UNIKASSEL VERSITÄT

Intelligent Embedded Systems

Intelligent Embedded Systems – Prof. Dr. B. Sick

Doxygen

Beispiel eines Dokumentationsgenerators in C

MAREK HERDE
marek.herde@uni-kassel.de
Universität Kassel, FB 16, FG IES
28. Mai 2019

1 Worum es geht – Motivation

Eine Kernkompetenz eines Informatikers/einer Informatikerin ist das Schreiben von Quelltext zur Entwicklung von Software. Eine nicht minder wichtige Kompetenz ist jedoch auch das Erstellen einer entsprechenden *Softwaredokumentation*. Softwareprogramme, die nicht hinreichend kommentiert werden, sind von geringem Nutzen. Einerseits leidet die Qualität der Entwicklung (insbesondere im Team), andererseits ist die Verwendung des finalen Produkts nur nach intensiver Einarbeitung möglich. **Dementsprechend ist eine gründliche Softwaredokumentation unabdingbar für den Erfolg eines entwickelten Softwareprogramms.**

1.1 Klassen von Softwaredokumentationen

Softwaredokumentationen lassen sich in verschiedene Klassen unterteilen, wobei sich die Klassifikation meist nach der angedachten Leserschaft richtet, sodass man zwischen folgenden Klassen differenziert [Wik].

- Die *Methodendokumentation* beschreibt die Grundlagen der Software aus Sicht der Anwender. Beispielsweise wird auf mathematische Verfahren verwiesen, die als Funktionalität der Software zu implementieren sind.
- Um potenziellen Nutzer die Benutzung der entwickelten Software zu ermöglichen, gibt es in der Regel eine *Installationsdokumentation*, welche die Hard- und Software-Anforderung zur Installation der entwickelten Software definiert.
- Da viele Softwareprogramme Operationen auf Daten ausführen, ist eine Erklärung der Struktur dieser Daten essenziell und wird in der *Datendokumentation* festgehalten.
- Damit man schwerwiegende Fehler im Quelltext eines Softwareprojekts frühzeitig entdeckt, ist das Schreiben ausgiebiger Tests erforderlich. Sie überprüfen die Funktionalität des Quelltext und identifizieren mögliche Fehler. Eine Übersicht über diese Tests findet man in der *Testdokumentation*.

- Zum Nachvollziehen des Fortschritts bezüglich der festgehaltenen Ziele und Anforderungen an eine Software wird eine *Entwicklungsdokumentation* erstellt, welche Erklärungen zu den verschiedenen Stadien bzw. Versionen der Software beinhaltet.
- Ein zentrales Element einer jeden Softwaredokumentation ist die sogenannte *Benutzerdokumentation*, die dem Endnutzer/der Endnutzerin die Benutzung der entwickelten Software auf möglichst verständliche Weise näher bringt.
- Damit der Quelltext zu jederzeit nachvollziehbar ist, wird eine *Programmiererdokumentation* angelegt und fortlaufend aktualisiert.

Insbesondere das Erstellen einer Programmiererdokumentation ist aus Sicht eines Informatikers/einer Informatikerin von großer Bedeutung. Sie erleichtert die Softwareentwicklung im Team und die Einarbeitung in komplizierten Quelltext. Obwohl es sich bei der Programmiererdokumentation um eine technische Dokumentation handelt, kann sie auch als Referenz für den Endnutzer/die Endnutzerin dienen. Dies ist insbesondere der Fall bei Entwicklung von Software-Bibliotheken. Neben aussagekräftigen Namen für Variablen und Funktionen ermöglichen sinnvolle Kommentare im Quelltext ein besseres Verständnis. Sinnvoll bedeutet in diesem Zusammenhang, dass ausschließlich Anweisungen im Quelltext kommentiert werden, die nicht auf den ersten Blick verständlich sind. Falls man die Kommentare in einem bestimmten Format verfasst, lässt sich mithilfe von Dokumentationswerkzeugen automatisch eine Übersicht über den Quelltext des Softwareprojektes erstellen.

