C++ introduzione

Alberto Ferrari

C++

STRUTTURA DI UN PROGRAMMA

Hello world

```
/*
 * First C++ program that says hello (hello.cpp)
 */
#include <iostream>    // Needed to perform IO operations
using namespace std;

int main() {                          // Program entry point
   cout << "hello, world" << endl;    // Say Hello
   return 0;                          // Terminate main()
}
```

namespace

- insiemi di definizioni di nomi (classi, funzioni, costanti)
- le librerie standard mettono i loro nomi nel namespace **std**
- per usare queste definizioni occorre
  - specificare nel codice il nome completo di namespace
    - std::cin …
  - oppure specificare il namespace per il nome con la direttiva
    - using std::cin;
  - oppure includere tutto il namespace con la direttiva
    - using namespace std;

```
#include <iostream>
int main() {
  int integer1, integer2, sum;              // declaration
  std::cout << "Enter first integer\n";     // prompt
  std::cin >> integer1;                     // read an integer
  std::cout << "Enter second integer\n";    // prompt
  std::cin >> integer2;                     // read an integer
  sum = integer1 + integer2;                // assignment of sum
  std::cout << "Sum is " << sum << std::endl; // print sum
  return 0; // indicate that program ended successfully
}
```

namespace per il nome

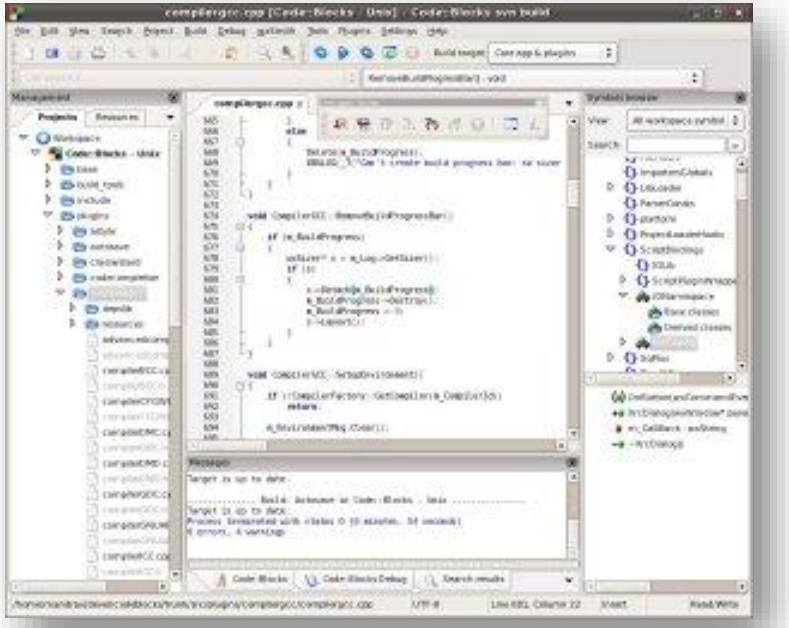
```
#include <iostream>
using std::cout; // program uses cout
using std::cin; // program uses cin
using std::endl; // program uses endl
int main() {
  int integer1, integer2, sum;             // declaration
  cout << "Enter first integer\n";   // prompt
  cin >> integer1;                         // read an integer
  cout << "Enter second integer\n"; // prompt
  cin >> integer2;                         // read an integer
  sum = integer1 + integer2;               // assignment of sum
  cout << "Sum is " << sum << std::endl; // print sum
  return 0; // indicate that program ended successfully
}
```

inclusione del namespace

```
#include <iostream>
using namespace std;
int main() {
  int integer1, integer2, sum;             // declaration
  cout << "Enter first integer\n";         // prompt
  cin >> integer1;                         // read an integer
  cout << "Enter second integer\n";        // prompt
  cin >> integer2;                         // read an integer
  sum = integer1 + integer2;               // assignment of sum
  cout << "Sum is " << sum << std::endl;   // print sum
  return 0; // indicate that program ended successfully
}
```

