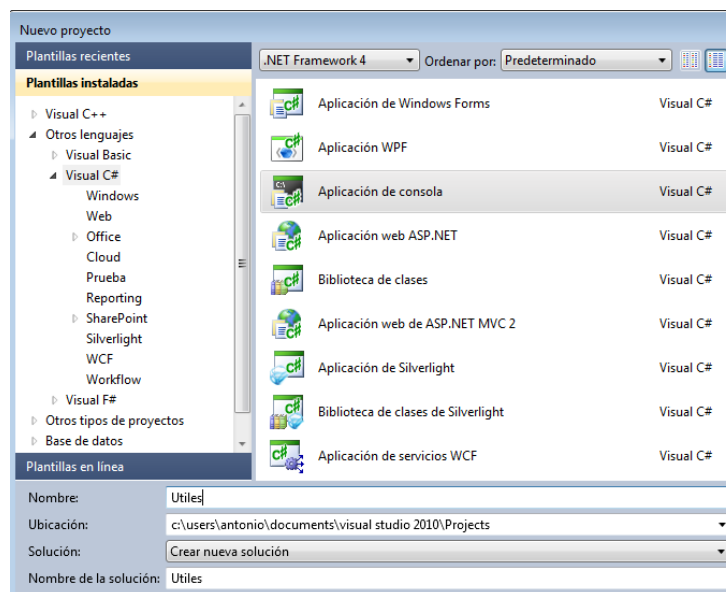
Antes de realizar la práctica debe estar familiarizado con los siguientes temas:

- Creación y uso de variables
- Instrucciones de C#

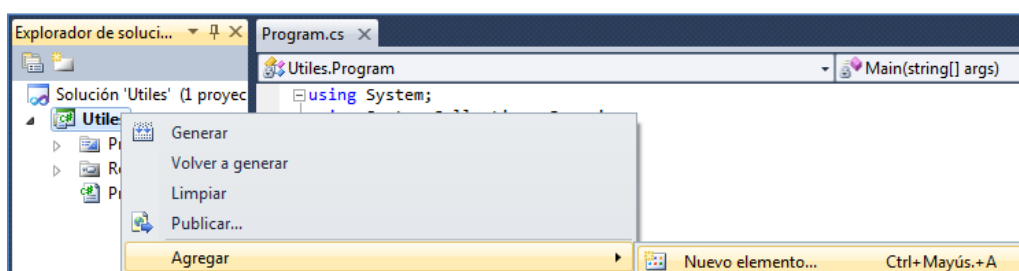
EJERCICIO 1

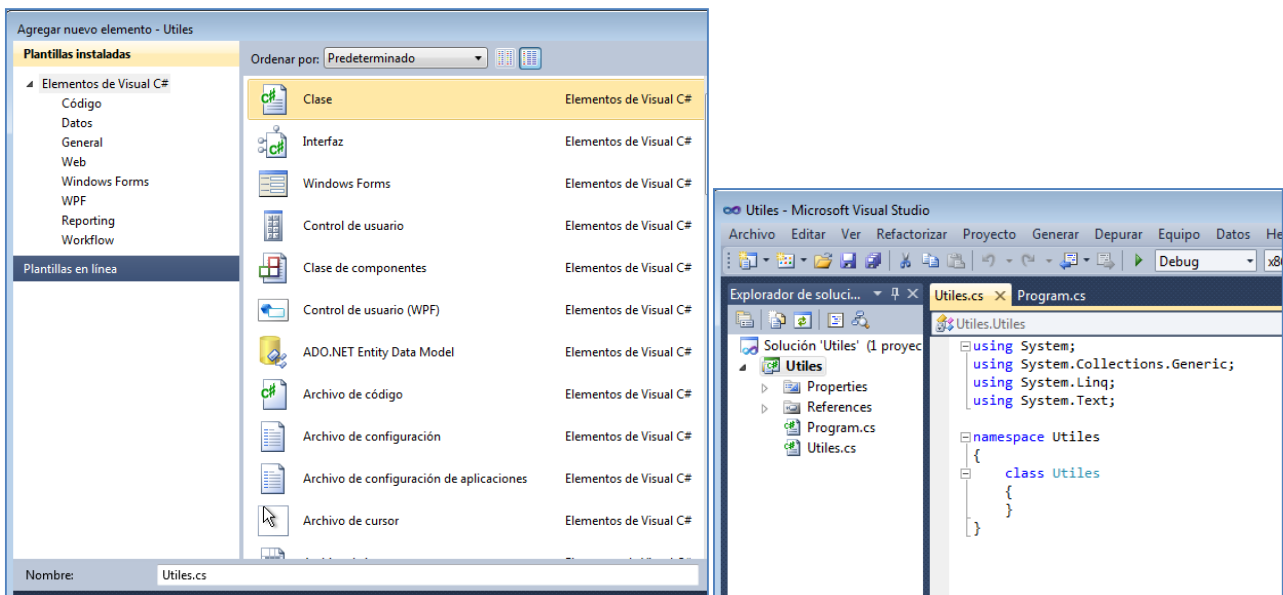
En este ejercicio definirá y usará parámetros de entrada en un método que devuelve un valor. Creará una clase de nombre `Utiles`. En esa clase creará un método llamado `Mayor` que aceptará como entrada dos parámetros enteros y devolverá el valor del que sea mayor. Para probar la clase creará otra clase llamada `Test` que pedirá dos números al usuario, llamará a `Utiles.Mayor` para determinar cuál de los dos es mayor e imprimirá el resultado.

