

## **PRÁCTICA 2**

Para redactar el informe de la segunda práctica de la asignatura de *Estructuras de Datos y Algoritmos* hemos dividido el contenido en tres grandes apartados. En primer lugar, capturas de pantalla de la ejecución, donde se reunirán diferentes capturas de pantalla que muestran el funcionamiento del programa. En segundo lugar, una guía de ejecución del programa para dar a conocer su correcto funcionamiento y composición. Y, por último, la justificación de por qué no hemos realizado las preguntas bonus.

### **1. Capturas de pantalla de la ejecución.**

#### **• Inicio del programa y menú principal.**

En la pantalla de inicio del programa se le da la bienvenida al usuario y se le proponen seis opciones diferentes a escoger, relacionadas con un número del 1 al 6, que el usuario deberá introducir para hacerle saber al programa qué acción quiere realizar. Las acciones disponibles que vemos en la Figura 1 las analizaremos en profundidad en los siguientes apartados del punto 1.

```
~~~~~  
Bienvenido al simulador TAD Árbol Binario de Búsqueda Equilibrado (AVL).  
A continuación se muestran por pantalla las opciones de las cuales dispone.  
  
1.- Insertar alumno.  
2.- Buscar un alumno por su NIA.  
3.- Mostrar los Nias de alumnos en recorrido Preorden.  
4.- Mostrar los Nias de alumnos en recorrido Inorden.  
5.- Mostrar los Nias de alumnos en recorrido Postorden.  
6.- Salir del programa.  
  
Introduzca el número correspondiente a la acción que desea realizar.  
~~~~~
```

**Figura 1.**

#### **• Insertar alumnos.**

La primera opción es insertar alumnos. Primero se le pide al usuario el número de alumnos que desea inscribir para agilizarle la faena en caso que el número sea superior a uno. A continuación, se pide el nombre del alumno, su número de identificador personal (NIA) y el número de teléfono. Cabe decir que estas tres variables tienen ciertas restricciones, como el espacio en memoria del nombre del alumno, que está limitado a 100 caracteres, o el número de teléfono, que debe ser un entero de nueve cifras y no puede contener ningún otro carácter. Así pues, si el usuario introduce valores erróneos, se le informará con un mensaje de error. Como podemos ver en la Figura 2, hemos escogido la opción de insertar dos alumnos a modo de ejemplo.

