Projecte: Dames

Metodologia de la Programació

Curs 2024 - 2025

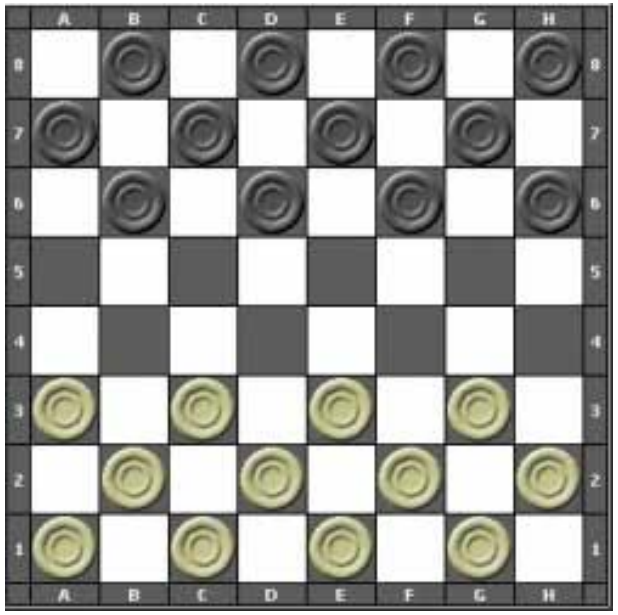
Objectius de la sessió

1. Presentació i organització del projecte
2. Explicar les tasques a realitzar a la primera versió del projecte

Organització del projecte

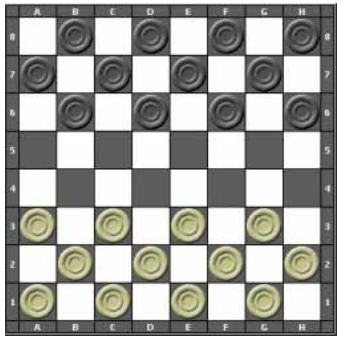
- **Grups de 2 persones:** hem obert al Campus Virtual una inscripció a grups perquè pugueu indicar amb qui fareu el projecte.
  - Podeu fer el projecte amb algú d'un altre grup de classe.
  - Apunteu-vos a un dels subgrups corresponents al vostre grup de classe.
  - La tutorització i avaluació del projecte la farà, en principi, el professor/a del vostre grup de classe. En alguns casos, però, pot no ser així, per equilibrar els grups de projecte assignats a cada professor/a.
- El projecte és part del **treball autònom** de l'assignatura. Durant les hores de classe dedicarem algunes sessions a seguiment i avaluació del treball.

Objectiu del projecte



Disseny i desenvolupament d'una versió completament funcional del joc de les **dames**, posant en pràctica els conceptes que anem explicant a les sessions de classe.

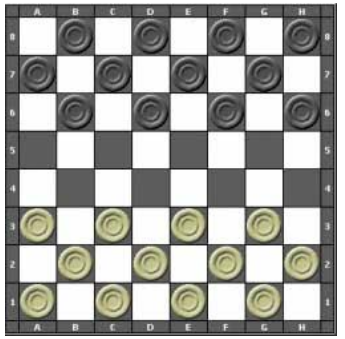
El joc de les dames



Regles del joc

- El joc consta d'un tauler quadrat de dimensions 8 x 8, alternant quadrats blancs i negres.
- Cada jugador disposa de 12 peces d'un mateix color (un jugador blanc i l'altre negre) que al principi de la partida es col·loquen en les caselles negres de les tres files més properes als marges del tauler de cada jugador.
- Es juga per torns alterns. Comença a jugar qui té les fitxes blanques. En el seu torn cada jugador mou una peça pròpia.
- Les peces es mouen (quan no mengen) una posició endavant (mai cap enrere) en diagonal a la dreta o a l'esquerra, a una posició adjacent buida.
- Es pot matar una peça contrària quan està en una posició adjacent en diagonal a la dreta o a l'esquerra i la següent posició adjacent continuant la diagonal està buida. En aquest cas, la peça es mou saltant la peça contrària fins a la posició adjacent buida i la peça contrària es retira del tauler.
- Es poden matar vàries peces contràries en un mateix moviment. Si després del primer salt per matar una peça contrària, la peça pot seguir avançant matant altres peces ho pot fer.