Paso 1. Crea un nuevo Proyecto en Visual C# de tipo Aplicación de Consola, de nombre `Utiles`.



Paso 2. Dentro del espacio de nombres llamado `Utiles`, cree una clase C# con el mismo nombre `Utiles.cs`.





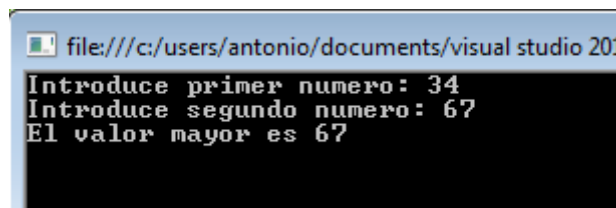
Paso 3. Cree el método public static Mayor en la clase Utiles, de manera que acepte dos parámetros int llamados a y b que se pasarán por valor, y devolverá un valor int que represente el mayor de los dos números.

```
public static int Mayor(int a, int b)
{
    if (a > b)
        return a;
    else
        return b;
    // return (a > b) ? (a) : (b);
}
```

Paso 4. Para probar el método Mayor, abra el fichero Program.cs y en el método Main defina dos variables enteras llamadas x e y. Añada instrucciones que lean dos enteros desde la entrada por teclado y los asignen a x e y. Defina otro entero llamado mayor y asígnele el valor devuelto por la función Mayor. Escriba el código necesario para mostrar el mayor de los dos enteros.

```
////////MAYOR////////
int x, y , mayor;
Console.Write("Introduce primer numero: ");
x = int.Parse(Console.ReadLine());
Console.Write("Introduce segundo numero: ");
y = int.Parse(Console.ReadLine());
mayor = Utiles.Mayor(x, y);
Console.WriteLine("El valor mayor es " + mayor);
```

Paso 5. Compile el proyecto y corrija los posibles errores. Ejecute y pruebe el programa.



EJERCICIO 2

Uso de métodos con parámetros referencia

En este ejercicio escribirá un método llamado Swap que intercambiará los valores de sus parámetros. Utilizará parámetros que se pasan por referencia.

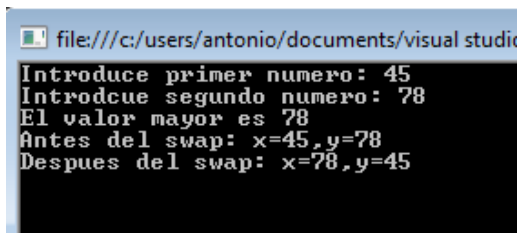
Paso 1. Abra el proyecto anterior Utiles.sln y añada el método public static void Swap a la clase Utiles, de manera que acepte dos parámetros int llamados a y b, que se pasarán por referencia. Escriba instrucciones en el cuerpo de Swap para intercambiar los valores de a y b. Tendrá que crear una variable int local en Swap para guardar temporalmente uno de los valores durante el intercambio. Llame a esta variable temp.

```
public static void Swap(ref int a, ref int b)
{
    int temp = a;
    a = b;
    b = temp;
}
```

Paso 2. Para probar el método Swap, haga una llamada a este método, pasando estos valores como parámetros. Visualice los nuevos valores de los dos enteros antes y después de intercambiarlos.

```
/////////SWAP/////////
Console.WriteLine("Antes del swap: " + "x=" + x + ",y=" + y);
Utiles.Swap(ref x, ref y);
Console.WriteLine("Despues del swap: " + "x=" + x + ",y=" + y);
```

