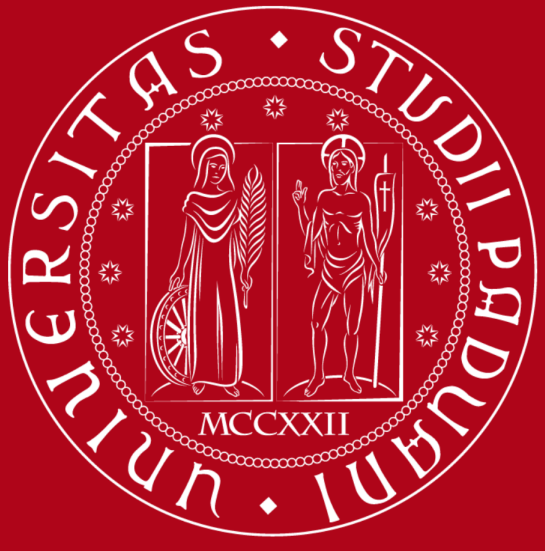


UNIVERSITÀ DEGLI STUDI DI PADOVA

Header file e template

Stefano Ghidoni

DIPARTIMENTO
DI INGEGNERIA
DELL'INFORMAZIONE

Agenda

- Uso dei template e header file
  - Gestione
  - Ripercussioni

Template

- I template combinano flessibilità ed efficienza

Effetti collaterali

- I template combinano flessibilità ed efficienza
- Effetti collaterali:
  - Poca separazione tra dichiarazione e definizione
  - Errori di compilazione poco comprensibili
    - Il compilatore controlla sia il template che il suo argomento

Effetti collaterali

- I compilatori spesso richiedono che i template siano completamente definiti prima di essere usati
  - Le definizioni sono spostate negli header!
- Non è imposto dallo standard, ma dallo specifico compilatore

Header con i template

- Come mantenere la separazione tra interfaccia e implementazione con i template?

Header con i template

- Come mantenere la separazione tra interfaccia e implementazione con i template?
- L'header viene suddiviso in due blocchi:
  - File .h – tipico contenuto da header:
    - Definizioni di classi
    - Dichiarazioni di funzioni
  - File .hpp – tipico contenuto da .cpp
    - Definizioni di funzioni
- I blocchi sono combinati tramite un #include atipico

Header con i template

```
// file vector.h
#ifndef vector_h
#define vector_h

template<typename T>
class vector{
// ...
    void resize(int newsize);
// ...
};

// ...

#include "vector.hpp"

#endif // vector_h
```

Il preprocessore prende il codice nel .hpp e lo incolla.
Ogni volta che viene fatta una modifica all'.hpp, bisogna ricompilare tutto, che diventa un'operazione molto onerosa molto velocemente.

Header con i template

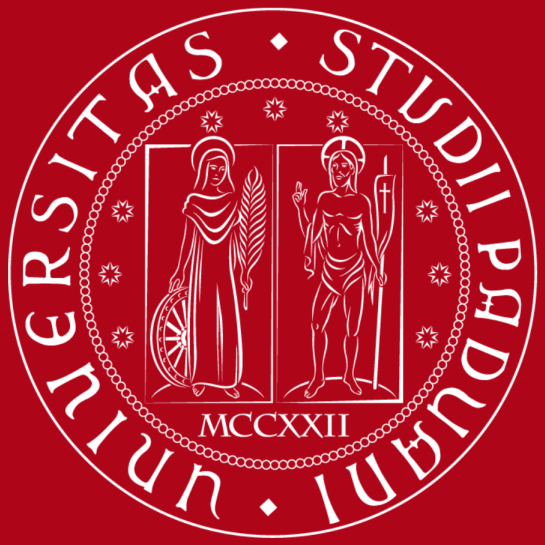
```
// file vector.hpp
#ifndef vector_hpp
#define vector_hpp

// ...

template <typename T>
void vector<T>::resize(int newsize)
{
// ...
}

// ...

#endif  // vector_hpp
```



UNIVERSITÀ
DEGLI STUDI
DI PADOVA

Header file e template

Stefano Ghidoni

DIPARTIMENTO
DI INGEGNERIA
DELL'INFORMAZIONE