Parte 13

Documentazione

[Frida Kahlo – Nucleus Of Creation, 1945]

Documentazione

- La **documentazione in campo informatico,** comprende il materiale utile alla comprensione delle **caratteristiche e funzionalità** di un dato sistema, strumento o procedura
- Solitamente diffusa in **formato elettronico o cartaceo** si suddivide in diverse categorie: **guide, manuali, HowTo, FAQ, ...**
- La **documentazione** di un generico software è **generabile automaticamente** a partire dal codice sorgente per mezzo di appositi tool

Doxygen

- **Tool per la generazione automatica**

  http://it.wikipedia.org/wiki/Doxygen
- Progetto **open source** e **multipiattaforma** (Windows, Mac OS, Linux)
- Opera con **diversi linguaggi**: C++, C, Java, Objective C, Python, IDL (versioni CORBA e Microsoft), PHP, C#
- È il sistema di documentazione di gran lunga **più utilizzato** nei grandi progetti open source in C++

Output di Doxygen

- Il risultato finale è disponibile sotto forma di pagine **HTML** (o nei formati RTF, PDF, LaTeX, PostScript o man pages di Unix)
- Il formato **HTML** prodotto offre un sistema di **hyperlink** molto curato che permette al lettore una **agevole navigazione** della struttura dei file sorgenti
- La documentazione prodotta riporta anche il **diagramma delle classi**, nei casi in cui sono presenti relazioni di ereditarietà tra strutture dati
- Diverse lingue disponibili, tra cui l'italiano

Formato dei commenti

- Il funzionamento di **Doxygen** richiede una particolare **formattazione dei commenti** inseriti nel codice sorgente
- Le **regole di formattazione**, oltre ad essere analoghe a quelle degli altri prodotti della categoria, sono chiaramente documentate nel manuale
  - Le approfondiremo di seguito
- Cominciare intanto a vedere il file *documentazione/GSeq.cc*

File di configurazione

- Doxygen associa ad ogni progetto da documentare un **file di configurazione** che contiene le impostazioni da utilizzare per la generazione della documentazione
- Questo file consiste in un elenco di assegnazioni di opportuni valori a determinati **parametri** (**TAG**)
  - Ogni **tag** è formato dalla **coppia di informazioni:**

    NOME_PARAMETRO = VALORE_PARAMETRO

Utilizziamo doxygen

- Creare un **file di configurazione generico** nella directory del progetto
  - Basta eseguire il comando: **doxygen -g** per creare un file di configurazione con nome predefinito (**Doxyfile**)
- Diamo uno sguardo al file generato
- Poi invochiamo **doxygen [config_file]**
  - Occorre specificare il nome del file di configurazione solo se si usa un file con un nome diverso da quello predefinito

Documentazione HTML

- Se tutto è andato bene, **doxygen** dovrebbe aver creato una directory chiamata **html**
- Apriamo il file **index.html** dentro tale directory **html** con un Web browser
- Prendiamoci qualche minuto per navigare la documentazione sul nostro progetto
- **Nota**: c'è anche una cartella **latex**

Nota

- In C++ le **strutture** sono un caso particolare di **classe**
- Pertanto, nella documentazione che generiamo, gli oggetti di tipo **struct** sono elencati come oggetti di tipo **class**
- In C++ si usano gli spazi di nomi (**namespace**), ma nei nostri programmi usiamo solo il **namespace std**

Alcuni problemi

- **Problemi**: la documentazione generata è essenzialmente vuota, in inglese, etc...
- Iniziamo a modificare qualche parametro del file di configurazione **Doxyfile**
- Modificare solo i seguenti parametri:
  - *PROJECT_NAME = “Generazione sequenze”*
  - *OUTPUT_DIRECTORY = doc*
  - *GENERATE_LATEX = NO*
  - *EXTRACT_ALL = YES*
  - *OUTPUT_LANGUAGE = Italian*

File di configurazione (1)

- I doppi apici nel valore del parametro **PROJECT_NAME** servono per non avere problemi nel caso il nome contenga spazi
- Specificare la **OUTPUT_DIRECTORY** è comodo perché altrimenti **doxygen** genera tutto nella directory corrente
- Abbiamo settato **GENERATE_LATEX** a **NO** per evitare di generare anche un manuale in formato latex

