Programación

Ejercicios Tema 6
Tipos de Datos Estructurados:
Estructuras Definidas por el Usuario

Autores:

M. Paz Sesmero Lorente
Paula de Toledo Heras
Fco. Javier Ordóñez Morales
Juan Gómez Romero
José A. Iglesias Martínez
José Luis Mira Peidro

SOLUCIONES

1. Para la gestión de los libros de una pequeña biblioteca es preciso conocer su título, su autor, el ISBN (cadena de 17 caracteres: 978-3-16-148410-0) y si está prestado o no. Escribir un programa en C que:

a) Defina una estructura denominada fichaLibro que permita almacenar los datos de cualquier libro

b) Declarare e inicialice dos variables del tipo fichaLibro

c) Compruebe si las variables declaradas representan ejemplares de un mismo libro o de libros distintos.

```
#include <stdio.h>
#include <stdlib.h>
/*-------------- Definir estructuras   -------------   */
struct fichaLibro {
       char titulo[20];        //Titulo del libro
       char autor [20];        //Autor del libro
       char ISBN[18];          //Debe almacenar 17 caracteres +
el caracter nulo
       int prestado;           //0: No prestado; 1: presado
};

int main (void){
   //Declaracion e inicialización de las variables
   struct fichaLibro libro1={"titulo1", "autor1", "000-0-00-
000000-0", 0};
   struct fichaLibro libro2={"titulo2", "autor2", "000-0-00-
000000-2", 1};
   //struct FichaLibro Libro2={"titulo1", "autor1", "000-0-00-
000000-0", 0};
   //Para probar el caso en el que las dos variables son
ejemplares del mismo libro

   int iguales=0; //1: Iguales; 0: Distintas.
                  //Inicialmente son distintos

   /*Para ver si las variables son idénticas es necesario
     comparar todos los campos. En este caso se omite la
     comparación respecto  al campo prestado*/
   if (strcmp(libro1.titulo,libro2.titulo)==0)
      if (strcmp(libro1.autor,libro2.autor)==0)
         if  (strcmp(libro1.ISBN,libro2.ISBN)==0)
              iguales=1;
   if (iguales==1)
      printf ("Las dos variables representan al mismo libro\n");
   else
      printf ("Los libros son distintos\n");

   return 0;
}
```

2. Una fábrica de tornillos nos ha solicitado desarrollar un programa en C que permita obtener e imprimir las estadísticas de una determinada muestra de tornillos. Para ello nos indican que cada tornillo viene caracterizado por su longitud y su diámetro y que ambas medidas deben almacenarse en un registro con dos campos. Desarrollar un programa en C que:

a) Solicite al usuario los datos de una muestra de tornillos (longitud y diámetro) y los almacene en un vector

b) A partir de los datos introducidos calcule y muestre por pantalla la media de las longitudes y la media de los diámetros.

Nota: El tamaño de la muestra se definirá como una constante denominada TAMANIO_MUESTRA.

```
#include <stdio.h>

#define TAMANIO_MUESTRA 5

//Declaracion de la estructura
struct tornillo
{    float longitud;        /*longitud del tornillo*/
     float diametro;        /*diámetro del tornillo*/
};

int main(void) {
     int i;
     float sumaLongitud=0;
     float sumaDiametro=0;
     float mediaLongitud;
     float mediaDiametro;
     struct tornillo  muestra[TAMANIO_MUESTRA];

    //Se solicitan los datos
    for(i=0;i<TAMANIO_MUESTRA; i++){
       printf ("Introduzca la longitud del tornillo: ");
       scanf  ("%f", &muestra[i].longitud);
       printf ("Introduzca el diámetro del tornillo: ");
       scanf  ("%f", &muestra[i].diametro);
     }

     //Se calcula la suma de todas las longitudes y todos los
diametros
     for(i=0;i<TAMANIO_MUESTRA; i++){
        sumaLongitud=sumaLongitud+muestra[i].longitud;
        sumaDiametro=sumaDiametro+muestra[i].diametro;
     }

     //Se calculan las medias
     mediaLongitud=sumaLongitud/TAMANIO_MUESTRA;
     mediaDiametro=sumaDiametro/TAMANIO_MUESTRA;

     //Se muestran los resultados por pantalla
     printf("La media de longitudes es: %f cm\n", mediaLongitud);
     printf("La media de diametros es: %f cm\n", mediaDiametro);

     return 0;
}
```

