Exercici avaluable: joc de l'oca
Tema 1: Programació orientada a objectes

Exercici

Volem implementar una versió simplificada del joc de la Oca. En aquesta versió simplificada el joc tindrà menys caselles que en la versió habitual i les caselles seran només de 5 tipus:

- Casella normal: no té cap acció associada.
- Oca: el jugador salta fins a la següent casella on hi hagi una oca i torna a tirar. Si no hi ha cap més oca des d'aquella posició fins a l'úlitma casella, es queda a la mateixa casella i no torna a tirar
- Pou: el jugador que hi cau es queda dos torns sense tirar.
- Mort: el jugador ha de tornar a la casella inicial.
- Casella final: última casella del tauler. El jugador que hi arriba primer guanya la partida. S'hi ha d'arribar amb un valor exacte del dau. Si la suma del valor del dau a la posició actual del jugador dóna un valor més gran que la posició de la casella final, el jugador es queda a la seva posició actual, sense poder-se moure.

Per implementar el joc voldrem crear tres classes: `Jugador`, `Casella` i `Tauler`.

Exercici

La **classe `Jugador`** servirà per guardar la informació necessària de cadascun dels jugadors que participen a la partida. Haurà de tenir atributs per poder guardar a quina posició del tauler està el jugador (valor entre 1 i nº màxim de caselles), si pot tirar o no al següent torn i, en cas de que no pugui tirar, quants torns ha d'estar sense tirar, i si és el guanyador de la partida. Haurà de tenir els següents mètodes públics:

- Un constructor per defecte que inicialitzi el jugador a la primera posició del tauler per començar la partida.
- Mètodes `getPosicio` i `setPosicio` per recuperar i modificar la posició del tauler on està el jugador.
- Mètodes `getTornsInactiu` i `setTornsInactiu` per recuperar i modificar el nº de torns que ha d'estar el jugador sense tirar.
- Un mètode `guanya`, que servirà per marcar el jugador com a guanyador de la partida.
- Un mètode `esGuanyador`, que retornarà si el jugador és el guanyador de la partida o no.
- Un mètode `potTirar` que retorna si el jugador pot tirar o no.

Exercici

```
class Jugador
{
public:
   Jugador();
   void setPosicio(int casella);
   void setTornsInactiu(int nTorns);
   void guanya();
   int getPosicio() const;
   int getTornsInactiu() const;
   bool esGuanyador() const;
   bool potTirar() const;
private:
   int m_casella;
   bool m_potTirar;
   int m_nTornsInactiu;
   bool m_guanyador;
};
```

Exercici

La **classe `Casella`** guarda la informació bàsica de cada casella del joc i permet gestionar les accions que s'han de fer quan el jugador arriba a la casella, en funció del tipus de casella. Haurà de tenir atributs per guardar la posició i el tipus de la casella. Com a mètodes ha de tenir els següents:

- Un constructor per defecte que incialitzi la posició de la casella a 0 i el tipus a casella normal.
- Getters i setters per recuperar i modificar tant la posició com el tipus de la casella.
- Un mètode `esOca` que retorni si la casella correspon a una oca o no.
- Un mètode `executaAccio`, que rep com a paràmetre un objecte de classe `Jugador` i ha de fer totes les accions necessàries que s'han de produir quan el jugador arribar a aquesta casella segons hem explicat abans per cada tipus de casella. Per totes les caselles ha de modificar la posició del jugador perquè coincideixi amb la posició de la casella. A més a més, segons el tipus de casella s'ha de fer el següent, cridant als mètodes de la classe `Jugador`:
  - Casella normal: no s'ha de fer res més.
  - Oca: s'ha de retornar `true` com a valor de retorn de la funció (indicant que és una oca i que el jugador ha de tornar a tirar). Per la resta de tipus de caselles, la funció retorna `false`.
  - Pou: s'ha de fixar a 2 el nº de torns sense poder tirar del jugador.
  - Mort: el jugador s'ha de moure a la casella inicial
  - Casella final: s'ha de marcar el jugador com a guanyador de la partida.

