Control PRO2 - Turno 2 (Primavera 2017) X38181_es

Hemos decido extender la clase `Cjt_estudiants` que habéis visto en el laboratorio con una nueva funcionalidad que selecciona automáticamente los estudiantes que pueden hacer el curso de re-evaluación (es decir, que están admitidos) teniendo en cuenta que hay un número limitado de plazas disponibles `MAX_PLA` para este curso. Un estudiante *puede optar al curso de re-evaluación* si tiene nota, y su nota es mayor o igual a 4 y menor que 5. Las plazas del curso de re-evaluación se asignan por orden decreciente de nota entre los estudiantes que pueden optar al curso de re-evaluación, y en caso de empate por orden decreciente de DNI. Esto significa que, dados dos estudiantes e_1 y e_2 que pueden optar al curso de re-evaluación, e_1 tendría preferencia sobre e_2 para ser admitido al curso de re-evaluación si e_1 tiene mejor nota que e_2, o si e_1 y e_2 tienen la misma nota y el DNI de e_1 es mayor que el DNI de e_2.
Hemos añadido dos métodos públicos a la clase `Cjt_estudiants`: 1) `n_admesos`, que devuelve el número de estudiantes del conjunto que están admitidos al curso de re-evaluación; 2) `pos_min_admes`, que devuelve la posición del estudiante admitido al curso de re-evaluación con menor nota, y en caso de empate con menor nota y menor DNI. Si ningún estudiante del conjunto está admitido al curso de re-evaluación, `pos_min_admes` devuelve -1.
En todo momento, el número de estudiantes admitidos `n_admes` será menor o igual que el número de plazas disponibles `MAX_PLA`. Además `n_admes` nunca será superior al número de estudiantes que cumplen las condiciones para optar al curso de re-evaluación. Finalmente, si el número de estudiantes que pueden optar al curso de re-evaluación es mayor o igual que el número de plazas disponibles `MAX_PLA`, el número de estudiantes admitidos `n_admes` será igual a `MAX_PLA`.
Para implementar eficientemente esta funcionalidad hemos modificado la representación y la invariante de la clase `Estudiant` de la manera descrita en el archivo `Estudiant.hh`. En particular, representamos la información sobre si un estudiante está admitido al curso de re-evaluación o no en los objetos de la clase `Estudiant`, y no en los objetos de clase `Cjt_estudiants`. Esto quiere decir que para admitir al curso de re-evaluación al estudiante situado en la posición `pos` de `vest` hemos de utilizar la instrucción `vest[pos].modificar_reaval(true);` y similarmente para no admitirlo `vest[pos].modificar_reaval(false);`, comprobar si está admitido `vest[pos].admes_reaval()` o si cumple las condiciones para optar al curso de re-evaluación `vest[pos].cond_reaval()`. Leed con atención el archivo `Estudiant.hh`, especialmente las descripciones de los nuevos atributos, la invariante y las especificaciones de las operaciones nuevas. Las principales novedades de la clase `Estudiant` son:

- hemos incorporado un nuevo atributo `reaval` que permite representar información sobre si el estudiante parámetro implícito está admitido al curso de re-evaluación o no;
- hemos añadido el consultor `admes_reaval` que permite consultar si el estudiante parámetro implícito está admitido al curso de re-evaluación o no;
- hemos añadido el modificador `modificar_reaval` que permite modificar la información sobre si el estudiante parámetro implícito está admitido al curso de re-evaluación o no;

- hemos añadido el consultor `cond_reaval` que permite comprobar si el estudiante parámetro implícito cumple las condiciones para optar al curso de re-evaluación;
- hemos añadido el método `static` y público `major_nota_dni` que permite comparar dos estudiantes por nota y en caso de empate por DNI;

