Programmazione I

Corso di Laurea in Informatica
a.a. 2017-2018

Dr. Gabriele Tolomei
Dipartimento di Matematica
Università degli Studi di Padova
gtolomei@math.unipd.it

UNIVERSITÀ
DEGLI STUDI
DI PADOVA

12 Dicembre 2017

Cos'è Eclipse?

- Ambiente di Sviluppo Integrato (IDE) open source scritto in Java
- Una piattaforma basata su *plugin* per lo sviluppo software che supporta molteplici linguaggi di programmazione:
  - Java (supporto "nativo")
  - C/C++
  - Perl
  - Python
  - …
- Ciascun plugin fornisce il supporto per uno specifico linguaggio

Sviluppo C/C++ su Eclipse

- Abbiamo visto come scrivere un programma in C++ usando un semplice editor di testo (ad es., **`gedit`**)
- Lo stesso programma può essere scritto con Eclipse che però fornisce strumenti aggiuntivi utili ai fini dello sviluppo (ad es., debugger, makefile, versioning, etc.)
- Ideale per progetti software di grandi dimensioni sviluppati in team
- Tuttavia, alcune di queste funzionalità possono essere utili anche per lo sviluppo di programmi meno complessi

Riferimenti Utili

- Homepage del progetto Eclipse (Oxygen)
- Homepage del progetto Eclipse CDT (C/C++ Development Tooling)
- C/C++ Software Development with Eclipse (testo online in inglese)

NOTA: *Le macchine del laboratorio hanno già un'installazione di Eclipse con plugin CDT per lo sviluppo in C/C++. Questi riferimenti sono però utili per replicare lo stesso ambiente sulle vostre macchine personali*

Installazione di Eclipse

- Eclipse è pensato per essere eseguito sulle principali piattaforme (Unix/Linux, Windows, mac OS)
- Requisito fondamentale per la portabilità: **Java**
  - Più precisamente, **J**ava **R**untime **E**nvironment (**J**ava **D**evelopment **K**it solo se si desidera *sviluppare* in Java)
- Potete scaricare Eclipse (Oxygen) da qui
- Avete **2 opzioni**:
  - Scaricare direttamente la variante C/C++ di Eclipse
  - Scaricare la versione “standard” di Eclipse e aggiungere in seguito il plugin CDT per C/C++
- Istruzioni complete disponibili qui

Toolchains C/C++

- Eclipse non contiene *toolchains* C/C++ (compilatore/linker, make, debugger)
- Per i sistemi **Linux**:
  - Nessuna operazione aggiuntiva: le distribuzioni Linux contengono già questi strumenti (**`gcc`**/**`g++`**, **`make`**, **`gdb`**)
- Per sistemi **mac OS**:
  - Installare i *Command Line Tools* (di Xcode)

```
xcode-select --install
```

- Per sistemi **Windows**:
  - Installare *MinGW* (istruzioni dettagliate qui e qui)

Avvio di Eclipse

- Eclipse può essere avviato tramite interfaccia grafica o da linea di comando (shell)
- Ad es., su Linux:
  - *Applications -> Programming -> Eclipse* (GUI)
  - **`> eclipse`** (shell)

NOTA: *Nel secondo caso, occorre accertarsi che il path di installazione dell'eseguibile di Eclipse (ad es.,* **`/opt/eclipse/eclipse`***) sia presente nella variabile d'ambiente* **`PATH`**

Eclipse: Workspace

- Un **workspace** è una sorta di directory “intermedia” per Eclipse in cui sono mantenute *preferenze* e *impostazioni*
- In un workspace possono coesistere molti **progetti** (l’unico vincolo è che questi devono avere un nome univoco)
- Ogni progetto, a sua volta, è costituito da una serie di **file** e **cartelle**
- È possibile avere più di un workspace ma per i nostri scopi assumeremo di utilizzarne solo uno

Eclipse: Projects

- Eclipse non è semplicemente un editor di testo ma un IDE
- Pertanto, il codice che scriverete dovrà essere inserito all'interno di un **progetto**
- Come già detto, all'interno di un workspace possono coesistere più progetti (a patto che abbiano nomi diversi)

