PROYECTO BIBLIOTECA

Diego Araújo y Guillermo Barreiro

2017

Pantalla

pantalla.c

Librería que se encarga de mostrar en pantalla los menús de Biblioteca. Por una parte incluye las funciones necesarias para imprimir la cabecera, y por otra muestra el mensaje de confirmación de salida del programa.

```
##########################################
#                                        #
#                                        #
#                BIBLIOTECA              #
#                                        #
#                                        #
##########################################
```

Funciones

1. **Pantalla:** se encarga de dibujar la carátula del programa nada más arrancar, una vez comprobados los ficheros de referencias y lectores. Para ello va llamando al resto de funciones de este módulo, que se encargan de dibujar las líneas que conforman la carátula.

   void pantalla();

```
for(k=0;k<7;k++){ //imprime línea a línea
      si k==0 ó k==6:
            imprimo una base;
      si k==3:
            imprimo el título “BIBLIOTECA”;
      en otro caso:
            imprimo una línea con el símbolo ‘#’ a cada lado;
      }
```

2. **Base**: imprime por pantalla el carácter recibido como argumento tantas veces como se le haya pedido, formando así una línea.

   void base(char caracter, int longitud);
   caracter: el carácter que se va a dibujar
   longitud: las veces que se va a repetir ese carácter

```
for(k=0;k<longitud;k++){
      imprimo el caracter;
}
imprimo un salto de línea;
```

3. **Lado**: imprime una línea de la longitud deseada con un carácter al principio y al final.

   void lado(char caracter, int longitud);
   caracter: el carácter que se va a dibujar
   longitud: longitud de la línea

```
for(k=0;k<longitud;k++){
      si k == 0 ó k == (longitud-1):
            imprimo el caracter;
      sino:
            imprimo un espacio;
}
imprimo un salto de línea;
```

4. **Título**: imprime una línea con un mensaje centrado y un carácter a cada lateral.

   void titulo(char* palabra, char caracter, int longitud);
   palabra: cadena de caracteres con el mensaje que se va a mostrar
   caracter: el carácter que se colocará en los márgenes de la línea
   longitud: longitud de la línea

```
int d = strlen(palabra)-1;
for(k=0;k<longitud;k++){
      si k == 0 ó k == (longitud-1):
            imprimo el caracter;
      si k == (longitud-d)/2:
            imprimo la palabra;
            avanzo k d unidades;
}
imprimo un salto de línea;
```

5. **Confirmar**: le pregunta al usuario si desea salir del programa. La respuesta tiene que ser obligatoriamente o una ’s’ o una ’n’ (mayúsculas o minúsculas, no importa). En caso de que la respuesta sea diferente, se le seguirá pidiendo indefinidamente hasta que escriba un carácter válido. En caso de que el usuario escoja ‘Sí’, la función devolverá un 1; si escoge ‘No’ un 0.

int confirmar();

```
mientras la respuesta del usuario no sea ’S’/’s’ ó ’N’/’n’:
      pídele que introduzca su respuesta;
      si la respuesta == ’s’/’S’:
            devuelve 1;
      si la respuesta == ’n’/’N’:
            devuelve 0;
      en otro caso repite el bucle;
```

Base de datos

base_datos.c

Librería que se encarga de gestionar la base de datos del programa Biblioteca. La base de datos se almacena en los ficheros de texto `referencias.txt` y `lectores.txt`.

Lectores (lectores.txt):

`:Codigo:Nombre:`

struct est_lector

Código	int 16 bits
Nombre	char [50]

Referencias (referencias.txt):

`:Signatura:Tipo:Autor:Titulo:Anho:Votantes:Votos:Criticos:`
`#Lector:Opinion:`

struct est_referencia

Signatura	int 32 bits
tipo	char =L ll A
Autor	char [50]
Titulo	char[80]
Anho	int 16 bits
Votantes	int 16 bits
Votos	int 32 bits
críticos	int 16 bits
lector	int 16 bits
opinión	char [80]

Modos de lectura:

r: abre la base de datos en modo lectura

w: crea el fichero si aún no existe, y en caso de que ya existiese lo borra

a: escribe al final del fichero

Hay tres tipos de funciones:

Menú principal:

Son invocadas cada vez que el usuario escoge una opción del menú principal.