También hemos modificado la representación y la invariante de la clase `Cjt_estudiants` de la manera descrita en el archivo `Cjt_estudiants.hh`. La principal novedad es que almacenamos la posición del estudiante admitido al curso de re-evaluación con menor nota, y en caso de empate con menor nota y menor DNI, en el atributo `i_min_admes`. De este modo, cuando añadamos un estudiante a un conjunto donde el número de estudiantes admitidos al curso de re-evaluación es igual al número de plazas disponibles, podremos determinar fácilmente si debemos admitir al nuevo estudiante al curso de re-evaluación o no. Es decir, podremos comprobar si el nuevo estudiante cumple las condiciones para optar al curso de re-evaluación y si tiene mejor nota o igual nota y mayor DNI que el estudiante admitido al curso de re-evaluación con menor nota, o con menor nota y menor DNI, del conjunto original. Leed con atención el archivo `Cjt_estudiants.hh`, especialmente las descripciones de los nuevos atributos, la invariante y las especificaciones de las operaciones nuevas. Las principales novedades de la clase `Cjt_estudiants.hh` son:

- hemos incorporado un nuevo atributo `static` y constante `MAX_PLA` que especifica el número de plazas disponibles para el curso de re-evaluación;
- hemos incorporado un nuevo atributo `n_admes` que contiene el número de estudiantes del conjunto que están admitidos al curso de re-evaluación;
- hemos añadido un consultor `n_admesos` que permite consultar el valor del atributo `n_admes`, es decir, el número de estudiantes admitidos;
- hemos incorporado un nuevo atributo `i_min_admes` que contiene la posición en `vest` del estudiante admitido al curso de re-evaluación con menor preferencia. Si hay estudiantes admitidos al curso de re-evaluación, el atributo `i_min_admes` contiene la posición del estudiante admitido con menor nota y, en caso de empate con menor nota y menor DNI, y su valor está dentro del intervalo $0 \leq$ `i_min_admes` $<$ `nest`; en caso contrario, si ningún estudiante del conjunto está admitido al curso de re-evaluación, `i_min_admes` es igual a -1;
- hemos añadido el consultor público `pos_min_admes`, para consultar la posición del estudiante admitido al curso de re-evaluación con menor nota y, en caso de empate con menor nota y menor DNI. Si hay estudiantes admitidos al curso de re-evaluación en el conjunto, devuelve `i_min_admes+1`; y si no hay estudiantes admitidos al curso de re-evaluación en el conjunto, devuelve -1.
- y hemos añadido el modificador privado `recalcular_pos_min_admes` para recalcular el valor del atributo `i_min_admes` cuando sea necesario.

Teniendo esto en cuenta debéis implementar eficientemente el siguiente método privado sin utilizar la operación `sort` de la biblioteca `<algorithm>`:

```
void recalcular_pos_min_admes();
/* Pre: cierto */
/* Post: Si hay estudiantes admitidos al curso de re-evaluación en el
conjunto parámetro implícito, entonces el atributo i_min_admes contiene
la posición en vest[0...nest-1] del estudiante admitido con menor nota
y, en caso de empate del estudiante admitido con menor nota y menor DNI;
```

```
si no hay estudiantes admitidos al curso de re-evaluación en el p.i.,
el atributo i_min_admes es igual a -1. */
```

y el siguiente método público sin utilizar la operación `sort` de la biblioteca `<algorithm>`. Observad que cuando añadimos al conjunto un estudiante que cumple las condiciones para optar al curso de re-evaluación es posible que haya plazas disponibles para el curso de re-evaluación no asignadas a otros estudiantes del conjunto, en cuyo caso debemos admitirlo. Pero también es posible que todas las plazas disponibles esten asignadas a otros estudiantes, y en ese caso debemos determinar si el nuevo estudiante tiene preferencia sobre el estudiante admitido con menor nota, o con menor nota y menor DNI, del conjunto original. Ya que si no hay plazas disponibles para los dos, el que tenga mayor preferencia debe pasar a ser admitido y el de menor preferencia pasar a no ser admitido.

```
void afegir_estudiant(const Estudiant& est, bool& trobat);
/* Pre: El número de estudiantes del parámetro implícito es menor que
la medida máxima permitida. */
/* Post: Si el p.i. original no contenía ningún estudiante con el DNI
de est, trobat es false, se ha añadido el estudiante est al p.i., se
han actualizado los estudiantes admitidos al curso de re-evaluación en
el p.i. si ha sido necesario, y se ha actualizado la posición del
estudiante admitido con menor nota, y en caso de empate con menor nota
y menor DNI, si ha sido necesario. En caso contrario, trobat es true y
el p.i. es igual al original. */
```