## PRÁCTICA 2

Estructuras de Dades i Algorismes

Diego Sáez y Ana Freire

Óscar Font

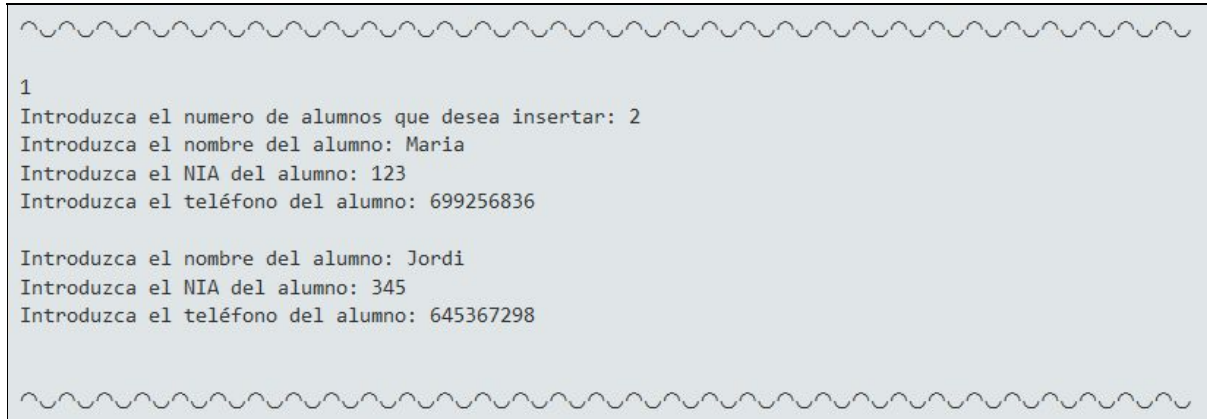
Marc Canal

Lucía Gasión

183826

183867

183849



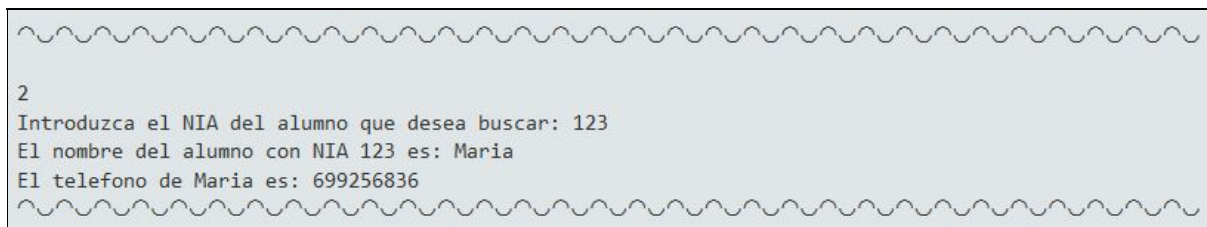
```
1
Introduzca el numero de alumnos que desea insertar: 2
Introduzca el nombre del alumno: Maria
Introduzca el NIA del alumno: 123
Introduzca el teléfono del alumno: 699256836

Introduzca el nombre del alumno: Jordi
Introduzca el NIA del alumno: 345
Introduzca el teléfono del alumno: 645367298
```

Figura 2.

- **Buscar un alumno a partir de su NIA.**

La segunda opción consiste en buscar a un alumno en concreto a partir de su número de identificación (NIA). Así pues, se le pide al usuario el NIA en cuestión. En la Figura 3 hemos usado como ejemplo el NIA de Maria, la misma alumna que hemos insertado en la Figura 2 del apartado anterior. Como podemos ver en la Figura 3, al introducir el número se imprime por pantalla el nombre del alumno y su número de teléfono.




```
2
Introduzca el NIA del alumno que desea buscar: 123
El nombre del alumno con NIA 123 es: Maria
El telefono de Maria es: 699256836
```

Figura 3.

- **Mostrar los NIAs de los alumnos en el recorrido Preorden.**

La tercera opción consiste en mostrar por pantalla el recorrido en Preorden del árbol AVL que se va creando a medida que insertamos alumnos. Recordemos que el Preorden sigue las siguientes características: raíz, hijo izquierdo, hijo derecho. Para disponer de más nodos y ver mejor el ejemplo que mostramos en la Figura 4, hemos usado los dos alumnos insertados previamente, juntamente con tres alumnos más: Pedro (NIA 567), David (NIA 789) y Mireia (NIA 023).



```
3
345 123 23 567 789
```

Figura 4.

- **Mostrar los NIAs de los alumnos en el recorrido Inorden.**

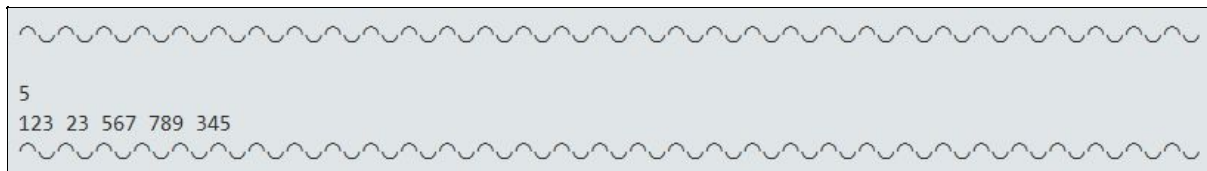
La cuarta opción consiste en lo mismo que la anterior pero esta vez con recorrido Inorden. Recordemos que el Inorden sigue las siguientes características: hijo izquierdo, raíz, hijo derecho. El ejemplo que se muestra en la Figura 5 se obtiene a partir de los mismos cinco alumnos tratados previamente.



**Figura 5.**

- **Mostrar los NIAs de los alumnos en el recorrido Postorden.**

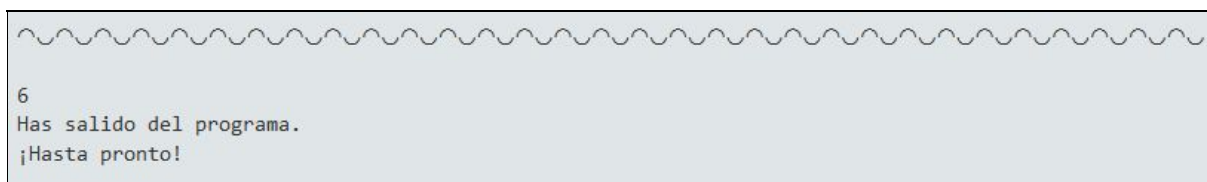
La quinta opción consiste en lo mismo que la anterior pero esta vez con recorrido Postorden. Recordemos que el Postorden sigue las siguientes características: hijo izquierdo, hijo derecho, raíz. El ejemplo que se muestra en la Figura 6 se obtiene a partir de los mismos cinco alumnos tratados previamente.