3. Un negocio de paquetería con entregas a domicilio nos ha solicitado un programa para almacenar datos sobre las calles y las viviendas de su Ciudad. Cada calle se identificará por dos elementos: Nombre y viviendas que tiene. Además, cada vivienda estará identificada por el número que ocupa en la calle y la planta. Admitiendo que la ciudad tiene 5 calles y cada calle tiene cinco viviendas, desarrollar un programa que permita al usuario:

a) Almacenar los datos sobre las calles y las viviendas de la ciudad.

b) Ver toda la información sobre una calle: número y viviendas que tiene

Nota: Por simplicidad se admitirá que los nombres de las calles no contienen espacios.

```
#include <stdio.h>

#define MAX_CALLES 2
#define MAX_CASAS  3

/*Definición de las estructuras*/

struct casa {
       int numero;        /*número de la Calle en el que está
situada la casa*/
       int planta;        /*piso que ocupa la vivienda*/
};

struct calle {
       char nombre[20];
       struct casa casas[MAX_CASAS];
};

void inicializar(struct calle ciudad[]);
int buscar(char nombreCalle[], struct calle ciudad[]);

int main(void) {

    int i,j;
    char nombreCalle[20];
    struct calle ciudad[MAX_CALLES];

    //Se solicitan los datos
    inicializar(ciudad);

    //Consultar los datos de una calle
    printf ("/nIntroduzca el nombre de una calle: ");
    scanf ("%s", nombreCalle);
    i=buscar(nombreCalle, ciudad);

   if (i==-1)
      printf ("no existe esa calle\n");
   else{
        printf("La calle %s tiene las siguientes viviendas:\n",
nombreCalle);
        for (j=0;j<MAX_CASAS;j++){
            printf("Numero: %d Planta %d \n",
ciudad[i].casas[j].numero, ciudad[i].casas[j].planta);
        }
   }
   return 0;
}
```

```
void inicializar(struct calle ciudad[]){
   /*Inicializa las componentes del vector ciudad
     Parámetros:  struct calle ciudad[]
     Retorno:     Ninguno */
   int i, j;
   for(i=0;i<MAX_CALLES; i++){
      printf ("Introduzca el nombre de la calle: ");
      scanf  ("%s", ciudad[i].nombre);

      for (j=0;j<MAX_CASAS;j++){
          printf ("Introduzca el numero de la calle en el que esta
situada la casa: ");
          scanf ("%d", &ciudad[i].casas[j].numero);
          printf ("Introduzca el numero de planta: ");
          scanf ("%d", &ciudad[i].casas[j].planta);
      }
    }
   return;
}

int buscar(char nombreCalle[], struct calle ciudad[]){
     /*Busca una calle dentro de la estructura ciudad
     Parámetros:  struct calle ciudad[]
                  char nombreCalle -> Calle a buscar
     Retorno:     Pos que ocupa nombreCalle en el vector ciudad
                  Si no está devuelve -1 */
    int i;
    int encontrado=0;

    //Se busca la calle dentro de la ciudad.
     i=0;
     while ((i<MAX_CALLES)&&(encontrado==0)){
       if (strcmp (ciudad[i].nombre, nombreCalle)==0)
          encontrado=1;
       else  //En la siguiente iteración se analiza la siguiente
             //componente del vector
         i=i+1;
   }
   if (!encontrado)
       i=-1;
   return (i);
}
```

