Memoria del proyecto biblioteca completo

Grado de Ingeniería de Tecnologías de Telecomunicación
Programación I

Adrián Villar Gesto
Sebastián Souto Robles

Definición del proyecto Biblioteca

El objetivo de este proyecto es el desarrollo de una aplicación denominada Biblioteca que permita gestionar una sencilla base de datos de referencias bibliográficas.

El programa estará constituido por tres módulos, el módulo principal, el módulo gestor de la base de datos y el módulo gestor de pantalla.

Adrián Villar Gesto
Sebastián Souto Robles

-Módulo biblioteca: Módulo donde se sitúa el main y donde empieza y acaba el programa. Se compone del main y de las funciones caratula y menú.

Prototipo: void caratula (char símbolo, int ancho, char *texto);

Llama a todas las funciones de impresión de caratula del módulo de gestión de pantalla. Lo que recibe por parámetro son los datos que le pasa a las funciones que llama.

Recibe por parámetro:

-El símbolo que constituye los bordes de la caratula.

-Un entero que se corresponde con el número de caracteres a imprimir por línea.

-Un puntero al inicio de la cadena a imprimir en el centro de la caratula.

Prototipo: void menú ();

Define las variables tam_lec y tambre y la iguala al retorno de las funciones leer_lectores y leer_referencia, que retornan el número total de lectores y referencias.

A continuación imprime el menú con las 10 opciones disponibles y nos invita a escoger una, mostrándonos que la selección es inválida si se selecciona algo distinto (del 0 al 9). Según la selección se llama a las siguientes funciones:

'0': Confirmar, que en caso de retornar un 0 saldrá del programa.

'1': Lector y Actualizar_lector, pasándole al primero la dirección de tam_lec y al segundo el valor de tam_ref.

'2': Referencia y Actualizar_referencia, pasándole al primero la dirección de tam_ref y al segundo el valor de este.

'3': Opinion y Actualizar_referencia, pasándole a ambos el valor de tam_ref y solo a la primera el valor de tam_lec.

'4': Informa a la que pasamos el valor de tam_ref.

'5': Voto y Actualizar_referencia pasándole a ambas el valor de tam_ref.

'6': Nota a la que le pasamos el valor de tam_ref.

'7': Listar a la que pasamos el valor de tam_ref.

'8': Buscar a la que pasamos el valor de tam_ref.

