Exercici L7

Ordenar i cercar

Fonaments de Programació II

Objectiu

L'objectiu d'aquesta pràctica és que practiqueu cerca i ordenació en un entorn més realista. Per això, ordenarem una taula de registres que representen dades de persones, que haurem d'ordenar per cognom, i un cop ordenat, en farem cerques.

Descripció de l'exercici

En aquest exercici farem servir l'algorisme d'**ordenació per inserció** i, posteriorment, el de **cerca binària**. Per aplicar-ho en un entorn més realista, disposarem d'un fitxer de text amb dades de persones. De cada persona en tenim el **nom**, el **cognom**, el **DNI** i l'**edat**.

Heu d'escriure un programa amb l'objectiu de **poder fer consultes sobre aquests registres**. Els passos a seguir són:

1. Llegir les dades del fitxer i emmagatzemar-les en una **taula dinàmica de structs** de tipus Persona.
2. Ordenar la taula pel **cognom**, fent servir l'algorisme d'**ordenació per inserció**.
3. Permetre que l'usuari faci **cerques pel cognom**, utilitzant **cerca binària** sobre la taula ja ordenada.

Per fer la cerca, implementareu un petit menú que demanarà quin és el cognom a buscar. Un cop introduït, el programa cercarà fent servir cerca binària.

- Si es troba alguna coincidència, s'imprimirà la informació completa de la persona (o persones).
- Si no es troba cap coincidència, s'imprimirà un missatge indicant-ho.

Aquest programa el desenvoluparem en **dues fases**:

- En la **primera fase**, si es troba un cognom coincident, només s'imprimirà **un únic registre**.

- En la **segona fase**, si hi ha diversos registres amb el mateix cognom, **es mostraran tots**.

Lectura de fitxer

El fitxer que se us proporciona té el format següent:

- La **primera línia** conté el nombre de persones que inclou el fitxer. Haureu de llegir aquest valor per poder reservar la taula de memòria dinàmica.
- La segona línia és una capçalera informativa perquè sapigueu què vol dir cada camp. Aquesta línia no conté informació que necessiteu.
- A partir de la tercera línia, cada línia conté les dades d'una persona, amb els camps **separats per tabuladors** (`\t`):
    - Nom
    - Cognom
    - DNI
    - Edat

Per facilitar-vos la feina (i perquè l'objectiu de la pràctica és **ordenar i cercar**, no pas llegir fitxers complicats), s'ha assegurat que **tant el nom com el cognom són una única paraula**, sense espais.

La funció de lectura haurà de desar aquesta informació en una **taula dinàmica de registres** de tipus `Persona`, que ha de contenir els quatre camps indicats.

Mostrar informació

Per tal de comprovar que la lectura s'ha fet correctament, i més endavant verificar també el resultat de l'ordenació, haureu d'implementar:

- Una funció que imprimeixi la informació d'un **únic registre** (nom, cognom, edat i DNI). Aquesta funció també s'utilitzarà per mostrar el resultat d'una cerca.
- Una altra funció que imprimeixi **tots els registres de la taula**. Consell**:** durant la fase de desenvolupament, us serà útil que aquesta funció també imprimeixi la posició dins la taula de cada registre.

Ordenació per inserció

L'algorisme d'ordenació serà el d'inserció, que s'ha vist a classe en pseudocodi[1], aplicat a una taula d'enters. Haureu d'adaptar aquest codi per tal que ordeni la taula de registres `Persona`, i ho faci tenint en compte el cognom.

Per ordenar alfabèticament pel cognom, podeu aprofitar la funció `strcmp`, que ja hem fet servir anteriorment. Si recordeu, aquesta funció compara dos strings i retorna:

- 0 si són iguals.
- Un valor **negatiu** si el primer string va **abans** que el segon segons l'ordre lexicogràfic.
- Un valor **positiu** si el primer string va **després** que el segon segons l'ordre lexicogràfic.

Exemple:

```
strcmp("Alonso", "Zamora");  // retorna un valor negatiu (Alonso < Zamora)
strcmp("Garcia", "Garcia");  // retorna 0 (Garcia = Garcia)
strcmp("Vila", "Ferrer");    // retorna un valor positiu (Vila > Ferrer)
```

Recordeu que quan moveu elements dins la taula, heu de copiar tot el registre, no només el cognom.