Exercici

```
class Casella
{
public:
   Casella();
   void setPosicio(int posicio);
   void setTipus(int tipus);
   int getPosicio() const;
   int getTipus() const;
   bool esOca() const;
   bool executaAccio(Jugador& j);
private:
   int m_posicio;
   int m_tipus;
};
```

Exercici

La **classe `Tauler`** haurà de tenir els atributs necessaris per poder guardar totes les caselles del joc (fins a un màxim de 63), les dades de tots els jugadors que hi participen (fins a un màxim de 4) i quin és el jugador que té el torn en cada moment del joc. Haurà de tenir com a mètodes:

- Un constructor per defecte que inicialitzi un tauler sense caselles ni jugadors.
- Un mètode inicialitza amb la capçalera següent:

```
void inicialitza(int tipusCaselles[], int nCaselles,
                 int nJugadors);
```

Aquest mètode ha d'inicialitzar totes les caselles del tauler a partir del nº de caselles i l'array `tipusCaselles` que rep com a paràmetres. L'array conté valors enters indicant el tipus de totes les caselles des de la casella 1 fins a l'última. També ha de fixar i inicialitzar el nº de jugadors al valor que es passa com a paràmetre, i donar el torn actual al primer jugador.

Els valors enters que indiquen el tipus de cada casella queden fixats per aquestes constants que estan declarades al fitxer `casella.h`:

```
const int NORMAL = 1;
const int OCA = 2;
const int POU = 3;
const int MORT = 4;
const int FINAL = 5;
```

Exercici

- Un mètode per simular un torn del joc amb la capçalera següent:

```
void tornJoc(int valorDau);
```

Aquest mètode ha de servir per actualitzar l'estat de la partida en funció del jugador que tingui el torn. Si el jugador que té el torn no pot tirar, no es fa res, però es disminueix el nº de torns sense tirar del jugador.

Si el jugador actual sí que pot tirar, el paràmetre `valorDau` tindrà el valor del dau entre 1 i 6. S'ha de calcular la nova posició a la que ha d'anar el jugador i si la posició és més gran que la casella final tampoc es fa res, considerem que el jugador no es pot moure.

Si es pot moure, s'haurà de moure el jugador a la casella corresponent i fer les accions associades al tipus de casella cridant al mètode `executaAccio` de la classe `Casella`. Com hem explicat abans si `executaAccio` retorna `true` vol dir que la casella on ha caigut el jugador és una oca. En aquest cas, s'haurà de buscar dins del tauler la següent casella que sigui una oca i moure el jugador a aquesta nova casella, conservant el torn a la següent tirada. Si ja no hi ha més oques abans de la casella final, el jugador no es mou i no conserva el torn.

Al final, si el jugador no conserva el torn s'ha de passar el torn al següent jugador modificant l'atribut corresponent de la classe.

Exercici

- Un mètode `getTipusCasella` amb la capçalera següent:

```
int getTipusCasella(int nCasella);
```

Aquest mètode ha de retornar el tipus de la casella de la posició que es passa com a paràmetre (començant per la posició 1).

- Un mètode `getEstatJugador` amb la capçalera següent:

```
int getEstatJugador(int nJugador, int& posicio,
    bool& potTirar, int& nTornsInactiu, bool& guanyador);
```

Aquest mètode ha de retornar l'estat del jugador (posició del tauler, si pot tirar, nº de torns sense tirar (si no ho pot fer) i si és el guanyador del joc) que es passa com a paràmetre (començant pel jugador 1) als paràmetres per referència `posició`, `potTirar`, `nTornsInactiu` i `guanyador`.

Exercici

```
class Tauler
{
public:
   Tauler();
   void inicialitza(int tipusCaselles[], int nCaselles,
                    const int& nJugadors);
   void tornJoc(int valorDau);
   int getTipusCasella(int nCasella) const;
   void getEstatJugador(int nJugador, int& posicio, bool& potTirar,
                    int& nTornsInactiu, bool& guanyador) const;
private:
   Casella m_caselles[NUMERO_CASELLES];
   int m_nCaselles;
   Jugador m_jugadors[NUMERO_JUGADORS];
   int m_nJugadors;
   int m_torn;
};
```