

UNIVERSITÀ
DEGLI STUDI
DI PADOVA

Organizzazione del software in progetti complessi

Stefano Ghidoni



Agenda

- Progetti su più file
- Il ruolo degli header
- Include guards
- Linking
- Stile



Progetti su file multipli

- Progetti software complessi devono essere organizzati in più file
 - Migliore organizzazione
 - Migliore leggibilità
 - Maggiore semplicità organizzativa se più sviluppatori lavorano allo stesso progetto
 - Progetto intermedio e finale ☺



Progetti su file multipli

- Esistono due tipi fondamentali di file
 - File header (.h), contengono:
 - Definizioni di classe
 - Dichiarazioni di funzione
 - Inclusi (`#include`) ma **non compilati direttamente**
 - `g++ -o my_program source1.cpp source2.cpp my_header.h`
 - File sorgente (.cpp), contengono:
 - Definizioni di funzione
 - Il main (uno solo in tutto il progetto)
 - Compilati dal compilatore, **ma non inclusi**
 - ~~`#include "source1.cpp"`~~



Progetti su file multipli

- Perché queste due modalità?
 - Header: inclusi perché dichiarazioni di funzioni e definizioni di classi sono funzionali alla scrittura dei file .cpp
 - I sorgenti .cpp non hanno questa funzione, non devono essere inclusi
 - Le *translation unit* (file oggetto) sono linkati assieme
- Vediamolo con un esempio (già visto nella lezione 3!)



Progetti software in file multipli

- Come possiamo distribuire un SW in molti file?

```
int f(int i);
```

Dichiarazione



Header file
(my_func.h)

```
int main(void)
{
    int i = 0;

    i = f(i);

    return 0;
}
```

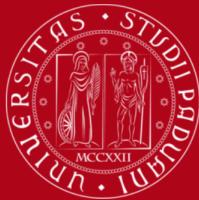
Chiamata

```
int f(int i)
{
    return i + 2;
}
```

Definizione



File sorgente di libreria
(my_func.cpp)



Header multipli

- Un file sorgente può includere più header

```
#include <iostream>
#include "date.h"
#include "book.h"
```

- Gli header possono includere altri header
 - book.h include date.h
- Questo può generare problemi
 - Inclusioni cicliche: a.h include b.h, e viceversa



Include guards

- Ogni header **deve** essere protetto dalle **include guards**

```
#ifndef RATIONAL_H
#define RATIONAL_H

// ...

#endif // RATIONAL_H
```

- Talvolta rese con:

```
#pragma once           // NON standard!!!
```

- Non è standard!



Errori di linking

- Lavorando con più file è più frequente la presenza di errori di linking
- È molto importante distinguere:
 - Errori di compilazione: problemi di sintassi, di conversione di tipo, ...
 - Segnalati da gcc/g++
 - Errori di linking: parti di software mancanti
 - Segnalati da ld Ad esempio l'utilizzo di funzione non definite o non dichiarate
- La prima verifica da fare in caso di errore è capire se è di compilazione o di linking



Errori di linking

- Tipico errore di linking: definizione mancante

```
double area(double w, double h);

int main()
{
    int w = 2, h = 4;
    int a = area(w, h);

    return 0;
}
```

Il compilatore accetta la chiamata di della funzione "area" perché ne individua la dichiarazione, lascia al linker il compito di cercare la definizione. Quando, come in questo caso, manca la definizione, avvène un errore segnalato da ld.



Coding style

- Dopo aver scritto un po' di codice è molto utile rivedere la Google C++ Style Guide
- Sezioni utili e comprensibili:
 - Header files
 - Scoping
 - Classes
 - Functions
 - Naming
 - Comments
 - Formatting



Coding style

- Header files
 - Self-contained Headers
 - The #define Guard
 - Include What You Use
 - Inline Functions
 - Names and Order of Includes



Coding style

- Scoping
 - Namespaces
 - Nonmember, Static Member, and Global Functions
 - Local Variables



Coding style

- Classes
 - Doing Work in Constructors
 - Alcuni commenti su funzioni virtuali che per il momento non avete studiato
 - Implicit Conversions
 - Copyable and Movable Types
 - Structs vs. Classes
 - Operator Overloading
 - Access Control
 - Declaration Order



Coding style

- Functions
 - Inputs and Outputs
 - Write Short Functions
 - Function Overloading
 - Default Arguments



Coding style

- Naming
 - General Naming Rules
 - File Names
 - Type Names
 - Variable Names
 - Constant Names
 - Function Names
 - Namespace Names
 - Enumerator Names



Coding style

- Comments
 - Comment Style
 - File Comments
 - Class Comments
 - Function Comments
 - Variable Comments
 - Implementation Comments
 - Punctuation, Spelling, and Grammar



UNIVERSITÀ
DEGLI STUDI
DI PADOVA

Organizzazione del software in progetti complessi

Stefano Ghidoni