Programa de Gestió de Matrícula i Expedients de Notes
Enunciat
Segona Pràctica

Volem un programa que ens gestioni la matricula i generació d'expedients de notes d'alumne. Volem que el programa sigui robust i tingui les suficients (però mínimes) funcionalitats per operar per un usuari i per practicar els conceptes de l'assignatura, però que a la vegada estigui ben dissenyat per poder-se estendre amb totes les funcionalitats d'interfície i operativitat possibles o necessàries en un futur.

Robust vol dir que el programa ha de funcionar correctament en tots els casos possibles i no pot fallar en cap cas. Això implica que totes les funcionalitats que s'implementin (per poques que siguin) han de ser complertes i contemplar les inconsistències i imprevistos per detectar-los i actuar amb conseqüència. Sempre que es pugui és millor mantenir el programa funcionant identificant opcions de coherència davant el problema per proposar a l'usuari una situació de coherència alternativa que si és confirmada es manté com situació acceptable i consistent. Però si no es pot identificar una alternativa no queda més remei que implementar una parada controlada del programa per evitar inconsistències i possibles situacions que es pugui penjar el programa sense avís. Parar el programa controladament vol dir que abans es poden fer les operacions que calgui per mantenir la consistència del programa per quan torni arrancar (en el nostre cas, vol dir gravar dades a fitxers abans de sortir). La robustesa s'ha d'aplicar en cada part del codi o funcionalitat del programa que s'implementi.

El programa ha de guardar les dades en fitxers text. Les dades de cada alumne han d'estar en un fitxer expedient separat, i la llista d'assignatures ofertades a un fitxer assignatures. Aquests fitxers són fitxers tipus text i tenen un format concret detallat en la secció 1.2.

No es proporciona la interfície per la construcció i manipulació del fitxer assignatures, i es suposa que es té el fitxer creat i si cal manipular-lo es fa amb un editor de text (es proporciona fitxer model : asgs.txt). És imprescindible carregar el fitxer per tenir el llistat d'assignatures a reconèixer en els expedients. Per tant el programa ha d'avortar si no troba el fitxer i informar del problema a l'usuari.

El programa ha de començar carregant els fitxers existents per començar amb les dades carregades a memòria i llavors entrar a un menú que visualitzi les opcions mínimes per operar (per acotar la pràctica) definides en la secció 1.1.

1 Funcionalitats del Programa

1.1 Menú

Les operacions mínimes que ha d'implementar el programa són:

1. Carregar fitxers: carrega fitxers existents
2. Alta d'expedient: Registra un alumne nou i crea un expedient amb les dades generals però sense assignatures
3. Entrar matricula: crea expedients nous i modifica fitxers expedients existents
4. Entrar notes manualment: entra totes les notes d'una assignatura (i actualitza fitxers)
5. (opcional) Estat dels expedients: per cada expedient en memòria mostra el resum de l'expedient i indica en quina situació es troba dins el pla docent que està matriculat (veure secció 1.2)

És possible que un cop haguem entrat creem més fitxers expedients o modifiquem amb un editor de text el fitxer assignatures o fitxers expedients existents. Per això, té sentit tenir al menú l'opció carregar fitxers que li permet a l'usuari dir-li implícitament al programa que carregui els fitxers. Al recarregar fitxers pot haver-hi inconsistència entre les dades ja entrades a memòria i les llegides de fitxers. Cal garantir consistència de dades dins el programa per tenir un programa robust (veure seccions 1.4 i 1.5).

L'opció de donar d'alta un expedient és per donar d'alta alumnes nous a l'escola/programa. Això implica assignar un número d'identificador d'alumne (NIA) únic dins l'escola, i entrar les dades generals de l'alumne. Al donar d'alta l'expedient no se li assigna assignatures. Aquesta opció ha de demanar totes les dades generals d'alumne (que apareixen al format del fitxer expedient) una a una a l'usuari. La creació d'un expedient implica que ha de crear-se el fitxer expedient corresponent amb les dades conegudes de l'alumne seguint el format establert.

L'opció matricula serveix per afegir assignatures a l'expedient d'un alumne i en el moment de la matricula l'assignatura està sense nota. Si l'assignatura que s'intenta matricular ja forma part de l'expedient només es pot tornar a matricular si la nota és suspesa i llavors la matricula és per la següent convocatòria.

Les opcions alta d'expedients i entrada de matricula operen sobre un expedient actiu. Per tant al executar l'opció cal demanar un número de NIA a l'usuari i després demanar totes les dades per crear l'expedient o per entrar la matricula segons el cas. Al acabar demanar un nou NIA per activar un nou expedient i repetir la operació sobre el nou expedient actiu o tornar al menú si el valor de NIA entrat no és vàlid.

L'opció d'entrar notes funciona igual a les anteriors però sobre assignatures, és a dir, demana un codi d'assignatura per fer una assignatura activa i a continuació demana totes les notes dels alumnes matriculats a l'assignatura. Al acabar, demana un nou codi d'assignatura per continuar entrant notes d'una altra assignatura o sortir si el codi d'assignatura no existeix.

L'opció estat d'expedients serveix per revisar l'estat d'expedients i saber si es poden expedir títols o quan falta per graduar-se. La funcionalitat d'aquesta opció bé definida pel pla d'estudis que es descriu en la secció 1.2. Aquesta opció és opcional i puja nota si es fa (veure avaluació a la secció 4).

En tots els casos del menú, abans d'acabar una opció i retornar al menú principal cal gravar les dades de memòria a fitxer per reduir el risc de perdre la consistència de les dades degut a imprevistos en el programa. Al gravar s'informa a l'usuari de que es fa i si hi ha algun problema s'actua en conseqüència.

El sistema de menús per interactuar amb l'usuari es pot fer com es vulgui, és a dir, es pot escollir com mostrar les opcions i què cal que contesti l'usuari, però cal que sigui clar i en tot moment doni les instruccions que l'usuari ha de seguir per continuar per tal de que el programa sigui robust. No ha d'incloure menús gràfics sinó simples instruccions que permetin interactuar amb el programa. L'objectiu de la pràctica és el disseny intern del programa i no la interfície.

