Semestre 2 - Pràctica 2
LSClassrooms

Índex

1. Introducció

A causa de l'augment del número d'alumnes a La Salle, des de Coordinació Acadèmica se'ns ha demanat que desenvolupem una eina de suport per poder assignar els alumnes a les aules corresponents segons el grau que desitgin cursar.

2. Objectius

Els objectius bàsics d'aquesta pràctica son:

- Entendre el funcionament de punters i memòria dinàmica.
- Entendre els avantatges, inconvenients i limitacions de la memòria dinàmica.
- Aprendre a gestionar correctament la memòria dinàmica.
- Posar en pràctica la utilització de punters per a crear, gestionar i modificar arrays dinàmics.
- Comprendre la diferència entre utilitzar un punter per simular el pas d'arguments per referència i el seu ús per a gestionar la memòria dinàmica.

3. Funcionament

El procés d'entrega d'aquesta pràctica varia respecte a les anteriors. Si us plau, llegiu la secció "Procediment de lliurament".

El primer que farà el programa serà donar la benvinguda i demanar els noms dels fitxers de text on es troben les dades dels graus i les classes de cada grau (*fitxer classrooms*) i estudiants (*fitxer students*). Una vegada es llegeixin correctament, es mostrarà el menú principal, que es detallarà en els següents apartats (veure Output1).

```
Welcome!

Type the name of the 'classrooms' file: class_1

Type the name of the 'students' file: stus_1

1. Summary | 2. Show degree students | 3. Move student | 4. Exit
Select option:
```

Output 1. Petició dels noms dels fitxers.

Després d'introduir els noms dels fitxer que contenen la informació, s'haurà de realitzar la seva lectura. Per a realitzar la lectura és important tenir en compte els següents punts:

- No es pot realitzar la lectura de cap fitxer directament a la funció *main*. S'hauran d'implementar una o més funcions o procediments a part, on es llegiran i guardaran les dades en les estructures adients.
- No sabem quin es el nombre màxim de graus, classes ni alumnes que hem de gestionar. Per aquest motiu (i com s'explicarà a continuació) el primer fitxer inclou el número d'elements que conté per poder gestionar correctament la memòria.
- Relacionat amb el punt anterior, no s'admetrà en cap cas l'ús d'arrays estàtics sobredimensionats per emmagatzemar les dades dels graus, classes i alumnes.

Si s'introdueix un nom de fitxer erroni (és a dir, si no existeix el fitxer), es mostrarà un error i es tornarà a demanar el nom del fitxer fins que aquest sigui correcte (veure Output 2).

```
Welcome!

Type the name of the 'classrooms' file: clas

ERROR: Can't open file 'clas'

Type the name of the 'classrooms' file: class_1

Type the name of the 'students' file: stuts

ERROR: Can't open file 'stuts'

Type the name of the 'students' file: stus_1

1. Summary | 2. Show degree students | 3. Move student | 4. Exit
Select option:
```

Output 2. Errors en els noms dels fitxers.

3.1 Estructura dels fitxers i tipus propis

Abans de poder avançar amb la interacció entre usuari i programa, serà necessari llegir i emmagatzemar correctament les dades dels fitxers.

Com s'ha comentat, el fitxer de graus i classes indicarà la quantitat total d'elements que conté amb un enter; això es així per al total de graus, el total de classes i l'aforament màxim de cada classe.

El format concret dels fitxers és el següent:

- Fitxer de **graus i classes**:

```
<quantitat graus>
<quantitat classes del grau> <nom del grauX>
<classe1> <capacitat max. classe1>
<classeX> <capacitat max. classeX>
```

El nom del grau podrà ser de més d'una paraula, mentre que el nom de la classe serà sempre d'una paraula (sense espais).

- Fitxer **d'estudiants**:

```
<nom>, <nom del grau>
<login>
```

Tant el nom com el login de l'alumne seran d'una sola paraula (sense espais).

