MANEJO DE UNA ESTRUCTURA SET

```
#include <cstdio>
#include <set>
using namespace std;

/**
 * Manejo de set.
 * Arbol binario de búsqueda (BST) balanceado de solo claves.
 * No acepta claves duplicadas.
 * Las operaciones de inserción/búsqueda/eliminación se realizan en O(log n).
 *
 */
int main() {

	set<int> claves;

	claves.clear();				// Elimina todos los elementos.

	/* ¿El árbol esta vacío? */
	printf("%d\n", claves.empty());		// 1 = verdad

	claves.insert(100);			// Ingresa 100
	claves.insert(10);			// Ingresa 10
	claves.insert(50);			// Ingresa 50
	claves.insert(1);			// Ingresa 1
	claves.insert(13);			// Ingresa 13

	claves.erase(10);			// Elimina clave 10

	// Muestra todos los elementos
	for (set<int>::iterator it = claves.begin(); it != claves.end(); it++)
		printf("%d,", *it);
	printf("\n");

	// Busca el 50, Imprime encontrado
	if (claves.find(50) == claves.end())
		printf("no encontrado\n");
	else
		printf("encontrado\n");

	// claves < 50, Imprime [1, 13]
	for (set<int>::iterator it = claves.begin(); it != claves.lower_bound(50); it++)
		printf("%d,", *it);
	printf("\n");

	// claves >= 50, Imprime [50, 100]
	for (set<int>::iterator it = claves.lower_bound(50); it != claves.end(); it++)
		printf("%d,", *it);
	printf("\n");

	return 0;
}
```

MANEJO DE UNA ESTRUCTURA MAP

```
#include <cstdio>
#include <string>
#include <map>
using namespace std;

/**
* Manejo de map.
* Arbol binario de búsqueda (BST) balanceado (clave -> dato)
* No acepta claves duplicadas.
* Las operaciones de inserción/búsqueda/eliminación se realizan en O(log n).
*
*/
int main() {

	map<int, string> mapa;

	mapa.clear();                                   // Elimina todos los elementos.

	/* ¿El árbol esta vacío? */
	printf("%d\n", mapa.empty());                   // 1 = verdad

	mapa[100] = "a";                                // Ingresa <100, "a">
	mapa[10] = "b";                                 // Ingresa <10, "b">
	mapa[50] = "c";                                 // Ingresa <50, "c">
	mapa[1] = "d";                                  // Ingresa <1, "d">
	mapa[13] = "e";                                 // Ingresa <13, "e">

	mapa.erase(10);                                 // Elimina <10, "b">

	// Muestra todos los elementos
	for (map<int, string>::iterator it = mapa.begin(); it != mapa.end(); it++)
		printf("%d %s\n", it->first, ((string)it->second).c_str());

	printf("%s\n", ((string)mapa[50]).c_str());     // Imprime "c"

	// Busca el 50, Imprime encontrado
	if (mapa.find(50) == mapa.end())
		printf("no encontrado\n");
	else
		printf("encontrado\n");

	// Obtiene el BST balanceado entre 13 <= clave <= 50
	for (map<int, string>::iterator it = mapa.lower_bound(13);
		it != mapa.upper_bound(50); it++)
		printf("%d %s\n", it->first, ((string)it->second).c_str());

	return 0;
}
```