IDE e compilatore

- Integrated Development Environment
- Code::Blocks
  - open source, cross platform, free C, C++ and Fortran IDE
  - www.codeblocks.org
  - https://www3.ntu.edu.sg/home/ehchua/programming/howto/CodeBlocks_HowTo.html
- GCC, the GNU Compiler Collection
  - gcc.gnu.org
  - licenza copyleft (GNU GPL)



CodeBlocks' Common Errors

- Cannot Compile any C/C++ Program after Installing CodeBlocks. Check:
  - You downloaded the CodeBlocks with "MinGW GNU C/C++ Compiler" (e.g., "codeblocks-10.05mingw-setup.exe").
  - Goto "Settings" menu ⇒ "Compiler..." ⇒ Select tab "Toolchain Executables" ⇒ Check the "Compiler's Installation Directory". It shall be set to the "MinGW" sub-directory of the CodeBlocks installation directory, e.g., "c:\Program Files\codeblocks\MinGW" suppose that CodeBlocks is installed in "c:\Program Files\codeblocks".
- Cannot Build or Run Program - Build/Run Buttons and Menu-Items are Grey and Not Selectable
  - A previous program is still running. You need to terminate the program by closing the output console window.
- Error: undefined reference to `WinMain@16'
  - Check that you have a main() function in your function. Check your spelling of main!

https://www3.ntu.edu.sg/home/ehchua/programming/howto/CodeBlocks_HowTo.html

Geany

- https://www.geany.org/
- *Geany is a small and lightweight Integrated Development Environment. It was developed to provide a small and fast IDE, which has only a few dependencies from other packages.*
- *Geany is known to run under Linux, FreeBSD, NetBSD, OpenBSD, MacOS X, AIX v5.3, Solaris Express and Windows.*
- *The code is licensed under the terms of the GNU General Public Licence.*

linguaggio C++

- linguaggio di programmazione basato sul paradigma orientato agli oggetti
- 1983 Bjarne Stroustrup (Bell Labs)
  - evoluzione del linguaggio C (*C with classes*)
- struttura di C++
  - nucleo del linguaggio
  - libreria standard STL (Standard Template Library)
- versioni:
  - ***C++98***, C++03, C++11, ***C++14*** e C++17

lessico e sintassi

- il lessico è l'insieme dei vocaboli che possono essere utilizzati per la costruzione di un programma è costituito da:
  - parole riservate esempi: main, while, for
  - identificatori esempi: Nomevariabile, PiGreco
  - costanti valori numerici o sequenze di caratteri esempi: 3, 4.515E-9, "Ciao"
  - simboli speciali o di interpunzione esempi: + - / % [ ] ( ) == { }
  - separatori (sono separatori tutti i simboli di interpunzione, lo spazio bianco e il carattere di fine linea (\n)
- commenti
  - /* <testo> */ oppure //testo (fino a fine riga)

identificatori

- linguaggio case-sensitive (differenza tra minuscole e maiuscole)
- <identificatore> ::= {<lettera> | <underscore>} | [{<lettera> | <underscore> | <cifra>}]
- <underscore> ::= _
  - alcuni identificatori corretti:
    - k — meglio usare nomi SIGNIFICATIVI
    - _id — meglio non usare inizio _
    - identificatore_valido2
    - f2_2
    - F2_2 — diverso da precedente
  - alcuni identificatori errati:
    - not#me
    - 101_via
    - for — parola chiave

compilatore gcc

https://gcc.gnu.org/

- GCC (GNU Compiler Collection, in origine GNU C Compiler) è un compilatore multi-target creato inizialmente da Richard Stallman, come parte del progetto GNU
- le versioni recenti sono incluse nelle principali distribuzioni del sistema operativo GNU/Linux e di molti altri sistemi
- nato inizialmente come un compilatore per il linguaggio C, dispone oggi di vari front end per altri linguaggi (Java, C++, Objective C ...)

