Escuela de Ingeniería Informática
Universidad de Oviedo

Cláusulas

Seminario 5
Programación Funcional

Ejercicio 1

- Usando LINQ, encuentre una solución al siguiente problema:
  - Calcular el **sumatorio de todos los múltiplos de 3 o 5 que se encuentren por debajo de un nº concreto, por ejemplo 1000**
  - Ejemplo: 3, 5, 6 y 9 son los múltiplos de 3 o 5 que están por debajo de 10
    - La suma de estos números es 23
- Importante: No usar ningún bucle imperativo (**`for`**, **`foreach`**, **`while`**…)

Cláusulas

- Una **cláusula** (*closure*) es una función de primer orden que tiene variables libres ligadas a valores en su ámbito estático
- En cálculo lambda λy. … **x** es una cláusula

  λx. x… λy. … **x**

  x es una variable libre en λy. … **x** pero guarda una referencia a la x externa de λx. x…
- En C#

```
int valor = 1;
Func<int> dobleDeValor = () => valor * 2;
dobleDeValor();  // 2
valor = 7;
dobleDeValor();  // 14
```

Cláusulas

- Las variables libres ligadas de una cláusulas representan **estado**
  - Este estado **puede**, además, **estar oculto** cuando el ámbito de la variable finaliza
- Por tanto, pueden representar **objetos**

```
static Func<int> Contador() {
  int contador = 0;
  return () => ++contador;
}
```

Ejercicio 2

- Las cláusulas también pueden representar **estructuras de control**

```
static void BucleWhile(Func<bool> condicion,
                       Action cuerpo) {
  if (condicion()) {
    cuerpo();
     BucleWhile(condicion, cuerpo);
  }
}
```

- ¿Por qué elemento de programación está sustituyendo la iteración?
- Utilice esta función `BucleWhile` para implementar otro método `Suma` que retorne la suma de un *array* de enteros

Ejercicio 3

- Siguiendo con el ejemplo anterior, haga una función **`Switch`** que implemente un condicional múltiple
  - La ventaja es que las condiciones no tienen por qué ser constantes (por ejemplo `x>0 && x<100`)
  - Hágalo al estilo funcional, es decir, que la función **`Switch`** devuelva un valor de un tipo dado
- Utilice esta función **`Switch`** para implementar otra función **`Cuadrante`** que retorne el cuadrante al que pertenece un ángulo pasado como parámetro
  - Sólo considere ángulos $\in$ [0, 360º]

Ejercicio 4 – Trabajo autónomo

- Vimos cómo la ocultación de información de las cláusulas permite representar objetos

```
static Func<int> Contador() {
  int contador = 0;
  return () => ++contador;
}
```

- Indique cómo implementar mediante cláusulas la creación de instancias de la siguiente clase