Eclipse: Building

- Nel contesto C/C++, “building” significa *tradurre* il codice sorgente in codice oggetto

compilazione + linking

- In realtà, per progetti complessi significa molto di più:
  - creazione di librerie, DLL, Shared Objects, unit test, etc.
- Utilizza il tool **`make`** (e relativo **`makefile`**)
- Eclipse supporta 2 tipi di builds:
  - **Eclipse Managed** (**`makefile`** creato automaticamente)
  - **Externally Managed** (**`makefile`** creato dal programmatore)

make

- È una *utility* che automatizza gli aspetti tipici che costituiscono la traduzione da codice sorgente a eseguibile
- Si appoggia ad un file (chiamato **`makefile`**) che contiene una serie di “regole” per la creazione del file eseguibile
- Per verificare le opzioni del comando **`make`**:

```
> make --help
```

- Per aprire il manuale completo:

```
> man make
```

Un Passo Indietro…

Codice Sorgente del Programma

```
1  // hello.cpp
2  #include <iostream>
3  using namespace std;
4
5  int main() {
6     cout << "Hello, world!" << endl;
7     return 0;
8  }
```

Compilazione/Linking tramite g++

```
> g++ -o hello.exe hello.cpp
   // Compile and link source hello.cpp into executable hello.exe
> hello
   // Execute under CMD shell
$ ./hello
   // Execute under Bash or Bourne shell, specifying the current path (./)
```

`make`: Esempio di `makefile`

- Il **`makefile`** consiste in una sequenze di **regole**
- Ciascuna regola è composta da **3 parti**:
  - **target**
  - **lista di pre-requisiti** (separati da spazio)
  - **comando**

```
target: pre-req-1 pre-req-2 ...
        command
```

- Target e pre-requisiti sono separati da "**:**"
- Il comando **deve** essere preceduto da un **`TAB`** (no spazi!)

make: Esempio di makefile


```
all: hello.exe

hello.exe: hello.o
	gcc -o hello.exe hello.o

hello.o: hello.c
	gcc -c hello.c

clean:
	rm hello.o hello.exe
```

make: Esempio di makefile


```
all: hello.exe

hello.exe: hello.o
	gcc -o hello.exe hello.o

hello.o: hello.c
	gcc -c hello.c

clean:
	rm hello.o hello.exe
```

target

make: Esempio di makefile


```
all: hello.exe

hello.exe: hello.o
        gcc -o hello.exe hello.o

hello.o: hello.c
        gcc -c hello.c

clean:
        rm hello.o hello.exe
```

target

pre-req

make: Esempio di makefile


```
all: hello.exe

hello.exe: hello.o
        gcc -o hello.exe hello.o

hello.o: hello.c
        gcc -c hello.c

clean:
        rm hello.o hello.exe
```

target

pre-req

comando

`make`: Esecuzione

- Il comando **`make`** invocato in una directory, utilizza il **`makefile`** presente in *quella* directory
- La sintassi del comando è la seguente:

```
> make target
```

Dove ***`target`*** è uno dei target definiti nel **`makefile`**

- Se invocato senza ***`target`***, **`make`** esegue di default il target **`all`** (che si aspetta di trovare nel **`makefile`**):

```
> make
gcc -c hello.c
gcc -o hello.exe hello.o
```

`make`: Esecuzione

- La valutazione di una regola da parte di **`make`** procede come indicato di seguito:
  - Si esamina ciascun file pre-requisito della regola
  - Per ogni pre-requisito cui corrisponde una regola (target) nel **`makefile`**, si tenta di aggiornare quel pre-requisito
- Il comando associato ad una regola viene eseguito solo se il target della regola **non** è aggiornato rispetto ai suoi pre-requisiti

Eclipse: Eclipse Managed Build

- In questa modalità, Eclipse si occupa della gestione dell'intero processo di building
- Eclipse fornisce un'interfaccia grafica per la modifica dei parametri di configurazione
- Il processo di building è guidato da **`make`** (interno o esterno ad Eclipse), utilizzando un **`makefile`** creato *automaticamente* dallo stesso Eclipse

