Classe conjunt de parelles d'enters — X58659_ca

Cal implementar la següent classe *cj_2enters* que ens permet representar i manipular conjunts de parelles d'enters.
Dins del conjunt no importa l'ordre de les parelles i no poden haver-hi parelles repetides. Dins de cada parella d'enters importa l'ordre dels dos enters i es poden repetir els dos enters. Per exemple (els elements del conjunt estan separats amb espais i la parella d'enters amb una coma):

- Els conjunts [1,1 3,1 1,2] i [3,1 1,2 1,1] són el mateix
- Els conjunts [1,2] i [2,1] són diferents

```
class cj_2enters {
public:

  // Constructora per defecte. Crea un conjunt buit.
  cj_2enters ();

  // Les tres grans: Constructora per còpia, destructora, operador d'assignació
  cj_2enters (const cj_2enters &cj);
  ~cj_2enters ();
  cj_2enters & operator=(const cj_2enters &cj);

  // Insereix la parella d'enters pe en el conjunt. No fa res si pe ja pertanyia al conjunt.
  void insereix (pair<int, int> pe);

  // Unió, intersecció i diferència de conjunts. Operen modificant el conjunt sobre el que
  // s'aplica el mètode, usant el segon conjunt com argument. P.e.: a.restar(b) fa que el
  // nou valor d'a sigui A - B, on A i B són els valors originals dels objectes a i b.
  void unir(const cj_2enters & B);
  void intersectar (const cj_2enters & B);
  void restar (const cj_2enters & B);

  // Unió, intersecció i diferència de conjunts. Operen creant un nou conjunt sense
  // modificar el conjunt sobre el que s'aplica el mètode. La suma de conjunts correspon
  // a la unió, la resta a la diferència i el producte a la intersecció.
  cj_2enters operator+(const cj_2enters& B) const;
  cj_2enters operator*(const cj_2enters& B) const;
  cj_2enters operator-(const cj_2enters& B) const;

  // Retorna cert si i només si pe pertany al conjunt.
  bool conte(pair<int, int> pe) const;

  // Retornen els elements màxim i mínim del conjunt, respectivament.
  // El seu comportament no està definit si el conjunt és buit.
  // pe1 és major que pe2 si el 1er enter de pe1 és major que el 1er enter de pe2. En cas
```

```
  // que siguin iguals, si el 2on enter de pe1 és major que el 2on enter de pe2.
  pair<int, int> max() const;
  pair<int, int> min() const;

  // Retorna el nombre d'elements (la cardinalitat) del conjunt.
  int card() const;

  // Operadors relacionals. == retorna cert si i només si els dos conjunts
  // (el paràmetre implícit i B) contenen els mateixos elements;
  // != retorna cert si i només si els conjunts són diferents.
  bool operator==(const cj_2enters& B) const;
  bool operator!=(const cj_2enters& B) const;

  // Imprimeix el conjunt de parelles d'enters, ordenats en ordre ascendent, sobre
  // el canal de sortida os; el format és [pe1 pe2 ... pen], és a dir, amb
  // espais entre els elements i tancant la seqüència amb corxets.
  // Els dos enters de la parella d'enters estan separats amb una coma.
  void print(ostream& os) const;

private:
    // Cal definir els atributs i mètodes privats dins del fitxer .rep
    #include "conjunt_2enters.rep"
};
```

Bàsicament el que cal fer és:

1. Trobar una representació adequada pels objectes de la classe i escriure els atributs necessaris en la part *private* de la classe (dins del fitxer *conjunt_2enters.rep*). També es poden especificar mètodes privats addicionals.

2. Implementar tots els mètodes de la classe els quals manipularan la representació anterior dins del fitxer *conjunt_2enters.cpp*.

Cal enviar la solució (els fitxers *conjunt_2enters.rep* i *conjunt_2enters.cpp*) comprimida en un sol fitxer *.tar*
Per testejar la classe disposes d'un programa principal que processa blocs que contenen dos conjunts A i B i vàries comandes que els manipulen.

Entrada

L'entrada conté varis blocs separats per línies amb 10 guions (————). Cada bloc consisteix en dues seqüències d'enters, una per línia, cadascuna d'elles són els elements que tindran originalment el conjunt A i el conjunt B. A continuació segueixen vàries comandes, una per línea, amb el següent format:

- insereix cjt1 e1 e2
- conte cjt1 e1 e2
- max cjt1
- min cjt1

- card cjt1
- unir cjt1 cjt2
- intersectar cjt1 cjt2
- restar cjt1 cjt2
- + cjt1 cjt2
- * cjt1 cjt2
- - cjt1 cjt2
- == cjt1 cjt2
- !=cjt1 cjt2
- print cjt1

On cjt1 i cjt2 poden ser 'A' o 'B' i e1 i e2 són enters.