El joc de les dames



Regles del joc

- Si pot, el jugador està obligat a matar. Davant de diverses opcions, està obligat a matar el màxim de peces possibles. Davant de diverses opcions en què es mata la mateixa quantitat de peces, cal triar el moviment en què es mati el màxim de dames. Si un jugador no fa la captura màxima se li *bufa* (es retira del tauler) la peça amb què podia matar més. El fet de *bufar* una peça no constitueix moviment. O sigui, després de *bufar* una peça contrària fem el nostre moviment normal.
- Quan una peça arriba a la frontera contrària, es converteix en una dama. La dama té les següents particularitats respecte els seus moviments:
  - Es pot desplaçar cap endavant i cap enrere el nombre de posicions buides que vulgui
  - En el cas de que, en el seu desplaçament, es trobi una fitxa del color contrari, la pot matar, sempre que la posició següent la fitxa contrària estigui buida. La posició final de la dama serà a la posició adjacent de la peça capturada.
  - Les dames també poden encadenar varis moviments que permetin matar vàries fitxes contràries.
- La partida s'acaba quan es compleix algun dels següents casos:
  - Un jugador es queda sense peces sobre el tauler.
  - Si quan arriba el torn d'un jugador no pot moure, ja que totes les peces que li resten en joc estan bloquejades. En aquest cas, perd a qui li correspon el proper moviment

Funcionalitats del projecte

El vostre programa final haurà de permetre:

- **Inicialitzar una partida** des de zero o a partir d'un **estat inicial** guardat en un fitxer de text.
- **Jugar la partida** a partir de l'estat inicial. A cada **torn** d'un jugador:
  - **Seleccionar una fitxa** per moure.
  - Determinar tots els **moviments vàlids** que pot fer la fitxa.
  - **Moure** la **fitxa** a una nova posició dins dels moviments vàlids.
  - **Actualitzar** l'estat del **tauler** tenint en compte si es mata alguna fitxa del contrari i si cal "*bufar*" alguna fitxa del jugador actual perquè no ha matat quan ho podia fer.
  - Si la fitxa arriba al final del tauler, **convertir** la peça en una **dama**.
  - Detectar si s'ha arribat al **final de la partida**.
- **Guardar** tots els **moviments** que es fan durant la partida en un fitxer.
- **Reproduir** una **partida** prèviament jugada executant els moviments guardats en un fitxer.

Podreu afegir **funcionalitats extres**, si voleu, que comptaran positivament com a **punts addicionals** a la **nota final** del projecte.

Planificació del projecte

El projecte el desenvoluparem en dues fases, que es correspondran amb el lliurament parcial i el lliurament final del projecte.

Primera versió del projecte:

- Inicialitzar el tauler del joc a partir de la informació guardada a un fitxer de text.
- Determinar els moviments vàlids de qualsevol fitxa del tauler.
- Moure una fitxa, comprovant que el moviment és vàlid i tenint en compte si es mata alguna fitxa del contrari, si cal "*bufar*" alguna fitxa del jugador actual o si la fitxa actual s'ha de convertir en dama.
- Mostrar l'estat actual del tauler.
- En aquesta primer versió treballarem sense visualització gràfica. Tindreu un **test d'autoavaluació a Gradescope** per poder validar el correcte funcionament de les diferents funcionalitats.

Planificació del projecte

El projecte el desenvoluparem en dues fases, que es correspondran amb el lliurament parcial i el lliurament final del projecte.

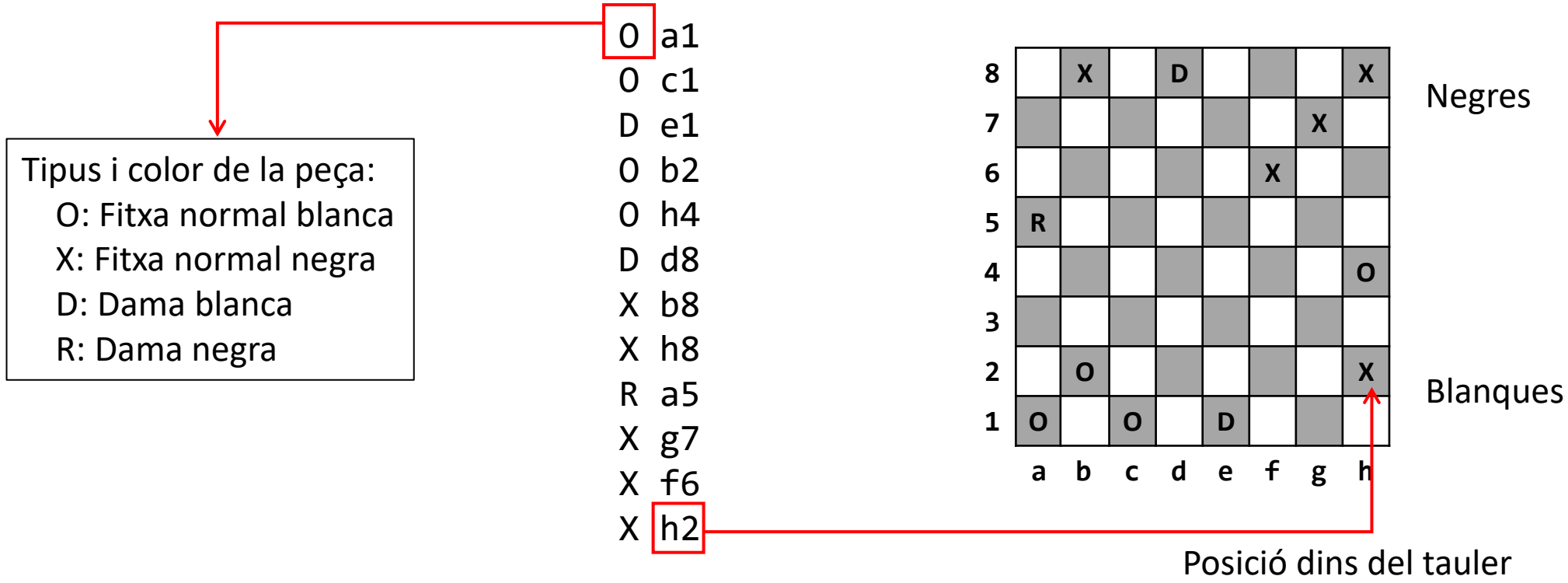
Segona versió del projecte:

