Sessió 11

Tema 2

Exercici 2.8:

2.8 Donades les següents declaracions de les classes Titulacio i Estudiant:

```
#include <string>
using namespace std;
#include "Estudiant.h"
#include <forward_list>
class Titulacio
{public:
   void mostraEstudiants();
   void afegeixEstudiant(Estudiant &e);
 private:
   string m_nom;
   int m_nEstudiants;
   std::forward_list<Estudiant> m_estudiants;
};
```

```
#include "Assignatura.h"
#include <string>
using namespace std;
class Estudiant
{public:
    void mostraAssignatures();
    string& getNIU() ;
    int getNAssigActual();
    Estudiant()
    Estudiant(const string &nom, const string &niu)
    bool afegeixAssignatura(int codi, int nCredits);
 private:
    string m_nom;
    string m_NIU;
    int m_nAssigActual;
    static const int MAX_ASSIGNATURES = 10;
    AssignaturaExpedient  m_assignatures[MAX_ASSIGNATURES];
};
```

Exercici 2.8:

2.8 Implementeu mètode

void eliminaEstudiants();

elimini tots els estudiants d'una titulació amb 0 assignatures.

```
#include <string>
using namespace std;
#include "Estudiant.h"
#include <forward_list>
class Titulacio
{public:
   void mostraEstudiants();
   void afegeixEstudiant(Estudiant &e);
 private:
   string m_nom;
   int m_nEstudiants;
   std::forward_list<Estudiant> m_estudiants;
};
```

```
#include "Assignatura.h"
#include <string>
using namespace std;
class Estudiant
{public:
    void mostraAssignatures();
    string& getNIU() ;
    int getNAssigActual();
    Estudiant()
    Estudiant(const string &nom, const string &niu)
    bool afegeixAssignatura(int codi, int nCredits);
 private:
    string m_nom;
    string m_NIU;
    int m_nAssigActual;
    static const int MAX_ASSIGNATURES = 10;
    AssignaturaExpedient  m_assignatures[MAX_ASSIGNATURES];
};
```

Exercici 2.8:

2.8 Implementeu mètode a Titulacio

```
bool buscaEstudiant(const string& niu,
                    std::forward_list<Estudiant>::iterator& actual,
                    std::forward_list<Estudiant>::iterator& anterior);
```

Que cerqui a la llista d'estudiants de la titulació un Estudiant:

Si el trovar retorna cert i un iterador a la seva posicio i a l'anterior

En cas contrari retorna fals.

```
#include <string>
using namespace std;
#include "Estudiant.h"
#include <forward_list>
class Titulacio
{public:
   void mostraEstudiants();
   void afegeixEstudiant(Estudiant &e);
 private:
   string m_nom;
   int m_nEstudiants;
   std::forward_list<Estudiant> m_estudiants;
};
```

Exercici 2.8:

```
bool buscaEstudiant(const string& niu,
                    std::forward_list<Estudiant>::iterator& actual,
                    std::forward_list<Estudiant>::iterator& anterior);
```

```
...
class Titulacio
{public:
   void mostraEstudiants();
   void afegeixEstudiant(Estudiant &e);
 private:
   string m_nom;
   int m_nEstudiants;
   std::forward_list<Estudiant> m_estudiants;
};
```

```
#include "Assignatura.h"
#include <string>
using namespace std;
class Estudiant
{public:
    void mostraAssignatures();
    string& getNIU() ;
    int getNAssigActual();
    Estudiant()
    Estudiant(const string &nom, const string &niu)
    bool afegeixAssignatura(int codi, int nCredits);
 private:
    string m_nom;
    string m_NIU;
    int m_nAssigActual;
    static const int MAX_ASSIGNATURES = 10;
    AssignaturaExpedient  m_assignatures[MAX_ASSIGNATURES];
};
```

Exercici 2.8:

2.8 Com quedaria ara el mètode afegeixEstudiant?

```
bool Titulacio::buscaEstudiant(const string& niu,
                               std::forward_list<Estudiant>::iterator& actual,
                               std::forward_list<Estudiant>::iterator& anterior)
```

```
void Titulacio::afegeixEstudiant (Estudiant &e)
{
   bool trobat = false;
   std::forward_list<Estudiant>::iterator anterior = m_estudiants.before_begin();
   std::forward_list<Estudiant>::iterator actual = m_estudiants.begin();
   while ((actual != m_estudiants.end()) && (!trobat))
   {
      Estudiant estActual = *actual;
      if (e.getNIU() < estActual.getNIU())
         trobat = true;
      else
      {
         anterior = actual;
         actual++;
      }
   }
   m_estudiants.insert_after(anterior, e);
   m_nEstudiants++;
}
```

Exercici 2.8:

2.8 Com quedaria ara el mètode afegeixEstudiant?

```
bool Titulacio::buscaEstudiant(const string& niu,
                               std::forward_list<Estudiant>::iterator& actual,
                               std::forward_list<Estudiant>::iterator& anterior)
```

```
void Titulacio::afegeixEstudiant (Estudiant &e)
{
   bool trobat = false;
   std::forward_list<Estudiant>::iterator anterior = m_estudiants.before_begin();
   std::forward_list<Estudiant>::iterator actual = m_estudiants.begin();
   while ((actual != m_estudiants.end()) && (!trobat))
   {
      Estudiant estActual = *actual;
      if (e.getNIU() < estActual.getNIU())
         trobat = true;
      else
      {
         anterior = actual;
         actual++;
      }
   }
   if (!trobat)
   { m_estudiants.insert_after(anterior, e);
     m_nEstudiants++;
   }
}
```

Exercici 2.8:

2.8 Implementeu mètode a Titulacio

```
bool afegeixAssignaturaEstudiant(const string& niu,
                                 int codiAssig, int nCredits)
```

Que cerqui a la llista d'estudiants de la titulació un Estudiant i si existeix i te lloc al seu array d'assignatures li agfegeixi una asignatura i retorni true i si o no existeix o existeix però no té lloc a l'array d'assignatures retorni false.

Suposeu que teniu un mètode a Estudiant:

```
bool afegeixAssignatura(int codi, int nCredits)
```

```
#include <string>
using namespace std;
#include "Estudiant.h"
#include <forward_list>
class Titulacio
{public:
   void mostraEstudiants();
   void afegeixEstudiant(Estudiant &e);
   bool buscaEstudiant(const string& niu,
                       std::forward_list<Estudiant>::iterator& actual,
                       std::forward_list<Estudiant>::iterator& anterior)
 private:
   string m_nom;
   int m_nEstudiants;
   std::forward_list<Estudiant> m_estudiants;
};
```