Cerca binària per cognom

Un cop la taula de registres estigui ordenada, ja podrem fer cerques usant l'algorisme de cerca binària. Aquest algorisme també l'hem vist a classe, en pseudocodi, aplicat a una taula d'enters; només cal que l'adapteu perquè funcioni amb una taula de registres i cerqui pel cognom. Altra vegada, necessitareu la funció `strcmp`.

Com sabeu i es va explicar a classe, l'algorisme de cerca binària, en cas que hi hagi diversos registres amb la mateixa clau, **no assegura quin dels registres retorna**. Per tant, haureu de buscar una manera que, un cop localitzat un registre, es **retornin tots els registres coincidents**.

Per no afegir complexitat de bones a primeres, es demana que ho feu en **dues fases**:

- **Primera fase**: adaptar l'algorisme de cerca binària perquè busqui pel cognom, i comprovar que funciona correctament. Si cerquem un cognom que és a la llista, n'imprimeix el registre. Si hi ha repetits, només n'imprimeix un.
- **Segona fase**: Un cop heu verificat que la cerca binària funciona correctament per retornar un únic registre, haureu d'adaptar-la per tal que **retorni tots els registres amb el mateix cognom**. *Pista: com que la taula està ordenada per cognom, tots els registres amb el mateix cognom estaran junts.*

[1] Consulteu la última versió de les transparències de teoria, disponible al Campus Virtual.

A la fase 2 podria ser temptador imprimir directament la informació de tots els registres amb el cognom buscat des de dins de la pròpia funció de cerca binària. Ara bé, això **violaria el principi de separació de responsabilitats**: aquesta funció s'ha d'encarregar exclusivament de **localitzar les coincidències**, no pas de mostrar-les per pantalla.

Per tant, haureu de trobar una manera perquè la funció de cerca indiqui al programa principal quines són les posicions dins la taula on es troben les coincidències. Serà llavors el programa principal qui s'encarregarà d'imprimir-ne la informació.

Tipus, funcions i estructura de fitxers

Com veureu, aviat aquesta pràctica es converteix en un projecte de mida considerable. Per tal de mantenir el codi organitzat i modular, es recomana separar la implementació en diversos fitxers:

- **persona.h**: declaracions del tipus Persona i de totes les funcions relacionades.
- **persona.c**: implementació de les funcions declarades a l'encapçalament.
- **main.c**: programa principal, que utilitza les funcions definides anteriorment.

Per pensar-hi més

Ordenació multicriteri: Com modificaries la funció d'ordenació si volguéssim que en cas d'empat de cognom, els registres estessin ordenats per un altre camp (DNI, nom...)?

Apèndix: gestió d'errors i alliberament de recursos

A classe hem vist en diverses ocasions que, quan una operació com la reserva de memòria falla, o bé quan no es pot obrir un fitxer, el programa ha de mostrar un missatge d'error i finalitzar. Ara bé, quan es produeix un error d'aquest tipus, cal tenir en compte si hi ha recursos oberts (fitxers, memòria dinàmica, etc.) que cal tancar o alliberar abans de sortir.

Per garantir que el programa finalitza de manera ordenada i controlada, és important no oblidar aquests passos previs. Per tal d'evitar duplicar codi a cada punt on hi pot haver un error, és recomanable encapsular aquest comportament en una funció com la següent:

```c
void cleanup_and_exit(int * taula, FILE *f, const char *missatge) {
    if (f != NULL){
       fclose(f);
    }
    if (taula != NULL){
       free(taula);
    }
    fprintf(stderr, "%s\n", missatge);
    exit(EXIT_FAILURE);
}
```

Aquesta funció rep tres paràmetres: un punter a una taula reservada amb memòria dinàmica, un punter a un fitxer obert i un missatge d'error. Si el fitxer està obert, el tanca. Si la taula ha estat reservada, l'allibera. A continuació, imprimeix el missatge d'error i finalitza l'execució del programa amb `exit(EXIT_FAILURE)`.

La manera de cridar-lo variarà depenent de la fase en la què esteu: per exemple, si esteu reservant memòria però el fitxer encara no està obert, podeu passar `NULL` al camp del fitxer. Aquesta funció és només una mostra i podeu adaptar-la al tipus de dades amb què treballeu, o afegir-hi paràmetres si teniu més recursos a alliberar.

Alternativa: també podeu separar responsabilitats i fer una funció només de *cleanup*, que s'encarregui exclusivament d'alliberar la memòria i tancar els fitxers. En aquest cas, seria el programa principal qui s'encarregaria de mostrar el missatge d'error i cridar `exit`. Aquesta opció pot ser útil si voleu tenir més control sobre el missatge que s'imprimeix.