



# Informàtica

**ICB0 Desenvolupament  
d'aplicacions multiplataforma**

**ICC0 Desenvolupament  
d'aplicacions web**

M03 Programació

UF3 Fonaments de gestió de fitxers.

**Diari d'activitats**

Isidre Guixà

Curs 2024/25



**CodeBlocks – Debugger: No s'activa si el path absolut del projecte conté algun espai o caràcter no ascii!!!**

**Instruccions del llenguatge C per gestionar arxius (comunes per textuais i binaris) (14-01-25)**

- Per poder treballar amb el contingut d'un arxiu, cal definir dins el programa una variable (fitxer intern) que enllaçarem amb el fitxer extern a gestionar.  
  
En el llenguatge C per definir un fitxer intern `f`, escriurem:  
`FILE *f;`
- L'enllaç d'un fitxer intern amb un fitxer extern s'acostuma a efectuar amb una instrucció d'obertura del fitxer.  
  
En el llenguatge C, si volem enllaçar `f` amb un fitxer `X`, escriurem:  
`f = fopen(X, mode);`  
on:
  - `mode` és una combinació de caràcters en funció de si el fitxer `X` a gestionar és textual o binari i si es vol obrir per lectura, o per modificació o...
  - `X` és el nom del fitxer amb la seva ruta sencera, que cal escriure segons la nomenclatura del S.O. (en Windows, les carpetes es separen amb `\` i en Linux amb `/`). En els projectes NetBeans, si no s'indica cap ruta, suposa que el fitxer ha de residir a l'arrel de la carpeta del projecte.  
Revisar informació detallada sobre instrucció `fopen` [aquí](#).
- L'enllaç del fitxer es manté fins que s'elimina amb una instrucció de tancament del fitxer.  
  
En el llenguatge C, per tancar (i eliminar l'enllaç) un fitxer, escriurem:  
`fclose(f);`

**Exemple:** Projecte 250114\_1\_PrimerPrograma

- Intenta crear un fitxer (`prova.txt`) comprovant que no existeix un altre fitxer amb el mateix nom. No hi incorpora res, doncs encara no sabem com introduir contingut. El tanca i ha de quedar buit.
- Executeu el programa i podeu comprovar que crea el fitxer dins la carpeta del projecte.
- Si torneu a executar el programa us ha de dir que ja existeix.

**Instruccions d'escriptura del llenguatge C per gestionar fitxers textuais (16-01-25)**

A continuació es detallen les principals instruccions de lectura/escriptura de C per fitxers textuais. Cada instrucció té enllaç a pàgina web amb informació detallada..

- Instruccions d'escriptura
- [fputs](#): Permet escriure una cadena dins un fitxer. No hi escriu el `\0` final.
- [fprintf](#): Similar a la funció `printf`.
- [fputc](#): Permet inserir un caràcter.

**Exemple d'escriptura:** Projecte 250116\_1\_IncorporarTextEnFitxer

- Obre (crea si no existeix) fitxer "prova.txt" i hi afegeix text pel final, comprovant funcionament de funcions `fprintf`, `fputs` i `fputc`.
- Si s'executa aquest programa diverses vegades, es va incorporant el mateix text al final.

**Instruccions de lectura del llenguatge C per gestionar fitxers textuais (21-01-25)**

- Instruccions de lectura
- [fgets](#): Permet llegir com a cadena fins `\n` o un valor màxim de caràcters. Alerta amb el `\n...` que queda dins la cadena llegida, abans del `\0`.
- [fscanf](#): Similar a la funció `scanf`. Alerta si llegiu fins un `\n`, per què aquest caràcter NO és llegeix i l'haurem de treure manualment.
- [fgetc](#): Permet llegir un caràcter.
- Instruccions per saber si s'ha arribat a final de fitxer
- [feof](#): Funció que retorna cert quan s'ha arribat a final de fitxer.  
  
Cal tenir clar que retorna cert quan ja s'ha llegit la marca de final de fitxer, i si estem utilitzant una funció com



`fgetc` o `fscanf` que intenten llegir varis caràcters a la vegada, podria ser que en aquesta lectura s'hagués llegit la marca de final de fitxer i haguéssim llegit o no, abans, alguns caràcters. Això no pot passar usant la funció `fgetc`, ja que llegeix només un caràcter i quan llegeixi la marca de final, no haurà pogut llegir cap altre caràcter.