Exemple de format:

- Fitxer de **graus i classes:**

```
1
2 Computer Engineering
JH1.1 3
JH1.2 2
```

- Fitxer **d'estudiants:**

```
Willow, Computer Engineering
frostmourne
```

Podeu trobar uns fitxers d'exemple amb aquest format juntament amb l'enunciat de la pràctica.

Sabent ja com s'estructuren els fitxers, revisem a continuació com s'haurien d'emmagatzemar aquestes dades.

3.2 Emmagatzemament de les dades

És **obligatori** emmagatzemar les dades en arrays dinàmics, dos per a les dades obtingudes del fitxer de *graus* i *classes* i un altre array dinàmic amb les dades extretes del fitxer *estudiants*. A continuació teniu els tipus propis que **són obligatoris** utilitzar per emmagatzemar tota aquesta informació. A la Figura 1 trobareu la seva representació.

```
#define MAX_STRING_LENGTH 70

typedef struct {
    char name[MAX_STRING_LENGTH];
    char login[MAX_STRING_LENGTH];
} Student;

typedef struct {
    char name[MAX_STRING_LENGTH];
    int max_capacity;
    int current_capacity;
    Student *students;
} Classroom;

typedef struct {
    char name[MAX_STRING_LENGTH];
    int num_classrooms;
    Classroom *classrooms;
} Degree;

typedef struct {
    int num_degrees;
    Degree *elements;
} Degrees;
```

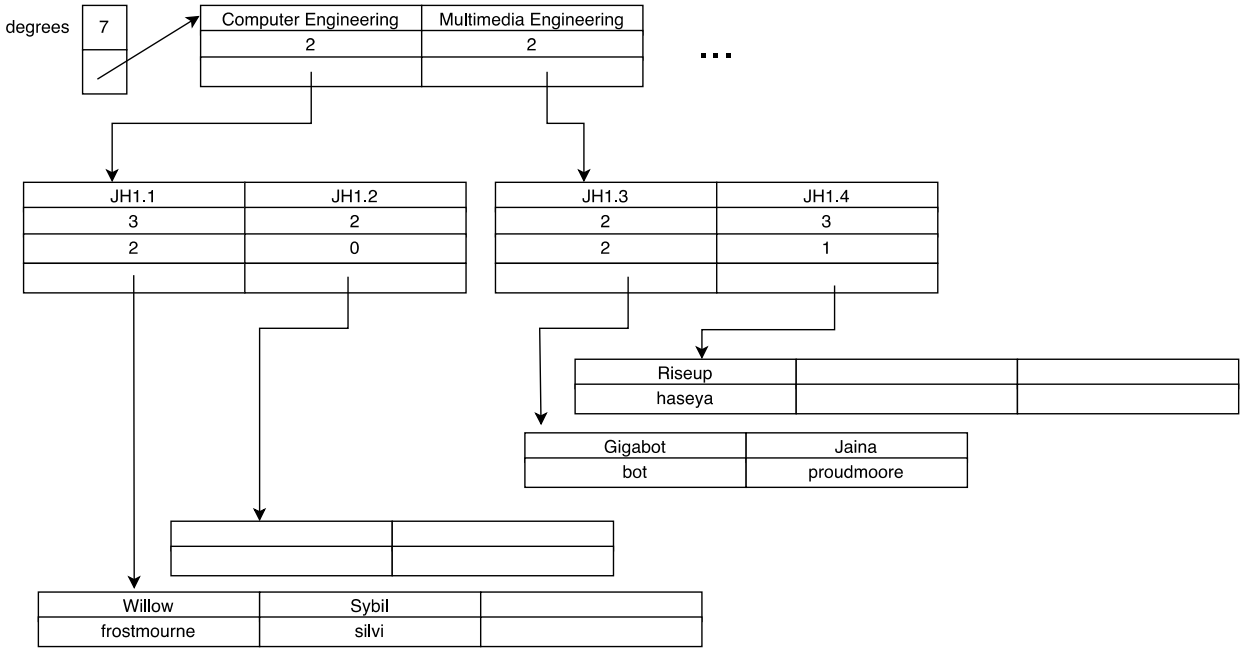


Figura 1. Representació d'estructures de dades d'emmagatzemament. Només es mostra part del contingut dels fitxers d'exemple. Es pot observar que el dimensionament dels arrays dinàmics es el màxim, tant pels graus (representat amb punts suspensius), per a les classes, com per al màxim aforament d'estudiants per classe.

