Classe diccionari (I) X68607_ca

Cal implementar la següent classe *dicc* que ens permet representar i manipular diccionaris, on les claus que identifiquen els elements són del tipus *Clau* que admet una relació d'ordre total, és a dir, tenim una operació de comparació < entre claus:

```
#include <iostream>
using namespace std;

template <typename Clau>
class dicc {

public:
    // Constructora per defecte. Crea un diccionari buit.
    dicc();

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

    // Insereix la clau k en el diccionari. Si ja hi era, no fa res.
    void insereix(const Clau &k);

    // Elimina la clau k del diccionari. Si no hi era, no fa res.
    void elimina(const Clau &k);

    // Consulta si la clau k està en el diccionari.
    bool consulta(const Clau &k) const;

    // Retorna quants elements (claus) té el diccionari.
    nat quants() const;

    // Impressió per cout de claus del diccionari en ordre ascendent, separades per
    // un espai, començant per '[' i acabant per ']', en dues versions:
    // Imprimeix totes les claus
    void print() const;
    // Imprimeix les claus entre k1 i k2 ambdós incloses. Pre: k1 <= k2
    void print_interval(const Clau &k1, const Clau &k2) const;

    // Retorna la clau més petita i la més gran respectivament.
    // Pre: El diccionari no està buit
    Clau min() const;
    Clau max() const;

    // Retorna la clau de la posició i-èssima (comptant-les en ordre ascendent).
    // Pre: El diccionari no està buit. 1 <= i <= quants()
```

```
    Clau iessim(nat i) const;

private:
  // Aquí van els atributs i mètodes privats
};

// Aquí va la implementació dels mètodes públics i privats
```

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*.
2. Implementar tots els mètodes de la classe els quals manipularan la representació anterior.

Degut a que jutge.org només permet l'enviament d'un fitxer amb la solució del problema, en el mateix fitxer hi ha d'haver l'especificació i l'implementació de la classe (el que normalment estarien separats en els fitxers *.hpp* i *.cpp*).
Per testejar la classe disposeu d'un programa principal que processa blocs que contenen un diccionari amb claus enteres i vàries comandes que el manipula.

Entrada

L'entrada conté varis blocs separats per línies amb 10 guions (————). Cada bloc consisteix en una línia que conté una seqüències d'enters, són els elements que tindrà originalment el diccionari. A continuació segueixen vàries comandes, una per línea, amb el següent format (k, k1 i k2 són claus enteres; i és un natural major que 0):

- insereix k
- elimina k
- consulta k
- quants
- print
- print_interval k1 k2
- min
- max
- iessim i

Sortida

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

- Si la línia és un diccionari, mostra el diccionari un cop inserit tots els seus elements.
- Si la línia és una comanda, mostra la comanda, el separador ": " i el resultat. Si la comanda modifica el diccionari, mostra quants elements té després d'aplicar-la.
- Si la línia és el separador de blocs format per 10 guions, mostra els mateixos 10 guions.

Observació

Només cal enviar les classes requerides; el programa principal serà ignorat. Seguiu estrictament la definició dels tipus de l'enunciat.
Per implementar el diccionari no es poden usar les classes *stack*, *queue*, *list*, *set* o *map* de la STL.
Els mètodes insereix, elimina, consulta, min, max i iessim almenys han de tenir cost logarítmic (en el cas mig) per superar els jocs de prova privats. El mètode quants ha de tenir cost constant i els mètodes print i print_interval cost lineal.

Exemple d'entrada 1

```
5 -3 8 2 -1 7 -7 -6
quants
consulta -3
consulta -4
consulta 6
consulta 9
insereix -4
consulta -4
insereix 6
consulta 6
insereix 9
consulta 9
insereix 8
print
print_interval -2 6
min
max
iessim 1
iessim 2
iessim 5
iessim 7
iessim 11
elimina -3
consulta -3
elimina -7
consulta -7
elimina 7
consulta 7
elimina 7
print
min
max
iessim 1
iessim 2
iessim 5
elimina -6
elimina 5
elimina 9
print
min
max
iessim 1
iessim 2
iessim 4
```

Exemple de sortida 1

```
[-7 -6 -3 -1 2 5 7 8]
quants: 8
consulta -3: 1
consulta -4: 0
consulta 6: 0
consulta 9: 0
insereix -4: 9
consulta -4: 1
insereix 6: 10
consulta 6: 1
insereix 9: 11
consulta 9: 1
insereix 8: 11
print: [-7 -6 -4 -3 -1 2 5 6 7 8 9]
print_interval -2 6: [-1 2 5 6]
min: -7
max: 9
iessim 1: -7
iessim 2: -6
iessim 5: -1
iessim 7: 5
iessim 11: 9
elimina -3: 10
consulta -3: 0
elimina -7: 9
consulta -7: 0
elimina 7: 8
consulta 7: 0
elimina 7: 8
print: [-6 -4 -1 2 5 6 8 9]
min: -6
max: 9
iessim 1: -6
iessim 2: -4
iessim 5: 5
elimina -6: 7
elimina 5: 6
elimina 9: 5
print: [-4 -1 2 6 8]
min: -4
max: 8
iessim 1: -4
iessim 2: -1
iessim 4: 6
```

Exemple d'entrada 2

```
5
quants
consulta 0
consulta 5
min
max
iessim 1
elimina 0
elimina 5
print
----------

consulta 0
consulta 1
insereix 1
consulta 1
insereix 1
print
print_interval 0 2
print_interval -3 -2
print_interval 2 3
min
max
iessim 1
elimina 1
consulta 1
elimina 1
print
```

Exemple de sortida 2

```
[5]
quants: 1
consulta 0: 0
consulta 5: 1
min: 5
max: 5
iessim 1: 5
elimina 0: 1
elimina 5: 0
print: []
----------
[]
consulta 0: 0
consulta 1: 0
insereix 1: 1
consulta 1: 1
insereix 1: 1
print: [1]
print_interval 0 2: [1]
print_interval -3 -2: []
print_interval 2 3: []
min: 1
max: 1
iessim 1: 1
elimina 1: 0
consulta 1: 0
elimina 1: 0
print: []
```

Informació del problema

Autoria: Jordi Esteve

Generació: 2026-01-25T17:44:08.385Z