Guion de prácticas 0

Paso de Parámetros y uso de estructuras

Febrero de 2018

DECSAI

Metodología de la Programación

Curso 2017/2018

Índice

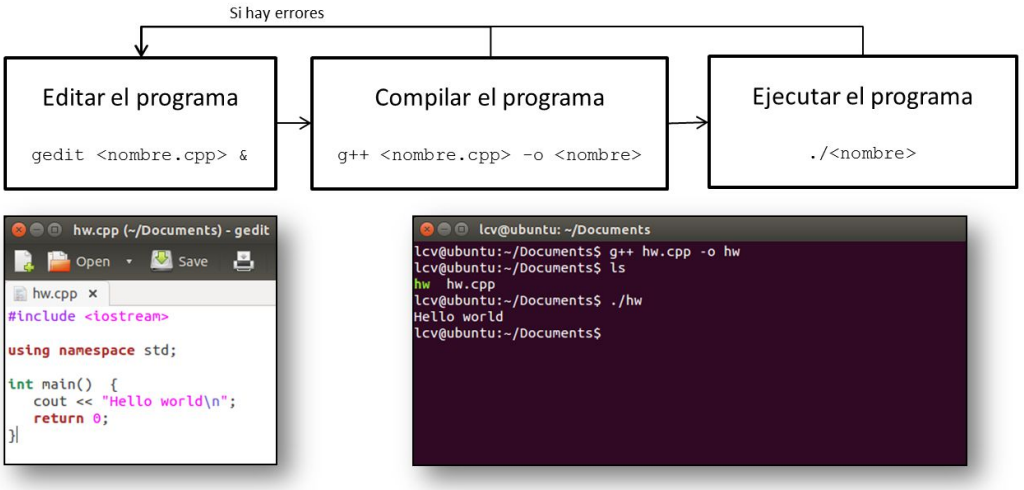

Figura 1: Ciclo de vida de la compilación y ejecución de programas en C++ en Linux

1. 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.

2. Un breve apunte sobre la compilación de C++ en Linux

Aunque el proceso completo de compilación se verá con todo detalle en el siguiente guion, esta sección muestra brevemente el ciclo de edición-compilación-ejecución en una instalación de Linux estándar.

2.1. Edición

Los ficheros con extensión cpp se pueden editar con programas estándar de la consola de Linux como `vi`, `vim` o en modo gráfico (Gnome) con programas como `gedit` (Ver Figura 1).

2.2. Compilación

Una vez guardado el programa en un fichero con extensión `cpp` el siguiente paso es compilarlo con `g++` mediante la orden mostrada en la Figura 1.

Los errores y/o advertencias que pueda generar el compilador son del mismo tipo de las que se obtenían al utilizar tanto `DevC++` como `CodeBlocks`. Dichos entornos utilizan una versión de `g++` para Windows, y por tanto, la gestión de los errores no debería representar ningún problema.

2.3. Ejecución

Si la compilación ha funcionado correctamente, la ejecución del programa se realiza tal y como muestra la Figura 1, observando en la consola el resultado.

3. Ejercicios

3.1. 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.
- **convertirATiempo**: realiza la operación inversa a la anterior, transforma un valor numérico en segundos en un instante de tiempo válido. Por ejemplo, transformaría el valor 3663 en el instante correspondiente a 1 hora, 1 minuto y 3 segundos.
- **incrementarTiempo**: 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.
- **repararTiempo**: Dado un instante de tiempo T que pudiese contener valores acumulados fuera de los límites permitidos, como por ejemplo 1 hora 59 minutos y **70** segundos, corrija estos valores dentro de sus límites permitidos, incrementando los valores que se consideren necesarios y de vuelva el resultado corregido. En el ejemplo anterior debería devolver 2 horas, 0 minutos y 10 segundos.

Implemente todo en un único fichero *tiempo.cpp* que podrá descargar desde `DECSAI` parcialmente relleno. Analice cuidadosamente el número y tipo de los parámetros (valor, referencia) de las funciones. En el *main* se muestran ejemplos de uso de las tres funciones y en la Figura 2 se muestran dos ejemplos de ejecución.

3.2. Compra en una Frutería

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.

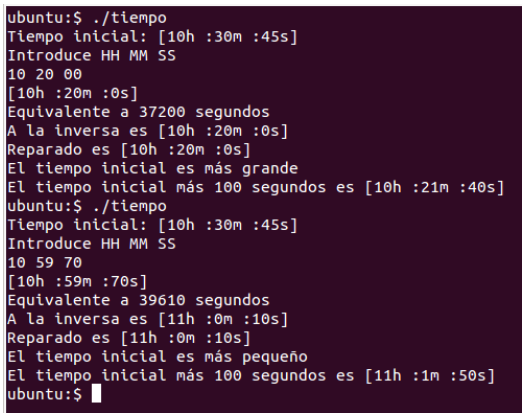

Figura 2: Dos ejemplos de ejecución del programa de manejo de tiempos

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 incluídas en el fichero *fruteria.cpp* que tiene disponible en DECSAI.

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. A partir de este código se pide completar la implementación de las funciones indicadas en el fichero fuente (donde se incluye una breve descripción de su funcionalidad).

La ejecución del código debe proporcionar una salida similar a la mostrada en la Figura 3. Para asignar los datos al vector *compra*, utilice el fichero *datos.txt* (disponible en DECSAI) y utilice la redirección de entrada para la lectura (la segunda línea de la imagen muestra como hacerlo).

```
ubuntu:$ g++ -o fruteria fruteria.cpp
ubuntu:$ ./fruteria < datos.txt

 ******** Prueba de funcion listarCompra ********
cereza          345     2.55
naranja         1380    1.1
kiwi            876     1.8
pera            1150    2
platano         890     1.19
melon           3500    1.5
uva             530     2.1
mango           456     2.5
manzana         750     1.69
limon           275     1.19

 ******** Prueba de funcion obtenerImporteYPeso ********

El importe de la compra es: 16.4314, su compra pesa:10 Kg.

 ******** Prueba de la funcion mostrarTicketCompra *******
cereza          345     2.55
naranja         1380    1.1
kiwi            876     1.8
pera            1150    2
platano         890     1.19
melon           3500    1.5
uva             530     2.1
mango           456     2.5
manzana         750     1.69
limon           275     1.19
Subtotal:               16.4314
IVA (21%):              3.45059
Total de la compra:     19.882

 ******** Prueba de la funcion incrementarPrecio ********
 ******** y listarCompra de nuevo                ********

cereza          345     2.805
naranja         1380    1.21
kiwi            876     1.98
pera            1150    2.2
platano         890     1.309
melon           3500    1.65
uva             530     2.31
mango           456     2.75
manzana         750     1.859
limon           275     1.309
ubuntu:$
```

Figura 3: Compilación, ejecución y salida a mostrar por el programa.