```c

#include "fechas.h"

int ingresarFechaDMA(tFecha *fec, const char *mensajeOpcional)
{
    if(mensajeOpcional)
        printf("%s", mensajeOpcional);
    else
        printf("Fecha (dd/mm/aaaa - 0=No Ingresa): ");
    fec->di = 0;
    fec->me = 0;
    fec->an = 0;
    fflush(stdin);
    scanf("%d/%d/%d", &fec->di, &fec->me, &fec->an);
    return fec->di && fec->me && fec->an;
}

void mostrarFechaDMA(const tFecha *fec)
{
    printf("%02d/%02d/%04d", fec->di, fec->me, fec->an);
}

int esFechaValida(const tFecha *fec)
{
    static const char dias[][12] = {
                        { 31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31 },
                        { 31, 29, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31 } };
    return  fec->me > 0 && fec->me <= 12 &&
            fec->an >= AN_MIN && fec->an <= AN_MAX &&
            fec->di > 0 && fec->di <= dias[esBisiesto(fec->an)][fec->me - 1];
}

int ingresarFechaValidaDMA(tFecha *fec, const char *mensajeOpcional)
{
    do
    {
        if(!ingresarFechaDMA(fec, mensajeOpcional))
            return 0;
    } while(!esFechaValida(fec));
    return 1;
}

int compararFecha(const tFecha *fec1, const tFecha *fec2)
{
    int cmp = fec1->an - fec2->an;

    if(cmp)
        return cmp;
    cmp = fec1->me - fec2->me;
    if(cmp)
        return cmp;
    return fec1->di - fec2->di;
}

int aJuliano(const tFecha *fec)
{
    int dias[][12] = {
            { 0, 31, 59, 90, 120, 151, 181, 212, 243, 273, 304, 334 },
            { 0, 31, 60, 91, 121, 152, 182, 213, 244, 274, 305, 335 } };

    return dias[esBisiesto(fec->an)][fec->me - 1] + fec->di;
}

int nroDeDiaDeLaSemana(const tFecha *fec)
{
    int distBase = fec->an % 400;

    return ( SABADO - 1 + distBase + distBase / 4 - distBase / 100 +
             (distBase != 0) + aJuliano(fec) ) % 7;
}

long diasEntreFechas(const tFecha *fecDesde, const tFecha *fecHasta)
{
    int anBase = fecDesde->an <= fecHasta->an ? fecDesde->an : fecHasta->an,
        distBaseDesde = fecDesde->an - anBase,
        distBaseHasta = fecHasta->an - anBase;
    long diasBaseAHasta = distBaseHasta * 365L + distBaseHasta / 4 -
                          distBaseHasta / 100 + distBaseHasta / 400 +
                          (distBaseHasta != 0) + aJuliano(fecHasta),
         diasBaseADesde = distBaseDesde * 365L + distBaseDesde / 4 -
                          distBaseDesde / 100 + distBaseDesde / 400 +
                          (distBaseDesde != 0) + aJuliano(fecDesde);
    return diasBaseAHasta - diasBaseADesde;
}
```

```
85
86  tFecha calcularEdad(const tFecha *fecDesde, const tFecha *fecHasta)
87  {
88      static const char dias[][13] = {
89                  { 0, 31, 31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30 },
90                  { 0, 31, 31, 29, 31, 30, 31, 30, 31, 31, 30, 31, 30 } };
91          ///          dic ene feb mar abr may jun jul ago set oct nov
92      tFecha  edad;
93
94      edad = *fecHasta;
95      if( (edad.di -= fecDesde->di) < 0)
96      {
97          edad.di += dias[esBisiesto(edad.an)][edad.me];
98          edad.me--;
99      }
100     if( (edad.me -= fecDesde->me) < 0)
101     {
102         edad.me += 12;
103         edad.an--;
104     }
105     edad.an -= fecDesde->an;
106     return edad;
107 }
108
109
```

```c
#ifndef FECHAS_H_
#define FECHAS_H_

#include <stdio.h>


/**
 *  tipo de dato para la fecha, los miembros de la 'struct' podrían estar en
 *      otro orden.
 **/
typedef struct
{
    int di,
        me,
        an;
} tFecha;

/**
 *  función booleana que permite el ingreso de tres enteros para el día, mes
 *      y año sin garantía de que correspondan a una fecha.
 *  contempla el caso en que se quiera mostrar un mensaje distinto del
 *      mensaje por defecto el que se mostrará si recibe NULL.
 *  contempla el caso en que no se quiera mostrar ningún mensaje cuando reciba
 *      una cadena vacía (o con blancos o tabulaciones), en este caso, de ser
 *      necesario, se mostrará el mensaje antes de invocarla.
