# Sessió 12

Tema 2

# Exercici 2.9:

2.9 Donades les classes Carta i Joc:

```
#include <cstddef>
#include <iostream>
using namespace std;
const int PALCARTABUIDA = 5;
class Carta
{public:
  Carta();
  Carta(int pal, int valor);
  Carta(const Carta& c);
  ~Carta();
  bool esCartaBuida();
  void setCarta(int noupal, int nouvalor);
  int getValor() const;
  int getPal() const;
  void escriuCarta() const;
  Carta& operator=(const Carta& c);
  bool operator==(const Carta& c);
 private:
  int m_valor;
  int m_pal;
};
```

```
#include "Carta.h"
#include <stack>
const int CORS = 0;
const int DIAMANTS = 1;
const int PIQUES = 2;
const int TREVOLS = 3;
const int N_CARTES = 13;
const int N_PALS  = 4;
class Joc
{ public:
    Joc();
    ~Joc();
    bool treureCartaBaralla();
    bool posarDestapadesABaralla();
    bool posarCartaAPal(int pal);
    void escriuJoc();
    void donaJoc(Carta& cBaralla,
                 Carta& cDestapada,
                 Carta cPila[N_PALS])const;
   private:
     //m_baralla;
     //m_destapades;
     //m_pilaResultat[N_PALS];
};
```

# Exercici 2.9:

2.9 Donades les classes Carta i Joc: volem definir el joc del solitari de manera simplificada. Tenim:

- **4 pals**: CORS=0; DIAMANTS = 1; PIQUES = 2; TREVOLS = 3;
- Cada pal te **13 valors**: de l'1 al 13, essent el 11, 12 i 13 els corresponents a J, Q, K respectivament i l'1 es l'As.
- **4 piles**, una per cada pal de cartes, a on anirem col·locant les cartes del pal que tenen per nom de forma ordenada, començant per l'as i acabant pel rei. Anomenades:
  - **`m_pilaResultat[CORS]`**
  - **`m_pilaResultat[DIAMANTS]`**
  - **`m_pilaResultat[PIQUES]`**
  - **`m_pilaResultat[TREVOLS].`**
- **1 pila** de cartes que anomenarem **m_baralla**, on tindrem inicialment totes les cartes ordenades:

  (Pal,Valor): (0,1),(1,1),(2,1),(3,1),(0,2),(1,2),(2,2),(3,2),…,(0,13),(1,13),(2,13),(3,13)
- **1 pila** de cartes que anomenarem **m_destapades**, on anirem posant les cartes de m_Baralla a mesura que les anem destapant. Quedant sempre a dalt la ultima carta destapada.

# Exercici 2.9.1: Definició piles stl

2.9.1 Definiu les piles: m_baralla, m_destapades, m_pilaResultat[N_PALS] de la classe joc com a piles de la llibreria stl

```
const int PALCARTABUIDA = 5;
class Carta
{public:
  Carta();
  Carta(int pal, int valor);
  Carta(const Carta& c);
  ~Carta();
  bool esCartaBuida();
  void setCarta(int noupal, int nouvalor);
  int getValor() const;
  int getPal() const;
  void escriuCarta() const;
  Carta& operator=(const Carta& c);
  bool operator==(const Carta& c);
 private:
  int m_valor;
  int m_pal;
};
```

```
#include "Carta.h"
#include <stack>
const int CORS = 0;
const int DIAMANTS = 1;
const int PIQUES = 2;
const int TREVOLS = 3;
const int N_CARTES = 13;
const int N_PALS  = 4;
class Joc
{ public:
    Joc();
    ~Joc();
    bool treureCartaBaralla();
    bool posarDestapadesABaralla();
    bool posarCartaAPal(int pal);
    void escriuJoc();
    void donaJoc(Carta& cBaralla,
                 Carta& cDestapada,
                 Carta cPila[N_PALS])const;
  private:
    //m_baralla;
    //m_destapades;
    //m_pilaResultat[N_PALS];
};
```

# Exercici 2.9.1: Solució definició piles stl

2.9.1 Definiu les piles: m_baralla, m_destapades, m_pilaResultat[N_PALS] de la classe joc com a piles de la llibreria stl

```
#include "Carta.h"
#include <stack>
const int CORS = 0;
const int DIAMANTS = 1;
const int PIQUES = 2;
const int TREVOLS = 3;
const int N_CARTES = 13;
const int N_PALS  = 4;
class Joc
{ public:
    Joc();
    ~Joc();
    bool treureCartaBaralla();
    bool posarDestapadesABaralla();
    bool posarCartaAPal(int pal);
    void escriuJoc();
    void donaJoc(Carta& cBaralla,
                 Carta& cDestapada,
                 Carta cPila[N_PALS])const;
```

```
private:
  std::stack<Carta> m_baralla;
  std::stack<Carta> m_destapades;
  std::stack<Carta> m_pilaResultat[N_PALS];

};
```

# Exercici 2.9.2: Constructor per defecte

2.9.2 Implementar el constructor per defecte de Joc que inicialitzi la pila m_baralla tal com hem descrit: Joc();

(Pal,Valor): (0,1),(1,1),(2,1),(3,1),(0,2),(1,2),(2,2),(3,2),...,(0,13),(1,13),(2,13),(3,13)

