Deusto
Facultad de Ingeniería

EXAMEN PROGRAMACIÓN IV

(Parcial lenguaje C, 02 de abril de 2020)

Nombre y apellidos: _____

INSTRUCCIONES PARA EL EXAMEN

- La entrega del examen resuelto se realiza a través de la plataforma ALUD. Existe una tarea para la entrega del examen llamada **Entrega examen 02/04/2020**
- El alumno debe crear un único **archivo ZIP** que contenga TODO el proyecto C y los ficheros requeridos, y subirlo a la plataforma. Solamente se corregirán los archivos entregados. Comprobar bien el contenido del ZIP antes de subirlo.
- La duración del examen es de **1 hora y 45 minutos**.
- Se permite el uso de cualquier material en el examen (prácticas y apuntes).

PREÁMBULO

Se parte de un módulo "producto" ya creado (ficheros **"producto.h"** y **"producto.c"**) que declara la estructura `Producto` (un entero a modo de referencia del producto, una cadena de caracteres para su nombre y un valor decimal para su precio) y un fichero **"main.c"** que contiene el programa principal. Este programa principal crea e inicializa cinco productos, así como un array con ese mismo tamaño donde los almacena. Todo este código que ya se te proporciona NO debe ser alterado en ningún momento.

Antes de comenzar el examen, prueba que este mínimo programa funciona correctamente de modo que estés seguro que el entorno de desarrollo se encuentra bien configurado. Deberían visualizarse los datos de uno de los productos. Una vez hecha esta comprobación, el examen consta de tres partes que deberás ir resolviendo incrementalmente.

NOTA: los parámetros que se indican con la expresión `<TYPE>` a lo largo de este enunciado para las distintas funciones a implementar en el examen son orientativos, es decir, no se detalla en ningún caso si éstos deben ser punteros o estructuras. Será el alumno quien deba decidir y actuar en consecuencia. En muchos casos existirán varias alternativas correctas.

PARTE 1 (0,5 puntos)

1.1 Crea un nuevo módulo "carrito" (con su correspondiente fichero de cabecera y de implementación), dentro del cual se defina inicialmente la estructura `Compra`. Esta estructura representará la compra de un cierto número de unidades de un único producto. Deberá contener por tanto el producto comprado (deberás hacer uso de la estructura `Producto` que ya se proporciona) y la cantidad comprada.

1.2 Añade dentro del módulo "carrito" estas dos funciones para operar con la estructura `Compra`:

> `void modificarCompra(<Compra> c, int cant)`: modifica la compra pasada como primer parámetro, asignándole como cantidad el valor del segundo parámetro.
>
> `void imprimirCompra(<Compra> c)`: visualiza por pantalla los datos de la compra pasada como parámetro. En el juego de ensayo que se proporciona después tienes un ejemplo de cómo visualizar esta información.

1.3 Dentro de tu programa principal crea una compra de alguno de los productos que ya tienes creados y modifica después su cantidad (por ejemplo poniéndola a cero). Visualiza los datos de la compra antes y después de la modificación de modo que se vea que su cantidad cambió. Deberás usar las dos funciones implementadas en el módulo "carrito" para ello.

Juego de ensayo. Tomando como producto el "Solomillo" y una compra de 3 kg/ud., el resultado por pantalla podría ser el siguiente:

```
ANTES de la modificación de la compra…
Ref.2  Solomillo     32.20 x 3 kg/ud
DESPUÉS de la modificación de la compra…
Ref.2  Solomillo     32.20 x 0 kg/ud
```

PARTE 2 (1,5 puntos)

2.1 Define dentro del módulo "carrito" una nueva estructura **`Carrito`**. Esta estructura representará un carrito de la compra, es decir, contendrá el conjunto de compras realizadas (debes utilizar la estructura **`Compra`** declarada anteriormente) y el importe total de esas compras. Ten presente que el número de compras que componen el carrito no es un dato conocido a priori.

