

# Programa 02: Arbol binario de busqueda

PROFESOR: NOÉ ORTEGA SANCHEZ  
VICTOR MANUEL MARTINEZ OROZCO



EDA2-D01-15888

El programa es un árbol binario de búsqueda dinámico el cuál almacena datos abstractos, para ser más preciso, alumnos. Este objeto alumno se compone del nombre, código y carrera.

Para la implementación de este, se definieron los siguientes métodos básicos:

(cabe aclarar que para la gran mayoría de métodos del árbol se crearon tanto métodos públicos que solo reciben el datos externos al árbol, como el dato tipo alumno, y su correspondientes métodos privados y parametrizados con la raíz del árbol y el mismo dato tipo alumno. Esto por la sencilla razón de que se buscó que, la raíz al ser únicamente manipulada por el árbol, podía ser privada y de esta manera el único que tendría “contacto” con ella sería el propio árbol.)

- **Inserción de datos:** Para este método se le solicitan al usuario 3 datos; nombre, código y carrera. Estos 3 datos de tipo string son asignados a los atributos del objeto alumno con sus correspondientes setters. Posteriormente, el objeto alumno es mandado al árbol para su inserción. El algoritmo de inserción del árbol, naturalmente al necesitar la comparación de un dato con otro para así asignarlo en el subárbol izquierdo o derecho implica un operador de comparación en el objeto alumno. Este operador recibe 2 objetos de su mismo tipo y retorna un booleano resultado de la comparación de estos dos objetos en su campo nombre y de esta manera se logra asignar el objeto de forma ordenada en el árbol. Cabe recalcar que para el método de inserción se crearon dos tipos, uno privado, parametrizado con la raíz y un dato tipo plantilla que equivale al objeto tipo alumno y otro método público, parametrizado para recibir únicamente el objeto tipo plantilla a insertar y llamar al método privado con la raíz y el mismo dato tipo plantilla como parámetros.
- **Muestreo valores:** Para este método se implementaron los 3 tipos de recorrido del árbol que son: preorden, inorden y postorden. Estos siguen el algoritmo clásico de siempre, no reciben nada por su parte pública al ser llamados desde el menú pero los métodos públicos llaman a los privados con la dirección de la raíz al árbol como parámetro.
  1. El preorden imprime en pantalla el dato de la raíz, luego inicia la recursividad por su subárbol izquierdo y posteriormente por su subárbol derecho.
  2. El inorden inicia la recursividad por su parte izquierda, luego imprime el dato y recorre recursivamente la parte derecha.
  3. El postorden recorre todo el subárbol izquierdo, luego el derecho recursivamente y finalmente imprime los datos.

Para ejecutar correctamente los algoritmos recursivos del código, se emplean los getter izquierda y derecha pertenecientes al nodo.

- **Mostrar elemento mayor y menor:** Este método emplea a un método de búsqueda “retrieve”, el cual recibe de parámetro una dirección de memoria y retorna un dato tipo alumno. Como parámetro entonces del retrieve se le manda el obtener mayor “gethighest”, para este método se crearon tanto su método parametrizado como su método sin parametrizar. El parametrizado recibe un puntero a nodo. El sin parametrizar llama al parametrizado con la dirección de la raíz, entonces, cuando el método es llamado sin parametrizar, este toma como referencia la altura del árbol correspondiente a la raíz principal, en cambio, si se le mandan parámetros, toma la altura en referencia al nodo que recibió. El algoritmo que emplea solamente es un recorrido del árbol o

subárbol empleando el `getRight` o `getLeft` hasta llegar a una dirección nula, este retorna la dirección anterior a la última llamada del `getRight` cuando fue nula y finalmente con estos dos parámetros retornados (la dirección y el objeto) se muestran en pantalla los datos del objeto.