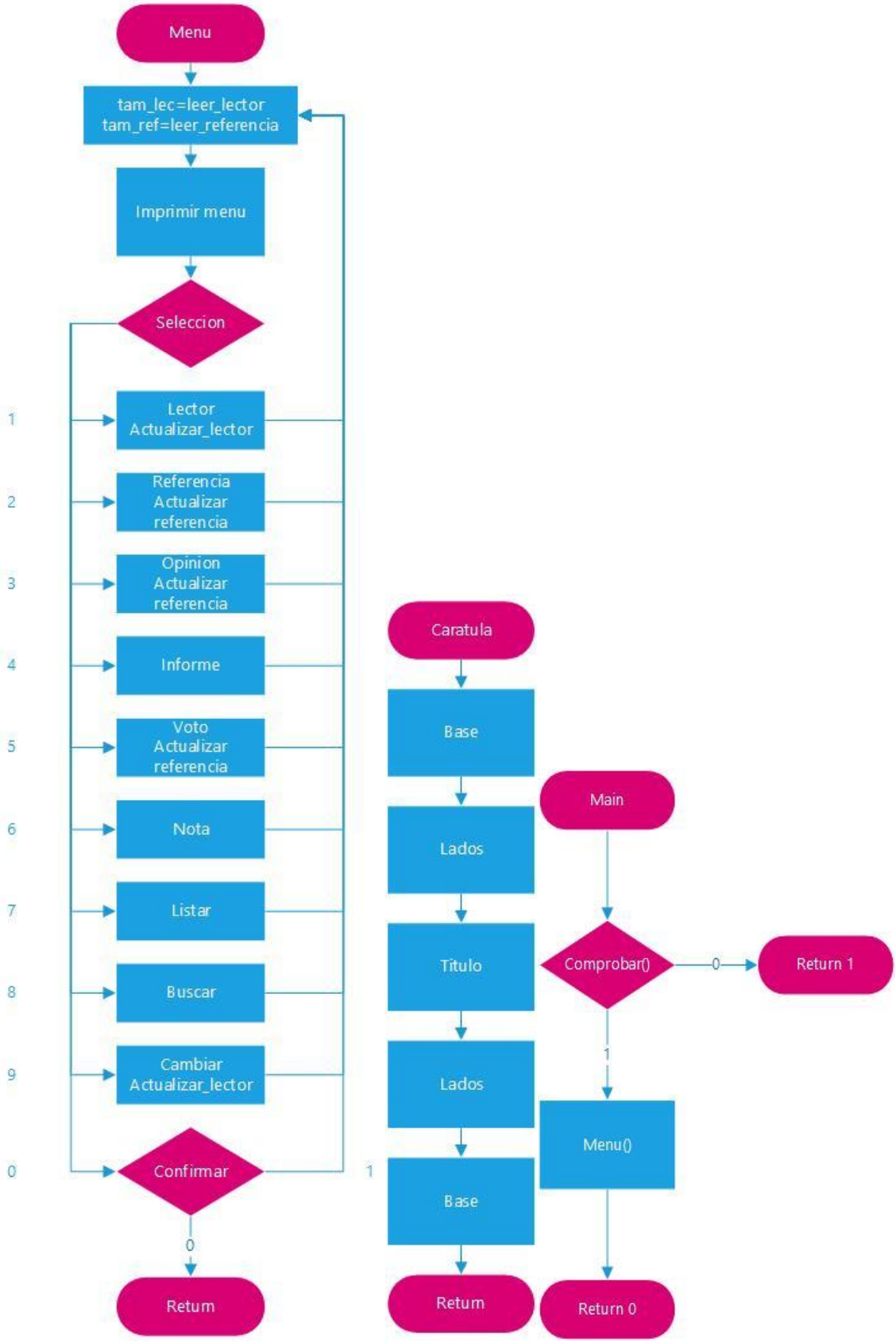
'9': Cambiar y Actualizar_lector pasándole a ambas el valor de tam_lec.

Al salir de una función vuelve a aparecer el menú hasta que se seleccione 0 y 'S'.

int main ();

Primero comprueba los txts llamando a la función confirmar, la cual si retorna un 0 (algún txt incorrecto) sale del main retornando 1 y acabando el programa.

Si no es así imprimimos la caratula y accedemos a la función menú, donde estará hasta que se salga, llegando al return 0 y acabando el programa.



Adrián Villar Gesto
Sebastián Souto Robles

-Modulo gestor de pantalla: Se encarga de la impresión de la caratula al inicio del programa más de confirmar la salida del mismo. Está compuesto por cuatro funciones, Confirmar, más las 3 encargadas de imprimir la caratula: Base, Lados y Titulo.

Prototipo: int confirmar ();

Se accede a ella en caso de querer salir del programa. Pregunta si se quiere salir del programa hasta que se introduzca por teclado 'S' o 'N' (indiferente que sean minúsculas ya que se convierten a mayúsculas con un toupper).

Retorna un 1 en caso de seleccionar 'N' o 'n' y un 0 en caso de seleccionar 'S' o 's'.

Prototipo: void base (char símbolo, int ancho);

Se usa para imprimir la primera o la última línea de la caratula, la cual consiste en una línea compuesta por un único tipo de símbolo y acabada en \n.

Recibe por parámetro:

-El símbolo que constituye toda la línea.

-Un entero que se corresponde con el número de caracteres a imprimir por línea.

Prototipo: void lados (char símbolo, int ancho, int lineas);

Se usa para imprimir las líneas intermedias de la caratula, que consisten en una línea compuesta por espacios, empezada por el simbolo y acabada en el símbolo y \n.

Recibe por parámetro:

-El símbolo que se imprime al inicio y al final de la línea.

-Un entero que se corresponde con el número de caracteres a imprimir por línea.

-El número de líneas idénticas consecutivas que se deben imprimir.

Prototipo: int titulo (char símbolo, int ancho, char *texto);

Se usa para imprimir la línea central de la caratula, que consisten en una línea compuesta por espacios, empezada por el símbolo y acabada en el símbolo y \n y con el texto que deseemos en el medio.