1.2 Doxygen

Eines der bekanntesten Dokumentationswerkzeugen ist *Doxygen* [vHa], welches frei verfügbar (GNU General Public Licence, Version 2 [Fou]) ist. Im Bereich der C++ Programmierung ist Doxygen der Standard zur automatischen Generierung von Dokumentationen anhand von kommentiertem Quelltext. Nichtsdestotrotz unterstützt es ebenfalls weitere Programmiersprachen wie zum Beispiel Java, Python, Objective-C, C# und C. Dieser Artikel fokussiert sich auf die zuletzt genannte Programmiersprache. Um den Einstieg in Doxygen für C möglichst verständlich zu gestalten, wird in den folgenden Abschnitten Doxygen anhand einer Dokumention für ein Beispielprojekt in C erklärt.

2 Anwendungsbeispiel

Da man als Studierender der Informatik des Öfteren aufwändige mathematische Rechnungen durchführen muss, wird das Erstellen einer Dokumentation mit Doxygen am Beispiel eines wissenschaftlichen Rechenprogramms beschrieben. Mithilfe einer entsprechenden Dokumentation wird der Aufbau eines solchen Programms ersichtlich, sodass mögliche Erweiterung ohne zeitintensive Einarbeitungen implementiert werden können. Die gesamte Dokumentation wird dabei in der englischen Sprache verfasst, da sie der Standard in der Softwareentwicklung ist. Die Hauptseite der mit Doxygen erstellten Dokumentation für das

Rechenprogramm ist dargestellt in Abbildung 1. Sie dient als erster Überblick über das Projekt.

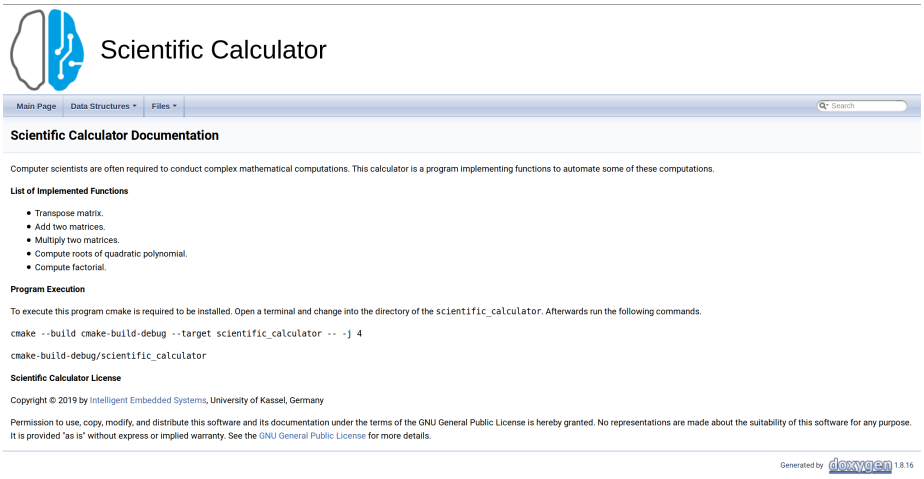


Abbildung 1: HTML Darstellung der Hauptseite des wissenschaftlichen Rechenprogramms liefert einen ersten Überblick über das Projekt, indem dessen Zweck beschrieben wird. Des Weiteren wird die Nutzung erläutert sowie die Lizenz definiert.

3 Hands-on-Tutorial

Um eine Dokumentation, wie in Abbildung 1 gezeigt, mit Doxygen zu generieren, werden im Folgenden die einzelnen Schritte im Detail beschrieben.

3.1 Installation

Bevor Doxygen verwendet werden kann, muss man selbiges natürlich zunächst installieren. Die exakte Prozedur der Installation hängt dabei von dem zugrundeliegendem Betriebssystem ab. Eine entsprechende Anleitung für die verschiedenen Systeme ist unter [vHb] verfügbar. Um die volle Funktionalität von Doxygen ausschöpfen zu können, wird in [vHb] die Installation weiterer Softwarepakete empfohlen. Allen voran sollten LaTeX [Lam94] zur Generierung von Dokumentationen im PDF Format und *Graphviz* [EGK+03] zur Darstellung von Aufruf-Graphen zwischen den Modulen eines Projektes installiert werden.

