Licenciatura en Gestión Tecnológica.

Universidad Nacional de La Matanza

Cátedra Programación Avanzada 2

Guía Práctica: Introducción C#

Equipo N° 8

Integrantes:

DNI	Apellido y Nombre	E-Mail
28.384.234	Rodriguez Pablo	pablrodriguez@alumno.unlam.edu.ar
94.142.822	Leiva Carballo Aldo	aleivacarballo@alumno.unlam.edu.ar
29.393.712	Montes Pablo	pmontes@alumno.unlam.edu.ar
32.891.140	Espantoso Luciano	lespantoso@alumno.unlam.edu.ar
38.425.874	Alfonzo Virginia	valfonzoaguirre@alumno.unlam.edu.ar
35.799.794	Maizares Nadina	nmaizares@alumno.unlam.edu.ar
25.100.799	Bollini Maximiliano	mbollinilandajo@alumno.unlam.edu.ar

AÑO 2022

Generalidades

1) Crear una función que devuelva la suma de dos números recibidos por parámetros

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
namespace ejercicio
{
    class Program
    {
        static void Sumar(int num1, int num2) {
            Console.WriteLine($"la suma de {num1} + {num2} es: {(num1 +
num2)}");
        }
        static void Main(string[] args)
        {
            Sumar(12, 15);

        }
    }
}
```

2) **Crear una función que reciba una cadena de 8 caracteres y retorne en el mismo parámetro la cadena cortada de izquierda a derecha en 4 caracteres.**

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
namespace ConsoleApp2
{
    internal class Program
    {
        static void Main(string[] args)
        {
            Console.WriteLine("Ingrese una frase de 8 o mas caracteres");
            String cadena = Console.ReadLine().ToString();
            cadena = cadena.Substring(0, 4);
            Console.WriteLine(cadena);

            Console.ReadKey();
        }
    }
}
```

3) Crear una función que devuelva la fecha y hora actual

```
*/
DateTime thisDate1 = new DateTime(2022, 6, 10);
Console.WriteLine("Hoy es" + thisDate1.ToString("MMMM dd, yyyy") + ".");

DateTimeOffset thisDate2 = new DateTimeOffset(2022, 6, 10, 15, 24, 16,
                                              TimeSpan.Zero);
Console.WriteLine("La fecha y hora actual: {0:MM/dd/yy H:mm:ss zzz}",
                  thisDate2);
// Esto se mostraría de la siguiente forma:
//  Hoy es 10 de Junio 2022.
//  La fecha y hora actual: 06/10/11 15:24:16 +00:00*/

private void Form1_Load(object sender, EventArgs e)
{
    timer1.Enabled = true;
}
private void timer1_Tick(object sender, EventArgs e)
{
    lblHora.Text = DateTime.Now.ToString("hh:mm:ss");
    lblFecha.Text = DateTime.Now.ToLongDateString();
}
```

4) Escribir un comentario con //

- Hecho en los ejercicios anteriores

5) Escribir un comentario con /* */

- Hecho en los ejercicios anteriores

Enumeraciones

1) **Crear una enumeración con los días de la semana, comenzando por Domingo con valor 1.**

```
public enum DayOfWeek
{
Sunday = 1,
Monday =  2,
Tuesday = 3,
Wednesday = 4,
Thursday = 5,
Friday = 6,
Saturday = 7
}
```

2) **Agregar a la enumeración la posibilidad de Imprimir un Texto por cada día de la semana**

```
public enum DayOfWeek
{
Sunday = 1,
Monday =  2,
Tuesday = 3,
Wednesday = 4,
Thursday = 5,
Friday = 6,
Saturday = 7
}

class Program
{
static void Main()
{
DayOfWeek day = DayOfWeek.Sunday;
int i = (int) DayOfWeek.Sunday;

System.Console.WriteLine(day);      // Imprime por pantalla el día Domingo
System.Console.WriteLine(i);        // Imprime el 1
}
}
```

Conversiones

1) **Realizar la conversión de true, false, 1 y 0 utilizando los métodos Convert., bool.Parse y bool.TryParse. Explique cómo responde en cada caso cada uno de los métodos indicados.**

Bool.Parse: espera un parámetro que en este caso es string.

```
public static void ParseMethod()
{
    string[] validString = { "true", "True", "    true   ", "false", "False",
"    false" };
    string[] invalidString = {null, "", string.Empty, "t", "    yes   ", "-
1", "0", "1" };
    var values = validString.Concat(invalidString);
    foreach (var value in values)
    {
        try
        {
            Console.WriteLine($"Converted '{value}' to
{bool.Parse(value)}.\n");
        }
        catch (Exception)
        {
            Console.WriteLine($"Unable to convert '{value}' to a
Boolean.\n");
        }
    }
}
```