1.2 (opcional) Definició de Grau

Un grau té una càrrega lectiva total concreta distribuïda en assignatures. Un cop es compleix tots els requisits es pot atorgar el títol corresponent. En el procés de matricula és quan es controla que l'alumne vagi seguint el pla d'estudis. Amb la política de progressió i permanència se li imposa un ritme en aquesta consecució. Així doncs el llistat de possibles assignatures que un alumne es pot matricular en un any acadèmic concret depèn:

- Grau que cursa que determina la càrrega lectiva total i el pla d'estudis que ha de seguir
- Pla d'estudis determina l'oferta d'assignatures que té que realitzar obligatòriament o d'on pot escollir les assignatures optatives.
- Política de progressió i permanència que limita les opcions d'assignatures als estudiants que no superen suficients assignatures.
- Les notes que determinen quines i quantes assignatures s'han superat i quines no i per tant s'han de repetir.

Si volem implantar un pla d'estudis en el nostre programa de matricula cal incorporar totes aquestes polítiques al procés de matricula i al entrar notes. És a dir, la matricula no pot ser només assignatures que existeixen en el fitxer d'assignatures sinó que han de ser les que toquen segons totes aquestes polítiques. Al entrar notes, també ens hauria de notificar si un estudiant ha complert tots els requisits per obtenir el grau, o si ha incomplert mínims per continuar el grau.

És obligatori que el programa controli que l'alumne es matriculi només d'assignatures reconegudes, però no es obligatori controlar res més (ni l'ordre en què ho fa, ni la quantitat, ni si ja ha acabat o no pot continuar ....). Qualsevol política que s'apliqui de pla d'estudis és opcional i comptarà com a nota extra (veure avaluació).

1.2.1 Pla d'estudis

Cal cursar les assignatures obligatòries del grau i un nombre d'optatives (considerar un total de 240 crèdits). L'oferta d'assignatures obligatòries i optatives ja està definida en el fitxer assignatures. Si aquest fitxer està ben definit, el programa només ha de comprovar que l'alumne es matricula algun moment de totes les obligatòries i de suficients optatives (d'entre l'oferta) per complir amb el pla d'estudis d'un grau determinat.

1.2.2 Progressió i permanència

És més complicat imposar una política de progressió i permanència. En aquest cas es limita a l'estudiant quin subconjunt d'assignatures es pot matricular en funció de les que ha de fer i de les que ha superat.

Podem agafar com a referència la política de progressió i permanència de la pompeu que la podeu consultar en qualsevol dels vostres plans d'estudis, i que es mostra a continuació:

> **Continuació d'estudis**
> Cal superar el 50% dels crèdits d'assignatures de formació bàsica i obligatòries de primer curs
> Crèdits a superar: 30
>
> **Progressió en els estudis**
> Accés a segon curs
> Cal superar el 66% dels crèdits de formació bàsica i obligatòries de primer curs.
> Crèdits a superar: 39
>
> Accés a tercer curs
> Cal superar tot el primer curs i el 66% dels crèdits de formació bàsica i obligatòries de segon curs.
> Crèdits a superar: 39
>
> Accés a quart curs
> Cal superar tot el segon curs i el 66% dels crèdits obligatoris i optatius de tercer curs.
> Crèdits a superar: 39

Això vol dir:

- Un expedient d'un alumne de primer que no superi 30 crèdits al entrar notes el primer any, no supera la regla de continuació d'estudis i no es pot tornar a matricular.
- Un alumne només es pot matricular d'assignatures de segon si té superats com a mínim 39 crèdits de primer. Sinó només es pot matricular d'assignatures de primer.
- Un alumne només es pot matricular d'assignatures de tercer si té superat tot primer i com a mínim 39 crèdits de segon. Sinó només es pot matricular d'assignatures que no té superades.
- Un alumne només es pot matricular d'assignatures de quart curs si té superat totes les de segon i com a mínim 39 crèdits d'assignatures de tercer. Sinó només es pot matricular d'assignatures no superades.

1.2.3 Graduació

Un cop l'alumne ha superat tot els crèdits obligatoris i suficients dels crèdits optatius se li pot expedir el títol. Per tant, al entrar notes, es pot verificar si l'alumne ha acabat la seva titulació i fer que el programa informi d'aquest fet en aquell moment.

També podria ser possible verificar en qualsevol moment si l'alumne pot acabar el curs acadèmic en curs si supera tots els crèdits verificant si està matriculat de tots els crèdits que li falten per acabar. En aquest cas es proposa en l'opció del menú de mostrar l'estat de l'expedient, però també podria ser possible mostrar aquesta informació al entrar la matricula.

1.3 Format dels fitxers

Aquest programa estarà limitat en funcionalitat i no serà un sistema complert de base de dades. En el nostre cas el programa s'engegarà i apagarà quan es vulgui. Per no tenir que entrar manualment les dades cada cop mantindrà les dades en fitxers text. Al tornar a començar el programa, aquests fitxers s'han de tornar a carregar per recuperar l'estat d'abans de parar. Com l'objectiu final és generar fitxers expedients d'alumne fem servir aquests mateixos fitxers per guardar i carregar les dades al programa. A més de les dades dels expedients, ens cal l'oferta d'assignatures de l'escola. Aquesta es guardarà i carregarà d'un fitxer assignatures. Així el programa ha de ser una eina per manipular aquests fitxers de forma automàtica imposant les regles pertinents d'operació.

Tenim dos tipus de fitxer text:

- Assignatures: Llistat d'un conjunt d'assignatures ofertades per l'escola (model adjunt: asgs-get.txt). En principi podria haver-hi més d'un fitxer assignatures però per simplificar suposarem que només n'hi pot haver un.

- Expedients: Expedient oficial d'un alumne (model adjunt: expedient_12345.txt). Hi ha tants fitxers com expedients d'alumne tingui registrats l'escola que són els matriculats actualment i els que ja han acabat o han deixat els estudis.

Ressaltar per tant, que tots aquests fitxers són tan per l'entrada de dades com per la sortida de dades del programa. El format concret de cada un d'aquests fitxers es detalla més endavant.

Remarcar que els fitxers són text, per tant l'usuari pot manipular les dades en el fitxer directament amb un editor de text i per tant al carregar-los al programa ha d'incorporar les verificacions pertinents per la consistència de les dades i robustesa del programa.

1.3.1 Fitxer assignatures

El fitxers assignatures és el fitxer que conté el llistats d'assignatures que s'ofereixen a una escola. És un únic fitxer que té un nom prefixat configurable (considerem asgs.txt). S'adjunta el fitxer asgs-get.txt que serveix de model del format d'assignatures (conté totes les assignatures del grau d'enginyeria telemàtica de l'escola). De totes maneres es recomana fer proves amb un fitxer reduït i ben dissenyat a partir del disseny de pla de proves.

