Estructuras de datos

Estefanía Martín Rojas

73133122-E

Universidad de Alcalá de Henares

Práctica 2 – Ábaco

curso 2017 - 18

1. **Detalles y justificación de la implementación:**
   1. **Especificación concreta de la interfaz de los TAD’s implementados.**
      1. **TAD’s creados.**

* Lista
* Nodo de la lista
* Pila
* Nodo de la pila
* Bolita
* Árbol (binario de búsqueda)
* Nodo del árbol
* Abaco
  + 1. **Definición de las operaciones TAD (Nombre, argumentos y retorno).**
* **Lista:**
  + Lista();
  + ~Lista();
  + void insertarNodo(Pila columna, char nombre, char tipo);
  + bool comprobarExistenciaColumna(char buscado);
  + bool listaVacia ();
  + void borrarActual();
  + bool borrarColumna(char buscado);
  + void mostrarLista();
  + void LmeterBolas(char buscado, string color);
  + void apilarActual(string color);
  + void LsacarBolas(char buscado);
  + void desapilarActual();
  + int LtamanoPila(char buscado);
  + int tamanoActual();
  + void concatenarListas(Lista& l1, Lista& l2)
* **Nodo de la lista:**
  + NodoLista(Pila colum, char nom);
  + char getNombre();
  + Pila getColumna();
* **Pila**:
  + Pila();
  + ~Pila();
  + void apilar(string color);
  + void desapilar();
  + bool esVacia();
  + string verCima();
  + int tamanoPila();
  + pnodoPila getcima();
* **Nodo de la pila:**
  + NodoPila(string colorBola, NodoPila \*sig);
  + string getColorPila();
* **Bolita:**
  + Bolita();
  + Bolita(string color);
  + string getColor();
* **Arbol:**
  + Arbol();
  + bool esVacio(NodoArbol \* nodo);
  + NodoArbol \* verRaiz();
  + void insertarNodo(long numero);
  + void inOrden(NodoArbol \* nodo);
  + void preOrden(NodoArbol \* nodo);
  + void postOrden(NodoArbol \* nodo);
  + bool buscarNumero(long num);
  + void mostrarArbol(NodoArbol \* nodo, int indent);
* **NodoArbol:**
  + NodoArbol(long numero);
* **Abaco:**
  + abaco();
  + void introducirColumnas();
  + void introducirBolitasEnAbaco(int un,int de,int ce,int unm, int dem);
  + int separar\_numero(int& numero, int& dm, int& um, int& c, int& d, int& u);
  + void verEstadoPilas();
  + void vaciarLista();
  + bool abacoEstaVacio(Lista& l);
  + void juntarAbacos(Lista& lista1, Lista& lista2);
  + long bolasANumero(Lista& list);
  + void introducirNumeroArbol(long num);
  + void imprimirArbol();
  + bool buscarNumeroEnArbol(long num);
  1. **Solución adoptada: explicación de la implementación.**

Para poder realizar esta práctica, he creado un array de 10 ábacos, es decir, un array de listas. Para la concatenación de dos listas, se llama al método concatenarListas en la clase Lista, se pasan dos listas del array y la segunda se une al final de la primera, desapareciendo ésta.

Para la implementación del árbol de búsqueda, se ha creado un árbol con nodos que incluyen un número de tipo long que corresponderá con el valor de su respectivo ábaco. El programa después es capaz de imprimir el árbol de una manera limpia para su fácil comprensión y además buscar un número concreto en él.

* 1. **Explicación de los métodos más destacados.**

En cuanto a la clase **Lista**, los más importantes son los siguientes:

* insertarNodo: gracias a este método, podemos crear una columna en el ábaco que después contendrá bolitas que podremos meter o sacar.
* LmeterBolas/LsacarBolas: gracias a este método, desde la lista, podemos meter o sacar bolas en una pila que nosotros queramos.
* LtamanoPila: gracias a este método sabremos cuantas bolas hay en una pila.

En cuanto a la **Pila**, los más importantes son:

* Apilar/desapilar: puede meter o sacar bolitas.
* tamanoPila: devuelve el número de bolitas que hay en la pila.