- Adaptar el codi de la primera versió a la utilització de memòria dinàmica i apuntadors.
- Implementar la part gràfica del joc i la interacció del jugador amb el tauler durant el seu torn.
- Implementar el desenvolupament complet d'una partida a partir d'un estat inicial, alternant els torns dels jugadors fins al final de la partida.
- Guardar en un fitxer els moviments que es fan durant el desenvolupament de la partida.
- Reproduir una partida prèviament jugada executant els moviments guardats en un fitxer.
- Opcionalment, afegir funcionalitats extra, com per exemple, fer una implementació senzilla d'un mode de joc que ens permeti jugar de forma automàtica contra l'ordinador.

Primera versió del projecte

Inicialitzar una partida

S'haurà de poder inicialitzar l'estat inicial de la partida a partir de la informació guardada en un fitxer. El fitxer contindrà la informació de la posició inicial de cada peça seguint el format que es mostra en aquest exemple:



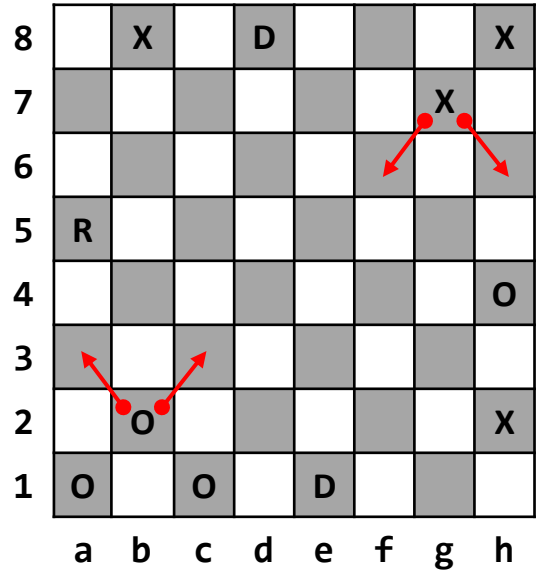
Tipus i color de la peça:
- O: Fitxa normal blanca
- X: Fitxa normal negra
- D: Dama blanca
- R: Dama negra

Primera versió del projecte

Determinar els moviments vàlids d'una peça

Donada una posició del tauler, s'haurà de poder recuperar el conjunt de moviments que pot fer la peça que ocupa aquella posició:

1. Moviments de desplaçament d'una **fitxa normal sense matar** una fitxa contrària
   - Desplaçament en diagonal a la posició adjacent
   - Les fitxes es poden moure només endavant (les fitxes blanques cap amunt i les negres cap avall)



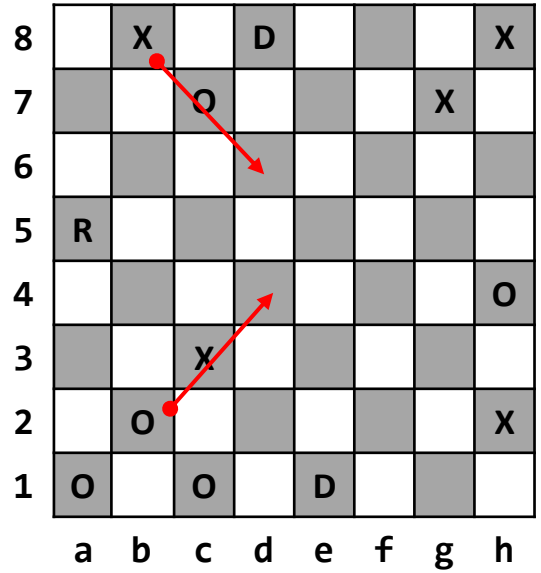
Tipus i color de la peça:
- O: Fitxa normal blanca
- X: Fitxa normal negra
- D: Dama blanca
- R: Dama negra