Paso 3. Compile el proyecto y corrija los posibles errores. Ejecute y pruebe el programa.



```
file:///c:/users/antonio/documents/visual studio
Introduce primer numero: 45
Introduce segundo numero: 78
El valor mayor es 78
Antes del swap: x=45,y=78
Despues del swap: x=78,y=45
```

```
namespace Utiles
{
    class Program
    {
        static void Main(string[] args)
        {
            int x, y, mayor;
            Console.Write("Introduce primer numero: ");
            x = int.Parse(Console.ReadLine());
            Console.Write("Introduce segundo numero: ");
            y = int.Parse(Console.ReadLine());
            mayor = Utiles.Mayor(x, y);
            Console.WriteLine("El valor mayor es " + mayor);
            Console.WriteLine("Antes del swap: " + "x=" + x + ",y=" + y);
            Utiles.Swap(ref x, ref y);
            Console.WriteLine("Despues del swap: " + "x=" + x + ",y=" + y);
            Console.ReadLine();
        }
    }
}
```

EJERCICIO 3

Uso de métodos con parámetros de salida

Escribirá un nuevo método llamado Factorial que recibirá un valor int y calculará su factorial. El factorial de un número es el producto de todos los números entre 1 y ese número; el factorial de cero es 1 por definición. A continuación se dan algunos ejemplos de factoriales:

Factorial(0) = 1

Factorial(1) = 1

Factorial(2) = 1 * 2 = 2

Factorial(3) = 1 * 2 * 3 = 6

Factorial(4) = 1 * 2 * 3 * 4 = 24

Paso 1. Abra el proyecto Utiles.sln del ejercicio anterior, si no está ya abierto.

Paso 2. Añada el método public static bool Factorial a la clase Utiles.

- Este método recibirá dos parámetros llamados n y respuesta. El primero se pasa por valor y es el int cuyo factorial se va a calcular. El segundo es un parámetro out int que se utilizará para devolver el resultado.

-El método Factorial debe devolver un valor bool que indique si ha funcionado (podría desbordarse y producir una excepción).

```
public static bool Factorial(int n, out int answer)
```

Paso 3. Añada funcionalidad al método Factorial. Ejecute los siguientes pasos para añadir funcionalidad al método:

- Cree una variable int llamada k en el método Factorial para utilizarla como contador del bucle.
- Cree otra variable int llamada f, que se usará como valor de trabajo dentro del bucle. Inicialice la variable de trabajo f con el valor 1.
- Use un bucle for para realizar la iteración. Comience con un valor de 2 para k, y termine cuando k llegue al valor del parámetro n. Incremente k cada vez que se ejecute el bucle.
- En el cuerpo del bucle, multiplique f sucesivamente por cada valor de k y almacene el resultado en f.
- El resultado de un factorial puede ser muy grande aunque el valor de entrada sea pequeña. Asegúrese de que todos los cálculos de enteros están en un bloque de comprobación (checked) y de que se capturan excepciones como el desbordamiento aritmético.
- Asigne el valor del resultado en f al parámetro de salida respuesta.
- Devuelva true desde el método si el cálculo se realiza sin problemas, y false en caso contrario (es decir, si se produce una excepción).

```

public static bool Factorial(int n, out int answer)
{
    int k;
    int f;
    bool ok = true;

    if (n < 0)
        ok = false;
    try {
        checked {
            f = 1;
            for (k = 2; k <= n; ++k)
            {
                f = f * k;
            }
        }
    } catch (Exception) {
        f = 0;
        ok = false;
    }
    answer = f;
    return ok;
}

```

Paso 4. Para probar el método Factorial en Main:

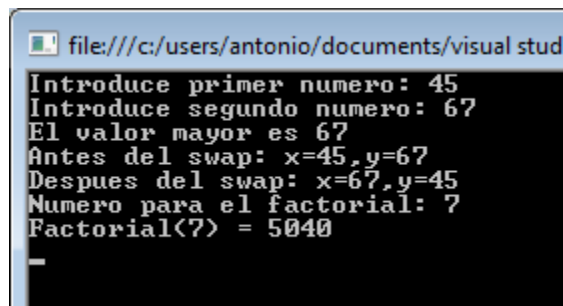
- Declare una variable bool llamada ok para el resultado true o false.
- Declare una variable int llamada f para el resultado del factorial.
- Pida un entero al usuario. Asigne el valor de entrada a la variable int x.
- Haga una llamada al método Factorial, pasando x como primer parámetro y f como segundo parámetro. Devuelva el resultado en ok.
- Si ok es true, muestre los valores de x y f; en caso contrario, muestre un mensaje para indicar que se ha producido un error.

```

////////FACTORIAL////////
int f;
bool ok;
Console.WriteLine("Numero para el factorial: ");
x = int.Parse(Console.ReadLine());
ok = Utilis.Factorial(x, out f);
if (ok) Console.WriteLine("Factorial(" + x + ") = " + f);
else    Console.WriteLine("No se puede calcular este factorial");

