Generalitat de Catalunya
Departament d'Educació
Institut Milà i Fontanals. Igualada

Dossier

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 2023/24

Milà i Fontanals IGUALADA
Unió Europea
Fons Social Europeu
ISO 9001 BUREAU VERITAS Certification

CodeBlocks – Debugger: No s'activa si el path absolut on hi ha el projecte conté algun caràcter no ascii!!!

Presentació "Fitxers – Resum Teoria" (09/01/24 - 11/01/24 – 16/01/24) – Penjada al Classroom

- **Respecte l'apartat 6 de la presentació: Tipus d'accés**

  Hi ha dues maneres d'accedir a la informació:

  a) Via accés seqüencial (S). Per accedir a un registre cal passar obligatòriament pels anteriors, des de l'inici.
  b) Via accés directe (D). Per a accedir a un registre no cal passar pels anteriors.

  Fixeu-vos que en aquestes definicions apareix l'expressió "pels anteriors", que implica l'existència d'algun tipus de classificació entre els registres. Tenim dues maneres de classificar els registres:

  a) Segons la posició (P) física que ocupen dins el fitxer.
  b) Segons el valor (V) d'un o diversos camps per als quals ha d'existir algun tipus d'ordenació.

  Si combinem els dos tipus d'accés amb els dos tipus de classificació obtenim quatre possibilitats: SP – DP – SV – DV

Exemple:

Suposem el fitxer representat a la taula següent. Observeu que es tracta d'un fitxer que té dos camps: `nom` i `edat`. Suposem que en aquests moments hi ha quatre registres. Les referències R1, R2, R3 i R4 indiquen la posició física que ocupen (primer, segon, tercer i quart).

	Nom	Edat
R1	Pep	17
R2	Lluís	21
R3	Joan	25
R4	Maria	40

- Suposem que es vol accedir al registre que està a la quarta posició. Cal, doncs, efectuar un accés seqüencial o directe per posició.

– En cas d'un accés seqüencial per posició, per a accedir a un registre cal haver passat pels registres anteriors. Per tant, per a arribar al registre R4 cal haver passat pels registres R1, R2 i R3.
– En cas d'un accés directe per posició, podem accedir directament a un registre sense haver de passar pels registres que ocupen una posició anterior. Per tant, accediríem directament al registre R4.

- Suposem que es vol cercar un registre que tingui per nom "Maria".

– Si el fitxer no disposa de cap classificació per al camp `nom`, no hi haurà altre remei que utilitzar l'accés seqüencial per posició i, començant per l'inici, anar recorrent els registres i comprovant que cadascun té el valor "Maria" per al camp `nom`. Per tant, passarem pels registres R1, R2, R3 i R4 i en aquest últim trobarem el valor cercat.
– Si el fitxer disposa d'accés seqüencial per al valor del camp `nom`, podrem utilitzar-lo, de manera que, començant pel primer registre segons el camp `nom` i seguint pels registres següents, anirem comprovant si trobem un registre amb el valor cercat. Així passarem pels registres R3 (Joan), R2 (Lluís) i R4 (Maria) on, en trobar-se el valor cercat, s'acaba el recorregut.
– Si el fitxer disposa d'accés directe per al valor del camp `nom`, podrem utilitzar-lo, de manera que podem sol·licitar directament el registre que tingui el valor "Maria" per al camp nom. El SGF ens comunicarà que existeix aquest registre i ens el subministrarà.

- Suposem que es vol cercar un registre que tingui per nom "Lídia".

– Si el fitxer no disposa de cap classificació per al camp `nom`, no hi haurà altre remei que utilitzar l'accés seqüencial per posició i, començant per l'inici, anar recorrent els registres i comprovant que cadascun té el valor "Lídia" per al camp `nom`. Per tant, passarem pels registres R1, R2, R3 i R4 sense trobar el valor cercat.

