```cpp
//1r - En la clase parte publica
friend ostream& operator<<(ostream& out, Matriu& m);
//2n - hacer su definicion correspondiente
//segundo parametro el objeto que vamos a hacer la sobrecarga
ostream& operator<<(ostream& out, Matriu& m)
{
    for (int i = 0; i < m.m_vecX.size(); i++)//en out add las cadenas
        out << "add texto que queramos mostrar";
    return out;//finalmente devolvemos el conjunto de cadenas
}
// ---------------- Lectura ficheros ----------------------
#include<fstream>
void leerFichero(nombreFichero)
{
    ifstream miFichero; //Declaro el objeto ifstream para poder leer el fichero
    miFichero.open(nomFitxer);//abrir ficheros

    if (miFichero.is_open())
    {
        int fila, col;
        string cadena;
        while (!miFichero.eof()) // mientras no sea el final del fichero
        {
            ...
            //se guarda con el tipo de variable de llegada
            miFichero >> fila >> col >> cadena;
            ...
        }
        miFichero.close();
    }
}
```

```cpp
template<class T>
inline T& SmartPointer<T>:: operator*()
{
    cout << "op" << endl;
    if (m_pointer == NULL)
        cout << "Error: apuntar a NULL" << endl;

    return *m_pointer
}
```

```cpp
bool LliuramentsEstudiant::eliminaTramesa(const string& data)
{
    bool eliminarTramesa = false;
    std::forward_list<Tramesa>::iterator itAnterior = m_trameses.before_begin();
    std::forward_list<Tramesa>::iterator itActual = m_trameses.begin();
    while (!eliminarTramesa && (itActual != m_trameses.end()))
    {
        if (itActual->getData() == data)
        {
            eliminarTramesa = true;
            itActual = m_trameses.erase_after(itAnterior);
        }
        else
        {
            itAnterior = itActual;
            itActual++;
        }
    }
    return eliminarTramesa;
}
```