Redispersió en taules de dispersió amb direccionament obert fent sondeig lineal X32546_ca

Donada la classe *dicc* que permet gestionar diccionaris amb claus enteres, cal implementar els mètodes:

```
float factor_de_carrega () const;
// Pre: Cert
// Post: Retorna el factor de càrrega de la taula de dispersió

void redispersio ();
// Pre: Cert
// Post: Redimensiona la taula de dispersió amb una mida el doble que
// l'anterior més un (_M passa a ser 2*_M+1)
```

Els diccionaris s'implementen amb taules de dispersió amb direccionament obert fent sondeig lineal.
Cal enviar a jutge.org la següent especificació de la classe *dicc* i la implementació dels mètodes dins del mateix fitxer (la resta de mètodes públics ja estan implementats). Indica dins d'un comentari a la capçalera de cada mètode el seu cost en funció del nombre d'elements del diccionari n.

```
#include <iostream>
using namespace std;
typedef unsigned int nat;

class dicc {
  // Taula de dispersió amb direccionament obert fent sondeig lineal.
  public:
    dicc(nat m);
    // Pre: m > 0
    // Post: Crea un diccionari buit en una taula de dispersió de mida m

    ~dicc();
    // Pre: Cert
    // Post: Destrueix el diccionari

    nat quants() const;
    // Pre: Cert
    // Post: Retorna quants elements (claus) té el diccionari.

    void print() const;
    // Pre: Cert
    // Post: Imprimeix per cout del contingut de la taula de dispersió

    void insereix (const int &k);
    // Pre: Cert
    // Post: Insereix la clau k en el diccionari. Si ja hi era, no fa res.
```

```
    // Redimensiona la taula de dispersió amb una mida el doble que
    // l'anterior més un si el factor de càrrega és superior a 0.8

    float factor_de_carrega () const;
    // Pre: Cert
    // Post: Retorna el factor de càrrega de la taula de dispersió

    void redispersio ();
    // Pre: Cert
    // Post: Redimensiona la taula de dispersió amb una mida el doble que
    // l'anterior més un (_M passa a ser 2*_M+1)

  private:
    enum Estat {lliure, esborrat, ocupat};
    struct node_hash {
      int    _k;      // Clau
      Estat  _est;
    };
    node_hash *_taula; // Taula amb les claus del diccionari
    nat _M;            // Mida de la taula
    nat _quants;       // Nº d'elements guardats al diccionari

    static long const MULT = 31415926;

    static long h(int k);
    // Pre: Cert
    // Post: Retorna un valor de dispersió entre 0 i LONG_MAX a partir de k

    nat busca_node(const int &k) const;
    // Pre: Cert
    // Post: Retorna la posició on es troba l'element amb la clau k o,
    // en cas que no trobi la clau, la primera posició no ocupada.

    // Aquí va l'especificació dels mètodes privats addicionals

};

// Aquí va la implementació dels mètodes públics factor_de_carrega, redispersio i
// dels mètodes privats addicionals
};
```

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 de la classe i la implementació dels mètodes *factor_de_carrega* i *redispersio* (el que normalment estarien separats en els fitxers *.hpp* i *.cpp*). Per testejar la classe disposes d'un programa principal que llegeix un conjunt d'elements, els insereix en un diccionari i mostra el seu contingut, després llegeix un segon conjunt d'elements, els insereix en el mateix diccionari i mostra novament el seu contingut.

Entrada

L'entrada té tres línies: la primera conté un natural positiu amb la dimensió inicial de la taula de dispersió i les altres dos contenen enters separats amb espais, són els enters que s'insereixen en el diccionari.

Sortida

Escriu el contingut del diccionari dos vegades: després d'inserir el primer conjunt d'enters i després d'inserir el segon conjunt d'enters. Cada vegada es mostra en diferents línies la quantitat d'elements que té, el factor de càrrega i el contingut de totes les caselles de la taula de dispersió(la clau si la casella està ocupada, "LL" si està lliure o "ES" si està esborrada).