Il progetto GNU è un progetto collaborativo lanciato nel 1983 da Richard Stallman per creare GNU: un sistema operativo Unix-like completo, utilizzabile esclusivamente utilizzando software libero. Il nome GNU è l'acronimo ricorsivo di "GNU's Not Unix".

versioni del compilatore (gcc –v)

- comando per visualizzare informazioni e versione del compilatore

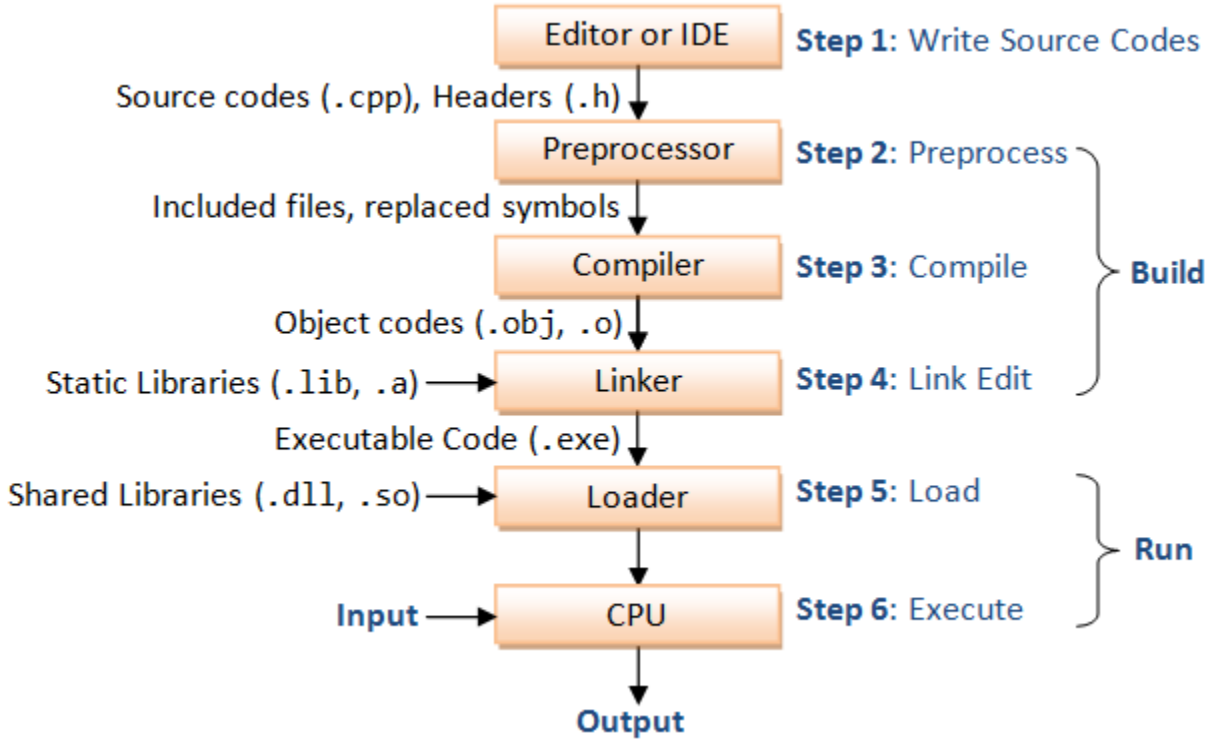
```
alb@alb-UNIPR:~$ gcc -v
Using built-in specs.
COLLECT_GCC=gcc
COLLECT_LTO_WRAPPER=/usr/lib/gcc/x86_64-linux
Target: x86_64-linux-gnu
Configured with: ../src/configure -v --with-p
--with-bugurl=file:///usr/share/doc/gcc-6/REA
++,java,go,d,fortran,objc,obj-c++ --prefix=/u
efix=x86_64-linux-gnu- --enable-shared --enab
r/lib --without-included-gettext --enable-thr
le-nls --with-sysroot=/ --enable-clocale=gnu
ibstdcxx-time=yes --with-default-libstdcxx-ab
disable-vtable-verify --enable-libmpx --enabl
h-system-zlib --disable-browser-plugin --enab
--with-java-home=/usr/lib/jvm/java-1.5.0-gcj-
th-jvm-root-dir=/usr/lib/jvm/java-1.5.0-gcj-6
jvm-exports/java-1.5.0-gcj-6-amd64 --with-arc
sr/share/java/eclipse-ecj.jar --enable-objc-g
or --with-arch-32=i686 --with-abi=m64 --with-
-multilib --with-tune=generic --enable-checking=release --build=x86_64-linux-gnu
 --host=x86_64-linux-gnu --target=x86_64-linux-gnu
Thread model: posix
gcc version 6.2.0 20161018 (Ubuntu 6.2.0-7ubuntu11)
alb@alb-UNIPR:~$
```

```
Microsoft Windows [Versione 10.0.16299.192]
(c) 2017 Microsoft Corporation. Tutti i diritti sono riservati.

C:\Users\alber>gcc -v
Using built-in specs.
COLLECT_GCC=gcc
COLLECT_LTO_WRAPPER=C:/Program\ Files\ (x86)/CodeBlocks/MinGW/bin/../libexec/gcc/mingw32/5.1.0/lto-wrapper.exe
Target: mingw32
Configured with: ../../../src/gcc-5.1.0/configure --build=mingw32 --enable-languages=ada,c,c++,fortran,lto,objc,obj-c++
--enable-libgomp --enable-lto --enable-graphite --enable-libstdcxx-debug --enable-threads=posix --enable-version-specifi
c-runtime-libs --enable-fully-dynamic-string --enable-libstdcxx-threads --enable-libstdcxx-time --with-gnu-ld --disable-
werror --disable-nls --disable-win32-registry --disable-symvers --enable-cxx-flags='-fno-function-sections -fno-data-sec
tions -DWINPTHREAD_STATIC' --prefix=/mingw32tdm --with-local-prefix=/mingw32tdm --with-pkgversion=tdm-1 --enable-sjlj-ex
ceptions --with-bugurl=http://tdm-gcc.tdragon.net/bugs
Thread model: posix
gcc version 5.1.0 (tdm-1)

C:\Users\alber>
```

