Guion de prácticas 0

Paso de Parámetros y uso de estructuras

Febrero de 2016

DECSAI

Metodología de la Programación

Curso 2015/2016

Índice

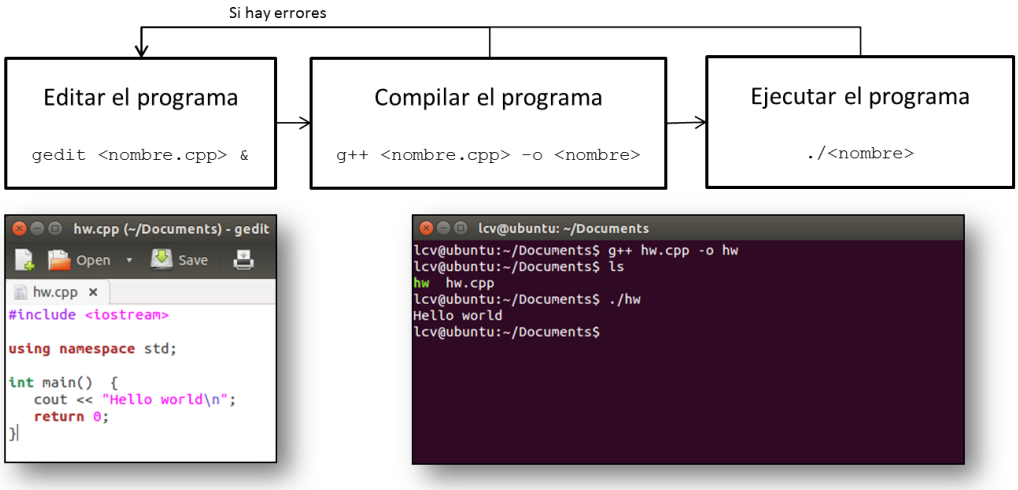

Figura 1: Ciclo de vida de la compilación y ejecución de programas en C++ en Linux

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.
- **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 (valor, referencia) de las funciones. En el *main* muestre ejemplos de uso de las tres funciones.

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.

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. Ambos elementos se muestran en las Figuras 3 y 4.

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).

