



## **Guion de prácticas 2**

*Paso de Parámetros y Uso de estructuras*



**Metodología de la Programación**

Curso 2017/2018



## Introducción al guion

En este guion se pondrán en práctica los conceptos asociados al uso de structs y el paso de parámetros a funciones. La programación se realizará utilizando herramientas provistas en una instalación Linux estándar.

## Ejercicio 1: Manejo de Tiempos

Defina una estructura para representar un instante de tiempo. Debe almacenar horas (entre 0 y 23), minutos (entre 0 y 59) y segundos (entre 0 y 59). Posteriormente defina las siguientes funciones:

- **esPosterior**: Dados dos instantes de tiempo devuelve *true* si el segundo instante es posterior al primero y *false* en caso contrario.
- **convertirASegundos**: transforma el instante de tiempo dado, a un valor en segundos. Por ejemplo, si tenemos 1 hora, 1 minuto y 3 segundos, debería devolver 3663 segundos.
- **obtenerNuevoTiempo**: Dados un instante de tiempo  $T$  y un valor entero  $S$  (que representa una cantidad de segundos), devuelva un nuevo instante de tiempo  $T2$  que represente la suma de  $S$  segundos a  $T$ . Los valores de  $T2$  deben estar en los intervalos correctos.

Implemente todo en un único fichero *tiempo.cpp*. Analice cuidadosamente el número y tipo de los parámetros de las funciones. En el *main* muestre ejemplos de uso de las tres funciones.

## Ejercicio 2: Lista de la Compra

En este ejercicio se pretende representar una compra realizada en una frutería. Para ello, se define una estructura *Producto* que permite representar UN producto que se haya comprado. De cada producto se almacena su nombre, peso (en gramos) y precio por Kg.

Posteriormente, podemos representar la información asociada a la compra de varios productos utilizando un struct *Compra* que contiene un vector de elementos de tipo *Producto*. Estas definiciones están incluidas en el fichero *fruteria.cpp* que tiene disponible en PRADO.

Además, el código incluye la definición de una función auxiliar para mostrar el contenido de una variable de tipo *Producto*, y un conjunto de instrucciones en la función *main* para probar las funciones a implementar.

```

Bitwise xterm - david@modo.ugr.es:22 - david@modo: ~/MP
david@modo:~/MP$ g++ fruteria.cpp -std=c++0x -o compra
david@modo:~/MP$ ./compra < datos.txt

***** Prueba de funcion listarCompra *****
cereza      345      2.550000
naranja     1380     1.100000
kiwi        876      1.800000
pera       1150      2.000000
platano     890      1.190000
melon      3500      1.500000
uva         530      2.100000
mango       456      2.500000
manzana     750      1.690000
limon       275      1.190000

***** Prueba de funcion obtenerImporteYPeso *****
El importe de la compra es: 16.4314, su compra pesa:10 Kg.

***** Prueba de la funcion mostrarTicketCompra *****
cereza      345      2.550000
naranja     1380     1.100000
kiwi        876      1.800000
pera       1150      2.000000
platano     890      1.190000
melon      3500      1.500000
uva         530      2.100000
mango       456      2.500000
manzana     750      1.690000
limon       275      1.190000
Subtotal:           16.4314
IVA (21%):           3.45059
Total de la compra:  19.882

***** Prueba de la funcion incrementarPrecio *****
***** y listarCompra de nuevo *****
cereza      345      2.805000
naranja     1380     1.210000
kiwi        876      1.980000
pera       1150      2.200000
platano     890      1.309000
melon      3500      1.650000
uva         530      2.310000
mango       456      2.750000
manzana     750      1.859000
limon       275      1.309000
david@modo:~/MP$

```

Figura 1: Compilación, ejecución y salida a mostrar por el programa. El parámetro `-std=c++0x` que aparece en la orden de compilación permite la utilización de la función `to_string()` que solo está disponible bajo el estándar de C++11. Dependiendo de la versión del compilador, puede que necesite usar `-std=c++11` o `-std=gnu++11`

## Tareas a Realizar

Cree un directorio `fruteria` que contenga a su vez los directorios `src`, `bin`, `include`, `obj`. Complete la implementación de las funciones indicadas en el fichero `fruteria.cpp` (donde se incluye una breve descripción de su funcionalidad). Construya una versión modular del código (incluyendo los ficheros `.cpp` y `.h` del módulo, un fichero `compra.cpp` y el `makefile` correspondiente).

Si el directorio actual es `fruteria`, la ejecución de los comandos `make`

`./bin/compra <datos.txt`

debe obtener una salida similar a la mostrada en la Figura 1. El fichero `datos.txt` (disponible en PRADO) contiene valores que se pueden asignar al vector `compra` utilizando la redirección de entrada para la lectura (esto es lo que hace el segundo comando).