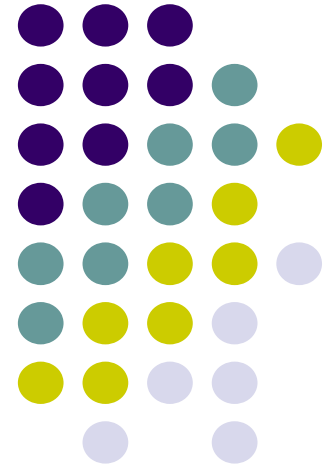


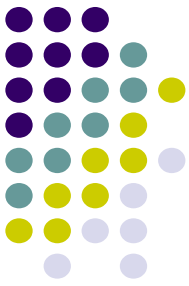


# Corso di Programmazione

*Richiami*  
*Compilazione Separata*



# Specifica e implementazione (1/3)



- E' buona norma tenere separata la *specifica* di un modulo dalla sua *implementazione*.
- Un programma utente di un modulo A deve conoscerne la specifica, ma disinteressarsi dei dettagli della sua implementazione.
- Ciò può essere realizzato scrivendo un file di intestazione o *header file* (con l'estensione **.h**) contenente le dichiarazioni che costituiscono l'**interfaccia** di A, ed un file separato per l'implementazione di A.
- Siccome ogni modulo deve essere *autoconsistente*, ovvero deve contenere tutte le informazioni necessarie per la compilazione, l'*header file* deve essere incluso (mediante la direttiva al preprocessore `#include`) nella implementazione di ogni modulo utente

Main. C

```
// Utilizzatore del modulo A  
#include "A.h"
```

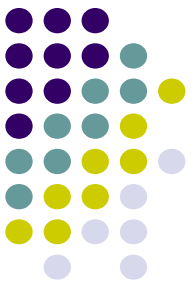
A.h

```
// Interfaccia di A
```

A.cpp

```
// Implementazione del modulo A  
#include "A.h"
```

# Specifica e implementazione (2/3)



- Esempio:
  - Un programma C++ consiste di più file sorgente che sono individualmente compilati in file oggetto
  - Questi sono poi collegati insieme per produrre la forma eseguibile del programma

```
// File: func.h:  
int somma(int, int);  
int prodotto(int, int);
```