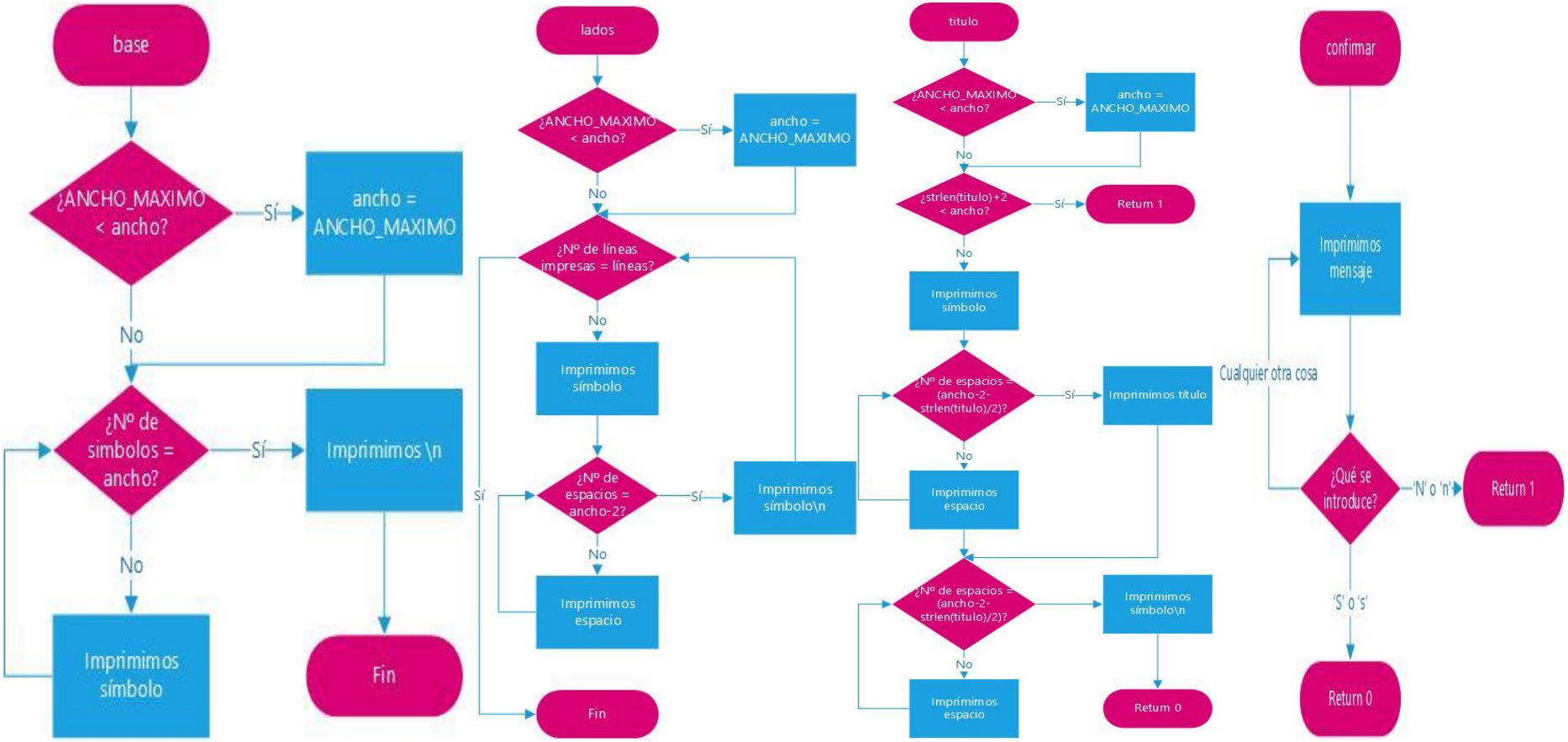
Recibe por parámetro:

-El símbolo que se imprime al inicio y al final de la línea.

-Un entero que se corresponde con el número de caracteres a imprimir por línea.

-Un puntero a la dirección de la cadena a imprimir en el medio de la línea.

Devuelve un 1 y no imprime nada si el ancho es insuficiente para imprimir la cadena más el símbolo de inicio y final de línea. Si esta circunstancia no se da devuelve un 0.