Convert.ToBoolean: espera un parametro

```
public static class Extensions
string sample = "True";
bool myBool = bool.Parse(sample);

///or
bool myBool = Convert.ToBoolean(sample);
```

Bool.TryParse: espera dos parametros, una entrada (string) y una salida (resultado). Si TryParse es Verdadero, la conversion fue correcta

```
string str = "True";
if(bool.TryParse(str, out bool result))
{
   //Correct conversion
}
else
{
     //Incorrect, an error has occurred
}
```

2) Explique qué sucede en los siguientes intentos de casteos de datos.

a) **int a = (int)344.4**;

Es un error de casting de datos.

Habría que llevarlo a la siguiente forma:

344.4 asignarlo a un decimal
Ej: n = 344.4;

A declararlo como un int

int a;

Y a la variable “a” asignarle de la siguiente forma:

```
decimal n = 344.4;
int a;
a = (int) n;
```

b) decimal a = 10;

La variable espera un decimal, y llega un INT.

c) int a=443444;

```
short b = (short)a;
```

3) Escriba una sentencia switch utilizando una enumeración con 3 colores (blanco, azul y negro) y para cada caso indicar un mensaje de cual es el color informado.

```
enum Color
{
      White,
Blue,
Black
}

class Test
{
    static void Main()
    {
        Console.WriteLine(StringFromColor(Color.White));
Console.WriteLine(StringFromColor(Color.Blue));
        Console.WriteLine(StringFromColor(Color.Black));
    }
    static string StringFromColor(Color c) {
        switch (c) {
            case Color.White:
                return String.Format("White = {0}", (int) c);

            case Color.Blue:
                return String.Format("Blue = {0}", (int) c);

            case Color.Black:
                return String.Format("Black = {0}", (int) c);

            default:
                return "Invalid color";
        }
    }
}
```

4) Si se tiene una variable entera a, realice una sentencia if para evaluar si la variable a es mayor a 10. Si es verdad, mostrar un mensaje indicando que el valor es mayor a 10.

```
using System;
class NumMayor
    {
    public static void Main()
{
        int ingresado;
        Console.WriteLine("Escriba el número: ");
        ingresado = Convert.ToInt32(Console.ReadLine());
        if (ingresado > 10)
 {
            Console.WriteLine("El número ingresado es Mayor a 10");
            }
        }
    }
```

5) Al ejercicio del punto 4), agregar la sentencia de else y, en ella, indicar un mensaje de error.

```
using System;
class NumMayor
    {
    public static void Main()
{
        int ingresado;
        Console.WriteLine("Escriba el número: ");
        ingresado = Convert.ToInt32(Console.ReadLine());
        if (ingresado > 10)
    {
            Console.WriteLine("El número ingresado es Mayor a 10");
            }
else
    {
            Console.WriteLine("El número ingresado NO es Mayor a 10");
            }
        }
    }
```

6) ¿Cuál es la diferencia entre la sentencia for y foreach? ¿Cuándo se debe utilizar cada una de ellas?

- La instrucción **for**: ejecuta su cuerpo mientras una expresión booleana especificada se evalúe como true.
- La instrucción **foreach**: enumera los elementos de una colección y ejecuta su cuerpo para cada elemento de la colección.
- La instrucción **do**: ejecuta condicionalmente su cuerpo una o varias veces.
- La instrucción **while**: ejecuta condicionalmente su cuerpo cero o varias veces.

En cualquier punto del cuerpo de una instrucción de iteración, se puede salir del bucle mediante la instrucción break, o bien se puede ir a la siguiente iteración del bucle mediante la instrucción continue.
La instrucción for ejecuta una instrucción o un bloque de instrucciones mientras una expresión booleana especificada se evalúa como true. En el ejemplo siguiente se muestra la instrucción for, que ejecuta su cuerpo mientras que un contador entero sea menor que tres:

```
for (int i = 0; i < 3; i++)
{
    Console.Write(i);
}
// Output:
// 012
```

En el ejemplo anterior se muestran los elementos de la instrucción for:

- La sección *inicializador*, que se ejecuta solo una vez, antes de entrar en el bucle. Normalmente, se declara e inicializa una variable de bucle local en esa sección. No se puede acceder a la variable declarada desde fuera de la instrucción for.
- La sección *inicializador* del ejemplo anterior declara e inicializa una variable de contador entero:

```
int i = 0
```

