Tema 2. Lliuraments

Exercici 2.4 (nivell mig). EXERCICI AVALUABLE

Volem crear un conjunt de classes per poder guardar tots els llibres d'una biblioteca i poder gestionar els préstecs que es fan dels llibres.

1. Declarar i implementar una **classe `Llibre`** per guardar les dades bàsiques d'un llibre: títol, autor i nº d'exemplars que hi ha a la biblioteca del llibre. Com a mètodes la classe ha de tenir:
   - Un constructor amb tres paràmetres (títol, autor i nº exemplars) que permeti inicialitzar les dades bàsiques del llibre.
   - Mètodes per recuperar el valor de cadascun dels atributs de la classe: `getTitol`, `getAutor` i `getNExemplars`.
   - Mètodes per modificar el valor de cadascun dels atributs de la classe: `setTitol`, `setAutor` i `setNExemplars`.
2. Declarar i implementar una **classe `Exemplar`** que permeti guardar les dades de cadascun dels exemplars dels llibres de la biblioteca. Aquesta classe haurà de tenir atributs per guardar un codi numèric que identifiqui l'exemplar, si està prestat o no, la data de l'últim préstec i la data de retorn del préstec. Com a mètodes la classe ha de tenir els següents:
   - Mètodes `getCodi` i `setCodi` per recuperar i modificar el valor del codi de l'exemplar.
   - Un mètode `prestar` per registrar el préstec de l'exemplar. Ha de rebre com a paràmetre un `string` que indiqui la data en què es fa el préstec que ha de quedar guardada a l'atribut corresponent. Si l'exemplar ja estava prestat el mètode retornarà fals, si no retornarà cert.
   - Un mètode `retornar` per registrar que es retorna un exemplar prestat a la biblioteca. Ha de rebre com a paràmetre un `string` que indiqui la data en què es retorna l'exemplar, que ha de quedar guardada a l'atribut corresponent. Si l'exemplar no estava prestat el mètode retornarà fals, si no retornarà cert.
   - Un mètode estaPrestat que retorni si l'exemplar està prestat o no.
3. **Modificar la classe `Llibre`** per incloure un array dinàmic que guardi les dades de tots els exemplars del llibre, utilitzant la classe `Exemplar` que heu creat a l'apartat anterior. A nivell de mètodes, feu les següents modificacions:
   - Modificar la classe Llibre per guardar els exemplars utilitzant un array dinàmic. Feu tots els canvis necessaris tant a la declaració de la classe, com al constructor i destructor de la classe. També haureu de modificar el mètode setNExemplars perquè inicialitzi correctament l'array dinàmic al fixar el valor del nº d'exemplars.
   - Modificar el constructor de la classe `Llibre` perquè inicialitzi el codi de tots els exemplars del llibre. Podeu suposar que els codis dels exemplars és un s'assignen correlativament des de 1 fins al nº màxim d'exemplars del llibre. Assegureu-vos que el constructor de la classe `Exemplar` inicialitza l'exemplar com a no prestat.
   - Afegir un mètode `prestar`, per poder fer el préstec del llibre, que tingui aquesta capçalera:
     ```
     bool prestar(const string &data, int &codiExemplar);
     ```
     Ha de mirar si hi ha algun exemplar del llibre que no estigui prestat. Si el troba, marcarà l'exemplar com a prestat, especificant com a data de préstec la data que es

passa com a paràmetre i retornarà el codi de l'exemplar prestat per referència al paràmetre `codiExemplar`. El mètode retornarà cert si es pot fer el préstec i fals si no queda cap exemplar lliure per prestar.

- Afegir un mètode `retornar`, per poder indicar que es retorna un exemplar prestat del llibre, que tingui aquesta capçalera:
  ```
  bool retornar(const string &data, int codiExemplar);
  ```
  Ha de buscar l'exemplar que tingui el codi passat per paràmetre, i si l'exemplar està prestat l'ha de marcar com a retornat indicant la data que es passa per paràmetre com a data de retorn. El mètode retornarà cert si l'exemplar existeix i estava prestat i fals en cas contrari.