dalla scrittura del codice all'esecuzione



passi della compilazione

- gcc processa file di input attraverso 4 passi:
  - ***preprocessing***
  - ***compilation***
    - traduzione del codice sorgente ricevuto dal preprocessore in codice assembly
  - ***assembly***
    - creazione del codice oggetto
  - ***linking***
    - combinazione delle funzioni definite in altri file sorgenti o definite in librerie con la funzione main() per creare il file eseguibile

https://www3.ntu.edu.sg/home/ehchua/programming/cpp/gcc_make.html

preprocessing

- rimozione dei commenti
- interpretazioni di speciali direttive per il preprocessore denotate da "#" (normalmente all'inizio del codice)
  - #include - include il contenuto di un determinato file
    - es. #include<cmath>
  - #define
    - macro e costanti
    - #define ELEMENTS 100
    - sostituisce in tutto il programma 100 alla parola ELEMENTS (per convenzione tutto maiuscolo
- esempio di utilizzo - comando cpp (c pre processing)
  - cpp test.cpp testPre.cpp

compilatore

- converte il sorgente in linguaggio macchina
  - oggetti
  - progetti multi file
  - riferimenti/indirizzi non completi
- risultato: file oggetto (.o, .obj etc.)
  - è già linguaggio macchina

linker

- unisce i file oggetto
  - risolve indirizzi/riferimenti
  - associa eventuali librerie esterne
- risultato: eseguibile
  - .exe, .com, nessuna estensione etc.

compilazione e linking

- g++ test.cpp
  - compila il file test.cpp e genera il file eseguibile
    - a.exe (Windows)
    - a.out (Linux)
- ***Windows***
  - g++ -o test.exe test.c
  - con l'opzione -o si specifica il file di output
  - per eseguire il programma
    - test
- ***linux***
  - g++ -o test test.c
  - con l'opzione -o si specifica il file di output
  - per eseguire il programma
  - ./test

