TADs: Biblioteca Ejercicio

Se desea realizar una determinada aplicación para gestionar los libros que hay en una biblioteca. Supóngase un modelo de biblioteca "simplificado" con las siguientes características:

- Cada libro se identifica por su **título** (cadena de < 80 caracteres) y **autor**.
- El **número de ejemplares** de cada título que hay en la biblioteca es diferente. Ej.: del libro "El peor viaje del mundo" de Apsley Cherry-Garrad hay 10 ejemplares, pero sólo 5 ejemplares de "Anatomía de un instante" de Javier Cercas.
- La biblioteca puede comprar **lotes de libros** diferente tamaño. Ej.: en un determinado instante puede añadir 5 ejemplares más a sus fondos del libro "Retrato de un hombre inmaduro".
- Nunca habrá más **25 ejemplares** de ninguno de los títulos y la biblioteca nunca tendrá más de **25.000 títulos** diferentes.
- Los usuarios sacan en **préstamo** libros de la biblioteca (obviamente siempre que la biblioteca tenga el título en sus fondos y existan ejemplares disponibles) y "devuelven" los libros prestados a la biblioteca.
- Los usuarios puede tener por **tiempo indefinido** los ejemplares sacados en préstamo y tampoco se registran a los usuarios que tienen los libros en préstamo.
- La biblioteca imprime periódicamente un **catálogo** de los libros que hay en sus fondos.

1) Especifique *informalmente* los TAD que considere necesarios para diseñar la aplicación (datos + primitivas).

Nota: Supóngase que disponemos ya de los TADs : boolean, status y cc (cadena de caracteres)

TAD LIBRO: cadena para el título, cadena para el autor, número de ejemplares y número de disponibles

Al menos hacen falta las siguientes primitivas:

Libro iniLibro(CC titulo, CC autor, entero ejemplares, entero prestados);

CC getTitulo (Libro lib);

CC getAutor (Libro lib);

entero getNumEjemplares (Libro lib); //-1 si error

entero getNumDisponibles (Libro lib); //-1 si error

status modNumEjemplares(Libro lib, entero num);

status modNumDisponibles (Libro lib, entero num);

entero comparaLibros (Libro lib1, Libro lib2);

status imprimeLibro (fichero fich, Libro lib);

status liberaLibro (Libro lib);

TAD BIBLIO: conjunto de libros y número de libros. Podría guardarse más información pero con esta nos sirve.

Hacel falta, al menos, las siguientes primitivas:

Biblio iniBiblio (Biblio bib);

bool esVaciaBiblio (Biblio bib);

bool esLlenaBiblio (Biblio bib);

Libro obtenerLibro (Biblio bib, CC titulo, CC autor);

status insertarLibro (Biblio bib, Libro lib);

bool disponibleTitulo (Biblio bib, CC titulo, CC autor);

status modificaEjemplaresLibro (Biblio bib, CC titulo, CC autor, entero num); //comprar fondos nuevos o desechar

status modificaDisponiblesLibro (Biblio bib, CC titulo, CC autor, entero num); //num negativo=prestar, num pos=devolver

status imprimeBiblio (fichero fich, Biblio bib);

status liberaBiblioteca (Biblio bib);

2) Escriba en código C las estructuras de datos (EdD) adecuadas para los TAD anteriores.

```
#define MAXEJEMPLARES 25
#define MAXCHARS 80

struct _ LIBRO {
  char titulo[MAXCHARS];  // ó char * titulo;
  char autor [MAXCHARS];  // ó char * autor;
   int nejemplares;  // valor máximo: MAXEJEMPLARES
   int ndisponibles; // al inicio,  =nejemplares
} ;
```

```
#define MAXLIBROS 25000

struct _ BIBLIO {
  Libro *plibs[MAXLIBRO];  //Tabla de punteros a libros
  int total_libros;
};
```

3) ¿En qué ficheros deben definirse las estructuras anteriores?

En libro**.c** y biblio**.c** respectivamente.

4) **¿Cómo se definen los nuevos tipos libro y biblioteca? ¿En qué ficheros se definen?**

En libro**.h:** **typedef struct _ LIBRO Libro;**

En biblio**.h:** **typedef struct _ BIBLIO Biblio;**

5) **Escriba en C el contenido del fichero libro.h**

```
#ifndef _LIBRO_H
#define _LIBRO_H
#include “tipos.h”

typedef  struct _ LIBRO Libro;

Libro * iniLibro (const char * titulo, const char * autor, const int ejemplares); //NULL si error
char * getTitulo (const Libro *plib);  //NULL si error
char * getAutor (const Libro *plib); //NULL si error
int getNumEjemplares (const Libro *plib);     //-1 si error
int getNumDisponibles (const Libro *plib);    //-1 si error
status modNumEjemplares(Libro *plib, const int num);
status modNumDisponibles (Libro *plib, const int num);
status imprimeLibro (FILE * pf, const Libro *plib);
status liberaLibro (Libro *plib);
int comparaLibros (const Libro *plib1, const Libro *plib2);  // 0 si son iguales (como strcmp), y -1 si no.

#endif
```

---

6) **Escriba en C el contenido del fichero biblio.h**

```
#ifndef _BIBLIO_H
#define _BIBLIO_H

#include “tipos.h”
#include “libro.h”

typedef  struct _ BIBLIO Biblio;

Biblio * iniBiblio ();
bool esVaciaBiblio (const Biblio *pbib);
bool esLlenaBiblio (const Biblio *pbib);
bool disponibleTitulo (const Biblio *pbib, const char * titulo, const char * autor);
Libro * obtenerLibro (const Biblio *pbib, const char *titulo,  char * autor);
status insertarLibro (Biblio *pbib, const Libro *lib);  //Inserta un libro ya creado en la biblioteca
//comprar fondos nuevos o desechar:
status modificaEjemplaresLibro (Biblio *pbib, const char * titulo, const char * autor, const int num);
//prestar y devolver (num negativo=prestar, num pos=devolver):
status modificaDisponiblesLibro (Biblio *pbib, const char * titulo, const char * autor, const int num);
status imprimeBiblio ( FILE * pf, const Biblio *pbib);
status liberaBiblioteca (Biblio *pbib);
#endif
```

