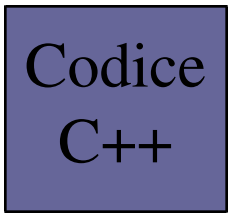
Creazione di un programma eseguibile

Livelli di programmazione

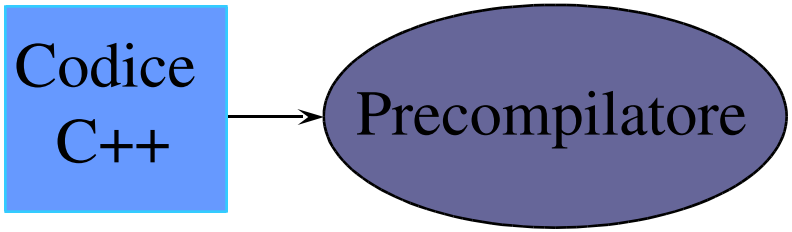
- Si può programmare a diversi livelli utilizzando linguaggi caratterizzati da diversi livelli di astrazione: più basso è il livello più vicino alla macchina (e ai suoi dettagli costruttivi) è il linguaggio.

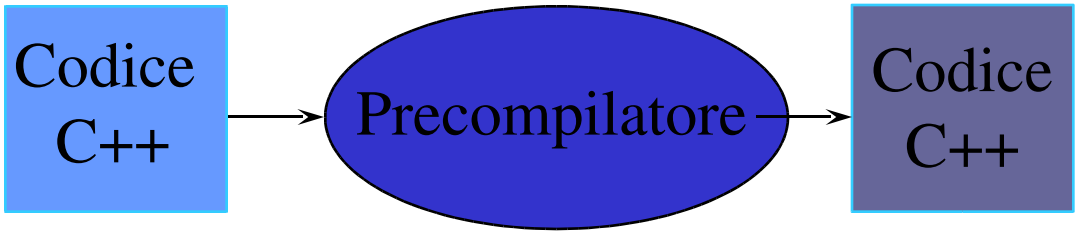
- Un programma scritto utilizzando il linguaggio proprio di un livello di programmazione è destinato ad essere tradotto (da un programma traduttore) in un linguaggio di livello inferiore o ad essere eseguito direttamente (da un programma interprete).


Codice
C++

Esempio di codice in C++

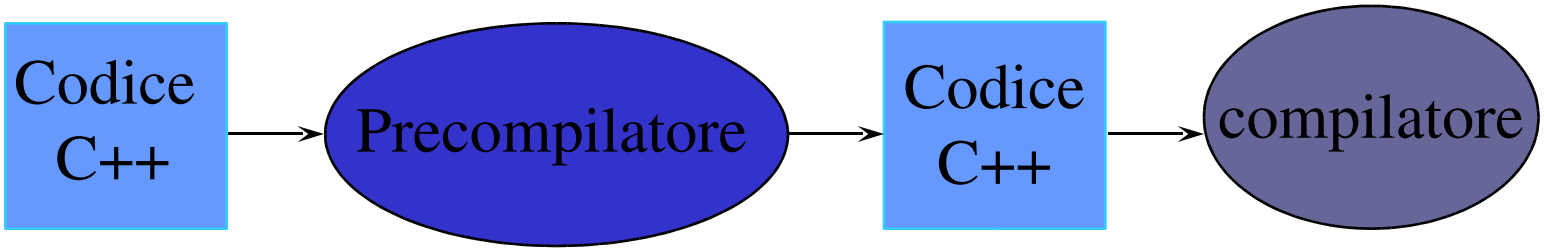
```
#define SCALA 2

int main(  ) {

    float x=10.;
    float y=20.;
    float z;
    z=x+y;
    z*=SCALA;
    return 1;
 }
```