El format del fitxer model proporcionat segueix les següents regles:

A l'inici de fitxer hi pot haver (o no) una **capçalera de fitxer** que conté text per comentar el contingut del fitxer. Aquesta capçalera la defineix i delimita perquè cada línia ha de tenir "//" a l'inici de la línia. Cal guardar aquest text amb el fitxer per poder reproduir el fitxer amb les mateixes dades quan es guarda més dades.

Seguit de la capçalera de fitxer, hi ha la **capçalera de les assignatures**, que és una línia amb noms que identifiquen cada columna. Aquesta línia no va precedida per "//".

A continuació ve el llistat d'assignatures on hi ha una fila per cada assignatura i cada columna són els valors de les noms de la capçalera anterior i que venen definits com :

- **Num**: Número de l'assignatura en ordre d'aparició. Aquest valor només serveix per veure quantes assignatures conté el fitxer. Aquesta columna es genera al gravar al fitxer. En la lectura del fitxer el número hi és però no es verifica ni s'utilitza (es salta). Al gravar al fitxer, aquesta columna sí que indica el número (de línia) d'assignatures (quantitat).
- **Tipus**: indica el tipus d'assignatura que pot ser (B) obligatòria o (O) optativa.
- **Curs**: Indica el curs que s'ha de cursar aquesta assignatura. Aquest valor ha de ser entre 1 i un màxim igual al nombre d'anys del grau (suposar 4). Només les assignatures obligatòries tenen un curs assignat i serveix per controlar el progrés i permanència de l'alumne. Les assignatures optatives no estan assignades a cap any en concret, i al fitxer això s'indica amb un "-".
- **Trim**: Indica el trimestre que s'ofereix l'assignatura.
- **Codi Asg**: El codi d'assignatura és un número d'un nombre prefixat de dígits (suposar 5) que identifica unívocament l'assignatura entre totes les assignatures ofertades en la mateixa escola. Totes les assignatures dels diferents graus tenen un número diferent. I de fet la mateixa assignatura ofertada en graus diferents té el mateix nom però codi diferent. Això permet, per exemple, que la mateixa assignatura pugui ser obligatòria en un grau i optativa en un altre.
- **Crèdits**: Quantitat de crèdits de l'assignatura.
- **Nom Asg**: Nom de l'assignatura que es un cadena de caràcters (sense espais en blanc) amb un màxim de caràcters totals (es pot considerar 100 caràcters). Si són varies paraules cal posar un separador diferent d'espai en blanc (el caràcter en concret ha de ser configurable, exemple Fonaments_programació, i ha de ser el mateix separador per tots els noms de les dades).

Entre les files pot haver-hi línies en blanc però no comentaris ni altre text. Al final també pot haver-hi línies en blanc. Però totes aquestes línies en blanc es perdran al generar el fitxer assignatures amb el programa.

1.3.2 Fitxer expedient

Els fitxers de tipus expedients són tots aquells fitxers que s'anomenen seguint el següent format de nom:

expedient_<nia>.txt

on NIA és el codi identificador de l'alumne d'aquest expedient.

S'adjunta el fitxer expedient_12345.txt que serveix de model del format d'assignatures. Podem descriure el format amb les següents regles.

Es poden identificar quatre àrees en el fitxer:

0. Capçalera del fitxer
1. Informació general de l'expedient
2. Llistat d'assignatures
3. Resum de l'expedient

A l'inici de fitxer hi pot haver una ***capçalera de fitxer*** que conté text per comentar el contingut del fitxer. Aquesta capçalera està definida igual que en el fitxer assignatures. Igualment, cal guardar aquest text amb el fitxer per poder reproduir el fitxer amb les mateixes dades quan el programa generi el fitxer. No poden aparèixer línies de comentari senceres en cap altre lloc del fitxer (però si a final de línia, veure llistat d'assignatures).

Seguit de la capçalera de fitxer hi ha l'***informació general*** de l'expedient que té un text i format específic que cal seguir estrictament:

- Nom: Nom de l'alumne definit igual que el nom de l'assignatura però amb un màxim de càracter configurable diferent (considerar 25 caràcters).
- Primer Cognom: Primer cognom de l'alumne amb la mateixa definició que el nom.
- Segon Cognom: Segon cognom de l'alumne amb la mateixa definició que el nom.
- NIA: Número d'identificació de l'alumne que és un número d'un nombre exacte de dígits (considerar 5 dígits). Això vol dir que si el nombre és de menys dígits cal posar tants zeros davant com calgui per tenir el nombre de dígits adequats.

A continuació hi ha el llistat d'assignatures amb una línia de capçalera que dóna nom a cada columna. Les columnes són:

- Curs: Curs de l'assignatura, que és un natural entre 1 i un màxim (considerar 4). Les assignatures optatives no estan assignades a un curs i s'hi posa un valor numèric fora de rang configurable (considerar 9).
- CodiAsg: Codi de l'assignatura, que ja hem definit el format.
- NomAsg: Nom de l'assignatura, que ja hem definit el format.
- Conv: Convocatòria actual de l'assignatura que és la convocatòria de l'última matricula de l'assignatura. És un número entre 1 i un màxim de convocatòries que disposa un alumne per aprovar les assignatures (considerar 5).
- Nota: Nota de l'assignatura que és un valor decimal entre 0..10, amb com a màxim un decimal i separat amb coma (Exemple: 6,7). Si una assignatura encara no té nota cal indicar un valor determinat en aquesta columna (nota blanc que pot ser un valor negatiu)
- Crèdits: Número de crèdits de l'assignatura.
- Comentari: Al final de cada línia d'assignatura pot haver-hi un comentari que s'identifica amb "//" davant del text. Aquest comentari s'ha de guardar tal qual a memòria per poder-lo escriure quan es torna a generar l'expedient. Aquest text només pot estar a final de línia, i és tot el text des dels símbol "//" fins a final de línia, i va assignat a l'assignatura corresponent.

Aquest llistat d'assignatures són totes les assignatures que l'alumne s'ha matriculat al grau corresponent durant tot el període d'estudis.

Finalment, hi ha el ***resum de l'expedient*** que aporta el resum de notes amb un format concret:

- Una capçalera del resum que diu "Resum de l'expedient" subratllat
- Nombre total de crèdits matriculats: la suma de tots els crèdits que apareixen a la taula (superats o no).
- Nombre total de crèdits superats: suma dels crèdits de totes les assignatures amb nota que supera el llindar d'aprovat (considerar 5).
- Nota promig expedient acadèmic: promig de totes les notes de les assignatures superades ponderades pel nombre de crèdits per assignatura.

