Tema 1 - Exercici Recuperació 2

En bastants dels problemes que hem anat fent fins ara hem tingut la necessitat de guardar llistes d'objectes de diferents tipus. Per exemple, una llista de punts al polígon, una llista d'estudiants d'una assignatura, una llista d'espectacles, etc. En tots aquests casos, tant els atributs com els mètodes necessaris per gestionar la llista són molt similars, independentment del tipus de l'objecte que s'hagi de guardar (punts, estudiants, espectacles, etc.)

En aquest exercici volem crear una classe `Llista` que sigui el més genèrica possible per poder guardar i manipular llistes d'objectes de diferents tipus amb poques modificacions. En aquest exercici la utilitzarem per guardar objectes del tipus `Estudiant` que hem utilitzat a algunes de les sessions de classe, però amb petites modificacions podria fer-se servir per guardar objectes de qualsevol altre tipus.

La classe `Llista` ha de permetre guardar un conjunt qualsevol d'estudiants (amb un màxim de 100 elements dins de la llista) i ha de permetre fer operacions per afegir, eliminar I recuperar elements de la llista, ordenar-la i llegir i escriure els elements de la llista a un fitxer.

1. Definiu els atributs necessaris per poder guardar tots els estudiants a la llista. Tingueu present que, tot I que el màxim d'elements que podeu guardar és 100, heu de poder tenir qualsevol nombre d'elements a la llista I per tant, heu de poder saber quants elements teniu en cada moment.

Tema 1 - Exercici Recuperació 1

2. Implementeu els mètodes següents:
   - Un constructor per defecte per inicialitzar la llista a la llista buïda sense cap element.
   - Un constructor que inicialitzi la llista amb els valors d'un array que es passa com a paràmetre (també es passa com a paràmetre el nº d'elements de l'array).
   - Un mètode per obtenir el nombre d'elements de la llista:

     ```
     int getNumElements();
     ```

   - Implementeu un mètode que retorni si un valor que es passa com a paràmetre està dins de la llista o no.

     ```
     bool pertany(const Estudiant& element);
     ```

     Sobrecarregueu l'operador == de la classe `Estudiant` de forma que retorni que dos objectes de la classe `Estudiant` són iguals si tenen el mateix nom i el mateix niu.
   - Implementeu un mètode que retorni la posició que ocupa dins de la llista l'estudiant que té el niu que es passa com a paràmetre.

     ```
     int cerca(const string& niu) const;
     ```

     Si l'estudiant no està dins de la llista retorna -1.

Tema 1 - Exercici Recuperació 1

2. Implementeu els mètodes següents:

   - Implementeu un mètode que retorni el valor que ocupa una posició determinada dins de la llista d'elements.

     ```
     Estudiant getElement(int posicio);
     ```

     Si la posició que es passa com a paràmetre està fora dels límits vàlids de la llista (des de 0 fins al nº d'elements – 1), s'ha de retornar un estudiant amb el nom i el niu inicialitzats amb la cadena buida.

   - Implementeu un mètode que modifiqui el valor d'una posició determinada dins de la llista d'elements.

     ```
     void setElement(int posicio, const Estudiant& estudiant)
     ```

     Si la posició que es passa com a paràmetre està fora dels límits vàlids de la llista (des de 0 fins al nº d'elements – 1), no s'ha de fer res.

   - Un mètode que permeti afegir un element a la llista en una posició determinada.

     ```
     bool insereix(const Estudiant& pt, int posicio);
     ```

     S'haurà de posar el nou element a la posició de la llista que s'indica com a paràmetre, desplaçant tots els elements posteriors una posició a la dreta. Els valors vàlids del paràmetre `posicio` són des del 0 fins al nº d'elements actuals del vector (en aquest cas s'afegeix al final de tot del vector, sense desplaçar cap element). Si es passa com a paràmetre de posicio un valor no vàlid el mètode retornarà `false` i `true` si el valor és vàlid. També retornarà `false` si la llista ja està plena i no hi caben més elements.

Tema 1 - Exercici Recuperació 1

2. Implementeu els mètodes següents:
   - Un mètode que permeti eliminar un element d'una posició determinada de la llista.

     ```
     bool elimina(int posicio);
     ```

     Eliminar un element de la llista implicarà decrementar el nº d'elements i desplaçar tots els elements posteriors una posició a l'esquerra. Els valors vàlids del paràmetre `posicio` són des del 0 fins al nº d'elements actuals de la llista – 1 (en aquest cas s'elimina l'últim element de la llista, sense desplaçar cap element). Si es passa com a paràmetre de posicio un valor no vàlid el mètode retornarà `false` i `true` si el valor és vàlid.
   - Sobrecarregueu l'operador + perquè afegeixi una data al final de la llista, com a últim element. Si la llista ja conté el màxim de 100 elements no l'ha d'afegir i en aquest cas, retornarà `false` com a resultat. Si el pot afegir retornarà `true`.
   - Sobrecarregueu l'operador - perquè elimini de la llista l'estudiant que es passa com a paràmetre. Si l'estudiant no està dins de la llista no s'ha de fer res i s'ha de retornar `false`. Si està dins de la llista i es pot eliminar s'ha de retornar `true`. Tingueu en compte que quan s'elimina un element de la llista és possible que s'hagin de desplaçar els altres elements de la llista per mantenir la coherència de la llista, igual que hem explicat abans pel mètode `elimina`.

Tema 1 - Exercici Recuperació 1

2. Implementeu els mètodes següents:

   - Sobrecarregueu l'operador de sortida << per mostrar la llista per pantalla. Cada element de la llista s'ha de mostrar en una línia diferent. Per mostrar cadascun dels estudiants per pantalla, sobrecarregueu l'operador << de la classe `Estudiant` perquè mostri el nom, el niu i la nota mitjana, separat per espais en blanc: `NOM NIU NOTA_MITJANA`.

   - Torneu a sobrecarregar l'operador << per guardar tots els elements de la llista en un fitxer, seguint el mateix format que hem explicat a l'apartat anterior per la sortida en pantalla. Haureu de sobrecarregar també l'operador << de la classe `Estudiant` perquè guardi les dades d'un estudiant al fitxer en el format especificat.

   - Sobrecarregueu l'operador >> per llegir els elements de la llista d'un fitxer que tingui una línia per cada estudiant, amb la informació de l'estudiant en aquest format:

     ```
     NOM NIU N_NOTES NOTA_1 NOTA_2 ... NOTA_N
     ```

     `N_NOTES` indica el nº de notes que s'han de llegir per aquell estudiant. Haureu de sobrecarregar també l'operador >> de la classe `Estudiant` perquè llegeixi del fitxer la informació d'un estudiant en el format indicat.