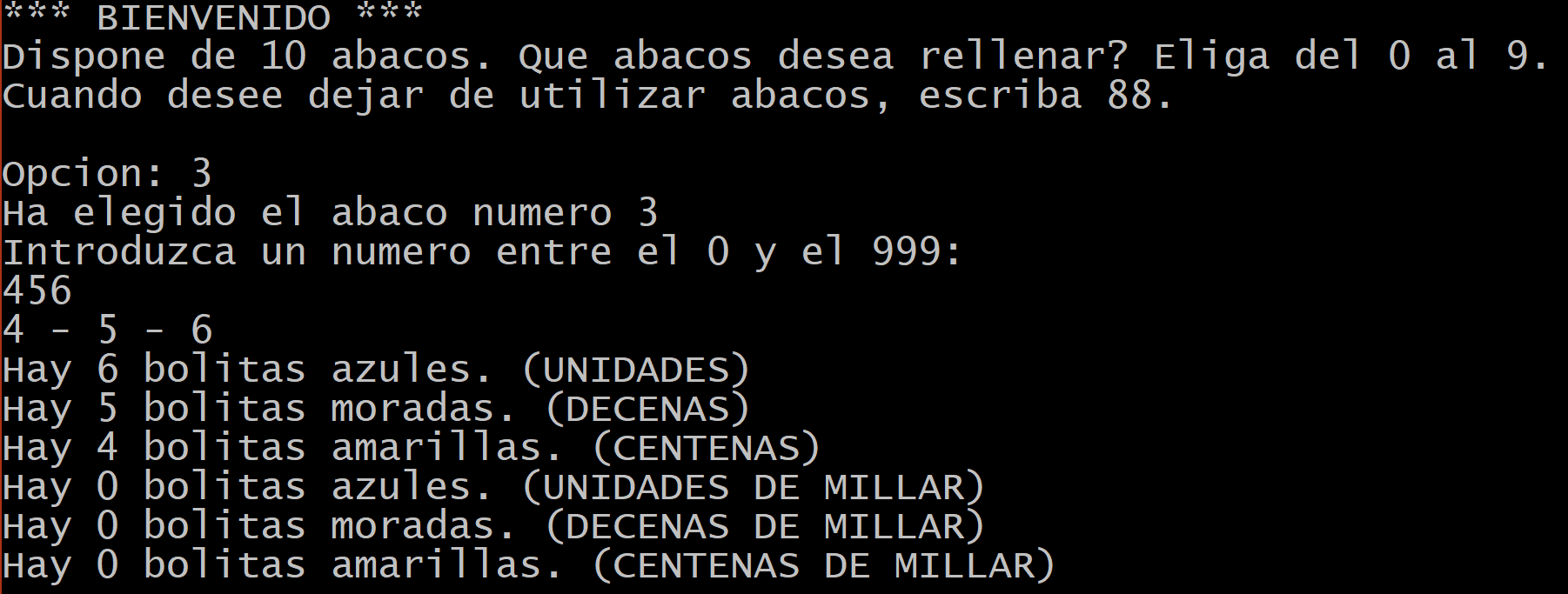
En la clase **Árbol**:

* buscarNumero: es el método encargado de buscar un número concreto que desee el usuario. Si el número existe, devuelve un True, si no, un False.
* mostrarArbol: imprime el estado actual del árbol de manera correcta de fácil comprensión.