Per tant, en usar `fgetc` o `fscanf`, per saber si ha arribat a llegir alguna cosa o no, fixeuvos bé en la documentació i en què retornen...

### Algorisme de recorregut de fitxers en C (21-01-25)

situar-nos al principi del fitxer (cosa que ja passa si l'acabem d'obrir)  
llegir\_de\_fitxer (empenant la/les variables que correspongui) => Podeu usar alguna de les funcions de lectura de C mentre no final\_fitxer fer => Cal usar `!feof(f)`  
fer el tractament que calgui  
llegir\_de\_fitxer (empenant la/les variables que correspongui) => Tornem a usar la funció prèvia de lectura de C fimentre  
(abans d'acabar el programa... sempre s'ha de tancar els fitxers oberts!!!)

### Exemple de recorregut per llegir un fitxer de text: Projecte 250121\_1\_RecuperarTextDeFitxer

- Obre fitxer "prova.txt" generat en anterior projecte comprovant la seva existència.
- Efectua recorregut per anar llegint text de 20 en 20, usant `fgetc`.

### Definició de dada que es voldrà emmagatzemar en diferents tipus de fitxers al llarg del curs

```
#define MAX_DNI 9
#define MAX_NOM 40
struct tPersona {
    char dni[MAX_DNI+1];    // Obligatori de MAX_DNI caràcters
    char nom[MAX_NOM+1];    // No buit i amb MAX_NOM caràcters com a molt
    short edat;              // Estrictament positiu
};
```

Per a la seva gestió inicial, sembla lògic disposar de les següents funcions:

- `void llegirPersonaT (struct tPersona *p)`, que demani les dades d'una persona per teclat, tot empenant la variable `p`.
- `void mostrarPersonaP (struct tPersona p)`, que mostri les dades de la persona `p` per pantalla.

**Exercici per 23/01/25:** Crear projecte `Persona` que implementi definició i funcions indicades. Incorporar-hi un `main.c` que en comprovi el funcionament.

**Solució:** Tot i que no s'havia especificat, si tenim en compte que es preveu desenvolupar programes que permetin efectuar altes, baixes, consultes i modificacions de persones en diferents tipus de fitxers, podem pensar que en moltes ocasions caldrà efectuar lectura individual per teclat de qualsevol dels camps de la persona (`dni-nom-edat`) i, per tant, sembla lògic implementar aquestes funcions i usar-les dins de `llegirPersonaT`.

### CodeBlocs: Com usar arxius d'un projecte en altre projecte

- Des de l'arrel del projecte on es vol usar arxius d'un altre projecte, botó secundari del ratolí i opció *Add Files*, seleccionant els arxius `.c` i `.h` que correspongui, els quals seran accessibles des del *Project Manager*.
- La incorporació dels arxius `.c` és obligatòria, per poder generar l'executable del nou projecte.
- La incorporació dels arxius `.h` és optativa, però és aconsellable per poder consultar el contingut.
- Independentment de que els arxius `.h` d'altres projectes siguin visibles des del *Project Manager*, en els arxius `.c` del projecte vigent, els `#include` que facin referència a arxius `.h` d'altres projectes, hauran d'incorporar la ruta absoluta o relativa de la ubicació dels arxius.

### Important per la correcció de tasques:

En les tasques que caldrà lliurar, es demanarà projectes separats ubicats dins una mateixa carpeta. Cal que les definicions dels `#include` usin rutes relatives, doncs en cas d'usar rutes absolutes, caldria reconstruir-les en canviar la carpeta d'ubicació.

### Tasca 1: Programa per gestionar fitxer textual de persones, sense cap tipus d'ordenació

Es vol gestionar fitxer textual de persones, que contingui persones enregistrades seqüencialment, una darrera l'altra, en format com es veu a la dreta, que anomenarem FTL (fitxer text per línies)

```
dni
nom
edat
dni
nom
edat
...
```



- Desenvolupar projecte **PersonaFTL** que usi la definició de `Persona` i implementi les funcions:
  - `void enregistrarPersonaFTL (FILE *f, struct tPersona p)`, que enregistri les dades de la persona `p` en fitxer `f` que es suposa en format FTL indicat, obert per escriure-hi i amb punter posicionat en ubicació on escriure.
  - `void recuperarPersonaFTL (FILE *f, struct tPersona *p)`, que recuperi les dades d'una persona des de fitxer `f` que es suposa en format FTL indicat, obert per llegir-hi i amb punter posicionat en ubicació on llegir.