4. Escriba un programa que, tratando una fracción como un estructura de 2 componentes, permita realizar las operaciones suma, multiplicación, división, opuesto e inverso. El programa deberá preguntar la operación a realizar y después los datos para ejecutarla, y repetir estos pasos hasta que se seleccione entre las operaciones la opción terminar. Cada una de las operaciones se implementará como una función.

```
#include <stdio.h>
struct fraccion {
          int num;
         int den;
};

int Mostrar_Menu(void);
struct fraccion suma(struct fraccion f1, struct fraccion f2);
struct fraccion multiplicar(struct fraccion f1, struct fraccion
```

```
f2);
struct fraccion dividir(struct fraccion f1, struct fraccion f2);
struct fraccion opuesto(struct fraccion f);
struct fraccion inverso(struct fraccion f);


int main()
{
    struct fraccion f1,f2, resultado;
    int op;

    do{
      op=Mostrar_Menu();
      switch (op)
      {
        case 1:{
            printf("Introduzca num y denominador de f1\n");
            scanf("%d", &f1.num);scanf("%d", &f1.den);
            printf("Introduzca num y denominador de f2\n");
            scanf("%d", &f2.num);scanf("%d", &f2.den);

            resultado=suma(f1,f2);
            printf("El resultado de la suma es
%d/%d\n",resultado.num, resultado.den);
            break;
          };
        case 2:{
            printf("Introduzca num y denominador de f1\n");
            scanf("%d", &f1.num);scanf("%d", &f1.den);
            printf("Introduzca num y denominador de f2\n");
            scanf("%d", &f2.num);scanf("%d", &f2.den);

            resultado=multiplicar(f1,f2);
            printf("El resultado de la multiplicacion es
%d/%d\n",resultado.num, resultado.den);
            break;
        };
        case 3:{
            printf("Introduzca num y denominador de f1\n");
            scanf("%d", &f1.num);scanf("%d", &f1.den);
            printf("Introduzca num y denominador de f2\n");
            scanf("%d", &f2.num);scanf("%d", &f2.den);

            resultado=dividir(f1,f2);
            printf("El resultado de la division es
%d/%d\n",resultado.num, resultado.den);
            break;
        };
        case 4:{
            printf("Introduzca num y denominador de f\n");
            scanf("%d", &f1.num);scanf("%d", &f1.den);

            resultado=opuesto(f1);
            printf("El opuesto de f1 es %d/%d\n",resultado.num,
resultado.den);
            break;
      };
```

```
case 5:{
            printf("Introduzca num y denominador de f\n");
            scanf("%d", &f1.num);scanf("%d", &f1.den);

            resultado=inverso(f1);
            printf("El inverso de f1 es %d/%d\n",resultado.num,
resultado.den);
            break;
      };
        case 6:
            break;
        default:
           printf("La operación seleccionada no es válida\n");

      }
    }while (op!=6);
    return(0);

}

int Mostrar_Menu(void)
{
   int op;
   printf("Seleccione la operación a realizar\n");
   printf("1. Suma\n");
   printf("2. Multiplicacion\n");
   printf("3. Division\n");
   printf("4. Opuesto\n");
   printf("5. Inverso\n");
   printf("6. Salir\n");
   scanf("%d", &op);
   return (op);

}
struct fraccion suma(struct fraccion f1, struct fraccion f2)
{
   struct fraccion r;
   r.num=f1.num*f2.den+f2.num*f1.den;
   r.den=f1.den*f2.den;

   return (r);
}

struct fraccion multiplicar(struct fraccion f1, struct fraccion f2)
{
   struct fraccion r;
   r.num=f1.num*f2.num;
   r.den=f1.den*f2.den;

   return (r);
}

struct fraccion dividir(struct fraccion f1,struct  fraccion f2)
{
   struct fraccion r;
   r.num=f1.num*f2.den;
   r.den=f1.den*f2.num;
   return (r);
}
```

```
struct fraccion opuesto(struct fraccion f)
{
   struct fraccion r;
   r.num=-f.den;
   r.den=f.num;
   return (r);
}

struct fraccion inverso(struct fraccion f)
{
   //Intercambia los términos de la fraccion
   struct fraccion r;
   r.num=f.den;
   r.den=f.num;
   return (r);
}
```