3.2 Erstellen einer Konfigurationsdatei

Nachdem man Doxygen installiert hat, kann man im Projekt eine sogenannte Konfigurationsdatei anlegen. Dazu führt man im entsprechendem Projektverzeichnis folgenden Befehl in der Konsole aus.

```
doxygen -g doxyfile
```

Nach Ausführung dieses Befehls existiert in dem Projektverzeichnis eine Konfigurationsdatei mit dem Namen `doxyfile`. Sie ist zentraler Bestandteil einer jeden Dokumentation, die mit Doxygen generiert wird. Letztendlich ist diese Datei eine Liste von Zuweisungen, über die man verschiedenste Einstellungen an der Dokumentation vornehmen kann. Die einzelnen Optionen sind dabei in der Datei selbst ausführlich erklärt und einige grundlegende Optionen werden noch im Verlauf dieses Hands-on-Tutorial thematisiert.

3.3 Generieren der Dokumentation

Auf Basis der zuvor angelegten Konfigurationsdatei kann nun eine erste Version der Dokumentation mittels Doxygens generiert werden. Dazu ist lediglich die Ausführung des folgenden Befehls auf Verzeichnisebene der Konfigurationsdatei notwendig.

```
doxygen doxyfile
```

In Abhängigkeit von der Konfigurationsdatei werden dann diverse Ordner angelegt. Standardmäßig und sofern eine LaTeX-Distribution vorhanden ist, werden zwei Ordner namens `html` und `latex` erstellt. Wie ihre Namen bereits suggerieren, befindet sich im Ordner `html` die HTML-Version der Dokumentation und im `latex` Ordner die entsprechenden LaTeX-Dateien, aus denen man eine PDF bauen kann.

Beim Ausführen des Befehls `doxygen` wird im aktuellen Verzeichnis nach entsprechenden Quelldateien gesucht, die anhand ihrer Dateiendung identifiziert werden. Bei dem betrachteten Beispiel eines in C verfassten Rechenprogramms sind das insbesondere C-Quell- und C-Header-Dateien (`.c` bzw. `.h` als Dateiendung). Zur Optimierung der Dokumentation für ein C basiertes Projekt, bietet sich an in der Konfigurationsdatei folgende Einstellung vorzunehmen.

```
OPTIMIZE_OUTPUT_FOR_C = YES
```

Das in diesem Artikel betrachtete Rechenprogramm besteht aus den drei Dateien `calculator.c`, `math_library.h` und `math_library.c`. Aus selbigen Dateien werden dann spezielle Kommentare extrahiert und entsprechend in der Dokumentation eingefügt. Die Syntax dieser für Doxygen optimierten Kommentare ist simpel und wird in den folgenden Abschnitten erklärt.

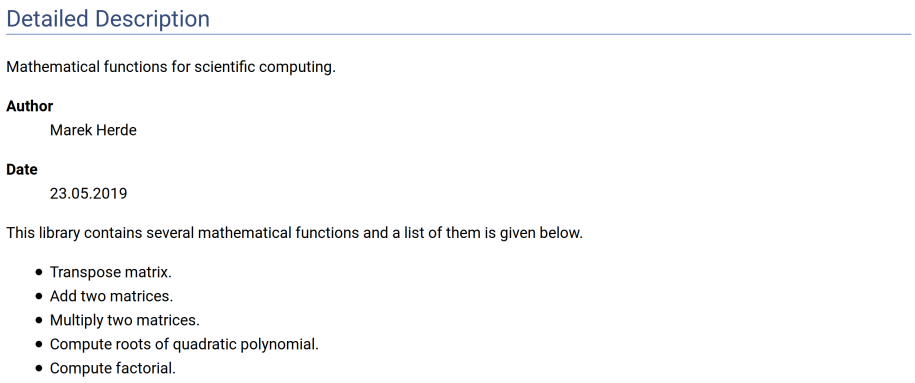
3.4 Kommentieren des Quelltextes

Damit Doxygen die entsprechenden Entitäten eines C-Projektes finden kann, bspw. Strukturen und Funktionen, nutzt man spezielle Kommentarblöcke. Diese werden direkt vor den zu dokumentierenden Entitäten definiert und beschreiben die wichtigsten Eigenschaften selbiger. In der C-Programmierung stehen verschieden Typen dieser Blockkommentare zur Verfügung, wobei der wohl gängigste Type *Javadoc* genannt wird. Man definiert einen solchen Kommentar durch einen zusätzlichen Asterisk (*) beim Einleiten eines C-Blockkommentars.

```
\**
 * ... text ...
 *\
```