Adrián Villar Gesto
Sebastián Souto Robles

-Modulo gestor de la base de datos: Contiene los structs más las principales funciones del programa, además de las funciones de gestión de los txts. Se listan a continuación:

//STRUCTS----------

struct lectores
  int codigo
  char nombre

struct informacion
  int lector
  char opinion

struct referencia
  int signatura
  char tipo
  char autor[51]
  char titulo[82]
  int anio
  int votantes
  int votos
  int criticos
  struct informacion info

//BASICAS----------

Lector, Referencia, Emitir, Opinion, Voto y Nota.

//AVANZADAS----------

Listar, Buscar (con sus funciones Compara e Imprime_bus) y Cambiar (con sus funciones Dame_cadenas, Compara2, Dame_confirmacion y Sustituir).

//DAME DATOS----------

Dame_cadena, Dame_numero y Dame_tipo.

//IMPRIMIR----------

void error(int linea).

//LEER TXTs----------

Leer_lector y Leer referencia (con sus funciones Signaturas y Opiniones) y Leer_info.

//ACTUALIZAR TXTs----------

Actualizar_lectores y Actualizar_referencias.

//COMPROBAR TXTs----------

Comprobacion (con sus funciones comprobar_lectores y comprobar_referencias) más las subfunciones Comprobar_puntos, Comprobar_numero, Comprobar_cadena y Comprobar_tipo.

-Lector:

Esta función permite al usuario incorporar un nuevo lector a la base de datos.

Prototipo: void lector (int *ult_lect)

-Recibe por parámetro la dirección de una variable de tipo entero que le indica el tamaño de la tabla de lectores.

-La función es tipo void por lo que no devuelve ningún valor.

Funciones:

Prototipo: void dame_cadena (char *buscar, char *nuevo)

Pseudocodigo:

void lector

//Inicio

Llamamos a la función dame_cadena pidiendo el nombre del lector pasándole por parámetro la longitud máxima que puede tener, el texto a imprimir por pantalla y la dirección de la cadena donde almacenarlo.

Recorremos la struct de lectores comparando el nombre por si ya existe

Si ya existe imprimimos error (4) y retornamos

Si no existe en el último struct de lectores le asignamos el último código más uno.

Copiamos la cadena del lector al último lector del struct

Imprimimos el nombre del ultimo lector más su código.

Incrementamos el contenido de ult_lector en 1.

//FIN

-Referencia:

Esta función permite al usuario incorporar una nueva referencia a la base de datos.

Prototipo: void referencia (int *ult_sign)

-Recibe por parámetro la dirección de una variable de tipo entero que le indica el tamaño de la tabla de referencias.

-La función es tipo void por lo que no devuelve ningún valor.

Funciones:

Prototipo: void dame_cadena (char *buscar, char *nuevo)
Prototipo: int dame_numero (char *buscar, char *nuevo)
Prototipo: char dame_tipo (char *buscar, char *nuevo)

Pseudocodigo:

void referencia

//Inicio

Llamamos a la función dame_cadena pidiendo el nombre del autor pasándole por parámetro la longitud máxima que puede tener, el texto a imprimir por pantalla y la dirección de la cadena donde almacenarlo.

Almacenamos el autor en el último struct de referencias.

Llamamos a la función dame_cadena pidiendo el titulo pasándole por parámetro la longitud máxima que puede tener, el texto a imprimir por pantalla y la dirección de la cadena donde almacenarlo.

Almacenamos el título en el último struct de referencias.

Al apartado año del último struct de referencias igualamos la función dame_numero pasándole por parámetro los límites del número y el texto a imprimir por pantalla.

Al apartado tipo del ultimo struct de referencias igualamos la función dame_tipo.

Igualamos a 0 el apartado crítico del último struct de referencias.

Igualamos a 0 el apartado votantes del ultimo struct de referencias.

Igualamos a 0 el apartado votos del ultimo struct de referencias.

Imprimimos todos los datos de la última referencia que creamos.

Incrementamos el contenido de ult_sign en 1.

//FIN

-Expresar opinión:

Se pasa a expresar una opinión (consistente en un simple comentario en forma de cadena de texto) acerca de una de las referencias existentes en la base de datos. Se emplean cuatro funciones nota, la principal, dame_numero, dame_cadena y error.