**Aquestes funcions les usarem en qualsevol programa que gestioni persones en fitxer de text amb format FTL.**

- Desenvolupar projecte **GestioPersonaFTL**, amb funcionalitats:
  - En iniciar, comprovi existència de fitxer `personaFTL.txt` i el creï en cas de no existir.
  - Menú amb dues opcions:
    - Afegir persona, que demani una persona per teclat i l'afegeixi al final del fitxer.
    - Mostrar persones, que recorri tot el fitxer i mostri totes les persones, tenint en compte que:
      - ✓ Ha d'informar d'inexistència de persones si està buit.
      - ✓ En cas que hi hagi persones, les mostrarà en format:

```
DNI      NOM      EDAT
*****
.....
.....
```

**Lliurament:** Carpeta **T1\_CognomNom** comprimida que contingui:

Projecte `Persona` sense carpetes `obj/bin`

Projecte `PersonaFTL` sense carpetes `obj/bin`

Projecte `GestioPersonaFTL` amb **CLEAN** passat, de manera que no contingui carpetes `obj/bin`

## Solució a la tasca 1

La solució a la tasca 1 es pot plantejar de dues maneres:

- Plantejament 1:
  - En opció *Afegir persona* obrir l'arxiu en mode "a" i tancar-lo en finalitzar
  - En opció *Mostrar persones* obrir l'arxiu en mode "r" i tancar-lo en finalitzar
- Plantejament 2:
 

Tenir obert el fitxer des d'inici de programa i tancar-lo al final. En aquest cas, ens cal obrir el fitxer en mode "a+", de manera que quan afegim, s'afegeixi pel final i tenim la possibilitat d'ubicar el cursor que recorre el fitxer en qualsevol lloc usant les funcions de posicionament [fseek](#), [ftell](#), [fsetpos](#), [fgetpos](#) i [rewind](#).

La solució que s'adjunta implementa el plantejament 2.

**Atenció!** La funció `recuperarPersonaFTL` ha d'estar implementada de manera que recuperi una persona sencera i no llegeixi cap caràcter de la següent persona.

Com que en `enregistrarPersonaFTL` finalitzem l'enregistrament bolcant el `dni` seguit d'un `\n`, en `recuperarPersona` hem d'estar segurs de recuperar el `dni` amb el `\n` i això ja es fa amb `fscanf(f, "%hd", &p->edat)`, sense afegir un `\n` després de `%hd`. En canvi, en la recuperació de `dni` i de `nom`, que són cadenes, amb `fscanf`, hem hagut d'afegir `\n` després de `%s` o `%[^\\n]` per què, del contrari, `\n` continua pendent de llegir.

## Tasca 2: Programa per gestionar fitxer textual (format FTL) de persones, no ordenat, amb DNI identificador

Es vol gestionar un fitxer textual de persones, seguint el format especificat en tasca 1, en el que el `dni` sigui identificador.

**Per enregistrar i recuperar persones del fitxer, usarem les funcions ja desenvolupades a `personaFTL`, sense cap modificació.**

Desenvolupar projecte **GestioPersonaFTLambDniIdentificador**, amb funcionalitats:

- En iniciar, comprovi existència de fitxer `personaFTL.txt` i el creï en cas de no existir.
- Menú amb dues opcions:
  - Afegir persona, que faciliti la funcionalitat d'enregistrar una persona que l'usuari haurà d'introduir per teclat, comprovant que el no existeix ja una persona amb el DNI indicat.



- Mostrar persones, amb mateixa funcionalitat que projecte GestioPersonaFTL.