– Si el fitxer disposa d'accés seqüencial per al valor del camp `nom`, podrem utilitzar-lo, de manera que, començant pel primer registre segons el camp `nom` i seguint pels registres següents, anirem comprovant si trobem un registre amb el valor cercat. Així passarem pels registres R3 (Joan) i R2 (Lluís). No cal continuar, ja que sense haver trobat el valor "Lídia" hem arribat a un registre que té un valor superior ("Lluís").

– Si el fitxer disposa d'accés directe per al valor del camp `nom`, podrem utilitzar-lo, de manera que podem demanar directament el registre que tingui el valor "Lídia" per al camp `nom`. El SGF ens comunicarà que no existeix cap registre amb aquest valor.

Fixeu-vos que la utilització d'accés segons el valor implica l'existència d'una via d'accés, la qual pot ser constituïda per un o diversos camps. En l'exemple anterior, hem estat treballant amb la via d'accés formada pel camp nom. Podríem parlar d'una altra via d'accés constituïda pel camp edat. Però també podem tenir vies d'accés constituïdes per diferents camps, com en el cas de nom + edat o edat + nom. En aquests casos, s'accedeix als registres classificats pel primer camp que forma la via d'accés, i en cas d'igualtat de valor, es té en compte el(s) camp(s) següent(s).

- **Respecte l'apartat 7 de la presentació: Tipificació de fitxers**

- Tenim quatre maneres d'accedir a la informació emmagatzemada en fitxers. Un SGF pot oferir qualsevol combinació de les quatre possibilitats. Per tant, quants tipus diferents de SGF podeu trobar?

  La resposta és 15. Aquesta solució s'obté sumant $C_{4,1} + C_{4,2} + C_{4,3} + C_{4,4}$.

  $C_{4,1}$: Combinacions de 4 possibilitats agafant-ne 1 = 4
  $C_{4,2}$: Combinacions de 4 possibilitats agafant-ne 2 = 6
  $C_{4,3}$: Combinacions de 4 possibilitats agafant-ne 3 = 4
  $C_{4,4}$: Combinacions de 4 possibilitats agafant-ne 4 = 1

  És a dir, tenim potencialment 15 tipus de SGF. Comercialment, però, n'hi ha tres tipus principals, que tot i ser SGF s'anomenen simplement fitxers:

a) **Fitxers seqüencials**, que permeten l'accés seqüencial segons la posició (SP).

b) **Fitxers relatius**, que permeten l'accés seqüencial i directe segons la posició (SP + DP).

c) **Fitxers seqüencials indexats**, que permeten l'accés seqüencial i directe segons el valor amb la possibilitat de diverses vies d'accés (SV + DV)

Altres dos tipus de SGF que cal destacar són: els fitxers calculats i els fitxers de text.

d) **Fitxers calculats**, també anomenats aleatoris o hashing, els quals no estan comercialitzats i que s'implementen amb la utilització de fitxers relatius. Permeten l'accés seqüencial i directe segons la posició (ja que es basen en fitxers relatius) i l'accés directe segons el valor d'una única via d'accés. En moltes ocasions se'ls considera com una especificitat de fitxers relatius.

e) **Fitxers de text**, que són una espècie de fitxers seqüencials que fan servir caràcters especials com fi_de_línia, fi_de_pàgina... Aquests fitxers s'utilitzen per a enregistrar informació sense seguir cap tipus d'estructura mitjançant registres i camps.

Operacions sobre el sistema d'arxius vs operacions sobre el contingut dels arxius (16/01/24)

Cal distingir dos tipus d'operacions:

- Operacions per obtenir informació sobre els fitxers i carpetes existents en el sistema d'arxius del S.O:
  - Quines carpetes hi ha en una determinada ruta
  - Quins arxius hi ha en una determinada ruta
  - Discriminar si `xxx` és carpeta o arxiu
  - Recuperar data de creació, data de darrera modificació, grandària,...
  - ...

  Aquestes operacions són facilitades per tots els llenguatges de programació.
- Operacions per gestionar el contingut dels fitxers, amb les anomenades operacions CRUD (C de `create`, R de `read`, U de `update` i D de `delete`).
  Aquestes operacions són facilitades pel SGF que correspongui.
  Molts llenguatges incorporen instruccions bàsiques de SGF per gestionar fitxers textuals i binaris.

