```c
/* ----------------------------x------o------x----------------------------
 *          fechas.h          funciones para trabajar con fechas (1.1)
 * ----------------------------x------o------x---------------------------- */

#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.  En ese caso, este sería el tipo de dado  tFecahaDMA
 *      las restantes combinaciones (más habituales), serían los tipos:
 *      tFecahaAMD, tFecahaMDA y se debería disponer de toda la colección
 *      de operaciones
 **/
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
 *  NOTA:
 *      Una fecha juliana se usa a veces para hacer referencia a un formato de
 *  fecha que es una combinación del año actual y el número de días desde el
 *  principio del año. Por ejemplo, 1 de enero de 2007 se representa como
 *  2007001 y 31 de diciembre de 2007 se representa como 2007365. Tenga en
 *  cuenta que este formato no se basa en el calendario juliano.
 **/
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);

/**
#include <stdlib.h>
#include <string.h>
 **/


/**
 *  tipo de dato 'extendido' para la fecha con mes y día en palabras
 ** /
typedef struct
{
    int     di,
            me,
            an;
    int     nroDia0Do1Lu;
    char    nombreDiaLargo[10];
    char    nombreDiaCorto[3];
    char    nombreMesLargo[11];
    char    nombreMesCorto[4];
} tFechaEx; **/
```

```
169
170 /**
171 int generarFechaEx(tFechaEx *fecEx, const tFecha *fec);
172  **/
173
174
175
176 #endif
177
178
179 /* ----------------------------x------o------x----------------------------
180  *          fechas.c         funciones para trabajar con fechas (1.1)
181  * ----------------------------x------o------x---------------------------- */
182
183 #include "fechas.h"
184
185 int ingresarFechaDMA(tFecha *fec, const char *mensajeOpcional)
186 {
187     if(mensajeOpcional)
188         printf("%s", mensajeOpcional);
189     else
190         printf("Fecha (dd/mm/aaaa - 0=No Ingresa): ");
191     fec->di = 0;
192     fec->me = 0;
193     fec->an = 0;
194     fflush(stdin);
195     scanf("%d/%d/%d", &fec->di, &fec->me, &fec->an);
196     return fec->di && fec->me && fec->an;
197 }
198
199 void mostrarFechaDMA(const tFecha *fec)
200 {
201     printf("%02d/%02d/%04d", fec->di, fec->me, fec->an);
202 }
203
204 int esFechaValida(const tFecha *fec)
205 {
206     static const char dias[][12] = {
207                         { 31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31 },
208                         { 31, 29, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31 } };
209     return  fec->me > 0 && fec->me <= 12 &&
210             fec->an >= AN_MIN && fec->an <= AN_MAX &&
211             fec->di > 0 && fec->di <= dias[esBisiesto(fec->an)][fec->me - 1];
212 }
213
214 int ingresarFechaValidaDMA(tFecha *fec, const char *mensajeOpcional)
215 {
216     do
217     {
218         if(!ingresarFechaDMA(fec, mensajeOpcional))
219             return 0;
220     } while(!esFechaValida(fec));
221     return 1;
222 }
223
224 int compararFecha(const tFecha *fec1, const tFecha *fec2)
225 {
226     int cmp = fec1->an - fec2->an;
227
228     if(cmp)
229         return cmp;
230     cmp = fec1->me - fec2->me;
231     if(cmp)
232         return cmp;
233     return fec1->di - fec2->di;
234 }
235
236 int aJuliano(const tFecha *fec)
237 {
238     int dias[][12] = {
239             { 0, 31, 59, 90, 120, 151, 181, 212, 243, 273, 304, 334 },
240             { 0, 31, 60, 91, 121, 152, 182, 213, 244, 274, 305, 335 } };
241
242     return dias[esBisiesto(fec->an)][fec->me - 1] + fec->di;
243 }
244
245 int nroDeDiaDeLaSemana(const tFecha *fec)
246 {
247     int distBase = fec->an % 400;
248
249     return ( SABADO - 1 + distBase + distBase / 4 - distBase / 100 +
250              (distBase != 0) + aJuliano(fec) ) % 7;
251 }
252
```

```
253  long diasEntreFechas(const tFecha *fecDesde, const tFecha *fecHasta)
254  {
255      int anBase = fecDesde->an <= fecHasta->an ? fecDesde->an : fecHasta->an,
256          distBaseDesde = fecDesde->an - anBase,
257          distBaseHasta = fecHasta->an - anBase;
258      long diasBaseAHasta = distBaseHasta * 365L + distBaseHasta / 4 -
259                            distBaseHasta / 100 + distBaseHasta / 400 +
260                            (distBaseHasta != 0) + aJuliano(fecHasta),
261           diasBaseADesde = distBaseDesde * 365L + distBaseDesde / 4 -
262                            distBaseDesde / 100 + distBaseDesde / 400 +
263                            (distBaseDesde != 0) + aJuliano(fecDesde);
264      return diasBaseAHasta - diasBaseADesde;
265  }
266
267  tFecha calcularEdad(const tFecha *fecDesde, const tFecha *fecHasta)
268  {
269      static const char dias[][13] = {
270                  { 0, 31, 31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30 },
271                  { 0, 31, 31, 29, 31, 30, 31, 30, 31, 31, 30, 31, 30 } };
272          ///          dic ene feb mar abr may jun jul ago set oct nov
273      tFecha  edad;
274
275      edad = *fecHasta;
276      if( (edad.di -= fecDesde->di) < 0)
277      {
278          edad.di += dias[esBisiesto(edad.an)][edad.me];
279          edad.me--;
280      }
281      if( (edad.me -= fecDesde->me) < 0)
282      {
283          edad.me += 12;
284          edad.an--;
285      }
286      edad.an -= fecDesde->an;
287      return edad;
288  }
289
290  /**
291  int generarFechaEx(tFechaEx *fecEx, const tFecha *fec)
292  {
293      static const char dias[][10] = {
294          { "domingo" }, { "lunes" },   { "martes" }, { "miercoles" },
295          { "jueves" },  { "viernes" }, { "sabado" } };
296      static const char meses[][12] = {
297          { "enero" },      { "febrero" }, { "marzo" },     { "abril" },
298          { "mayo" },       { "junio" },   { "julio" },     { "agosto" },
299          { "septiembre" }, { "octubre" }, { "noviembre" }, { "diciembre" } };
300      int aux;
301
302      memset(fecEx, 0, sizeof(tFechaEx));
303      fecEx->di = fec->di;
304      fecEx->me = fec->me;
305      fecEx->an = fec->an;
306      if(!esFechaValida(fec))
307          return 0;
308      aux = nroDeDiaDeLaSemana(fec);
309      fecEx->nroDia0Do1Lu = aux;
310      strcpy(fecEx->nombreDiaLargo, dias[aux]);
311      *fecEx->nombreDiaCorto = '\0';
312      strncat(fecEx->nombreDiaCorto, dias[aux], 2);
313
314      aux = fec->me - 1;
315      strcpy(fecEx->nombreMesLargo, meses[aux]);
316      *fecEx->nombreMesCorto = '\0';
317      strncat(fecEx->nombreMesLargo, meses[aux], 3);
318
319      return 1;
320  } **/
321
322
```