

Seminario de Lenguajes (.NET)

Práctica 2

Ejercicio obligatorio para el coloquio: Los **ejercicios 14, 16 y 17** de esta práctica forman parte del conjunto de ejercicios de programación obligatorios que el alumno debe resolver y exponer de manera oral sobre máquina el día del coloquio hacia final de la cursada.

1. Qué líneas del siguiente código provocan conversiones *boxing* y *unboxing*.

```
char c1 = 'A';
string st1 = "A";
object o1 = c1;
object o2 = st1;
char c2 = (char)o1;
string st2 = (string)o2;
```

2. Sea el siguiente método **Main**.

```
static void Main(string[] args)
{
    object o1 = "A";
    object o2 = o1;
    o2 = "Z";
    Console.WriteLine(o1 + " " + o2);
}
```

El tipo **object** es un tipo referencia, por lo tanto luego de la sentencia **o2 = o1** ambas variables están apuntando a la misma dirección. ¿Cómo explica entonces que el resultado en la consola no sea “**Z Z**”?

3. Analizar la siguiente porción de código para calcular la sumatoria de 1 a 10. ¿Cuál es el error? ¿Qué hace realmente?

```
int sum = 0;
int i = 1;
while (i <= 10);
{
    sum += i++;
}
```

4. ¿Cuál es la salida por consola si no se pasan argumentos por la línea de comandos?

```
class Program
{
    public static void Main(string[] args)
    {
        System.Console.WriteLine(args == null);
        System.Console.WriteLine(args.Length);
    }
}
```

5. ¿Qué hace la instrucción

```
int[] vector = new int[0];
```

¿asigna a vector el valor **null**?

6. Determinar qué hace el siguiente programa y explicar qué sucede si no se pasan parámetros cuando se invoca desde la línea de comandos.

```
class Program
{
    public static void Main(String[] args)
    {
        Console.WriteLine("¡Hola {0}!", args[0]);
    }
}
```

7. Analizar el siguiente código. ¿Qué líneas producen error de compilación y por qué?

```
public static void Main(string[] args)
{
    char c;
    char? c2;
    string st;
    c = "";
    c = ' ';
    c = null;
    c2 = null;
    c2 = (65 as char?);
    st = "";
    st = ' ';
    st = null;
    st = (char)65;
    st = (string)65;
    st = 47.89.ToString();
}
```

8. Escribir un programa que reciba una lista de nombres como parámetro e imprima por consola un saludo personalizado para cada uno de ellos.
 - a) utilizando la sentencia **for**
 - b) utilizando la sentencia **foreach**
9. Investigar acerca de la clase **StringBuilder** del espacio de nombre **System.Text** ¿En qué circunstancias es preferible utilizar **StringBuilder** en lugar de utilizar **string**? Implementar un caso de ejemplo en el que el rendimiento sea claramente superior utilizando **StringBuilder** y otro en el que no.
10. Investigar sobre el tipo **DateTime** y usarlo para medir el tiempo de ejecución de los algoritmos implementados en el ejercicio anterior.
11. ¿Para qué sirve el método **Split** de la clase **string**? Usarlo para escribir en la consola todas las palabras (una por línea) de una frase ingresada por el usuario.
12. Comprobar el funcionamiento del siguiente programa y dibujar el estado de la pila y la memoria *heap* cuando la ejecución alcanza los puntos indicados (comentarios en el código)

```
using System;
using System.Text;
class Program
{
    private static void Main(string[] args)
    {
        object[] v = new object[10];
        v[0] = new StringBuilder("Net");
        for (int i = 1; i < 10; i++)
        {
            v[i] = v[i - 1];
        }
        (v[5] as StringBuilder).Insert(0, "Framework .");
        foreach (StringBuilder s in v)
            Console.WriteLine(s);

        //dibujar el estado de la pila y la mem. heap
        //en este punto de la ejecución


        v[5] = new StringBuilder("CSharp");
        foreach (StringBuilder s in v)
            Console.WriteLine(s);

        //dibujar el estado de la pila y la mem. heap
        //en este punto de la ejecución
    }
}
```

13. Definir el tipo de datos enumerativo llamado **Meses** y utilizarlo para:
 - a) Imprimir en la consola el nombre de cada uno de los meses en orden inverso (diciembre, noviembre, octubre ..., enero)
 - c) Solicitar al usuario que ingrese un texto y responder si el texto tipeado corresponde al nombre de un mes**Nota:** en todos los casos utilizar un **for** iterando sobre una variable de tipo **Meses**

14. Implementar un programa que muestre todos los números primos entre 1 y un número natural dado (pasado al programa como argumento por la línea de comandos). Definir el método `static bool esPrimo(int n)` que devuelve `true` sólo si n es primo. Esta función debe comprobar si n es divisible por algún número entero entre 2 y la raíz cuadrada de n . (Nota: `Math.sqrt(d)` devuelve la raíz cuadrada de d)
15. Escribir una función (método `static int Fib(int n)`) que calcule el término n de la serie de Fibonacci.
- `Fib(n) = 1, si $n \leq 2$`
`Fib(n) = Fib(n-1) + Fib(n-2), si $n > 2$`
16. Escribir una función (método `static int Fac(int n)`) que calcule el factorial de un número n pasado al programa como parámetro por la línea de comando
- a) Definiendo una función no recursiva
 - b) Definiendo una función recursiva
 - c) idem a b) pero con *expression-bodied methods* (Tip: utilizar el operador condicional ternario)
17. Ídem. al ejercicio 16.a) y 16.b) pero devolviendo el resultado en un parámetro de salida
- `static void Fac(int n, out int f)`
18. Codificar el método `Swap` que recibe 2 parámetros enteros e intercambia sus valores. El cambio debe apreciarse en el método invocador.
19. Codificar el método `Imprimir` para que el siguiente código produzca la salida por consola que se observa. Considerar que el usuario del método `imprimir` podría querer más adelante imprimir otros datos, posiblemente de otros tipos pasando una cantidad distinta de parámetros cada vez que invoque el método.
- Tip:** usar `params`

```
static void Main(String[] args)
{
    Imprimir(1, "casa", 'A', 3.4, DayOfWeek.Saturday);
    Imprimir(1, 2, "tres");
    Imprimir();
    Imprimir("-----");
}
```



```
1 casa A 3,4 Saturday
1 2 tres
-----
```