Instruccions del llenguatge C per gestionar arxius (comunes per textuals i binaris) (16-01-24)

- 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 `240116_1_PrimerPrograma`

- Intenta crear un fitxer (`prova.txt`) per enregistrar-li un text. Usa la funció `fprintf`. Les funcions de lectura/escriptura les veurem amb més detall en apartat següent.
- Executeu el programa i podeu comprovar que crea el fitxer dins la carpeta del projecte.
- Amb un editor de textos, modifiqueu el fitxer.
- Torneu a executar el programa i podeu comprovar que el fitxer existent ha quedat modificat.
- Des de l'explorador de fitxers de Windows, aneu a les propietats del fitxer i feu-lo de "només lectura".
- Torneu a executar el programa i podeu comprovar que no es pot crear el fitxer (està protegit).

Exemple: Projecte `240118_1_FitxerTextAmbEnters`.

- Intenta crear un fitxer amb els 100 primers enters separats per un espai en blanc. Usa `fprintf`.

Instruccions del llenguatge C per gestionar fitxers textuals

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

- Instruccions d'escriptura (ja hem usat `fprintf` en projecte anterior)

- `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.

- 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 `fgets` 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 `fgets` o `fscanf`, per saber si ha arribat a llegir alguna cosa o no, fixeu-vos bé en la documentació i en què retornen...

Algorisme de recorregut de fitxers en C

```
obrir_fitxer
llegir_de_fitxer (emplenant la/les variables que correspongui) => Podeu usar alguna de les funcions de lectura de C
mentre no final_final_de_fitxer fer  => Cal usar !feof(f)
   fer el tractament que calgui
   llegir_de_fitxer (emplenant la/les variables que correspongui) => Tornem a usar la funció prèvia de lectura de C
fimentre
tancar_fitxer
```

Exercici per dimarts, 23-gener:

- Suposem que ens donen un fitxer textual que conté valors enters separats per espai en blanc, similar al que hem creat en el projecte 240118_1. Podeu usar el fitxer d'aquell projecte o qualsevol fitxer creat amb un editor de textos (*notepad*, *notepad*++,...).
- Se'ns demana mostrar per pantalla els valors enters que conté el fitxer.
- No sabem quants valors conté. Per tant no podem fer un `for`. Cal usar l'algorisme anterior.
- Per llegir, com que sabem que conté enters, usem `fscanf` que, com `scanf`, ens permet llegir directament un enter. Podríem usar altres funcions que llegeixen cadenes, però hauríem de fer posterior conversió a enter. Sempre cal usar la funció més adequada. Em aquest cas: `fscanf`.

Solució: Projecte `240123_1_FitxerTextAmbEnters`

Variant al projecte anterior

- Donat un fitxer com en exercici, se'ns demana emplenar una taula de com a molt 250 enters, amb com a molt 250 enters del fitxer, per després poder gestionar els valors recuperats (podria ser ordenació, càlculs,...) i posteriorment mostrar els valors per pantalla.
- Cal controlar que, en cas que el fitxer contingui més de 250 enters, només en recuperi 250 i que, en cap cas, es sobrepassi la capacitat màxima de la taula.

Solució: Projecte `240123_2_OmplirTaulaEntersDeFitxer`

Exercici per dijous, 25-gener:

- Cal disposar d'un fitxer d'enters separats per espai en blanc, ja existent i que finalitza en espai en blac. Podeu copiar el fitxer dels projectes anteriors, però no usar l'original per què en modificarem el contingut.
- Fer programa que repeteixi el següent procés:
  - Pregunti a l'usuari, per consola, si vol introduir enter.
  - En cas afirmatiu, se li demani la introducció del valor i s'afegeixi al final del fitxer ja existent.

