Využití nástroje Doxygen pro vytváření dokumentace programů ze zdrojového kódu

Filip Dorňák

Bakalářská práce
2017

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky
akademický rok: 2016/2017

ZADÁNÍ BAKALÁŘSKÉ PRÁCE
(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Filip Dorňák**
Osobní číslo:	**A14082**
Studijní program:	**B3902 Inženýrská informatika**
Studijní obor:	**Informační a řídicí technologie**
Forma studia:	**prezenční**
Téma práce:	**Využití nástroje Doxygen pro vytváření dokumentace programů ze zdrojového kódu**
Téma anglicky:	**Exploiting the Doxygen Tool for the Generation of Program Documentation from Source Codes**

Zásady pro vypracování:

1. **Zpracujte literární rešerši na téma dokumentace zdrojového kódu a příslušných nástrojů.**
2. **Popište nástroj Doxygen.**
3. **Navrhněte a vytvořte sadu zdrojových kódů v jazyce C, která bude demonstrovat možnosti nástroje Doxygen ke generování dokumentace z komentářů ve zdrojovém kódu.**
4. **Vytvořte příručku popisující postup použití nástroje Doxygen a přehled běžně používaných funkcí tohoto nástroje.**
5. **Vytvořte s využitím nástroje Doxygen ukázkovou dokumentaci pro navržené zdrojové kódy.**

Rozsah bakalářské práce:

Rozsah příloh:

Forma zpracování bakalářské práce: **tištěná/elektronická**

Seznam odborné literatury:

1. **BARR, Michael a Anthony J MASSA. Programming embedded systems: with C and GNU development tools. 2nd ed. Sebastopol: O'Reilly, 2006, xxi, 301 s. ISBN 978-0-596-00983-0.**
2. **CATSOULIS, John. Designing embedded hardware. 2nd ed. Sebastopol, CA: O'Reilly, 2005, xvi, 377 p. ISBN 0596007558.**
3. **MCCONNELL, Steve. Dokonalý kód: umění programování a techniky tvorby software. Vyd. 1. Brno: Computer Press, 2005, 894 s. ISBN 802510849x.**
4. **PINKER, Jiří. Mikroprocesory a mikropočítače. 1. vyd. Praha: BEN – technická literatura, 2004, 159 s. ISBN 80-7300-110-1.**
5. **Doxygen [online]. 2017 [cit. 2017-01-16]. Dostupné z: http://www.stack.nl/ dimitri/doxygen/index.html**

Vedoucí bakalářské práce: **Ing. Jan Dolinay, Ph.D.**
Ústav automatizace a řídicí techniky

Datum zadání bakalářské práce: **24. února 2017**

Termín odevzdání bakalářské práce: **24. května 2017**

Ve Zlíně dne 24. února 2017

UNIVERZITA TOMÁŠE BATI VE ZLÍNĚ
Fakulta aplikované informatiky
L.S.

doc. Mgr. Milan Adámek, Ph.D.
děkan

prof. Ing. Vladimír Vašek, CSc.
ředitel ústavu

Prohlašuji, že

- beru na vědomí, že odevzdáním diplomové/bakalářské práce souhlasím se zveřejněním své práce podle zákona č. 111/1998 Sb. o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších právních předpisů, bez ohledu na výsledek obhajoby;
- beru na vědomí, že diplomová/bakalářská práce bude uložena v elektronické podobě v univerzitním informačním systému dostupná k prezenčnímu nahlédnutí, že jeden výtisk diplomové/bakalářské práce bude uložen v příruční knihovně Fakulty aplikované informatiky Univerzity Tomáše Bati ve Zlíně a jeden výtisk bude uložen u vedoucího práce;
- byl/a jsem seznámen/a s tím, že na moji diplomovou/bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb. o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) ve znění pozdějších právních předpisů, zejm. § 35 odst. 3;
- beru na vědomí, že podle § 60 odst. 1 autorského zákona má UTB ve Zlíně právo na uzavření licenční smlouvy o užití školního díla v rozsahu § 12 odst. 4 autorského zákona;
- beru na vědomí, že podle § 60 odst. 2 a 3 autorského zákona mohu užít své dílo – diplomovou/bakalářskou práci nebo poskytnout licenci k jejímu využití jen připouští-li tak licenční smlouva uzavřená mezi mnou a Univerzitou Tomáše Bati ve Zlíně s tím, že vyrovnání případného přiměřeného příspěvku na úhradu nákladů, které byly Univerzitou Tomáše Bati ve Zlíně na vytvoření díla vynaloženy (až do jejich skutečné výše) bude rovněž předmětem této licenční smlouvy;
- beru na vědomí, že pokud bylo k vypracování diplomové/bakalářské práce využito softwaru poskytnutého Univerzitou Tomáše Bati ve Zlíně nebo jinými subjekty pouze ke studijním a výzkumným účelům (tedy pouze k nekomerčnímu využití), nelze výsledky diplomové/bakalářské práce využít ke komerčním účelům;
- beru na vědomí, že pokud je výstupem diplomové/bakalářské práce jakýkoliv softwarový produkt, považují se za součást práce rovněž i zdrojové kódy, popř. soubory, ze kterých se projekt skládá. Neodevzdání této součásti může být důvodem k neobhájení práce.

Prohlašuji,

- že jsem na diplomové/bakalářské práci pracoval samostatně a použitou literaturu jsem citoval. V případě publikace výsledků budu uveden jako spoluautor.
- že odevzdaná verze diplomové práce a verze elektronická nahraná do IS/STAG jsou totožné.

Ve Zlíně, dne 22.5.2017

.....
podpis diplomanta

ABSTRAKT

Cílem této práce je seznámit čtenáře s možnostmi dokumentace zdrojového kódu a blíže představit nástroj Doxygen. Řešení zahrnuje příručku pro použití softwaru Doxygen, šablony pro usnadnění procesu psaní a ukázkovou dokumentaci vytvořenou pro dříve existující projekt. Hlavní přínos je pro uživatele, kteří mají zájem začít používat, případně si rozšířit znalosti o nástroji Doxygen.

Klíčová slova: dokumentace, Doxygen, programovací jazyk C

ABSTRACT

The primary focus of this work is to inform the reader about possibilities in documenting source code and further introduce software called Doxygen. The practical part of thesis consists of manual for Doxygen software, templates to ease the process of writing and generated example of documentation for previously existing project. The main contribution of this thesis is for people who are starting to use, or want to expand their knowledge about Doxygen software.

Keywords: documentation, Doxygen, C programming language

Na tomto místě bych rád poděkoval rodině a přátelům, kteří mě v průběhu celého studia podporovali. Vážím si práce všech kantorů, kterým jsem vděčný za jejich trpělivost a předané vědomosti, zvláště pak vedoucímu této bakalářské práce Ing. Janu Dolinayovi Ph.D. za příjemné jednání, cenné připomínky a návrhy.

Prohlašuji, že odevzdaná verze bakalářské/diplomové práce a verze elektronická nahraná do IS/STAG jsou totožné.

OBSAH

ÚVOD

I přesto, že dokumentování zdrojového kódu je více a více aktuální téma, stále existují projekty, které dokumentaci podceňují. Jedním z důvodů pro tento přístup může být časová a finanční náročnost. Existují malé projekty, které si vystačí s dokumentací pomocí krátkých komentářů uvnitř zdrojového kódu, avšak pro větší projekty je výhodné využít některý ze specializovaných nástrojů. Jedním z nich je právě Doxygen.

V teoretické části je probrána dokumentace obecně. Je popsán přínos jak z pohledu programátorského, tak z pohledu uživatele. Také jsou zde zmíněny základní zásady, které by kvalitní dokumentace měla splňovat a čemu se vyvarovat. Druhé část je věnována nástroji Doxygen, jeho historii a základním myšlenkám, které provázeli vývoj. Jeho možnosti a popularita je podložena několika konkrétními příklady. Na závěr teoretické části bylo poukázáno na další software podobného charakteru. U těchto alternativ jsou zmíněny především jejich odlišnosti, které by mohl uživatel ocenit, avšak také nedostatky při porovnání s programem Doxygen.

V praktické části byly vytvořeny šablony, které se snaží pokrýt základní použití nástroje Doxygen. Tento materiál pak může být použit uživatelem pro urychlení tvorby dokumentačních bloků. Jejich součástí jsou také části zdrojového kódu, který demonstruje jejich použití na konkrétním případě. Šablony nejsou omezeny programovacím jazykem a lze je využít při tvorbě jakéhokoliv projektu. Použití šablon je ukázáno také v části použití nástroje Doxygen, kde je rozebrán postup tvorby krok za krokem. V závěru praktické části byla vytvořena o něco komplexnější dokumentace dříve existujících zdrojových kódů, která slouží jako náhled na strukturu komentovaného textu, ale také jako prezentace exportované podoby.

Bakalářská práce obsahuje přílohu v podobě příručky, která popisuje proces generování dokumentace od samotné instalace, přes editaci souboru s natavením a komentování zdrojového kódu, až po závěrečný export do různých formátů, které jsou k dispozici.