**Lliurament:** Carpeta **T2\_CognomNom** comprimida que contingui:

Projecte Persona sense carpetes obj/bin

Projecte PersonaTFL sense carpetes obj/bin

Projecte GestioPersonaTFLambdaDniIdentificador amb *CLEAN* passat, de manera que no contingui carpetes obj/bin

### **Tasca 3: Incorporem noves funcionalitats en programa de tasca 2**

Ampliar projecte **GestioPersonaFTLlambdaDniIdentificador**, amb funcionalitats:

- Cercar persona, a partir de dni a introduir per usuari.
- Modificar persona, que permeti modificar les dades (menys dni) d'una persona a partir de dni a introduir per usuari, modificant el registre en el mateix fitxer.  
**Atenció!** Una vegada detectem la persona a modificar, només podríem sobre escriure-la (situant-nos prèviament a la posició on començava via `fseek` o `fsetpos`) si les noves dades (nom i/o edat) tinguessin mateixa llargada que les dades antigues i això molt possiblement no sigui així.  
Algorisme de modificació amb còpia en fitxer temporal:
  - Situar-se a inici de fitxer
  - Crear un fitxer temporal
  - Començar a cercar la persona dins fitxer, fins trobar persona a modificar i, per cada persona consultada, copiar-la en el fitxer temporal.
  - Si es troba la persona, demanar les noves dades (nom i/o edat) i enregistrar-la en fitxer temporal; acabar de copiar resta de persones en fitxer temporal, tancar fitxers, eliminar fitxer original, reanomenar fitxer temporal com a fitxer original i obrir-lo com s'espera tenir-lo obert en la resta d'opcions.
  - Si no es troba la persona, tancar fitxer temporal i eliminar-lo, com si no hagués existit mai.
- Eliminar persona, que permeti eliminar una persona a partir de dni a introduir per usuari.  
Algorisme d'eliminació amb còpia en fitxer temporal:
  - Situar-se a inici de fitxer
  - Crear un fitxer temporal
  - Començar a cercar la persona dins fitxer, fins trobar persona a eliminar i, per cada persona consultada, copiar-la en el fitxer temporal.
  - Si es troba la persona, acabar de copiar resta de persones en fitxer temporal, tancar fitxers, eliminar fitxer original, reanomenar fitxer temporal com a fitxer original i obrir-lo com s'espera tenir-lo obert en la resta d'opcions.
  - Si no es troba la persona, tancar fitxer temporal i eliminar-lo, com si no hagués existit mai.

Noves funcions de fitxers que cal usar: [remove](#) i [rename](#).

**Lliurament:** Carpeta **T3\_CognomNom** comprimida que contingui:

Projecte Persona

Projecte PersonaTFL

Projecte GestioPersonaTFLambdaDniIdentificador amb *CLEAN* passat, de manera que no contingui carpetes obj/bin

### **Observació a la solució facilitada:**

- La funcionalitat de permetre modificar les dades d'una persona –menys dni– per teclat, és una funcionalitat que té a veure amb la gestió de persones, independentment de si es guarden en línies d'un fitxer textual o en un fitxer binari o les tenim només en una taula en memòria o... En conseqüència, és lògic tenir aquesta funcionalitat dins `persona.h/c`, de manera que la puguem usar en qualsevol programa on convingui.
- Les opcions que es demanen (cercar, modificar, eliminar) impliquen crear noves funcions que facin aquesta feina, **usant les funcions `recuperarPersonaFTL` i `enregistrarPersonaFTL` implementades dins `personaFTL.h/c`, sense fer cap modificació a la versió creada a la tasca1.**

### **Tasca 4: Programa per gestionar fitxer textual de persones (format FTL), ordenat per DNI identificador**

El projecte ha de tenir nom **GestioPersonaFTLOrdenatPerDniIdentificador**.

El programa ha de tenir les mateixes opcions que el programa de la tasca 3, però mantenint el fitxer ordenat i cal fer-ho de manera eficient, és a dir, en cercar una persona, caldrà aturar-nos en el moment en que es trobi un DNI que superi el DNI de la persona cercada.



L'opció `afegirPersona` haurà d'usar algorisme d'inserció amb còpia en fitxer temporal.

**Lliurament:** Carpeta **T4\_CognomNom** comprimida que contingui:

Projecte `Persona`

Projecte `PersonaTFL`

Projecte `GestioPersonaTFLOrdenatPerDniIdentificador` amb *CLEAN* passat, de manera que no contingui carpetes `obj/bin`

### Instruccions del llenguatge C per gestionar fitxers binaris (25/02/2025)

- Instrucció d'escriptura: [fwrite](#).
- Instrucció de lectura: [fread](#).