Solució: Projecte `240125_1_AfegirEntersEnFitxer`

Tasca 1 - Per dijous, 1 de febrer – Lliurar en tasca de Classroom fins 20:30 de 1 de febrer

Volem gestionar vehicles (codi, marca, model i cv) enregistrats en un fitxer de text. Decisions que prenem:

- En el programa, les vehicles les gestionarem via una tupla (`struct` de C) i, per aquest motiu, definim:

```
typedef struct {
   int codi;
   char marca[LEN_MARCA+1];
   char model[LEN_MODEL+1];
   int cv;
} tVehicle;
```

- En el fitxer de text, guardarem les vehicles en línies consecutives, seguint el format, com exemple:

```
100                    // codi
Ford                   // marca
Fiesta 5P 2018 1.0     // model
100                    // cv
200                    // codi
Ferrari                // marca
812GTS                 // model
800                    // cv
```

Ubiquem, dins carpeta `PROJ_VEHICLES`:

- **Projecte P0** amb contingut:

- `vehicles.h`, que contingui la definició de `tVehicle` i els prototipus de les funcions següents que podem usar en molts programes:
  - `void llegir_vehicle_teclat (tVehicle *v)` que demani les dades d'un vehicle a l'usuari per teclat i que validi els valors de tots els camps.
  - `void mostrar_vehicle_pantalla (tVehicle v)` que mostri un vehicle per pantalla
- `vehicles.c`, amb la definició (codi) de les funcions amb prototipus dins `vehicles.h`.

- **Projecte T1P1**, que permeti AFEGIR en un fitxer de text amb contingut com l'indicat, vehicles que introdueixi l'usuari per teclat. En cas que no existeixi, l'ha de crear.

  Nom de fitxer: `vehicles.txt`
  Ubicació: Carpeta `PROJ_VEHICLES`

  El `main` ha d'incorporar un bucle que pregunti a l'usuari si vol afegir vehicle i, si l'usuari afirma, cal invocar la funció `llegir_vehicle_teclat` i posteriorment, per afegir-lo en el fitxer, invocar funció `afegir_vehicle_fitxer_text` que haurem desenvolupat en el mateix projecte.

- **Projecte T1P2**, que permeti recuperar d'un fitxer de text amb contingut com l'indicat, tots els vehicles existents i els mostri per pantalla.

  Nom de fitxer: `vehicles.txt`
  Ubicació: Carpeta `PROJ_VEHICLES`

  El `main` ha d'incorporar un bucle que recuperi tots els vehicles del fitxer de text, invocant funció `recuperar_vehicle_fitxer_text` que haurem desenvolupat en el mateix projecte i mostri cada vehicle per pantalla invocant funció `mostrar_vehicle_pantalla`.

Es facilita carpeta `T1_MATERIAL` amb esquelet dels 3 projectes a desenvolupar.

Lliurament: Els 3 projectes + fitxer `vehicles.txt` amb algunes dades en carpeta **T1_CognomNom** comprimida

Tasca 2 - Per dijous, 8 de febrer – Lliurar en tasca de Classroom fins 20:30 de 8 de febrer

Gestió dels vehicles en fitxer de text amb codi de vehicle identificador

Cal fer una nova versió del projecte T1P1, que permetia afegir vehicles pel final, de manera que ha de comprovar que no existeixi dins el fitxer un vehicle amb el mateix codi. Anomeneu **T2P1** aquest projecte.

Ens convé disposar d'aquestes funcions que afegirem al projecte **P0**:

- `void llegir_codi_vehicle_teclat (int *codi)` que demani el codi de vehicle a l'usuari per teclat
- `void llegir_vehicle_teclat_menys_codi (tVehicle *v)` que demani totes les dades d'un vehicle, menys el codi, a l'usuari per teclat i que validi els valors de tots els camps.

Pseudocodi del programa a desenvolupar:

```
preguntar a usuari si vol introduir dades
mentre vulgui introduir fer
        demanar codi de vehicle
        obrir fitxer en mode lectura
        llegir vehicle de fitxer
        mentre no final de fitxer i vehicle llegit no mateix codi fer
                llegir vehicle de fitxer
        fimentre
        si final de fitxer fer        // vol dir que no s'ha trobat vehicle amb mateix codi
                demanar resta dades de vehicle
                tancar fitxer         // estava obert en mode lectura
                obrir fitxer en mode afegir
                enregistrar vehicle
        sinó    // vol dir que s'ha trobat el vehicle amb mateix codi i l'acabem de llegir
                informar que existeix i mostrar-lo
                tancar fitxer
        fisi
        preguntar a usuari si vol continuar introduint dades
fimentre
```

Amb el programa del projecte T1P2, que mostra el contingut del fitxer, podeu comprovar el correcte funcionament del programa T2P1 a desenvolupar.