Consideracions importants del format de l'expedient oficial (i per tant del contingut d'aquest fitxer):

a. El llistat d'assignatures en l'expedient oficial ha d'estar ordenat per ordre creixent de curs, primer mostrant les obligatòries i al final les optatives (que no tenen curs assignat). Entre el mateix curs han d'estar ordenades en ordre creixent de codi d'assignatura. Al generar un fitxer des de programa cal assegurar que està ordenat, però al carregar-lo no se suposa que les assignatures hagin d'estar ordenades perquè potser l'usuari n'afegeix directament al fitxer text i no té perquè posar-ho en ordre.
b. No hi pot haver línies en blanc al mig de la capçalera, ni entre les línies d'identificació de l'assignatura, ni al mig de la taula, però si abans i després de cada bloc.
c. Després de cada línia de notes en la taula pot haver-hi un comentari que comença amb "//" i va fins a final de línia. Aquest comentari s'ha de guardar amb la nota i incloure'l a l'expedient imprès.
d. El nom del fitxer d'entrada d'expedient ha de ser coherent amb el seu contingut, per tant el NIA al nom del fitxer ha de ser el NIA que s'indica dins. Si no ho és cal informar a l'usuari i demanar si cal continuar.
e. Una nota No Presentat s'identifica amb un valor numèric no vàlid configurable (considerar -1).
f. Una assignatura sense nota (nota blanc) s'identifica amb un valor numèric no vàlid configurable (considerar -2). Això garanteix que en totes les línies hi ha en qualsevol cas tants valors com columnes.
g. No es pot assumir que el resum hi sigui sempre (quan l'usuari genera el fitxer a ma). Quan hi és cal llegir-lo i verificar si és coherent amb les dades, si no ho és cal informar a l'usuari que hi ha una inconsistència, però continuar operant amb el resum de l'expedient correctament calculat.

1.3.3 Organització de fitxers

Per l'organització dels fitxers, considerem que el programa busca els fitxers expedients i assignatures en un sub-directori anomenat **dades** situat al directori on està l'executable (.exe). Aquest subdirectori (**dades**) inclou tots els fitxers d'expedient (un per alumne), i un únic fitxer d'**assignatures**. No pot contenir altres fitxers.

1.4 Operacions i dades

Consideracions a tenir en compte de les operacions i dades :

1. El NIA és un valor únic que identifica l'alumne. Cal assegurar que tots els alumnes estan carregats quan es genera un expedient nou per poder assegurar que el nou NIA no existeix ni es repeteix.
2. En principi no hi ha una llista d'alumnes existents sinó que la llista la determina els fitxers expedient en el directori dades. Això vol dir que qualsevol nom o codi d'alumne que s'entri serà acceptat sempre i quan segueixi el format establert per la definició de la dada.
3. El NIA identifica l'alumne i per tant no es pot tenir dos alumnes amb el mateix NIA. Si es dóna el cas el segon alumne que s'intenta entrar o modificar no es pot admetre i cal que el programa ho notifiqui.
4. Considerem que es pot donar el cas que dues persones es diguin igual. Per tant s'accepten alumnes amb NIA diferent i nom i cognoms idèntics.
5. Un alumne té dret a un nombre màxim de convocatòries. S'ha de comprovar que no es dóna un número de convocatòria més gran a aquest valor. Si es dóna cal notificar i no acceptar el valor entrat i deixar-lo buit.
6. El programa pot considerar límits establerts en les dimensions dels vectors de dades que calgui, sempre i quan el programa avorti quan s'intenti sobrepassar aquest valor.
7. El codi d'una assignatura l'identifica i per tant en cap cas dues assignatures poden tenir el mateix codi. Si es dóna el cas no s'admet la segona i es notifica.
8. Dues assignatures poden tenir el mateix nom però codi d'assignatura diferent, com és el cas de la mateixa assignatura ofertada a varis graus. Però si les dues assignatures pertanyen al mateix grau no poden tenir el mateix nom. Aquest és el cas ja que la pràctica no contempla el grau en les dades. Si es dóna el cas no s'admet la segona i es notifica.
9. L'ordre de les assignatures en el fitxer d'entrada, tant pel fitxer assignatures com fitxers expedients, no té perquè estar ordenat. Això permet afegir manualment una nova assignatura a un fitxer ja existent sense mirar on s'afegeix. Però si que és imprescindible que els fitxers generats pel programa tinguin les assignatures ordenades segons el criteri establert.
10. Cap fitxer no es gravarà si la versió actual és idèntica per tal de que la data del fitxer correspongui realment amb la data de modificació del seu contingut.
11. Sempre que s'identifica una inconsistència cal informar l'usuari i donar-li l'opció de no continuar l'operació que està fent. Per exemple si hi ha duplicitat de NIA cal donar l'opció de dir si deixar la informació de memòria, modificar el NIA amb les dades del fitxer o avortar l'operació de carregar de fitxer per resoldre el problema abans de tornar-ho intentar.

1.5 Robustesa

Hem ja dit que la robustesa vol dir que el programa ha de funcionar correctament en tots els casos possibles i no pot fallar en cap cas. Això vol dir que no pot haver imprevistos.

Evitar imprevistos vol dir garantir en tot moment que l'**estat del programa** és conegut i està previst com actuar en cada estat. L'estat del programa el controla el (bon) flux d'execució i ve influenciat pel valor de les constants internes i les dades entrades per l'usuari sigui manualment o via fitxer. Per tant el propi programa ha de verificar el seu estat sempre que se sap quin ha de ser, per notificar quan no és el què toca. Això detecta i aborda els errors d'implementació. Però també pot haver-hi els errors humans fets pels usuaris al entrar les dades sense seguir les normes estipulades. Per tant en la lectura de la dada hem de sempre assumir totes les possibilitats que l'usuari pugui entrar (i no només les que esperem) si volem tenir un programa robust. Si l'usuari no compleix les normes no ha de fallar el programa sinó que el programa li ha de notificar i demanar-li la dada altre cop (o permetre-li acabar l'opció que estava fent, per si no sàpigues quina dada donar).