Primera versió del projecte

Determinar els moviments vàlids d'una peça

Donada una posició del tauler, s'haurà de poder recuperar el conjunt de moviments que pot fer la peça que ocupa aquella posició:

2. Moviments de desplaçament d'una **fitxa normal** que **maten una fitxa contrària**
   - Hi ha d'haver una fitxa contrària a la posició diagonal adjacent
   - Hi ha d'haver una posició lliure a la posició diagonal següent a la fitxa contrària



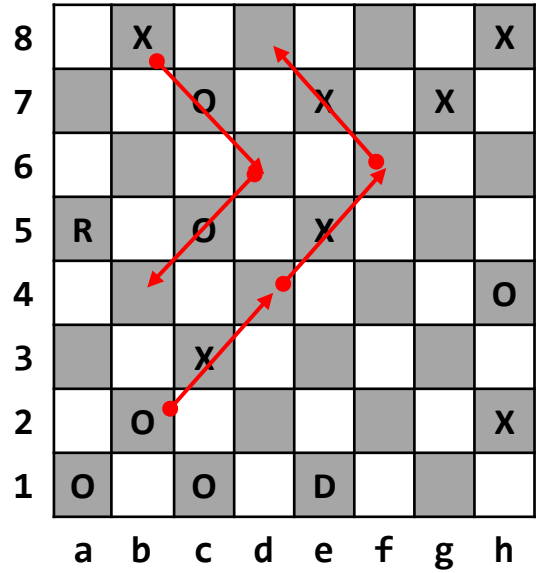
Tipus i color de la peça:
- O: Fitxa normal blanca
- X: Fitxa normal negra
- D: Dama blanca
- R: Dama negra

Primera versió del projecte

Determinar els moviments vàlids d'una peça

Donada una posició del tauler, s'haurà de poder recuperar el conjunt de moviments que pot fer la peça que ocupa aquella posició:

3. Moviments de desplaçament d'una **fitxa normal que maten vàries fitxes contràries**
   - En un mateix moviment es poden encadenar varis desplaçaments consecutius que impliquin matar una fitxa contrària.



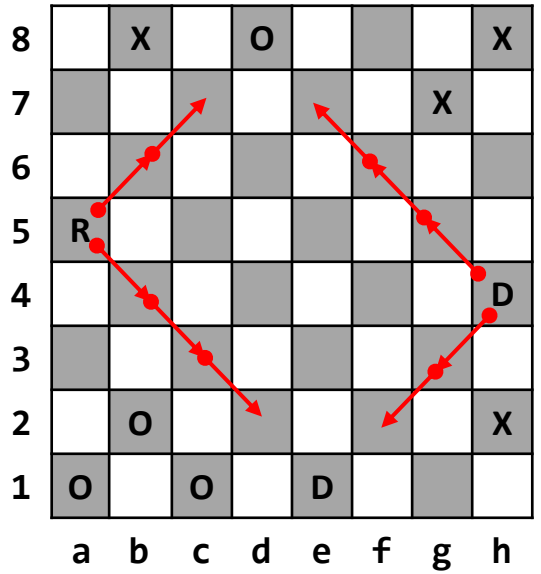
Tipus i color de la peça:
- O: Fitxa normal blanca
- X: Fitxa normal negra
- D: Dama blanca
- R: Dama negra

Primera versió del projecte

Determinar els moviments vàlids d'una peça

Donada una posició del tauler, s'haurà de poder recuperar el conjunt de moviments que pot fer la peça que ocupa aquella posició:

4. Moviments de desplaçament d'una **dama sense matar** una fitxa contrària
   - Desplaçament en diagonal a qualsevol posició lliure fins que es trobi una altra fitxa (pròpia o contrària)
   - Es pot desplaçar en qualsevol direcció (cap endavant i cap enrera)