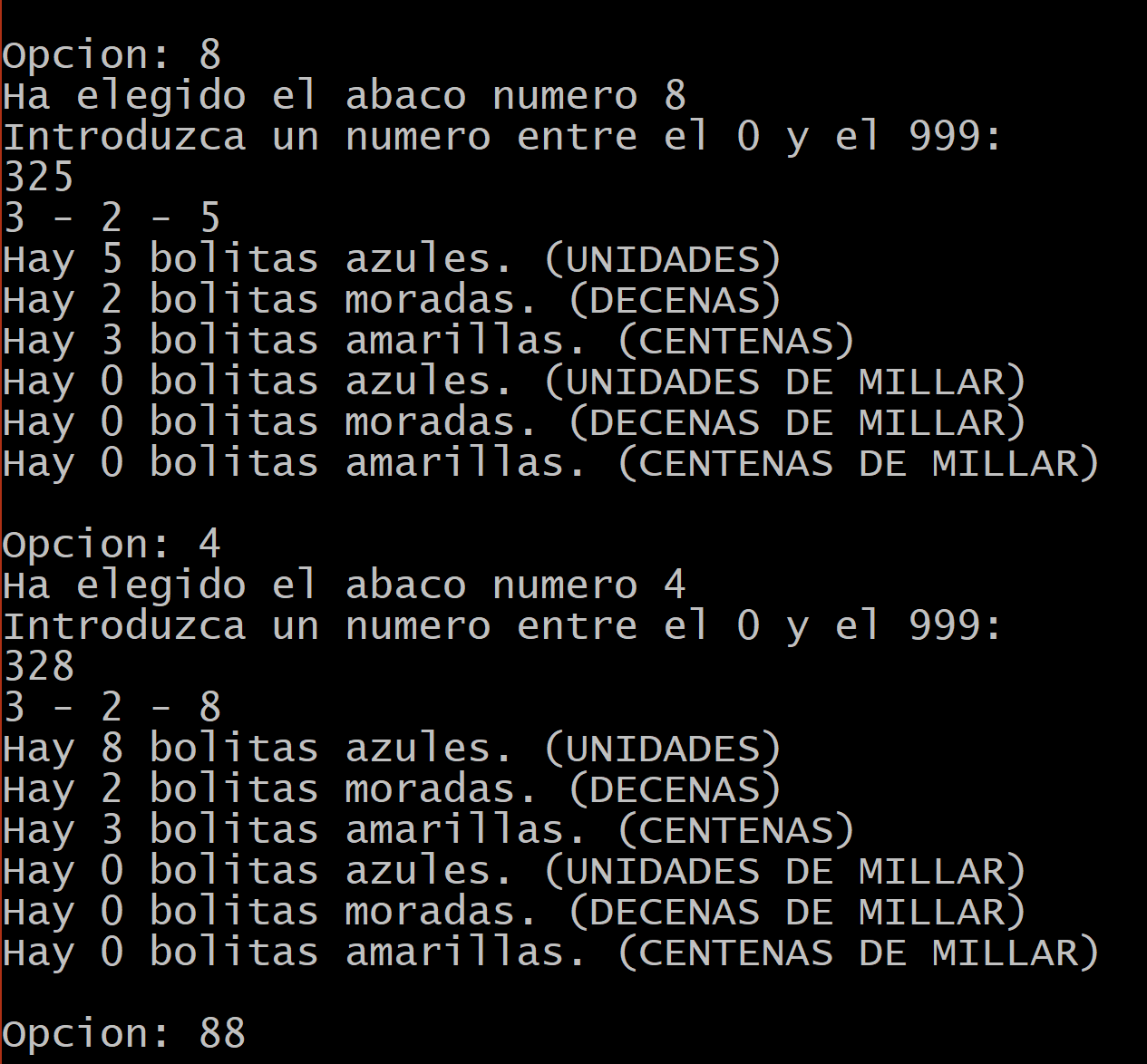
Por último, en la clase **MAS IMPORTANTE** que es **Abaco**, los métodos más importantes son:

* separar\_numero: en este método introducimos un numero entero y nos devuelve ese número partido en unidades, decenas, centenas… Esencial para saber cuántas bolitas corresponden a cada columna.
* juntarAbacos: que es el método encargado de concatenar dos listas que el usuario elige en la clase principal.
* bolasANumero: es un método que reconoce cuantas bolas hay en cada columna del ábaco y convierte estos números en un único número sumándolas para después introducirlo en el árbol de búsqueda.
* introducirNumeroAArbol: es el método encargado de meter un número tipo long al árbol de búsqueda.
* imprimirArbol: Este ábaco imprime de manera limpia y de fácil comprensión el árbol con sus datos actuales.
* buscarNumeroEnArbol: este método busca en el árbol un número concreto.
  1. **Manual de usuario.**

