**Documento de pruebas unitarias programa Arboles Binarios.**

Autor: Diego Alfonso Vega Fernández

Email: [davf64@yahoo.com](mailto:davf64@yahoo.com)

Fecha: 2021-05-24

GIT: <https://github.com/davf64/arbolesBinariosDV>

# 1. Lista de pruebas unitarias:

## 1.1 Crear un árbol y visualizarlo.

Se crea un árbol binario, se agregan nodos y se imprime por pantalla.

## 1.2. Búsqueda de ancestros comunes.

Dados dos números enteros, se busca el menor ancestro común entre ellos.

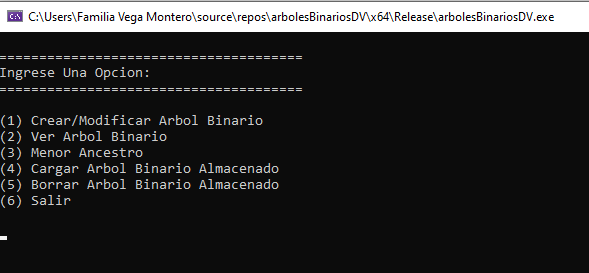
## 1.3. Carga de un árbol previamente guardado.

Se realiza la carga en memoria de un árbol previamente guardado en la base de datos.

## 1.4. Eliminar un árbol binario previamente guardado.

Se realiza la carga en memoria de un árbol previamente guardado en la base de datos.

Menú principal de la aplicación:

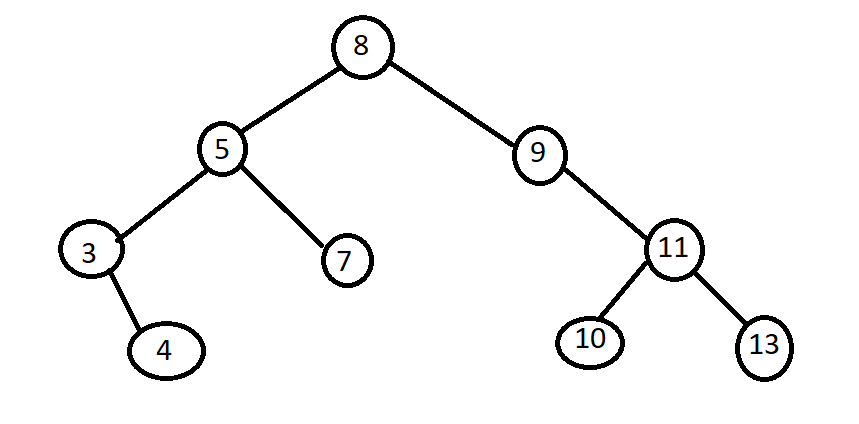


Se debe ingresar un numero entre el 1 y el 6 seguido de la tecla enter.

# 2. Despliegue de las pruebas:

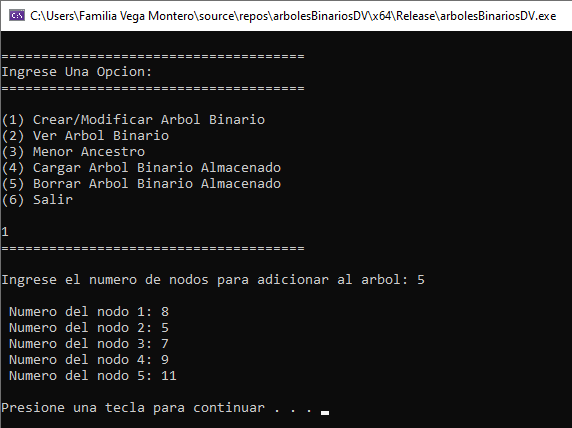
## 2.1 Crear un árbol y visualizarlo.