File di configurazione (2)

- Doxygen normalmente genera documentazione solo per le parti di codice opportunamente commentate
- Abbiamo invece settato **EXTRACT_ALL** a **YES** per fargli generare documentazione su tutto il codice
  - Comparirà la scheda **File** (a fianco di Classi) attraverso cui si può navigare il codice
- Mediante il parametro **OUTPUT_LANGUAGE** abbiamo settato la lingua italiana

Riproviamo

- Riproviamo dopo aver modificato i parametri in **Doxygen**
  - Rimuoviamo le cartelle html e latex
- Invochiamo

  doxygen [config_file]
- Ora vedete che la **directory html** con la documentazione è stata creata dentro la **directory doc**, come abbiamo indicato nel file di configurazione

Osservazione (1)

- Come si può notare, la documentazione non è altro che un **elenco, più o meno organizzato, degli oggetti presenti** nel nostro programma
- Doxygen **non aggiunge automaticamente alcuna documentazione**
- L'errore più grave nell'usare questo tipo di tool è credere che basti lanciare il tool per ottenere automaticamente **documentazione di qualità**
  - Quello che si ottiene in questo modo è una sorta di documentazione vuota

Osservazione (2)

- Il **contenuto informativo extra** dobbiamo introdurlo noi
- Infatti, la generazione automatica diviene estremamente **vantaggiosa** nel momento in cui arricchiamo il sorgente con dei **commenti opportuni**, come sarà chiaro a breve

Commenti

- I commenti nel formato /* ... */ erano gli unici inizialmente disponibili in C, pertanto vengono spesso chiamati **Cstyle comments**
- Le linee di commento inizianti con // erano disponibili solo in C++ e per questo motivo vengono spesso chiamate **C++ comment lines**

Documentazione del codice

http://www.stack.nl/~dimitri/doxygen/

- Leggere la prima sezione per farsi un'idea dell'applicazione

- Le seguenti slide riportano frammenti della pagina “Documenting the code”:

  http://www.stack.nl/~dimitri/doxygen/docblocks.html

Documentation block (1)

- A **special documentation block** is a C or C++ style comment block with some additional markings, so **doxygen** knows it is a piece of documentation that needs to end up in the generated documentation
- For each code item there are **two** (or in some cases **three**) types of descriptions, which together form the documentation: a **brief description** and **detailed description**, both are **optional**
- As the name suggests, a brief description is a short one-liner, whereas the detailed description provides longer, more detailed documentation

Documentation block (2)

- There are several ways to mark a comment block as a description
- You can use the JavaDoc style, which consists of a C-style comment block starting with two *'s, like this:

  /

  *** ... text ...**

  ***/**

- Oppure anche su una sola linea

  / ... text ... */**

JavaDoc

- Come si deduce dalla precedente slide, l'ambiente di sviluppo **java** dispone di un proprio tool di creazione automatica della documentazione, chiamato **JavaDoc**
- **Doxygen** è più flessibile e permette anche altri formati, ma se impariamo a scrivere i commenti in un formato compatibile con **JavaDoc** siamo già pronti per creare documentazione automatica per codice **java**

Posizione (1)

- Se si utilizza la sintassi vista finora, il **blocco di documentazione** va messo immediatamente prima dell'oggetto da documentare (variabile, funzione, tipo strutturato, ...)
- O prima della dichiarazione o prima della definizione (non entriamo in ulteriori dettagli)

Posizione (2)

- Si può anche porre il blocco **subito dopo l'oggetto da documentare**, a patto di farlo iniziare con /**< anziché con /**
- Es.:

  int maxval ; /**< valore massimo */
- Supponiamo di non generare documentazione per il codice presente nel corpo delle funzioni

Proviamo

- Documentiamo almeno una funzione, una dichiarazione di oggetti di tipo struct, ed una variabile
- Rigeneriamo la documentazione
- Controlliamo il risultato

Descrizione breve (1)

- Come si può verificare, nella pagina delle classi ed in quella dei file della documentazione prodotta da **doxygen**, c'è uno **spazio vuoto** affianco ai nomi delle strutture e dei file
- Tale spazio è destinato ad ospitare una **descrizione breve** dell'oggetto
- In generale, come visto in precedenza si può dare una descrizione breve ed una dettagliata

