TAD Conjunto

FP2 grupo B. Curso 2020/2021. Convocatoria única.
Actividad complementaria

1. ¿Qué es un TAD?

Un TAD (Tipo Abstracto de Datos) es un conjunto de operaciones definidas sobre un conjunto de datos. Por una parte, un TAD a través de la implementación de sus tipos permite la representación de la información que se procesa. Por otra, a través de las funciones que proporciona, permite al programador usuario acceder a esta información sin preocuparse de los detalles de la implementación. Es decir, un TAD ofrece funciones que representen las operaciones lógicas de esa información.

Si quieres saber más sobre esto puedes mirar aquí: `https://es.wikipedia.org/wiki/Tipo_de_dato_abstracto`

2. Mi primer TAD

En esta práctica vamos a hacer un TAD de conjuntos de enteros. Estamos interesados en tener un tipo `tConjunto` que nos permita trabajar utilizando las operaciones de intersección, unión, diferencia, etc.

Diseña un módulo en C++ que proporcione las siguientes operaciones del TAD conjunto:

- `void crearConjuntoVacio(tConjunto &c)`: crea un conjunto vacío.
- `bool pertenece(tConjunto c, int e)`: determina si `e` pertenece a `c` o no.
- `bool conjuntoVacio(tConjunto c)`: determina si `c` es vacío o no.
- `void insertar(tConjunto &c, int e)`: inserta `e` en `c`. Importante: solo puede haber una aparición de c.
- `void mostrar(tConjunto c)`: muestra por pantalla el contenido de `c`.
- `void eliminar(tConjunto &c, int e)`: elimina `e` en `c`. Importante: si no existe c no hace nada.
- `bool operator==(tConjunto a, tConjunto b)`: determina si `a` y `b` son iguales o no.
- `tConjunto operator*(tConjunto a, tConjunto b)`: devuelve la intersección de `a` y `b`.
- `tConjunto operator+(tConjunto a, tConjunto b)`: devuelve la unión de `a` y `b`.
- `tConjunto operator-(tConjunto a, tConjunto b)`: devuelve la diferencia de `a` y `b`.
- `bool operator<<(tConjunto a, tConjunto b)`: determina si `a` está contenido en `b` o no.

Si no te acuerdas de la teoría de conjuntos puedes mirar aquí: `https://es.wikipedia.org/wiki/Conjunto#Operaciones_con_conjuntos`

Importante: no puedes cambiar nada en estos prototipos. Puedes utilizar más funciones.

2.1. Detalles de la implementación

Para representar el contenido de los conjuntos utilizaremos arrays dinámicos. Como nuestro array puede no estar lleno utilizaremos un contador para saber el número de elementos y la primera posición libre.

Algunas observaciones:

- Cuando se crea un conjunto se crea un array de enteros de tamaño 10.

- Si queremos insertar en el array y no hay posiciones libres, creamos otro array con 10 posiciones más. Copiamos los elementos del anterior en el array nuevo y destruimos el anterior.
- Si al eliminar un elemento el número de posiciones con información es menor del 25 %, realizamos el proceso inverso. Creamos un array con 10 posiciones menos, copiamos la información y eliminamos el contenido anterior.
- Diseña el tipo `tConjunto` de tal forma que permita este tipo de procesamiento.

2.2. Entrega de la práctica

- El código debe ser correcto y legible. La corrección incluye la gestión adecuada de la memoria dinámica.
- Se proporcionará, a través de un módulo, tanto un tipo como un conjunto de operaciones permitidas.
- Se recomienda hacer la práctica en parejas. Si es así, solo uno de los autores debe subirla. En caso contrario, se anularán la práctica de ambos.
- La práctica se corregirá en Visual Studio 2019.
- La entrega se realiza a través del Campus Virtual.