- Mostrar altura: Para este método, de la misma manera que el buscar elemento mayor/menor del árbol, se implementaron dos métodos; uno público que no recibe parámetros y mide la altura del árbol con respecto a la raíz principal y otro privado que sí recibe parámetros, en este caso un puntero a nodo el cual mide la altura del árbol con respecto a tal puntero a nodo. En cuanto a la implementación del código, ambos métodos (público y privado) funcionan recursivamente sumando a dos variables enteras de uno en uno dependiendo la altura de cada subárbol. En caso de que no se le manden parámetros a este método, se retorna el resultado de comparar cuál altura es mayor, si el subárbol izquierdo o el derecho, más 1.
- Eliminar todo el árbol: El método de eliminar antes que todo, verifica si el árbol está vacío con un booleano, en caso de ser negativa la respuesta prácticamente hace que la raíz del árbol apunte a nulo, desreferenciando así cualquier nodo que dependa de esta y al mismo tiempo dejando inicializado el árbol en dado caso de que se busquen reinsertar datos.
- Guardar en archivo: Para el método de guardado en archivo únicamente se necesitó el mismo algoritmo de preorden ya que este naturalmente sirve para recorrer el árbol en el mismo orden en que los datos fueron insertados al árbol. Más detalladamente, se crearon 2 métodos (público y privado). El público no recibe parámetros y solamente llama al método de `preordenGuardar`, el cual es privado y parametrizado con la raíz del árbol que se busca guardar. Este método de guardar declara dos variables para operar con el documento de texto, ya que se utiliza la técnica poco eficiente pero funcional de guardar los datos en un documento auxiliar, eliminar el documento original en caso de que exista y sustituir el nombre del auxiliar por el original. Se implementó esta técnica porque al momento de cargar los datos contenidos en un `.txt`, modificarlos y luego guardarlos nuevamente, estos generaban un especie de concatenación errónea. Es decir, si se tenían los datos 1, 2, 3 y eran cargados al árbol, posteriormente si se le modificaban o añadían datos, suponiendo 4, 5, 6, al momento de guardar nuevamente en el archivo de texto, se guardaban los datos anteriores "1,2,3" (la información original) más "1,2,3,4,5,6" (la información original concatenada con la nueva). A mí parecer no era agradable de ver y preferí evitar concatenaciones de información no planeadas. Para incluir los separadores de campo y de registro, se implementó un método al objeto `alumno` el cual retorna la concatenación de datos pertenecientes a los atributos del alumno con sus correspondientes separadores de campo y de registro. En pocas palabras, tras haber recibido la dirección de la raíz, se verifica si esta apunta a nulo y en caso de ser negativa la respuesta, se obtienen los datos a los que apunta la variable raíz (en este caso se obtienen datos tipo `alumno` con sus atributos llenos) y usando el objeto instanciado como `alumno` auxiliar, toma los datos de ese puntero para su posterior obtención de atributos con el método `.toString`, el cual no recibe parámetros y retorna los atributos del objeto concatenados con su separador de campo en una sola variable tipo `string`. Tal variable tipo `string` es guardada en el documento de texto y luego se siguen los mismos pasos del recorrido preorden (la raíz toma el valor de la izquierda recursivamente, luego de la derecha también recursivamente) y finalmente cuando termina la recursividad, se cierra el documento de texto.

- Cargar de archivo: Este método prácticamente reutiliza el algoritmo del programa 1, pues los datos del árbol anterior, al momento de guardarlos con el método de preorden, están ordenados en el documento .txt de manera idéntica a la que se insertaron los datos del anterior árbol y se logra una réplica exacta del árbol. En más detalles, se declara una variable tipo ifstream para operar con la lectura del archivo, 3 variables auxiliares tipo string y un objeto alumno auxiliar para insertar los datos obtenidos del documento y guardados en las variables string para su posterior asignación de atributos del objeto mediante el auxiliar alumno. Con una iteración preprueba, se repite la lectura y asignación de datos al objeto hasta que el documento .txt llega a su fin.
- Eliminación de datos: Este método a mi parecer no se logró implementar de la manera más eficiente posible pero es totalmente funcional. Se declararon; una variable alumno que serviría para operar con los atributos del mismo, 5 variables tipo puntero a nodo; uno para guardar la dirección del dato referencia solicitado inicialmente al usuario en el menú (más adelante se explica a detalle), otro para almacenar la dirección del padre del nodo a eliminar, un puntero temporal para almacenar la referencia de datos cuando un nodo tiene datos por la izquierda o derecha, un puntero al lado izquierdo de un nodo y otro al lado derecho.