I. TEORETICKÁ ČÁST

1 DOKUMENTACE

Dokumentace softwaru je nezbytná činnost, která ovlivňuje efektivitu vývoje. Každý zdrojový kód může být znovupoužit, buďto přímo nebo pouze jako materiál pro pochopení řešení. V obou případech dokumentace slouží pro specifikaci chování programu. Bez ní jsme často nuceni vytvářet nebezpečné domněnky. Taktéž existuje spousta informací, které nelze vyjádřit programovacím jazykem, protože vyžadují sílu a ohebnost běžného jazyka [1**Chyba! Nenalezen zdroj odkazů.Chyba! Nenalezen zdroj odkazů.Chyba! Nenalezen oj odkazů.Chyba! Nenalezen zdroj odkazů.Chyba! Nenalezen zdroj odkazů.Chyba! Nenalezen zdroj odkazů.Chyba! Nenalezen zdroj odkazů.**].

1.1 Typy dokumentace

Mezi části dokumentace softwarového projektu patří interní a externí. Převážně u větších projektů se setkáme s externí částí, která je ve formě oddělených dokumentů. Externí část není soustředěna na zdrojový kód a jeho implementaci. Můžeme zde najít definici problému nebo přístup k řešení. Naopak v interní části nalezneme zdrojový kód a jeho detailní popisy implementace. Vzhledem k úzkému spojení se zdrojovým kódem je tato část snadněji udržovatelná [2].

1.2 Důvody k dokumentování

Primárním cílem každé dokumentace má být pochopení zdrojového kódu. To napomáhá ke snadnému čtení a údržbě. Pochopení může spočívat v objasnění složitého zápisu nebo upozornění na nedostatky programu. Existují však další důvody, které jsou s pochopením úzce spjaty [3].

1.2.1 Časový faktor

Pokud se programátor navrátí po delší době ke svému zdrojovému kódu, je pravděpodobné, že narazí na části, které nebudou snadné k pochopení, i přesto, že se jedná o jeho kód. Pouhá věta v dokumentaci může následně ušetřit několik hodin procházení kódem [4].

1.2.2 Přístupnost

Bez ohledu na to, o jaký typ projektu se jedná, kvalitně napsaná dokumentace může výrazně ovlivnit zájem ze strany uživatelů. Zvláště pak u open-source projektů, kde pochopení zdrojového kódu je zásadní, pro nové přispěvatele [4].

1.2.3 Reflexe kódu

Popsání kódu slovy vyžaduje jeho čitelnost a naše pochopení. Pokud jedna z těchto vlastností bude nedostatečná, tvorba dokumentace je nemožná, a tak vzniká potřeba přepsat kód do srozumitelnější podoby. Navíc procházení dokumentace vede ke snadnějšímu odhalení příčin chyb [3].

1.3 Programovací styl

Hlavním zdrojem informací v interní dokumentaci nejsou komentáře, ale kvalitní programovací styl. Takový styl je definován programovou strukturou, minimalizací komplexnosti použitých řešení a dobrým pojmenováním proměnných a funkcí. Špatný zdrojový kód nelze napravit výborným komentářem [5].

1.4 Komentování

Existují však informace, které zdrojovým kódem předáme jen velmi těžce a je výhodné využít komentáře. Časová náročnost efektivního komentování není vysoká. Používání příliš mnoha komentářů může být méně přínosné, než nekomentování vůbec. Pokud je pro nás náročné popsat slovy, co daná část kódu dělá, může to být znamení, že kódu dostatečně nerozumíme a čas, který vložíme do psaní komentáře, bychom stejně později použili pro pochopení. Komentáře by měli být krátké a k věci, lze je rozdělit do šesti kategorií, kde použití prvních tří bychom se měli vyvarovat [2].

1.4.1 Opakující

Jedná se o pouhý přepis kódu jinými slovy. Negativní vlastností je, že nabízí čtenáři pouze více textu bez předání nové informace [2]. Typickým příkladem může být:

```
// Otestování zda proměnná cislo1 je rovna proměnné cislo2

if ( cislo1 == cislo2)
```

1.4.2 Vysvětlující

Tento typ komentářů má za úkol objasnit komplikovanou část kódu. V některých případech se může jednat o opodstatněné použití, nicméně ve většině situacích je příčinou matoucí kód a je vhodnější věnovat čas zkvalitnění kódu, než do komplikovaného komentáře [2].

1.4.3 Označující

Obvykle se vyskytují v zdrojovém kódu jen dočasně. Slouží vývojářům například k tomu, aby označili místo v kódu, které není dokončeno. Hlavní problém vzniká, když každý vývojář používá jiný zápis takových míst. Řešením může být vytvoření standartu pro tento typ komentáře v rámci celého týmu, v lepším případě využít některých vývojových prostředí, které umožňují vytvořit monitorovací seznam takových míst [2].

1.4.4 Shrnující

Účelem je shrnout několik následujících řádků kódu do jedné až dvou vět. Umožňují nám rychleji procházet kód a jsou zvláště cenné pro ty, kteří se setkávají s kódem poprvé [2].

1.4.5 Upozorňující

Pokud je funkčnost části kódu podmíněna, měli bychom o této skutečnosti informovat. Ať již se jedná o chyby nebo chtěné chování, jakákoliv objasňující poznámka výrazně ušetří čas při dalším vývoji a vytváření testujících jednotek [5].

1.4.6 Organizační

Některé informace nelze vyjádřit kódem, ale je žádoucí, aby se ve zdrojových kódech nacházeli. Mezi takové patří autorské práva, odkazy na online reference, nedokončené části (tzv. TODO), optimalizační poznámky a také komentáře, které jsou využívány nástroji pro generování dokumentace. Také je možné zde ukládat informace o verzích, autorech nebo datech, nicméně moderní verzovací systémy zvládají tuto práci mnohem lépe [2].

1.5 Jednoduchá údržba

Jedna z nejnáročnějších částí, spojená s tvorbou dokumentace, je její údržba. Zdrojový kód se časem mění a vyvíjí, kdežto na aktualizaci komentářů se často zapomíná. Zde je důležité, aby na žebříčku priorit každého programátora v týmu byla snaha udržet dokumentaci aktuální. Jakmile komentáře stárnou výrazně rychleji než zdrojový kód, stávají se nespolehlivými. Dokonce může nastat situace, kdy nekorektní komentář je nebezpečnější, než-li žádný. Komplikovaná syntaxe může generovat velmi kvalitní dokumentaci, nicméně může odradit od častých změn. V rámci zachování dokumentace aktuální je důležité, aby proces tvorby a úprav byl co nejsnadnější [5].

1.6 Konference Write the Docs

Zájem o dokumentaci potvrzuje existence každoroční konference Write the Docs. Základy této komunity sahají na začátek roku 2013 a od této chvíle se uskutečnilo celkem osm konferencí. Trvání je obvykle dva až tři dny. V tomto rozmezí mají účastníci k dispozici několik přednášek od různých řečníků, které se týkají dokumentování zdrojového kódu. Následně jsou přednášky k dispozici online. Výhodou této konference je rozmanitost řečníků. Lze zde nalézt světově uznávané programátory, ale také méně známe, kteří přestavují zajímavé pohledy z jejich práce na menších projektech [4].