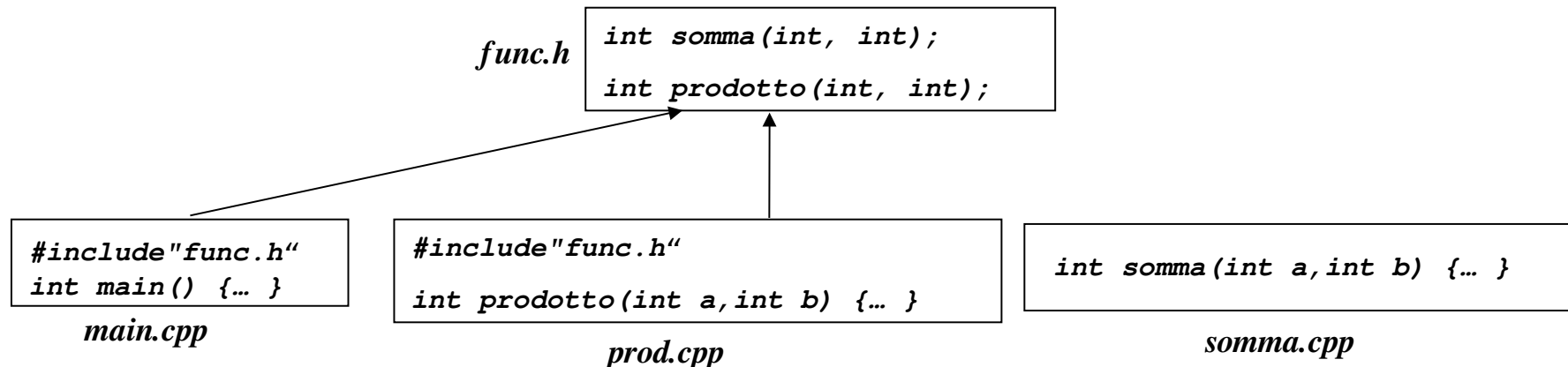
```
// File: somma.cpp:  
  
int somma(int a, int b)  
{  
    return a+b;  
}
```

```
// File: prod.cpp:  
  
#include "func.h"  
  
int prodotto(int a, int b) {  
    int prod=0;  
    for (int i=b; i>=1; i--) {  
        prod=somma(prod,a);  
    }  
    return prod;  
}
```



```
// File: main.cpp:
#include "func.h"
#include <iostream.h>    // pre-compilatore
#include <stdlib.h>       // pre-compilatore

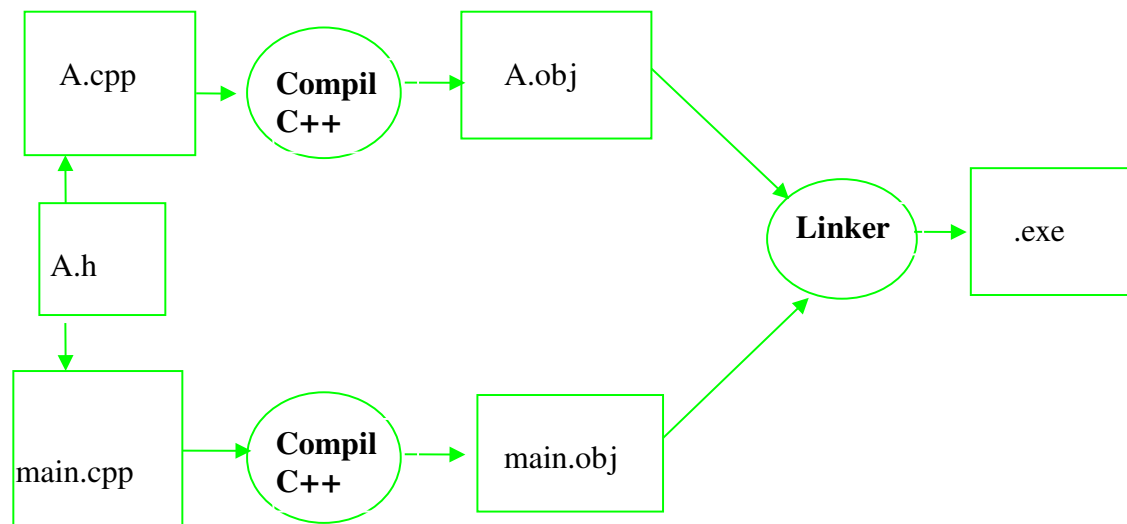
int main(){
    int m,n;
    cout << "inserire due numeri interi positivi:";
    cin >> m >> n;
    cout << "somma:" << somma(m,n);
    cout << "prodotto:" << prodotto(m,n);
    return 0;
}
```



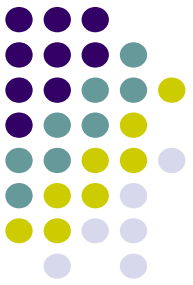
# Specifica e implementazione (3/3)



- Si osservi che, fintanto che l'interfaccia resta inalterata, l'implementazione può essere modificata senza dover ricompilare il modulo utente (ma naturalmente occorre ricollegare i moduli oggetto).



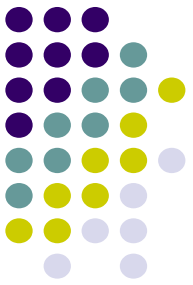
- La specifica, contenuta nel file di intestazione, può essere riguardata come una sorta di *contratto* sottoscritto tra l'implementatore e l'utente.
- Quando più programmatori lavorano simultaneamente ad un progetto di grandi dimensioni, una volta accordatisi sulla specifica dei vari moduli, possono procedere all'implementazione dei rispettivi moduli indipendentemente l'uno dagli altri.



# Librerie di moduli software

- Queste tecniche di sviluppo modulare consentono lo sviluppo su base professionale di librerie di moduli software.
- Il produttore di una libreria distribuisce:
  - i file di intestazione (che devono essere inclusi dall'utilizzatore nel codice sorgente) dei moduli che fanno parte della libreria;
  - i moduli di libreria in formato oggetto (già compilati), che l'utilizzatore deve collegare assieme ai propri moduli oggetto.
- Tale scelta è tipicamente motivata da esigenze di tutela della proprietà, ed inoltre evita di dover ricompilare i moduli di libreria.