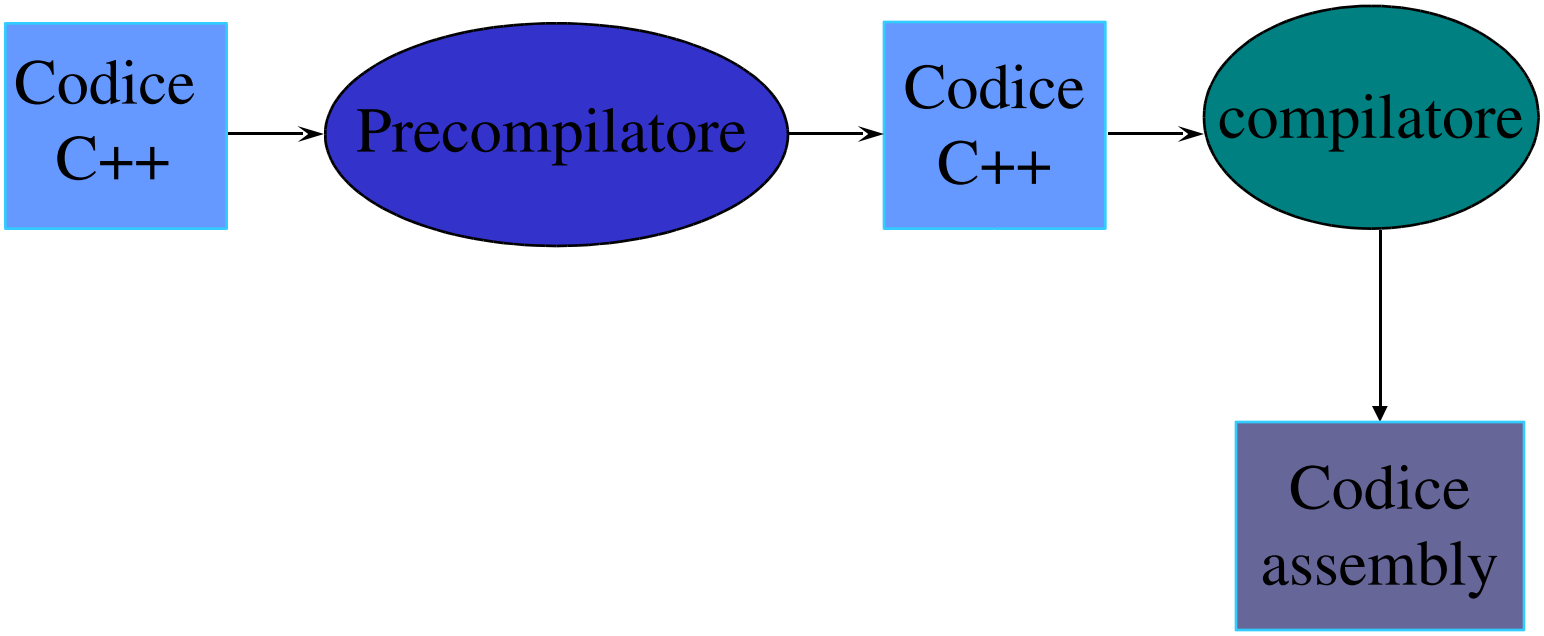

Codice
C++
Precompilatore


Codice
C++
Precompilatore
Codice
C++

Esempio di codice in C++ precompilato