Prototipo: void opinion (int ult_sign,int ult_lect);

-Recibe por parámetro, una variable de tipo entero que le indica el tamaño de la tabla de Referencia.

-La función es tipo void por lo que no devuelve ningún valor.

Funciones:

Prototipo: void dame_cadena (int limite, char *pantalla, char *cadena);

Esta función da una cadena de caracteres que cumplan una serie de requisitos.

-Recibe por parámetro tres variables, la primera es de tipo int he indica el número máximo de caracteres de la cadena, el segundo es un puntero a un array que le indica la dicciones de comienzo del enunciado que se va a sacar por pantalla y el tercero es un puntero a un array en donde queremos la cadena introducida.

-Es de tipo void por lo que no retorna ningún valor.

Prototipo: int dame_numero (int limite1, int limite2, char *pantalla);

Esta función saca por pantalla una cadena que le pasamos por parámetro y retorna un número entero entre que se encuentre entre limite1 y limite2.

-Recibe por parámetro tres variables, las dos primeras son enteros que representan los límites entre los que se va a pedir, la ultima es una puntero de un array tipo char que indica el comienzo del enunciado que vamos a sacar por pantalla.
-La función es tipo int por lo que devuelve el valor entero que se introdujo por teclado.

Prototipo: void error (int linea);

Saca por pantalla en un error asociado a la línea que se pasa por parámetro.

-Recibe por parámetro, una variable de tipo entero que le indica la línea a imprimir.
-La función es tipo void por lo que no devuelve ningún valor.

Pseudocodigo:

Void nota (int ult_sign).

-Si el tamaño de la tabla es cero llama a error y pásale el entero 6.

-Sino:

-id=dame_numero

-codigo= dame_numero

-dame_cadena

- Aumentamos el número de críticos.

-Le asignamos el código del lector.

-Pasamos el contenido de la cadena al de la opinión.

-Imprimimos el correcto registro de la opinión.

Return

-Obtener informe:

Se obtiene un informe con los comentarios previamente expresados acerca de una de las referencias existentes en la base de datos. Se emplean tres funciones nota, la principal, dame_numero y error.

Prototipo: void informa (int ult_sign);

-Recibe por parámetro, una variable de tipo entero que le indica el tamaño de la tabla de Referencia.
-La función es tipo void por lo que no devuelve ningún valor.

Funciones:

Prototipo: int dame_numero (int limite1, int limite2, char *pantalla);

Esta función saca por pantalla una cadena que le pasamos por parámetro y retorna un número entero entre que se encuentre entre limite1 y limite2.

-Recibe por parámetro tres variables, las dos primeras son enteros que representan los límites entre los que se va a pedir, la ultima es una puntero de un array tipo char que indica el comienzo del enunciado que vamos a sacar por pantalla.
-La función es tipo int por lo que devuelve el valor entero que se introdujo por teclado.

Prototipo: void error (int linea);

Saca por pantalla en un error asociado a la línea que se pasa por parámetro.

-Recibe por parámetro, una variable de tipo entero que le indica la línea a imprimir.
-La función es tipo void por lo que no devuelve ningún valor.

Pseudocodigo:

Void nota (int ult_sign).

-Si el tamaño de la tabla es cero llama a error y pásale el entero 6.
-Sino:
    -id=dame_numero
    -Imprimimos el número de comentarios de la referencia
    -Si el número de críticos!=0
        -Imprimimos la opinion.
Return

-Emitir votos:

Un voto (consistente en un número entero entre 0 y 10) acerca de una de las referencias existentes en la base de datos. Se emplean tres funciones nota, la principal, dame_numero y error.

Prototipo: void voto (int ult_sign);

-Recibe por parámetro, una variable de tipo entero que le indica el tamaño de la tabla de Referencia.
-La función es tipo void por lo que no devuelve ningún valor.

Funciones:

Prototipo: int dame_numero(int limite1, int limite2, char *pantalla);

Esta función saca por pantalla una cadena que le pasamos por parámetro y retorna un número entero entre que se encuentre entre limite1 y limite2.