Basierend auf solchen Blockkommentaren kann man nun Kommentare zu jeglichem Quelltext verfassen und somit in die Dokumentation integrieren.

3.4.1 Dateien



Detailed Description

Mathematical functions for scientific computing.

Author

Marek Herde

Date

23.05.2019

This library contains several mathematical functions and a list of them is given below.

- Transpose matrix.
- Add two matrices.
- Multiply two matrices.
- Compute roots of quadratic polynomial.
- Compute factorial.

Abbildung 2: Ausschnitt aus der HTML Darstellung der Datei `math_library.h`.

Am direkten Anfang einer `.c` bzw. einer `.h` Datei befindet sich oft ein einleitender Blockkommentar, der die wesentlichen Information über die entsprechende Datei beinhaltet. Dazu gehört eine kurze Beschreibung ihres Zwecks, der Autor sowie das Datum, an dem die Datei erstellt wurde. Als Beispiel ist der Blockkommentar für die Datei namens `math_library.h` im Folgenden gegeben.

```
/**
 * @file math_library.h
 * @author Marek Herde
 * @date 23.05.2019
 * @brief Mathematical functions for scientific computing.
 *
 * @details This library contains several mathematical functions and a list of
 * them is given below.
 *
 *  - Transpose matrix.
 *  - Add two matrices.
 *  - Multiply two matrices.
 *  - Compute roots of quadratic polynomial.
 *  - Compute factorial.
 */
```

Auffällig sind die @-Symbole, die sogenannte *Tags* definieren. Ein solcher Tag verleiht dem Blockkommentar Struktur und trifft eine spezifische Aussage. Der Tag `@file` kommuniziert, dass sich der Blockkommentar auf eine komplette

Datei bezieht. Die Funktion der Tags ist bereits an ihren Namen ersichtlich, so definieren die Tags `@author` und `@date` den Autor und das Datum. Ein zentraler Tag ist `@brief`, welcher eine kurze Erklärung zum Zweck der Datei liefert. Hingegen kann man mit dem Tag `@details` noch genauer auf den Inhalt einer Datei eingehen. Ein Ausschnitt des Resultats in der HTML Version der Dokumentation ist in Abbildung 2 dargestellt.

3.4.2 Strukturen

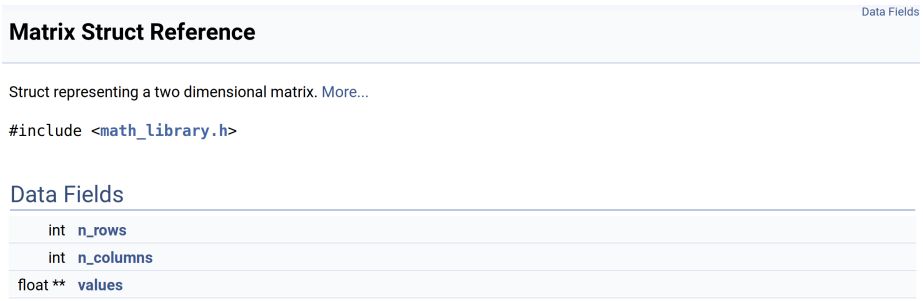
Ein wichtiger Bestandteil der Sprache C sind Strukturen. Diese sollten dementsprechend dokumentiert werden. Das betrachtete Rechenprogramm beinhaltet insgesamt zwei Strukturen. Eine Struktur bildet komplexe Zahlen ab, während die andere zur Repräsentation von Matrizen genutzt wird. Der Programmtext inklusive Blockkommentar zur Struktur `Matrix` hat folgende Form.

```
/**
* @struct Matrix
*
* @brief Struct representing a two dimensional matrix.
*
* @var Matrix::n_rows
* Member 'n_rows' represents the number of rows of the matrix.
* @var Matrix::n_columns
* Member 'n_columns' represents the number of columns of the matrix.
* @var Matrix::values
* Member 'values' is a double pointer containing the values of the matrix.
*
* @see https://en.wikipedia.org/wiki/Matrix_(mathematics) (last access: 23.05.2019)
*/
struct Matrix {
        int n_rows;
        int n_columns;
        float **values;
};
```