Observació

Per calcular el valor de dispersió utilitza el mètode *h* que ja està implementat i que permet calcular un valor de dispersió entre 0 i *LONG_MAX* (el valor long int més gran que permet el compilador) a partir d'una clau entera.
Només cal enviar la classe requerida i la implementació dels mètodes *factor_de_carrega* i *redispersio*. Pots ampliar la classe amb mètodes privats. Segueix estrictament la definició de la classe de l'enunciat.
Indica dins d'un comentari a la capçalera de cada mètode el seu cost en funció del nombre d'elements del diccionari *n*.

Exemple d'entrada 1

```
13
5 -3 8 2 -1 7 -7 -6
7 -2 9 5 -3 2 -7 4
```

Exemple de sortida 1

```
Nº elem: 8
Factor de càrrega: 0.615385
0: -1
1: 7
2: LL
3: 8
4: LL
5: -3
6: 5
7: 2
8: -6
9: -7
10: LL
11: LL
12: LL
-----------
Nº elem: 11
Factor de càrrega: 0.407407
0: -1
1: LL
2: LL
3: LL
4: -3
5: 7
6: 2
7: -7
8: -2
9: LL
10: 9
11: LL
12: LL
```

```
13: 8
14: LL
15: 5
16: -6
17: LL
18: LL
19: 4
20: LL
21: LL
22: LL
23: LL
24: LL
25: LL
26: LL
-----------
```

Exemple d'entrada 2

```
29
5 -5 3 -3 9 -9 2 -2 -5 5 1 -1 7 -7 0 4 -4 -8 6 -6
2 11 4 12 8 14 0 10 17 13
```

Exemple de sortida 2

```
Nº elem: 19
Factor de càrrega: 0.655172
0: -1
1: 0
2: 9
3: -2
4: -3
5: 2
6: 1
7: 7
8: 3
9: -7
10: -4
11: 5
12: -8
13: 6
14: -6
15: LL
16: LL
17: -5
18: -9
19: 4
20: 8
21: LL
22: LL
23: LL
24: LL
25: LL
26: LL
27: LL
28: LL
-----------
Nº elem: 25
Factor de càrrega: 0.423729
0: -1
1: 0
2: LL
3: LL
4: LL
5: LL
6: 9
7: LL
8: -5
9: 4
10: 17
11: LL
12: 5
13: -6
14: LL
15: LL
```

```
16: 10
17: LL
18: LL
19: 3
20: -4
21: 12
22: LL
23: -2
24: 1
25: -9
26: 8
27: LL
28: LL
29: LL
30: LL
31: 11
32: LL
33: LL
34: LL
35: LL
36: LL
37: LL
38: LL
39: -3
40: 2
41: LL
42: LL
43: LL
44: LL
45: LL
46: LL
47: LL
48: 13
49: 7
50: -8
51: LL
52: -7
53: 6
54: 14
55: LL
56: LL
57: LL
58: LL
-----------
```

Exemple d'entrada 3

```
13
5 -5 3 -3 9 -9 2 -2 -5 5 1 -1 7 -7 0 4 -4 8 -8 6 -6
2 11 4 12 8 14 0 10 17 13 21 18 15 20 16 19
```

Exemple de sortida 3

```
Nº elem: 19
Factor de càrrega: 0.703704
0: -1
1: 0
2: LL
3: LL
4: 1
5: -3
6: 2
7: 7
8: 3
9: -2
10: 9
11: -7
12: -4
13: -9
14: 8
15: 5
16: -8
17: 6
18: -6
19: -5
20: 4
21: LL
22: LL
23: LL
24: LL
25: LL
26: LL
-----------
Nº elem: 31
Factor de càrrega: 0.563636
0: -1
1: 0
2: 19
3: -5
```

```
4: 4
5: 18
6: LL
7: LL
8: LL
9: LL
10: LL
11: LL
12: LL
13: LL
14: LL
15: LL
16: LL
17: LL
18: LL
19: 1
20: -2
21: LL
22: 7
23: -8
24: LL
25: 10
26: LL
27: LL
28: LL
29: LL
```

```
30: LL
31: LL
32: 5
33: -6
34: 9
35: 16
36: LL
37: LL
38: 14
39: LL
40: 17
41: 12
42: -7
43: -9
44: 8
45: 3
46: -4
47: 6
48: 11
49: 13
50: -3
51: 2
52: 15
53: 20
54: 21
-----------
```

Informació del problema

Autoria: Jordi Esteve

Generació: 2026-01-25T15:11:40.627Z

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