**Figura 6.**

- **Salir del programa.**

El menú principal se seguirá imprimiendo cada vez que se finalice una acción hasta que introduzcamos el número 6, número que hace que el while con el que está montada la estructura se pare y se imprima un mensaje para darle una cordial despedida al usuario.



**Figura 7.**

## 2. Guía de Ejecución.

### • Diseño general.

El programa está compuesto por tres ficheros de código, relacionados entre ellos: `main.c`, `Funciones.c` y `definiciones.h`.

El fichero `main.c` contiene la función principal del programa, aquella con la que se decide qué acción hacer según el número que se introduce por pantalla. Dicha función está compuesta por un `while` y un `switch`, mediante los cuales se crea un bucle que sólo se puede anular al introducir un 6, que es la opción de salir del programa, por lo tanto, el `while` se ejecuta siempre y cuando el número introducido no sea el 6.

El fichero `Funciones.c`, como bien podemos deducir del nombre, contiene todas las funciones que hemos ideado para la práctica. Las funciones en cuestión son muy diversas: desde funciones que detectan los posibles errores a la hora de introducir valores por teclado, pasando por distintas impresiones como menús y explicaciones, hasta todas las posibles rotaciones necesarias para un árbol AVL.

El fichero `definiciones.h` está compuesto por diferentes sub-apartados. Primeramente, todas las librerías `#include` de C necesarias para las funciones de los dos ficheros anteriores, después un seguido de definiciones `#define` para las funciones booleanas, para las dimensiones de los strings y para clasificar los errores. A continuación, la estructura que hemos utilizado, que recibe el nombre `nodo` y está compuesta por las variables que representan el nombre del alumno, el teléfono del alumno, el NIA del alumno, la altura del árbol y el hijo derecho y el hijo izquierdo de los nodos. Por último, se encuentran las declaraciones de todas las funciones del fichero `Funcions.c`, para evitar declararlas en otro fichero y así poder tenerlo todo más ordenado y visual.

Además, como hemos dicho previamente, los tres ficheros están conectados entre ellos para poder funcionar correctamente. Hemos realizado una representación esquemática para entender la relación, como podemos ver en la Figura 8.

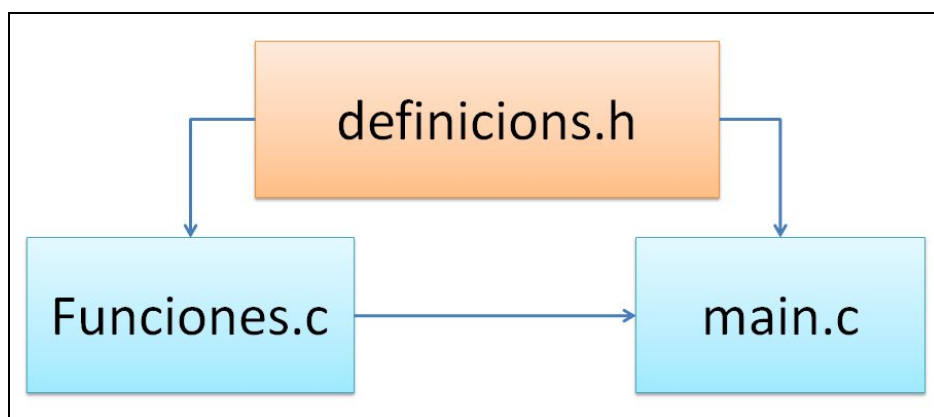


Figura 8.

## PRÁCTICA 2

Estructuras de Dades i Algorismes

Diego Sáez y Ana Freire

Óscar Font

Marc Canal

Lucía Gasión

183826

183867

183849

### • Errores.

Para evitar que el programa tenga fallos de datos, hemos ideado un sistema de printado de errores que aparecen por pantalla cada vez que el usuario introduce algún valor erróneo. El método que hemos utilizado ha sido definir todos los posibles errores de forma general en el fichero `definicions.h` con un `#define` delante. A continuación hemos realizado una tabla que recopila todos los errores posibles, el mensaje que imprimen y la explicación del por qué hemos definido dichos errores.