5. Escriba un programa en C que permita almacenar mediante el uso de estructuras los datos de 100 clientes de una empresa de automóviles. El programa permitirá ir almacenando los datos de los clientes, asignándole a cada uno un número según el orden en el que se van grabando. Los datos que se guardarán de cada cliente son el nombre, los apellidos, el teléfono y e-mail. Tras leer un cliente se dará la opción de terminar tecleando un 0.
Posteriormente modifique el programa para que evite que un cliente sea guardado dos veces. Para ello se comparará el teléfono con los ya guardados y en caso de que estuviera repetido se mostrará el nombre del cliente, el teléfono repetido y la posición en la que está grabado.

```
#include <stdio.h>

#define MAX_CLIENTES 100

/*Definicion de la estructura datos_personales.
Esta estructura almacena los datos personales de cada cliente:
Nombre
Apellido1
Apellido2
Telef.
e_mail*/

#include <stdio.h>
struct datosPersonales
        {char nombre [10];
        char apellido1 [10];
        char apellido2 [10];
        int telefono;
        char e_mail [20];
        };


struct datosPersonales leerDatosCliente(void);
int comprobarTelefono( struct datosPersonales clientes[], int tam,
int telf);
void mostrarClientes (struct datosPersonales clientes[], int tam);
int main()
{
    //Se define la variable clientes. Esta variable debe almacenar
los datos de 100 clientes.
    //Se declara como un vector de 100 elementos en el que cada
```

```
elemento es del tipo datosPersonales
    struct datosPersonales clientes[MAX_CLIENTES];
    struct datosPersonales nuevoCliente;

    int repetido;
    int nClientes=0;
    int salir; //variable para controlar cuando no se quieren
grabar más clientes

    printf("Introduzca los datos de los clientes\n");
    printf("Use 0 para terminar\n");
    printf("maximo 100 clientes\n");

    /*Se solicitan los datos del primer cliente y se introduce
      en el vector*/
    clientes[0]=leerDatosCliente();
    nClientes++;

    printf("Introduzca 0 para terminar, cualquier otro valor para
seguir leyendo datos\t");
    scanf("%i", &salir);
    while ((nClientes<MAX_CLIENTES)&&(salir!=0)){
        //leer los datos del cliente i
        nuevoCliente=leerDatosCliente();
        repetido=comprobarTelefono(clientes, nClientes,
nuevoCliente.telefono);
        if (!repetido){
          clientes[nClientes]= nuevoCliente;
          nClientes++;
        }
        printf("Introduzca 0 para terminar, cualquier otro valor
para seguir leyendo datos\t");
        scanf("%i", &salir);
    } // while ((nClientes<MAX_CLIENTES)&&(salir!=0));
    mostrarClientes(clientes, nClientes);
    return 0;
}

struct datosPersonales leerDatosCliente(void){
    /*Lee los datos relativos a un cliente:
          Parámetros: Niguno
          Retorno:    Estructura con los datos leidos*/

    struct datosPersonales cliente;
    printf ("Introduzca el nombre y los dos apellidos del cliente
\n");
    scanf ("%s", cliente.nombre);
    scanf ("%s", cliente.apellido1);
    scanf ("%s", cliente.apellido2);
    printf ("Introduzca el telefono y e-mail del cliente\n");
    scanf ("%i", &cliente.telefono);
    scanf ("%s", cliente.e_mail);

    return cliente;
}
```

```
int comprobarTelefono( struct datosPersonales clientes[], int tam,
int telf){
    /*Comprueba si un telf coincide con alguno de los telefonos
      de los clientes cuyos datos están almacenados en el vector
      Parámetros:  clientes -> Vector con los clientes actuales
                        tam -> Núm de elmentos del vector
                        telf-> Telefono a comprobar
      Retorno:     int  0 si no hay coincidencia; 1 en c.c.*/
    int j=0;
    int repetido=0;
    while ((j<tam) && (repetido==0)){
       if (clientes[j].telefono==telf){
            repetido=1;
            printf("Imposible grabar los datos\n");
            printf("El telefono %d ya esta asignado al cliente\n",
clientes[j].telefono);
            printf("%s %s\n", clientes[j].nombre,
clientes[j].apellido1);
            printf("almacenado en la posición %d\n", j);
        }
        j++;
    }
    return repetido;
}

void mostrarClientes (struct datosPersonales clientes[], int tam){
   /*Muestra por pantalla los datos de los clientes almacenados
     Parámetros:   clientes-> Vector con los clientes actuales
                       tam-> Núm de elementos del vector
     Retorno:      nada*/
   int j;
   printf("La empresa tiene %i clientes\n", tam);
   for (j=0;j<tam;j++){
       printf ("Cliente %i : %s  %s  %s \n", j, clientes[j].nombre,
clientes[j].apellido1, clientes[j].apellido2);
       printf ("\t telefono: %i\n", clientes[j].telefono);
       printf ("\t e_mail: %s\n", clientes[j].e_mail);
   };
   return;
}
```