4. Declarar i implementar una classe `Biblioteca` per guardar les dades de tots els llibres de la biblioteca utilitzant un array dinàmic. La classe ha de tenir els mètodes següents:
   - Un mètode per llegir les dades de tots els llibres de la biblioteca d'un fitxer, amb aquesta capçalera:
     ```
     void llegirLlibres(const char *nomFitxer);
     ```
     El fitxer tindrà a la primera línia el nº total de llibres que hi ha a la biblioteca. Aquest número l'haureu de fer servir per inicialitzar correctament l'array dinàmic. A continuació del nº de llibres, una línia per cada llibre. A cada línia hi haurà el títol, l'autor i el nº d'exemplars del llibre separats per un espai en blanc.
   - Un mètode per buscar un llibre a la biblioteca, amb la capçalera següent:
     ```
     bool cercarLlibre(const string &titol, Llibre &llibre);
     ```
     El mètode ha de retornar cert si existeix algun llibre que tingui el títol que es passar com a paràmetre i fals en cas contrari. Si existeix el llibre s'ha de retornar utilitzant el paràmetre per referència `llibre`. Penseu que per poder retornar el Llibre necessitareu tenir definit l'operador d'assignació **a la classe Llibre operator=.**

Tingueu en compte:

- A Caronte trobareu un programa principal per validar que la implementació que heu fet de les diferents classes és correcta. També hi trobareu un fitxer amb les dades dels llibres per inicialitzar la biblioteca.
- Si teniu algun dubte de quin tipus de dades assignar als atributs o de com s'utilitzen els mètodes de les classes, fixeu-vos en el programa principal que us donem per veure com fa servir les classes que us demanem.

Exercici 2.5 (nivell mig) – EXERCICI NO AVALUABLE

Volem fer part d'un programa que permeti gestionar un sistema de venda d'entrades per espectacles teatrals.
Per guardar les dades de cada espectacle crearem una classe `Espectacle`. De cada espectacle s'ha de guardar un conjunt de dades bàsiques (nom, teatre on es representa, dia i hora de representació, preu de les entrades, nº total d'entrades que es poden vendre i nº d'entrades que queden lliures).

1. Declareu i implementeu la **classe `Espectacle`**:
   - Declareu atributs per totes les dades bàsiques de l'espectacle indicades anteriorment.
   - Declareu i implementeu mètodes `getXXX` i `setXXX` per cadascun dels atributs de la classe.

A més a més volem guardar per cada espectacle les dades de les persones que fan les reserves de les entrades. Crearem una nova classe Espectador que contingui el nom de la persona i nº d'entrades que ha reservat.

2. Declareu i implementeu la **classe `Espectador`**:
   - Declareu atributs per guardar les dades bàsiques de cada espectador indicades anteriorment.
   - Declareu i implementeu mètodes `getXXX` i `setXXX` per cadascun dels atributs de la classe.
3. Afegiu a la **classe `Espectacle`** un array dinàmic amb les dades de tots els espectadors que han fet alguna reserva per l'espectacle.
   - Afegiu a la classe `Espectacle` els atributs necessaris per incloure l'array dinàmic d'espectadors.
   - L'array dinàmic s'ha de crear amb tants elements com entrades totals de l'espectacle. Modifiqueu el mètode que fixa el valor del nº total d'entrades per inicialitzar l'array dinàmic.
   - Modifiqueu el destructor de la classe `Espectacle` per eliminar correctament l'array dinàmic.
   - Afegiu un mètode a la classe `Espectacle` per afegir les dades (nom i nº d'entrades) d'un nou espectador.

Finalment tindrem una **classe `LlistatEspectacles`** per poder guardar les dades de tots els espectacles. Aquesta classe ha de guardar un array dinàmic amb les dades de tots els espectacles i els atributs necessaris per controlar el nº d'espectacles que s'han donat d'alta.

4. Declareu i implementeu la classe `LlistatEspectacles` amb els mètodes següents:

- Un constructor que rebi com a paràmetre el nº total d'espectacles que es volen guardar i inicialitzi l'array dinàmic que guarda les dades dels espectacles.
- Un destructor que alliberi l'array d'espectacles.
- `bool afegeixEspectacle(string nom, string teatre, string dia, string hora, int entradesTotals, float preu);`

  Afegeix un nou espectacle inicialitzat amb les dades que es passen com a paràmetre. Ha de comprovar que no es superi el nº màxim d'espectacles permesos. Si ja s'ha arribat al límit d'espectacles retorna fals, si no, afegeix l'espectacle i retorna cert.