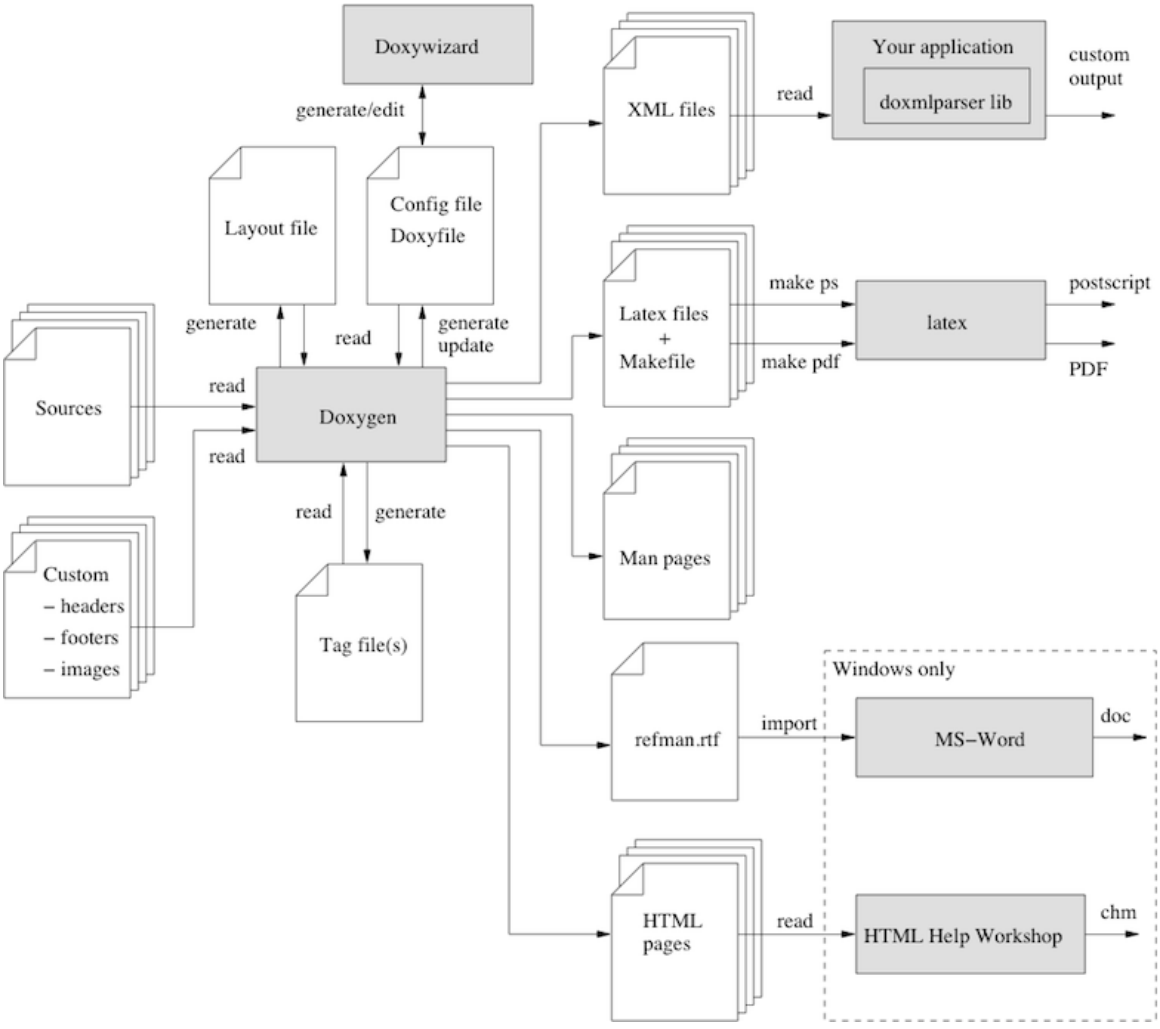
2 DOXYGEN

Doxygen lze definovat jako nástroj pro tvorbu dokumentace na základě speciálně označeného zdrojového kódu pomocí komentářů. Pro generování samotné dokumentace využívá, kromě komentářů, také zdrojový kód samotný, především pro vytvoření vazeb mezi jednotlivými prvky. Hlavním cílem tohoto projektu byla podpora pro jazyk C++, nicméně dnes podporuje mnoho dalších (2.4) [6].

2.1 Historie

První verze tohoto programu pochází z října roku 1997 a jejím autorem je Dimitri van Heesch. Byla vydána pod licencí GNU General Public Licence v2.0. Od roku 1999 byl tento projekt přesunut na servery GitHub, kde stále probíhá aktivní vývoj [7].

2.2 Proces tvorby



Obrázek 1 - Doxygen - tok informací [12]

Předchozí obrázek (Obrázek 1) představuje tok informací v procesu generování dokumentace. Nezbytné vstupy pro tento proces jsou zdrojové kódy, obsahující komentující bloky a konfigurační soubor Doxyfile. Kaskádové styly, obsah hlavičky nebo patičky je nutné dodat pouze v případě, že nám nevyhovuje výchozí, který je dodán při generování. Konfigurační soubor slouží ke specifikaci, jakým způsobem se mají zdrojové kódy zpracovávat. Nese informace o použitém jazyku, umístění zdrojových souborů, ale také nastavení výstupních formátů [6].

2.3 Využití

Primárním účelem tohoto programu je tvorba dokumentace, nicméně při správném nastavení lze jeho možnosti využít i jinak. Jedním z takových případů, je vygenerování vizuální struktury programu jen na základě zdrojového kódu, není potřeba žádných komentujících bloků. To lze využít především u velkých projektů, pro dodatečnou orientaci. Další využití může být generování externí dokumentace, která opět není závislá na zdrojovém kódu. Jedním z příkladů je právě oficiální manuál pro použití Doxygenu [6].

2.4 Programovací jazyky

Popularita Doxygenu je pozitivně ovlivněna také širokou škálou programovacích jazyků, pro které lze dokumentaci generovat. V aktuální verzi jsou oficiálně podporované jazyky C, C++, C#, Objective-C, D, IDL, Java, VHDL, PHP, Python, Tcl a Fortran. Lze použít také nestandardní přípony souboru s uvedenými parsery přidáním přípon do konfiguračního souboru. Pokud hledáte podporu pro jiný jazyk, pak existuje spousta neoficiálních nástrojů, které to umožňují. Tyto nástroje přepisují zdrojový kód z jazyka neznámého pro Doxygen, do jazyka, kterému rozumí. Nejedná se však o kompletní kód, postačí základní struktura, z které je generována dokumentace. Obvykle se tedy vynechávají celá těla funkcí, protože nejsou využita. Tímto způsobem lze vytvořit podporu pro téměř jakýkoliv jazyk [6].

2.5 Výstupní formáty

V konfiguračním souboru lze povolit přímý export do formátu HTML, Latex, Man pages, RTF, XML a Docbook. Na základě HTML výstupu lze poté vygenerovat Microsoft's HTML Help, Qt Compressed Help, Eclipse Help nebo XCode DocSets. A pomocí formátu Latex lze vytvořit PostScript nebo PDF [6].

2.6 Grafy a diagramy

Doxygen dokáže vytvářet grafy a diagramy pomocí nástroje Graphviz. Diagramy použít pro vizualizaci závislostí nebo dědičností pro jednotlivé třídy. Tyto možnosti jsou cílené především pro objektově orientované programovací jazyky. Diagram dědičnosti můžeme vygenerovat i pro celý projekt, a vytvořit tak prostředek pro snadnou orientaci. Omezením u této funkce je, že většina typů grafů je podporována pouze pro výstup do HTML, případně RTF formátu [6].

2.7 Vyhledávání v dokumentaci

Vyhledávání v HTML dokumentaci lze zajistit několika způsoby. Nejjednodušší je použití offline umístění, kde je vyhledávání zajištěno pomocí Javascriptu, který je interpretován internetovým prohlížečem uživatele. Tímto způsobem však nelze vyhledávat v popisech, pouze v názvech funkcí, souborů nebo proměnných. Další z možností je umístit dokumentaci na web server, který dokáže pracovat s PHP kódem. Poté může být vyhledávání prováděno na straně serveru. Výhodou je, že se jedná o plně fulltextové vyhledávání. Od verze 1.8.3 lze dodat externí vyhledávající a indexující nástroje a dosáhnout tak optimalizovaných výsledků pro velké projekty. U jiných výstupních formátů jsme odkázáni na možnosti programů, v kterých dokumentaci otvíráme [6].

2.8 Jazykové verze

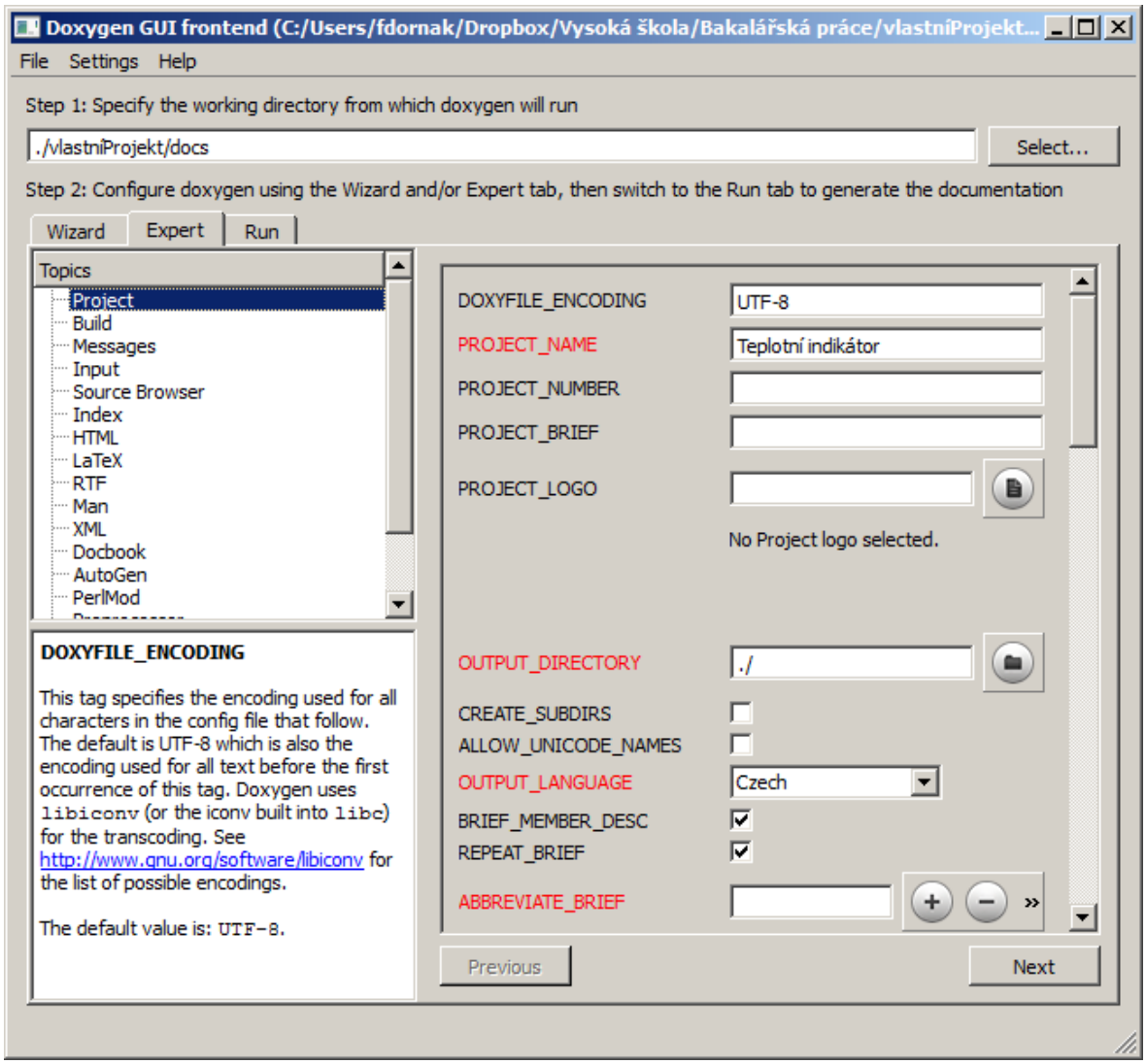
Pro některé projekty je žádoucí, aby dokumentace existovala ve více jazykových podobách. Toho lze docílit tím, že každý komentující blok označíme názvem jazyka, kterým je komentář napsán a při generování Doxygen použije jen bloky s konkrétním jazykem. Nevýhodou tohoto postupu je, že pro každý jazyk se generuje unikátní dokumentace a tím se zvyšuje jak časová, tak paměťová náročnost [6].