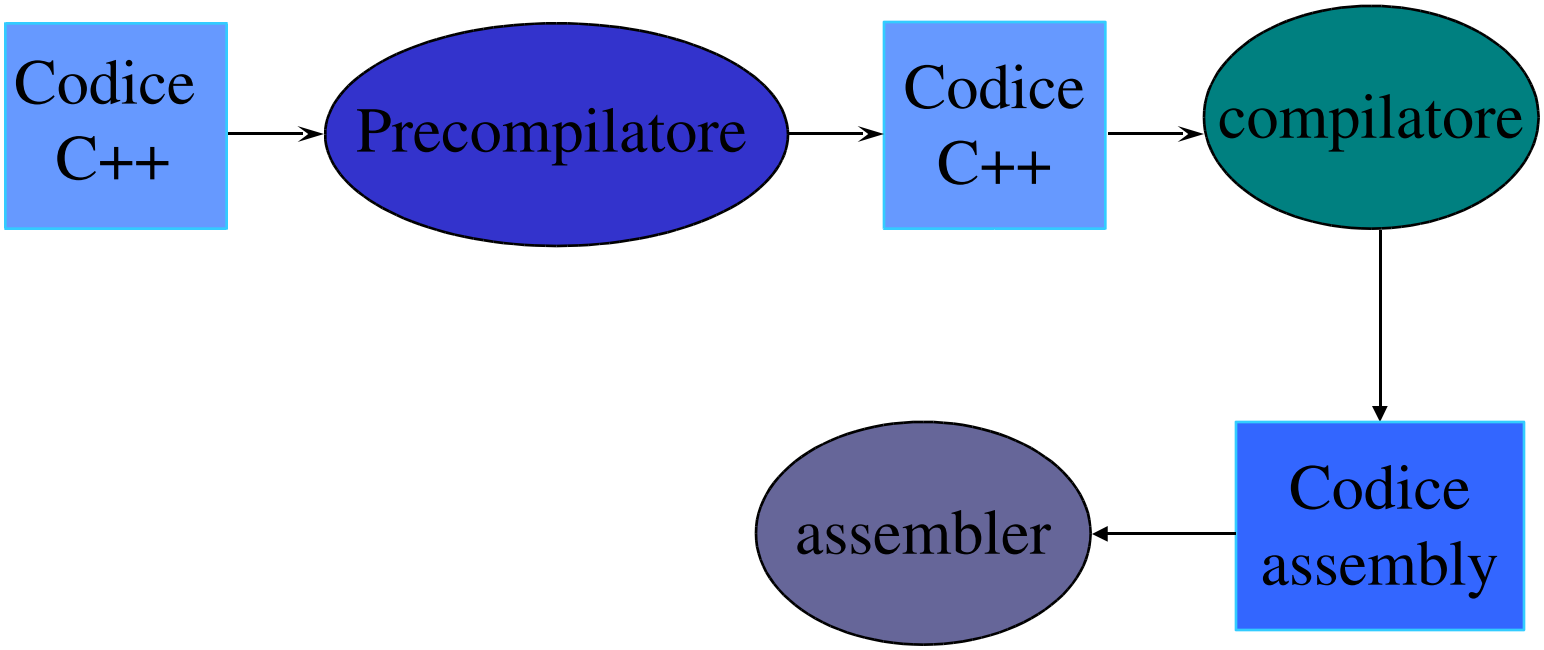
```
int main(  ) {

   float x=10.;
   float y=20.;
   float z;
   z=x+y;
   z*=2;
   return 1;
}
```