En ambdós casos, retornen la quantitat de dades gestionades (escrites o llegides).

#### Exemples:

- Projecte `250225_1`, que crea un fitxer binari i enregistra uns valors (int, float, cadena, taula de double)
- Projecte `250225_2`, que recupera de fitxer binari anterior els valors que conté. **Evidentment**, s'ha de conèixer el format de les dades que conté per poder-les recuperar.

Si via S.O. observeu les propietats del fitxer generat, veureu que ocupa 37 bytes (4 per int + 4 per float + 5 per la cadena de caràcters + 8 per cadascun dels 3 double guardats).

### ALERTA en fitxers (textuals o binaris) oberts en mode + (update)

Mireu projecte `250304_1_ExempleAlerta` i comproveu funcionament.

Sembla que no funciona. Per què?

Quan varem presentar la funció `fopen`, segurament no varem posar atenció en una informació fonamental que està al final:

*For files open for update (those which include a "+" sign), on which both input and output operations are allowed, the stream shall be flushed ([fflush](#)) or repositioned ([fseek](#), [fsetpos](#), [rewind](#)) before a reading operation that follows a writing operation. The stream shall be repositioned ([fseek](#), [fsetpos](#), [rewind](#)) before a writing operation that follows a reading operation (whenever that operation did not reach the end-of-file).*

En els programes que hem fet fins ara, quan hem canviat de lectura a escriptura o d'escriptura a lectura ha donat la "casualitat" de que sempre hem fet un `fseek` sense adonar-nos que complíem la restricció anterior.

En aquest programa `ExempleAlerta`, no ho hem fet i per això el funcionament no és correcte.

El projecte `250304_2_ExempleAlerta` és la versió correcta, on abans de fer un `fread` o `fwrite`, com que abans hem fet un `fwrite` o `fread` respectivament, executem un reposicionament.

### Tasca 5: Programa per gestionar fitxer binari de persones (format FB), ordenat per DNI identificador

Crear projecte `PersonaFB` equivalent a `PersonaFTL` però pensat per contenir les funcions necessàries per enregistrar i recuperar persones de fitxer binari de persones. Per tant, aquest projecte haurà de contenir les funcions:

- `void enregistrarPersonaFB (FILE *f, struct tPersona p)`, que enregistri les dades de la persona `p` en fitxer `f` que es suposa en format binari, obert per escriure-hi i amb punter posicionat en ubicació on escriure.
- `void recuperarPersonaFB (FILE *f, struct tPersona *p)`, que recuperi les dades d'una persona des de fitxer `f` que es suposa en format binari, obert per llegir-hi i amb punter posicionat en ubicació on llegir.

**Aquestes funcions les usarem en qualsevol programa que gestioni persones en fitxer binari de persones.**

Crear projecte de nom `GestioPersonaFBOrdenatPerDniIdentificador` amb programa que faciliti les mateixes opcions que el programa de la tasca 4, però mantenint el fitxer ordenat per DNI i cal fer-ho de manera eficient, és a dir, en cercar una persona, caldrà aturar-nos en el moment en que es trobi un DNI que superi el DNI de la persona cercada.

Cal refer els algorismes `afegirPersona`, `modificarPersona` i `eliminarPersona` sense usar fitxer temporal, és a dir, fent "el que calgui" en el mateix fitxer. Per tant, tingueu present que:

- Abans de `fread` seguint a un `fwrite`, cal fer `fflush` o reposicionar-se amb `fseek-fsetpos-rewind`.
- Abans de `fwrite` seguint a un `fread`, cal reposicionar-se amb `fseek-fsetpos-rewind`.