Abans d'executar, si voleu continuar usant el fitxer `vehicles.txt` d'anterior exercici, comproveu que no conté vehicles amb codi repetit, i si és així, modifiqueu el codi amb qualsevol editor de textos.

Lliurament: Els projectes P0-T1P2-T2P1 + fitxer `vehicles.txt` amb en carpeta **T2_CognomNom** comprimida

Instruccions del llenguatge C per gestionar fitxers binaris (02/02/2023)

- Instrucció d'escriptura: `fwrite`.
- Instrucció de lectura: `fread`.

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

Exemples:
- Projecte `240208_1`, que crea un fitxer binari i enregistra uns valors (int, float, cadena, taula de double)
- Projecte `240208_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 29 bytes (4 per `int` + 4 per `float` + 5 per la cadena de caràcters + 8 per cadascun dels 2 `double` guardats).

Tasca 3 - Per dimarts, 13 de febrer – Lliurar en tasca de Classroom fins 20:30 de 13 de febrer

Crear una carpeta que es digui `T3_Cognom1Nom` (amb els vostres cognom i nom).
Ubiqueu dins aquesta carpeta els 2 projectes següents. Cal lliurar la carpeta T3 comprimida.

Programa 1, que enregistri valors `double` en 2 fitxers (un textual i un binari). Si els fitxers no existeixen, cal crear-los. Si existeixen, s'ha d'afegir les dades al final del contingut actual. Funcionalitat:
- Demani a l'usuari la introducció de valors `double` fins que l'usuari digui prou.
- Els valors introduïts, cal emmagatzemar-los simultàniament en 2 fitxers:
  - Un textual, de nom `valorsDouble.txt`, en format `########.#####` (8 dígits per part entera + 5 dígits per part decimal), separats amb un espai en blanc. La part entera completada amb espais per l'esquerra i la part decimal completada amb zeros per la dreta.
  - Un binari, de nom `valorsDouble.bin`.

Programa 2, que recuperi dels fitxers textual/binari generats en programa anterior, les dades que contenen, i les mostri per pantalla.
- En primer lloc, que recuperi les dades del fitxer textual, i mostri:
  `Dades de fitxer textual: .... (separades per guions)`
- En segon lloc, que recuperi les dades del fitxer binari, i les mostri en format:
  `Dades de fitxer binari: .... (separades per guions)`

Recordatori de 1r: Per forçar un determinat format de lectura/escriptura textual, cal utilitzar adequadament el format `%xxxxx` en les funcions `scanf`/`printf` i `fscanf`/`fprintf`.

Per a que els 2 programes treballin amb els mateixos fitxers, els ubiquem a la carpeta T3, fora dels 2 projectes.

Tasca 4 - Per dijous, 15 de febrer – Lliurar en tasca de Classroom fins 20:30 de 15 de febrer

Gestió dels vehicles en fitxer binari amb codi de vehicle identificador

Cal fer:

- Nova versió del projecte T2P1, que permetia afegir vehicles pel final d'un fitxer de text, de manera que havia de comprovar que no existís dins el fitxer un vehicle amb el mateix codi, però ara en un fitxer binari de vehicles. Anomeneu **T4P1** aquest projecte.

- Nova versió del projecte T1P2, que mostra el contingut del fitxer de text, mostrant els vehicles d'un fitxer binari. Anomeneu **T4P2** aquest projecte. Podeu comprovar el correcte funcionament del programa T4P1 executant aquest programa.

Lliurament: Els projectes P0-T4P1-T4P2 + fitxer `vehicles.bin` en carpeta **T4_CognomNom** comprimida

Instruccions del llenguatge C per moure's de posició (textuals i binaris) (15/02/2024)

Veure instruccions `fseek`, `ftell`, `fsetpos`, `fgetpos` i `rewind`.