2.9 Externí dokumentace

V případě, že využíváme externí knihovny nebo nástroje, které mají vytvořenou dokumentaci pomocí Doxygenu, lze tyto dokumentace připojit k té naší externě. Tento druh propojení dokáže ušetřit místo na disku a také urychlit generování dokumentace. Taktéž některé projekty nenabízí kompletní zdrojový kód, pouze dokumentaci [6].

2.10 Doxywizard

Doxywizard je volitelná komponenta, která přišla s verzí 1.1.3. Jedná se o grafické rozhraní pro editaci konfiguračního souboru a generování dokumentace. Ovládání je rozděleno do tří částí. V první části lze editovat základní nastavení projektu jako název, výběr složky se zdrojovými soubory nebo volba programovacího jazyka a výstupních formátů. V druhé záložce se nachází možnosti s označením expert. Zde jsou k dispozici pokročilá nastavení projektu. V poslední záložce se nachází tlačítko pro vygenerování dokumentace. Také zde nalezneme prostor pro text, kde budou přesměrovány zprávy o průběhu generování včetně upozornění a varování na případné chyby v dokumentaci [6].



Obrázek 2 - Ukázka uživatelského rozhraní komponenty Doxywizard

2.11 Alternativy

I přesto, že se Doxygen řadí mezi jeden z nejlepších ve své kategorii, existuje několik alternativ. U těch můžete ocenit jejich jednoduchost, odlišnou syntaxi, případně podporu jiných jazyků.

2.11.1 Sphinx

První vydaná verze vyšla pod licencí BSD v roce 2008, kde autorem byl Georg Brandl. Hlavní cíl bylo generování dokumentace pro jazyk Python, postupem času se však dostavila oficiální podpora pro další jazyky jako C, C++ nebo JavaScript. Podpora dalších jazyků (tzv. domény), lze doinstalovat jako rozšíření, ty byly vytvořeny komunitou. Podporované výstupní formáty jsou HTML, Latex, Plain text, Man pages, JSON a XML. V případě, že chcete využívat Sphinx, ale nejste spokojeni s podporou konkrétního jazyka, je možné sáhnout po programu Breathe, který poskytuje propojení mezi programy Sphinx a Doxygen. Jedním z důvodů popularity tohoto projektu je také web READTHEDOCS.ORG, který nabízí zdarma hostování dokumentace vytvořené tímto nástrojem [8].

2.11.2 Docurium

Docurium byl vytvořen pod licencí MIT, účelem bylo nahrazení Doxygenu pro projekt libgit2 a je zástupcem jednoduchosti. Výsledkem je statický web vygenerovaný pouze na základě hlavičkových souborů jazyka C. Kromě omezení na jeden jazyk je také nutné dokumentaci provozovat na web serveru, protože využívá volání XMLHttpRequest. I přes značná omezení, může být pro vytváření veřejné dokumentace API postačující [9].

2.11.3 Natural Docs

Jedná se o další open-source projekt, tentokrát s licencí GPLv2, který podporuje devatenáct programovacích jazyků. Jedním z omezení je možnost vyexportovat dokumentaci pouze do formátu HTML. Dalším negativním faktem je stagnující vývoj a neaktivní komunita [10].

II. PRAKTICKÁ ČÁST

3 ŠABLONY PRO TVORBU DOKUMENTACE

Cílem navrhnutých šablon je usnadnit práci při vytváření dokumentace. Byly vytvořeny tak, aby případný programátor mohl jednoduše zkopírovat části, které potřebuje a urychlit tak vytváření dokumentace. Při tvorbě bylo cíleno na základní potřeby majoritní části projektů. Doxygen dokáže splnit mnohé individuální požadavky, nicméně pokrytí všech prostřednictvím šablon a následné orientování mezi nimi by bylo velmi náročné.

3.1 Rozdělení

Šablony jsou rozděleny do několika malých souborů, podle použití (Tabulka 1). Každý z těchto souborů je rozčleněn do tří částí. V první části se nachází prázdná šablona, v další části jsou přidány popisy jednotlivých složek a na závěr je uveden názorný příklad použití. Speciálním případem je soubor obsahující formátování textu, ten slouží pouze jako rychlá nápověda a nejedná se o šablonu v pravém slova smyslu.

Tabulka 1 - Rozdělení šablon do souborů

Název souboru	Použití šablony
enum.dox	výčtový typ enum
file.dox	soubor
function.dox	funkce
macro.dox	makra pro preprocesor
mainPage.dox	úvodní strana
textFormat.dox	formátování textu

3.2 Programovací jazyk

I přesto, že celá bakalářská práce je převážně soustředěna na programovací jazyk C, šablony nejsou určeny pro jeden konkrétní jazyk. S programovacím jazykem je pevně spjatá i přípona souborů. V případě šablon byla použita přípona *.dox*, která je rozpoznána jako soubor obsahující syntaxi Doxygenu. Tento typ souboru lze použít například pro vytvoření úvodní stránky, která neobsahuje zdrojový kód.

3.3 Postup použití

Nejdříve si uživatel vybere prvek, který chce zakomponovat do své dokumentace a nalezne k němu patřičný soubor mezi šablonami. Ten obsahuje pouze text a lze jej otevřít v každém textovém editoru. Rozdělení na tři části slouží pro oddělení podle úrovně nápovědy k použití. Obvykle zkopírujeme do schránky pouze obsah prázdné šablony, která se nachází v horní části a vložíme ji před dokumentovaný prvek do našeho zdrojového kódu. V případě, že si nejsme jisti, co jednotlivé klíčové slova znamenají, pak může pomoci druhá část šablony, která popisuje syntaxi jednotlivých slov, z tohoto důvodu však není vhodná pro kopírování. V poslední části nalezneme názorný příklad použití dokumentujícího bloku včetně ukázkového zdrojového kódu, který je k němu vztažený. I přesto, že šablony lze použít pro jakýkoliv jazyk, ukázky ve třetí části jsou napsány výhradně programovacím jazykem C. Kromě šablon, je zde obsažen také soubor, který slouží jako nápověda pro formátování textu a lze využít v kterémkoliv dokumentujícím bloku.

4 POUŽITÍ NÁSTROJE DOXYGEN

Cílem této kapitoly je demonstrovat použití softwaru Doxygen pro vytvořený zdrojový kód s detailním popisem jednotlivých kroků.

4.1 Popis projektu

Pro dokumentaci byl zvolen projekt pro mikropočítač FRDM-KL25Z napsaný v programovacím jazyce C. Účelem tohoto projektu bylo vytvoření teplotního senzoru s nastavitelnou indikací pomocí LED diod. Uživatel pomocí sériového rozhraní zasílá specifické příkazy pro přiřazení konkrétních hraničních teplot jednotlivým led diodám. V případě, že okolní teplota přesáhne tyto hranice, je rozsvícena uživatelem nastavená led dioda.

4.2 Struktura projektu

Projekt se skládá ze tří souborů. Jedná se o hlavní programovou smyčku umístěnou v souboru *main.c* a ovladač teploty *drv_temp.c* a jeho hlavičkový soubor *drv_temp.h*. Ostatní ovladače pro displej, časovač a sériovou komunikaci byly použity z materiálů pro předmět Programování mikropočítačů.