6. Un amigo nos ha solicitado que desarrollemos un programa en C que le ayude a organizar su discografía. Nos indica que le gustaría tener sus temas organizados por autor, que para cada autor quiere almacenar sus datos (nombre, fecha de nacimiento y origen), número de temas que de ese autor posee y la descripción de todos sus temas (título, CD en el que está incluido y año de publicación).
El programa debe comenzar con un menú que permita introducir datos o mostrar todas las canciones de un año. En el primer caso irá pidiendo canciones y tras cada una, preguntará si se quiere continuar. En el segundo, preguntará por el año y mostrará todas las canciones en de ese año.

Notas:
En la resolución del problema se deben usar estructuras anidadas.
El número de autores que componen la discografía y el número máximo de canciones por autor se definirán como constantes: N_AUTORES y NC

```
#include <stdio.h>

#include <string.h>

#define N_AUTORES    20   //Consideramos un máximo de 20 autores
#define NC           10   //Consideramos un máximo de 10 canciones
por autor

  //Estructura para almacenar los datos de autor: Nombre, fecha de
nacimiento y origen
  struct datosAutor {
         char nombre [20];
         int fechaNacimiento;
         char origen [20];};

  //Estructura para almacenar la información de una canción:
Nombre, CD y año de edición)
  struct datosCancion {
         char titulo [20];
         char CD [20];
         int edicion;};

  //Estructura para almacenar los datos de una colección de discos:
Datos del autor, datos de las canciones y numero de canciones de
ese autor.
  //Se considera que un autor puede tener como mucho NC canciones
  struct Discos {
         struct datosAutor autor;
         struct datosCancion song[NC];
         int cancionesAutor; };



int mostrarMenu(void);
void introducirDatos(struct Discos Discografia[N_AUTORES], int n);
void mostrarCancionesAnio(struct Discos Discografia[N_AUTORES], int
n);
int main()
{
  //Se define la variable Discografia. Esta variable debe almacenar
los datos relativos a todas nuestras canciones.

  struct Discos Discografia[N_AUTORES];
  int n=0;        //Numero de autores que hay en mi discografía
(Inferior a N_AUTORES)
  int i;
  int continuar;
  int op;

  //Inicialización de variables. Inicialmente no se tienen datos
almacenados.
  for (i=0; i<N_AUTORES; i++)
      Discografia[i].cancionesAutor=0;

  //Se muestra el menu por pantalla
  do{
     op=mostrarMenu();
```

```
      switch (op){
        case 1:
        {
           do{
            if (n<N_AUTORES){
               introducirDatos(Discografia, n);
               n++;
               printf ("Desea introducir canciones de otros
autores?(Si=1/No=0)");
                    scanf("%i", &continuar);
             }
             else
               printf ("Su discografia está completa\n");
             }while((continuar==1)&&(n<N_AUTORES));
         break;
       }
       case 2:
       {
             mostrarCancionesAnio(Discografia, n);
             break;
       }
         case 0:
              break;
         default:
              printf ("La opcion introducida no es correcta\n");
      }//switch
   }while (op!=0);

   return (0);
}

int mostrarMenu(void){
     int op;
     printf ("Seleccione una de las siguientes opciones\n");
     printf ("1: Introducir datos\n");
     printf ("2: Mostrar canciones\n");
     printf ("0: Salir\n");
     scanf("%d", &op);
     return op;
}

void introducirDatos(struct Discos Discografia[N_AUTORES], int n){
      int continuar=1;

      printf("Datos del Autor\n");
      printf("Nombre\t\t");
      scanf(" %[^\n]",Discografia[n].autor.nombre);

      printf("Anio_Nac.\t");
      scanf("%i",&Discografia[n].autor.fechaNacimiento);

      printf("Origen\t\t");
      scanf(" %[^\n]",Discografia[n].autor.origen);

      do{
         printf("Datos de la Cancion\n");
         printf("Titulo\t");
         scanf("
%[^\n]",Discografia[n].song[Discografia[n].cancionesAutor].titulo);
          printf("CD\t");
```

```
            scanf("
%[^\n]",Discografia[n].song[Discografia[n].cancionesAutor].CD);
              printf("Edicion\t");

scanf("%i",&Discografia[n].song[Discografia[n].cancionesAutor].edic
ion);
            Discografia[n].cancionesAutor++; //Se actualiza el
contador de canciones/autor

            printf ("Desea introducir más canciones del mismo
autor?(Si=1/No=0)");
            scanf("%i", &continuar);
         }while ((continuar==1)&&(Discografia[n].cancionesAutor<NC));
         return;
}

void mostrarCancionesAnio(struct Discos Discografia[N_AUTORES], int
n){
      int i,j;
      int year;
      printf ("Introduzca el year\n");
      scanf ("%i", &year);
      printf ("En el año %i se editaron las siguientes
canciones:\n", year);

      for (i=0; i<n; i++){
         for (j=0; j<Discografia[i].cancionesAutor; j++){
              if (Discografia[i].song[j].edicion==year){
                    printf ("Cancion: %s\t Autor: %s \n",
Discografia[i].song[j].titulo, Discografia[i].autor.nombre);
              }
         }
      }
   return;
}
```