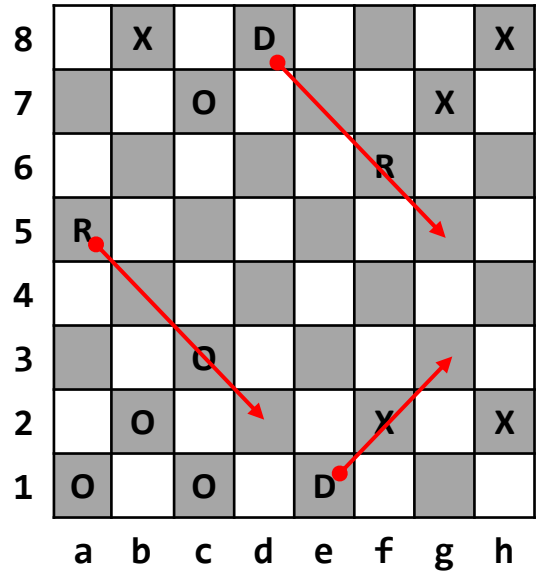
Tipus i color de la peça:
- O: Fitxa normal blanca
- X: Fitxa normal negra
- D: Dama blanca
- R: Dama negra

Primera versió del projecte

Determinar els moviments vàlids d'una peça

Donada una posició del tauler, s'haurà de poder recuperar el conjunt de moviments que pot fer la peça que ocupa aquella posició:

5. Moviments de desplaçament d'una **dama que maten una fitxa contrària**
   - Hi ha d'haver una fitxa contrària en alguna posició de qualsevol de les diagonals amb una posició lliure a continuació.
   - La dama ha de quedar col·locada a la posició següent de la fitxa que es mata.



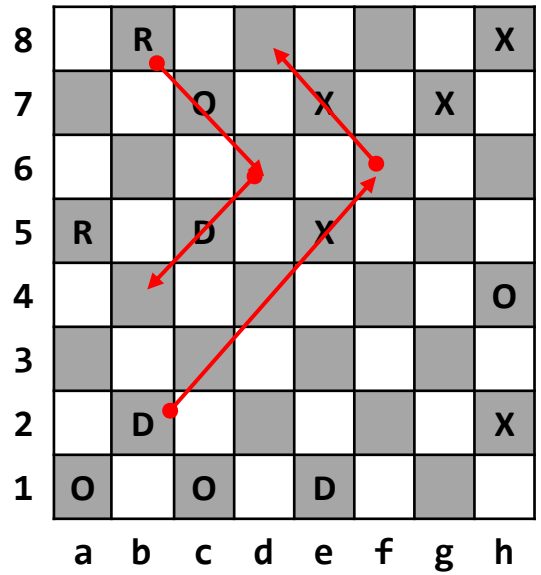
Tipus i color de la peça:
- O: Fitxa normal blanca
- X: Fitxa normal negra
- D: Dama blanca
- R: Dama negra

Primera versió del projecte

Determinar els moviments vàlids d'una peça

Donada una posició del tauler, s'haurà de poder recuperar el conjunt de moviments que pot fer la peça que ocupa aquella posició:

6. Moviments de desplaçament d'una **dama que maten vàries fitxes contràries**
   - En un mateix moviment es poden encadenar varis desplaçaments consecutius que impliquin matar una fitxa contrària.



Primera versió del projecte

Estructura de classes

Tauler

```
Fitxa m_tauler[N_FILES][N_COLUMNES];
```

```
void inicialitza(const string& nomFitxer);
void actualitzaMovimentsValids();
void getPosicionsPossibles(cont Posicio& origen,
            int& nPosicions, Posicio posicionsPossibles[]);
bool mouFitxa(const Posicio& origen, const Posicio& desti);
string toString() const;
```

Fitxa

```
...
```

```
...
```

Moviment

```
...
```

```
...
```

Posicio

```
...
```

```
Posicio(const string& posicio);
bool operator==(const Posicio& posicio) const;
...
```

Primera versió del projecte

Classe `Fitxa`

- Guarda la informació d'una peça del tauler de joc
- Haureu de pensar quins atributs fan falta per guardar el color (de quin jugador és), el tipus de la peça i el conjunt de moviments vàlids que pot fer la peça en cada moment
- Afegiu tots els mètodes que facin falta: constructor, getters/setters necessaris, mètodes per gestionar els moviments vàlids de la peça, convertir la fitxa en dama, ...
- Per codificar el tipus i el color de les peces us donem ja declarats aquests dos tipus `enumerate`:

```
typedef enum
{
    TIPUS_NORMAL,
    TIPUS_DAMA,
    TIPUS_EMPTY
} TipusFitxa;
```

```
typedef enum
{
    COLOR_NEGRE,
    COLOR_BLANC,
} ColorFitxa;
```

Primera versió del projecte

Classe `Moviment`

- Ha de servir per guardar la informació dels moviments que poden fer les peces del tauler
- Haureu de pensar quins atributs fan falta per poder guardar tota la informació del moviment, tenint en compte que en un sol moviment la peça pot saltar per vàries posicions del tauler i matar vàries fitxes contràries.
- Haureu d'afegir tots els mètodes que facin falta per poder gestionar qualsevol tipus de moviment

Primera versió del projecte

Classe `Posicio`

- Ha de servir per codificar qualsevol de les posicions del tauler
- Penseu els atributs que ha de tenir per poder guardar una posició del tauler
- Afegiu els mètodes que facin falta: constructor, getters/setters necessaris i també algun mètode per convertir la posició en un string o inicialitzar-la a partir d'un string.
- Per poder executar el test que us donarem a Gradescope:
  - Cal que tingui un constructor que rebi com a paràmetre un string en el format que hem indicat abans quan hem explicat la inicialització de la partida.
  - Cal que tingui definida la sobrecàrrega de l'operador == per poder comparar si dues posicions són iguals o no.

