

Examen final de junio de 2017

Tiempo disponible: 3 horas

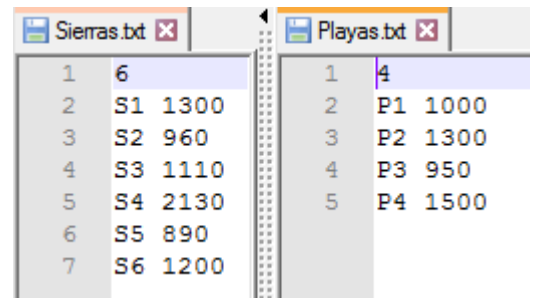
No sé dónde irme de vacaciones este año, pero tengo claro que quiero ir a la playa y a la sierra. Dispongo de información de lugares con playas y montañas en dos ficheros que cargaré en dos listas. Con esas dos listas quiero hacer una lista de viajes que visitan ambos lugares, playa y montaña.

El programa constará de los módulos: *ListaLugares*, *ListaOpciones* y *Principal*.

Programa Principal (1,5 puntos)

Comienza cargando la información proporcionada por los archivos “Playas.txt” y “Sierras.txt”. Si alguno de los archivos no existe, se indica y termina la ejecución. Si existen, entonces la primera línea es el número de líneas con lugares y precios que aparecen a continuación. Se cargan en sendas listas llamadas *playas* y *sierras*. A continuación se muestra el siguiente menú.

1. Mostrar lugares con playa
2. Mostrar lugares con montaña
3. Introduce un nuevo posible viaje
4. Mostrar todas las configuraciones de viaje
0. Terminar.



Sierras.txt		Playas.txt	
1	6	1	4
2	S1 1300	2	P1 1000
3	S2 960	3	P2 1300
4	S3 1110	4	P3 950
5	S4 2130	5	P4 1500
6	S5 890		
7	S6 1200		

Las listas *playas* y *sierras* son del tipo `tListaLugares`. El menú se repite hasta que el usuario elige la opción 0.

Módulo ListaLugares (3,5 puntos)

Define la información de cada lugar con un tipo de estructura llamado `tLugar` y con dos campos: *nombre* (string sin espacios en blanco, por ejemplo S1) y *precio* (int).

Declara un tipo de estructura `tListaLugares`. Esta lista se implementa con un **array estático de punteros a datos dinámicos** (es decir, un array de punteros a `tLugar`), máximo 100.

Con respecto a la implementación se pide que se sobrecarguen los operadores `>>` y `<<` para hacer la carga y visualización de la lista de lugares. Recuerda que los prototipos de estas funciones son:

```
istream& operator>> (istream & in, tListaLugares & lista)
ostream& operator<< (ostream & out, tListaLugares const& lista)
```

También se debe implementar una función **recursiva** que busca un lugar por su nombre en las listas **no ordenadas** de tipo `tListaLugares`.

Por último, define la función que libera la memoria dinámica que se usa.

Módulo *ListaOpciones* (5 puntos)

Define un tipo de estructura `tOpc` con dos campos: `OpcPlaya` y `OpcSierra`. Ambos campos son de tipo puntero a `tLugar`, y apuntarán a lugares ya existentes en las listas *playa* y *sierra*. Además, la estructura `tLugar` tiene un campo entero que indica el precio o coste de ir a ambos lugares.

Declara un tipo de estructura `tListaOpc` para listas de `tOpc` implementadas con **array dinámico** y **ordenada crecientemente por precio** (me interesa el viaje más barato). El array inicialmente tiene una capacidad de 3 y se redimensiona al doble de su capacidad.

El módulo implementa, al menos:

- Sobrecarga del operador `<<` para mostrar los datos de la lista.
- Una función que libere la correspondiente memoria dinámica usada.
- Una función `nuevoViaje` que añade la configuración de un nuevo viaje *playa-sierra* a la lista. Es decir, se pide al usuario que elija un lugar con playa y un lugar con sierra. Estos lugares se buscan en las correspondientes listas y se apuntan con los punteros `OpcPlaya` y `OpcSierra`. Además, se calcula el precio total de ir a la *playa* y a la *sierra*. Este nuevo viaje se inserta en la lista de opciones. Observa que las listas de tipo `tListaOpc` están ordenadas, así que tendrás que buscar la posición dónde insertar este nuevo viaje.

EJEMPLO DE EJECUCIÓN

1. Mostrar lugares con playa

2. Mostrar lugares con montaña

3. Introduce un nuevo viaje

4. Mostrar todas las opciones de viaje

0. Terminar.

Introduce una opción: 3

Elige un lugar con playa

Nombre	Precio
P1	1000
P2	1300
P3	950
P4	1500

P2

Elige un lugar con sierra

Nombre	Precio
S1	1300
S2	960
S3	1110
S4	2130
S5	890
S6	1200

S3

1. Mostrar lugares con playa

2. Mostrar lugares con montaña

3. Introduce un nuevo viaje

4. Mostrar todas las opciones de viaje

0. Terminar.

Introduce una opción: 3

Elige un lugar con playa

Nombre	Precio
P1	1000
P2	1300
P3	950
P4	1500

P3

Elige un lugar con sierra

Nombre	Precio
S1	1300
S2	960
S3	1110
S4	2130
S5	890
S6	1200

S5

1. Mostrar lugares con playa

2. Mostrar lugares con montaña

3. Introduce un nuevo viaje

4. Mostrar todas las opciones de viaje

0. Terminar.

Introduce una opción: 4

Lista de opciones:

Viaje: P3 y S5, precio: 1840

Viaje: P2 y S3, precio: 2410

Presione una tecla para continuar

. . .

Observaciones

El programa debe liberar la memoria dinámica utilizada. Se recuerda que el comando que se añade al inicio de la función main y muestra información de la memoria no liberada es:

```
_CrtSetDbgFlag(_CRTDBG_ALLOC_MEM_DF | _CRTDBG_LEAK_CHECK_DF);
```

Además, se debe añadir el archivo “checkML.h” a los archivos cpp. Este archivo contiene:

```
#ifdef _DEBUG
#define _CRTDBG_MAP_ALLOC
#include <stdlib.h>
#include <crtdbg.h>
#ifdef DBG_NEW
#define DBG_NEW new ( _NORMAL_BLOCK , __FILE__ , __LINE__ )
#define new DBG_NEW
#endif
#endif
```

Entrega del examen:

1- Añade al inicio de tus archivos añade un comentario con tus datos:

```
/*
```

Apellidos:

Nombre:

Puesto:

```
*/
```

Entrega únicamente el código del programa (sólo .cpp y .h)