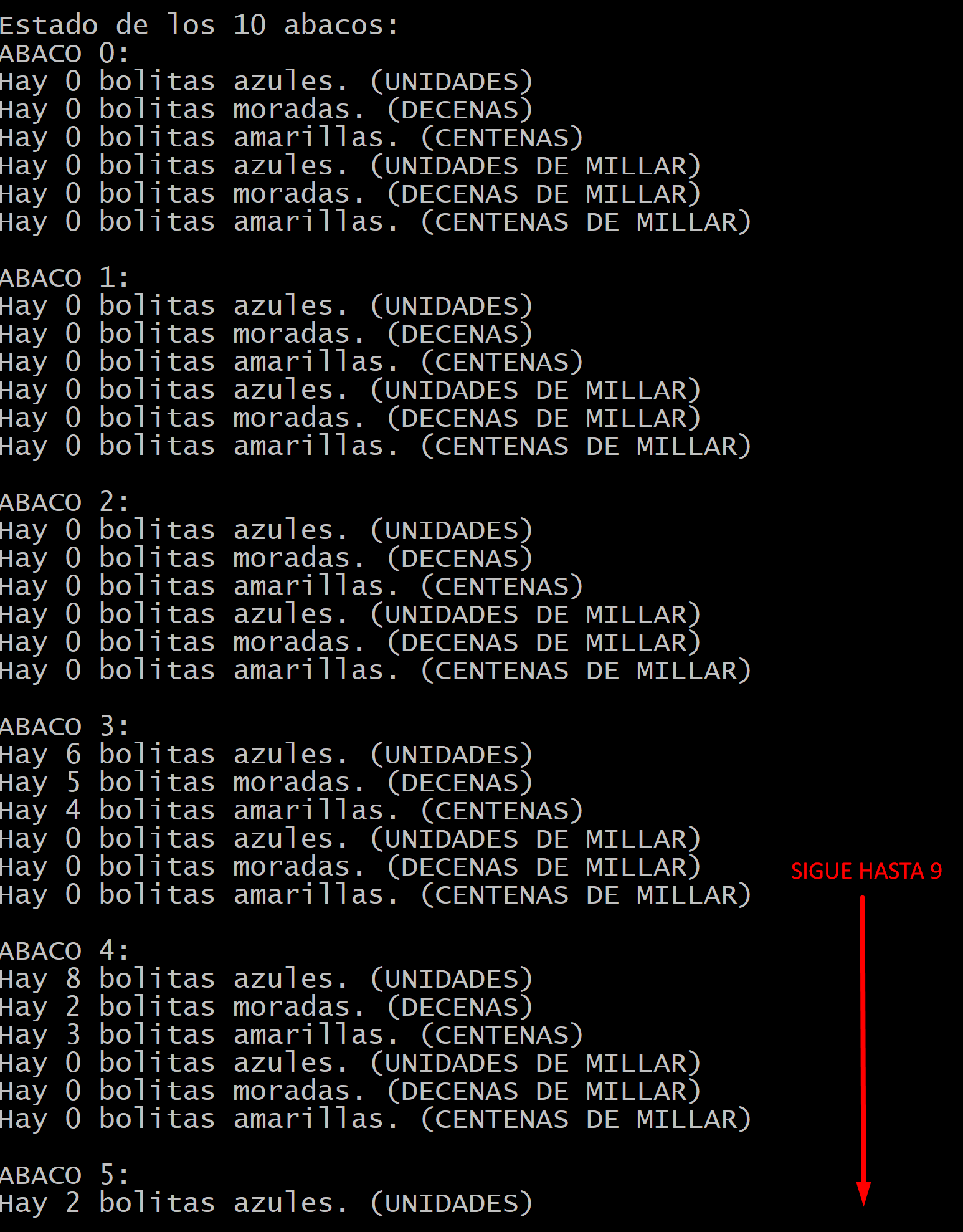
Al iniciar el programa, se le pide al usuario que elija entre diez ábacos (del cero al nueve) para rellenar un ábaco con un número de tres dígitos.