Eclipse: Externally Managed Build

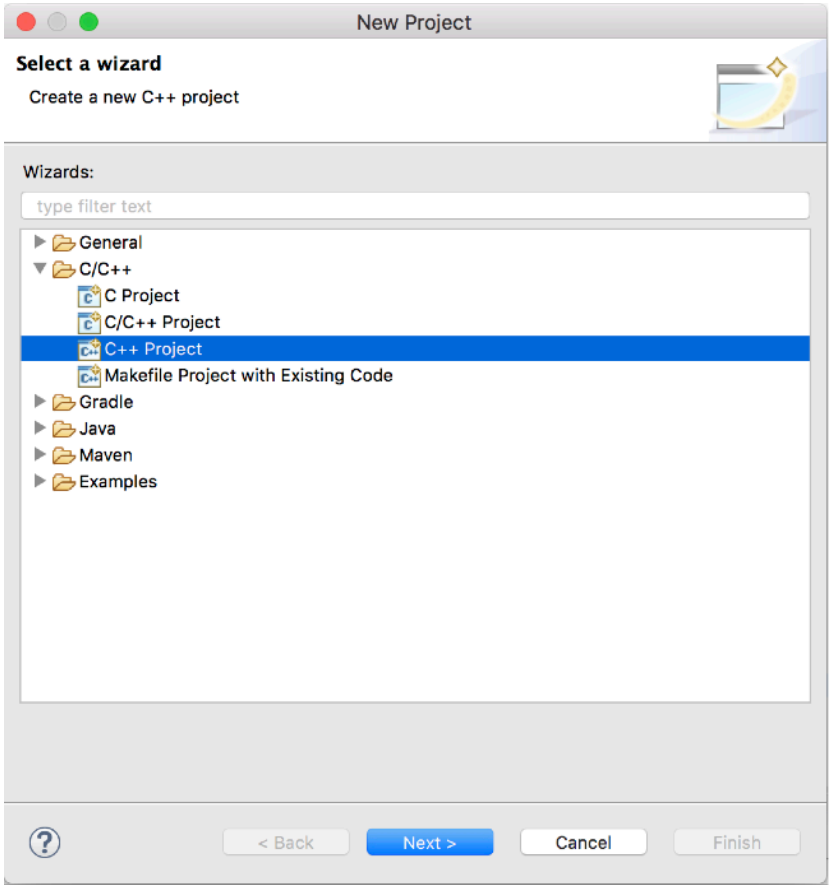
- In questa modalità, Eclipse lascia la responsabilità dell'intero processo di building a tool esterni (ad es., **`make`**)
- È compito del programmatore creare un opportuno **`makefile`** per eseguire il processo di building
- In generale, questa opzione è utile quando:
  - Si vuole importare codice esistente in un nuovo progetto Eclipse
  - Si vuole utilizzare Eclipse solo per la fase di sviluppo

Eclipse: Create New Project

- Per creare un nuovo progetto:
  - Selezionare *File* > *New* > *Project*
  - Selezionare il tipo di progetto (C++)
  - Indicare il nome del progetto (ad es., **HelloWorld**)
  - Specificare la categoria di progetto C++:
    - ***Executable*** (`makefile` creato automaticamente)
    - *Shared Library*
    - *Static Library*
    - *Makefile project* (`makefile` **non** viene creato)