És **obligatori i molt important** emmagatzemar les dades en el mateix ordre en què es troben en el fitxer. És a dir, el primer grau llegit ha de ser el que ocupi la posició 0 a l'array, el següent la posició 1 i així successivament i per a tots els fitxers i arrays.

En el cas de la càrrega, emmagatzemament i assignació d'estudiants a les classes d'un grau, cada vegada que es llegeixi un estudiant aquest es guardarà en el grau corresponent i en el següent espai lliure de la primera classe amb espai disponible. Dit d'una altra manera, s'aniran col·locant els estudiants de forma seqüencial en les classes d'un grau. Ens asseguren que sempre hi haurà espai entre totes les classes d'un grau per emmagatzemar a tots els estudiants d'aquest grau.

3.3 Menú

Una vegada s'hagin llegit els fitxers i s'hagin emmagatzemat totes les dades, es mostrarà un breu menú on es demanarà a l'usuari quina acció desitja realitzar (veure Output 3).

```
Welcome!

Type the name of the 'classrooms' file: class_1

Type the name of the 'students' file: stus_1

1. Summary | 2. Show degree students | 3. Move student | 4. Exit
Select option:
```

Output 3. Menú principal.

Aquest menú només admetrà opcions compreses en el rang [1,4]. Si s'introdueix un valor invàlid s'ha de mostrar un error i tornar a demanar l'opció fins que s'introdueixi un valor correcte (veure Output 4).

```
1. Summary | 2. Show degree students | 3. Move student | 4. Exit
Select option: 0

ERROR: Wrong option number

1. Summary | 2. Show degree students | 3. Move student | 4. Exit
Select option:
```

Output 4. Menú principal. Selecció de l'opció incorrecta.

3.4 Opció 1: Summary

Mostra un resum dels graus, les classes de cada grau i la capacitat actual i màxima. Una vegada finalitzada l'acció, es tornarà al menú principal (veure Output 5).

```
1. Summary | 2. Show degree students | 3. Move student | 4. Exit
Select option: 1

Computer Engineering
JH1.1 2/3
JH1.2 0/2

Multimedia Engineering
JH1.3 2/2
JH1.4 1/3

Telecommunications Systems Engineering
Measurements 3/4

Telematics
CCNA 1/3

Audiovisual Systems Engineering
JH1.5 1/2

Electronic Engineering
JH2.1 3/3
JH2.2 0/3

Engineering in ICT Management
JH2.3 3/4

1. Summary | 2. Show degree students | 3. Move student | 4. Exit
Select option:
```

Output 5. Opció 1: Resum de graus i aules.

3.5 Opció 2: Show degree students

Pregunta el nom d'un grau i mostra els estudiants d'aquest grau i, per a cada un, la classe (veure Output 6). Si el nom del grau no existeix, es mostrarà un missatge d'error (veure Output 7). En qualsevol cas, es tornarà al menú principal una vegada finalitzada l'acció.

```
1. Summary | 2. Show degree students | 3. Move student | 4. Exit
Select option: 2

Degree to show? Computer Engineering

Willow (frostmourne): JH1.1
Sybil (silvi): JH1.1

1. Summary | 2. Show degree students | 3. Move student | 4. Exit
Select option:
```

Output 6. Resum d'alumnes d'un grau.

```
1. Summary | 2. Show degree students | 3. Move student | 4. Exit
Select option: 2

Degree to show? CE

ERROR: Can't find degree

1. Summary | 2. Show degree students | 3. Move student | 4. Exit
Select option:
```

Output 7. Error en introduir el nom del grau.

3.6 Opció 3: Move student

Aquesta opció permet moure a un alumne de classe, sempre dins del mateix grau. En cas que succeeixi un error dels que us hem indicat a continuació, s'haurà de mostrar un missatge d'error genèric (veure Outputs 9, 10 i 11) i tornar al menú principal.