Posicio

```
...
```

```
Posicio(const string& posicio);
bool operator==(const Posicio& posicio) const;
...
```

Primera versió del projecte

Classe `Tauler`

- Guarda la informació del tauler del joc
- Té un atribut per guardar una matriu on cada posició correspon a una de les caselles del joc i conté la informació de la peça que hi ha en aquella posició.
- A les posicions on no hi ha cap peça es pot guardar una peça de tipus `FITXA_EMPTY`
- Haureu d'implementar els mètodes públics de la classe `Tauler`. Per implementar-los haureu d'implementar altres mètodes privats auxiliars que us ajudin a estructurar el codi seguint els principis de la descomposició modular.

Tauler

```
Fitxa m_tauler[N_FILES][N_COLUMNES];
```

```
void inicialitza(const string& nomFitxer);
void actualitzaMovimentsValids();
bool mouFitxa(const Posicio& origen, const Posicio& desti);
void getPosicionsPossibles(cont Posicio& origen,
            int& nPosicions, Posicio posicionsPossibles[]);
string toString() const;
```

Tauler

```
Fitxa m_tauler[N_FILES][N_COLUMNES];
```

```
void inicialitza(const string& nomFitxer);
void actualitzaMovimentsValids();
void getPosicionsPossibles(cont Posicio& origen,
            int& nPosicions, Posicio posicionsPossibles[]);
bool mouFitxa(const Posicio& origen, const Posicio& desti);
string toString() const;
```

Primera versió del projecte

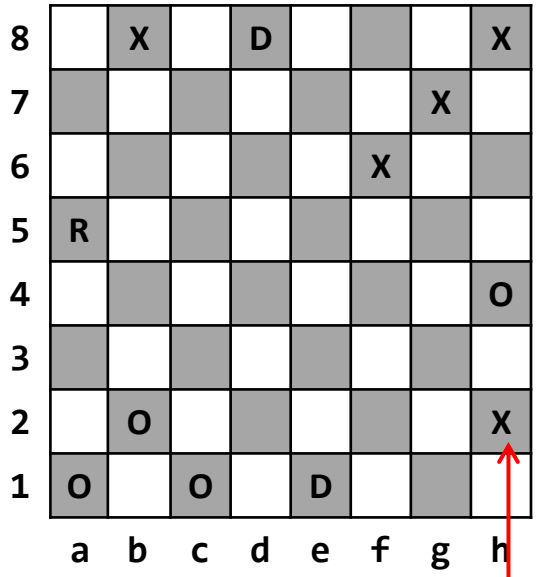
Classe `Tauler`: mètodes

```
void inicialitza(const string& nomFitxer)
```

- Inicialitza el tauler de joc a partir de la informació d'un fitxer de text en el format que s'ha explicat abans

Tipus i color de la peça:
- O: Fitxa normal blanca
- X: Fitxa normal negra
- D: Dama blanca
- R: Dama negra

```
O a1
O c1
D e1
O b2
O h4
D d8
X b8
X h8
R a5
X g7
X f6
X h2
```



Posició dins del tauler

Tauler

```
Fitxa m_tauler[N_FILES][N_COLUMNES];
```

```
void inicialitza(const string& nomFitxer);
void actualitzaMovimentsValids();
void getPosicionsPossibles(cont Posicio& origen,
            int& nPosicions, Posicio posicionsPossibles[]);
bool mouFitxa(const Posicio& origen, const Posicio& desti);
string toString() const;
```

Primera versió del projecte

Classe `Tauler`: mètodes

`const actualitzaMovimentsValids() const`

- Actualitzar tots els moviments vàlids que pot qualsevol de les peces del tauler
- Per guardar cadascun dels moviments es pot fer servir la **classe `Moviment`.**
- Penseu com podeu guardar tots els moviments de totes les peces del tauler utilizant les classes `Fitxa`, `Moviment` i `Posicio`
- Per implementar aquest mètode penseu bé quins mètodes auxiliars us fan falta a la classe `Tauler` i també quins mètodes hauríeu d'afegir a les classes `Fitxa` i `Moviment`

(*) Més endavant, en una altra sessió de classe, us explicarem com podeu plantejar l'algorisme que determina tots els moviments vàlids que pot fer una fitxa.