7. Implementar en C un algoritmo que permita gestionar un array de estructuras para archivar 150 libros que se pueden adquirir en Amazon. Los libros están clasificados por área temática (1= Novela negra 2=Novela histórica 3=Informática 4=Ciencias sociales 5=Ensayo), el autor, palabra clave del título, año de publicación, posición en ranking de ventas. Los compradores quieren poder acceder a los libros más vendidos dentro de cada área temática. Para ello el algoritmo nos debe permitir realizar la búsqueda por área temática y que muestre los libros ordenados de los más a menos vendidos.

```
#include <stdio.h>

#include <string.h>

#define DIM 150
struct biblio {
         int temat;
         char autor[30];
         char clave [30];
         int year;
         float ranking;
       };
```

```
int  inicializar(struct biblio libros[]);
void mostrarDatos(struct biblio libros[], int n);
int  solicitarArea(void);
void mostrarMenu(int area);
int  seleccionarPorArea(struct biblio libros[], int n, int area,
struct biblio selec[]);
void ordenar(struct biblio selec[], int nAreaSelec);
int main()
{
    /*Declaro dos arrays de estructuras sobre la estructura biblio,
     la segunda es para almacenar los libros seleccionados*/

     struct biblio libros[DIM];
     struct biblio selec[DIM]; //Almacena los documentos de un área
     int n;          //Almacena el núm. de documentos
     int nAreaSelec;//Almacena el número de documentos de un área

     int i,j, area;

      n=inicializar(libros);
      printf ("Mis libros por orden de lista son:\n");
      mostrarDatos(libros, n);
      area=solicitarArea();
      mostrarMenu(area);
      nAreaSelec=seleccionarPorArea(libros, n, area, selec);
      //Imprimimos las libros seleccionadas sin ordenar
      printf ("Los libros incluidos en el area temática %d son:\n",
i);
      mostrarDatos(selec, nAreaSelec);
      ordenar(selec, nAreaSelec);
     //Imprimimos los libros seleccionados ORDENADOS por ranking de
ventas
      printf ("Los libros seleccionados ordenados por ranking de
ventas son:\n");
      mostrarDatos(selec, nAreaSelec);

   system("PAUSE");
   return 0;
}


int inicializar(struct biblio libros[]){
    //Se inicializa el array de estructuras miembro a miembro
    //La inicialización termina cuando se han introducido 150
    //documentos o cuando se ha introducido toda la bibliografía.
    //Se admite que la bibliografía contiene al menos un docuento.
    int n=0;
    int continuar;

     do{
        printf ("Introduzca Tematica, autor, clave, year y ranking
de ventas del num %i de la lista\n", n+1);
        scanf ("%i" "%s" "%s" "%i" "%f" , &libros[n].temat,
libros[n].autor,
libros[n].clave,&libros[n].year,&libros[n].ranking);
        n++;
        printf ("Desea introducir más datos: Si:1  No:0 \n");
        scanf ("%d", &continuar);
     }while ((n<DIM)&&(continuar==1));
```

```
    return n;
}

void mostrarDatos(struct biblio libros[], int n){
   int i;
   //Se muestran los datos almacenados
   for(i=0; i<n; i++){
      printf ("%i %s %s %i %3.2f\n", libros[i].temat,
