Control PRO2 - Torn 1 (Primavera 2017) P61562_ca

Hem decidit estendre la classe `Cjt_estudiants` que heu vist al laboratori amb una nova funcionalitat que assigna automàticament un nombre limitat de becas als estudiants aprovats amb millors notes, i en cas d'empat als estudiants aprovats amb millors notes en ordre descendent per DNI. El nombre de becas assignades `n_bec` mai serà superior al nombre de becas disponibles `MAX_BEC`, ni al nombre d'estudiants aprovats `na`, i si `na<MAX_BEC` el nombre de becas asignades serà igual a `na`.
Per implementar eficientment aquesta funcionalitat

- hem modificat la representació i l'invariant de la classe de la manera descrita al archiu `Cjt_estudiants.hh`.
- hem incorporat un nou atribut `static` i constant `MAX_BEC` que especifica el màxim nombre de becas que es pot concedir.
- també hem incorporat un nou atribut `n_bec` que conté el nombre de becas concedides a estudiants de vest[0...nest-1] de les `MAX_BEC` disponibles.
- així mateix hem incorporat un nou atribut `i_max_no_becat` que conté la posició a `vest` de l'estudiant no becat i aprovat amb millor nota, i en cas d'empat la posició de l'estudiant amb DNI més gran dels estudiants no becats, aprovats i amb millor nota. Si tots els estudiants aprovats tenen beca o cap estudiant està aprovat el valor de `i_max_no_becat` és -1.
- finalment hem afegit dos mètodes: `recalcular_pos_max_no_becat` modificador privat i `pos_max_no_becat` consultor pùblic.

Tenint això en compte heu d'implementar eficientment el següent mètode privat:

```
void recalcular_pos_max_no_becat();
/* Pre: cert */
/* Post: i_max_no_becat és la posició a vest de l'estudiant no becat,
  i aprovat amb millor nota, i en cas d'empat és la posició de
  l'estudiant amb DNI més gran dels estudiants no becats, aprovats i
  amb millor nota. Si tots els estudiants aprovats tenen  beca o cap
  estudiant està aprovat i_max_no_becat = -1 */
```

i el següent mètode públic:

```
void esborrar_estudiant(int x, bool& trobat);
/* Pre: cert  */
/* Post: Si el paràmetre implícit original contenia un estudiant amb
     DNI = x, trobat és true i el p.i. conté els mateixos estudiants
     que l'original menys l'estudiant amb DNI = x; altrament, trobat
     és false i el p.i. és igual a l'original. */
```

Observació

Heu de lliurar un fitxer `solucio.cc` amb una implementació eficient de les operacions `recalcular_pos_max_no_becat` i `esborrar_estudiant` que ha de tenir el següent format:

```
#include "Cjt_estudiants.hh"

void Cjt_estudiants::recalcular_pos_max_no_becat() {
  ... // codi de la implementació
}

void Cjt_estudiants::esborrar_estudiant(int x, bool& trobat) {
  ... // codi de la implementació
}
```

Copieu aquesta plantilla en el vostre `solucio.cc` i completeu-la. El vostre `solucio.cc` no pot contenir la implementació d'altres operacions de la classe.
A l'apartat *Public files* del Jutge us proveïm amb material addicional comprimit en un fitxer .tar. Podeu descomprimir aquest fitxer amb la comanda

```
tar -xvf nom_fitxer.tar
```

Aquest material addicional consisteix en els següents fitxers:

- `Cjt_estudiants.hh`: l'especificació Pre/Post de totes les operacions públiques i privades d'aquesta nova versió de la classe `Cjt_estudiants`, així como la definició dels camps privats. Fixeu-vos que **hem afegit tres atributs** `n_bec`, `i_max_no_becat` i `MAX_BEC`, i que **hem modificat l'invariant** de la representació de `Cjt_estudiants`. **És molt important que la implementació de les operacions que us hem encarregat tingui en compte i preservi l'invariant de la representació**. Fixeu-vos també que hi ha dos operacions noves: el modificador privat `recalcular_pos_max_no_becar` i el consultor pùblic `pos_max_no_becat`.

- `Cjt_estudiants.cc`: la implementació de totes de les operacions de la nova versió de la classe `Cjt_estudiants` tret de les operacions que us demanem.

- `Estudiant.hh`: l'especificació de la classe `Estudiant` i la definició dels seus atributs. Les novetats que presenta son un atribut `amb_beca` que indica si s'ha concedit una beca al estudiant paràmetre implícit, un atribut `MIN_APR` constant i `static` que especifica la nota mínima per aprovar, i un métode públic `major_nota_dni` que permet comparar dos estudiants per nota i en cas d'empat per DNI.

- `Estudiant.cc`: la implementació dels mètodes de la classe `Estudiant`.

- `pro2.cc`: un programa principal que podeu fer servir per provar els mètodes públics d'aquesta versió de la classe `Cjt_estudiants`.

- `llegeixme.txt`: instruccions per a generar l'executable del programa `pro2` i provar-lo.

Valorarem positivament que la solució no contingui instruccions (especialment bucles o crides a operacions costoses) ni objectes (especialment vectors o conjunts) innecessaris, que no faci recorreguts quan hauria de fer cercas, i que usi correctament les operacions més eficients de la classe sempre que sigui possible. No es pot emprar cap estructura de dades que no hagi aparegut a les sessions 1-4 de laboratori.

Quan feu els enviaments el Jutge us indicarà quants jocs de proves passeu i de quin tipus (public o privat). El joc de proves anomenat `public` correspon al fitxers `entrada.txt` i `sortida_correcta.txt` de l'apartat *Public files*.

Informació del problema

Autor : Professors de PRO2
Generació : 2017-02-28 09:24:53