Sortida

Per a cada línia d'entrada, escriu una línia amb el resultat:

- Si la línia és un conjunt, mostra el conjunt un cop inserit tots els seus elements.
- Si la línia és una comanda, mostra la comanda, el separador ": " i el resultat. Si la comanda retorna o modifica algun conjunt, mostra aquest conjunt.
- Si la línia és el separador de blocs format per 10 guions, mostra els mateixos 10 guions.

Observació

Aquest problema proporciona la definició pública de la classe *cj_2enters* dins del fitxer *conjunt_2enters.hpp*, el programa principal *main.cpp* i un fitxer *Makefile* per facilitar la compilació.
Per implementar el conjunt no es poden usar les classes *stack*, *queue*, *list* o *set* de la STL.
Pots fer un parell de versions, una implementada amb memòria estàtica (usant per exemple arrays de C++) i una altra implementada amb memòria dinàmica. Si la versió amb memòria estàtica té un límit en el màxim nombre d'element d'un conjunt, segurament no passarà els jocs de prova privats on hi ha casos amb conjunts molt grans.
Si els mètodes d'unir, intersectar, restar, igualtat i diferència no es programen de forma eficient (cost lineal) tampoc passaran els jocs de prova privats degut a un excés de temps d'execució.

Exemple d'entrada 1

```
1 0
conte A 0 0
conte A 1 0
conte B 0 0
conte B 1 0
max B
min B
card A
card B
----------
4 -8 -6 14 0 1
1 0 -5 13 12 -8
conte A 0 1
conte A 1 0
conte B 0 1
conte B 1 0
max A
min A
max B
min B
card A
card B
```

Exemple de sortida 1

```
[]
[1,0]
conte A 0 0: 0
conte A 1 0: 0
conte B 0 0: 0
conte B 1 0: 1
max B: 1,0
min B: 1,0
card A: 0
card B: 1
----------
[-6,14 0,1 4,-8]
[-5,13 1,0 12,-8]
conte A 0 1: 1
conte A 1 0: 0
conte B 0 1: 0
conte B 1 0: 1
max A: 4,-8
min A: -6,14
max B: 12,-8
min B: -5,13
card A: 3
card B: 3
```

Exemple d'entrada 2

```
1 0
insereix B 1 0
conte B 0 0
conte B 1 0
max B
min B
card B
insereix B 0 0
conte B 0 0
conte B 1 0
max B
min B
card B
+ A A
* A A
- A A
== A A
!= A A
+ B B
* B B
- B B
== B B
!= B B
+ A B
* A B
- A B
- B A
== A B
!= A B
unir A B
intersectar A B
restar A B
```

```
----------
4 -8 -6 14 0 1
1 0 -5 13 4 -8
+ A A
* A A
- A A
== A A
!= A A
+ B B
* B B
- B B
== B B
!= B B
+ A B
* A B
- A B
- B A
== A B
!= A B
unir A B
intersectar A B
restar A B
----------
4 -8 12 0
12 0 4 -8
== A B
!= A B
```

Exemple de sortida 2

```
[]
[1,0]
insereix B 1 0: [1,0]
conte B 0 0: 0
conte B 1 0: 1
max B: 1,0
min B: 1,0
card B: 1
insereix B 0 0: [0,0 1,0]
conte B 0 0: 1
conte B 1 0: 1
max B: 1,0
min B: 0,0
card B: 2
+ A A: []
* A A: []
- A A: []
== A A: 1
!= A A: 0
+ B B: [0,0 1,0]
* B B: [0,0 1,0]
- B B: []
== B B: 1
!= B B: 0
+ A B: [0,0 1,0]
* A B: []
- A B: []
- B A: [0,0 1,0]
== A B: 0
!= A B: 1
unir A B: [0,0 1,0]
intersectar A B: [0,0 1,0]
restar A B: []
----------
[-6,14 0,1 4,-8]
[-5,13 1,0 4,-8]
+ A A: [-6,14 0,1 4,-8]
* A A: [-6,14 0,1 4,-8]
- A A: []
== A A: 1
!= A A: 0
+ B B: [-5,13 1,0 4,-8]
* B B: [-5,13 1,0 4,-8]
- B B: []
== B B: 1
!= B B: 0
+ A B: [-6,14 -5,13 0,1 1,0 4,-8]
* A B: [4,-8]
- A B: [-6,14 0,1]
- B A: [-5,13 1,0]
== A B: 0
!= A B: 1
unir A B: [-6,14 -5,13 0,1 1,0 4,-8]
intersectar A B: [-5,13 1,0 4,-8]
restar A B: []
----------
[4,-8 12,0]
[4,-8 12,0]
== A B: 1
!= A B: 0
```

Informació del problema

Autoria: Jordi Esteve

Generació: 2026-01-25T21:17:40.516Z