Estructuras de Datos

Grado en Informática
Segundo Curso, segundo cuatrimestre
Escuela Politécnica Superior de Córdoba
Universidad de Córdoba
Curso académico 2014-2015

Práctica 3. Uso de árboles binarios ordenados.

- **Almacenamiento y obtención de la lista doblemente enlazada desde un fichero de texto**
  - El estudiante debe implementar una función que permita **guardar** los elementos de una lista ordenada doblemente enlazada de personas en un fichero de texto.
    - La función recibirá como parámetros la lista y el nombre del fichero.
  - El estudiante debe implementar una función que permita **cargar** los elementos de una lista ordenada doblemente enlazada de personas desde un fichero de texto.
    - La función recibirá como parámetros la lista y el nombre del fichero.

- **Ampliación de la clase Persona**
  - El estudiante debe ampliar la clase Persona para trabajar con ficheros de texto.
  - El código nuevo y mejorado del tipo ***Persona*** lo tenéis a vuestra disposición.
  - Sólo deberéis sobrecargar los operadores << y >> para leer o escribir una persona en un fichero texto.
    - La sobrecarga se hace de manera análoga a los << y >> que ya están sobrecargados para entrada y salida por pantalla.

```
std::ostream operator<< (std::ostream& out, Persona const& p);
std::istream operator>> (std::istream& in, Persona& p);
```

- **Creación de la clase Clave**
  - El estudiante ha de crear una clase ***Clave***, que contendrá el dni de un alumno y un entero que se usará para indicar su posición en la lista.
  - Este tipo ha de ser implementado de manera similar al tipo ***Persona***.
  - La sobrecarga de los operadores de comparación se hará en función del dni.
  - Habrá que sobrecargar los operadores << y >> para leer o escribir una clave en un fichero texto.

- **Creación del árbol de claves a partir de la lista ordenada doblemente enlazada.**
  - Partiendo de la clase lista ordenada doblemente enlazada implementada en la práctica anterior y de la clase árbol binario ordenado, cuyo código se suministra, el estudiante debe implementar una función que, a partir de una lista de personas, **cree** un árbol binario ordenado de claves, ordenado por el dni.
    - La función recibirá como parámetros la lista y el árbol.

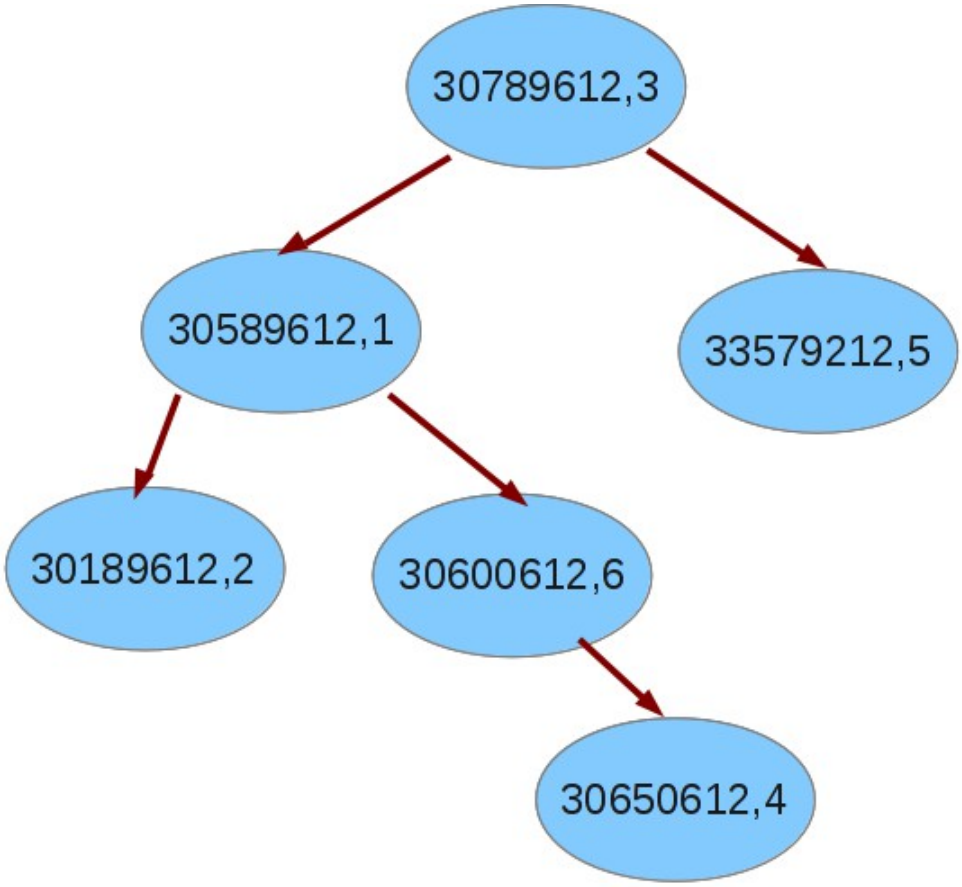
- **Almacenamiento y obtención del árbol de claves desde un fichero**
  - El estudiante debe implementar una función que permita guardar los elementos del árbol de claves en un fichero de texto.
  - La función recibirá como parámetros el árbol y el nombre del fichero.
  - Para ello se debe sobrecargar el operador '<<' para el tipo ArbolBinario<Clave>*.

```
std::ostream operator<< (std::ostream& out, const ArbolBinario<Clave>* t);

precondition: t!=0;
```

- El estudiante debe implementar una función que permita cargar los elementos del árbol de claves desde un fichero de texto.
  - La función recibirá como parámetros el árbol de claves y el nombre del fichero. Para ello se debe sobrecargar el operador '>>' para el tipo ArbolBinario<Clave>*.

    ```
    std::istream operator>> (std::istream& in, ArbolBinario<Clave>* t);

    precondition: t!=0 and t->isEmtpy()
    ```

- **Formato para guardar el árbol.**
  - Caracteres para marcar comienzo y final de nodo: '(' y ')'
  - Caracteres para marcar un árbol vacío ':'
  - Se utilizará un recorrido prefijo para guardar/cargar el árbol.
    - Ejemplo: si tenemos un árbol de la figura el fichero texto que almacena el árbol es mostrado.




```
( 30789612,3 ( 30589612,1 ( 30189612,2 : : ) ( 30600612,4 : (
30650612,6 : : ) ) ) ( 3579212,5 : : ) )
```