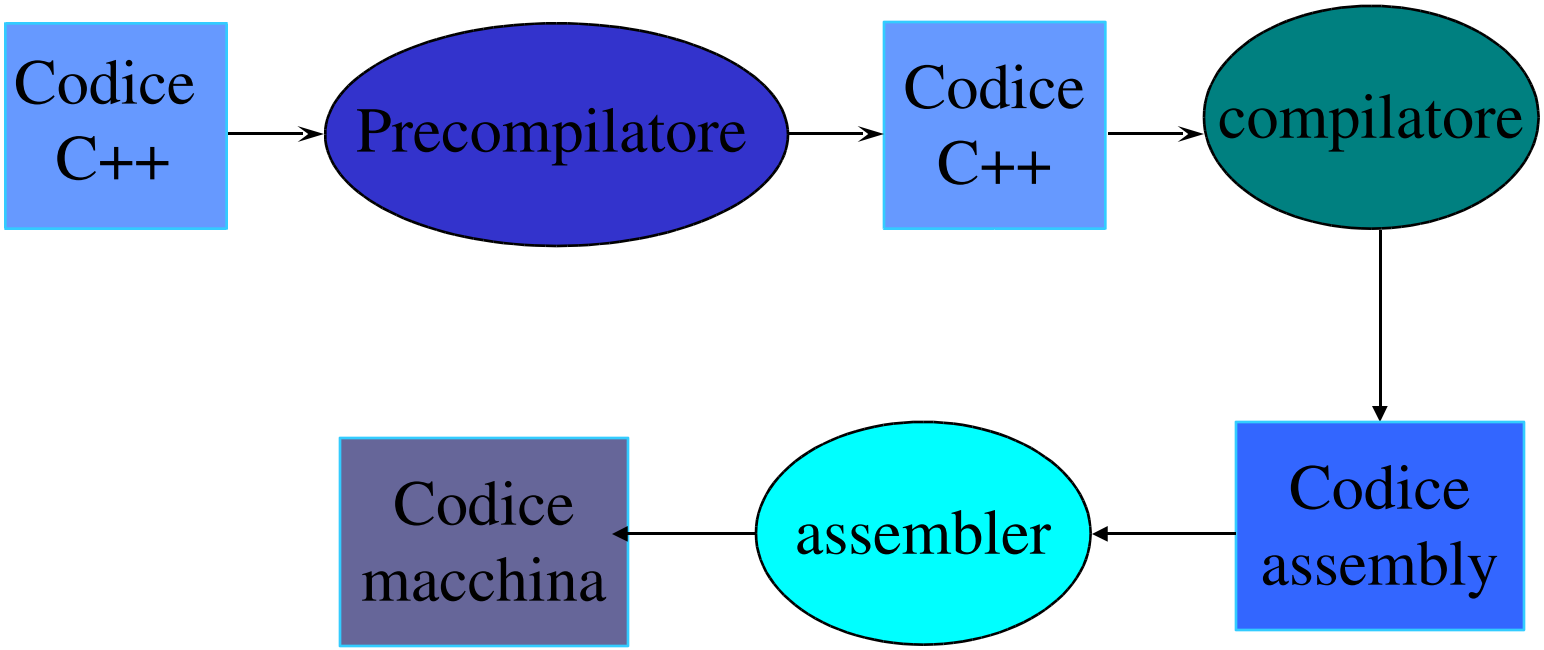

Codice
C++
Precompilatore
Codice
C++
compilatore


Codice
C++
Precompilatore
Codice
C++
compilatore
Codice
assembly

Esempio di codice assembly

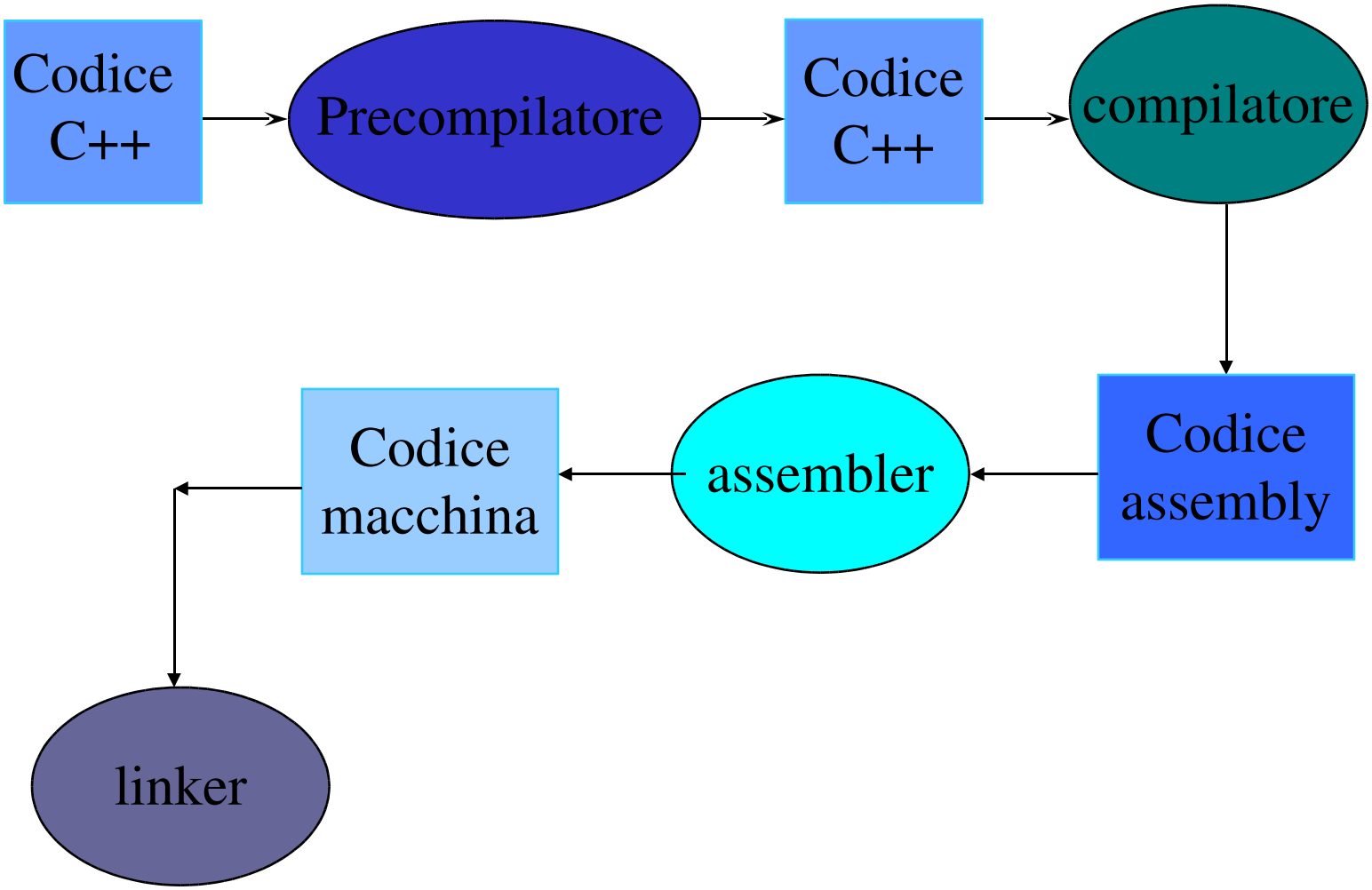
```
    .file       "prog2.cxx"
    .version    "01.01"
gcc2_compiled.:
.text
    .align 4
.globl main
    .type        main,@function
main:
    pushl %ebp
    movl %esp,%ebp
    subl $12,%esp
    movl $1092616192,-4(%ebp)
    movl $1101004800,-8(%ebp)
    flds -4(%ebp)
    fadds -8(%ebp)
    fstps -12(%ebp)
    flds -12(%ebp)
    fld %st(0)
    faddp %st,%st(1)
    fstps -12(%ebp)
    movl $1,%eax
    jmp .L1
    .align 4
    xorl %eax,%eax
    jmp .L1
    .align 4
.L1:
    leave
    ret
.Lfe1:
    .size        main,.Lfe1-main
    .ident       "GCC: (GNU) 2.7.2.3.f.1"
```