Tauler

```
Fitxa m_tauler[N_FILES][N_COLUMNES];
```

```
void inicialitza(const string& nomFitxer);
void actualitzaMovimentsValids();
void getPosicionsPossibles(cont Posicio& origen,
            int& nPosicions, Posicio posicionsPossibles[]);
bool mouFitxa(const Posicio& origen, const Posicio& desti);
string toString() const;
```

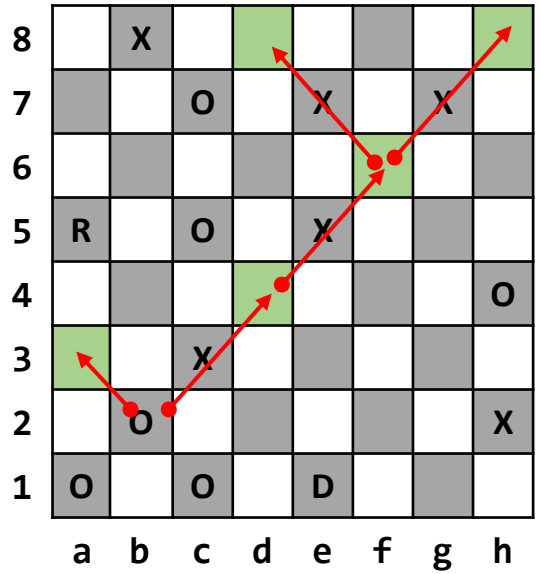
Primera versió del projecte

Classe `Tauler`: mètodes

```
void getPosicionsPossibles(const Posicio& origen, int& nPosicions,
                           Posicio posicionsPossibles[])
```

- El mètode retorna a l'array `posicionsPossibles` totes les posicions on es podria moure la peça que ocupa la posició `origen`, tenint en compte tots els passos de qualsevol moviment vàlid de la peça. `nPosicions` indica el nº total de posicions possibles.

Exemple:



Posicions possibles de la posició 'b2':
['a3','d4','f6','d8','h8']

Tauler

```
Fitxa m_tauler[N_FILES][N_COLUMNES];
```

```
void inicialitza(const string& nomFitxer);
void actualitzaMovimentsValids();
void getPosicionsPossibles(cont Posicio& origen,
            int& nPosicions, Posicio posicionsPossibles[]);
bool mouFitxa(const Posicio& origen, const Posicio& desti);
string toString() const;
```

Primera versió del projecte

Classe ChessBoard: mètodes

```
bool mouFitxa (const Posicio& origen, const Posicio& desti)
```

- Mou la peça que ocupa la posició del paràmetre `origen` a la posició del paràmetre `desti`.
- Ha de comprovar que la posició destí estigui dins dels moviments vàlids de la peça.
- Si no ho està no fa el moviment i retorna `false`.
- A part de moure la peça de la posició origen a la posició destí ha de fer el següent:
  - Eliminar del tauler totes les peces que es matin fent el moviment.
  - Si la peça arriba al final del tauler, convertir-la en una dama.
  - Si el moviment no ha matat una peça contrària quan hi ha altres moviments que sí permeten fer-ho o el moviment no ha matat el màxim de peces contràries possibles, *bufar* una peça del jugador (eliminar la peça del tauler) , seguint els criteris que s'han explicat a les regles del joc.

Tauler

```
Fitxa m_tauler[N_FILES][N_COLUMNES];
```

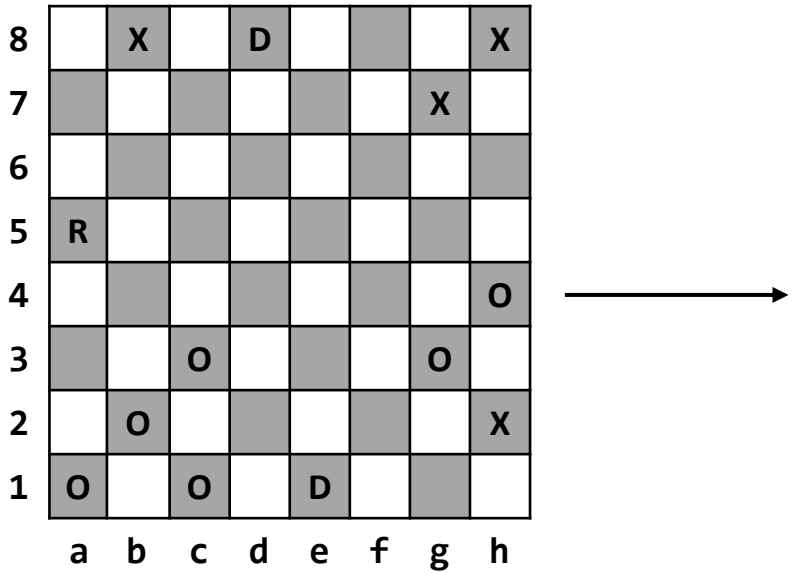
```
void inicialitza(const string& nomFitxer);
void actualitzaMovimentsValids();
void getPosicionsPossibles(cont Posicio& origen,
            int& nPosicions, Posicio posicionsPossibles[]);
bool mouFitxa(const Posicio& origen, const Posicio& desti);
string toString() const;
```