#### Algorisme per afegirPersona sense usar fitxer temporal:

- Cerquem persona amb DNI indicat, fins que la trobem o ens passem o arribem a final de fitxer.
- Si la trobem, informem i finalitza.
- Si ens hem passat, sense arribar al final, tenim una persona llegida en variable `aux1`
  - Demanem resta de dades.
  - Movem el punter on estem ubicats una persona abans, per situar-nos en el lloc on cal incorporar la nova persona i hi gravem la nova persona.
  - Des del punt on estem ara i fins al final, repetim el procés:
    - ✓ Llegim següent persona en `aux2`
    - ✓ Tirem enrere una persona
    - ✓ Gravem persona `aux1` i copiem `aux2` en `aux1`.
  - Al final, quan ja no hi ha més persones, la darrera persona que teníem a `aux1` s'ha d'afegir al fitxer
- Si havíem arribat al final:
  - Demanem resta de dades
  - Afegim persona

#### Algorisme per modificarPersona sense usar fitxer temporal:

- Cerquem persona amb DNI indicat, fins que la trobem o ens passem o arribem al final del fitxer.
- Si ens passem o arribem al final del fitxer, informem i finalitza.
- En el moment en què la trobem, demanem resta de dades i:
  - Tirem enrere una persona
  - Escrivim dades modificades

#### Algorisme per eliminarPersona sense usar fitxer temporal:

- Cerquem persona amb DNI indicat, fins que la trobem o ens passem o arribem al final del fitxer.
- Si ens passem o arribem al final del fitxer, informem i finalitza.
- En el moment en què la trobem, repetim el següent procés fins el final:
  - Llegim següent persona en variable `aux`
  - Tirem dues persones enrere
  - Gravem persona `aux`
  - Tirem una persona endavant
- Al final, hem de truncar el fitxer amb una persona menys i això... com es fa?

Hi ha llenguatges que faciliten aquesta funcionalitat (per exemple, la classe `RandomAccessFile` de Java té el mètode `setLength()` que permet definir la llargada que li vols donar al fitxer i serveix per truncar. Algunes versions de C també ho tenen, però no l'estàndard que estem usant. Ens ho haurem de programar...

Us facilito projecte `UtilsFitxers` que conté una funció per `truncarFitxers` a una determinada longitud, provada en Windows i suposo que també funcionaria en Linux. Fixeu-vos que es guarda una còpia de seguretat en cas que el truncament generi algun error (per no perdre el fitxer!!!).

Com saber a quants bytes truncar? Dues opcions:

- Haver anat comptant quantes persones "q" tenim i llavors calcular els bytes via `q*sizeof(...)`
- Usar funció `ftell(f)` que dona el byte en el què estem posicionats i que és perfecte si estem en el lloc on cal truncar.

**Lliurament:** Carpeta **T5\_CognomNom** comprimida que contingui:

Projecte `Persona`  
Projecte `PersonaTB`  
Projecte `UtilsFitxers`  
Projecte `GestioPersonaTBOrdenatPerDniIdentificador`  
(tots els projectes amb **CLEAN** passat, de manera que no continguin carpetes `obj/bin`)

#### Observacions a la solució facilitada:

El fitxer s'ha d'obrir en mode `rb+`. Si l'obrim amb `wb+` es crearia cada vegada i si l'obrim amb `ab+` només afegiria pel final i no podríem sobreescriure. Però per obrir-lo amb `rb+` ha d'existir. Per això, en començar el programa, cal comprovar la seva existència i si no existeix, crear-lo buit, tancar-lo i obrir-lo ja en mode `rb+`.



## Fitxers Relatius

Els fitxers relatius són aquells fitxers que permeten accés directe per posició dels seus registres. Fins ara, en tots els programes que hem desenvolupat, hem fet recorregut seqüencial, començant pel principi.

Els fitxers relatius acostumen a facilitar una instrucció del tipus: `llegir(f, numReg)`.

El SGF de C/C++ no facilita cap instrucció similar però en fitxers on tots els registres tinguin la mateixa llargada (longitud fixa), es pot simular usant `fseek` seguit de `fread`.

## Tasca 6 – Cerca dicotòmica o binària en fitxer relatiu ordenat

El fitxer binari de persones gestionat a la tasca 5 està ordenat per DNI.