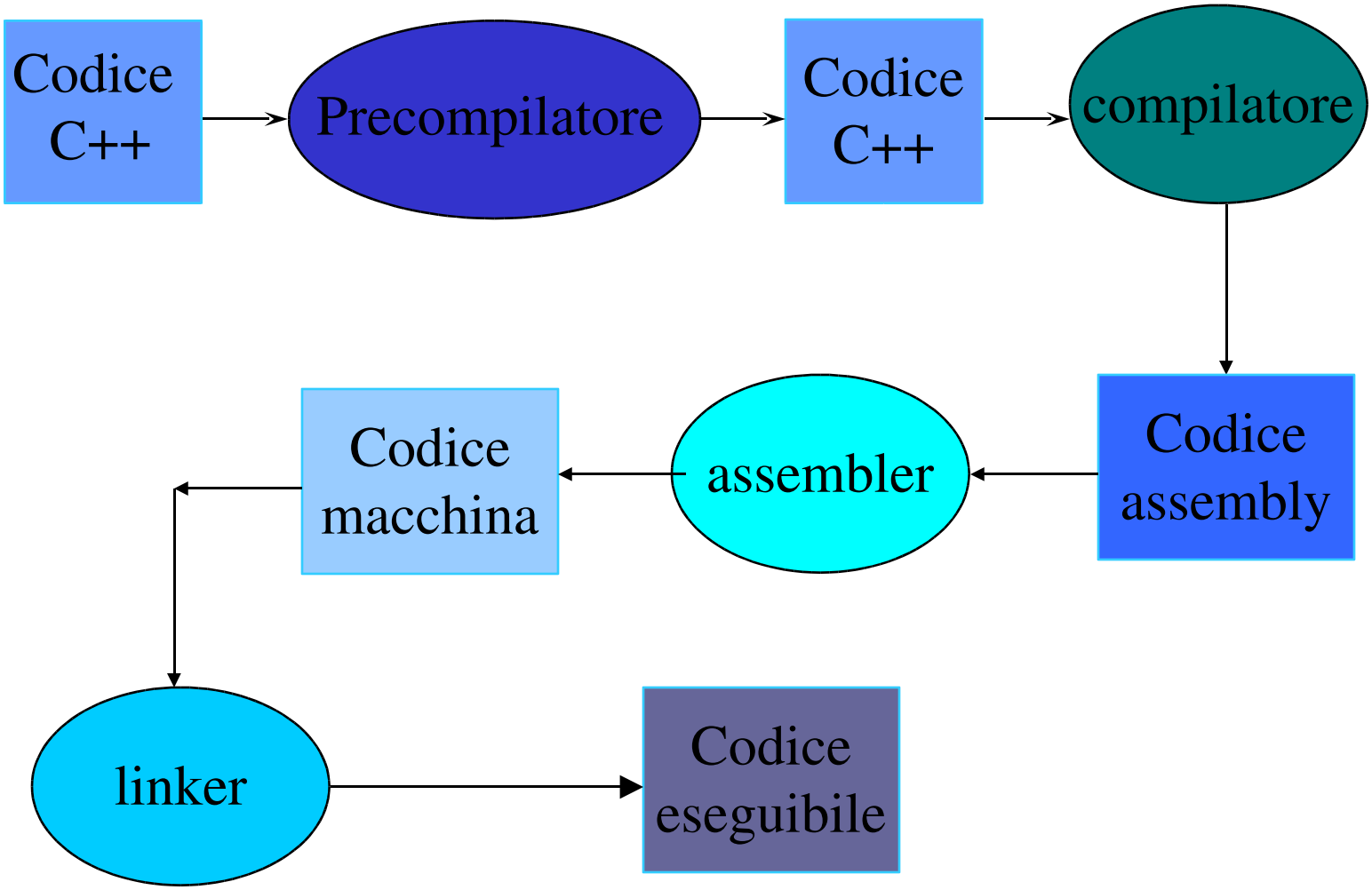

Codice
C++
Precompilatore
Codice
C++
compilatore
Codice
assembly
assembler

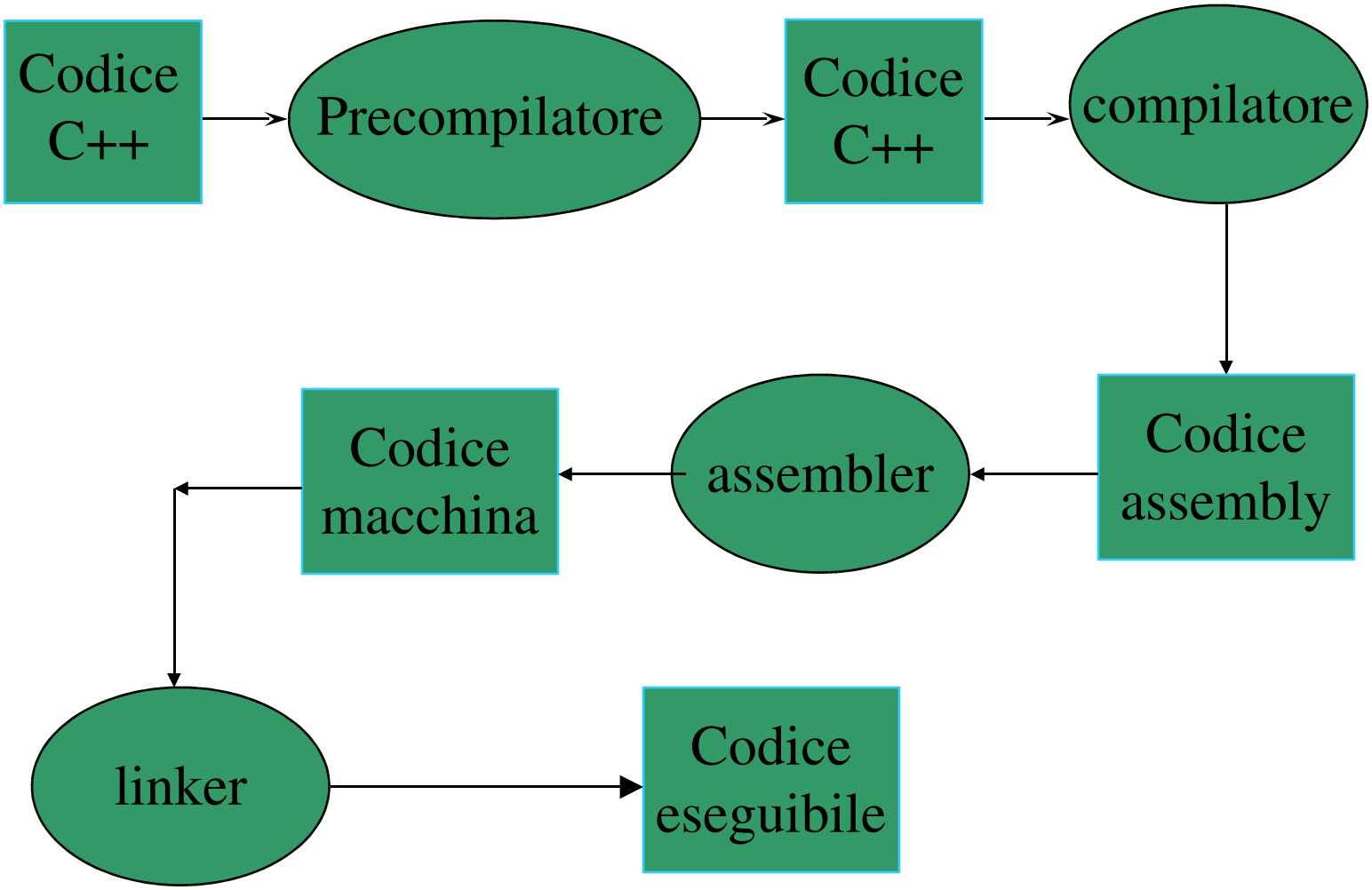

Codice C++
Precompilatore
Codice C++
compilatore
Codice assembly
assembler
Codice macchina

Esempio di codice macchina

```
10011010110111011010011010001011
10011011000001011010011010001111
11111010110001011011111011011001
```


Codice C++
Precompilatore
Codice C++
compilatore
Codice assembly
assembler
Codice macchina
linker


Codice C++
Precompilatore
Codice C++
compilatore
Codice assembly
assembler
Codice macchina
linker
Codice eseguibile


Codice C++
Precompilatore
Codice C++
compilatore
Codice assembly
assembler
Codice macchina
linker
Codice eseguibile

Linker

- Riprendendo il nostro esempio supponiamo di volere visualizzare il risultato finale. Abbiamo già visto che a tale scopo dobbiamo usare la direttiva del precompilatore #include <iostream.h>

  per definire l'oggetto cout e poi dopo avere calcolato z aggiungere l'istruzione

  cout << "risultato " << z << endl;

- Ma dove si trova il codice di ostream? Nella libreria (dall'inglese library: biblioteca) del C++ che contiene il codice macchina per tutti gli oggetti predefiniti.