4.3 Cíle dokumentace

Hlavní prioritou je zdokumentování všech vytvořených zdrojových prvků, včetně jednoduché úvodní strany a umístění ovladače pro teplotní senzor do samostatného modulu.

4.4 Vytvoření souboru s nastavením

1. Vytvoříme dvě složky *docs* a *src*
2. Do složky *src* vložíme veškeré zdrojové kódy
3. Přejdeme do složky *docs* a zavoláme příkaz *doxygen -g*
4. Vznikne nový soubor *Doxyfile*, ten otevřeme v textovém editoru
5. Provedeme úpravu následujících položek (Tabulka 2), ostatním ponecháme výchozí hodnoty

Tabulka 2 - Úprava souboru s nastavením

Název položky	**Nová hodnota**
PROJECT_NAME	"Teplotní indikátor"
OUTPUT_DIRECTORY	./
OUTPUT_LANGUAGE	Czech
OPTIMIZE_OUTPUT_FOR_C	YES
INPUT	../src/

4.5 Vytvoření komentujících bloků

Komentující bloky budou umístěny v souboru *main.c* a pro ovladač v souboru *drv_temp.h*.

1. Na začátek každého ze souborů vložíme komentující blok, který bude popisovat soubor jako celek
2. Soubor označíme pro dokumentovaní tak, že do komentujícího bloku umístíme klíčové slovo */file*
3. Víceřádkové komentující bloky umístíme nad všechny výčtové typy enum, struct a funkce
4. Do víceřádkových komentujících bloků umístíme klíčové slovo */brief* následované větou, která stručně popisuje blok kódu.
5. Za stručný popis vždy umístíme jeden prázdný řádek, který jej oddělí od detailního popisu
6. Nyní můžeme vložit několikařádkový detailní popis
7. V případě funkcí, které mají návratový typ, použijeme klíčové slovo */return* následováno návratovou hodnotou a popisem.
8. V případě funkcí s parametry, používáme klíčové slovo */param[]*, kde do hranaté závorky umisťujeme *in/out*, což odlišuje vstupní od výstupních parametrů. Následuje jméno proměnné a její popis.
9. Jednořádkové komentující bloky umístíme nad globální proměnné a jednotlivé prvky výčtových typů a struktur

10. Do těchto komentujících bloků vkládáme stručný popis bez potřeby uvozovat jej klíčovým slovem

```
/**
 * \file
 * \brief Ovladač pro získání teploty
 *
 * Teplota je měřena pomocí teplotního snímače LM75
 */
```

Obrázek 3 - Ukázka komentujícího bloku pro soubor drv_temp.h

4.6 Úvodní strana

Vzhledem k tomu, že se jedná o malý projekt, který nevyžaduje komplikovanou úvodní stranu, bude vytvořena přímo v souboru *main.c*.

1. Do prvního dokumentujícího bloku, který obsahuje klíčové slovo */file*, umístíme za jeho detailní popis klíčové slovo */mainpage*
2. Na nový řádek můžeme vložit několikařádkový popis celého projektu.

4.7 Rozdělení do modulů

Je žádané, zvláště u velkých projektů, oddělit zdrojové soubory do logických celků. V našem případě se jedná o ovladač pro senzor teploty.

1. Přejdeme do souboru *drv_temp.h*
2. Do prvního komentujícího bloku, který obsahuje popis souboru, umístíme na úplný začátek klíčové slovo */addtogroup* následované názvem *ovladaceSenzoru*.
3. Na nový řádek vložíme stručný a detailní popis, stejně jako u jiných prvků. Tento se však vztahuje pro modul jako celek
4. Poté vložíme *@{*, čímž vytvoříme ohraničení textu, které má být obsaženo v modulu.
5. Na závěr souboru vložíme nový dokumentační blok a do něj *@}*, kterým uzavřeme obsah modulu.

Pokud bychom chtěli další soubor umístit do stejného modulu, postačí mu přiřadit stejný název *ovladaceSenzoru*.

4.8 Vygenerování dokumentace

1. Přejdeme do složky *docs*
2. Zavoláme příkaz *doxygen Doxyfile*
3. Výstupní dokumentace nalezneme ve složce *docs/html* a *docs/latex*

4.9 Převod do formátu PDF

V našem případě, jsme použili pro ovládací prvky české znaky a před převodem je potřeba udělat jednoduchou úpravu, jinak je generování nefunkční. Problém je podrobněji popsán v kapitole (

Problémy s exportem z formátu Latex do PDF).

1. Přejdeme do složky *docs/latex*
2. Otevřeme soubor *refman.tex* v libovolném textovém editoru
3. Za řádek *\usepackage[czech]{babel}* vložíme následujících šest řádků

   \usepackage{regexpatch}

   \makeatletter

   % Change the `-` delimiter to an active character

   \xpatchparametertext\@@@cmidrule{-}{\cA-}{}{}

   \xpatchparametertext\@cline{-}{\cA-}{}{}

   \makeatother
4. Stáhneme a nainstalujeme software TeX Live 2016
5. Spustíme dávkový soubor *make.bat*
6. Výsledný pdf soubor nalezneme pod názvem *refman.pdf*

5 UKÁZKOVÁ DOKUMENTACE

V této části byla vytvořena dokumentace pro předem existující skupinu projektů. Hlavní prioritou této dokumentace je snadný přístup ze strany uživatele i budoucího programátora.

5.1 Specifikování zdrojového kódu

Jedná se o skupinu dvaceti sedmi projektů, které slouží pro výuku předmětu "Programování mikropočítačů". Projekty byly napsány pro mikropočítač FRDM-KL25Z, který je vyvíjen společností NXP Semiconductor. Každý z nich výhradně používá programovací jazyk C a většina z nich obsahuje pouze jeden zdrojový soubor, výjimečně doprovázený hlavičkovým souborem. Účelem této dokumentace je nejen ukázat použití nástroje Doxygen, ale také poskytnout další strukturovaný materiál pro výuku tohoto předmětu.

5.2 Výběr vývojového prostředí

I přesto, že společnost NXP dodává pro tento mikropočítač vývojové prostředí Kinetis® Design Studio, které je postaveno na open-source projektu Eclipse, pro psaní dokumentace bylo upřednostněno prostředí Codelite. Při této práci nejsou vyžadovány úpravy zdrojového kódu, pouze psaní komentářů. Vzhledem k situaci, bylo snadné přepínání mezi projekty a podpora syntaxe klíčových slov Doxygenu prioritou. Všechny tyto vlastnost ve výchozím nastavení Codelite splňuje. Výběr ovlivnily pozitivní zkušenosti s tímto programem v minulosti.

5.3 Vytvoření úvodní strany

Tvorba dokumentace započata úvodní stranou. Tato strana by měla sloužit jako rozcestník k jednotlivým projektům, které jsou v dokumentaci obsaženy. Rozdělení bylo vytvořeno jako chronologický průběh jednotlivých cvičení. Vždy je uvedeno pořadové číslo cvičení, následované tématem, kterým se cvičení zabývá a poté odkazy na zdrojové kódy. Vzhledem k přehlednosti jazyka markdown a jeho podpory ze strany Doxygenu, byla využita právě pro formátování úvodní strany. Na závěr správnost zápisu zkontroloval nástroj Markdown Lint Tool.

5.4 Umístění komentujících bloků

Vzhledem k velikosti jednotlivých souborů a také možnosti skrývat komentáře v prostředí Codelite bylo preferováno vkládat komentující bloky interně, tedy přímo do zdrojového

kódu. Výhodou je především snadná údržba. U projektů, které obsahují hlavičkový soubor, byly umístěny komentující bloky do tohoto souboru. Převážná většina projektů tento soubor nemá a tak v těchto případech vložil dokumentující bloky přímo do zdrojového souboru.

5.5 Komplikace se stejnojmennými funkcemi

Dříve než byly komentovány jednotlivé projekty, vyskytla se komplikace spojená se stejnojmennými funkcemi. Jednalo se konkrétně o funkci main(), která byla obsažena v každém projektu a označuje místo, kde program začíná. Nástroj je primárně určen pro vygenerování dokumentace pro jeden konkrétní projekt bez ohledu na jeho velikost, zde se jedná o několik poměrně malých projektů. Pro řešení byly zohledněny tři možnosti.

5.5.1 Rozdělení projektů do externích dokumentací

Jedním z řešení bylo vytvoření separovaných dokumentací pro jednotlivé projekty a následné propojení. I přesto, že by se jednalo o pravděpodobně funkční řešení, velké množství projektů by proces linkování značně komplikovalo a zautomatizování generování by bylo podmíněno vytvořením skriptu. Další komplikace by se vyskytly v případě přidávání dalších projektů. Z těchto důvodů je řešení nepřípustné a použití externí dokumentace je možné pouze v případě několika málo projektů.