libros[i].autor, libros[i].clave,libros[i].year,libros[i].ranking);
   }
   return;
}

int solicitarArea(void){
    int area;
    printf ("Introduzca un area tematica\n");
    do{
        scanf ("%i",&area);
        if ((area<1)||(area>5))
           printf ("Area Tematica no valida, introduzca nueva
area\n");

    }while ((area<1)||(area>5));
    return area;
}

void mostrarMenu(int area){
    printf ("La busqueda se realizara en el area tematica: ");
    switch (area){
       case 1:
            printf ("Novela Negra\n");
            break;
       case 2:
            printf ("Novela Historica\n");
            break;
       case 3:
            printf ("Informatica\n");
            break;
       case 4:
            printf ("Ciencias Sociales\n");
            break;
       case 5:
            printf ("Ensayo\n");
            break;

    }
    return;
}

int seleccionarPorArea(struct biblio libros[], int n, int area,
struct biblio selec[]){
    int nArea=0;
    int i;
    for (i=0; i<n; i++){
        if (libros[i].temat==area){
              selec[nArea]=libros[i];
          nArea++;
         }
     }
     return nArea;
```

```
}

void ordenar(struct biblio selec[], int nAreaSelec){
    int i, j;
    struct biblio aux;
    for(i=1;i<=nAreaSelec-1;i++){
      //coge los N-i primeros elementos y compáralos con
      // el inmediatamente posterior
      for(j=0;j<nAreaSelec-i;j++){
      // si estan desordenados se intercambian
          // usando una variable auxiliar.
            if(selec[j].ranking<selec[j+1].ranking) {
              aux=selec[j+1];
              selec[j+1]=selec[j];
              selec[j]=aux;
            }
       }
    }

   return;
}

/*int inicializar (struct biblio libros[]){
     // Inicializaciones para probar el programa:
     libros[0].temat=1;
     strcpy(libros[0].autor,"Aut1");
     strcpy(libros[0].clave,"p1");
     libros[0].year=1980;
     libros[0].ranking=0.70;

     libros[1].temat=2;
     strcpy(libros[1].autor,"Aut2");
     strcpy(libros[1].clave,"p2");
     libros[1].year=1981;
     libros[1].ranking=0.75;

     libros[2].temat=2;
     strcpy(libros[2].autor,"Aut3");
     strcpy(libros[2].clave,"p3");
     libros[2].year=1981;
     libros[2].ranking=1.70;

     libros[3].temat=1;
     strcpy(libros[3].autor,"Aut4");
     strcpy(libros[3].clave,"p4");
     libros[3].year=1981;
     libros[3].ranking=1.75;

     libros[4].temat=2;
     strcpy(libros[4].autor,"Aut5");
     strcpy(libros[4].clave,"p5");
     libros[4].year=1980;
     libros[4].ranking=1.70;

     libros[5].temat=1;
     strcpy(libros[5].autor,"Aut6");
     strcpy(libros[5].clave,"p6");
     libros[5].year=1973;
     libros[5].ranking=2.50;
```

```
        libros[6].temat=1;
        strcpy(libros[6].autor,"Aut7");
        strcpy(libros[6].clave,"p7");
        libros[6].year=1980;
        libros[6].ranking=0.20;

        libros[7].temat=1;
        strcpy(libros[7].autor,"Aut8");
        strcpy(libros[7].clave,"p8");
        libros[7].year=1973;
        libros[7].ranking=1.30;

        return 8;
}       */
```

