

# Parte 8 – Sviluppo e testing

## 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**  
<https://www.doxygen.nl/index.html>
- 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 **Doxxygen** 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

# File di configurazione

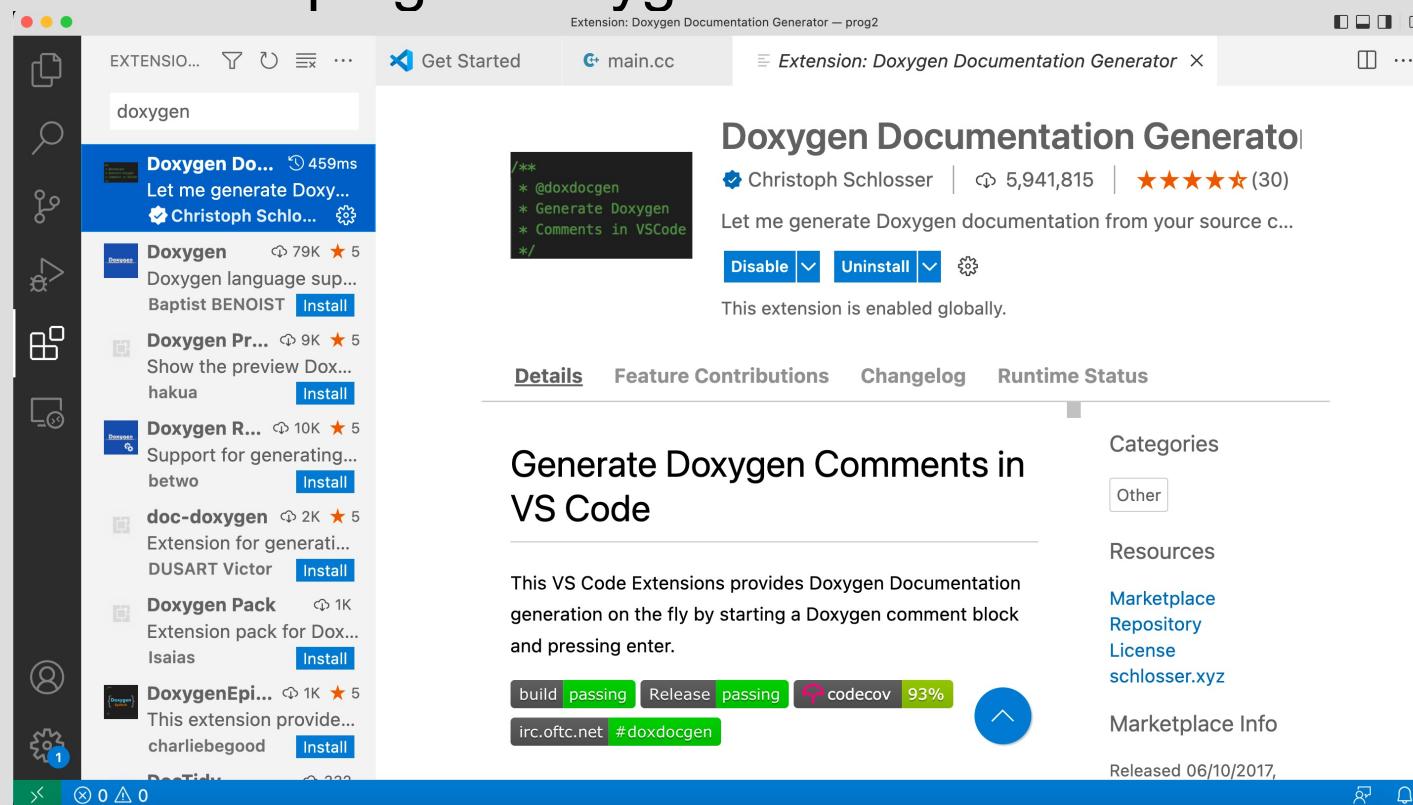
- Doxygen associa ad ogni progetto da documentare un **file di configurazione** che contiene le impostazioni da utilizzare per la generazione della documentazione
- Questo file consiste in un elenco di assegnazioni di opportuni valori a determinati **parametri (TAG)**
  - Ogni tag è formato dalla **coppia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

# Come usare doxygen

- Da linea di comando (comando **doxygen**)
- In VS Code attraverso un plugin (**Doxxygen Documentation Generation**)
- Una soluzione non preclude l'altra
- Si usa sempre un file di configurazione per la documentazione del progetto
- E si invoca doxygen usando questo file di configurazione

# Installiamo Doxygen e il plugin di VS Code

- Installiamo Doxygen <https://www.doxygen.nl/download.html>
- Su VS Code andiamo sull'installazione dei plugin
- Digitiamo doxygen
- Installiamo il plugin Doxygen Documentation Generator



# Usiamo il plugin

- Come prima cosa dobbiamo creare un file di esempio per doxygen
- Questo vuol dire usare l'opzione **-g** quando chiamiamo doxygen per la prima volta
- Invochiamo quindi
  - **doxygen -g**
- Dovremmo notare che viene creato un file chiamato **«Doxyfile»** nella cartella del nostro progetto

# II doxyfile

```
189 SHORT_NAMES          = NO
190
191 # If the JAVADOC_AUTOBRIEF tag is set to YES then doxygen will interpret the
192 # first line (until the first dot) of a Javadoc-style comment as the brief
193 # description. If set to NO, the Javadoc-style will behave just like regular Qt-
194 # style comments (thus requiring an explicit @brief command for a brief
195 # description.)
196 # The default value is: NO.
197
198 JAVADOC_AUTOBRIEF      = NO
199
200 # If the JAVADOC_BANNER tag is set to YES then doxygen will interpret a line
201 # such as
202 # /*****
203 # as being the beginning of a Javadoc-style comment "banner". If set to NO, the
204 # Javadoc-style will behave just like regular comments and it will not be
205 # interpreted by doxygen.
206 # The default value is: NO.
207
208 JAVADOC_BANNER         = NO
209
```

# Il doxyfile

- E' un elenco di coppie chiave valore
- Nell'esempio precedente:
  - Il campo «SHORT\_NAMES» è settato a **NO**
  - Il campo «JAVADOC\_AUTOBRIEF» è settato a **NO**
  - Il campo «JAVADOC\_BANNER» è settato a **NO**
- Noterete che esistono moltissime opzioni
- Ognuna di queste opzioni configura il modo in cui andremo a creare la nostra documentazione.