 *  contempla el caso en que no se quiera ingresar nada, por ejemplo con
 *      cero en alguno de los enteros, con lo que devolverá 0 (cero)
 **/
int ingresarFechaDMA(tFecha *fec, const char *mensajeOpcional);

/**
 *  muestra una fecha en el formato dia/mes/año
 **/
void mostrarFechaDMA(const tFecha *fec);

/**
 *  Vigencia del Calendario Gregoriano
 *  Al jueves -juliano- 4 de octubre de 1582 le sucede el
 *      viernes -gregoriano- 15 de octubre de 1582.
 *  Diez días desaparecen debido a que ya se habían contado de más en el
 *      calendario juliano.
 *  Si hubiera habido calendario gregoriano:
 *      01/01/1582 -> VIERNES
 *      01/01/1581 -> JUEVES
 *      01/01/1580 -> MARTES
 *      ...
 *  tomaremos como mínimo válido el año 1 (uno), teniendo en cuenta que no
 *      existió el año 0 (cero), 'extrapolando' la vigencia del calendario
 *      gregoriano (su vigencia es -en España, Italia y Portugal, a saber, a `
 *      partir del 15 de octubre de 1582)
 *  por no ser adoptado en forma universal, su vigencia depende del país
 *  NOTA: se ponen topes mínimo y máximo, y nuestros algoritmos quedarán
 *      abiertos a modificaciones (que hasta ahora están más allá)
 *  OTRA: a partir de su vigencia, por una pequeña diferencia, cada 3623 años
 *      habrá un día más y para corregirlo, se dejará de contar un bisiesto
 *      cada 3000 años
 *  esto quedará como tema abierto a quién tenga intereses muy trascendentes, y
 *      deberá tener en cuenta determinaciones astronómicas más precisas
 *  si extrapolamos dentro de un rango razonable nos queda...
 **/
#define     AN_MIN          1
#define     AN_MAX          5000

/**
 *  para determinar si un año es bisiesto se lo puede hacer con una función, o
 *      mejor, como en este caso con un macroreemplazo
 **/
#define esBisiesto( X ) ( ( ( X ) % 4 == 0 && ( X ) % 100 != 0 ) || \
                          ( X ) % 400 == 0 )

/**
 *  determina si una fecha es válida
 **/
int esFechaValida(const tFecha *fec);

/**
 *  permite el ingreso de una fecha válida (valiéndose de las dos anteriores),
 *      contemplando el caso de que no se quiera ingresarla, además de la
 *      posibilidad de mostrar un mensaje distinto del mensaje por defecto
 **/
int ingresarFechaValidaDMA(tFecha *fec, const char *mensajeOpcional);

/**
 *  compara dos fechas devolviendo 0 si son iguales, algún valor negativo
```

```
 *      si la primera es menor que la segunda, algún valor positivo si la
 *      primera es mayor que la segunda
 **/
int compararFecha(const tFecha *fec1, const tFecha *fec2);

/**
 *  calcula y devuelve el día del año de esa fecha
 *  precondición: que sea una fecha válida
 **/
int aJuliano(const tFecha *fec);

/**
 *  la convención, más comunmente utilizada, adoptada a continuación se puede
 *      alterar a gusto del programador, a costa de apartarse del estándar
 **/
#define DOMINGO                 0
#define LUNES                   1
#define MARTES                  2
#define MIERCOLES               3
#define JUEVES                  4
#define VIERNES                 5
#define SABADO                  6

/**
 *  determina el número de día de la semana de una fecha
 *  se tiene en cuenta que:
 *  -   en un lapso de 400 años la cantidad de días es múltiplo de 7
 *  -   para cada uno de los años consecutivos de esos intervalos el primero de
 *      enero de cada año comienza en el mismo día
 *  -   el primero de enero de los años múltiplo de 400 es  SABADO  (6)
 *  tomamos como año base el año múltiplo de 400 inmediato anterior al año
 *      pero sólo es necesario calcular la distancia al año base en años
 *  por cada año se suma 1 al nro de día del año base, salvo que
 *      cada cuatro años es bisiesto y se suma uno más, pero cada 100 años
 *      no es bisiesto y se resta 1.
 *  si el año no fuera el año base calculado, se suma 1 porque es bisiesto
 **/
int nroDeDiaDeLaSemana(const tFecha *fec);

/**
 *  determina la cantidad de días desde fecDesde hasta fecHasta, que será
 *      negativa si fecDesde es mayor que fecHasta
 **/
long diasEntreFechas(const tFecha *fecDesde, const tFecha *fecHasta);

/**
 *  calcula la cantidad de días, meses y años entre dos fechas, y lo devuelve
 *      como una fecha
 **/
tFecha calcularEdad(const tFecha *fecDesde, const tFecha *fecHasta);



#endif

```