- `int reservaEspectacle(string nom, string dia, int nEntrades, string espectador, float& preu);`

  Aquest mètode ha de permetre fer reserva d'entrades per un determinat espectacle. Rep com a paràmetres el nom de l'espectacle pel que es volen buscar les entrades, el dia en què s'hi vol anar, quantes entrades es volen i el nom de l'espectador que fa la reserva. El mètode ha de buscar l'espectacle que coincideixi amb el nom de l'espectacle i el dia que es passen com a paràmetres. Si queden suficients entrades lliures per aquella representació, es farà la reserva d'entrades disminuint el nº d'entrades lliures de l'espectacle i afegint les dades de l'espectador a l'espectacle. En aquest cas, el mètode retornarà un 0 com a valor de retorn i al paràmetre per referència `preu`, el valor total a pagar per les entrades. Si no queden suficients entrades per fer la reserva retornarà un -1 i si la representació no existeix retornarà -2.

- `bool guardaEspectadors(string nom, string dia, const char *nomFitxer);`

  Aquest mètode ha guardar en el fitxer que es passa com a paràmetre les dades (nom i nº d'entrades) de tots els espectadors que han fet reserva d'entrades per l'espectacle amb el nom i el dia que es passen com a paràmetres. En el fitxer hi ha d'haver una línia per cada espectador, amb el nom i el nº d'entrades separats per un espai en blanc. Si l'espectacle existeix i es poden guardar les dades al fitxer el mètode retorna cert, si no retorna fals.

  Per implementar aquest mètode segurament necessitareu algun mètode a la classe `Espectacle` que permeti recuperar les dades dels espectadors.

Tingueu en compte:

- A Caronte trobareu un programa principal per validar que la implementació que heu fet de les diferents classes és correcta.

Exercici 2.6 (nivell mig). EXERCICI AVALUABLE

Volem implementar una classe `Vector` que permeti fer una gestió intel·ligent d'un array dinàmic. Un objecte de la classe `Vector` ha de poder guardar un array dinàmic afegint alguns controls i precaucions en l'ús dels índexs per accedir als elements de l'array i en la gestió de la memòria dinàmica.
El vector serà de Punts.
Com a atributs, la classe vector haurà de tenir un array dinàmic de Punt i a més a més, tots els altres atributs que facin falta per poder implementar els mètodes que us indiquem a continuació:

1. Un constructor per defecte que inicialitza tots els atributs de forma segura. Inicialment l'array ha de tenir longitud 0.
2. El destructor, per alliberar la memòria de l'array.
3. Un mètode que permeti canviar la mida de l'array dinàmic. S'haurà de reservar memòria per un nou array amb la nova mida, fer la còpia de tots els elements de l'array antic al nou i alliberar l'array original. Si el nou array és més petit que l'original, es perden tots els elements que no caben en el nou array. Si el nou array és més gran, les posicions noves queden buides.

   ```
   void redimensionar(int mida);
   ```

4. Un mètode per recuperar la longitud de l'array:

   ```
   int longitud();
   ```

5. L'operador [] per poder accedir a un element qualsevol de l'array. La definició de l'operador la teniu a continuación:

   ```
   Punt& operator[](int nIndex);
   ```

   L'operador ens retornarà el valor que està a la posición indicada per nIndex. Però si s'intenta accedir a un element fora dels índex vàlids de l'array s'ha de mostrar per pantalla un missatge d'error i retornar el valor per defecte del Punt (obtingut a partir del constructor per defecte). Tingueu en compte que com que s'ha de retornar per referència, el valor per defecte haurà de correspondre al valor d'una variable

6. Un constructor de còpia. Tingueu en compte que copiar un objecte de la classe vector implica també copiar tots els elements del vector original al nou.

7. L'operador d'assignació, que ha de tenir en compte el mateix que s'ha comentat pel constructor de còpia.

   ```
   Vector& operator = (const Vector& v);
   ```

Per avaluar aquest exercici us donarem un programa principal i la classe `Punt` que us donem implementada i que guarda simplement les coordenades x i y d'un punt:

```
class Punt
{
public:
	Punt() { m_x = 0; m_y = 0; };
	Punt(int x, int y) { m_x = x; m_y = y; };
	Punt(const Punt& p) { m_x = p.m_x; m_y = p.m_y; };
	void setX(float x) { m_x = x; };
	void setY(float y) { m_y = y; };
	float getX()const { return m_x; };
	float getY()const { return m_y; };
	bool operator==(const Punt& p)
		{ return ((m_y == p.m_y) && (m_x == p.m_x)); };
private:
	float m_x;
	float m_y;
};
```

Tingueu en compte que per poder utilitzar el Vector amb aquesta classe Punt, és necessari que la classe `Punt` tingui implementats l'operador de comparació `==`. Aquest operador us el donem fet.