Com llegir i escriure simultàniament en un mateix fitxer (textual i binari) sense tancar-lo/obrir-lo?

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).

És a dir:
- Per passar de lectura a escriptura, cal usar `fseek` o `fsetpos` o `rewind` abans d'escriure
- Per passar d'escriptura a lectura, cal usar `fflush` o `fseek` o `fsetpos` o `rewind` abans de llegir

El projecte `240215_1_Exemple_Lectura+Escriptura` mostra un exemple en fitxer textual.
Gestionem un fitxer `prova.txt` que conté un text amb moltes paraules.
Fixeu-vos que en primer lloc, usem la funció `system()` per invocar l'odre `copy` de Windows per a què ens copiï el fitxer `prova_original.txt` a `prova.txt`, ja que cada vegada que executem aquest programa, el fitxer prova.txt queda modificat i d'aquesta manera el podem tornar a executar partint del mateix fitxer.
1r. – Llegim 4 paraules.
2n. – Ens posicionem a l'inici (amb `rewind` o `fseek`)
3r. – Enregistrem un text
4t. – Llegim 5 paraules a partir del lloc on estem (ens ha calgut un `fflush` o `fseek` o `fgetpos+fsetpos`)
5è – Enregistrem un text a partir del lloc on estem (ens ha calgut executar un `fseek` o `fgetpos+fsetpos`)
6è – Saltem fins el final (`fseek`)
7è – Enregistrem un text.
Fitxer original:
`Avui és dimarts. Ha plogut tot el dia... Esperem que demà hi hagi sol.`
Fitxer final:
`ABRACADABRAmarts. Ha plogut tot elXXXX... Esperem que demà hi hagi sol.YYYY`
I per consola hem observat les paraules que anava llegint el programa.

Tasca 5 - Per dijous, 22 de febrer – Lliurar en tasca de Classroom fins 20:30 de 22 de febrer

Crear una carpeta que es digui `T5_Cognom1Nom` (amb els vostres cognom i nom).
Ubiqueu dins aquesta carpeta els 2 projectes següents. Cal lliurar la carpeta T4 comprimida.

Considerar els dos fitxers `valorsDouble.txt` i `valorsDouble.bin` que es faciliten i que contenen valors `double` en el format requerit a la tasca 3. Hi ha els mateixos valors en els 2 fitxers.

Programa 1:

Fer un programa que permeti a l'usuari recuperar valors indicant la posició que ocupen. És a dir:
- Demani a l'usuari la posició a recuperar, fins que digui prou.
- El programa mostri el valor que hi ha en aquella posició, tant en el fitxer textual com en el fitxer binari.
  Cal informar adequadament en cas que la posició no existeixi.
  No s'hi val a començar des de l'inici per anar a la posició demanada per l'usuari.

Programa 2:

Fer un programa que permeti a l'usuari substituir valors double existents en el fitxer, per altres valors. És a dir:
- Demani a l'usuari un valor `double` a substituir i el nou valor, fins que digui prou.
- El programa cerqui el valor en el fitxer, tant en el fitxer textual com en el fitxer binari i el substitueixi.
  Cal informar adequadament en cas que el valor a substituir no existeixi.

Tasca 6 - Per dijous, 29 de febrer – Lliurar en tasca de Classroom fins 20:30 de 29 de febrer

Crear una carpeta que es digui `T6_Cognom1Nom` (amb els vostres cognom i nom).

Cal fer programa **T6P1** que permeti modificar les dades (excepte el codi) dels vehicles en un fitxer binari on el codi és identificador, com el fitxer `vehicles.bin` generat a la tasca 4.

Flux del programa:
- Demanar codi de vehicle
- Cercar-lo
- Si no hi és, informar
- Si hi és:
  - mostrar-lo per pantalla, en un format similar a:
    1. Marca

2. Model
3. Potència
0. Finalitzar

- Permetre a l'usuari que indiqui el camp a modificar, demanant el nou valor, de forma repetida fins que introdueixi 0 per finalitzar.
- Substituir el vehicle inicial pel vehicle amb les dades modificades.