Primeramente, se le solicita al usuario que digite un nombre de referencia para ser buscado en el árbol. Posteriormente, se le llama al método público “eliminar dato” que recibe un objeto tipo plantilla. Tal método le asigna a la variable del dato referencia, la dirección que retorna el método de *búsqueda de datos*, método el cual funciona simplemente comparando datos contenidos en el árbol con el dato referencia, aproximándose a los posibles datos mediante la comparación del atributo nombre perteneciente al objeto (si es mayor, recorre la parte derecha del puntero que se tiene, sino recorre la parte izquierda), en caso de no encontrar el dato, retorna una dirección nula. Entonces, tras haber asignado puntero a la variable dato referencia, se verifica si se encontró el dato. En caso de no haber sido encontrado solo se muestra un mensaje “dato no encontrado”, caso contrario, entra a diferentes validaciones para los varios casos que se presentan con la eliminación de datos en el árbol.

1. Si el dato a eliminar es la raíz y tiene un hijo por la izquierda o derecha.
  - a. Se verifica si la raíz es hoja/no tiene datos por la izquierda ni por la derecha. En caso de ser hoja, basta con eliminar el dato, asignar la raíz a un puntero nulo y salir del método.
  - b. Se verifica si la raíz solo tiene un hijo por la izquierda. Para este caso, solo se le asigna a la raíz el valor de su hijo por la izquierda y se elimina el dato raíz. Sin dejar pasar que el árbol está implementado con un puntero a padre, es necesario actualizar la nueva dirección del padre a la raíz que en este caso es sí mismo. Tras esto, sale del método.
  - c. Se verifica si la raíz solo tiene un hijo por la derecha y en caso de ser afirmativa la respuesta, el hijo de la derecha pasa a ser la nueva raíz, su padre es sí mismo, se elimina la raíz anterior y sale del método.
2. Si el dato es raíz con hijos tanto por izquierda como por derecha o si es diferente de la raíz, proceden las siguientes condiciones:
  - a. Si el dato es hoja, se busca por cuál lado del papá se encuentra el dato a eliminar y se apunta a nulo, es decir, si el padre por su lado izquierdo coincide

con el dato a eliminar, se sabe que el dato a eliminar está por el lado izquierdo del padre, entonces a la dirección del padre por su lado izquierdo se le asigna nulo y así se logra desreferenciar el dato a eliminar o caso contrario, sucede lo mismo pero por su desreferenciando el lado derecho. Antes de salir de la llamada al método, se libera memoria eliminando el dato y asignando su puntero a nulo.

- b. Si el dato es nodo con un hijo:
  - i. En caso de tener el hijo por la izquierda, al padre del nodo a eliminar se le asigna la dirección siguiente del nodo a eliminar. Se liberan datos y se le asigna el nuevo padre al nodo siguiente de eliminar.
  - ii. En caso de tener un hijo por la derecha sucede lo mismo pero en sentido contrario, al padre del nodo a eliminar se le asigna la dirección del nodo a eliminar por su lado derecho y se le reasigna el padre.
- c. Si el dato tiene hijos por ambos lados, se guardan las direcciones de los nodo a los que apunta el nodo a eliminar (el izquierdo y el derecho), luego se busca el mayor de los menores/el mas derecho de los izquierdos y se le asigna a un puntero auxiliar. Posteriormente al nodo a eliminar se le asignan los datos del puntero auxiliar (en este caso un dato tipo T/alumno), se elimina puntero temporal el cual almacenaba la dirección de los datos ahora asignados al puntero que se busca eliminar, se libera memoria y finalmente, se religan anteriores direcciones inicialmente guardadas del nodo a eliminar, esto para evitar un religado completo de todos los nodos. Tras esto, se detiene el método y el dato ya puede considerarse eliminado.

