Examen de la pràctica - Torn 1 (Primavera 2015) X32693_ca

Es vol dotar una versió simplificada de la pràctica amb una nova operació de consulta amb la qual es pugui obtenir una llista de les tasques que compleixin una certa expressió booleana d'etiquetes i, a més, tinguin lloc fins una hora determinada. El format de la nova consulta és una línia que comença per `?`, seguida d'una expressió booleana i finalitzada per una hora. Per exemple,

```
? ((#salut,#esport).(#lectura,#escriptura)) 20:00
```

és una consulta de les tasques que contenen `#salut` o `#esport` i `#lectura` o `#escriptura` i, addicionalment, estan fixades entre les 00:00 i les 20:00 hores.
La pràctica sobre la qual cal afegir aquesta operació conté una classe Agenda, que organitza tasques i etiquetes, i una classe Tasca, amb mètodes per llegir i escriure tasques, per consultar el títol o l'hora d'una tasca i per comprovar si una tasca conté una etiqueta determinada. Tant els atributs com l'especificació de les operacions públiques de les dues classes els trobareu a l'apartat *Public files* del Jutge.
Es demana un mètode per a la classe Agenda amb l'especificació següent:

```
void Agenda::tasques(string &expressio, const string &hora);
// Pre: "expressio" és una expressió booleana d'etiquetes amb format correcte,
//        "hora" és una hora correcta amb format hh:mm
// Post: s'han escrit pel canal estàndard de sortida totes les tasques del p.i. tals que:
//        - les etiquetes associades satisfan l'expressió booleana "expressio"
//        - la seva hora de realització és entre les 00:00 i "hora" (ambdues incloses)
//        ordenades cronològicament
```

Per implementar el mètode `tasques`, podeu fer-ho d'una de dues maneres possibles. La primera és recorrent les tasques una per una i comprovant si compleixen l'expressió boolena. En aquest cas, heu d'implementar el mètode públic següent:

```
bool Tasca::compleix_expressio(string &e);
// Pre: e és una expressió booleana d'etiquetes amb format correcte
// Post: cert si i només si les etiquetes del p.i. compleixen l'expressió boolena e
```

L'altra possibilitat és generar recursivament una llista (o vector) de les tasques que compleixen les subexpressions, i fer la reunió o la intersecció segons convingui. Si trieu aquesta solució, heu d'implementar el mètode privat següent:

```
vector<int> Agenda::compleixen_expressio(string &e);
// Pre: e és una expressió booleana d'etiquetes amb format correcte
// Post: retorna les posicions del vector t del p.i. de totes les tasques que compleixen
//         l'expressió booleana e, en ordre creixent
```

El mètode anterior pot fer ús dels mètodes privats `reunió` i `intersecció` que trobareu especificats en el fitxer `agenda.hh`.
Amb qualsevol dels dos mètodes anteriors, necessitareu extreure `substrings` a partir d'un `string` determinat. Podeu fer-ho mitjançant el mètode `substr()` de la classe `string`. Donat un `string x` i dos enters `i, j`, `x.substr(i)` retorna el sufix de `x` que comença en la posició `i`, mentre que `x.substr(i,j)` retorna el `substring` de mida `j` que comença en la posició `i` (les posicions comencen per la 0).

Observació

Heu de lliurar un fitxer `solucio.cc` amb una implementació eficient del mètode `tasques` (de la classe `Agenda`) i, a més, un dels dos següents:

- `compleix_expressio` (de la classe `Tasca`)
- `compleixen_expressio` (de la classe `Agenda`).

El format del fitxer `solucio.cc` ha de ser el següent:

```
#include "agenda.hh"

void Agenda::tasques(string &expressio, const string &hora)
{
      // codi de la implementació
}

/* Treieu les marques de comentari si implementeu aquest mètode
bool Tasca::compleix_expressio(string &e)
{
       // codi de la implementació
} */

/* Treieu les marques de comentari si implementeu aquest mètode
vector<int> Agenda::compleixen_expressio(string &e)
{
      // codi de la implementació
} */
```

Copieu aquesta plantilla en el vostre `solucio.cc` i completeu-la, deixant com a comentari o eliminant el mètode que no implementeu. El vostre `solucio.cc` no pot contenir la implementació d'altres operacions de les classes, però podeu afegir-hi funcions o procediments auxiliars externs a les classes.
Podeu descomprimir el fitxer .tar que us proporcionem a l'apartat *Public files* del Jutge mitjançant la comanda

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

Aquest material addicional consisteix en els fitxers següents:

- `llegeixme.txt`: instruccions per generar l'executable del programa `pro2.cc` i provar-lo.
- `pro2.cc`: un programa principal que podeu fer servir per provar el funcionament dels mètodes que heu d'implementar.
- `agenda.hh`: l'especificació de totes les operacions públiques i privades i la definició dels atributs privats. Fixeu-vos que l'atribut `t` conté les tasques ordenades cronològicament i l'atribut `m` conté un `map` que associa a cada etiqueta (la clau) un vector amb les seves tasques, que s'identifiquen mitjançant les seves posicions en `t`.
- `agenda.cc`: implementació de les operacions de la classe Agenda que no es demanen en la solució.
- `tasca.hh`: l'especificació de totes les operacions públiques i privades i la definició dels atributs privats.

- `tasca.cc`: implementació de les operacions de la classe Tasca que no es demanen en la solució.
- Els fitxers que ja coneixeu de la pràctica `comanda.hh`, `comanda.cc`, `token.hh` i `token.cc`.
- Un fitxer `Makefile` per generar l'executable tal com es descriu a `llegeixme.txt`.
- `enunciat.pdf`: l'enunciat de la pràctica.
- `vectors.pdf`: introducció a l'ús del `push_back` en vectors.
- `diccionaris.pdf`: introducció als diccionaris (`maps`, en `C++`).

Informació del problema

Autoria: Professors de PRO2

Generació: 2026-01-25T15:12:07.186Z

© *Jutge.org*, 2006–2026.
https://jutge.org