```

Paso 5. Compile el proyecto y corrija los posibles errores. Ejecute y pruebe el programa.



```

file:///c:/users/antonio/documents/visual stud
Introduce primer numero: 45
Introduce segundo numero: 67
El valor mayor es 67
Antes del swap: x=45,y=67
Despues del swap: x=67,y=45
Numero para el factorial: 7
Factorial(7) = 5040
-

```

```

namespace Utiles
{
    class Program
    {
        static void Main(string[] args)
        {
            //////////MAYOR//////////
            int x, y , mayor;
            Console.Write("Introduce primer numero: ");
            x = int.Parse(Console.ReadLine());
            Console.Write("Introduce segundo numero: ");
            y = int.Parse(Console.ReadLine());
            mayor = Utiles.Mayor(x, y);
            Console.WriteLine("El valor mayor es " + mayor);
            //////////SWAP//////////
            Console.WriteLine("Antes del swap: " + "x=" + x + ",y=" + y);
            Utiles.Swap(ref x, ref y);
            Console.WriteLine("Despues del swap: " + "x=" + x + ",y=" + y);
            //////////FACTORIAL//////////
            int f;
            bool ok;
            Console.Write("Numero para el factorial: ");
            x = int.Parse(Console.ReadLine());
            ok = Utiles.Factorial(x, out f);
            if (ok)
                Console.WriteLine("Factorial(" + x + ") = " + f);
            else
                Console.WriteLine("No se puede calcular este factorial");
            Console.ReadLine();
        }
    }
}

```

EJERCICIO 4

Método con recursividad

En este ejercicio reescribirá el método Factorial creado en el Ejercicio 3 utilizando recursividad en lugar de un bucle. El factorial de un número se puede definir recursivamente de la siguiente manera: el factorial de cero es 1, y el factorial de cualquier otro entero más grande se puede calcular multiplicando ese entero por el factorial del número anterior. En resumen:

Si $n = 0$, entonces $\text{Factorial}(n) = 1$; en los demás casos, $\text{Factorial}(n) = n * \text{Factorial}(n-1)$

Paso 1. Abra el proyecto Utiles.sln del ejercicio anterior, si no está ya abierto.

Paso 2. Edite la clase Utiles y añada el método Factorial para que emplee recursividad en lugar de iteración.

- Los parámetros y tipos de retorno serán los mismos, pero cambiará el funcionamiento interno del método. Si desea conservar la solución del Ejercicio 3, tendrá que utilizar otro nombre para este método.

- Utilice el pseudocódigo mostrado más arriba para escribir el cuerpo del método Factorial (tendrá que convertirlo a la sintaxis de C#).

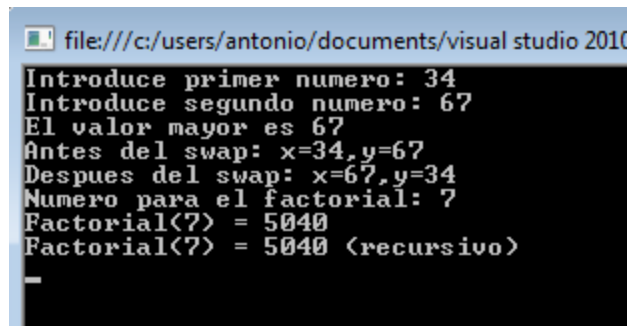
```
public static bool FactorialRecursivo(int n, out int f)
{
    bool ok=true;

    if (n < 0) {
        f = 0;
        ok = false;
    }
    if (n <= 1)
        f = 1;
    else {
        try {
            int pf;
            checked {
                ok = FactorialRecursivo(n-1,out pf);
                f = n * pf;
            }
        } catch(Exception) {
            f = 0;
            ok = false;
        }
    }
    return ok;
}
```

Paso 3. Añada código a la clase Test para probar el nuevo método.

```
//////////FACTORIAL RECURSIVO//////////
ok = Utiles.FactorialRecursivo(x, out f);
if (ok) Console.WriteLine("Factorial(" + x + ") = " + f + " (recursivo)");
else Console.WriteLine("No se puede calcular este factorial (recursivo)");
Console.ReadLine();
```

