

Facultad de Informática – Universidad Complutense
1º curso de los Grados
Fundamentos de la programación – Grupos A, B, C, I
Curso 2015-2016

Examen de febrero de 2016

Tiempo disponible: 3 horas

Se quiere desarrollar una aplicación para gestionar la información de los vuelos que diariamente salen de un aeropuerto. La aplicación mantiene la siguiente información acerca de cada uno de los vuelos: nº de vuelo (cadena de caracteres formada por una sola palabra), que lo describe de manera unívoca, destino (una o más palabras), terminal (carácter), puerta de embarque (nº entero) y estado del vuelo (enumerado: embarcando, operado, retrasado o suspendido).

La aplicación debe gestionar una lista de N vuelos (ni uno más ni uno menos; p.e. N = 50), donde cada elemento contiene la información de un vuelo. Aquí tienes un ejemplo de cómo estarían dispuestos los vuelos en la lista (los 8 primeros del total de N vuelos):

IB1231	AF1231	KLM1231	AL1236	LU1301	TP1302	AM1401	AV1411	...
Lisboa	Paris	Ámsterdam	Roma	Zagreb	Oporto	Cancún	Sao Paulo	
A	A	B	A	A	B	B	B	
2	3	1	2	5	2	1	3	
operado	operado	retrasado	despegando	embarcando	retrasado	embarcando	programado	

La información de los vuelos se mantiene en un archivo de texto `salidas.txt` organizado de manera que los datos de un vuelo aparecen en una línea separados por un espacio: número de vuelo, terminal, puerta, estado (0 = embarcando, 1 = operado, 2 = retrasado, 3 = suspendido) y destino. El archivo puede contener más o menos de N vuelos y termina en XXX como número de vuelo (centinela).

Lo primero que hay que hacer es modelar los diferentes elementos de información necesitados ajustándose a las restricciones indicadas y redactar las declaraciones globales (constantes y tipos) correspondientes.

De cara a desarrollar el resto de la aplicación ten en cuenta que la ejecución debe comenzar cargando en la lista de vuelos la información que haya en el archivo `salidas.txt`, realizando el chequeo de errores oportuno (a saber: comprobar que existe el archivo y que éste contiene información acerca de, al menos, N vuelos). Para ello debes implementar el subprograma `cargar()`, que devuelva la lista con los datos cargados del archivo `salidas.txt` así como un booleano que indique si la carga se ha podido realizar sin errores (`true`) o no (`false`). Si hay más de N vuelos en el archivo no se considera un error, simplemente los vuelos que no caben en la lista se ignoran.

En caso de que haya errores en la carga de la información desde el archivo, se informará al usuario y la aplicación finalizará. Si no existen errores se mostrará un menú con las siguientes opciones:

1. Obtener información de un vuelo
2. Actualizar el estado de un vuelo
3. Salir

Se permitirá realizar cada una de ellas tantas veces como sea necesario hasta que el usuario decida salir. En ese momento la aplicación guardará en el archivo `pendientes.txt` la información de aquellos vuelos de la lista que se encuentren en estado retrasado o suspendido. Para ello debes implementar el subprograma `guardar()`, que reciba la lista y guarde en el archivo `pendientes.txt` los datos de los vuelos retrasados o suspendidos, y lo haga con una estructura igual a la que tiene el archivo `salidas.txt`.

La opción 1 solicitará que se introduzca por teclado el nº del vuelo cuya información se quiere obtener y, en caso de coincidir con el nº de vuelo de alguno de los vuelos de la lista, visualizará en la pantalla toda la información del vuelo indicado. En caso contrario se mostrará un mensaje de error.

La opción 2 solicitará que se introduzca por teclado el nº de vuelo a actualizar hasta que coincida con el de un vuelo de la lista. Seguidamente solicitará el nuevo estado y procederá a realizar el cambio en la información almacenada sobre dicho vuelo en la lista.

Para las opciones 1 y 2 debes implementar, al menos, los siguientes subprogramas:

- ✓ `buscarVuelo()`, que recibe la lista de vuelos y un nº de vuelo y devuelve la posición donde se encuentra el vuelo en la lista (-1 si no existe uno igual).
- ✓ `mostrarVuelo()`, que recibe un nº de vuelo y muestra la información del mismo en la pantalla.
- ✓ `actualizarVuelo()`, que recibe la lista, una posición de la misma y un estado de vuelo, y actualiza a ese estado el estado del vuelo en la posición indicada de la lista.

PUNTUACIONES

`carga()`: 2,5; `guardar()`: 1,5; `buscarVuelo()`: 2,5; `mostrarVuelo()`: 1,0;
`actualizarVuelo()`: 0,5; Declaraciones, programa principal y diseño del programa: 2,0

ENTREGA DE LA SOLUCIÓN

Debes subir al campus virtual el archivo `DNI.cpp` con tu código solución (donde DNI es tu número de DNI, NIE o similar).

Asegúrate además de que al comienzo del archivo `cpp` pones un comentario con tu nombre completo y grupo.