# Approfondimento

# Compilazione, linking, building, executing

**Codice sorgente salvato su dico come somma.cxx**

```
#include <iostream>

using namespace std;

int main()
{
        int sum, val=1;

        while(val <=10)
        {
         sum += val;
         //cout<< "val " <<  val << "\t sum " <<  sum  << endl;
         val++;
        }

        cout<< "Somma da 1 a 10: " <<  sum << endl;

        return 0;
}
```

# Compilazione, linking, building, executing

**Preprocessamento** (sostituzioen header files)
$ g++ -E somma.cxx -o somma.prep
$less somma.prep

Compilazione **Assembler**
$g++ -S somma.cxx -o somma.s
$less somma.s

Compilazione **oggetto (binario)**
$g++ -c somma.cxx -o somma.o
$less somma.o

**Linking (dinamico)**
$g++ somma.o -o somma
$ldd somma
$ls -lhtr somma

**Linking statico**
$g++ somma.o -o somma.static
$ldd somma.static
$ls -lhtr somma.static

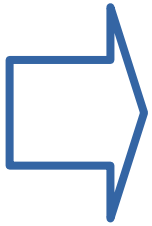
**Eseguire**
$./somma
Oppure
./$somma.static

Ad ogni passaggio provate ad
1) inviare in std outpu t il contenuto del file compilato:
$less nomefile
2) verificare dimensione file di output

# Automatizziamo il tutto

**Makefile:** programma per automatizzare il processo di creazione di file che dipendono da altri file, risolvendo le dipendenze e invocando programmi esterni per il lavoro necessario.  In pratica, compiliamo, linkiamo, cotruiamo il file eseguibile

```
# I commenti usano il cancelletto
target: dipendenze
        comando 1
        comando 2
            .
            .
            .
        comando n
```



```
#makefile per il file somma.cxx

all: somma.exe

somma.exe: somma.o
        g++ somma.o -o somma.exe

somma.o: somma.cxx
        g++ -c -g somma.cxx -o somma.o

clean:
        rm somma.o somma.exe
```

Create con un editor di testo il file Makefile così come riportato sopra (oppure scaricatelo da Moodle). Provate da riga di comando ad eseguire I seguenti comandi:

$ make clean
$ make somma.o
$ make somma.exe
$ make clean
$ make all

# Errori sintattici e logici

■ Se commetto **errore di scrittura delle istruzioni** all'interno del mio programma avrò un errore in fase di compilazione.

Il compilatore indica la linea dove ha incrontrato l'errore, si può dunque vericare, correggere l'errore nel codice sorgente e ricompilare.

■ Più subdoli sono gli **errori logici:** il codice è sintatticamente corretto, tuttativa **in esecuzione non vengono eseguete le operazioni** che l'utente desiderava o pensava di aver programmato. Questa tipologia di errrori non è semplice da individuare e si può procede comunente in due modi:

1. Interrompo al bisogno e ove necessario il flusso di esecuzione stampando il valore della variabili (ad esempio in somma.cxx `cout<< "val " << val << "\t sum " << sum << endl;`)
2. Uso dei "debugger" che mi permettono di interrompere il flusso del programma e verificare il valore delle variabili. Pe fare ciò è necessario compilare il codice sorgente con appositi simboli che verranno poi riconosciuti dal programma di debugging (prossima slide)

# Debugger gdb

**Compilo con simboli per il debugger**
$gcc -g somma.cxx -o somma

**Eseguo il debugger**
$gdb

A questo punto sono nell'**ambiente di debugging** che indico come (gdb)
(gdb) file somma
(gbd) run
(gdb) brek 13
(gdb) run
Breakpoint 1, main () at somma.cxx:13
13 val++;
(gdb) print val
$1 = 1
(gdb) print sum
$2 = 2
(gdb) continue
(gdb) print val
$1 = 2
(gdb) print sum
$2 = 3
..etc etc
$(gbb) quit