Primera versió del projecte

Classe ChessBoard: mètodes

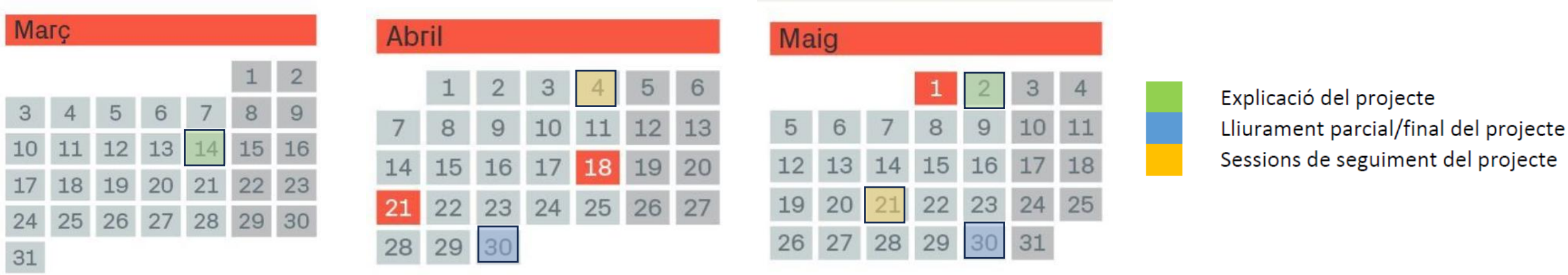
`string ToString() const`

- Genera un string amb l'estat actual del tauler de joc en el format que especifiquem a continuació




```
8: _ X _ D _ _ _ X
7: _ _ _ _ _ _ X _
6: _ _ _ _ _ _ _ _
5: R _ _ _ _ _ _ _
4: _ _ _ _ _ _ _ O
3: _ _ O _ _ _ O _
2: _ O _ _ _ _ _ X
1: O _ O _ D _ _ _
   A B C D E F G H
```

Organització del projecte



- La propera sessió de seguiment del projecte serà el **divendres 4 d'abril**.
  - Revisió del que heu avançat en el projecte, de com plantegeu el disseny de les classes (atributs i mètodes de cada classe) i la implementació dels mètodes de la classe `Tauler` (passos 1-5 descrits a la següent diapositiva)
  - Comptarà un **20% de la nota del lliurament parcial**.
- El lliurament parcial del projecte amb la implementació de la primera versió serà el dimecres **30 d'abril**. Farem una avaluació online individual del lliurament de cada grup.
- A Gradescope tindreu un **test d'autoavaluació** dels mètodes públics de la classe `Tauler` que heu d'implementar.
- Us demanarem crear un **repositori a GitHub** amb el codi del projecte i que ens hi doneu accés per poder veure i avaluar el vostre codi.

Propers passos: alguns consells

1. Penseu quins mètodes us poden fer falta a la classe `Posicio`. Feu una primera implementació de la classe.
2. Penseu quins atributs i mètodes us poden fer falta a la classe `Fitxa`. Feu la declaració de la classe (no cal que implementeu els mètodes, de moment)
3. Penseu quins atributs i mètodes us poden fer falta a la classe `Moviment`. Feu la declaració de la classe (no cal que implementeu els mètodes, de moment)
4. Feu la implementació del mètode `inicialitza` de la classe `Tauler` per poder inicialitzar el tauler de fitxer. Implementeu els mètodes que us facin falta a les classes `Fitxa` i `Posicio`, Si fa falta, modifiqueu la declaració inicial que hagueu fet d'aquestes classes.
5. Penseu com podeu descomposar el codi del mètode `actualitzaMovimentsValids` amb funcions més petites i simples.
   - Penseu bé com podeu organitzar el codi del mètode per poder implementar de la forma més simple possible i evitant duplicitats, el codi per recuperar qualsevol tipus de moviment vàlid per cada qualsevol tipus de peça de qualsevol color
6. Comenceu a implementar el codi del mètode `actualitzaMoviments`, pels moviments més simples (desplaçaments sense matar de fitxes normals i desplaçaments matant només una fitxa contrària).