Nombre	Mensaje	Explicación
ERRORNUM	ERROR. Por favor, introduzca un número dentro del rango.	Error general para todas aquellas funciones que trabajan con números y tienen un cierto rango, como por ejemplo en el menú principal.
ERRORNOM	ERROR. Por favor, introduzca un nombre sin números ni caracteres especiales.	Error específico para el nombre del alumno, el cual no puede contener números ni caracteres especiales como "\$", entre otros.
ERRORNIA	ERROR. El identificador del alumno debe ser de entre 3 y 4 cifras.	Error específico para el NIA del alumno, el cual debe ser un número entero de entre tres y cuatro cifras (si NIA < 3 cifras, entonces rellenar con 0).
ERRORNIA2	ERROR. El NIA no debe contener ni letras ni caracteres especiales.	Error específico para el NIA del alumno, el cual no debe contener letras ni caracteres especiales.
ERRORTELE	ERROR. El número de teléfono del alumno debe ser un número de 9 cifras.	Error específico para el número de teléfono del alumno, que debe ser un entero de nueve cifras exclusivamente.
ERRORTELE2	ERROR. El número de teléfono no debe contener ni caracteres especiales ni letras.	Error específico para el número de teléfono del alumno, que no puede contener letras ni caracteres especiales.
ERRORARB	ERROR. El árbol está vacío.	Error específico para cuando el árbol está vacío, como por ejemplo cuando se quiere imprimir el preorden y no hay nodos que mostrar.
ERRORREP	ERROR. El NIA introducido ya existe, introduzca otro NIA por favor.	Error específico para cuando se quiere insertar un alumno con un NIA que ya existe.
ERRORALUM	ERROR. El Alumno solicitado no se encuentra en la base de datos.	Error específico para cuando se intenta buscar un alumno y éste no ha sido insertado previamente.

**3. Bonus.**

En el enunciado de la práctica se nos pedían dos cuestiones bonus, es decir, opcionales. La primera consistía en implementar una función capaz de borrar un nodo escogido por el usuario y la segunda en imprimir el árbol creado.

**• Función borrar.**

Para implementar la función borrar teníamos diferentes ideas, la mayoría de ellas encontradas por internet. Tuvimos problemas a la hora de entender los códigos que habíamos encontrado, puesto que usaban funciones en lenguaje C que aún no hemos dado en nuestras lecciones de Programación. Así pues, tras estar cerca de conseguir el propósito de implementar una función que borrara los nodos y equilibrara de nuevo el árbol, hemos decidido prescindir de éste apartado.

**• Función imprimir.**

Para la función imprimir estuvimos pensando muchas formas de poder imprimir el árbol. Estuvimos estudiando diseños y llegamos a la conclusión que los que se muestran en la Figura 9 eran los más acertados.

**Figura 9.**

Pero empezaron a surgir problemas que nos dificultaron mucho la realización del código, juntamente con el hecho de trabajar a contrarreloj. Primeramente, en los dos árboles de la Figura 9 había un grave problema: si el nodo 3 tuviera que tener un hijo izquierdo, éste se sobre-pondría con el hijo derecho del nodo 2. En el árbol de la izquierda aún se podría hacer un arreglo, pero en el árbol de la derecha, tal y como pensábamos, es imposible saber si el nodo 5 es hijo derecho de 2 o hijo izquierdo de 3.

Había que pensar una solución para el punto 1 y, la verdad es que no avanzamos mucho. Estuvimos pensando en hacerlo con tabuladores en función de la profundidad o altura del árbol. Llegamos incluso a pensar una función que calculaba la cantidad de espacios que había que imprimir por nivel, pero se nos estaba yendo de las manos.

A la desesperada, hicimos una profunda búsqueda en Google sobre la impresión de árboles AVL en lenguaje C y C++, pero presentaban unos criterios demasiado avanzados para nuestro nivel. Finalmente, encontramos algunas opciones en lenguaje Java pero, ni tenemos nivel suficiente para programar en Java ésta práctica, ni disponíamos del tiempo necesario para traducir todo el código ya hecho en C, así que simplemente decidimos descartar el hecho de realizar la segunda función bonus.