- La sección *condición* que determina si se debe ejecutar la siguiente iteración del bucle. Si se evalúa como true o no está presente, se ejecuta la siguiente iteración; de lo contrario, se sale del bucle. La sección *condición* debe ser una expresión booleana.
- La sección *condición* del ejemplo anterior comprueba si un valor de contador es menor que tres:

```
C#Copiar
i < 3
```

- La sección *iterador*, que define lo que sucede después de cada iteración del cuerpo del bucle.
- La sección *iterador* del ejemplo anterior incrementa el contador:

```
C#Copiar
i++
```

- El cuerpo del bucle, que es una instrucción o un bloque de instrucciones.

La sección iterador puede contener cero o más de las siguientes expresiones de instrucción, separadas por comas:

- expresión de incremento de prefijo o sufijo, como ++i o i++
- expresión de decremento de prefijo o sufijo, como --i o i--
- asignación
- invocación de un método
- expresión await
- creación de un objeto mediante el operador new

Si no declara una variable de bucle en la sección inicializador, también puede usar cero o varias de las expresiones de la lista anterior de dicha sección. En el ejemplo siguiente se muestran varios usos menos comunes de las secciones inicializador e iterador: asignar un valor a una variable externa en la sección inicializador, invocar un método en las secciones inicializador e iterador, y cambiar los valores de dos variables en la sección iterador:

```
int i;
int j = 3;
for (i = 0, Console.WriteLine($"Start: i={i}, j={j}"); i < j;
i++, j--, Console.WriteLine($"Step: i={i}, j={j}"))
{
    //...
}
// Output:
// Start: i=0, j=3
// Step: i=1, j=2
// Step: i=2, j=1
```

Todas las secciones de la instrucción for son opcionales. En el ejemplo, el siguiente código define el bucle for infinito:

```
for ( ; ; )
{
    //...
}
```

Foreach

La instrucción foreach ejecuta una instrucción o un bloque de instrucciones para cada elemento de una instancia del tipo que implementa la interfaz System.Collections.IEnumerable o System.Collections.Generic.IEnumerable<T>, como se muestra en el siguiente ejemplo:

```
var fibNumbers = new List<int> { 0, 1, 1, 2, 3, 5, 8, 13 };
foreach (int element in fibNumbers)
{
    Console.Write($"{element} ");
}
// Output:
// 0 1 1 2 3 5 8 13
```

La instrucción foreach no está limitada a esos tipos. Puede usarla con una instancia de cualquier tipo que cumpla las condiciones siguientes:

- Un tipo tiene el método público GetEnumerator sin parámetros. A partir de C# 9.0, el método GetEnumerator puede ser el método de extensión de un tipo.
- El tipo de valor devuelto del método GetEnumerator tiene la propiedad pública Current y el método público MoveNext sin parámetros, cuyo tipo de valor devuelto es bool.

En el siguiente ejemplo se usa la instrucción foreach con una instancia del tipo System.Span<T>, que no implementa ninguna interfaz:

```
Span<int> numbers = new int[] { 3, 14, 15, 92, 6 };
foreach (int number in numbers)
{
    Console.Write($"{number} ");
}
// Output:
// 3 14 15 92 6
```

Si la propiedad Current del enumerador devuelve un valor devuelto de referencia (ref T donde T es el tipo de un elemento de colección), se puede declarar una variable de iteración con el modificador ref o ref readonly, como se muestra en el siguiente ejemplo:

```
            Span<int> storage = stackalloc int[10];
            int num = 0;
            foreach (ref int item in storage)
            {
                item = num++;
            }
            foreach (ref readonly var item in storage)
            {
                Console.Write($"{item} ");
            }
            // Output:
            // 0 1 2 3 4 5 6 7 8 9
```

Si la instrucción foreach se aplica a null, se produce NullReferenceException. Si la colección de origen de la instrucción foreach está vacía, el cuerpo de la instrucción foreach no se ejecuta y se omite.

7) Defina una variable a que en cada ciclo de una sentencia while incremente su valor en 5. Cuando la variable a exceda el valor de 50, el ciclo while debe finalizar.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace EstructuraRepetitivaWhile1
{
    class Program
    {
        static void Main(string[] args)
        {
            int x;
            x = 0;
            while (x <= 50)
            {
                Console.Write(x);
                Console.Write(" - ");
                x = x + 5;
            }
            Console.ReadKey();
        }
    }
}
```