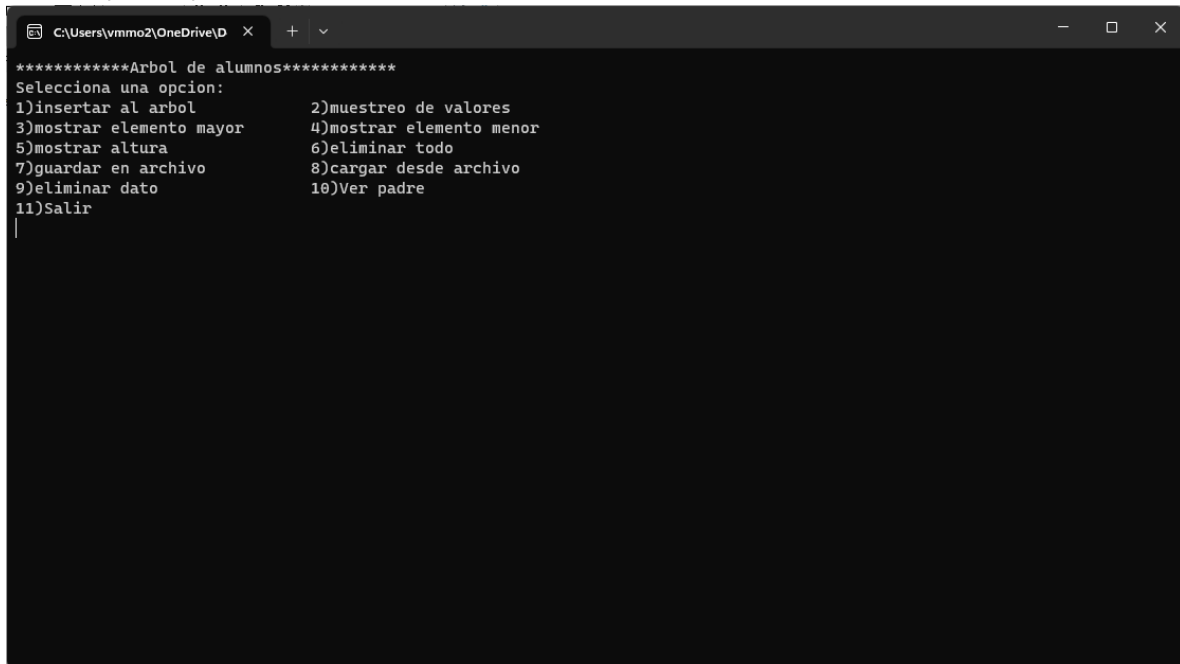
- Ver padre: Para este método solo se le solicita al usuario un nombre referencia, el cual es asignado a un objeto tipo alumno y este objeto tipo alumno es mandado como parámetro al método de "verPadre" el cual, reutilizando el ya mencionado y explicado "búsqueda de datos," verifica primero si no se encontró el dato (cuando la variable puntero a nodo auxiliar es nula). Si sí se encontró el dato, verifica si el dato referencia coincide con la raíz del árbol, si es así, se menciona tal hecho y se muestra la información que almacenan los atributos del objeto al que apunta la raíz y si ninguno de los anteriores casos coincide, es imprime la información que almacenan los atributos del objeto al que apunta el puntero auxiliar.

El manejo de los datos en el documento de texto...

La información en el TDA se guarda de forma individual, es decir, no se concatena la información de cada campo con su delimitador de campo o con su delimitador de registro. A diferencia de los archivos .txt que insertan la información de cada campo almacenado en el nodo junto con el delimitador de campo y el delimitador de registro. Algo peculiar en la manera de almacenar la información de los archivos es que toda la información de cada campo se guarda de forma continua, es decir, sin saltos de línea, esto para darle un uso a los delimitadores de registro, pues cada que aparecen el algoritmo de lectura lo interpreta como un salto de línea y de esta manera no inserta todos los campos concatenados en la lista.

