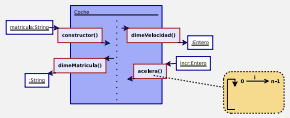
Grado Tecnologías Interactivas

Programación 2



Práctica 1

UNIVERSIDAD POLITECNICA DE VALENCIA

Escola Politècnica Superior de Gandia

DSIC

Departament de Sistemes Informàtics i Computació

Práctica 1

¡ Atención !

▷ Se recuerda que las prácticas deben prepararse antes de acudir al aula informática, anotando en el enunciado las dudas que se tengan.

▷ Los diseños y algortimos que se piden en esta práctica deben escribirse en la libreta de apuntes para poder ser revisados.

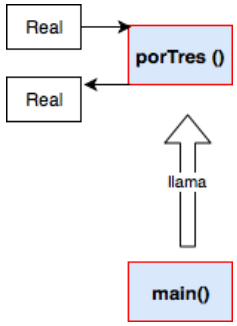
▷ Utiliza *git* en cada ejercicio haciendo *commit* cada vez que consigas un "hito".

▷ La realización de las prácticas es un trabajo individual y original. En caso de plagio se excluirá al alumno de la asignatura. Por tanto, es preferible presentar el trabajo realizado por uno mismo aunque éste tenga errores.

1

Funciones sencillas y "callbacks"

La función porTres() recibe un número real y devuelve el resultado de multiplicarlo por tres. Su diseño es el siguiente:



Y su algoritmo es éste:

```
Datos de entrada: a : ℝ
Datos de salida: ℝ


Devolver a · 3
```

1. En el fichero mainPorTres1.js, implementa el diseño de la función porTres() utilizando un return para devolver el resultado. Llama a esta función escribiendo una prueba automática para comprobar que va correctamente.

▷ Atención: "devolver" no es escribir en pantalla.

▷ Atención: Hay dos tipos mutuamente excluyentes de funciones:
- las de entrada/salida: se comunican con el usuario (escriben en pantalla y/o leen de teclado). No hacen cálculos. Ejemplo: main().
- las de cálculo: no se comunican con el usuario (ni leen ni escriben), sólo calculan. Ejemplo: porTres().

2. En el fichero mainPorTres2.js, implementa el diseño de la función porTres() utilizando un callback para devolver el resultado. Llama a esta función escribiendo una prueba automática para comprobar que va correctamente.

2 Asincronía

1. La función de biblioteca `setTimeout()` recibe un `callback` y la cantidad de milisegundos que debe esperar antes de llamar al callback.

```
//
// main ()
//
console.log( " antes " )

setTimeout( function() {
                    console.log( " pasaron dos segundos " )
            }, 2000)

console.log( " después " )
```

   Escribe el anterior código en un fichero `.js`, pruébalo y deduce:
   ¿por qué lo último en escribirse en pantalla es "pasaron dos segundos" y no "después"?

2. Prueba el siguiente código

```
setTimeout ( function () {
                console.log (" hola 1 ");
             }, 2000 )

setTimeout ( function () {
                console.log (" hola 2 ");
             }, 2000 )
```

   Responde: ¿por qué salen los dos mensajes al mismo tiempo?

   Modifica el anterior código para que el primer mesaje tarde 2 segundos en aparecer y el siguiente mensaje aparezca 2 segundos tras el primero.
3. Escribe una versión del diseño de la función `porTres()` en la que ésta tarde tres segundos en devolver el resultado. Escribe una prueba automática para esta versión de la función.

3 Filtros

```
//
//
//
function sumar( inicio, fin, condicion ) {
  var total = 0
  for( var i=inicio; i<=fin; i++ ) {
    if ( condicion(i) ) {
      total = total + i
    }
  } // for
  return total
} // ()

//
// main()
//
var s = sumar( 1, 10, function(e) {
                        if (e % 3 == 0 ) {
                               return true
                        }
                        return false
                      })

console.log( s )
```

- ¿Qué hace la función sumar()?
- ¿Es asíncrona la función sumar()?

- ¿Es la función `function(e) { ...` un callback?
- ¿Cuál es el diseño de la función `sumar()`?
- ¿Por qué cuando una función recibe un callback, éste no aparece en el diseño?
- Diseña, implementa y realiza pruebas automáticas para una función que calcule:

$$\int_a^b f(x)dx$$

30 enero 2019