Ejercicio de prácticas de EDA
del 2014.02.25

Implementa módulos C++ con distintas variantes del TAD Conjunto

OBJETIVO

Adquirir familiaridad con el uso de #includes y sus protecciones (#ifndef / #define / #endif). Definir y usar módulos C++

ENTREGA

Entrega un único fichero llamado **nombre1_apellidos1_nombre2_apellidos2.zip** (si tienes compañero), ó **nombre1_apellidos1.zip** (si no tienes compañero) que contenga, *sin subdirectorios*, los .h y el main.cpp de tu entrega, comprimido mediante Zip (no RAR, no 7Z: **Zip**), vía la tarea correspondiente de **Campus Virtual**.

1. Crea un fichero llamado excepciones.h

Esta clase debe contener las siguientes definiciones:

```
/// Excepción de conjunto lleno
class ConjuntoLleno {};

/// Excepción de conjunto vacío
class ElementoInvalido {};
```

Y debe estar protegida de inclusiones múltiples usando

```
#ifndef _EXCEPCIONES_H
#define _EXCEPCIONES_H

// (resto del contenido aquí)

#endif _EXCEPCIONES_H
```

2. Crea un fichero conjunto_malo.h

Se asume que un Conjunto implementa las operaciones de

- **Insertar** un elemento (falla si no cabe)
- **Eliminar** un elemento dado (falla si no lo contiene)
- **Buscar** si contiene un elemento dado

Implementa un `conjunto_malo.h`, protegido mediante el `#ifndef _CONJUNTO_MALO_H` correspondiente, donde los elementos insertados se almacenen al final de un array estático. *Esta implementación es la que se encuentra en la página 29 de los apuntes del capítulo.*

Importante: usa un `#include "excepciones.h"` en lugar de definir clases para las excepciones de elemento inválido y exceso de elementos.

La clase definida debe llamarse `ConjuntoMalo`. Inicializa la constante `ConjuntoMalo::MAX` a 1000.

3. Crea un fichero conjunto_bueno.h

Crea un fichero llamado `conjunto_bueno.h`, protegido mediante el `#ifndef _CONJUNTO_BUENO_H` correspondiente, donde los elementos insertados se almacenen en la posición que les corresponda de un array estático que se mantenga **siempre ordenado**.

Al igual que en el caso anterior, deberás incluir `#include "excepciones.h"` en lugar de definir tú esas excepciones. Llama a tu clase `ConjuntoBueno`.

La comprobación de si un elemento está o no presente, la inserción y el borrado se deben implementar usando búsqueda binaria (y por tanto con complejidad ***O(log n)***). Es posible, con algo de imaginación, usar una función auxiliar que permita usar la misma búsqueda binaria tanto para inserción como para búsqueda y eliminación.

Inicializa la constante `ConjuntoBueno::MAX` a 1000.

4. Crea un main() de prueba

Crea un `main()` que #incluya tanto `"conjunto_malo.h"` como `"conjunto_bueno.h"`, que cree un conjunto de enteros de cada tipo (bueno y malo), y que sobre cada uno de ellos

- Inserte 800 elementos al azar entre 1 y 1000
- Compruebe si existen 800 elementos al azar entre 1 y 1000; y los que existan, los elimine

Usa `#include <ctime>` e `#include<cstdlib>` para tener acceso a la generación de números aleatorios. Usa `srand((unsigned int)time(0));` como primera instrucción de tu `main()` para inicializarlos, y `((rand()%1000) + 1)` para generar enteros aleatorios entre el 1 y el 1000.

Una vez finalizado el main, y probado sin encontrar errores, entrega todos los .h y el .cpp resultantes según se solicita al comienzo del enunciado.