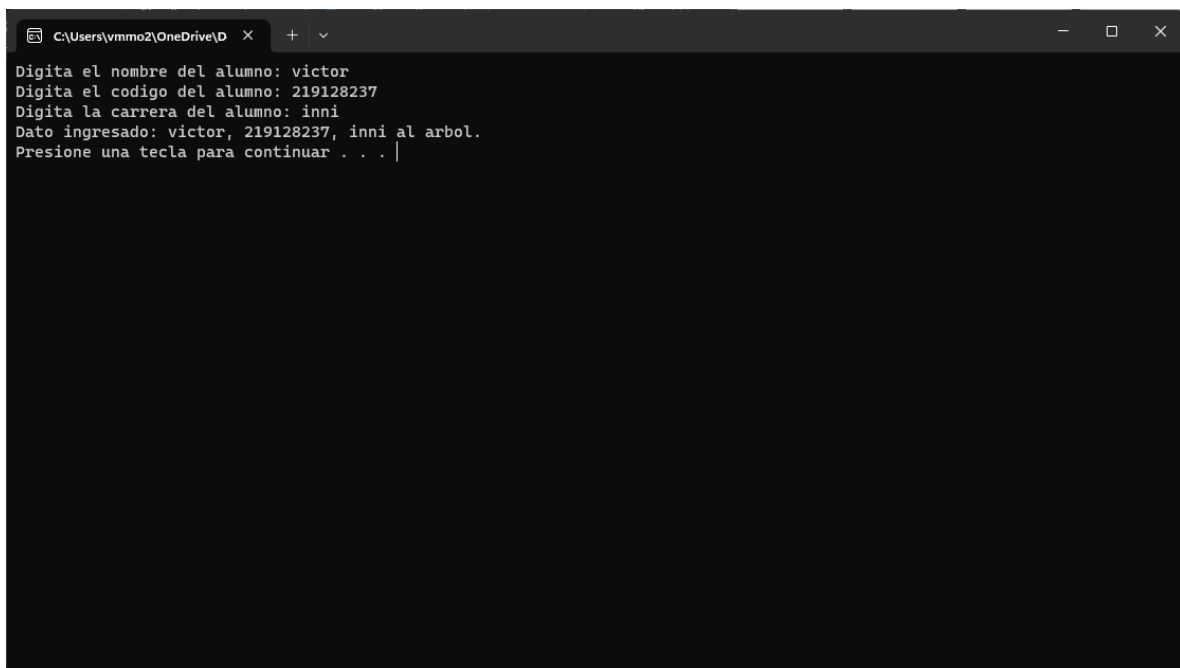
# Impresiones de pantalla

## Menu principal



```
C:\Users\vmmo2\OneDrive\D x + v
*****Arbol de alumnos*****
Selecciona una opcion:
1)insertar al arbol          2)muestreo de valores
3)mostrar elemento mayor    4)mostrar elemento menor
5)mostrar altura            6)eliminar todo
7)guardar en archivo        8)cargar desde archivo
9)eliminar dato             10)Ver padre
11)Salir
|
```

## Inserción



```
C:\Users\vmmo2\OneDrive\D x + v
Digita el nombre del alumno: victor
Digita el codigo del alumno: 219128237
Digita la carrera del alumno: inni
Dato ingresado: victor, 219128237, inni al arbol.
Presione una tecla para continuar . . . |
```

## Menu recorridos

```
C:\Users\vmmo2\OneDrive\D x + v
*****Arbol de alumnos*****
Selecciona una opcion:
1)insertar al arbol          2)muestreo de valores
3)mostrar elemento mayor    4)mostrar elemento menor
5)mostrar altura            6)eliminar todo
7)guardar en archivo        8)cargar desde archivo
9)eliminar dato             10)Ver padre
11)Salir
2
preorder [1]
inorder [2]
postorder[3]
|
```

Datos en preorden

```
C:\Users\vmmo2\OneDrive\D x + v
Muestreo de los valores en PreOrder:
victor, 219128237, inni
1, 1, 1
2, 2, 2
3, 3, 3
4, 4, 4
5, 5, 5
Presione una tecla para continuar . . . |
```

Datos inorden

```
C:\Users\vmimo2\OneDrive\D  X + v
Muestreo de los valores en InOrder:
1, 1, 1
2, 2, 2
3, 3, 3
4, 4, 4
5, 5, 5
victor, 219128237, inni
Presione una tecla para continuar . . . |
```

Datos en postorden

```
C:\Users\vmimo2\OneDrive\D  X + v
Muestreo de los valores en PostOrder:
5, 5, 5
4, 4, 4
3, 3, 3
2, 2, 2
1, 1, 1
victor, 219128237, inni
Presione una tecla para continuar . . . |
```

Mostrar elemento mayor



```
C:\Users\vmimo2\OneDrive\D  X + v - □ X
El elemento mayor del arbol es: victor, 219128237, inni
Presione una tecla para continuar . . . |
```

Mostrar elemento menor

```
C:\Users\vmimo2\OneDrive\D  X + v - □ X
El elemento menor del arbol es: 1, 1, 1
Presione una tecla para continuar . . . |
```