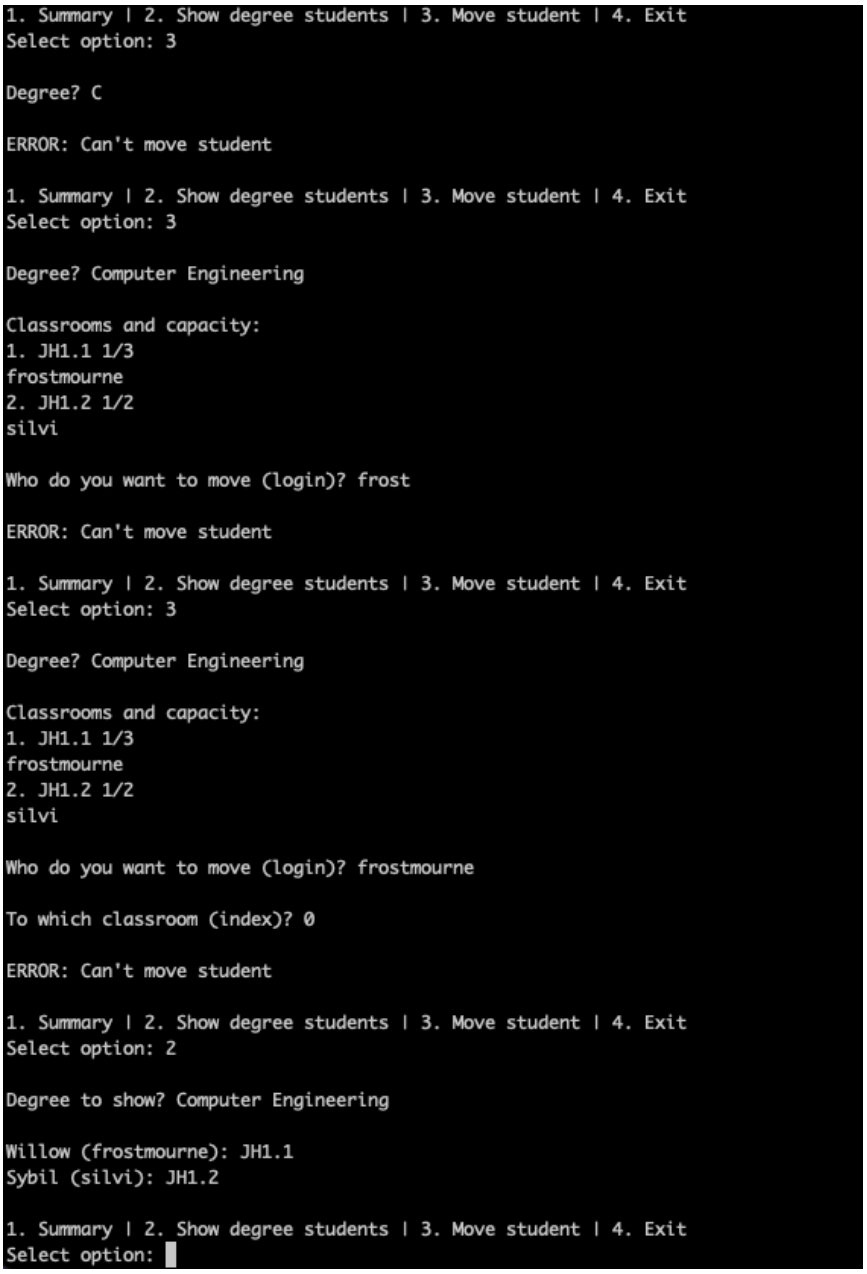
Així doncs, quan l'usuari, des de el menú principal, seleccioni l'opció 3:

1. El programa demana el nom d'un grau. S'haurà de controlar que el grau existeixi.
2. Es mostren les classes del grau (amb un índex auto-incremental davant de cada classe) amb un llistat dels *logins* dels estudiants de cada classe.
3. El programa demana el *login* de l'alumne que es vol moure de classe. S'haurà de controlar que el *login* existeix en aquest grau.
4. El programa pregunta l'índex de la classe on es vol moure l'alumne. Es controlarà que l'índex de la classe sigui vàlid (es troba entre els índexs del llistat de la classe anterior), que la classe de destí no sigui la d'origen, i que hi hagi espai disponible a la classe de destí.

