
Consulta per intervals de notes dels estudiants d'un conjuntU48948_ca

Hem decidit estendre la classe `Cjt_estudiants` que heu vist al laboratori amb una nova funcionalitat que permeti escriure els estudiants del paràmetre implícit amb nota compresa dins del interval tancat $[nota_inf, nota_sup]$ verificant-se $0 \leq nota_inf \leq nota_sup \leq nota_maxima()$. Els estudiants s'escriuen en ordre ascendent per nota i, en cas d'empat, en ordre ascendent per DNI.

Per implementar eficientment aquesta operació

- hem modificat la representació i l'invariant de la classe de manera que els estudiants sense nota del atribut "vest" precedeixen als estudiants amb nota i estan ordenats creixentment per DNI, i els estudiants amb nota del atribut "vest" estan ordenats creixentment per nota i, en cas d'empat, creixentment per DNI.
- hem incorporat un nou atribut `primer_est_amb_not`a que conté la posició del primer estudiant amb nota.
- hem afegit tres operacions privades: `cerca_lineal_per_dni`, `cerca_dicot_per_dni` i `cerca_dicot_per_not`a.

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

```
int posicio_not(a(double not_a_b) const;
/* Pre: 0 <= not_a_b <= not_a_maxima() */
/* Post: el resultat és la posició del primer estudiant amb not_a major
o igual que not_a_b a vest[0...nest-1]. Si no hi ha estudiants amb
not_a major o igual que not_a_b, el resultat és nest. */
```

i els següents mètodes públics:

```
void afegir_estudiant(const Estudiant &est, bool &trobat);
/* Pre: el paràmetre implícit no està ple */
/* Post: trobat indica si el p.i. original conté un estudiant amb
el dni d'est; si trobat és cert no es modifica el p.i., i si
trobat és fals s'ha afegit l'estudiant est al paràmetre implícit */

void escriure_i(double not_a_i, double not_a_s) const;
/* Pre: 0 <= not_a_i <= not_a_s <= not_a_maxima() */
/* Post: s'han escrit pel canal estàndard de sortida els estudiants
del paràmetre implícit amb not_a dins del interval tancat
[not_a_i, not_a_s]. Els estudiants s'escriuen en ordre ascendent
per not_a i en cas d'empat en ordre ascendent per DNI */
```

Observació

Heu de lliurar un fitxer `solucio.cc` amb una implementació eficient de les operacions `posicio_not`, `afegir_estudiant` i `escriure_i` que ha de tenir el següent format:

```
#include "Cjt_estudiants.hh"

int Cjt_estudiants::posicio_not(a(double not_a_b) const {
    ... // codi de la implementació
}
```

```

void Cjt_estudiants::afegir_estudiant(const Estudiant &est, bool &trobat)
{
    ... // codi de la implementació
}

void Cjt_estudiants::escriure_i(double nota_i, double nota_s) const
{
    ... // 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 un nou atribut** `primer_est_amb_nota` i **hem modificat l'invariant** de la representació de la classe `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 tres operacions privades: `cerca_lineal_per_dni`, `cerca_dicot_per_dni` i `cerca_dicot_per_nota_dni` que podeu d'usar per fer implementacions eficients.
- `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. La novetat que presenta és una operació, `comp_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 les operacions públiques d'aquesta versió de la classe `Cjt_estudiants`.

Valorarem positivament que la solució no contingui instruccions (especialment bucles o crides a operacions costoses) ni objectes (especialment vectors) innecessaris, que no faci recorreguts quan hauria de fer cerques, i que usi correctament les operacions més eficients de la classe sempre que sigui possible. No es pot emprar cap estructura de dades addicional ni implementar més operacions noves.

Important: lliurar repetidament solucions que no funcionin amb el sample públic és una pèrdua de temps. Al fitxer `.tar` del *Public files* teniu tot el material necessari per provar el vostre codi abans de lliurar-lo.

Informació del problema

Autor :

Generació : 2024-11-11 23:09:59

© Jutge.org, 2006–2024.

<https://jutge.org>