Paso 4. Compile el proyecto y corrija los posibles errores. Ejecute y pruebe el programa.



```
file:///c:/users/antonio/documents/visual studio 2010
Introduce primer numero: 34
Introduce segundo numero: 67
El valor mayor es 67
Antes del swap: x=34,y=67
Despues del swap: x=67,y=34
Numero para el factorial: 7
Factorial<7> = 5040
Factorial<7> = 5040 (recursivo)
```

```
namespace Utiles
{
    class Program
    {
        static void Main(string[] args)
        {
            //////////MAYOR//////////
            int x, y , mayor;
            Console.Write("Introduce primer numero: ");
            x = int.Parse(Console.ReadLine());
            Console.Write("Introduce segundo numero: ");
            y = int.Parse(Console.ReadLine());
            mayor = Utiles.Mayor(x, y);
            Console.WriteLine("El valor mayor es " + mayor);
            //////////SWAP//////////
            Console.WriteLine("Antes del swap: " + "x=" + x + ",y=" + y);
            Utiles.Swap(ref x, ref y);
            Console.WriteLine("Despues del swap: " + "x=" + x + ",y=" + y);
            //////////FACTORIAL//////////
            int f;
            bool ok;
            Console.Write("Numero para el factorial: ");
            x = int.Parse(Console.ReadLine());
            ok = Utiles.Factorial(x, out f);
            if (ok) Console.WriteLine("Factorial(" + x + ") = " + f);
            else Console.WriteLine("No se puede calcular este factorial");
            //////////FACTORIAL RECURSIVO//////////
            ok = Utiles.FactorialRecursivo(x, out f);
            if (ok) Console.WriteLine("Factorial(" + x + ") = " + f + " (recursivo)");
            else | Console.WriteLine("No se puede calcular este factorial (recursivo)");
            Console.ReadLine();
        }
    }
}
```


EJERCICIO 5

Inclusión de un método de instancia con dos parámetros

En prácticas anteriores se desarrolló una clase CuentaBancaria. En este ejercicio volverá a utilizar esa clase y añadirá un nuevo método de instancia, llamado TransferenciaDe, que transfiere dinero de una cuenta especificada a la actual.

Paso 1. Abra el proyecto CuentaBancaria.sln con la siguiente situación de partida:

```
namespace CuentaBancaria
{
    class Program
    {
        static void Main(string[] args)
        {
            CuentaBancaria cuenta1 = NewCuentaBancaria();
            Write(cuenta1);
            TestIngreso(cuenta1);
            Write(cuenta1);
            TestRetirar(cuenta1);
            Write(cuenta1);

            CuentaBancaria cuenta2 = NewCuentaBancaria();
            Write(cuenta2);
            TestIngreso(cuenta2);
            Write(cuenta2);
            TestRetirar(cuenta2);
            Write(cuenta2);
            Console.Read();
        }
    }
}

static CuentaBancaria NewCuentaBancaria()
{
    CuentaBancaria created = new CuentaBancaria();
    Console.WriteLine("Introduce el saldo de cuenta: ");
    decimal balance = decimal.Parse(Console.ReadLine());

    created.Populate(balance);
    return created;
}

static void Write(CuentaBancaria acc)
{
    Console.WriteLine("Numero de cuenta es {0}", acc.getNumber());
    Console.WriteLine("Saldo de cuenta es {0}", acc.getBalance());
    Console.WriteLine("Tipo de cuenta es {0}", acc.getTipo());
}

static void TestIngreso(CuentaBancaria acc)
{
    Console.WriteLine("Entra la cantidad a depositar: ");
    decimal cantidad = decimal.Parse(Console.ReadLine());
    acc.Ingresar(cantidad);
}

static void TestRetirar(CuentaBancaria acc)
{
    Console.WriteLine("Entre cantidad a retirar: ");
    decimal cantidad = decimal.Parse(Console.ReadLine());
    if (!acc.Retirar(cantidad))
    {
        Console.WriteLine("Fondos insuficientes.");
    }
}
}
```

```
namespace CuentaBancaria
{
    class CuentaBancaria
    {
        private long cuentaNum;
        private decimal cuentaSaldo;
        private TipoCuenta cuentaTipo;
        private static long sigNumCuenta = 123;

        private static long SigNumber() {
            return sigNumCuenta++;
        }

        public void Populate(decimal balance) {
            cuentaNum = SigNumber();
            cuentaSaldo = balance;
            cuentaTipo = TipoCuenta.Corriente;
        }

        public long getNumber() {
            return cuentaNum;
        }

        public decimal getBalance() {
            return cuentaSaldo;
        }

        public string getTipo() {
            return cuentaTipo.ToString();
        }

        public bool Retirar(decimal amount) {
            bool fondoSuficiente = cuentaSaldo >= amount;
            if (fondoSuficiente)
            {
                cuentaSaldo -= amount;
            }
            return fondoSuficiente;
        }

        public decimal Ingresar(decimal cantidad) {
            cuentaSaldo += cantidad;
            return cuentaSaldo;
        }
    }
}
```