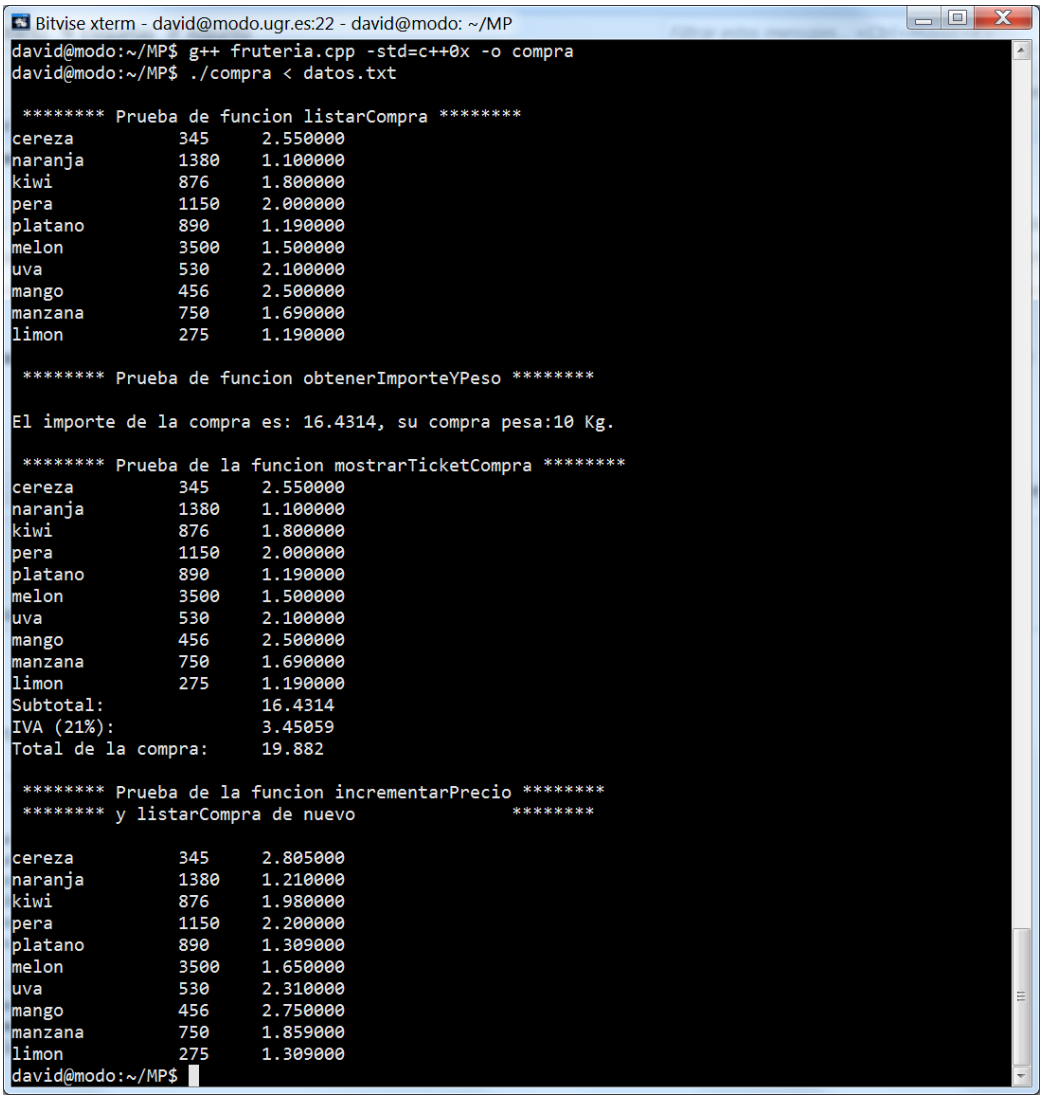

Figura 2: Compilación, ejecución y salida a mostrar por el programa. El parámetro *-std=c++0x* que aparece en la orden de compilación permite la utilización de la función *to_string()* que solo está disponible bajo el estándar de C++11. Dependiendo de la versión del compilador, puede que necesite usar *-std=c++11* o *-std=gnu++11*

La ejecución del código debe proporcionar una salida similar a la mostrada en la Figura 2. 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).

```cpp
#include<iostream>
#include<string>

using namespace std;

struct Producto
{
    string nombre;
    int peso; // en gramos
    float precio_kg;
};

struct Compra
{
    static const int MAX = 10;
    Producto lista[MAX];
};

// función para devolver los datos de un producto como un string
string productoToString(Producto a)
{
    string rta;
    rta = a.nombre + "\t\t" + to_string(a.peso) + "\t" + to_string(a.precio_kg);
    return rta;
}

// crea una variable de tipo Producto y la devuelve
Producto creaProducto(string nombre, int peso, float precio)
{
}

// incrementa el precio de cada producto en un k %
void incrementarPrecios(Compra & c, int k)
{
}

// muestra el listado de productos comprados
void listarCompra(const Compra & c)
{
}

// función que devuelve el importe total de la compra (sinIVA) y su peso (en kgs)
void obtenerImporteYPeso(const Compra & c, float & precio, int & peso)
{
}

// función que muestra el "ticket" de compra según el formato sugerido
void mostrarTicketCompra(const Compra & c)
{
}
```

Figura 3: Definiciones y funciones a implementar.

```
int main()
{
    const int TAM = 10;
    Producto actual;
    Compra mi_compra;
    float precio;
    int peso;
    string nombre;

    for(int i = 0; i < TAM; i++)
    {
        cin >> nombre >> peso >> precio;
        mi_compra.lista[i] = creaProducto(nombre, peso, precio);
    }

    // se muestra la lista de productos.
    cout << "\n ******** Prueba de funcion listarCompra ******** \n";
    listarCompra(mi_compra);

    // se muestra el importe total de la compra y el peso
    cout << "\n ******** Prueba de funcion obtenerImporteYPeso ******** \n";
    obtenerImporteYPeso(mi_compra, precio, peso);
    cout << "\nEl importe de la compra es: " << precio << ", su compra pesa:"
         << peso << " Kg. " << endl;

    // se muestra el ticket de la compra.
    cout << "\n ******** Prueba de la funcion mostrarTicketCompra ******** \n";
    mostrarTicketCompra(mi_compra);

    cout << "\n ******** Prueba de la funcion incrementarPrecio ********";
    incrementarPrecios(mi_compra, 10);
    cout << "\n ******** y listarCompra de nuevo   \t\t********\n\n";
    listarCompra(mi_compra);

    return(0);
}
```

Figura 4: Estructura de la función main. No modifique este código.