

```

template <class T> class set {

// Tipus de mòdul : dades

// Descripció del tipus: Conjunt que conté elements de tipus T; els elements
// estan ordenats creixentment per l'ordre definit a T, que no poden estar
// repetits; es pot consultar i modificar elements pels extrems, on des de
// cada element es pot accedir a l'element anterior i posterior (si
// existeixen), i que admet afegir-hi i esborrar-hi elements

// El cost temporal de totes les operacions és constant, tret de
//
// - les cerques, l'insert (sense iterador o sense la posició bona) i l'erase
// (sense iterador), que són logarismiques respecte a la mida del set
//
// - la copiadora, la destructora i clear, que tenen cost lineal respecte a la
// mida del set original (en aquestes, i en insert i erase, també cal tenir en
// compte el cost de la còpia o esborrat de cada objecte implicat de tipus T)

private:

public:

// Constructores

set();
/* Pre: cert */
/* Post: El resultat és un set sense cap element */

set(const set& original);
/* Pre: cert */
/* Post: El resultat és un set còpia d'original */

// Destructor: Esborra automàticament els objectes locals en sortir d'un
// àmbit de visibilitat

~set();

// Modificadores

void clear();
/* Pre: cert */
/* Post: El paràmetre implícit és un set buit */

pair<iterator,bool> insert(const T& x);
/* Pre: cert */
/* Post: si x hi és al p.i., el second del resultat és fals, el p.i. no canvia
i el first del resultat apunta a l'element de valor x al p.i; en cas
contrari, el second del resultat és true, x s'afegeix al p.i. i el first del
resultat apunta al nou element del p.i. */

iterator insert(const_iterator it, const T& x);
/* Pre: it referencia algun element existent al paràmetre implícit o
és igual a l'end d'aquest */
/* Post: si x hi és al p.i., el p.i. no canvia i el resultat apunta a
l'element de valor x al p.i ; en cas contrari, x s'afegeix al p.i. i el
resultat apunta al nou element del p.i. */

int erase(const T& x);
/* Pre: cert */
/* Post: si x hi és al p.i., el p.i. no canvia i el resultat és 0; en cas
contrari, l'element x s'esborra del p.i. i el resultat és 1 */

```

```

iterator erase(const_iterator it);
/* Pre: it referencia algun element existent al paràmetre implícit,
   que no és buit */
/* Post: El paràmetre implícit és com el paràmetre implícit original sense
   l'element referenciat per l'it original; el resultat referencia
   l'element següent al que referenciava it al p.i. original */

// Consultores

bool empty() const;
/* Pre: cert */
/* Post: El resultat indica si el paràmetre implícit té elements o no */

int size() const;
/* Pre: cert */
/* Post: El resultat és el nombre d'elements del paràmetre implícit */

iterator find(const T& x); // + l'equivalent const
/* Pre: cert */
/* Post: si x hi és al p.i., el resultat apunta a aquest element; en cas
   contrari, apunta a l'end() */

iterator lower_bound(const T& cx); // + l'equivalent const
/* Pre: cert */
/* Post: si x hi és al p.i., el resultat apunta a aquest element; en cas
   contrari, apunta al primer element del p.i. que aniria després de x (si x
   és més gran que tots els elements del p.i. seria l'end()) */

// Iteradors típics

iterator begin();
/* Pre: cert */
/* Post: El resultat és un iterator al principi del paràmetre implícit */

const_iterator begin() const;
/* Pre: cert */
/* Post: El resultat és un const_iterator al principi del paràmetre implícit */

iterator end();
/* Pre: cert */
/* Post: El resultat és un iterator a un element fictici immediatament posterior
   al final del paràmetre implícit */

const_iterator end() const;
/* Pre: cert */
/* Post: El resultat és un const_iterator a un element fictici immediatament
   posterior al final del paràmetre implícit */

// Notes:
// a) si s és buit, s.begin() és el mateix que s.end()
// b) si s ve qualificat com a const, s.begin() i s.end() retornen un
// const_iterator; en cas contrari, retornen un iterator
};

/* Operacions amb iterators:

++it : Avança al següent element, no vàlid a l'end

--it : Retrocedeix a l'anterior element, no vàlid al begin

```

\*it : Designa l'element referenciat per it; no vàlid per a l'end o per a iteradors que no referencien res; **només es pot fer servir com a "read only"**

it1=it2 : Assigna l'iterador it2 a it1; un const\_iterator no es pot assignar a un iterator

it1==it2 : val true si els iteradors it1 i it2 són iguals; false si no

it1!=it2 : val true si els iteradors it1 i it2 són diferents; false si no

\*/