Ingresaremos el siguiente árbol binario de ejemplo:

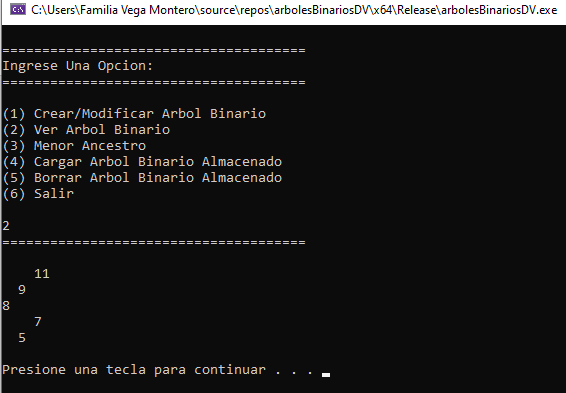


Se realizará la carga del nodo raiz y 4 nodos y se visualizarían, luego realizamos la carga de los otros 4 nodos restantes, esto para demostrar que el árbol es escalable.

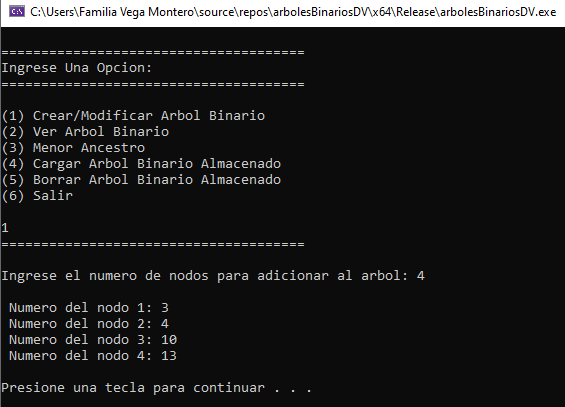
*NOTA: Por conveniencia los arboles binarios se imprimirán de manera horizontal de izquierda a derecha desde la raiz a las hojas.*



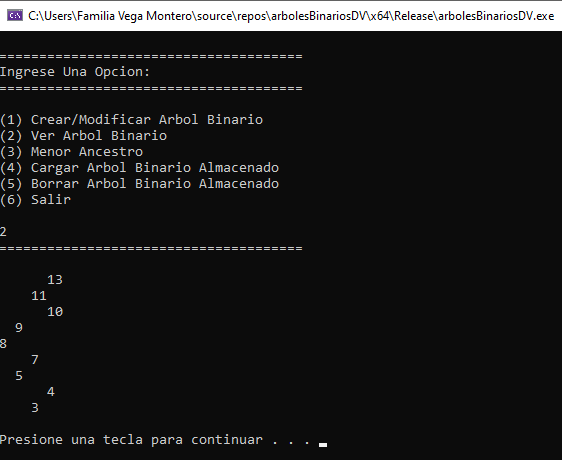
Con opción 2 imprimimos el árbol binario parcial.



Ingresamos los valores restantes, nuevamente ingresamos a la opción 1.

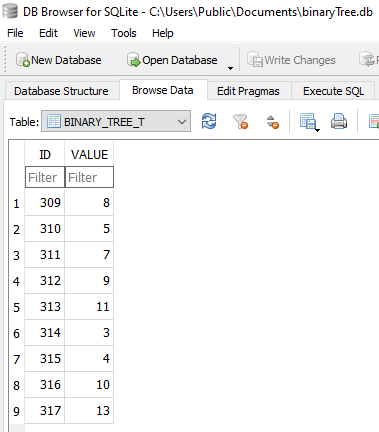


Visualizamos el árbol final:



Apoyado del programa “DB Browser for SQLite” abrimos el archivo de la base de datos.

(El programa la genera en el directorio genérico C:\Users\Public\Documents\binaryTree.db)

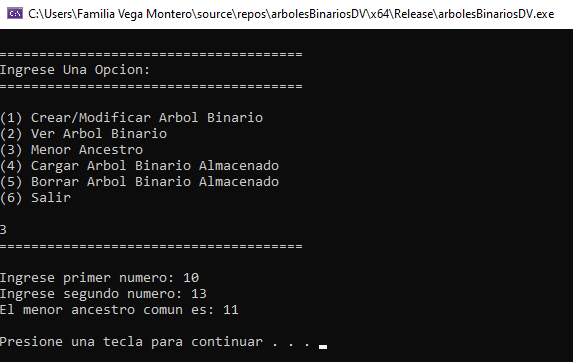


NOTA: En la base de datos se crea una tablita llamada BINARY\_TREE\_T en la que se van guardando los valores, junto a un ID auto generado, con la premisa que el valor con menor ID es el que representa la raíz del árbol binario.

