Estructuras de Datos y de la Información

Práctica 5

Búsqueda e Inserción en Ficheros Ordenados

OBJETIVO: El objetivo de la práctica es implementar un tipo abstracto de datos que permita buscar e insertar de forma ordenada los registros de un fichero. Los registros son de longitud fija para permitir la implementación de la búsqueda binaria.

FECHA: Esta práctica se entregará los días 11, 12 y 13 de enero de 2005.

ENUNCIADO: Desarrolla un programa en lenguaje C++ que reciba como entrada un fichero de libros en formato de registros de longitud fija, y genere como salida una copia del fichero con los registros ordenados de menor a mayor.

El programa tomará los parámetros de la línea de comandos:

```
$> filesort biblio_fix.dat biblio_sort.dat
```

Para ello se requiere el siguiente prototipo de la función `main()`.

```
int main(int argc, char* argv[]) {
  // argc = 3
  // argv[0] = "filesort"
  // argv[1] = "biblio_fix.dat"
  // argv[2] = "biblio_sort.dat"
  ...
}
```

NOTAS DE IMPLEMENTACION:

- Definir la clase genérica `FileSort` que deriva de la clase `FileBuf` y que implementa un fichero ordenado de registros de longitud fija. Esta clase modifica el comportamiento de algunos métodos de la clase ancestro.

```
template <class RECORD>
class FileSort: public FileBuf<RECORD> {

   // Abre un fichero ya existente.
   // Retorna un error si el formato del fichero no tiene registros de
   // longitud fija.
   int FileSort<RECORD>::Open(char *file, int mode = ios::in | ios::out |
               ios::binary | ios::nocreate);

   // Crea un nuevo fichero en el formato especificado.
   int FileSort<RECORD>::Create(char *file, int format,
               int mode = ios::in | ios::out | ios::binary | ios::trunc);

   // Inserta de forma ordenada un nuevo registro en el fichero.
   // Retorna el número de bytes escritos en el fichero.
   TSize FileSort<RECORD>::Append(const RECORD& r);

   // Busca y devuelve un registro en el fichero.
   // También devuelve la posición del registro en el fichero.
   // Si no encuentra el registro devuelve PNULL en la posición.
   bool FileSort<RECORD>::Search(RECORD& r, TPointer &pos);
};
```

- La clase base `RECORD` utilizada en un fichero `FileSort` debe disponer de los siguientes métodos y operadores: Pack(IOBuffer*), UnPack(IOBuffer*) y operator==(const RECORD&) u otros operadores de comparación utilizados.