Analog zu Dateien gibt es einen Tag `@struct`, der kennzeichnet, dass es sich um eine Struktur handelt und dessen Namen bekannt macht. Des Weiteren wird mittels des Tags `@brief` die Funktion der Struktur beschrieben. Um die Felder/Variablen einer Struktur zu erläutern, wird der Tag `@var` verwendet. Auf diesen Tag folgt direkt der Name des Felds in der Syntax `struct_name::field_name` sowie die entsprechende Erklärung. Der letzte Tag des Blockkommentars, namentlich `@see`, kann in jedem Blockkommentar verwendet werden und dient zur Integration nützlicher Quellen bzw. Erklärungen zu Implementierungen. Ein Ausschnitt des HTML Resultats ist in Abbildung 3 dargestellt.

3.4.3 Funktionen

In der Programmiersprache C sind Funktionen eine Gruppe von Anweisungen, die zusammen eine Aufgabe erfüllen. Da diese Funktionen in der Regel Parameter übergeben bekommen und ein Resultat zurückliefern, ist die exakte



Data Fields

Matrix Struct Reference

Struct representing a two dimensional matrix. More...

`#include <math_library.h>`

Data Fields

int	n_rows
int	n_columns
float **	values

Abbildung 3: Ausschnitt aus der HTML Darstellung der Struktur `Matrix`.

Definition dieser Parameter und des Rückgabewertes zur Benutzung der einer Funktion essenziell. Der folgende Quelltext ist aus der Datei `math_library.h` und zeigt beispielhaft die Dokumentation einer Funktion, die die Diskriminante eines quadratischen Polynoms berechnet.

```
/**
* @brief Computes the discriminant of a polynomial
* function of degree two.
*
* @details Polynomial function of degree two is defined by \f$f(x) = ax^2 + bx + c\f$.
* The corresponding discriminant is given by \f$b^2 − 4ac\f$.
*
* @param a coefficient of polynomial of second order
* @param b coefficient of polynomial of first order
* @param c coefficient of polynomial of zeroth order
*
* @return discriminant of polynomial function
*
* @see https://en.wikipedia.org/wiki/Discriminant
*/
float poly_discriminant(float a, float b, float c);
```

Erneut gibt es eine kurze (`@brief`) sowie eine etwas ausführlichere (`@details`) Erklärung. Die detailliert Erklärung enthält zudem zwei Formeln, wobei der Beginn und das Ende einer Formel durch \f$ gekennzeichnet ist. Die Syntax einer Formel ist analog zu der LaTeX-Formelschreibweise. Damit Formeln korrekt und gut lesbar dargestellt werden, ist *MathJax* [CSLPK] zu aktivieren, indem man

```
USE_MATHJAX = YES
```

in der der Konfigurationsdatei definiert. Um die Funktionsparameter zu beschreiben, wird der Tag `@param` verwendet. Hingegen adressiert man den Rückgabewert mittels des Tags `@return`. Das entsprechende Resultat in der HTML Dokumentation ist in Abbildung 4 dargestellt.