# Utilizziamo doxygen

- Possiamo modificare tutta una serie di parametri (che vedremo in seguito)
- Come prima cosa, indichiamo al plugin dove andare a leggere il nostro file «Doxyfile»

## Preferences > Settings

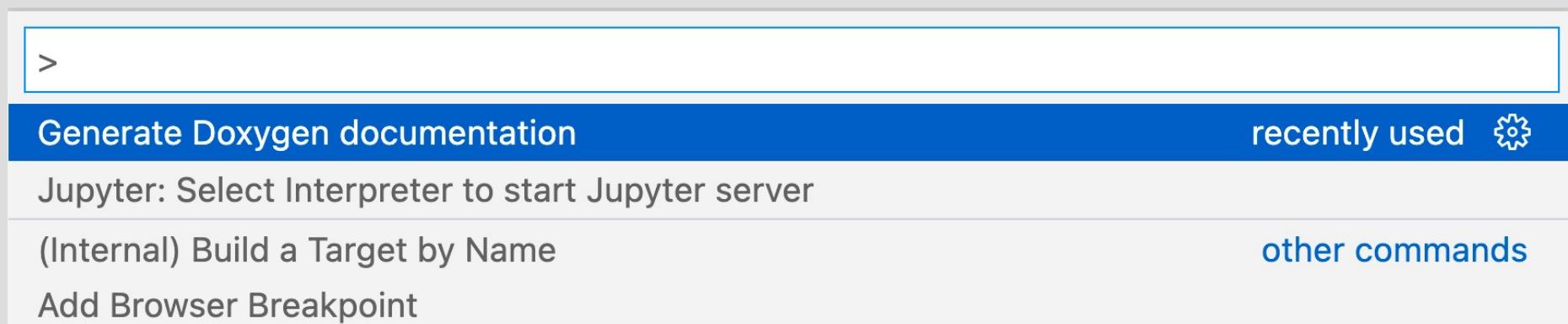
The screenshot shows the VS Code settings interface. A search bar at the top contains the text "doxygen". To the right of the search bar is a button labeled "39 Settings Found" with a three-line menu icon. Below the search bar are two tabs: "User" and "Workspace", with "User" selected. On the far right, there is a blue button labeled "Turn on Settings Sync". The main area displays a list of settings under the "User" tab. One setting is expanded, showing its details. The expanded setting is "Doxygen\_runner: Configuration\_file\_override", which is described as follows:

**Doxygen\_runner: Configuration\_file\_override**  
If set, disable crawling the workspace for Doxygen configuration and use this path.  
You can use \${workspaceFolder} here. The variable will be replaced with all folders added to this workspace.  
(In case there are multiple matches, an error will be shown.)

A code snippet box at the bottom of the expanded setting shows the value: \${workspaceFolder}/Doxyfile

# Utilizziamo doxygen

- Ora possiamo generare la nostra documentazione
- Apriamo la lista dei comandi (CTRL+SHIFT+P, CMD+SHIFT+P) e digitiamo Doxygen



- Se volessi invocare da terminale, basterebbe digitare **doxygen**

# Documentazione HTML

## My Project

Main Page

Classes ▾

Files ▾

Search

### Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

 adj\_node

 elem

 graph

Generated by  1.8.18

La documentazione (HTML) appena generata per  
un progetto

Le struct vengono inserite nella documentazione  
come class

# 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* = “Cronologia Web”

*OUTPUT\_DIRECTORY* = doc

*GENERATE\_LATEX* = NO

*EXTRACT\_ALL* = YES

# 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

# Riproviamo

- Riproviamo dopo aver modificato i parametri in **Doxygen**
  - Rimuoviamo le cartelle html e latex
  - Rigeneriamo la documentazione
  - Dovremmo avere la **directory html** con la documentazione 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**

# Documentation block (1)

- Un blocco di documentazione speciale è un blocco in stile C o C++ più commenti particolari comprensibili per doxygen
- Per ogni pezzo di codice ci sono almeno 2 tipi di descrizione, quella **breve** e quella **dettagliata** (opzionali)
- Per far capire a doxygen che stiamo scrivendo una descrizione possiamo mettere 2 asterischi al commento

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

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

# JavaDoc

- L'ambiente di sviluppo **java** dispone di un proprio tool di creazione automatica della documentazione, chiamato **JavaDoc**
- **Doxxygen** 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 `/**`

## ESEMPIO

```
int maxval ; /**< valore massimo */
```

Supponiamo di non generare documentazione per il codice presente nel corpo delle funzioni

# 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** di fianco 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

In questo caso 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 ...  
 */
```

# 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

# 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 ::

## ESEMPIO

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

# Alcuni comandi

*@mainpage*

*@author*

*@file*

*@param*

*@return*

**Sostituire \ con @**  
*Esempio:*  
\file con @file

In JavaDoc i comandi posso iniziare solo con @

# 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