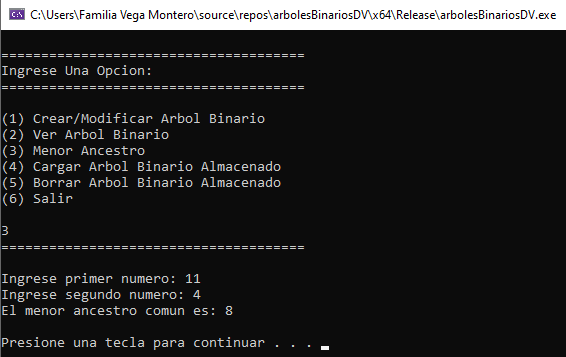
## 2.2. Búsqueda de ancestros comunes.

Dados dos números enteros, se realizarán búsquedas del menor ancestro común (MCA en inglés).

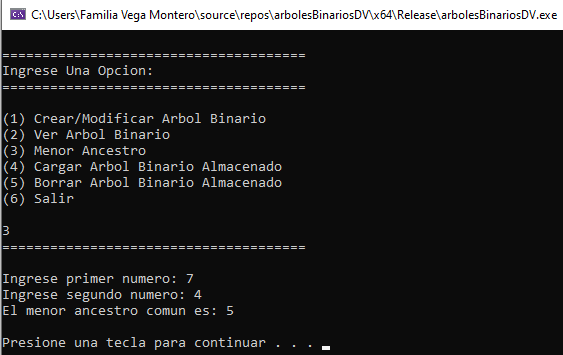
### 2.2.1 MCA (10,13) = 11



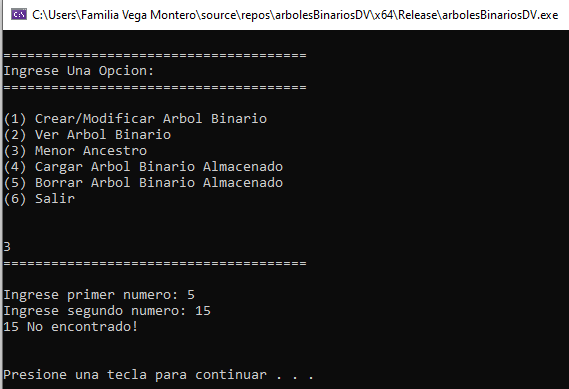
### 2.2.2 MCA (11,4) = 8



### 2.2.3 MCA (7,4) = 5

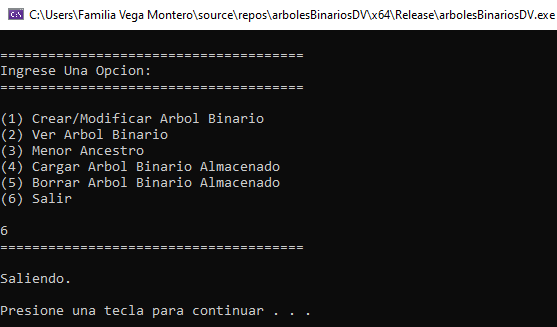


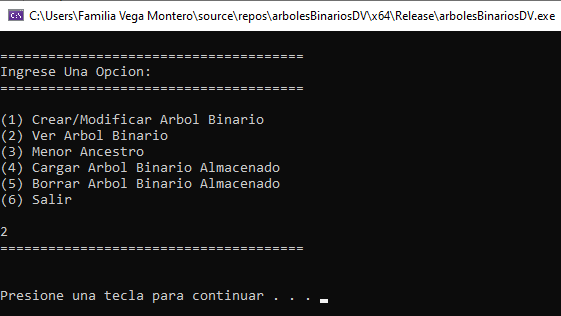
### 2.2.4 MCA (5,15) = N/A, el numero 15 no existe en el árbol binario.