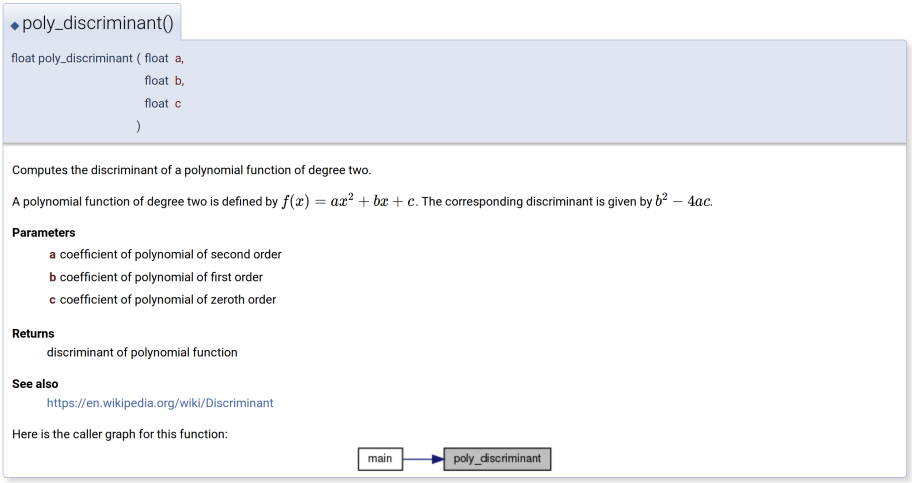


Abbildung 4: HTML Darstellung der Funktion `poly_discriminant` inklusive des Aufruf-Graphs.

3.5 Gestalten der Hauptseite

Die oben vorgestellte Techniken zur Kommentierung habe keine Auswirkung auf die Hauptseite, die in Abbildung 1 dargestellt ist. Zum expliziten Gestalten einer Hauptseite kann man eine separate Datei anlegen. Für das Textformat der Datei bietet sich dabei die einfache Sprache *Markdown* an, die in HTML Format umgewandelt werden kann. Die zentralen Elemente und Möglichkeiten zur Benutzung von Markdown sind in [Pri] zusammengefasst. Im beispielhaften Rechenprogramm wird die Hauptseite in der Datei `mainpage.md` gestaltet, wobei die Endung `.md` anzeigt, dass der Text konform zu Markdown ist. Der folgende Ausschnitt von `mainpage.md` zeigt beispielhaft, wie man in Markdown Quelltext einfügen kann.

```
**Program Execution**

To execute this program cmake is required to be installed.
Open a terminal and change into the directory of the 'scientific_calculator'.
Afterwards run the following commands.

'cmake --build cmake-build-debug --target scientific_calculator -- -j 4'

'cmake-build-debug/scientific_calculator'
```

Damit Doxygen `mainpage.md` als Hauptseite identifiziert, muss folgende Anpassung in der Konfigurationsdatei gemacht werden.

```
USE_MDFILE_AS_MAINPAGE = mainpage.md
```

Die obige Definition nimmt an, dass die Konfigurationsdatei und die Datei zur Gestaltung der Hauptseite im selben Verzeichnis ist. Falls dem nicht so ist, muss der Dateipfad entsprechend angepasst werden.

3.6 Darstellung von Aufruf-Graphen

Textuelle Erklärungen sind bei komplexem Softwareprojekten meist schwierig nachvollziehen. Dementsprechend bietet Doxygen die Option sogenannte Aufruf-Graphen automatisch zu generieren. Diese Graphen stellen die Relationen zwischen Modulen, Strukturen und Funktion dar, indem sie illustrieren, an welchen Stellen im Projekt ein Modul, eine Funktion oder eine Struktur verwendet bzw. aufgerufen wird. Auf diese Weise erhält man einen klaren Überblick über den Aufbau eines Projektes. Ein Beispiel für einen einfachen Aufruf-Graphen ist in Abbildung 4 gegeben. Die Erstellung solcher Graphen durch Doxygen erfordert die Installation von Graphviz, das bereits in Abschnitt 3.1 angesprochen wurde.

Damit die Aufruf-Graphen in der Dokumentation entsprechend angezeigt werden, ist die standardmäßige Konfigurationsdatei anzupassen, indem man folgende Definitionen macht.

```
HAVE_DOT = YES
CALL_GRAPH = YES
CALLER_GRAPH = YES
```

4 Nützliche Tipps

In diesem Abschnitt befinden sich eine Reihe nützlicher Tipps für die Erstellung einer Dokumentation mit Doxygen.