5.5.2 Použití modulů

Doxygen nabízí rozdělení částí zdrojových kódů do tzv. modulů, což jsou skupiny, které jsou na sobě nezávislé. Toto řešení je značně elegantnější a po určitý čas bylo využíváno i v této dokumentaci. Rozdělením kódu do modulů vznikají tři místa pro komentování. Je možné okomentovat modul jako celek, následně jednotlivé soubory a také jednotlivé prvky zdrojového kódu, jako jsou funkce, proměnné aj. V případě, že by některý z komentářů chyběl, i přesto by byla vygenerována stránka v dokumentaci, která by obsahovala pouze název, ale nepodávala by další informace. Zde se jednalo o malé projekty a nucená tvorba komentářů pro popis modulu a také popis souboru by mohla vést k redundantním informacím nebo prázdným stránkám. Tento způsob byl nakonec zamítnut. Použití modulů je vhodné v případě, že by se jednalo o větší množství zdrojových souborů, které lze popsat jako celek.

5.5.3 Nahrazení popisem souboru

Poslední řešení, které bylo nakonec zvoleno je elegantní a nebyly nalezeny žádné komplikace. Vzhledem k tomu, že se jednalo funkci main(), která v projektech definovala postup celého programu, byl tento popis zanechán prázdný, nicméně tato informace byla vložena jako komentář celého souboru. Řešení tak nevytváří žádné redundantní ani prázdné stránky v dokumentaci.

5.6 Struktura komentářů

Prvky jednotlivých souborů byly rozděleny do tří kategorií. Základem je popis souboru, který je obvykle nejpodrobnější. Další podstatnou částí jsou popisy funkcí a do poslední části bylo umístěno vše ostatní. Každé z těchto skupin byl přiřazen specifický komentující styl tak, aby bylo snadné udržovat dokumentaci aktuální.

5.6.1 Popis souboru

Na začátku každého souboru je umístěn komentující blok. Zde musí být obsaženo klíčové slovo *file,* které označí celý soubor pro použití v dokumentaci. Následuje stručný popis (*brief)* ukázkového programu. Dále se zde nachází sekce pro detailní popis, která nevyžaduje žádné klíčové slovo, ale pro oddělení od stručného popisu je doporučeno začít prázdným řádkem. V této části se nachází popis funkce main() a tedy celého programu. Na závěr je umístěna kategorie "Viz také" (*sa)*, která je využívána v případě, že existují další soubory pro stejné cvičení, uživatel se tak může jednoduše mezi nimi přepínat. *I* přesto, že Doxygen nabízí hned několik způsobů rozdělení do sekcí, bylo rozhodnuto do šablony použít pouze výše zmíněné, především z důvodu snadné budoucí údržby. V některých ze souborů lze nalézt ještě klíčová slova pro poznámku (*note)* a varování (*warning)*. Do poznámky byly umístěny například odkazy na další cvičení, které se zabývají podobným příkladem, avšak s odlišným řešením. Blok s varování obsahuje informace o nutném nastavení kompilátoru pro správnou funkčnost.

5.6.2 Popis funkcí

Každý popis funkce lze rozdělit na pomyslné čtyři části. Stejně jako u souboru, začátek je popsán pomocí stručného (*brief*) a detailního popisu. Ve spoustě případů funkce nebyla komplikovaná a stručný popis byl dostatečný. Pokud funkce obsahuje parametry, pak každý z nich je popsán klíčovým slovem *param* s hranatou závorkou, která obsahuje *in* nebo

out podle toho, jestli se jedná o vstupní, případně výstupní parametr. Toto rozdělení je nezbytné, zvláště v případě jazyka jako je C, protože u parametrů datového typu ukazatel nelze tato vlastnost určit bez zkoumání zdrojového kódu, který se v dokumentaci nenachází. Poslední část popisuje návratový typ (*return*). V případě datového typu *void* je tento popis vynechán.

5.6.3 Popis ostatních prvků

Do této skupiny jsou řazeny výčtové typy, globální proměnné a makra pro preprocesor. V případě, že byly potřebné dodatečné informace, nebylo použito žádné klíčové slovo, pouze byl text umístěn do jednořádkového komentujícího bloku, tím se automaticky označil jako stručný popis. Výjimečně byl využit detailní popis, především tedy u výčtových typů.

5.7 Vizuální úprava textu

Mnohé z projektů využívají piny, jak na výukovém kitu nebo přímo na mikropočítači. Seznam použitelných pinů byl umístěn vždy do popisu souboru a naformátován do tabulky. Pro vytvoření tabulky je využíváno HTML syntaxe a Doxygen dodal výchozí vzhled pomocí kaskádových stylů. Dále bylo využito ztučnění textu jako odlišení nadpisů tematických bloků nebo kurzívu pro označení textu, který se nachází přímo ve vývojovém prostředí.

5.8 Vzhled dokumentace

Co se týká vzhledu výstupní dokumentace, tak do výsledku nebyly vložený žádné vlastní úpravy. Byly vyzkoušeny různé barevné odstíny, ale v rámci čitelnosti byl ponechán výchozí modrý vzhled.

5.9 Problémy s exportem z formátu Latex do PDF

K převodu byl použit software TeX Live. Ve výstupní složce latex byl spuštěn dávkový soubor *make.bat.* V případě, že byly použity anglické ovládací prvky, vše proběhlo v pořádku a výstup byl k dispozici jako nový soubor pojmenovaný *refman.pdf.* V případě českého jazyka při provádění příkazu se vyskytla chyba *! Paragraph ended before \@cline was complete*. Po bližším pročtení chybových hlášení bylo zjištěno, že problém nejspíše vzniká při tvorbě tabulky, která používá právě *cline*. Důvodem je TeX balíček *babel* pro český, ale také slovenský jazyk. Vzniklá chyba může být spojená s tím, že balíček přepisu-

je chování pomlčky [16]. Funkčního výstupu bylo docíleno přidáním následujících řádku do souboru *refman.tex*, který se nachází taktéž ve složce *latex*.

```
\usepackage{regexpatch}
\makeatletter
% Change the `-` delimiter to an active character
\xpatchparametertext\@@@cmidrule{-}{\cA-}{}{}
\xpatchparametertext\@cline{-}{\cA-}{}{}
\makeatother
```

Výše uvedený text vyhledá příkazy *cmidrule* a *cline*, které používají v parametru pomlčku. Pomlčka standardně slouží k definování rozsahu mezi dvěma sloupci, avšak toto chování je balíčkem *babel* pro český jazyk upraveno. Po upravení souboru je význam pomlčky pro tyto příkazy navrácen. Je nutné podotknout, že tuto úpravu musí uživatel provést po každém generování dokumentace.

5.9.1 Oznámení o zjištěné chybě

Zjištěná chyba byla nahlášena vývojáři programu Doxygen pomocí webu BUGZILLA.GNOME.ORG, který je výhradně určen pro oznamování. Chyba je zde vedena pod ID s číslem 782082.

ZÁVĚR

Cílem práce bylo vytvoření materiálu, který usnadní vytváření dokumentace ke zdrojovým kódům s využitím nástroje Doxygen.

Dokumentace je důležitá především pro znovupoužitelnost zdrojového kódu. Základem dobré dokumentace je její jednoduchost a konzistentnost, kterou by měl zajistit vývojový tým. Doxygen je nástroj, který poskytuje rozhraní k dosažení žádané kvality. Na základě specificky psaných komentářů je schopen generovat přehlednou dokumentaci do různých výstupních formátů. Snadná orientace v dokumentaci je zajištěna pomocí křížových odkazů a abecedně řazených seznamů všech prvků zdrojového kódu. Na trhu existují další nástroje s podobným cílem, nicméně jen málokterý lze srovnávat s možnostmi a individuálním nastavením právě Doxygenu. Díky této přizpůsobitelnosti a snadnému použití nalezne uplatnění v široké škále projektů a i přesto, že jeho první verze byly vydány před téměř dvaceti lety, stále na něm probíhá aktivní vývoj.

Pro usnadnění tvorby dokumentace byla vytvořena skupina šablon, která obsahuje konkrétní příklady jednotlivých komentujících bloků včetně zdrojového kódu. Dále byl popsán postup tvorby dokumentace pro vytvořený jednoduchý projekt. Na závěr bylo demonstrováno použití na dříve existujícím výukovém materiálu pro předmět Programování mikropočítačů. Součástí bakalářské práce je také příručka pro použití v podobě přílohy. Příručka se snaží shrnout proces tvorby do tří hlavních kroků, které jsou následně rozepsány podrobněji.

Práce by měla přispět ke snadnějšímu využití nástroje Doxygen a tím ke zkvalitnění dokumentace a přehlednosti zdrojového kódu.

SEZNAM POUŽITÉ LITERATURY

[1] KOTULA, Jeffrey. Source Code Documentation: An Engineering Deliverable. Proceedings of the Technology of Object-Oriented Languages and Systems. Washington, DC, USA: IEEE Computer Society, 2000, 1(34), 505.