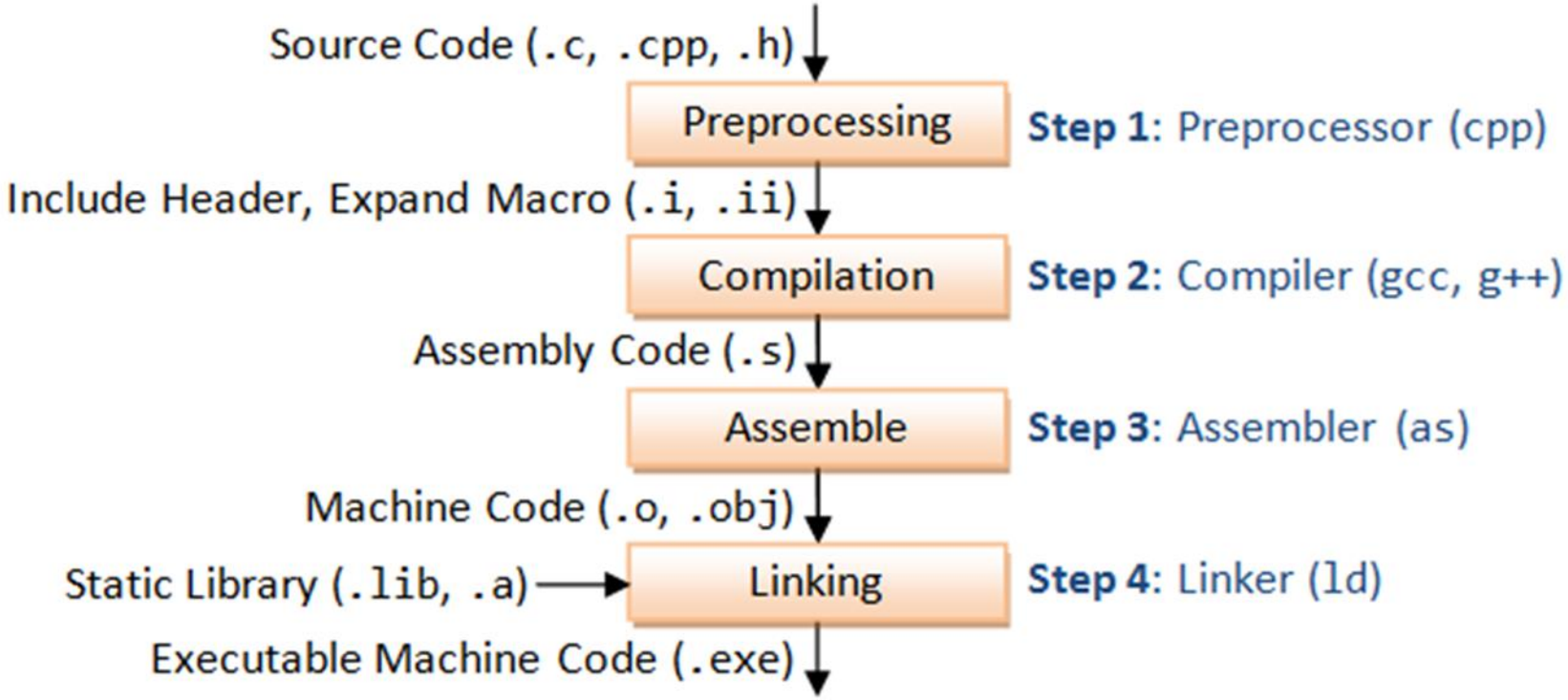
compilazione senza linking

- g++ -c test.c
  - l'opzione -c effettua la compilazione ma non il linking
  - viene generato il file oggetto test.o

error e warning

- ***warning***
  - messaggi di avvertimento (attenzione!)
  - non interrompono la compilazione
  - avvisano della (possibile) presenza di irregolarità nel codice
  - inibire i messaggi di warning (sconsigliato)
    - opzione -w
    - g++ -w test.cpp
  - settare al massimo il livello di warning (consigliato)
    - opzione -wall
    - g++ -wall test.cpp
- ***error***
  - interrompono la compilazione
  - indicano errori che devono essere corretti

passi di sviluppo di un programma C++



passi di sviluppo di un programma C++

- step 1: ***preprocessing***
  - processa le direttive al compilatore (iniziano con #)
    - esempio #include e #define
    - le direttive devono essere processate prima della compilazione
- step 2: ***compile***
  - compilazione e generazione del codice oggetto (.obj, .o)
- step 3: ***link***
  - collega il codice compilato con altro codice compilato e librerie (.lib, .a) e produce il codice eseguibile (.exe)
- step 4: ***load***
  - carica il codice eseguibile in memoria
- step 5: ***run***
  - esegue il codice