Observación

Debéis entregar un fichero `solucio.cc` con una implementación eficiente de las operaciones `recalcular_pos_min_admes` y `afegir_estudiant` que ha de tener el siguiente formato:

```
#include "Cjt_estudiants.hh"

void Cjt_estudiants::recalcular_pos_min_admes() {
  ... // código de la implementación
}

void Cjt_estudiants::afegir_estudiant(const Estudiant& est, bool& trobat) {
  ... // código de la implementación
}
```

Copiad esta plantilla en vuestro `solucio.cc` y completadla. Vuestro `solucio.cc` no puede contener la implementación de otras operaciones de la clase.
En el apartado *Public files* del Jutge os proporcionamos material adicional comprimido en un fichero .tar. Podéis descomprimir este fichero con el comando

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

Este material adicional contiene los siguientes ficheros:

- `Cjt_estudiants.hh`: la especificación Pre/Post de todas las operaciones públicas y privadas de esta nueva versión de la clase `Cjt_estudiants`, así como la definición de los campos privados. Fijaos que **hemos añadido tres atributos** `n_admes`, `i_min_admes` y `MAX_PLA`, y que **hemos modificado la invariante** de la representación de `Cjt_estudiants`. **Es muy importante que la implementación de las operaciones que os hemos encargado tenga en cuenta y preserve la invariante de la representación**. Fijaos también en que hay operaciones nuevas: el modificador privado `recalcular_pos`

_min_admes, los consultores públicos n_admesos, places_disp, pos_min_admes, y el modificador público afegir_estudiant.

- Cjt_estudiants.cc: la implementación de todas las operaciones de la nueva versión de la clase Cjt_estudiants excepto la de las operaciones que os pedimos.
- Estudiant.hh: la especificación de la clase Estudiant y la definición de sus atributos. Las principales novedades que presenta son un atributo reaval que indica si el estudiante parámetro implícito está admitido al curso de re-evaluación, los consultores públicos admes_reaval, cond_reaval y major_nota_dni, y el modificador público modificar_reaval.
- Estudiant.cc: la implementació de los métodos de la clase Estudiant.
- pro2.cc: un programa principal que podéis usar para probar los métodos públicos de esta versión de la clase Cjt_estudiants.
- llegeixme.txt: instrucciones para generar el ejecutable del programa pro2 y probarlo.

Valoraremos positivamente que la solución no contenga instrucciones (especialmente bucles o llamadas a operaciones costosas) ni objetos (especialmente vectores o conjuntos) innecesarios, que no haga recorridos cuando debería hacer búsquedas, y que use correctamente las operaciones más eficientes de la clase siempre que sea posible. No se puede usar ninguna estructura de datos que no haya aparecido en las sesiones 1-4 de laboratorio.

La utilización de la operación sort de la biblioteca <algorithm> en el archivo solucio.cc comportará una calificación de 0 en la corrección manual del control.

Cuando hagáis envíos, el Jutge os indicará cuantos juegos de pruebas pasa vuestro programa y de qué tipo (público o privado). El juego de pruebas denominado público corresponde a los ficheros entrada.txt y sortida_correcta.txt del apartado *Public files*.

Información del problema

Autor : Professors de PRO2
Traductor : Professors de PRO2
Generación : 2017-03-19 14:17:48