-Recibe por parámetro tres variables, las dos primeras son enteros que representan los límites entre los que se va a pedir, la ultima es una puntero de un array tipo char que indica el comienzo del enunciado que vamos a sacar por pantalla.
-La función es tipo int por lo que devuelve el valor entero que se introdujo por teclado.

Prototipo: void error (int linea);

Saca por pantalla en un error asociado a la línea que se pasa por parámetro.

-Recibe por parámetro, una variable de tipo entero que le indica la línea a imprimir.
-La función es tipo void por lo que no devuelve ningún valor.

Pseudocodigo:

Void nota (int ult_sign).

-Si el tamaño de la tabla es cero llama a error y pásale el entero 6.
-Sino:
-id=dame_numero
-voto=dame_numero
-Aumentamos los votos.
-Aumentamos los votantes.
-Samos por pantalla el registro del voto.
Return

-Calcular nota:

Se calcula la nota media de los votos previamente emitidos para una de las referencias existentes en la base de datos. Se emplean tres funciones nota, la principal, dame_numero y error.

Prototipo: void nota (int ult_sign).

-Recibe por parámetro, una variable de tipo entero que le indica el tamaño de la tabla de Referencia.
-La función es tipo void por lo que no devuelve ningún valor.

Funciones:

Prototipo: int dame_numero (int limite1, int limite2, char *pantalla);

Esta función saca por pantalla una cadena que le pasamos por parámetro y retorna un número entero entre que se encuentre entre limite1 y limite2.

-Recibe por parámetro tres variables, las dos primeras son enteros que representan los límites entre los que se va a pedir, la ultima es una puntero de un array tipo char que indica el comienzo del enunciado que vamos a sacar por pantalla.
-La función es tipo int por lo que devuelve el valor entero que se introdujo por teclado.

Prototipo: void error (int linea);

Saca por pantalla en un error asociado a la línea que se pasa por parámetro.

-Recibe por parámetro, una variable de tipo entero que le indica la línea a imprimir.
-La función es tipo void por lo que no devuelve ningún valor.

Pseudocodigo:

Void nota (int ult_sign).

-Si el tamaño de la tabla es cero llama a error y pásale el entero 6.
-Sino:
    -id=dame_numero
    -Imprimimos el número de votantes que tiene ese id.
    -Si los votantes==0->La nota media es 0.00
    -Sino: nota media=votos/votantes.
Return

-Listar referencia:

Mediante esta funcionalidad, se le muestra al usuario un subconjunto de las referencias registradas en la base de datos. Con el siguiente formato SIGNATURA) AUTOR. "TITULO" (ANHO) Se emplean 2 funciones, listar y error, la primera es la principal.

Prototipo: void listar (int ult_sign).

-Recibe por parámetro, una variable de tipo entero que le indica el tamaño de la tabla de Referencia.
-La función es tipo void por lo que no devuelve ningún valor.

Funciones:

Prototipo: void error (int linea);

Saca por pantalla en un error asociado a la línea que se pasa por parámetro.

-Recibe por parámetro, una variable de tipo entero que le indica la línea a imprimir.
-La función es tipo void por lo que no devuelve ningún valor.

Pseudocodigo:

void buscar

//Inicio

Haz esto:

-Pedir por pantalla el tipo de referencia que quieren listar.
-Lo ponemos en un array.
-Pasamos a mayúsculas el carácter
-Si es L p A:
-x=1
-Si tiene más de un carácter:
-X=0
-Si x=0 llamamos a la función error y le pasamos como entero el 8

Mientras x==0

Recorremos la tabla:

-Si el valor tipo de la posición de tabla==al tipo que buscamos:
-Imprimimos.

Return

//Fin

-Buscar referencia:

Esta funcionalidad permite mostrarle al usuario una referencia que tenga un determinado contenido. Se emplean 4 funciones, buscar, compara, imprime_bus y error, la función principal es buscar.

Prototipo: void buscar (int ult_sign).

-Recibe por parámetro, una variable de tipo entero que le indica el tamaño de la tabla de Referencia.

-La función es tipo void por lo que no devuelve ningún valor.

Funciones:

Prototipo: int compara (char *buscada, char *cadena);

Esta función compara dos array de tipo char y devuelve el resultado de la comparación.