---

7) **Escriba el código C de la primitiva que crea un libro.**

Libro * iniLibro (const char * titulo, const char * autor, int ejemplares) {

```
Libro *paux = NULL;
if ( titulo == NULL ||  autor ==NULL)
        return NULL;
paux = (Libro*) malloc (sizeof (Libro));
if (!paux)          //if (paux ==NULL)
        return NULL;
strcpy (paux->titulo, titulo);
strcpy (paux->autor, autor);
paux->nejemplares = paux->ndisponibles = ejemplares;
return paux;
}
```

8) **Escriba el código C de las primitivas que permiten obtener el número de ejemplares de un libro, el número de ejemplares disponibles de dicho libro, el título y el autor, respectivamente.**

int getNumEjemplares (const Libro *plib);

int getNumDisponibles (const Libro *plib);

```
int getNumEjemplares (const Libro *plib) {
        if (plib == NULL)
                return -1;
        return plib->nEjemplares;
}
int getNumDisponibles (const Libro *plib) {
        if (plib == NULL)
                return -1;
        return plib->nDisponibles;
}
char * getTitulo (const Libro *plib){
        if (plib == NULL)
                return -1;
        return plib->titulo;
}
char * getAutor (const Libro *plib) {
        if (plib == NULL)
                return -1;
        return plib->autor;
}
```

¿En qué fichero debe estar el código de estas primitivas?

En libro.c

9) **Escriba el código C de la primitiva para comparar libros:**

int comparaLibros (const Libro *plib1, const Libro *plib2); ***// 0 si son iguales (como strcmp), y -1 si no.***

```
int comparaLibros (const Libro *plib1, const Libro *plib2) {
    if ( !plib1  ||  !plib2)
            return -1;
    if ( strcmp(plib1->titulo, plib2->titulo) == 0  &&   strcmp (plib1->autor, plib2->autor) == 0)
            return 0;  // Son iguales
    return -1;
}
```

¿En qué fichero debe estar el código de estas primitivas?

En libro.c

---

10) **Escriba el código C de la primitiva para liberar un libro:**

```
void liberaLibro (Libro *plib) {
    if (plib != NULL)
            free(plib);
}
```

---

11) Escriba el código C de la primitiva que crea e inicializa una biblioteca

```
Biblio *iniBiblio () {
    Biblio *paux;
    int i;
    paux = (Biblio*) malloc(sizeof(Biblio));
    if(!paux)
            return NULL;
    for(i=0; i<MAXLIBROS; i++)                  //Bucle opcional?
            paux->plibs[i] = NULL;
    paux -> total_libros = 0;
    return paux;
}
```

---

12) Escriba el código C de las primitivas que indican si una biblioteca está vacía ó llena.

```
BOOL esVaciaBiblio (const BIBLIO *pbib) {
    if(  pbib== NULL  ||   pbib->total_libros == 0)
            return TRUE;
    return FALSE;
}

BOOL esLlenaBiblio (const BIBLIO *pbib) {
    if( (pbib==NULL  ||  pbib->total_libros < MAXLIBROS)
            return FALSE;
    return TRUE;
}
```

---

13) Escriba el código C de la primitiva que indica si un título determinado está disponible en la biblioteca

```
// Busca en la biblioteca bib el libro con el título y autor indicados en los
// parámetros de entrada, y devuelve TRUE en caso de haber ejemplares disponibles
// de ese título y FALSE en caso contrario.

bool disponibleTitulo (const Biblio *pbib, const char * titulo, const char * autor) {
    LIBRO * pLibro=NULL;

    //Buscar en la biblioteca el libro con el título y autor indicados.
    pLibro = obtenerLibro (bib, titulo, autor);
    if (pLibro == NULL)
        return FALSE;  //No existen libros en la biblio con ese título y autor
    //Si existen, obtener el nº de disponibles del libro cuya dirección tiene pLibro (del libro al que apunta pLibro)
    if (getNumDisponibles (pLibro) > 0)
        return TRUE;
    return FALSE;
}
```

---

14) Escriba el código C de la primitiva que modifica el número de ejemplares de un libro determinado.

Idea:
Ver si el libro está en la biblioteca
sí → obtener el libro y modificar su nº de ejemplares
no →crear el libro e insertarlo en la biblioteca

```
status modificaEjemplaresLibro (Biblio *pbib, const char * titulo, const char * autor, const int num);
    LIBRO * pLibro;
    // Mirar en bib si existe libro con esos titulo y autor.
    // sí→ se guarda la dirección de ese libro en la variable pLibro
    pLibro = obtenerLibro (bib, titulo, autor);
    if (pLibro == NULL)
        //No existen libros con ese título y autor → si num es positivo, crearlo
        if (num > 0){
            if ( (pLibro=iniLibro(titulo, autor, num, 0)) == NULL)
                return ERROR;  //Error al crearlo
            if (insertarLibro (bib, pLibro) == ERROR) {
                liberarLibro (pLibro);
                return ERROR; //Error al insertarlo en la biblioteca
            }
        }
    }
    else  //Sí existe el libro → datos están en la dir pLibro → modificar
        if (modNumEjemplares(pLibro, num) == ERROR)
            return ERROR;
    return OK;
}
```