Menú altura

```
C:\Users\vmimo2\OneDrive\D  X + v - □ X
Altura de arbol...
derecho [1]
izquierdo [2]?
|
```

Altura de árbol por su lado derecho

```
C:\Users\vmimo2\OneDrive\D  X + v - □ X
La altura correspondiente al subarbol derecho respecto de la raiz es: 0
Presione una tecla para continuar . . . |
```

Altura de árbol por su lado izquierdo

```
C:\Users\vmimo2\OneDrive\D  X + v - □ X
Altura de arbol...
derecho [1]
izquierdo [2]?
2
La altura correspondiente al subarbol izquierdo respecto de la raiz es: 5
Presione una tecla para continuar . . . |
```

Eliminar todo el árbol

```
C:\Users\vmimo2\OneDrive\D  X + v - □ X
Eliminando todo el contenido del arbol...
Presione una tecla para continuar . . . |
```

```
C:\Users\vmimo2\OneDrive\D  X + v - □ X
Eliminado :)
Presione una tecla para continuar . . . |
```

```
C:\Users\vmimo2\OneDrive\D  X + v - □ X
Arbol vacio, intenta insertar datos
Presione una tecla para continuar . . . |
```

Guardar datos

```
C:\Users\vmimo2\OneDrive\D  X + v
*****Arbol de alumnos*****
Selecciona una opcion:
1)insertar al arbol          2)muestreo de valores
3)mostrar elemento mayor    4)mostrar elemento menor
5)mostrar altura            6)eliminar todo
7)guardar en archivo        8)cargar desde archivo
9)eliminar dato            10)Ver padre
11)Salir
7
Dato: 1|1|1, Dato: 2|2|2, Dato: 3|3|3, Dato: 4|4|4, Dato: 5|5|5, Dato: victor|219128237|inni, Datos guardados exitosamente...
Presione una tecla para continuar . . . |
```

Cargar datos desde archivo

```
C:\Users\vmimo2\OneDrive\D  X + v
*****Arbol de alumnos*****
Selecciona una opcion:
1)insertar al arbol          2)muestreo de valores
3)mostrar elemento mayor    4)mostrar elemento menor
5)mostrar altura            6)eliminar todo
7)guardar en archivo        8)cargar desde archivo
9)eliminar dato            10)Ver padre
11)Salir
8
Datos cargados exitosamente...
Presione una tecla para continuar . . . |
```

Eliminar dato

```
C:\Users\vmmo2\OneDrive\D x + v
*****Arbol de alumnos*****
Selecciona una opcion:
1)insertar al arbol          2)muestreo de valores
3)mostrar elemento mayor    4)mostrar elemento menor
5)mostrar altura            6)eliminar todo
7)guardar en archivo        8)cargar desde archivo
9)eliminar dato            10)Ver padre
11)Salir
9
Digita el nombre del alumno a eliminar:
victor
Dato encontrado: victor
Nodo padre: 5, 5, 5
El dato es hoja.
Dato eliminado :)
Presione una tecla para continuar . . . |
```

## Mostrar padre

```
C:\Users\vmmo2\OneDrive\D x + v
*****Arbol de alumnos*****
Selecciona una opcion:
1)insertar al arbol          2)muestreo de valores
3)mostrar elemento mayor    4)mostrar elemento menor
5)mostrar altura            6)eliminar todo
7)guardar en archivo        8)cargar desde archivo
9)eliminar dato            10)Ver padre
11)Salir
10
Digita un nombre de referencia:
1
Nodo raiz, el padre es el mismo elemento...
1, 1, 1
Presione una tecla para continuar . . . |
```

```
C:\Users\vmimo2\OneDrive\D  X + v
*****Arbol de alumnos*****
Selecciona una opcion:
1)insertar al arbol          2)muestreo de valores
3)mostrar elemento mayor    4)mostrar elemento menor
5)mostrar altura            6)eliminar todo
7)guardar en archivo        8)cargar desde archivo
9)eliminar dato             10)Ver padre
11)Salir
10
Digita un nombre de referencia:
victor
Nodo padre: 5, 5, 5
Presione una tecla para continuar . . . |
```