[2] MCCONNELL, Steve. Code complete [online]. 2nd. Redmond, Washington: Microsoft Press, 2004 [cit. 2017-03-21]. ISBN 0735619670. Dostupné z: https://books.google.cz/books?id=LpVCAwAAQBAJ

[3] HYDE, Randall. Write Great Code: Understanding the Machine [online]. No Starch Press, 2004 [cit. 2017-03-27]. ISBN 1593270038. Dostupné z: https://books.google.cz/books?id=tfP4UND566AC

[4] Write the Docs [online]. Write the Docs, 2017 [cit. 2017-03-30]. Dostupné z: http://www.writethedocs.org/

[5] MARTIN, Robert C. Clean code: a handbook of agile software craftsmanship. Upper Saddle River, NJ: Prentice Hall, 2009. ISBN 978-013-2350-884.

[6] VAN HEESCH, Dimitri. Doxygen [online]. Netherlands, 2016 [cit. 2017-04-03]. Dostupné z: www.doxygen.org

[7] Doxygen: Official doxygen git repository. GitHub: Built for developers [online]. San Francisco, 2017 [cit. 2017-04-03]. Dostupné z: https://github.com/doxygen/doxygen

[8] Docurium: Doxygen replacement for the libgit2 project. GitHub: Built for developers [online]. San Francisco, 2017 [cit. 2017-04-05]. Dostupné z: https://github.com/libgit2/docurium

[9] Sphinx: Python Documentation Generator [online]. Sphinx team, 2017 [cit. 2017-04-05]. Dostupné z: http://www.sphinx-doc.org/en/stable/

[10] VALURE, Greg. Natural Docs [online]. 2011 [cit. 2017-04-13]. Dostupné z: http://www.naturaldocs.org/

[11] BARR, Michael a Antony J MASSA. Programming embedded systems: with C and GNU development tools. 2nd ed. Sebastopol: O'Reilly, 2006, xxi, 301 s. [cit. 2017-03-21] ISBN 978-0-596-00983-0

[12] Doxygen information flow. In: Doxygen [online]. Netherlands, 2016 [cit. 2017-04-25]. Dostupné z: https://www.stack.nl/~dimitri/doxygen/manual/infoflow.png

[13] PINKER, Jiří. Mikroprocesory a mikropočítače. 1. vyd. Praha: BEN - technická literatura, 2004, 159 s. [cit. 2017-03-21] ISBN 80-7300-110-1

[14] CATSOULIS, John. Designing embedded hardware. 2nd ed. Sebastopol, CA: O'Reilly, 2005, xvi, 377 s. [cit. 2017-04-25] ISBN 0596007558

[15] Markdown. GRUBER, John. Daring fireball [online]. The Daring Fireball Company, 2017 [cit. 2017-04-26]. Dostupné z: https://daringfireball.net/projects/markdown/

[16] Slovak and Czech babel gives problems with cmidrule and cline. TeX - LaTeX Stack Exchange [online]. Stack Exchange, 2013 [cit. 2017-05-02]. Dostupné z: https://tex.stackexchange.com/questions/111999/slovak-and-czech-babel-gives-problems-with-cmidrule-and-cline

SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK

TODO	Označení pro nedokončené části zdrojového kódu
GNU	Hnutí založené na svobodném softwaru
IDL	Interactive Data Language
Tcl	Tool Command Language
HTML	HyperText Markup Language
RTF	Rich Text Format
XML	Extensible Markup Language
Qt	Framework pro vytváření aplikačního softwaru
PDF	Portable Document Format
BSD	Berkeley Software Distribution
JSON	JavaScript Object Notation
MIT	Massachusetts Institute of Technology
API	Application Programming Interface
GPL	General Public License
ASCII	American Standart Code for Information Interchange

SEZNAM OBRÁZKŮ

SEZNAM TABULEK

SEZNAM PŘÍLOH

PŘÍLOHA P I: OBSAH PŘILOŽENÉHO CD

Příloha obsahuje informace o obsahu na přiloženém CD

Přiložené CD obsahuje:

- bakalářskou práci ve formátu PDF/A
- složku s ukázkovou dokumentací včetně výstupu do formátu HTML, Latex a PDF
- složku s šablonami pro tvorbu dokumentace
- složku s vlastním projektem v podobě teplotního senzoru s nastavitelnou indikací

PŘÍLOHA P II: PŘÍRUČKA K POUŽITÍ NÁSTROJE DOXYGEN

V této příručce budou vysvětleny všechny potřebné kroky pro základní použití nástroje Doxygen.

1 INSTALACE

Vzhledem k faktu, že se jedná o program s otevřeným zdrojovým kódem, existují dva způsoby jak jej nainstalovat. Nabízí se instalace z již předpřipraveného binárního souboru nebo pomocí zdrojového kódu. Instalaci ze zdrojového kódu lze využít, vyžadujeme-li nejnovější nebo naopak starší verzi, pro kterou nejsou binární soubory k dispozici. Další výhodou je možnost úprav programu. Zdrojový kód programu je zveřejněn na webu Github (WWW.GITHUB.COM/DOXYGEN/DOXYGEN). Kontrolu správnosti instalace lze provést příkazem pro získání čísla verze.

doxygen -v

Po obdržení čísla verze je vše připraveno k použití a postup instalace můžete přeskočit.

1.1 Windows - instalace z binárního souboru

Jedná se o uživatelsky velmi snadnou operaci, kde postačí stažení instalačního souboru z oficiálních stránek. Po spuštění staženého souboru se zobrazí průvodce, kde můžeme zvolit adresář instalace a vybrat volitelné komponenty, které chceme nainstalovat.

1.2 Windows - instalace ze zdrojového kódu

Před kompilací je nutné nainstalovat následující závislosti:

1. cmake
2. winflexbison
3. python (verze 2.6 nebo vyšší)
4. Qt 4.3 (pro komponentu doxywizard)

Jako kompilátor je oficiálně testován pouze ten, dodávaný s nástrojem Visual Studio 2013 express edice, avšak i jiné kompilátory mohou fungovat. Před kompilací je nutné přejmenovat flex a bison tak, aby neobsahovali předponu "win_".

Po instalaci závislostí stáhneme zdrojový kód. Stažený zdrojový kód je třeba rozbalit do připraveného, prázdného adresáře například pomocí programu 7-Zip. V tomto adresáři

vytvoříme novou složku *"build"*, ve které se bude nacházet sestavený program. Na závěr otevřeme terminál, přesuneme se do složky *"build"* a provedeme sestavení pomocí příkazu

cmake -G "Visual Studio 12 2013" ..

1.3 Unix - instalace z binárního souboru

Pro systém Mac OS X je předpřipraven soubor s příponou .dmg, který běžným způsobem nainstalujete. Co se týká populárních linuxových distribucí, například Debian, Fedora nebo Arch, ty mají Doxygen k dispozici ve svých oficiálních repozitářích. Tento typ instalace je výhodný, protože správce balíčků za nás vyřeší závislosti, instalaci, odinstalaci a aktualizace. Používáte-li distribuci, která tento balíček neobsahuje v repozitářích, poté lze stáhnout zkompilovaný soubor z oficiálních stránek projektu. Pokud ani tato možnost nebude vyhovující, je třeba provést instalaci ze zdrojového kódu, popsanou v části (Unix - instalace ze zdrojového kódu).

Příklad použití zde uvedu pro instalaci na systému Debian.

sudo apt-get update && sudo apt-get install doxygen

Pro doinstalování komponenty doxywizard poté platí

sudo apt-get install doxygen-gui

1.4 Unix - instalace ze zdrojového kódu

Pro kompilaci je nutné nainstalovat následující závislosti:

1. flex
2. bison
3. libiconv
4. make
5. strip
6. cmake
7. Qt 4.3 (pro komponentu doxywizard)

Po instalaci závislostí je třeba získat zdrojový kód. Pro stažení můžeme přejít v terminálu do předpřipravené, prázdné složky a využít programu git pro stažení.

git clone https://github.com/doxygen/doxygen.git

Ve složce doxygen vytvoříme prázdnou složku build a přejdeme do ní

mkdir doxygen/build && cd doxygen/build

Následně připravíme kompilaci pomocí nástroje cmake, který rozhodne na jaké platformě se nacházíme a zkontroluje splněné závislosti

cmake -G "Unix makefiles" ..

Kompilaci dokončíme programem make, na který zavoláme

make

Na závěr provedeme instalaci zkompilovaných souborů do systému

sudo make install

Pokud vše proběhlo bez závažných chyb, program se poté nachází v adresáři

/usr/local/bin/doxygen

2 TVORBA DOKUMENTACE

Tvorba dokumentace se skládá ze tří hlavních kroků. Prvním krokem je vytvoření souboru s nastavením a jeho případná editace.

2.1 Soubor s nastavením

2.1.1 Vygenerování

Soubor s nastavením je textový soubor s formátování ASCII. Jedná se o seznam přiřazení. Ke každé možnosti, kterou nám Doxygen poskytuje, můžeme vložit za rovnítko hodnotu. Pro vytvoření si nejdříve otevřeme příkazový řádek. Následně je vhodné přejít do složky, kde chceme tento soubor uložit. V této složce poté zavoláme následující příkaz.

doxygen -g [Název_souboru]

V případě, že nezadáme název souboru, bude automaticky vytvořen soubor s názvem *Doxyfile*. Pro předchozí příkaz můžeme použít ještě přepínač *-s*, ten vynechá všechny komentáře jednotlivých nastavení.