8. Implementar en C un programa que maneje un array de estructuras para almacenar las fechas de cumpleaños de 20 amigos. Los datos a almacenar sobre cada amigo son nombre, día, mes y año. El programa debe indicarnos cuántos de nuestros amigos cumplen los años un día y mes determinados introducidos por teclado.

```
#include <stdio.h>
#include <stdlib.h>
#define DIM 5

struct tipoFecha {
        int dia;
        int mes;

};
struct tipoAmigo {
        char nombre[30];
        struct tipoFecha cumple;
};


void Inicializar (struct tipoAmigo agenda[], int l);
void Mostrar (struct tipoAmigo agenda[], int l);
struct tipoFecha Solicitar_Fecha(void);
void Buscar_Amigos(struct tipoAmigo agenda[], int l, struct
tipoFecha f);

int main(void)
{
   //Vector en el que se almacenan los datos de mis amigos
   struct tipoAmigo amigos[DIM];
   //Fecha que queremos comprobar
   struct tipoFecha fecha;

   Inicializar(amigos, DIM);
   Mostrar(amigos, DIM);
   fecha=Solicitar_Fecha();
   Buscar_Amigos(amigos, DIM, fecha);


  system("PAUSE");
  return 0;
}
```

```
//Definicición de las funciones:
void Inicializar (struct tipoAmigo agenda[], int l){
     //Inicializa los datos de nuestros 20 amigos
     int i;
     for (i=0; i<l; i++){
            printf ("Introduzca el nombre, dia, mes del amigo %i de
la lista\n", i+1);
            scanf ( "%s" "%i" "%i", agenda[i].nombre,
&agenda[i].cumple.dia, &agenda[i].cumple.mes);
   }
   return;

}

void Mostrar (struct tipoAmigo agenda[], int l){
     //Muestra los datos de nuestros 20 amigos
     int i;
     for (i=0; i<l; i++){
            printf ("%s\t %2i/%2i\n", agenda[i].nombre,
agenda[i].cumple.dia, agenda[i].cumple.mes);
   }
   return;
}


struct tipoFecha Solicitar_Fecha(void){
    struct tipoFecha f;
    printf ("Introduzca dia y mes\n");
    scanf ("%i", &f.dia);
    scanf ("%i", &f.mes);
    return (f);
}

void Buscar_Amigos(struct tipoAmigo agenda[], int l, struct
tipoFecha f){
     //Busca todos los amigos que cumplen años en f.
   int i;
   int contador=0;
   for (i=0; i<l; i++)
   {
      if ((agenda[i].cumple.dia==f.dia) &&
(agenda[i].cumple.mes==f.mes))
      {
         printf ("%s cumple aï¿½os el dia indicado\n",
agenda[i].nombre);
         contador++;
      }
   }
   if (contador==0)
      printf ("No tienes amigos que cumplan aï¿½os el %2d/%2d\n",
f.dia, f.mes);
   return;
}
```