-Recibe por parámetro, dos puteros, el primero le indica el comienzo de array de tipo char que será la cadena introducida por teclado. El segundo parámetro indica el comienzo de un array tipo char que está en la tabla.
-La función es de tipo int por lo que devuelve entero. 0 si se ha encontrado coincidencia entre el valor introducido por teclado y el valor de la tabla y 1 si no se ha encontrado coincidencias.

Prototipo: int imprime_bus (int x);

Esta función saca por pantalla la referencia encontrada en el siguiente formato: SIGNATURA) [TIPO] AUTOR. "TITULO" (ANHO), si la referencia tiene opiniones las imprime también, con el siguiente formato: LECTOR1 -> OPINION1.

-Recibe por parámetro, una variable de tipo entero que le indica el valor de la tabla que ha de imprimir por pantalla.
-La función es tipo int, devuelve un cero si se ha impreso correctamente.

Prototipo: void error (int linea);

Saca por pantalla en un error asociado a la línea que se pasa por parámetro.

-Recibe por parámetro, una variable de tipo entero que le indica la línea a imprimir.
-La función es tipo void por lo que no devuelve ningún valor.

Pseudocodigo:

void buscar
//Inicio
Haz esto:
-Pedir cadena de búsqueda por teclado.
-Introducir la cadena recibida por teclado a un array.

-Recorremos el array, paramos en longitud-1.
-Si el carácter es diferente a un espacio o tabulación correcto=1
-Si el primer carácter es un salto de línea correcto=0.
-Si correcto==0 llama a error y pásale el valor 8.

Mientras correcto==0
Introducimos el fin de cadena en el array.
Recorremos la tabla de referencia:
-Si la comparación entre la cadena y el autor es exitosa, imprime por pantalla la referencia, correcto=0.
-Si la comparación entre la cadena y el título es exitosa, imprime por pantalla la referencia, correcto=0.

Si correcto==1
-Saca por pantalla-> "Ninguna referencia contiene la cadena de búsqueda".
//FIN

-Cambiar lector:

A través de esta funcionalidad, el usuario puede cambiar parte del nombre de un (y sólo un) lector. Se emplean 5 funciones, cambiar, dame_cadenas, compara2, dame_confirmacion y sustituir, la función principal es cambiar.

Prototipo: void cambiar (int ult_lect).

-Recibe por parámetro, una variable de tipo entero que le indica el tamaño de la tabla de Lector.
-La función es tipo void por lo que no devuelve ningún valor.

Funciones:

Prototipo: int compara2 (char *buscada, char *cadena, int *inicio2);

Esta función compara dos array de tipo char y devuelve el resultado de la comparación.

-Recibe por parámetro, tres puteros, el primero le indica el comienzo de array de tipo char que será la cadena introducida por teclado. El segundo parámetro indica el comienzo de un array tipo char que está en la tabla. El último puntero es donde se guardara la posición en la cual la comparación es exitosa.
-La función es de tipo int por lo que devuelve entero. 0 si se ha encontrado coincidencia entre el valor introducido por teclado y el valor de la tabla y 1 si no se ha encontrado coincidencias.

Prototipo: void dame_cadenas (char *buscar, char *nuevo);

Esta función obtiene dos arrays de tipo char, el primero se empleara para buscar el nombre del usuario que se quiere cambiar, nuevo será la cadena por el que se sustituirá.

-Recibe por parámetro, dos punteros de dos arrays de tipo char, en el primero se deposita la cadena a buscar y en el segundo el de la cadena por la que se va sustituir.
-La función es tipo void por lo que tras su ejecución no devuelve nada.

Prototipo: char dame_confirmacion (int x);

Saca por pantalla el usuario que se quiere cambiar si la búsqueda ha sido exitosa.

-Recibe por parámetro, una variable de tipo entero que le indica la posición que ocupa el usuario a imprimir en la tabla.
-La función es tipo char por lo que devuelve el carácter de la confirmación.

Prototipo: void sustituir (int x, char *nuevo, char *buscar, int inicio):

Esta función cambiar el nombre de la tabla por el nuevo.
-Recibe parámetros de entrada, el primero es un entero que hace referencia a la posición en la tabla que ocupa el usuario a sustituir, el segundo y tercero son dos punteros a dos arrays, el primero es la cadena por la cual se va sustituir y la segunda la cadena buscada. El último valor que se pasa por parámetro indica la posición en donde comienza la igualdad, a partir de cual se comenzara a sustituir.