[top de la pila] [bottom de la pila]

```
const int PALCARTABUIDA = 5;
class Carta
{public:
  Carta();
  Carta(int pal, int valor);
  Carta(const Carta& c);
  ~Carta();
  bool esCartaBuida();
  void setCarta(int noupal, int nouvalor);
  int getValor() const;
  int getPal() const;
  void escriuCarta() const;
  Carta& operator=(const Carta& c);
  bool operator==(const Carta& c);
 private:
  int m_valor;
  int m_pal;
};
```

```
#include "Carta.h"
#include <stack>
const int CORS = 0;
const int DIAMANTS = 1;
const int PIQUES = 2;
const int TREVOLS = 3;
const int N_CARTES = 13;
const int N_PALS  = 4;
class Joc
{ public:
    Joc(); ...
  private:
     //m_baralla;
     //m_destapades;
     //m_pilaResultat[N_PALS];
};
```

# Exercici 2.9.2: Solució constructor per defecte

2.9.2 Implementar el constructor per defecte de Joc que inicialitzi la pila m_baralla tal com hem descrit: Joc();

```
Joc::Joc()
{
    for (int valor = N_CARTES; valor > 0; valor--)
        for (int pal = N_PALS - 1; pal >= 0; pal--)
        {
            Carta c;
            c.setCarta (pal, valor);
            m_baralla.push(c);
        }
}
```

# Exercici 2.9.3: treureCartaBaralla()

2.9.3 Escriu el mètode de “Joc”:bool treureCartaBaralla();

a)Si m_Baralla té cartes treu carta de m_Baralla i la posa a m_destapades, retornant true.

b)Si m_Baralla no té cartes ens retorna false.

```
const int PALCARTABUIDA = 5;
class Carta
{public:
  Carta();
  Carta(int pal, int valor);
  Carta(const Carta& c);
  ~Carta();
  bool esCartaBuida();
  void setCarta(int noupal, int nouvalor);
  int getValor() const;
  int getPal() const;
  void escriuCarta() const;
  Carta& operator=(const Carta& c);
  bool operator==(const Carta& c);
 private:
  int m_valor;
  int m_pal;
};
```

```
#include "Carta.h"
#include <stack>
const int CORS = 0;
const int DIAMANTS = 1;
const int PIQUES = 2;
const int TREVOLS = 3;
const int N_CARTES = 13;
const int N_PALS  = 4;
class Joc
{ public:
    Joc(); ...
    bool treureCartaBaralla();
  private:
     //m_baralla;
     //m_destapades;
     //m_pilaResultat[N_PALS];
};
```

# Exercici 2.9.3: Solució treureCartaBaralla()

2.9.3 Escriu el mètode de “Joc”:bool treureCartaBaralla();

a)Si m_Baralla té cartes treu carta de m_Baralla i la posa a m_destapades, retornant true.

b)Si m_Baralla no té cartes ens retorna false.

```
// Treu una carta de la baralla i la posa a m_destapades
bool Joc::treureCartaBaralla()
{
    bool movimentFet = false;

    if (!m_baralla.empty())
    {
        m_destapades.push(m_baralla.top());
        m_baralla.pop();
        movimentFet = true;
    }
    return movimentFet;
}
```

# Exercici 2.9.4: posarCartaAPal()

2.9.4 Escriu un mètode de "Joc" que passa una carta de m_destapades a m_pilaResultat[pal], si és possible. Per fer-ho primer comprova que m_destapades tingui cartes i, si les té, que la primera sigui del pal correcte i el número sigui correlatiu al que ja està a m_pilaResultat[pal], o un as en cas de no haver-hi cap carta a m_pilaResultat[pal]. Si el moviment es pot fer retorna true i false en cas contrari.

```
const int PALCARTABUIDA = 5;
class Carta
{public:
  Carta();
  Carta(int pal, int valor);
  Carta(const Carta& c);
  ~Carta();
  bool esCartaBuida();
  void setCarta(int noupal, int nouvalor);
  int getValor() const;
  int getPal() const;
  void escriuCarta() const;
  Carta& operator=(const Carta& c);
  bool operator==(const Carta& c);
 private:
  int m_valor;
  int m_pal;
};
```

```
#include "Carta.h"
#include <stack>
const int CORS = 0;
const int DIAMANTS = 1;
const int PIQUES = 2;
const int TREVOLS = 3;
const int N_CARTES = 13;
const int N_PALS  = 4;
class Joc
{ public:
    Joc(); ...
    bool posarCartaAPal(int pal);
  private:
     //m_baralla;
     //m_destapades;
     //m_pilaResultat[N_PALS];
};
```

# Exercici 2.9.4: Solució posarCartaAPal()

```
bool Joc::posarCartaAPal(int pal)
{  bool movimentFet = false;

   if ( !m_destapades.empty())
   {
      if (m_destapades.top().getPal() == pal)
      {
         if (m_pilaResultat[pal].empty())
         {  if ( (m_destapades.top().getValor()) == 1)
            {  movimentFet = true;
               m_pilaResultat[pal].push(m_destapades.top());
               m_destapades.pop();
            }
         }
         else
         {  if ( (m_destapades.top().getValor()) ==
                  m_pilaResultat[pal].top().getValor()+1)
            {  movimentFet = true;
               m_pilaResultat[pal].push(m_destapades.top());
               m_destapades.pop();
            }
         }
      }
   }

   return movimentFet;
}
```

Cas en que no hi ha cartes:
Has de posar l'As

Cas en que ja hi ha cartes:
Has de posar carta següent