1) Incorporar lector:

Se agrega a la base de datos un nuevo usuario.

lectores.txt, a

int lector (int *id, struct est_lector* lectores);

id: puntero a la variable en la que se almacena la posición del último lector
lectores: array en el que se almacenan los lectores de la base de datos

```
el usuario introduce un nombre
si el formato es correcto y no existe ese lector:
      lo guarda en el array
      lo almacena al final del fichero lectores.txt
```

2) Agregar referencia:

Almacena en la base de datos una nueva referencia (libro o artículo).

referencias.txt, a

int referencia (int *q, struct est_referencia* referencias);

q: puntero a la variable en la que se almacena la posición de la última referencia
referencias: array en el que se almacenan las referencias de la base de datos

```
el usuario introduce los datos de la referencia
si son correctos:
      la guarda en el array
      la almacena al final del fichero referencias.txt
```

3) Expresar opinión:

El lector agrega una reseña de una determinada referencia.

int opinion (int indice_lector, int indice_signatura, struct est_referencia* referencias);

indice_lector: posición del último lector en el array de lectores
indice_signatura: posición de la última referencia en el array de referencias
referencias: array en el que se almacenan las referencias de la base de datos

```
      el usuario introduce la signatura de la referencia sobre la que
quiere opinar
      el usuario introduce el código del lector que va a opinar
      el usuario introduce el comentario
      si los datos introducidos son válidos:
            guarda el comentario en el array de referencias
            sobreescribe 'referencias.txt' con el nuevo contenido del
array
```

4) Obtener informe:

Muestra en pantalla las opiniones de los lectores sobre una determinada referencia.

int informa (int indice, struct est_referencia* referencias, struct est_lector* lectores);

indice: posición de la última referencia en el array de referencias

referencias: array en el que se almacenan las referencias de la base de datos

lectores: array en el que se almacenan los lectores de la base de datos

```
      el usuario introduce la signatura de la referencia sobre la que
desea obtener las opiniones
      si la referencia es válida:
            muestra el número de comentarios
            imprime los comentarios en pantalla, uno a uno
```

5) Emitir voto:

El lector agrega una nota entre 0 y 10 para una referencia determinada

int voto (int indice, struct est_referencia* referencias);

indice: posición de la última referencia en el array de referencias

referencias: array en el que se almacenan las referencias de la base de datos

```
      el usuario introduce la signatura de la referencia que va a votar
      si la signatura es válida:
            introduce voto (entero entre 0 y 10)
            guarda el voto en el array de referencias
            actualiza el fichero referencias.txt
```

6) Calcular nota:

Calcula la nota media de una referencia con los votos que introducen los lectores.

int nota (int indice, struct est_referencia* referencias);

indice: posición de la última referencia en el array de referencias

referencias: array en el que se almacenan las referencias de la base de datos

```
el usuario introduce la signatura
si es válida:
      imprime en pantalla el número de votos
      muestra en pantalla la media de los votos
```

7) Listar referencias:

Muestra en pantalla una lista con todas las referencias del tipo seleccionado.

int listar (int indice, struct est_referencia* referencias);

indice: posición de la última referencia en el array de referencias

referencias: array en el que se almacenan las referencias de la base de datos

```
el usuario introduce el tipo de referencia que desea listar
si el tipo es correcto:
      imprime por pantalla todas las referencias de ese tipo, una a
una
```

Entrada / salida:

Son las funciones a las que `biblioteca.c` llama al principio, antes de mostrar el menú principal. Se encargan de comprobar que el formato de los archivos es correcto y de cargar en memoria su contenido.