Així tenim que la robustesa té les següents fases:

1. Identificar clarament els valors dels paràmetres o estat del programa acceptables
2. Identificar les maneres per les quals poden entrar al programa valors o situacions inconsistents
3. Integrar dins el programa
    a. Mecanismes per detectar aquests estats indeguts
    b. Notificar a l'usuari quan es detecten
    c. Implementar una solució al problema, que si no es troba és la parada controlada del programa

1.5.1 Imprevisions en l'estat del programa

Aquest programa pot tenir 4 possibles fonts d'errors que poden portar-lo a un estat incorrecte:

1. Configuració: El programa ha de tenir un conjunt de constants que han d'estar ben definides. Cada una ha de tenir clarament identificat quins són els possible valors que admet. Tot i així és possible que el programador o usuari del programa variï el valor a un que no compleixi les especificacions donades. (exemple: que el percentatges de pes de cada part de la nota no sumi 100%)
2. Entrada de dades manual: L'usuari al respondre les preguntes del menú pot no seguir les instruccions correctament per error (exemple entrar una lletra on es demana un número).
3. Entrada de dades via fitxer: La modificació incorrecte d'un fitxer via editor de text pot resultar en un fitxer que viola el format esperat pel programa (exemple: deixar una columna que sempre ha de tenir valor en buit).
4. Mal funcionament: és possible que es pensi que el programa funciona correctament i en realitat no sigui el cas.

Fer un programa robust vol dir abordar en detall totes les possibles situacions en les tres primeres fonts d'errors anteriors, i assegurar que la quarta no es pugui donar mai. Acceptant que la garantia 100% és impossible, el programa més robust és aquell que contempla més situacions impossibles i les aborda. Abordar una situació ha d'incloure almenys detectar el problema, notificar-lo, i avortar el programa controladament. Si a més s'implementa una forma de corregir la situació (amb la intervenció de l'usuari o no) encara millor.

Quan s'està treballant la implementació, incloure verificacions de funcionament ajuda molt al programador ja que és el propi programa que notifica de la seva inconsistència. Això és especialment útil quan un programa s'estén i la modificació (correcte o incorrecte) genera situacions abans prohibides. Integrar aquests controls és important per avançar ràpid i es deixa al vostre criteri com incorporar-ho.

2 Requeriments a complir per l'entrega

Es recomana llegir el document de guia de treball de la pràctica [P2-GT] el qual aborda el com treballar la pràctica i aporta mètode en el treball en equip i gestió de la feina i del temps. Aquests són elements cabdals en la programació i en adquirir experiència per anar abordant cada vegada conceptes i problemes més grans i complexes.

2.1 Disseny: Mapes Conceptuals i Pla de Proves

Abans de començar la implementació es demana construir els mapes conceptuals i pla de proves tal i com s'ha explicat a teoria i que es té d'exemple pel cas pràctic. Els mapes conceptuals del cas pràctic que us poden servir d'exemple estan a [CP-IDX], i el disseny del banc de proves de la funció notes a [CPv1] [T-VC]. Reviseu aquests documents i dissenyeu els mapes conceptuals i pla de proves d'aquest problema abans de programar-lo. Aquest material es demana en la pre-entrega.

Es demana que aporteu el banc de proves de :

1. Funció de carregar fitxer expedient aportant els fitxers exemples que cal per fer totes les proves identificades
2. Funció de carregar fitxer assignatures aportant els fitxers exemples que cal per fer totes les proves identificades
3. Funció de calcular nota promig de l'expedient: els fitxers expedients de proves han d'incloure totes les opcions de les proves que necessiti aquesta funció
4. Funció ordenar
5. (opció) Funció de l'estat d'expedient per calcular progressió d'estudis: els fitxers expedients de proves han d'incloure totes les opcions de les proves que necessiti aquesta funció

Cal aportar tants fitxers d'entrada d'expedients i la corresponents d'assignatures per incloure totes les proves que els hi cal comprovar totes les funcions. Segons com vagi les proves, pot ser que només un fitxer de cada sigui possible. Però és possible que calgui més d'un expedient o diferents fitxers d'assignatures per contemplar diferents opcions.

La preentrega cal que detalli la llista de consideracions a fer en una taula, i indiqui quin fitxer model aporta per contemplar aquesta prova. Per tant la preentrega contindrà part d'informe i els fitxers a executar, a més del disseny del programa representat gràficament amb els mapes conceptuals.

2.2 Robustesa

Un programa professional se li exigeix que sigui exhaustiu per ser robust. En aquesta pràctica **acotem la definició i implementació a 2 (o més) regles de robustesa de cada un dels 4 tipus d'imprevistos** definits en la secció 1.5.1. Cal escollir i justificar les més importants en funció del problema, errors més comuns que podeu identificar i la implementació en particular que feu. En l'informe s'espera una taula que indiqui la llista d'aquestes regles amb la justificació de perquè cada una d'elles i una explicació de com es surt de la incoherència i perquè aquesta solució i com s'ha implementat.

2.3 Cerca i ordenació

És important entendre els algorismes de cerca d'un element en un vector i l'algorisme d'ordenació del contingut d'un vector segons criteris. En aquest procés de comprensió és demana:

- Definir les funcions de cerca i ordenació com funcions separades amb els paràmetres ben definits

- Construir una traça que visualitzi l'operació d'aquests dos algorismes per separat que s'activi i desactivi amb una etiqueta tipus DEBUG.
- Comentar quin algorisme implementa en el codi i detallar el funcionament a l'informe mostrant un exemple de traça generat pel vostre codi i explicant tots els casos a considerar segons un disseny de banc de proves de la funció.

En el cas de la cerca la traça ha de visualitzar l'estat de l'algorisme (posició/ns de comparació,..) i el nombre de comparacions acumulat i en cada pas en la cerca puntual en qüestió. Cal veure almenys un cop l'estat de tot el vector ja que no varia en tot l'algorisme (es pot mostrar en cada pas si es vol o el subvector efectiu, decidiu el què us serveix per fer el seguiment i poder-lo explicar en detall). En el cas de l'ordenació la traça ha de mostrar el mateix que la cerca (tenint en compte que les operacions son comparacions i intercanvis) i en aquest cas sí que és imprescindible mostrar tot el vector en cada pas per veure com evoluciona amb els intercanvis.

2.4 Modularitat

Degut a la gran importància de la modularitat s'imposa elements concrets a l'entrega.