Descrizione breve (2)

- Vi sono più modi per dare una descrizione breve
- Vediamo nuovamente solo quello compatibile con JavaDoc
- Innanzitutto bisogna settare

  JAVADOC_AUTOBRIEF = YES

  nel file di configurazione

Descrizione breve (3)

- Se

  JAVADOC_AUTOBRIEF = YES

  allora la descrizione breve è data dai caratteri che vanno dall'inizio del blocco di documentazione fino al **primo carattere . seguito da uno spazio o da un newline**

Esempi

```
/** Questa è la descrizione breve. Da
* qui in poi c'è descrizione
* dettagliata ...
*/
```

Oppure

```
/** Questa è la descrizione breve.
*
* Da qui in poi c'è descrizione
* dettagliata ...
*/
```

Proviamo

- Aggiungiamo **documentazione breve e dettagliata** per almeno una funzione
- Rigeneriamo la documentazione
- Verifichiamo il risultato

Inclusione del codice

- Può essere conveniente includere anche il codice sorgente delle funzioni, dichiarazioni, e così via nella documentazione
- Per ottenere questo risultato, settare

  INLINE_SOURCES = YES

  nel file di configurazione

Comandi speciali

- Con doxygen si può realizzare **documentazione professionale, sofisticata e complessa**
  - *Grafi, diagrammi, link, controllo della formattazione* ...
- Vedere ad esempio alcune delle documentazioni generate con **doxygen** e riportate in

  http://www.stack.nl/~dimitri/doxygen/results.html
- Qui analizzeremo solo l'**inserimento di link** ed alcuni **comandi speciali**

Creazione di link (1)

- Quando nella documentazione si menziona una variabile, funzione, tipo di dato, può essere estremamente conveniente che **doxygen generi automaticamente un link** a quell'oggetto nella documentazione stessa
- In questo modo, il lettore può vedere la documentazione dell'oggetto cliccandoci semplicemente sopra

Creazione di link (2)

- Una delle sintassi per ottenere questo risultato è far precedere il nome dell'oggetto dai caratteri **::**
- Es.:

```
/** ...
*
*  Vedere la documentazione della funzione
*  ::main per maggiori dettagli sulle funzionalità e
*  sulla loro implementazione.
*/
```

Comandi speciali

http://www.stack.nl/~dimitri/doxygen/commands.html

Oltre alla sezione introduttiva, vedere i **comandi**:

- *@mainpage*
- *@author*
- *@file*
- *@param*
- *@return*

Sostituire \ con @
Esempio:
\file con @file

In JavaDoc i comandi posso iniziare solo con **@**

NOTA

- Il comando **@author** può dare problemi:
  - Difficoltà di terminare la lista degli autori all'interno di un blocco di documentazione
- **Soluzione possibile**: metterlo in fondo al blocco di documentazione

Esercizio

- Far includere il codice sorgente
- Far inserire almeno un link da **doxygen**
- Utilizzare almeno una volta ciascuno dei comandi speciali precedentemente elencati
- Rigenerare la documentazione
- Controllare il risultato

Programma completo

- Documentare brevemente tutto il codice del progetto
- Esempio di progetto documentato:

  Gseq_1file_complete.cc
- Generare la documentazione
- Apportare le modifiche al file **Doxyfile** come visto nelle slide
- Confrontare la documentazione con quella prodotta precedentemente

Vantaggi ed equivoci (1)

- Uno dei **principali vantaggi** dell'uso di **doxygen** è che permette di generare in modo automatico documenti che illustrano il contenuto e la struttura di un programma
- MA è importante avere chiaro che per documentare realmente il codice sono fondamentali le **informazioni inserite esplicitamente** nei blocchi di documentazione

Vantaggi ed equivoci (2)

- Un notevole vantaggio dell'uso di strumenti come **doxygen** è che la fonte della documentazione (da scrivere comunque manualmente) si trova insieme al codice
  - Molto più **immediato e semplice aggiornare la documentazione mentre il codice cambia**
- Inoltre**, doxygen** riporta **automaticamente** nella documentazione anche la nuova versione del codice se questo cambia