Paso 2. Cree un método de instancia público llamado `TransferenciaDe` en la clase `CuentaBancaria`. El primer parámetro es una referencia a otro objeto `CuentaBancaria` llamado `accFrom`, desde donde se va a transferir el dinero. El segundo parámetro es un valor decimal llamado `cantidad`, que se pasa por valor e indica la cantidad que se va a transferir. El método no tiene valor de devolución.

Paso 3. Añada al cuerpo de `TransferenciaDe` dos instrucciones que realicen las siguientes tareas:

- Deducir cantidad del saldo de `accFrom` (usando `Retirar`).
- Comprobar que el dinero se ha podido retirar sin problemas. Si es así, sumar cantidad al saldo de la cuenta actual (usando `Ingresar`).

```
public void TransferenciaDe(CuentaBancaria accFrom, decimal cantidad)
{
    if (accFrom.Retirar(cantidad))
        this.Ingresar(cantidad);
}
```

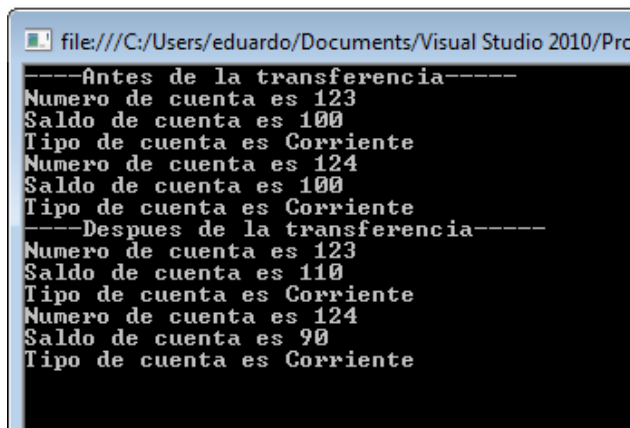
Paso 4. En el método Main, añade código para crear dos objetos CuentaBancaria, ambos con un saldo inicial de 100\$ (use el método Populate).

Paso 5. Añade código para mostrar el tipo, número de cuenta y saldo actual de cada cuenta.

Paso 6. Añade código para llamar a TransferenciaDe y transferir 10€ de una cuenta a otra.

Paso 7. Añade código para mostrar los saldos después de la transferencia.

Paso 8. Compile el proyecto y corrija los posibles errores. Ejecute y pruebe el programa.



```
file:///C:/Users/eduardo/Documents/Visual Studio 2010/Pro...
---Antes de la transferencia-----
Numero de cuenta es 123
Saldo de cuenta es 100
Tipo de cuenta es Corriente
Numero de cuenta es 124
Saldo de cuenta es 100
Tipo de cuenta es Corriente
---Despues de la transferencia-----
Numero de cuenta es 123
Saldo de cuenta es 110
Tipo de cuenta es Corriente
Numero de cuenta es 124
Saldo de cuenta es 90
Tipo de cuenta es Corriente
```

```
static void Main(string[] args)
{
    CuentaBancaria cuenta1 = new CuentaBancaria();
    cuenta1.Populate(100);
    CuentaBancaria cuenta2 = new CuentaBancaria();
    cuenta2.Populate(100);
    Console.WriteLine("----Antes de la transferencia-----");
    Write(cuenta1);
    Write(cuenta2);
    cuenta1.TransferenciaDe(cuenta2, 10);
    Console.WriteLine("----Despues de la transferencia-----");
    Write(cuenta1);
    Write(cuenta2);
    Console.Read();
}
```

EJERCICIO 6

Inversión de una cadena de caracteres

En una práctica anterior se desarrolló una clase Utils que contenía distintos métodos. En este ejercicio añadirá un nuevo método estático llamado Reverse (Invertir) a la clase Utils. Este método recibe una cadena de caracteres y devuelve otra nueva con los caracteres en orden inverso.