Respecte a la modularitat de fitxers s'imposa que el programa estigui separat en com a mínim els següents fitxers:

1. Llibreria entrada/sortida: Totes les funcions d'entrada i sortida de dades (de/a fitxer i entrada manual de dades)
2. Llibreria d'ordenació: Les funcions relacionades (d'alguna manera) en la cerca i ordenació de les dades.
3. Programa principal: Almenys la funció main i les funcions addicionals que calgui o tingui sentit.
4. Estructura de dades: .h amb les declaracions de les estructures de dades.
5. (optatiu) Llibreria grau: les funcions relacionades amb la definició del grau i polítiques de permanència.
6. Makefile: és imprescinbible aportar un makefile fet per vosaltres que executi el programa en línia de comanda. Encara que normalment feu servir un IDE i s'utilitzi el makefile que crea automàticament, cal saber construir un makefile simple correctament. No s'aceptarà el makefile automàtic.

Si creieu convenient separar-ho amb més fitxers dels anteriors, o diferent però amb almenys el mateix nombre de fitxers, podeu fer-ho raonant la vostra estructura de programa en l'informe.

Respecte a la modularitat funcional s'imposa que el programa tingui com a mínim les següents funcions:

- Funció d'inicialitzar
- Funció de cerca: que identifica un element dins un vector d'estructures i calcula el nombre d'operacions (comparacions)
- Funció d'ordenació : que ordena un vector d'elements segons un criteri i calcula el nombre d'operacions (comparacions i intercanvis)
- Funcions afegir i eliminar element a un vector d'estructures (d'alumnes/expedients o assignatures)
- Funcions menú i si cal submenús
- Almenys una funció de verificació de dades

Aquesta estructura en principi s'hauria de resultar de manera natural del treball en equip i comprensió del problema que s'aborda. No és important aplicar aquestes regles perquè s'imposin sinó entendre com funciona la modularitat i que realment se'n deriva una estructura similar. És a dir, el important de la modularitat no és tan sols saber fer servir les comandes sinó veure la seva utilitat en el treball en equip del dia a dia pel qual està definida. El document de guia de treball vol fer entendre l'objectiu perquè es posi en pràctica durant l'elaboració de la pràctica. Si es treballa amb mentalitat modular durant tot el projecte el codi resultant (estructura) ho reflecteix amb una naturalitat modular inherent.

2.5 Restriccions de programació

Per assegurar que s'adquireixen uns cert hàbits cabdals de programació s'imposa certes condicions que ha de complir el codi desenvolupat. No s'acceptarà ni es revisarà codi que no compleixi aquestes condicions. És important que agafeu com a hàbit aquestes condicions ja que la vostra velocitat de programació, detecció d'errors i treball en equip seran molt més ràpids i fluïts. Com més aviat ho apliqueu més ràpid avançareu en el vostre aprenentatge. No es tracta de polir el codi just abans de l'entrega per complir les condicions sinó d'aplicar-les inconscientment des del principi del programa.

Criteri	Definició	Excepcions / Comentaris (veure codi del cas pràctic [CD-CODI] per exemples en cada cas)
Variables globals	Les variables globals estan prohibides i no hi pot haver cap variable global en el programa.	No s'accepta cap excepció
Números	Els números estan prohibits dins el codi. Cal assignar a un número un identificador que el representi segons el seu ús. Un mateix número pot servir per vàries coses però cal definir dos identificadors diferents quan representin conceptes diferents del mateix valor que potser en un futur poden tenir valors diferents.	El 0 i el 1 poden ser números que no els hi calgui identificador, en segons quines operacions. Exemples: sumar 1 per incrementar no cal identificador (però cal considerar si una extensió del programa podria considerar un altre increment, llavors si cal). El valor 0 per inicialitzar variables,

		no cal identificador. Però l'opció 0 i 1 del menú si cal donar-els-hi un identificador.
Capçalera de comentari a cada fitxer i indentació	Cal aportar una capçalera de comentari en cada fitxer del programa indicant programa, autors, data, funcionalitat del fitxer i informació que calgui per entendre el fitxer. També cal assegurar que el codi està correctament indentat.	Excepcions cap
Comentari a estructura de dades i variables importants	És imprescindible que cada declaració i estructura de dades i camp tingui el seu comentari que expliqui què és. També les variables importants (que serien globals si no fossin prohibides) que gestionen les dades principals del programa.	No hi ha excepcions pel què fa el fitxer capçalera, qualsevol declaració cal comentar-la. El comentari és imprescindible per les variables importants (que podrien ser globals) del programa i la resta és a criteri del programador decidir si és variable irrellevant o important per comentar-la (exemple: variables i,j,k temporals per recórrer un bucle no cal comentar-la, però potser sí si té un nom concret).
Comentari de funcions	És imprescindible que cada funció tingui el seu comentari que expliqui què fa, els paràmetres que passa i quin valor retorna i el què calgui explicar del com ho fa	No s'admet cap excepció, totes les funcions tenen que aportar un comentari.
Duplicitat de codi	Està prohibit tenir trossos de codi iguals (o només diferents pels noms de variables o camps...) en diferents llocs del programa. Cal definir una funció i parametritzar-la correctament per poder-la fer servir en tot els llocs on es fa el mateix però amb dades diferents.	Hi ha casos que amb els coneixements actuals no sabem parametritzar (cal punters a funcions o punters void). Exemples: ordenar ascendent o descendentment, o ordenar segons un camp diferent d'una estructura. Aquests casos s'acceptaria la mateixa funció operant amb comparació diferent o amb el camp diferent però tot el codi igual.

2.6 Verificació: Banc de proves i Traces

La verificació passa per executar tots els casos possibles (per disseny i segons la implementació) amb la conseqüent comprovació del correcte funcionament. La comprovació de resultats es pot fer mirant els valors finals, però la procedural requereix d'un seguiment de les traces en cada prova.

Cada prova del banc de proves ha d'indicar clarament els valors de configuració de les constants internes (o tenir un fitxer .h de referència), els valors d'entrada (o tenir els fitxers de dades de referència) i recaptar la sortida relacionada amb l'entrada (per tant tenir els fitxers de sortida per cada prova. La verificació passa per observar la sortida i interpretar-la per comprovar que és correcte.

S'espera que l'entrega aporti una taula de proves, amb els fitxers adjunts de configuració, d'entrada (definits en la preentrega però potser modificats) i sortida (que produeix el programa) corresponent. La sortida de l'execució serà els fitxers que produeix però també pot produir missatges i traces el fitxer. Per tant al explicar el funcionament del programa cal els fitxers expedients de sortida per la verificació de resultats, i les traces per la verificació procedural.