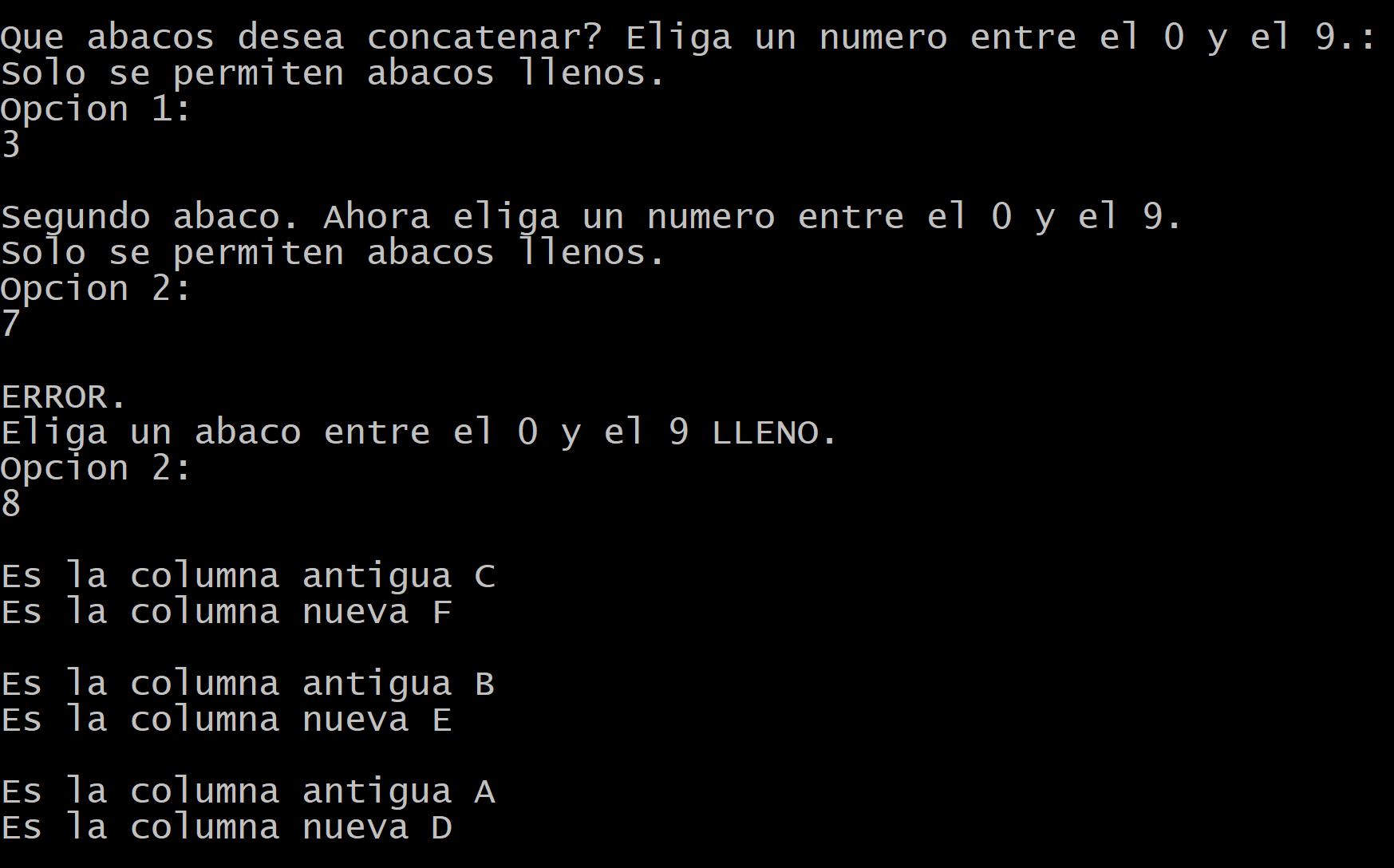
Una vez que se ha metido correctamente el número, el programa pide otro ábaco a rellenar. Si el usuario desea no rellenar más ábacos, escribirá “88”.