- Direkt zu Beginn eines Softwareprojektes sollten man eine Dokumentation erstellen, die dann fortlaufend aktualisiert wird. Dies erspart am Ende eines Projektes viel Zeit. Außerdem ist man dann gezwungen neuen Quelltext stets zu kommentieren, sodass zu jeder Zeit eine aktuelle Dokumentation zur Verfügung steht.
- Beim Erstellen von C-Header-Dateien sollte man stets diese kommentieren und nicht die entsprechen C-Quelltext-Dateien. Doxygen ist in der Lage automatisch die Zuordnung zwischen Funktions-Prototypen in der Header-Datei und den tatsächlichen Implementierung in der Quelltext-Datei zu identifizieren. Kommentiert man jedoch beide Dateien entstehen redundante Kommentare und Redundanz führt in der Regel zu erhöhtem Aufwand.
- Die Dokumentation sollte kompakt sein. Dementsprechend sollte man nur das Notwendigste beschreiben. Ansonsten erfordert das Erstellen der Dokumentation zu viel Zeit und das Lesen selbiger ebenfalls.
- Die Konfigurationsdatei bietet eine Menge Optionen zur Anpassung einer Dokumentation. Demzufolge ist es von Vorteil die Möglichkeiten einer Konfigurationsdatei genaustens zu studieren.
- Beim Aufrufen des Befehls `doxygen` wird der Prozess der Erstellung der Dokumentation detailliert protokolliert bzw. ausgegeben. Diese Ausgabe sollte man stets auf mögliche Warnungen oder Fehler überprüfen.

- Zum Bauen einer PDF der Dokumentation muss man die LaTeX-Datei `refman.tex` entsprechend benutzen, wobei diese Datei im Ordner `latex` vorliegt.

5 Referenzen und weitere Hilfsquellen

Anbei dieses Artikels befindet sich das beispielhafte Rechenprogramm *Scientific Calculator* inklusive Dokumentation. Das Projekt kann genutzt werden, um den Umgang mit Doxygen praktisch zu üben und das eigene Wissen zu vertiefen.

Falls Probleme beim Erstellen einer Dokumentation mit Doxygen auftreten, bietet sich das Aufsuchen der Dokumentation von Doxygen [vHa] an. Dort finden sich ausführliche Erklärung zu allen Elementen von Doxygen. Alternativ bieten sich diverse Foren an. Insbesondere auf *Stack Overflow* [AS] findet man schnell Lösungen für aufgetretene Probleme mit Doxygen oder kann selbst das eigene Problem als Frage formulieren.

[AS] Jeff Atwood und Joel Spolsky: *Stack Overflow*. `https://stackoverflow.com/`. Letzter Zugriff: 23.05.2019.

[CSLPK] Davide Cervone, Volker Sorge, Christian Lawson-Perfect und Peter Krautzberger: *MathJax*. `https://www.mathjax.org`. Letzter Zugriff: 23.05.2019.

[EGK+03] John Ellson, Emden R. Gansner, Eleftherios Koutsofios, Stephen C. North und Gordon Woodhull: *Graphviz and dynagraph – static and dynamic graph drawing tools*. In: *GRAPH DRAWING SOFTWARE*, Seiten 127–148. Springer-Verlag, 2003.

[Fou] Free Software Foundation: *GNU General Public License, version 2*. `http://www.gnu.org/licenses/old-licenses/gpl-2.0.html`. Letzter Zugriff: 23.05.2019.

[Lam94] Leslie Lamport: *LATEX: a document preparation system: user's guide and reference manual*. Addison-wesley, 1994.

[Pri] Adam Pritchard: *Markdown Cheatsheet*. `https://github.com/adam-p/markdown-here/wiki/Markdown-Cheatsheet`. Letzter Zugriff: 23.05.2019.

[vHa] Dimitri van Heesch: *Doxygen*. `http://www.doxygen.nl/`. Letzter Zugriff: 23.05.2019.

[vHb] Dimitri van Heesch: *Doxygen Installation*. `http://www.doxygen.nl/manual/install.html`. Letzter Zugriff: 23.05.2019.

[Wik] Wikipedia: *Softwaredokumentation*. `https://de.wikipedia.org/wiki/Softwaredokumentation`. Letzter Zugriff: 23.05.2019.