Paso 1. Abra el proyecto anterior Utils.sln. Situación de partida:

```
namespace Utils
{
    class Utils
    {
        public static int Mayor(int a, int b)
        {
            if (a > b)
                return a;
            else
                return b;
            // return (a > b) ? (a) : (b);
        }

        public static void Swap(ref int a, ref int b)
        {
            int temp = a;
            a = b;
            b = temp;
        }

        public static bool Factorial(int n, out int answer)
        {
            int k;
            int f;
            bool ok = true;

            if (n < 0)
                ok = false;
            try
            {
                checked
                {
                    f = 1;
                    for (k = 2; k <= n; ++k)
                    {
                        f = f * k;
                    }
                }
            }
            catch (Exception)
            {
                f = 0;
                ok = false;
            }
            answer = f;
            return ok;
        }
    }
}
```

```

namespace Uutils
{
    class Program
    {
        static void Main(string[] args)
        {
            //----MAYOR----//
            int x, y, mayor;
            Console.WriteLine("Introduce primer numero: ");
            x = int.Parse(Console.ReadLine());
            Console.WriteLine("Introduce segundo numero: ");
            y = int.Parse(Console.ReadLine());
            mayor = Uutils.Mayor(x, y);
            Console.WriteLine("El valor mayor es " + mayor);
            //----SWAP-----//
            Console.WriteLine("Antes del swap: " + "x=" + x + ",y=" + y);
            Uutils.Swap(ref x, ref y);
            Console.WriteLine("Despues del swap: " + "x=" + x + ",y=" + y);
            //----FACTORIAL-----//
            int f;
            bool ok;
            Console.WriteLine("Numero para el factorial: ");
            x = int.Parse(Console.ReadLine());
            ok = Uutils.Factorial(x, out f);
            if (ok) Console.WriteLine("Factorial(" + x + ") = " + f);
            else Console.WriteLine("No se puede calcular este factorial");
            //----FACTORIAL RECURSIVO-----//
            ok = Uutils.FactorialRecursivo(x, out f);
            if (ok) Console.WriteLine("Factorial(" + x + ") = " + f + " (recursivo)");
            else Console.WriteLine("No se puede calcular este factorial (recursivo)");
            Console.ReadLine();
        }
    }
}

```

Paso 2. Añada un método public static llamado Reverse a la clase Uutils:

- Tiene un solo parámetro, llamado s, que es una referencia a una string.
- El método tiene un tipo de devolución void.

Paso 3. En el método Reverse:

- Cree una variable string llamada sRev que contendrá la cadena devuelta como resultado. Inicialice esta variable a "".
- Escriba un bucle que extraiga caracteres uno a uno desde s. Comience por el final (use la propiedad Length) y retroceda hasta llegar al principio de la cadena. Puede emplear notación de tablas ([]) para examinar un carácter concreto en una cadena.

Consejo: El último carácter en una cadena está en la posición Length - 1, y el primero en la posición 0.

```

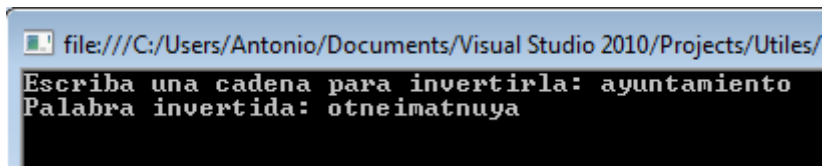
public static void Reverse(ref string s)
{
    string sRev = "";
    for (int k = s.Length - 1; k >= 0; k--)
        sRev = sRev + s[k];
    s = sRev;
}

```

Paso 4. En el método Main, cree una variable string.

- Paso 5.** Utilice `Console.ReadLine` para leer un valor en la variable `string`.
- Paso 6.** Pase la cadena de caracteres a `Reverse`. No olvide la palabra reservada `ref`.
- Paso 7.** Muestre el valor devuelto por `Reverse`.
- Paso 8.** Compile el proyecto y corrija los posibles errores. Ejecute y pruebe el programa.

```
static void Main(string[] args)
{
    string mensaje;
    Console.Write("Escriba una cadena para invertirla: ");
    mensaje = Console.ReadLine();
    Utiles.Reverse(ref mensaje);
    // Mostrar el resultado
    Console.WriteLine("Palabra invertida: " + mensaje);
    Console.Read();
}
```



```
file:///C:/Users/Antonio/Documents/Visual Studio 2010/Projects/Utiles/
Escriba una cadena para invertirla: ayuntamiento
Palabra invertida: otneimatnuya
```

EJERCICIO 7

Copia en mayúsculas de un archivo

En este ejercicio escribirá un programa que pida al usuario el nombre de un archivo de texto. El programa comprobará que el archivo existe, y en caso contrario mostrará un mensaje y se cerrará. Abrirá el archivo y lo copiará a otro (cuyo nombre preguntará al usuario), pero con todos los caracteres en mayúsculas.