Es pot veure una execució sense errors a l'Output 8. A l'Output 9 es pot observar que, en cas que hi hagi algun dels errors llistats anteriorment, es mostra el mateix missatge genèric d'error. L'Output 10 i l'Output 11 mostren l'error genèric quan s'intenta moure a un estudiant a la mateixa classe i quan s'intenta moure un estudiant a una classe plena, respectivament.

```
1. Summary | 2. Show degree students | 3. Move student | 4. Exit
Select option: 3

Degree? Computer Engineering

Classrooms and capacity:
1. JH1.1 2/3
frostmourne
silvi
2. JH1.2 0/2

Who do you want to move (login)? silvi

To which classroom (index)? 2

1. Summary | 2. Show degree students | 3. Move student | 4. Exit
Select option: 2

Degree to show? Computer Engineering

Willow (frostmourne): JH1.1
Sybil (silvi): JH1.2

1. Summary | 2. Show degree students | 3. Move student | 4. Exit
Select option:
```

Output 8. Es realitza l'opció 3 sense errors. A mode de comprovació, es realitza també l'opció 2 per mostrar si l'alumne s'ha mogut.



Output 9. Es realitzen diversos intents per moure a un estudiant. A cada intent l'usuari s'equivoca en introduir alguna dada i es mostra un error genèric. Finalment l'usuari selecciona l'opció 2 per comprovar que no s'ha mogut l'estudiant.

```
Select option: 3

Degree? Computer Engineering

Classrooms and capacity:
1. JH1.1 1/3
frostmourne
2. JH1.2 1/2
silvi

Who do you want to move (login)? frostmourne

To which classroom (index)? 1

ERROR: Can't move student

1. Summary | 2. Show degree students | 3. Move student | 4. Exit
Select option:
```

Output 10. No es pot moure un estudiant a la seva mateixa classe.

```
Select option: 3

Degree? Multimedia Engineering

Classrooms and capacity:
1. JH1.3 2/2
bot
proudmoore
2. JH1.4 1/3
haseya

Who do you want to move (login)? haseya

To which classroom (index)? 1

ERROR: Can't move student

1. Summary | 2. Show degree students | 3. Move student | 4. Exit
Select option:
```

Output 11. No es pot moure un estudiant a una classe plena.

3.7 Exit

En el menú principal, quan l'usuari seleccioni l'opció Exit, s'haurà de sortir del programa i alliberar tota la memòria dinàmica reservada (veure Output 12).

```
1. Summary | 2. Show degree students | 3. Move student | 4. Exit
Select option: 4

Bye!
```

Output 12. Sortida del programa.

4. Procediment de lliurament

El lliurament d'aquesta pràctica s'haurà de realitzar tenint en compte alguns canvis respecte les pràctiques anteriors.

Com sabeu, les pràctiques s'han d'implementar en el servidor Matagalls, compilant amb GCC. En canvi, per a que es puguin realitzar les proves d'execució a CodeRunner, haureu de canviar el nom de la vostra funció **main** per **myMain**. D'aquesta manera la capçalera de la vostra funció quedaria:

int myMain ()

Observeu la Imatge 1 per a tenir un exemple:

```
1  // Code
2
3  // Rewrite your "main" to "myMain"
4  int myMain() {
5      // Code
6  }
7
8  /**
9   * There should NOT be an "int main() { ... }" here
10  *
11  **/
12
```

Imatge 1. Anomenem a la funció main, myMain

Recordeu que aquest canvi només s'ha de realitzar quan vulgueu executar els tests de CodeRunner. Per poder compilar satisfactòriament el vostre codi quan estigueu treballant a Matagalls haureu d'usar un *main* estàndard com heu fet fins ara.