Es demana aportar una comprovació procedural de com a mínim les següents funcionalitats:

1. Funció d'ordenació : cal aportar una traça que mostri els passos de l'ordenació i com va variant el contingut en cada posició i com es va acumulant el nombre d'operacions fetes.
2. Funció cerca : cal aportar una traça que mostri com avança la cerca fins arribar a la posició adequada en tots els casos possibles a considerar segons l'algorisme de cerca implementat.

L'informe ha d'incloure les traces tretes del programa i explicar el detall (línia per línia) de la traça per ressaltar tots els casos a comprovar a l'hora de decidir les proves en el banc de proves.

2.7 Informe Final

L'entrega ha d'anar acompanyat d'un informe per incloure la documentació indicada anteriorment. Disposeu d'un document plantilla per l'informe on està recopilat el què cal incloure a l'informe final [P2-INF]. Podeu fer servir aquesta plantilla o fer un altre document, però cal incloure tota aquesta informació a l'informe que s'entregui.

3 Material de Suport

Disposeu del següent material a l'aula global que us pot ajudar a dissenyar i programar el vostra programa

- Cas pràctic de càlcul de notes:
  - [CP-CODI] Codi: disposeu del codi organitzat en vàries versions per anar introduint funcionalitats progressivament (veure secció codi a la zona de teoria)

- [CPv1]: Transparències Cas Pràctic v1 que fa el repàs complert de la versió 1 com a repàs dels conceptes del segon trimestre
- [CP-MC] Mapes conceptuals: Revisar els mapes conceptuals del cas pràctic per elaborar el vostre abans de programar i així planificar l'estructura del vostre programa i tenir mètode a l'hora de distribuir la feina entre els membres de l'equip. Els mapes estan en les transparències d'índex del cas pràctic [CP-IDX]
- [CP-IDX] Transparències d'índex del càs pràctic (aula global)

- [P2-GT] P2: Programa de Gestió de Matrícula i Expedients de Notes, **Guia de Treball** (aula global)... en construcció
- [P2-INF] P2: Programa de Gestió de Matrícula i Expedients de Notes, **Informe** (aula global) ... en construcció
- Transparències de teoria (aula global)
  - [T-VC] Verificació de Codi

És imprescindible que tingueu un manual de c a ma al programar. Teniu un apartat de recursos a l'aula global amb links a tutorials útils i molt pràctics. Entre ells destacar pel què fa la pràctica:

- Plantilla de c (2 pàgines): http://web.stanford.edu/~nwh/cme212/files/c-refcard.pdf
- Tutorial on-line de programació de c : http://www.tutorialspoint.com/cprogramming
- Tutorial on-line de les llibreries de c : http://www.tutorialspoint.com/c_standard_library

4 Avaluació

Criteris d'avaluació s'apliquen si l'entrega passa els criteris d'acceptació:

		Sub-criteris				
	Concepte	D 35%	F 30%	C 15%	I 20%	Ponderació
1	Estructura de dades i variables					10%
2	Lectura de fitxer					10%
3	Escriptura a fitxer					5%
4	Entrada manual de les dades					5%
5	Computació de l'expedient					5%
6	Cerca					10%
7	Ordenació					10%
8	Robustesa del programa					10%
9	Modularitat del codi (llibreries+makefile)					10%
10	Mapes conceptuals: estructura de programa					10%
11	Verificació: Pla de proves: disseny i implementació					10%
12	Verificació: traces (execució)					5%
13	Informe	No aplica				Transversal (20%)
14	Pla d'estudis[1]					15%
15	Informe part opcional[2]	No aplica				Transversal 2%
	[1] Opcional: Afegeix un màxim del 15% a la nota final d'aquesta pràctica (es pot tenir 11,5/10) però aquest increment de nota s'afegeix un cop les condicions mínimes d'aprovar pràctiques i l'assignatura es compleix. Està pensat pels alumnes que vulguin anar més enllà en les habilitats a adquirir i no per substituir les habilitats mínimes i principals per aprovar. [2] Cal que l'informe documenti correctament la part opcional si es presenta. No cal que sigui un informe diferent però sí que cada una de les preguntes en l'informe cal indicar si s'ha fet la part opcional per aquest apartat i aportar la informació rellevant d'aquesta part en la pregunta.					

La definició dels criteris anteriors és la següent:

1. **Estructura de dades i variables**: El disseny de l'estructura de dades, i les variables declarades són correctes, simples i adequades al problema i admetre possibles extensions.
2. **Lectura de fitxer**: El programa accepta el format del fitxer d'entrada indicat amb totes les seves variants i carrega correctament les dades
3. **Escriptura a fitxer**: El programa genera un fitxer de sortida amb el nom de fitxer i format correctes i amb les dades correctes.
4. **Entrada manual de les dades**: El programa demana correctament les dades necessàries i afegeix correctament les dades a l'estructura de dades per utilitzar per la resta del programa.
5. **Computació de l'expedient**: Les funcionalitats que cal aplicar a les notes base per construir l'expedient (càlcul de nota promig, nombre de crèdits superats,...) són correctes
6. **Cerca**: La funcionalitat de cerca implementada en el programa està ben dissenyada i funciona correctament.
7. **Ordenació**: La funcionalitat d'ordenació demanada a l'enunciat és correcte.
8. **Robustesa**: El programa adreça la robustesa correctament i té implementat comprovacions internes de dades com es demana en la secció 2.2.

9. **Modularitat del codi**: El programa està separat en fitxers codi de manera coherent. S'aporta un makefile propi que funciona.
10. **Mapes conceptuals**: S'aporten mapes conceptuals i corresponen al codi entregat i s'explica el disseny dels mapes per explicar i justificar l'estructura del programa.
11. **Verificació Pla de proves**: S'aporta el disseny d'un banc de proves considerant un mètode exhaustiu d'identificar casos i agrupar-los en proves. Incorpora codi per la creació de les traces.
12. **Verificació Traces**: Execució del pla de proves comentades amb les traces i fitxers de sortida rellevants.
13. **Informe**: L'informe és útil i informatiu aportant la informació necessària i suficient explicant de forma clara, concisa i precisa els aspectes que calen per entendre el disseny, implementació, verificació i documentació del programa (tal i com es demana en l'enunciat i plantilla d'informe). Imprescindible aportar els mapes conceptuals del programa. És un element transversal sobre tots els conceptes i per tant s'han de tractar tots els aspectes d'avaluació i s'avaluarà proporcionalment.
14. (opcional) **Pla d'estudis**: Implementació de la part opcional de la pràctica (incrementar nota proporcionalment al fet si no es fa tot.
15. (opcional) **Informe de la part opcional**: L'informe ha d'explicar la part opcional si es presenta. Es pot posar un apart separat en l'informe per la part opcional (no ha de ser un informe separat).

