Organización de Computadoras
Departamento de Ciencias e Ingeniería de la Computación
Universidad Nacional del Sur
Segundo Cuatrimestre de 2024

Laboratorio N° 4
Programación en lenguaje C

IMPORTANTE: lea este documento detenidamente para no cometer errores o malos entendidos en su contenido. APAGUE el celular. NO SE PERMITE el uso de sistemas de mensajería instantanea. SI SE PERMITE el uso de Internet, ChatGPT, Gemini.

Introducción

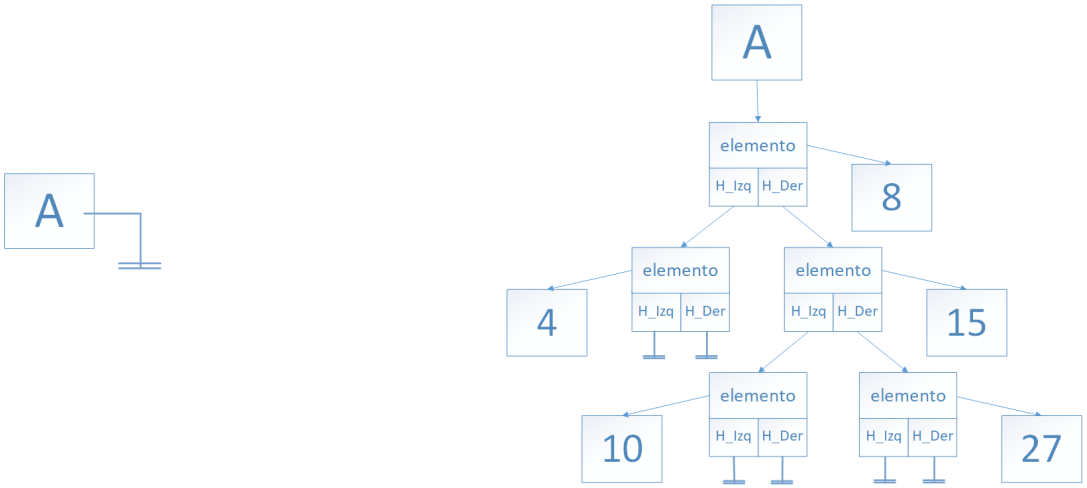
Un Tipo de Dato Abstracto (TDA) es uno de los primeros ejemplos de buenas prácticas a la hora de programar y se utiliza para describir una estructura de datos junto con las operaciones que pueden realizarse sobre esa estructura de datos. Ejemplos de TDA incluyen listas, pilas, colas, árboles, conjuntos, etc. Cada uno de éstos, tiene operaciones específicas asociadas que permiten a los programadores interactuar con los datos de manera efectiva sin necesidad de conocer los detalles de implementación. Esto permite que los TDA sean considerados un concepto fundamental para la programación y la estructuración de código en proyectos de software a gran escala.

Enunciado

Nos enfocaremos en el TDA Árbol Binario de Búsqueda (ABB), el cual consta de una estructura de datos que organiza y almacena un conjunto de elementos donde cada nodo contiene un valor y tiene como máximo dos hijos: un hijo izquierdo y un hijo derecho. La característica distintiva de un ABB es que el valor de cada nodo es mayor o igual que todos los valores en su subárbol izquierdo y menor que todos los valores en su subárbol derecho. Para este laboratorio se deberá implementar en forma parcial un TDA ABB. Dos bosquejos como ejemplos de ABBs válidos son representados en la figura 2.

Para ello se deberá descargar de la Plataforma Moodle el archivo **Lab04.zip** que contiene cuatro documentos necesarios para la implementación, dónde: i) `main.c` es el programa principal, ii) `abb.h` es el archivo encabezado (header) correspondiente al TDA Árbol Binario de Búsqueda que contiene la declaración completa de la estructura y sus operaciones, iii) `abb.c` es el archivo que tiene la implementación de las operaciones del TDA y finalmente iv) `define.h` posee la definición de las constantes utilizadas en este proyecto.
La estructura utilizada, declaración ubicada en el header:

```
typedef int tElem;
typedef struct tNodo{
  tElem *elemento;
  struct tNodo *H_izq, *H_der;
} *tABB;
```



(a) ABB vacío (b) ABB con 5 elementos

Figura 1: Ejemplos de Árboles Binarios de Búsqueda

A continuación se detalla una breve descripción de las operaciones junto con la indicación de la que deberá ser implementada por la comisión:

- `void crearABBVacio(tABB *A)`: Crea una árbol binario de búsqueda vacío.
- `void insertarElemABB(tElem E, tABB *A)`: Inserta el elemento `E` en el árbol binario de búsqueda `A`. **Esta operación deberá ser IMPLEMENTADA**
- `void eliminarElemABB(tElem E, tABB *A)`: Elimina el elemento `E` del árbol binario de búsqueda `A`. **Esta operación deberá ser IMPLEMENTADA**
- `int esABBVacio(tABB A)`: Devuelve verdadero ($\neq 0$) si el árbol binario de búsqueda está vacío y falso ($= 0$) si contiene al menos un elemento.
- `void preorden(tABB A)`: Muestra por pantalla el listado del recorrido en preorden del árbol binario de búsqueda `A`.

IMPORTANTE: De los archivos descargados, sólo se podrá y deberá editar el archivo "abb.c", agregando la operación solicitada. Se considerará una FALTA cualquier modificación adicional, lo que resultará en la DESAPROBACIÓN del Laboratorio.