- **void comprobar_fichero_lectores()**: comprueba que existe el fichero `lectores.txt`. En caso negativo lo crea.
- **void comprobar_fichero_referencias()**: comprueba que existe el fichero `referencias.txt`. En caso negativo lo crea.
- **int comprobar_formato_lectores()**: comprueba que el fichero `lectores.txt` siga el formato indicado en la página 4. Si es así, devuelve un 0; en caso contrario, un 1. Cada línea se comprueba con la función `comprobar_lectores(cadena)`, donde *cadena* es un array de char en el que previamente se ha cargado una línea del fichero de texto.
- **int comprobar_formato_referencias()**: comprueba que el fichero `referencias.txt` siga el formato indicado en la página 4. Si es así, devuelve un 0; en caso contrario, un 1. Cada línea se comprueba con la función `comprobar_referencias(cadena)`, donde *cadena* es un array de char en el que previamente se ha cargado una línea del fichero de texto.
- **void arranque(struct est_lector* lectores, struct est_referencia* referencias, int *ultimo_lector, int *ultima_ref)**: inicializa los arrays de lectores y de referencias (primer y segundo parámetro, localizados en `biblioteca.c`), cargando el contenido de los ficheros en memoria, valiéndose para ello de las funciones auxiliares detalladas debajo. En las variables a las que apuntan el tercer y cuarto parámetro almacena la posición del último lector y de la última referencia en los arrays.
- **void referencias_to_file(struct est_referencia* referencias)**: abre `referencias.txt` en modo escritura ("**w**"), borrando el contenido que hubiese previamente, y guarda todas las referencias almacenadas en el array referencias, línea a línea, almacenando también los comentarios.

Auxiliares:

Funciones de uso interno de la librería. Permiten comprobar que el formato de un número o una cadena introducida por el usuario es correcto, o convertir una struct a una línea de texto y viceversa.

- **void lector_to_struct(FILE* puntero, struct est_lector* lector):** crea una `struct est_lector` a partir de la línea de `lectores.txt` a la que apunta `puntero`.
- **void referencia_to_struct(FILE *puntero, struct est_referencia* referencia)**: crea una `struct est_referencia` a partir de la línea de `referencias.txt` a la que apunta `puntero`.
- **void comentario_to_struct(FILE* puntero, struct est_comentario* comentario)**: crea una `struct est_comentario` a partir de la línea de `referencias.txt` a la que apunta `puntero`.
- **int comprobar_cadena(char* cadena, int longitud)**: Comprueba si una cadena cumple el formato correcto. También sustituye el salto de línea final por un fin de cadena.
  - Si la cadena no está vacía, no supera la longitud máxima (segundo parámetro) y no contiene el carácter ':', **devuelve un 0**.
  - Si hay algún error lo indica por pantalla y **devuelve un 1**.
- **int comprobar_numero(char* cadena, int inicio, int final)**: Comprueba si una cadena obtenida mediante `fgets` es un número comprendido entre los parámetros inicio y final (incluidos). En caso afirmativo **devuelve dicho número** como un entero, y en caso negativo **devuelve -1**. Los números de inicio y final tienen que ser mayores o iguales a 0. Si no devuelve -1.

Biblioteca

biblioteca.c

Programa principal del proyecto. Se encarga de arrancar la aplicación y de mostrar el menú principal. Para dibujar la carátula inicial llama a `pantalla()`, de la librería `pantalla`. La interacción con la base de datos la realiza a través de la librería `base_datos`.

Variables declaradas

- **struct est_lector lectores[100]**: array con capacidad para almacenar hasta 100 estructuras de tipo `est_lector` (definido en *base_datos.h*).
- **struct est_referencia referencias[100]**: array con capacidad para almacenar hasta 100 estructuras de tipo `est_referencia` (definido en *base_datos.h*).
- **int ultimo_lector**: posición en `lectores[]` del último lector registrado. *código = índice + 1*.
- **int ultima_ref**: posición en `referencias[]` de la última referencia registrada. *signatura = índice + 1*.

Función main():

1. Comprueba que los ficheros de lectores y referencias están bien formateados, y si no es así el programa finaliza. Si no existen se crean.

   base_datos: `comprobar_fichero_lectores()`, `comprobar_formato_lectores()`, `comprobar_fichero_referencias()`, `comprobar_formato_referencias()`.

2. Imprime la carátula del programa mediante una llamada a la función `pantalla()`, de la librería `pantalla`.
3. Carga en los arrays `lectores[]` y `referencias[]` el contenido de la base de datos, usando la función `arranque()` de la librería `base_datos`.
4. Muestra en pantalla el **menú principal** y le pide al usuario que escoja una de las opciones. Si la opción escogida es válida (número entre el 1 y el 7) llama a la función correspondiente de `base_datos`. Si es un 0, llama a la función `confirmar()` de `pantalla` para confirmar si el usuario realmente desea salir del programa. Si la selección no es válida, se le informa al usuario de ello y se vuelve a mostrar el menú.