A continuación, el programa mostrará por pantalla el estado de todos los ábacos, con y sin bolas.

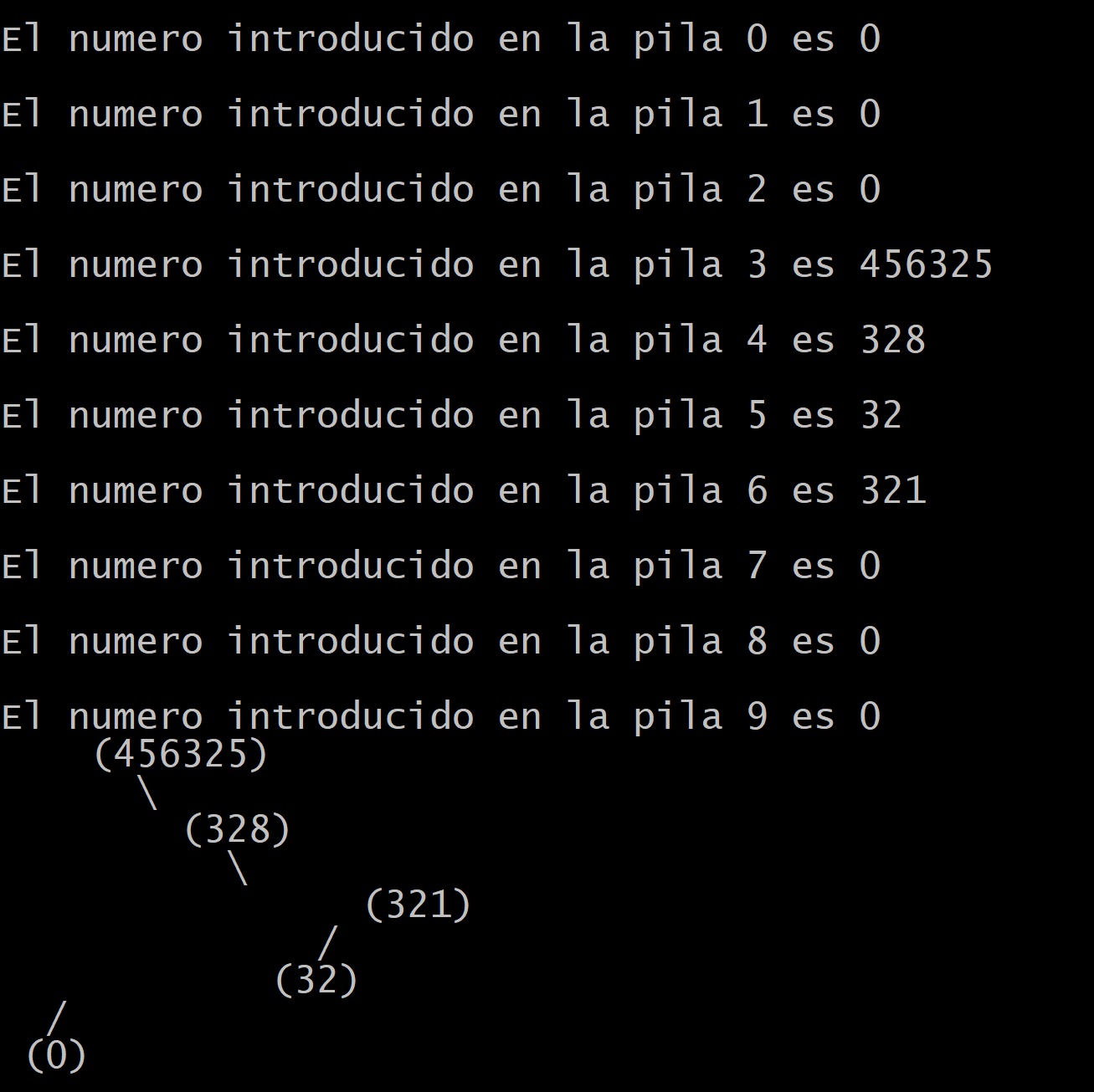


Después, solicita al usuario dos listas (llenas) para concatenarlas, dando así el resultado de la segunda lista introducida al final de la primera introducida, dando este resultado:

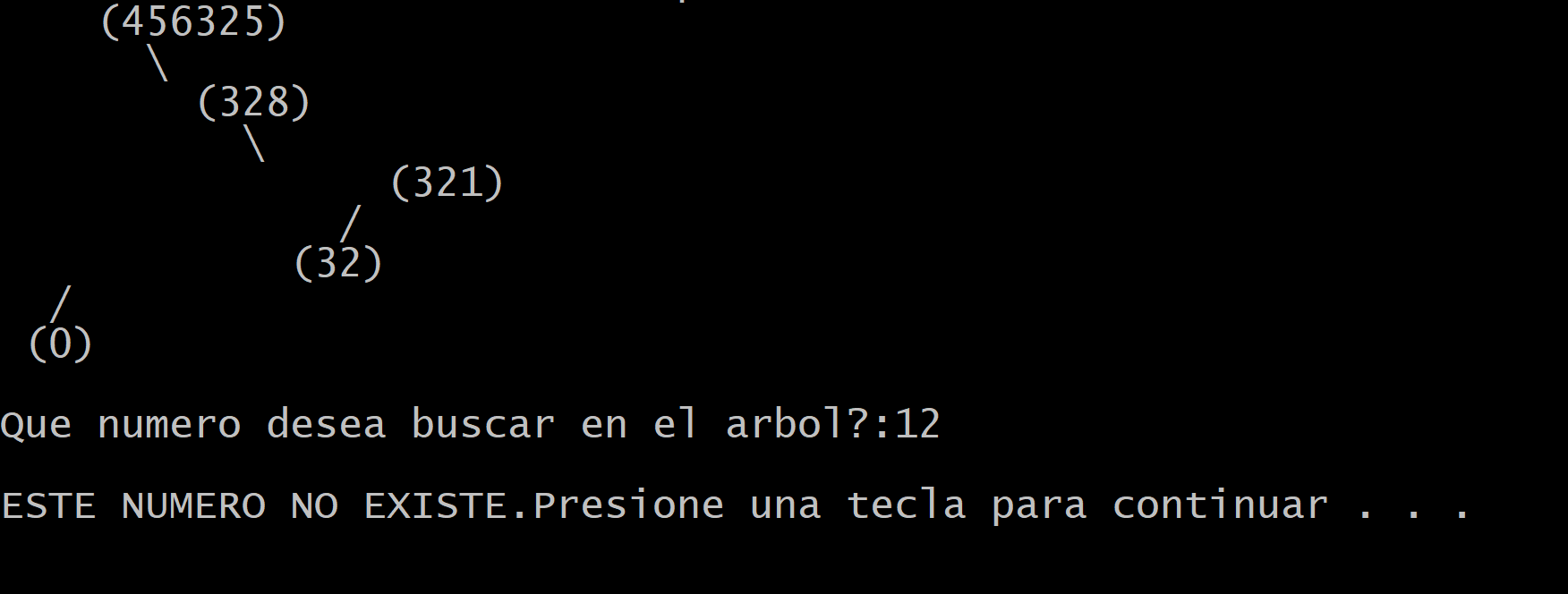


**Importante: la segunda lista se vaciará.**

Ahora el programa volverá a imprimir el estado actual de los ábacos, ya que esos valores se introducirán en un árbol de búsqueda.



Por último, el programa solicitará al usuario un número para comprobar si existe en el árbol:



1. **Ejemplos de funcionamiento. Pruebas.**

Con lo que se observa en el manual de usuario, el árbol que se imprime, no se imprime correctamente ya que el procesamiento que éste sigue, es añadir primero el número mayor introducido y recursivamente, va añadiendo los demás nodos, creando así un árbol de búsqueda erróneo.

1. **Bibliografía.**

* Concatenar dos listas: <https://es.stackoverflow.com/questions/116811/concatenar-dos-listas-simplemente-enlazadas>
* Imprimir un árbol como nosotros conocemos: <http://stackoverflow.com/questions/13484943/print-a-binary-tree-in-a-pretty-way>