4.1 Tests

Tal i com s'indica en aquest enunciat, en aquesta pràctica s'haurà d'usar memòria dinàmica per a crear certes estructures de dades. Es per això, que en l'execució dels tests de CodeRunnner es comprovarà que això s'ha realitzat satisfactòriament. Concretament, si no es compleixen certes condicions, en els tests apareixerà, a la columna "Got", un missatge d'error al final de la sortida del programa. Es distingeixen els següents casos:

- "`ERROR: Mismatch size`": Aquest error apareixerà si s'ha demanat menys memòria de la necessària per a crear les estructures dinàmiques descrites a l'enunciat.
- "`ERROR: Some pointer exceeded its allocated memory`": Aquest error apareixerà si s'ha accedit a alguna direcció de memòria que no ha sigut reservada pel programa.
- "`ERROR: Some pointers have not been deallocated`" Aquest error apareixerà si no s'ha alliberat tota la memòria dinàmica en finalitzar el programa.
- També apareixerà un error si s'intenta alliberar memòria dinàmica ja alliberada (doble *free*).

5. Consideracions

Per a la implementació d'aquesta pràctica heu de tenir en compte les següents consideracions:

1. S'ha de seguir el format mostrat en els exemples.
2. La forma de gestionar els errors es estrictament la mostrada en l'enunciat.
3. El programa s'ha d'estructurar correctament en procediments i funcions. No s'acceptarà un programa amb codi mal estructurat.
4. Podeu suposar que els fitxers no estaran buits i seguiran el format específic, tot i que s'han de controlar els errors indicats i els possibles errors d'obertura.
5. Disposareu dels fitxers d'exemple usats a les imatges d'aquest enunciat per comprovar el funcionament del programa.
6. Haureu d'emmagatzemar les dades en arrays dinàmics tal i com s'ha explicat en aquest enunciat seguint les estructures descrites.
7. Es obligatori utilitzar únicament estructures de dades equivalents a les llistades en aquest enunciat (Degrees, Degree, Classroom, Student).

6. Requisits mínims

Aquesta pràctica ha de complir amb una sèrie de requisits de qualitat mínims per poder ser avaluada:

- El codi ha d'estar correctament comentat de manera que sigui llegible sense dificultats.
- Ha de seguir la guia d'estils de l'assignatura.
- Per poder aprovar es necessari que la pràctica superi satisfactòriament els tests de Coderunner.
- El programa hauria de desenvolupar-se íntegrament en l'entorn Linux de Matagalls, utilitzant l'editor Vim per escriure el codi C i el compilador gcc per a l'obtenció de l'executable corresponent.
- No es pot utilitzar cap eina ni instrucció de C que no s'hagi explicat a classe.
- El codi ha d'estar estructurat correctament utilitzant **un mínim de 6 procediments o funcions** significatius/ves. En aquest sentit, es valorarà especialment que en el codi **no es repeteixin conjunts de sentències.**
- No s'acceptarà cap pràctica que no compleixi amb la normativa de pràctiques.

7. Data de lliurament i avaluació

La data de lliurament d'aquesta pràctica per poder obtenir la màxima qualificació és el 24 d'abril de 2022.

Per a que aquesta pràctica es consideri **lliurada** s'hauran de complir les següents condicions:

1. Executar el codi en el CodeRunner habilitat expressament i superar satisfactòriament els tests.
2. Lliurar en el pou corresponent un fitxer .c amb el codi de la pràctica.

Recordeu que la pràctica serà avaluada de la següent manera:

- Execució: té un pes del 80% i avalua el correcte funcionament de la pràctica. Serà la qualificació obtinguda en els tests de CodeRunner.

- Qualitat del SW: té un pes del 20% i s'avaluarà la qualitat del codi lliurat i el seguiment de la Guia d'Estils de Programació que teniu disponible a l'eStudy.

Important: El fet de superar únicament els tests no implica que la pràctica s'hagi aprovat.