A1r curs vàreu estudiar la **cerca dicotòmica** o **cerca binària** en taules ordenades... Recordeu l'algorisme:

```
funció cerca_dicotòmica (t: taula[MAX] de TD, nt: natural, v: tc) retorna enter és
/*
  Arguments:
    t : Taula ordenada ASC d'un tipus TD que pot tenir molts camps i que està ordenada pel camp tc
        pel que cal efectuar la cerca d'valor v
    v : Valor que es vol cercar
    nt : Quantitat real d'elements existents a la taula; posicions 0 a nt-1
  Retorna:
    pos>=0: Primera posició de la taula en què es troba un element amb el valor cercat
    pos<0 : No s'ha trobat però -pos-1 conté la posició on hauria d'estar
*/
var  inf: natural; /* Conté la posició menor del tros de taula que s'està tractant */
     sup: natural; /* Conté la posició major del tros de taula que s'està tractant */
     mig: natural; /* Conté la posició per on trenquem la taula en dos trossos */
     possible: lògic; /* Per saber si hem de continuar -pot estar o no en algun tros- */
fivar
si nt==0 llavors retorna -1; fisi
inf=0;
sup=nt-1;
si v<t[inf].tc llavors retorna -1; fisi
si v>t[sup].tc llavors retorna -sup-2; fisi
possible = cert;
mentre possible i inf < sup fer
  mig = (inf + sup) div 2;
  si v ≤ t[mig].tc
    llavors
      /* Cas en què el valor pot estar en el primer tros. Continuem amb el primer tros */
      sup = mig;
  sinó si v ≥ t[mig+1].tc
    llavors
      /* Cas en què el valor pot estar en el segon tros. Seguim amb el segon tros */
      inf = mig+1;
  sinó
    /* El valor no pot estar a la taula perquè no pot estar en cap tros */
    possible = fals;
  fisi
fisi
fimentre
/* Quan se surt del mentre cal decidir el motiu pel qual s'ha sortit. Comproveu que
en cas d'existir, se surt del mentre quan s'arriba a una taula d'un únic element i,
per tant, per comprovar si s'ha trobat cal preguntar sobre el valor que conté tal
element; pot donar-se el cas que s'arribi a una taula d'un sol element i aquest no
contingui el valor cercat; per tant, és obligatori preguntar sobre el valor de la
posició inf en el moment en què se surt del mentre i no s'hi val a usar "possible"*/
si t[inf].tc != v
  llavors
    /* No hi ha cap element i hauria d'estar a mig+1 */
    retorna -(mig+2); /* per què en calcular -pos-1 surti mig+1 */
  sinó
    /* La variable inf té la posició del primer element que conté el valor cercat */
    retorna inf;
fisi
fifunció
```

Aquest algorisme es pot usar en taules per què aquestes permeten accés per posició i es pot anar a una posició en concret sense passar per les anteriors.

Doncs en arxius amb accés directe per posició també es pot aplicar aquest algorisme per efectuar la cerca d'un registre pel valor del camp pel què el fitxer està ordenat.





I en els fitxers de C/C++ de longitud de registre fixa també es pot aplicar.

Es demana fer una nova versió del programa desenvolupat a la tasca 5 (fitxer binari de persones ordenat per DNI) en el que les cerques s'efectuïn via cerca dicotòmica. Concretament cal:

1. Incorporar dins PersonaFB la funció:  
`int cercaBinaria(FILE *f, char *dni)`  
que efectuï la cerca binària del dni indicat en el fitxer f, que es suposa binari de persones, retornant els mateixos valors que la funció `cerca_dicotòmica` facilitada:  
`pos>=0`: Posició de primer registre (numerant-los a partir de 0) on es troba element amb valor cercat  
`pos<0` : No s'ha trobat però `-pos-1` conté la posició on hauria d'estar
2. Refer `cercarPersona(FILE *f)` de manera que utilitzi la funció `cercaBinaria` per trobar la persona.
3. Refer `afegirPersona (FILE *F)` de manera que utilitzi la funció `cercaBinaria` per cercar la persona i també per saber en quina posició s'haurà d'inserir en cas de no trobar-la.
4. Refer `eliminarPersona (FILE *F)` de manera que utilitzi la funció `cercaBinaria` per cercar la persona i eliminar-la.

**Lliurament:** Carpeta **T6\_CognomNom** comprimida que contingui:

Projecte Persona  
Projecte PersonaTB  
Projecte UtilsFitxers  
Projecte GestioPersonaTBOrdenatPerDniIdentificador  
(tots els projectes amb *CLEAN* passat, de manera que no continguin carpetes obj/bin)



## Pràctica final

Segons programació de la UF:

- Obligatòria
- Puntua un 25% de la nota de la UF
- Nota mínima 5

Es vol crear una aplicació per gestionar els resultats d'un torneig d'escacs (participants i partides).