-La función es de tipo void por lo que no devuelve ningún valor.

Pseudocodigo:

void cambiar
//Inicio
Llamamos a la función dame_cadenas, y obtenemos dos arrays.
Haz esto:
Mientras, x<num_lectores en tabla o correcto ==0:
-correcto=resultado de la comparación.
-Si correcto==0:
Sigo=confirmación.
-Si sigo==S:
-Cambio el nombre.
-Correcto=1; retorno
-Si sigo==N:
-correcto=0
Si correcto!=1 o cambio==0:
-Saco por pantalla: “Ningún lector cambiado”
-retorno.

Mientras, x<num_lectores o sigo ==S

//FIN

Prototipo: void actualizar_lector (int ult_lect)

Esta función actualiza el txt lectores.txt con el contenido del struct lectores. Imprime una línea con el formato :código: lector:\n correspondiente a cada lector hasta llegar al ultimo

-Por parámetro le pasamos el total de lectores.

Prototipo: void actualizar_refe (int ult_sing)

Esta función actualiza el txt referencias.txt con el contenido del struct referencias y el struct informacion. Imprime la línea de cada asignatura seguida las líneas de sus opiniones si las tiene, así hasta imprimir el total de signaturas

-Por parámetro le pasamos el total de signaturas.

Prototipo: int leer_lector()

Esta función lee el contenido del txt lectores.txt y lo almacena en el struct lectores

-Retorna el número total de lectores del fichero lectores.txt

Prototipo: int leer_referencia ()

Esta función lee el contenido del txt referencias.txt y lo almacena en el struct referencias y el struct información.

-Retorna el número total de signaturas del fichero referencias.txt

Prototipo: void leer_info (char *cadena, char *texto , int pos)

Esta función lee las cadenas de los txt existentes entre dos ‘:’ establecidos.

-Por parámetro le pasamos el puntero a la cadena donde almacenar la información, el puntero a la cadena extraída del txt y la posición de los ‘:’ a partir de los cuales nos interesa la información

Prototipo: void signaturas (char *cadena, int *sign)

Esta función almacena el contenido del txt en el struct referencias

-Por parámetro el puntero a la cadena extraída del txt y el entero de la signatura a guardar

Prototipo: void opiniones (char *cadena, int *opinion, int *sign)

Esta función almacena el contenido del txt en el struct información de una referencia determinada

-Por parámetro el puntero a la cadena extraída del txt y el entero de la opinión a guardar y la signatura a la que corresponde.

Prototipo: void dame_cadena (int limite, char *pantalla, char *cadena);

Esta función da una cadena de caracteres que cumplan una serie de requisitos.

-Recibe por parámetro tres variables, la primera es de tipo int he indica el número máximo de caracteres de la cadena, el segundo es un puntero a un array que le indica la dirección de comienzo del enunciado que se va a sacar por pantalla y el tercero es un puntero a un array en donde queremos la cadena introducida.
-Es de tipo void por lo que no retorna ningún valor.

Prototipo: int dame_numero (int limite1, int limite2, char *pantalla);

Esta función saca por pantalla una cadena que le pasamos por parámetro y retorna un número entero entre que se encuentre entre limite1 y limite2.

-Recibe por parámetro tres variables, las dos primeras son enteros que representan los límites entre los que se va a pedir, la ultima es una puntero de un array tipo char que indica el comienzo del enunciado que vamos a sacar por pantalla.
-La función es tipo int por lo que devuelve el valor entero que se introdujo por teclado.

Prototipo: char dame_tipo ();

Esta función devuelve un carácter 'L' o 'A', según se le haya introducido únicamente 'l' 'L' 'a' o 'A'.

-La función es de tipo char y devuelve el carácter 'L' o 'A' según lo que se le introduzca.

Prototipo: void error (int linea);

Saca por pantalla en un error asociado a la línea que se pasa por parámetro.

-Recibe por parámetro, una variable de tipo entero que le indica la línea a imprimir.
-La función es tipo void por lo que no devuelve ningún valor.