- Il linker cerca automaticamente il codice cui riferito dall'utente e non presente nel codice sorgente nelle librerie di sistema.

Linker

- E se si volesse usare del codice non di sistema? Per esempio dell'altro codice già scritto dall'utente?

- Il linker accetta in input più di un file e quindi è sufficiente passagli oltre al programma utente anche gli altri file o librerie contenenti il codice che vuole riutilizzare.

Estensione dei file

- Nella maggior parte dei sistemi operativi il nome dei file è costituito da due parti separate da un punto:

  nome.estensione

- Il nome può essere un qualunque identificatore valido, l'estensione in genere identifica il tipo di file e i programmi che accettano file in input generalmente controllano l'estensione per verificarne il tipo.

- Le estensioni dipendono dal sistema operativo e dai programmi applicativi.

- In questo corso useremo l'estensione .cc per i file che contengono codice sorgente C++ (estensioni valide sono anche cxx e cpp).

Uso di g++ (compilatore GNU)

- *g++ esempio.cc*

  applica precompilatore, compilatore,
  assembler e linker e produce l'eseguibile
  a.out

- *g++ esempio.cc -o esempio*

  chiama l'eseguibile esempio

- *g++ esempio.o altrofile.o libreria.a -o esempio*

  invoca il linker, cercando i riferimenti esterni in
  altrofile.o e in libreria.a

Uso di g++ (compilatore GNU)

- Possiamo anche effettuare i singoli passaggi intermedi usando le opzioni di g++ (consultabili con *man g++*)
- *g++ -E esempio.cc*

  invoca solo il precompilatore (e non salva l'output)
- *g++ -S esempio.cc*

  invoca precompilatore e compilatore e salva il file assembly in esempio.s
- *g++ -c esempio.cc*

  invoca precompilatore, compilatore e assembler e salva il file in linguaggio macchina in esempio.o, può agire anche sul file esempio.s, invocando solo l'assembler.

Uso di g++ (compilatore GNU)

Comando	***Azione***	***Files accettati in input***	***File di output***
g++	*Produce l'eseguibile*	*.cc (sorgente)* *.s (assembly)* *.o (linguaggio macchina)*	*a.out,* *modificabile con l'opzione –o*
g++ -c	*Produce il file in linguaggio macchina*	*.cc (sorgente)* *.s (assembly)*	*.o (linguaggio macchina)*
g++ -s	*Produce il file in assembly*	*.cc (sorgente)*	*.s (assembly)*
g++ -E	*Applica il precompilatore*	*.cc (sorgente)*	*Output su schermo*

Uso di bcc
(compilatore Borland per MS DOS)

Comando	***Azione***	***Files accettati in input***	***File di output***
bcc	*Produce l'eseguibile*	*.cpp (sorgente)* *.obj (linguaggio macchina)*	*.exe (stesso nome file in ingresso)* *modificabile con l'opzione –e*
bcc -c	*Produce il file in linguaggio macchina*	*.cpp (sorgente)*	*.obj (linguaggio macchina)*
bcc -S	*Produce il file in assembly*	*.cpp (sorgente)*	*.ASM (assembly)*
cpp	*Applica il precompilatore*	*.cpp (sorgente)*	*Output su file .I*

Per poter compilare un file generato dal precompilatore usare prima: ***cpp -P- sorgente.cpp***

e poi: ***bcc –P sorgente.I*** *(l'opzione –P permette di compilare sorgenti con estensione diversa da .cpp)*