- Respecte els participants:

Es vol enregistrar el número de federat (enter estrictament positiu), nom, sexe i data de naixement i han de residir en un fitxer binari de nom `participa.dat` ordenat pel número de federat, que evidentment és identificador.

- Respecte les partides:

Es vol enregistrar la data, hora d'inici, els números de federat dels contrincants, durada en minuts i un valor 1 si ha guanyat el primer contrincant, 2 si ha guanyat el segon contrincant o 0 si ha estat taules. Les partides han de residir en un fitxer binari de nom `partida.dat` ordenat per data de la partida, que no és identificador. La combinació (data, hora, contrincant1, contrincant2) és identificador.

- Dissenyar projecte de nom **PF\_Cognom1Nom** amb la següent estructura de programa:

Menú principal	Opcions	Observacions
1. Gestió de participants	1. Alta 2. Baixa 3. Modificació 4. Consulta 5. Llistat 0. Tornar	- No és possible efectuar una baixa si consta en alguna partida. - Cal poder modificar totes les dades menys el número de federat. - Per efectuar una baixa o una modificació o una consulta, es demanarà el número de federat per localitzar el participant. - El llistat ha de mostrar els participants ordenats per número de federat, en format: <div style="text-align: right;"> <u>Número</u>   <u>Nom</u>                      <u>Sexe</u>      <u>Data naixement</u>              .....              .....              .....           </div>
2. Gestió de partides	1. Alta 2. Baixa 3. Modificació 4. Consulta 5. Llistat per data 0. Tornar	- Els participants d'una partida han d'existir en el sistema. - En modificació, només es pot modificar la durada i el resultat. - Per efectuar una baixa o una modificació o una consulta, es demanarà data, hora i números de federat dels contrincants. - En el llistat per data, el sistema demanarà la data i mostrarà les partides efectuades en dita data, en format: <div style="text-align: right;"> <u>Data</u>   <u>Hora</u>   <u>NF</u> <u>Nom</u>                      <u>NF</u> <u>Nom</u>                      <u>Temps</u> <u>Guanyador</u>              ....              ....              ....              on columna <u>NF</u> és el número de federat de cada contrincant i columna <u>Guanyador</u> mostrarà 1-2-0 com consta en el fitxer.           </div>
3. Configuració	1. Buidar partides 2. Buidar tot	- L'opció <i>Buidar partides</i> ha d'eliminar totes les partides - L'opció <i>Buidar tot</i> ha d'eliminar partides i participants

0. Sortir

- Respecte les dates, com que es demana en algun lloc ordenació per data, el més fàcil és guardar-les en una cadena amb format `yyyy-mm-dd` i així, l'ordre de lexicogràfic de les cadenes serveix per l'ordenació de dates. Eps! Les dates han de ser sempre correctes!!! Aconsellable disposar d'una utilitat `llegirData` que efectui la lectura i controli la seva correctesa.
- Respecte les hores, guardeu-les amb format `hh:mm` i controleu que siguin correctes. Aconsellable disposar d'una utilitat `llegirHora` que efectui la lectura i controli la seva correctesa.
- En totes les eliminacions, cal demanar confirmació *Segur que vol eliminar?*
- Estructureu la implementació, de manera que no estigui tot en un únic fitxer font. Així, seria adequat tenir:
  - `participant.h` i `partida.h` amb la definició de les corresponents estructures de dades i les funcions necessàries per mostrar dades de participant, llegir dades de participant, mostrar dades de partida, llegir dades de partida,...
  - `participant.c` i `partida.c` amb la implementació de les funcions definides en els corresponents `.h`.



- `programa.c` amb el programa principal, que defineixi els menús i cada opció de menú invoqui una funció que faci la feina, implementada en el mateix `programa.c`.
- En posar en marxa el programa, cal comprovar l'existència dels 2 fitxers i, en cas d'inexistència informar i demanar autorització de creació i en cas que l'usuari no ho permeti, finalitzar l'aplicació amb missatge "*No és possible usar l'aplicació per manca dels fitxers de dades*".

**Lliurament:** El projecte PF\_Cognom1Nom comprimit  
amb CLEAN passat, de manera que no continguin carpetes `obj/bin`