# Inclusione di file

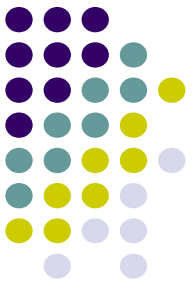


- Il preprocessore C, tramite la direttiva **#include**, puo' ricercare il file indicato in alcune directory standard o definite al momento della compilazione ed espanderlo testualmente in sostituzione della direttiva.
- La direttiva **#include** puo' essere impiegata in due forme:

`#include <nomefile>`  
`#include "nomefile"`

- Nel 1° caso il **nomefile** viene ricercato in un insieme di directory standard definite dall'implementazione ed in altre che sono specificate al momento della compilazione.
- Nel 2° caso il **nomefile** viene ricercato nella directory corrente e poi, se non e' stato trovato, la ricerca continua nelle directory standard e in quelle specificate al momento della compilazione come nel 1° caso.
- **N.B.** - Nel caso che un header venga modificato, e' necessario ricompilare tutti i sorgenti che lo includono.

# Con riferimento all'esempio GestioneNumeriComplessi


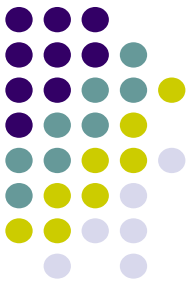


- ▼ GestioneNumeriComplessi
  - ▼ build
    - ▼ Debug
      - ▼ GNU-MacOSX
        - Complex.o.d
        - VettComplex.o
        - VettComplex.o.d
        - complex.o
        - main.o
        - main.o.d
  - ▼ dist
    - > Debug
  - > nbproject
    - .dep.inc
    - Complex.cpp
    - Complex.h
    - > GestioneNumeriComplessiNEW.zip
    - Makefile
    - VettComplex.cpp
    - VettComplex.h
    - complessi.txt
    - main.cpp

Tab Files del progetto in  
Netbeans




# Compilazione a riga di comando



Complex.cpp  
Complex.h  
main.cpp  
VettComplex.cpp  
VettComplex.h

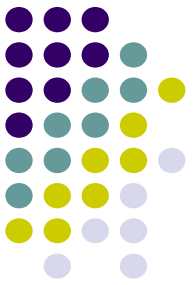
```
g++ -c Complex.cpp
```

Chiamata del compilatore



Complex.cpp  
Complex.h  
Complex.o  
main.cpp  
VettComplex.cpp  
VettComplex.h

# Compilazione a riga di comando

A file explorer window showing a list of files in a dark-themed interface. The files are: Complex.cpp, Complex.h, Complex.o, main.cpp, VettComplex.cpp, and VettComplex.h. Each file has a small icon to its left representing its extension (e.g., a document icon for .cpp, a document with a red 'h' for .h, and a document with a white 'o' for .o).

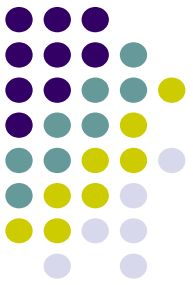
Complex.cpp  
Complex.h  
Complex.o  
main.cpp  
VettComplex.cpp  
VettComplex.h

```
g++ -c VettComplex.cpp
```

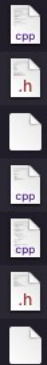
Chiamata del compilatore

A file explorer window showing the same list of files as before, but with an additional file, VettComplex.o, at the bottom. This new file is highlighted with a light blue background. The icons for the files remain the same as in the previous state.

Complex.cpp  
Complex.h  
Complex.o  
main.cpp  
VettComplex.cpp  
VettComplex.h  
VettComplex.o




# Compilazione a riga di comando



Complex.cpp  
Complex.h  
Complex.o  
main.cpp  
VettComplex.cpp  
VettComplex.h  
VettComplex.o

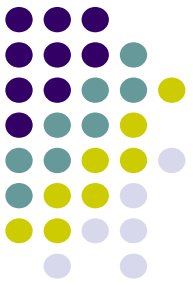
```
g++ -c main.cpp
```

Chiamata del compilatore



Complex.cpp  
Complex.h  
Complex.o  
main.cpp  
main.o  
VettComplex.cpp  
VettComplex.h  
VettComplex.o

# Compilazione a riga di comando



```
Complex.cpp
Complex.h
Complex.o
main.cpp
main.o
VettComplex.cpp
VettComplex.h
VettComplex.o
```

Chiamata del linker

Nome che diamo all'eseguibile

```
g++ -o eseguibile main.o VettComplex.o Complex.o
```

```
Complex.cpp
Complex.h
Complex.o
eseguibile
main.cpp
main.o
VettComplex.cpp
VettComplex.h
VettComplex.o
```

Per eseguire da terminale su Mac o Linux

`./eseguibile`

Per eseguire il .exe su Win basta  
richiamare il nome del file eseguibile  
`eseguibile`

```
domenico@MacBook-Air-di-Domenico-2 GestioneNumeriComplessiNEW % ./eseguibile
Inserisci il nome del file:
```