- Podeu usar el programa **T4P2** per comprovar que el contingut del fitxer binari ha quedat correctament modificat.

Lliurament: Els projectes P0-T6P1-T4P2 + fitxer `vehicles.bin` en carpeta **T6_CognomNom** comprimida

Tasca 7 - Per dimarts, 12 de març – Lliurar en tasca de Classroom fins 20:30 de 12 de març

Crear una carpeta que es digui `T7_Cognom1Nom` (amb els vostres cognom i nom).

Cal fer programa **T7P1** que permeti eliminar un vehicle d'un fitxer binari on el codi és identificador, com el fitxer `vehicles.bin` generat a la tasca 4.

Per eliminar dades d'un fitxer, ens ajudarem d'un fitxer temporal on anirem bolcant les dades a conservar (totes menys les que hem d'eliminar) i finalment eliminarem fitxer original i anomenarem el fitxer temporal amb el nom que tenia el fitxer original.

Flux del programa:

1. Demanar codi de vehicle
2. Obrir fitxer original en lectura i fitxer temporal en creació.
3. Començar a cercar vehicle en el fitxer original copiant en el fitxer temporal cada vehicle recuperat que no s'hagi d'eliminar.
4. Si no es troba el vehicle a eliminar, informar, tancar 2 fitxers i eliminar fitxer temporal.
5. Si es troba el vehicle a eliminar, NO copiar-lo en el fitxer temporal, i continuar recorrent fitxer original copiant els vehicles restants en el fitxer temporal. Finalment tancar els 2 fitxers, eliminar fitxer original i reanomenar el fitxer temporal.

- Instrucció en C per eliminar fitxer: `unlink("nomFitxer")`
- Instrucció en C per canviar nom a un fitxer: `rename("nomVell","nomNou")`

- Podeu usar el programa **T4P2** per comprovar que el contingut del fitxer binari ha quedat correctament modificat.

Lliurament: Els projectes P0-T7P1-T4P2 + fitxer `vehicles.bin` en carpeta **T7_CognomNom** comprimida

Tasca 8 - Per dijous, 4 d'abril – Lliurar en tasca de Classroom fins 20:30 de 4 d'abril

Crear una carpeta que es digui `T8_Cognom1Nom` (amb els vostres cognom i nom).
Ubiqueu dins aquesta carpeta els 3 projectes següents. Cal lliurar la carpeta T8 comprimida.

Programa **T8P1**: *Ordenar un fitxer petit bolcant-lo a la RAM*

- Considerar el fitxer `vehicles.bin` elaborat en tasca 4 (codi de vehicle és identificador)
- Ordenar-lo per la marca del vehicle

  Ordenar un fitxer no és una cosa senzilla... En aquest cas aplicarem el següent mecanisme, que **<u>només</u>** es pot usar quan es tracta d'un fitxer "petit".
  - Crear taula de vehicles de manera que hi càpiguen tots els vehicles del fitxer `vehicles.bin`.
  - Bolcar tots els vehicles de `vehicles.bin` dins la taula
  - Ordenar la taula per la marca dels vehicles (aplicar qualsevol dels algorismes d'ordenació vistos a M03-UF2)
  - Bolcar tots els vehicles de la taula en un nou fitxer `vomarca1.bin`

Posteriorment, feu una còpia del programa T4P2 en **T8P1R** (canviant nom `vehicles.bin` per `vomarca1.bin`) per comprovar que el fitxer `vomarca1.bin` ha quedat ordenat per marca de vehicle.