Cada un dels criteris principals es valoren respecte els 5 subcriteris transversals:

1. D – **Disseny** (35%): Aquest criteri té un bon disseny respecte la divisió de codi en funcions, paràmetres que es passen entre funcions, variables a declarar.....
2. F – **Funciona** (30%) : Les prestacions del criteri funcionen i fan el què han de fer
3. C – **Codi Entenedor** (15%): El codi corresponent a aquest criteri és entenedor (noms representatius d'identificadors, comentat adequadament, indentació,....) i per tant el codi és fàcil d'entendre.
4. I - **Informe** (20%): La informació en l'informe és útil i hi ha d'haver els raonaments necessaris i suficient per justificar el què i perquè de cada concepte a avaluar (disseny, implementació, i verificació). Aquest 20% és el mateix 20% de l'informe.
5. V – **Verificació**: Hi ha mecanismes implementats per verificar les prestacions per fer traces per activar i desactivar i proves realitzades. Hi ha els resultats de les proves que cal fer: inputs amb els seus outputs. Aquest criteri és transversal però s'avalua per separat en els criteris 11 i 12, per limitar-ho a només els punts concrets limitats i no a l'aplicació sencera.

5 Instruccions per l'entrega

5.1 Grups de treball

Com es vol treballar la modularitat efectivament amb treball en equip, la pràctica s'ha de fer en grups de 2. Només en casos excepcionals s'acceptarà fer la pràctica sol i cal aprovació del professors per fer-ho. No s'acceptarà cap pràctica feta sol si no s'ha acordat prèviament.

Els grups de pràctiques s'han de posar a treballar tan aviat com es pugui a partir del moment que es publica l'enunciat. L'enunciat es publica molt abans d'haver fet tota la matèria que incorpora la pràctica però el disseny i una part d'implementació ja es pot anar elaborant. És per això que es posa una data límit de registre dels grups.

Per facilitar la gestió dels grups i trobar companys de grup, cal registrar el grup en el google doc següent. D'aquí obteniu una lletra que identifica el vostre grup i que cal fer servir en les vostres entregues. Si no teniu company d'equip, poseu-vos a la llista com únic membre, i contacteu els altres alumnes que estiguin també en la vostra situació.

La data límit per registrar el vostre grup per l'entrega d'aquesta pràctica és: xxxx. Dins d'aquest termini cal fer la petició de fer la pràctica sol i s'estudiarà el cas. No s'acceptarà sol·licituds de fer-ho sol fora d'aquest termini. Es podrà canviar de grup per la següent pràctica si es vol.

S'espera que els dos membres del grup participin activament en la feina de manera regular. Les situacions personals són diferents per cada persona i a vegades gestionar la diversitat és difícil. Si teniu dificultat en el treball d'equip demaneu ajuda als professors, que estem també per ajudar en aquests aspectes. No cal esperar a tenir problemes greus per consultar el funcionament del grup. Si espereu massa ja es pot tenir implicacions irreversibles, i es tracta d'evitar i aprendre a gestionar aquestes situacions.

5.2 Pre-entrega

La pre-entrega serveix com hem dit abans per obligar a pensar abans de programar, però també serveix per veure el progrés dels grups. La preentrega té una data límit perquè els grups tinguin una data per superar una fita intermitja de treball. Aquesta pre-entrega es pot fer qualsevol moment abans de la data límit i com més aviat més temps té el grup d'avançar cap al següent pas.

La pre-entrega no es corregeix i per tant no es posa nota. És responsabilitat dels alumnes preguntar als professors en les hores de consulta qualsevol dubte que tinguin sobre el seu disseny (o sobre qualsevol tema). Els professors consultaran les pre-entregues quan sigui necessari.

La pre-entrega ha d'incloure:

1. Mapa conceptual del disseny del programa (que seran varis dibuixos de blocs tants com mòduls) i breu explicació del mapa. Veure
2. Disseny del banc de proves:
    a. Determinar quin o quins fitxers d'entrada d'expedients i assignatures cal fer servir per verificar tots els possibles casos:
        i. Quants fitxers assignatures diferents calen? Què han de tenir diferent cada un? quin llistat *mínim* d'assignatures cal incloure en cada fitxer i amb quins valors per verificar tots els casos?
        ii. Quants fitxers expedient diferents calen? Quantes assignatures tenen que estar matriculats cada un i quins valors tenen que tenir assignats? Quins altres valors han de tenir en la resta d'informació?
        iii. Com verifiquem que el càlcul del promig de notes d'expedient és correcte i quins casos cal contemplar i com es tradueix en els fitxers anteriors?
    b. Fer un llistat de totes les possibles opcions que l'usuari pot entrar (correctes i incorrectes) en el menú d'opcions i que el programa ha de detectar i actuar correctament.

5.3 Entrega final

- Cal fer servir l'identificador (codi_grup) de grup per fer l'entrega.
- Crear un **directori** anomenat FP-P2-G<codi_grup> que inclourà tot els fitxers per l'entrega. Exemple el grup A anomenarà el directori com: FP-P2-GA.
- Entregar l'informe en format pdf i amb nom de fitxer: FP-P2-informe-G<codi_grup>.pdf. Exemple el grup A anomenarà el fitxer FP-P2-informe-GA.pdf. Posar una còpia del fitxer al directori d'entrega.
- Crear un subdirectori **codi** on es posa tot el codi c del programa (inclòs el makefile propi). Aquest directori ha de contenir el subdirectori **dades** d'on llegeix els fitxers tal i com es diu a l'enunciat.
- Crear un subdirectori **proves** que inclogui els fitxers d'entrada i els seus corresponents fitxers de sortida que demostrin el correcte funcionament del programa, també fitxers de sortida que inclogui entrada manual de dades (i combinacions). Aportar també els fitxers entremitjos que generi el programa en aquestes proves. Anomenar els fitxers adequadament perquè es pugui identificar quins fitxers formen part d'una mateixa prova. Comentar el contingut d'aquest directori adequadament a l'apartat de verificació de codi de l'informe. Aquest directori proves inclou totes les proves possible per verificar el programa mentre que el directori dades només conté una prova.
- Compactar el directori de l'entrega amb .zip o .rar i entrar el fitxer compactat a l'entrega de la pràctica 2 a l'aula global dins la data límit.