Sobre la implementación

- El archivo fuente principal (main.c) se debe renombrar a **ABBTest.c**.
- La compilación debe realizarse con el *flag* `-Wall` habilitado. El código debe compilar **sin advertencias** de ningún tipo.
- La copia o plagio del laboratorio es una falta grave. Quien incurra en estos actos de deshonestidad académica desaprobará el laboratorio y **el cursado de la materia**.

Estilo de programación

- El código implementado debe reflejar la aplicación de las técnicas de programación modular estudiadas a lo largo de la carrera.
- En el código, entre eficiencia y claridad, se debe optar por la claridad. Toda decisión en este sentido,de hacer falta, debe constar en la documentación interna del código fuente asociado al programa implementado.
- El código debe estar indentado, comentado y debe reflejar el uso adecuado de nombres significativos para la definición de variables, funciones y parámetros.

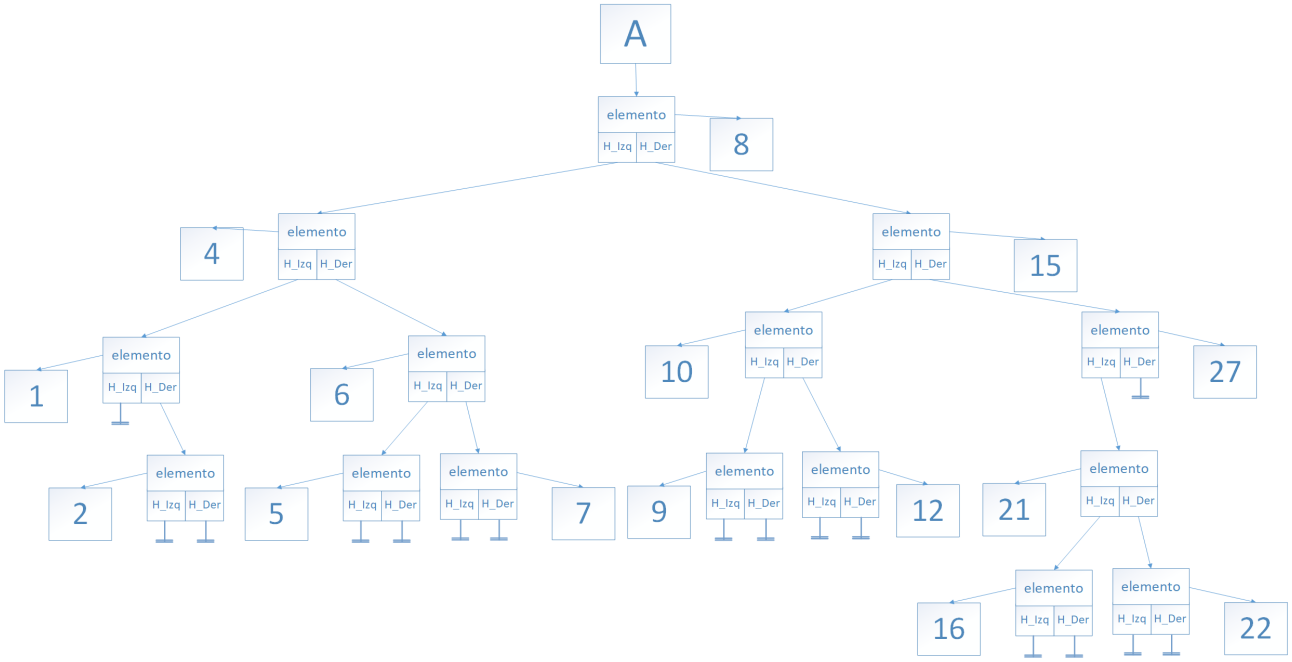
Entrega

- La entrega del código fuente se realizará a través de un archivo comprimido **zip** o **rar**, denominado ***LAB4-Comision-XX*** (el valor XX debe modificarse por el número de comisión correspondiente) donde se deben incorporar el/los archivos fuente ".c" y ".h".
- El archivo comprimido debe subirse a la plataforma Moodle, en el apartado *Laboratorios de programación*, dentro de la tarea Laboratorio 4 [actividad]. Para esto, tenga en cuenta las siguientes consideraciones:
  - **Fecha de entrega: jueves 05/11/24, 14:15 hs.**
    Considere que la plataforma Moodle de forma **automática inhabilitará** la opción de entrega a partir de esta hora, por lo que se recomienda no esperar a último momento para realizar la entrega.
  - **Intentos de entrega:** la plataforma Moodle **no permitirá** realizar más de una entrega de los archivos. No se aceptará ninguna entrega fuera de la plataforma Moodle.
  - El archivo comprimido debe subirlo **sólo uno** de los integrantes de la comisión.
  - La cátedra no se responsabiliza por cualquier inconveniente surgido a partir del envío del proyecto a último momento. Es importante **no esperar hasta último momento** para realizar el envío evitando así todo tipo de situación excepcional que pudiese ocurrir (sobrecarga o caída del sistema, fallas en la conexión a Internet, etc.).

Corrección

- La cátedra evaluará tanto el **diseño**, **implementación**, **presentación** y cumplimiento de **todas** las condiciones de entrega.

Como ayuda se proporciona un ABB ya cargado para realizar las pruebas necesarias en el testing



(a) int arr[15]=$\{8, 4, 6, 15, 1, 7, 2, 10, 27, 12, 21, 5, 9, 16, 22\}$

Figura 2: ABB para realizar las pruebas