Programa **T8P2**: *Inserció ordenada en fitxer ordenat*

- Considerar el fitxer `vomarca1.bin` elaborat en programa anterior (codi identificador i ordenat per marca).
- Demanar les dades vehicle per teclat i, si no existeix en el fitxer cap vehicle amb el mateix codi, inserir-lo en el fitxer, que està ordenat per marca, de manera que continuï ordenat per marca. (Marca es pot repetir)

Pseudocodi del programa a desenvolupar:

```
preguntar a usuari si vol introduir dades
mentre vulgui introduir fer
        /* 1r. Demanar codi de nou vehicle i comprovar que no existeix cap vehicle amb tal codi */
        /* Cal tenir en compte que el fitxer no està ordenat per codi (ho està per marca) */
        demanar codi de vehicle
        obrir fitxer en mode lectura
        llegir vehicle de fitxer
        mentre no final de fitxer i vehicle llegit no mateix codi fer
                llegir vehicle de fitxer
        fimentre
        si no final de fitxer fer    // vol dir que s'ha trobat vehicle amb mateix codi
                informar que existeix un vehicle amb igual codi i mostrar-lo
                tancar fitxer
        sinó    // vol dir que no hi havia vehicle amb mateix codi i podem continuar
                tancar fitxer
                demanar resta dades de vehicle
                /* Procedir a inserir-lo invocant funció inserir_ord_marca que programem a part
        fisi
        preguntar a usuari si vol continuar introduint dades
fimentre
```

Pseudocodi de la funció `inserir_ord_marca (nom_arxiu_ord, vehicle)`:

```
obrir el fitxer original ordenat en mode lectura
obrir un fitxer temporal en mode creació
llegir vehicle de fitxer ordenat
mentre no final de fitxer ordenat i marca de vehicle llegit <= marca de vehicle a inserir (paràmetre vehicle) fer
        escriure vehicle llegit en fitxer temporal
        llegir vehicle de fitxer ordenat
fimentre
/* Ja hem copiat tots els vehicles amb marca anterior o igual a la del vehicle que volem inserir en f. temp. */
escriure vehicle a inserir (paràmetre vehicle) en el fitxer temporal
mentre no final de fitxer ordenat fer
        escriure vehicle llegit en fitxer temporal
        llegir vehicle de fitxer ordenat
fimentre
tancar els dos fitxers.
eliminar el fitxer original
anomenar el temporal amb el nom de l'original.
```

Posteriorment, executeu programa **T8P1R** per comprovar el vehicle ha quedat inserit en el lloc adequat, és a dir, que els vehicles del fitxer `vomarca1.bin` continuen ordenats per marca.

Programa **T8P3**: *Ordenar un arxiu pel mètode d'inserció de programa anterior*

- Considerar el fitxer `vehicles.bin` de la tasca 4 (vehicles amb codi identificador)
- Ordenar-lo en fitxer `vomarca2.bin`, aplicant la funció `inserir_ord_marca(nom_arxiu_ord, veh)` desenvolupada anteriorment.

  És a dir:
  - Crear fitxer `vomarca2.bin` buit
  - Fer un recorregut per l'arxiu original (`vehicles.bin`) aplicant per cada vehicle la funció `inserció_ord` per a que la insereixi en el fitxer `vomarca2.bin` que anirà creixent de forma ordenada.

Posteriorment, feu una còpia del programa T8P1R en **T8P3R** (canviant nom `vomarca1.bin` per `vomarca2.bin`) per comprovar que el fitxer `vomarca2.bin` conté tots els vehicles de `vehicles.bin` ordenats per marca.

Pràctica final - Lliurar en tasca de Classroom fins 23:59 de dilluns, 22 d'abril

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:
		`Número Nom Sexe Data naixement`
		`..... ..... ... .....`
		`..... ..... ... .....`

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:
		`Data Hora NF Nom NF Nom Temps Guanyador`
		`.... .... .. ..... .. ..... ..... .....`
		`.... .... .. ..... .. ..... ..... .....`
		on columna `NF` és el número de federat de cada contrincant i columna `Guanyador` mostrarà 1-2-0 com consta en el fitxer.

3. Configuració	1. Buidar partides 2. Buidar tot	- L'opció *Buidar partides* ha d'eliminar totes les partides - L'opció *Buidar tot* 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 efectuï 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 efectuï 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*".

El lliurament ha de contenir el projecte comprimit amb fitxers a l'arrel amb dades correctes incorporades.