2.2 Añade dentro del módulo "carrito" estas tres funciones para operar con la estructura **`Carrito`**:

`void crearCarrito(<Carrito> c, <array-productos> prods, <array-enteros> cants, int tamanyo)`: partiendo de los arrays de productos y cantidades recibidos como segundo y tercer parámetro, se configura el carrito devuelto como primer parámetro. Al primer producto del array le corresponde la primera cantidad del otro array, al segundo la segunda, y así sucesivamente. El cuarto parámetro indica el tamaño de los otros dos arrays. Ten en cuenta que el importe del carrito se calculará con el sumatorio del precio de cada producto por su cantidad.

`void imprimirTicket(<Carrito> c)`: visualiza por pantalla los datos del carrito pasado como parámetro, como si fuese un ticket de compra. En el juego de ensayo que se proporciona después tienes un ejemplo de cómo visualizar esta información.

`void devolverCarrito(<Carrito> c)`: libera la memoria dinámica reservada para los distintos elementos que componen el carrito.

2.3 Dentro de tu programa principal llama a la función **`crearCarrito`** para obtener un nuevo carrito. Se espera que este carrito se configure con los datos del array **`comprados`** que ya tienes, más el array con las cantidades que quieras. Visualiza después la información del carrito con la función **`imprimirTicket`**.

Juego de ensayo. El resultado por pantalla de un posible carrito sería el siguiente:

```
TICKET
------
Ref.1  Platanos      12.00 x 1 kg/ud
Ref.2  Solomillo     32.20 x 3 kg/ud
Ref.3  Chocolate     25.50 x 5 kg/ud
Ref.4  Rodaballo     35.40 x 7 kg/ud
Ref.5  Cereales      10.50 x 9 kg/ud
.................................
TOTAL: 578.40 euros
```

PARTE 3 (1 punto)

3.1 Añade dentro del módulo "carrito" la siguiente función:

> **`void modificarCarrito(<Carrito> carrito, int ref, int cant)`**: busca en el carrito pasado como primer parámetro la compra correspondiente al producto con la referencia pasada como segundo parámetro y modifica su cantidad, asignándole el valor del tercer parámetro. Recuerda que está modificación debe implicar un recalculo del importe total del carrito. Si no existiese ninguna compra de ese producto, no se hace nada.

3.2 En el programa principal usa esta función para modificar el carrito creado en el apartado 2.3. Haz con ella, por ejemplo, una devolución del producto con referencia 3, asignándole la cantidad cero. Y visualiza después los datos del ticket del carrito.

Juego de ensayo. En el caso anterior el resultado por pantalla sería el siguiente:

```
TICKET
------
Ref.1  Platanos      12.00 x 1 kg/ud
Ref.2  Solomillo     32.20 x 3 kg/ud
Ref.3  Chocolate     25.50 x 0 kg/ud
Ref.4  Rodaballo     35.40 x 7 kg/ud
Ref.5  Cereales      10.50 x 9 kg/ud
.................................
TOTAL: 450.90 euros
```

3.3 Con el enunciado se te proporciona un fichero de texto **"productos.txt"** con los datos de los mismos cinco productos con los que empezaste el examen, cada uno de ellos en una línea. Implementa en el programa principal la siguiente función que trabaje con este fichero:

> **`void leerProductos(Producto* productos[], char* fichero)`**: lee los datos de los productos del fichero cuyo nombre se recibe como segundo parámetro y los devuelve cargados en el array pasado como primer parámetro. Se asume que en el fichero hay siempre 5 productos y la estructura de la información de cada línea es siempre la siguiente (entre corchetes el carácter de inicio y fin de cada dato): referencia [1-1], nombre [2-15], precio [16-20]. Esta función debe tener exactamente la declaración de los argumentos que se indican.

3.4 En el programa principal llama a la función anterior para que lea del fichero "productos.txt" y cargue su información en el array `comprados` que ya se te proporcionaba. Deberás llamar a la función justo en la línea posterior a la declaración e inicialización de ese array, pasándoselo como primer parámetro.

Juego de ensayo. Si lo haces como se te indica, la información los tickets del carrito saldrá igual que en los anteriores juegos de ensayo pero con el sufijo "(f)" en los nombres de todos los productos.

3.5 Finalmente, no olvides liberar la memoria de aquellas estructuras que así lo requieran, si las hubiera.