Header and Source Files

Header File -> A.h

```
#ifndef A_H
#define A_H

class A {
private:
   int number;
public:
   A();
   int getNumber() const;
   void setNumber(int n);
};
#endif
```

Source File -> A.cpp

```
#include "A.h"

A::A():number(0){}

int A::getNumber() const {
   return number;
}

void A::setNumber(int n){
   number = n;
}
```

Header and Source Files

Header File -> B.h

```
#ifndef B_H
#define B_H

#include "A.h"

class B {
private:
   A objectA;
public:
   B();
   const A& getObjectA() const;
   void setObjectA(const A& objA);
};
#endif
```

Source File -> B.cpp

```
#include "B.h"

B::B():objectA(){}

const A& B::getObjectA() const {
   return objectA;
}

void B::setObjectA(const A& objA){
   objectA = objA;
}
```

Header and Source Files

Source File contenente il main -> main.cpp

```
#include "B.h" // Non serve includere A.h
#include <iostream>
using namespace std;

int main(){
   B objectB;
   A objectA;
   cout<<
objectB.getObjectA().getNumber()<<endl;
   objectA.setNumber(1);
   objectB.setObjectA(objectA);
   cout<<
objectB.getObjectA().getNumber()<<endl;
   return 0;
}
```

Per compilare bisogna utilizzare **SOLO** i source files:

```
g++ main.cpp A.cpp B.cpp
```

Non bisogna utilizzare gli header files:

```
g++ main.cpp A.h B.h A.cpp B.cpp -> ERRATO
```

L'opzione **-o** consente di specificare il nome dell'eseguibile:

```
g++ main.cpp A.cpp B.cpp
 -o esempioClassiAB
```

Per eseguire:

```
./esempioClassiAB
```

Software Utili

- Esistono diverse alternative per compilare ed eseguire codice c++
- Si consiglia l'installazione di una qualsiasi distribuzione di Linux, come Ubuntu
- Può bastare anche in una macchina virtuale, https://www.virtualbox.org/
- In genere, tale sistema operativo ha già installato di base i software per lo sviluppo utili per il corso:
  - Il compilatore **g++**
  - Un editor di testo, ad esempio **gedit**

Attenzione a...

- Utilizzare virgolette per includere gli header files
  - Ad esempio: `#include "A.h"` e non `#include <A.h>`
- Evitare le inclusioni superflue
  - Ad esempio: l'inclusione di B.h, ci permette di evitare l'inclusione di A.h
- È bene definire le classi template interamente in header files
- Evitare di usare caratteri accettati come nomi di variabili, classi, etc.
- Aggiungere sempre le direttive `#ifndef`, `#define` ed `#endif` negli header files
- Compilare frequentemente, a seguito dell'aggiunta di un metodo, di una classe, etc.
- Creare sempre un main per testare il corretto svolgimento
- Per utilizzare le features di c++11, in fase di compilazione, occorre specificare l'opzione

  `–std=c++11`
  - Ad esempio: `g++ -std=c++11 main.cpp A.cpp B.cpp`
- In modo equivalente per c++14, usare l'opzione `-std=c++14`