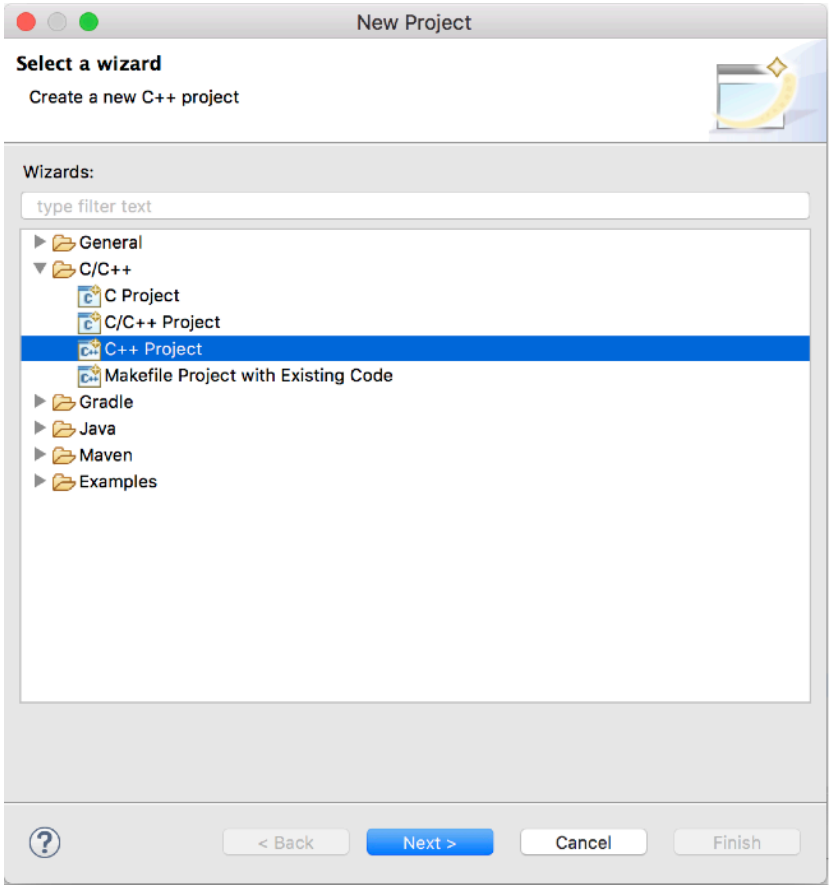
Eclipse: Create New Project

1

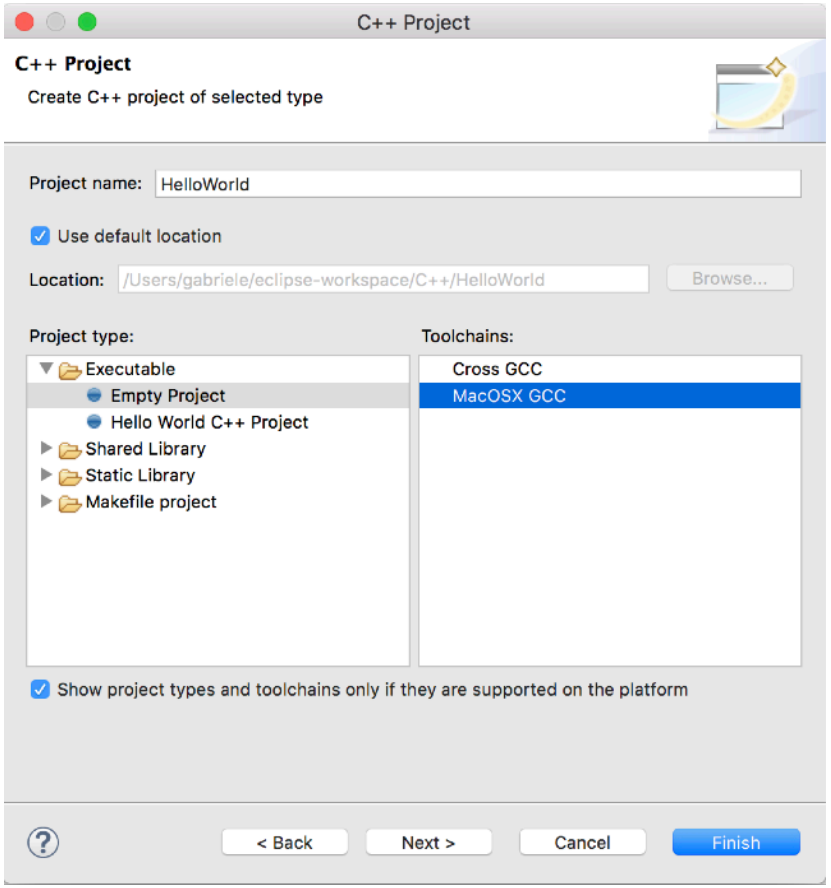


Eclipse: Create New Project

1

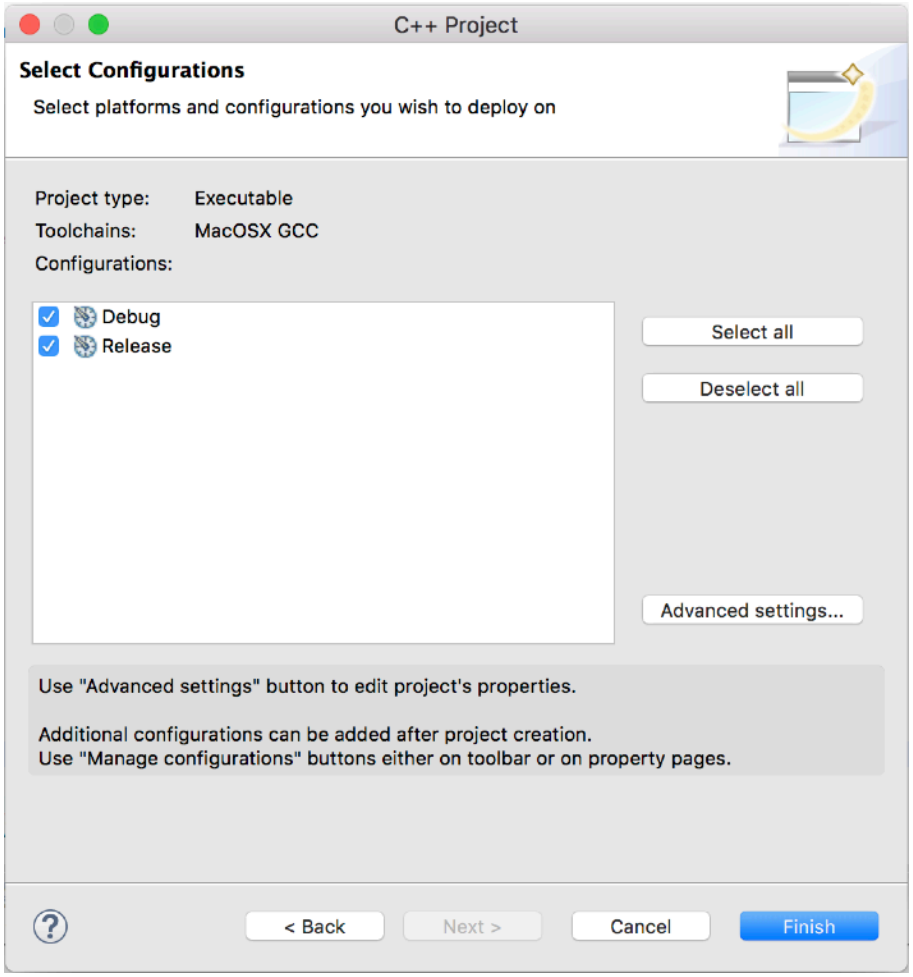


2



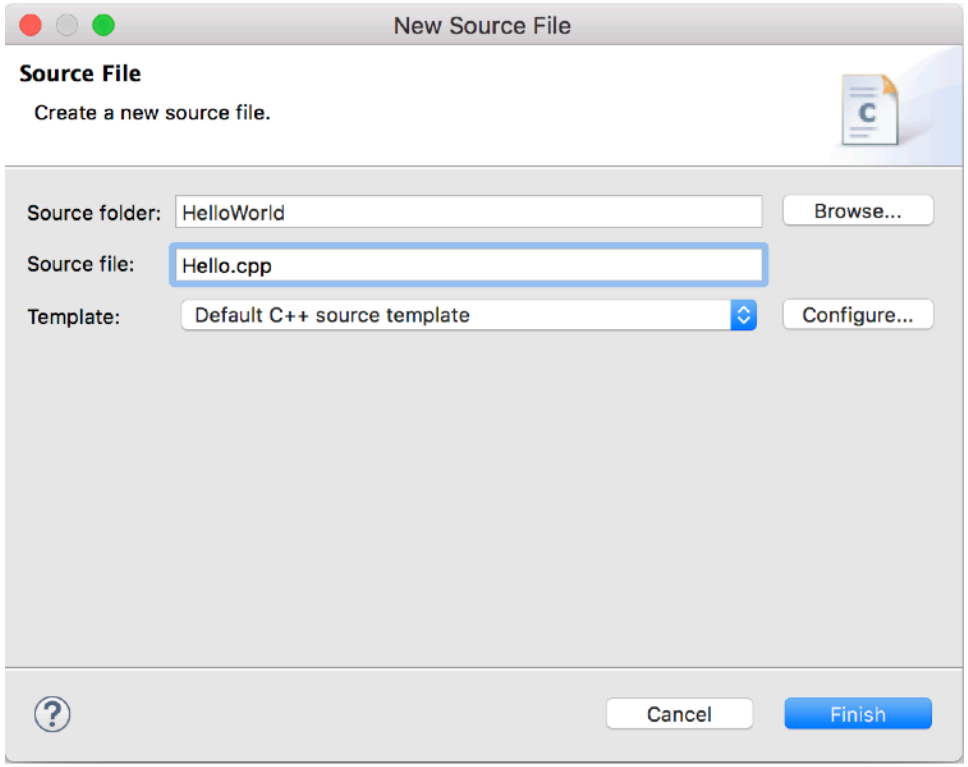
Eclipse: Create New Project

3



Eclipse: Creare File `.cpp`

- Per aggiungere un file **`.cpp`** al progetto creato:
  - Tasto destro sul progetto *New* > *Source File*
  - Specificare il nome comprensivo di estensione (ad es., **`Hello.cpp`**)



Eclipse: Building

- Una volta terminata la scrittura del codice possiamo eseguire il building tramite Eclipse
  - Dal menu principale: *Project* > *Build Project*
  - Questa operazione invoca il comando **`make [all]`** sul **`makefile`** generato automaticamente da Eclipse
- In alternativa, possiamo sempre gestire la compilazione ed il linking utilizzando **`g++`** da linea di comando

```
$ g++ -Wall -g -o Hello.exe Hello.cpp
```

- `-o`: specifies the output executable filename.
- `-Wall`: prints "`all`" warning messages.
- `-g`: generates additional symbolic debuggging information for use with `gdb` debugger.