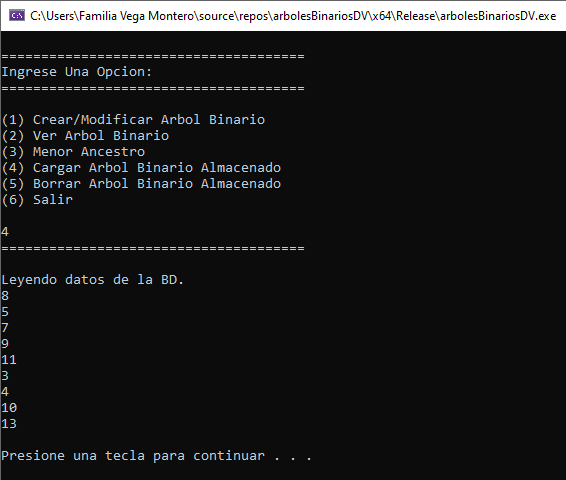
## 2.3. Carga de un árbol previamente guardado.

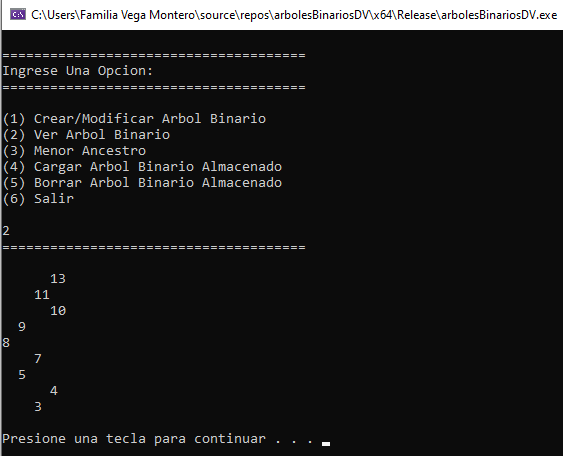
Para esta prueba reiniciamos la ejecución del programa y visualizamos con la opción 2 que no existe el árbol binario.





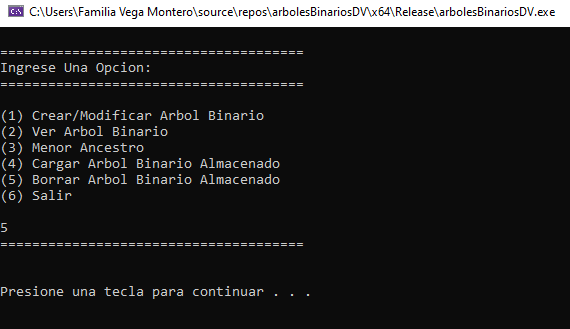
Ingresamos a la opción 4 y de nuevo la opción 2.



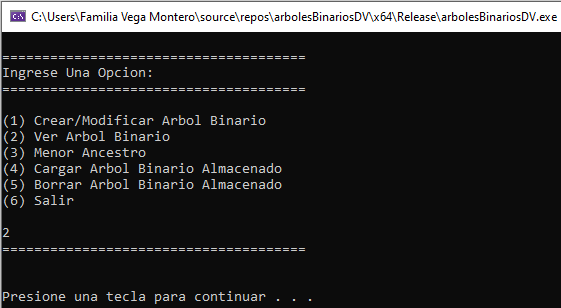


## 2.4. Eliminar un árbol binario previamente guardado.

Para realizar esta acción ingresamos a la opción 5, esto borrara el árbol binario tanto de la base de datos, como de la memoria del sistema.



Con la opción 2 comprobamos que ya no hay árbol binario



Por medio del programa “DB Browser for SQLite” abrimos el archivo de la base de datos y verificamos que la tabla se encuentra vacía.