Antes de empezar, tal vez sea conveniente que revise brevemente la documentación para `System.IO` en los documentos de ayuda del SDK de .NET Framework. En particular, consulte la documentación para las clases `StreamReader` y `StreamWriter`.

- Paso 1.** Abra el proyecto `MayusculasFichero`.
- Paso 2.** Edite la clase `Program.cs` y añada una instrucción `using` para el espacio de nombres `System.IO`.
- Paso 3.** En el método `Main`, declare dos variables `string` llamadas `sFrom` y `sTo` que contendrán los nombres de los archivos de entrada y salida.
- Paso 4.** Declare una variable de tipo `StreamReader` llamada `srFrom`. Esta variable contendrá la referencia al archivo de entrada.
- Paso 5.** Declare una variable de tipo `StreamWriter` llamada `swTo`. Esta variable contendrá la referencia a la secuencia de salida.
- Paso 6.** Pregunte el nombre del archivo de entrada, lea el nombre y almacénalo en la variable `string sFrom`.
- Paso 7.** Pregunte el nombre del archivo de salida, lea el nombre y almacénalo en la variable `string sTo`.

```
string sFrom;
string sTo;
StreamReader srFrom;
StreamWriter swTo;

Console.Write("Path del Fichero de lectura: ");
sFrom = Console.ReadLine();
Console.Write("Path del Fichero de escritura: ");
sTo = Console.ReadLine();
```

Paso 8. Las operaciones de entrada/salida que va a utilizar pueden producir excepciones. Comience un bloque try-catch que pueda capturar FileNotFoundException (para archivos que no existen) y Exception (para cualquier otra excepción). Imprima un mensaje adecuado para cada excepción.

```
try
{
}
catch (FileNotFoundException)
{
    Console.WriteLine("Fichero de entrada no encontrado");
}
catch (Exception e)
{
    Console.WriteLine("Excepción inesperada");
    Console.WriteLine(e.ToString());
}
```

Paso 9. En el bloque try, cree un nuevo objeto StreamReader usando el nombre del archivo de entrada en sFrom, y almacénelo en la variable referencia StreamReader srFrom.

Paso 10. Del mismo modo, cree un nuevo objeto StreamWriter usando el nombre del archivo de salida en sTo, y almacénelo en la variable referencia StreamWriter swTo.

```
srFrom = new StreamReader(sFrom);
swTo = new StreamWriter(sTo);
```

Paso 11. Añada un bucle while que se ejecute si el método Peek de la secuencia de entrada no devuelve -1. Dentro del bucle:

- Use el método ReadLine sobre la secuencia de entrada para leer la línea siguiente en una variable string llamada sBuffer.
- Aplique el método ToUpper a sBuffer.
- Use el método WriteLine para enviar sBuffer a la secuencia de salida.

```
while (srFrom.Peek() != -1)
{
    string sBuffer = srFrom.ReadLine();
    sBuffer = sBuffer.ToUpper();
    swTo.WriteLine(sBuffer);
}
```

Paso 12. Una vez finalizado el bucle, cierre las secuencias de entrada y de salida.

```
srFrom.Close();
swTo.Close();
```

Paso 13. Guarde el trabajo realizado. Compile el proyecto y corrija los posibles errores.

```
static void Main(string[] args) {  
    string sFrom;  
    string sTo;  
    StreamReader srFrom;  
    StreamWriter swTo;  
  
    Console.WriteLine("Path del Fichero de lectura: ");  
    sFrom = Console.ReadLine();  
    Console.WriteLine("Path del Fichero de escritura: ");  
    sTo = Console.ReadLine();  
  
    try {  
        srFrom = new StreamReader(sFrom);  
        swTo = new StreamWriter(sTo);  
  
        while (srFrom.Peek() != -1) {  
            string sBuffer = srFrom.ReadLine();  
            sBuffer = sBuffer.ToUpper();  
            swTo.WriteLine(sBuffer);  
        }  
  
        srFrom.Close();  
        swTo.Close();  
        Console.Read();  
    } catch (FileNotFoundException) {  
        Console.WriteLine("Fichero de entrada no encontrado");  
    } catch (Exception e) {  
        Console.WriteLine("Excepción inesperada");  
        Console.WriteLine(e.ToString());  
    }  
}
```