2.1.2 Editace

Vygenerovaný soubor můžeme otevřít v jakémkoliv textovém editoru. Tento soubor obsahuje kolem 250 různých nastavení, kde výchozí hodnoty jsou zvoleny tak, aby vyhovovaly většině uživatelům a pro zprovoznění obvykle postačí změnit pouze nastavení týkající se rozmístění souborů v adresáři. V následujících podkapitolách naleznete postup nastavení projektu obecně, zmíněny jsou pouze některé z možností. Specifické vlastnosti jsou detailně popsány ve vygenerovaném nastavení pomocí komentářů. Pro funkčnost tohoto souboru je nutné zachovat pravidlo, že každé z nastavení je na novém řádku a komentáře začínají znakem #.

2.1.3 Nastavení projektu

Prioritou je upravit jméno projektu (PROJECT_NAME), toto jméno bude uvedeno na každé stránce dokumentace, stejně jako stručný popis (PROJECT_BRIEF). Verzování lze docílit změnou čísla projektu (PROJECT_NUMBER). Další důležitou položkou je výstupní adresář (OUTPUT_DIRECTORY). Pokud není specifikován bude dokumentace vytvořena v adresáři, ze kterého voláme příkaz pro generování. Nastavení jazyka pro všechny ovládací prvky se provádí v nastavení výstupní jazyk (OUTPUT_LANGUAGE). V případě, že budete dokumentovat výhradně jeden z následujících jazyku - C, Java, Fortran nebo VHDL, je doporučeno nastavit příslušnou optimalizaci (OPTIMIZE_OUTPUT_{*Jazyk*}) na hodnotu *YES*. Zajímavé možnosti v této kategorii jsou úprava výstupního kódování (DOXYFILE_ENCODING) a nastavení loga projektu (PROJECT_LOGO).

2.1.4 Nastavení sestavení

Zde lze nastavit, které části kódu mají, či nemají být obsaženy v dokumentaci. A také podle kterého kritéria mají být položky v dokumentaci seřazeny.

2.1.5 Nastavení chybových hlášení

Chybová hlášení vedou k odhalení chyb v dokumentaci a jejich vypnutí by mělo být použito pouze ve výjimečných případech. Jediné zajímavé nastavení je přesměrování chybových hlášení (WARN_LOGFILE) ze standardního chybového výstupu (stderr) do souboru.

2.1.6 Nastavení vstupních souborů

Nejdůležitějším nastavením je zde umístění zdrojových souborů (INPUT). Dále je možné nastavit kódování těchto souborů (INPUT_ENCODING). Užitečné může být nastavení

vlastních přípon vstupních souborů (FILE_PATTERNS). Můžeme taktéž některé adresáře nebo soubory z dokumentace vynechat (EXCLUDE). Zajímavá volba, kterou mohou využít například projekty hostované na serveru GitHub, je nastavení markdown souboru jako uvítací stránky (USE_MDFILE_AS_MAINPAGE). Markdown je zjednodušený značkovací jazyk, který slouží k formátování textu, který lze poté převést na HTML.

2.1.7 Nastavení procházení zdrojových kódů

Tato kategorie se zabývá propojením jednotlivých prvků. Lze nastavit, které části kódu mají být součástí dokumentace. V případě, že vyžadujete, aby žádné zdrojové kódy nebyly obsaženy v dokumentaci, poté nastavte položky (SOURCE_BROWSER) a (VERBATIM_HEADERS) na hodnotu *NO*.

2.1.8 Nastavení alfabetického výčtu

Ve výchozí konfiguraci je tato možnost zapnutá a jedná se o vypsání všech položek jako struktur, výčtových typů a tříd do seznamu, který je abecedně seřazen. Tuto funkcionalitu ocení především velké projekty.

2.1.9 Nastavení výstupních formátů

Následuje skupina nastavení, které se týkají jednotlivých výstupních formátů. Ve výchozím nastavení se generuje dokumentace do formátu HTML a LaTeX. V případě potřeby je možné přidat další formáty jako RTF, man pages, XML nebo změnit podobu jednotlivých výstupů. U HTML lze například nastavit hlavička, patička, barevné schéma nebo změnit celkové rozložení dodáním vlastního souboru s kaskádovým stylem.

2.1.10 Nastavení preprocesoru

Předposlední část je věnována zpracování zdrojového kódu, který má na starost preprocesor. Nalezneme zde nastavení týkající se zpracování maker.

2.1.11 Nastavení externích zdrojů

Využíváte-li knihovny nebo jiné nástroje, pro které již existuje dokumentace vytvořená pomocí Doxygenu, je výhodné jí připojit externě. To lze provést přidáním mezi hodnoty (TAGFILES). Takto přidaná dokumentace už není zpracovávána při generování. Tímto způsobem lze ušetřit nejen výpočetní čas, ale také paměť. Dalším důvodem může být uzavřenost zdrojového kódu vkládaných knihoven a jedná se tedy o jedinou možnost.

2.2 Okomentování zdrojového kódu

Generování dokumentace je provedeno na základě struktury zdrojového kódu a příslušných komentářů k jednotlivým prvkům. Doxygen rozeznává několik typů komentářů a různé způsoby umístění.

2.2.1 Typ komentářů

V dokumentaci dělíme komentáře do dvou skupin

1. Stručný popis
2. Dodatečné informace

Stručný popis je obvykle jednořádkový a ve výstupní dokumentaci se nachází vždy na první pozici za názvem prvku. V dodatečných informacích to mohou být detailní popisy implementace, u funkcí popis vstupních a výstupních parametrů, návratové hodnoty. Obě tyto části jsou dobrovolné. Pokud bychom neuvedli ani jeden z komentářů pro daný prvek, bude i přesto zahrnut v dokumentaci společně s informacemi, které lze vyčíst ze zdrojového kódu. Taktéž je možné použít více komentářů stejného typu pro jeden prvek. Tyto informace budou sloučeny, nicméně pořadí nejsme schopni ovlivnit, a proto se tento přístup nedoporučuje používat, zvláště v případě stručných popisů.

2.2.2 Dokumentující bloky

Z důvodu rozlišení mezi běžným komentářem a komentářem, který má být zahrnut v dokumentaci, Doxygen hledá ty ve specifickém tvaru. Nejpoužívanější tvary se nazývají podle původu JavaDoc style a Qt style, kde v obou případech jsou hvězdičky uvnitř těla dobrovolné. V následujících ukázkách budu používat výhradně JavaDoc style, nicméně se jedná o rovnocenné zápisy.

```
/**
 * JavaDoc style komentář
 */
```

```
/*!
 * Qt style komentář
 */
```

K dispozici je několik způsobů jak odlišit stručný popis od dodatečných informací. Jedním z nich je použití únikové sekvence pomocí znaku zpětného lomítka a poté speciální slovo *brief*. Pro oddělení od dodatečných informací je vyžadován prázdný řádek za stručným popisem.

```
/**
 * \brief Stučný popis
 *
 * Detailní popis
 */
```

Další možnost můžeme použít v případě, že budeme používat pouze stručný popis. Jedná se o zápis jako jednořádkový komentář.

```
//* Stručný popis pomocí jednořádkového komentáře
```

2.2.3 Umístění bloků

Doxygen nabízí tři možnosti, kde můžeme dokumentační bloky umístit. Každý ze způsobů má své výhody a omezení.

2.2.3.1 Před dokumentovaný prvek

Jedná se o nejčastější způsob. Před jakýkoliv prvek zdrojového kódu na nový řádek umístíme dokumentující blok, automaticky je pak k němu přiřazen. Výhodou tohoto umístění je, že se nemusíme starat o případné přejmenování dokumentovaného prvku.

2.2.3.2 Na stejný řádek

Pokud vyžadujeme pouze stručný popis, stačí za středník umístit jednořádkový dokumentující blok a obsah tohoto bloku je přiřazen jako *brief* popis k aktuálnímu řádku. Využití je především u krátkých řádků, jako je popis globálních proměnných, případně jednotlivých prvků ve výčtových datových typech. Hlavní výhodou je úsporný zápis.

2.2.3.3 Odděleně od zdrojového kódu

K dispozici je také možnost separovat dokumentační bloky od zdrojového kódu úplně. Tento způsob je podmíněn tím, že na začátek každého dokumentačního bloku umístíme klíčové slovo o jaký prvek se jedná a poté jeho konkrétní název ve zdrojovém kódu. Komplikace nastávají v případě přejmenovávání, kde musíme dbát na to, aby jsme přejmenování provedli jak ve zdrojovém kódu tak v separovaném dokumentačním souboru.

2.2.4 Klíčová slova

Pro zápis v dokumentačních blocích lze využít klíčových slov, které ovlivňují výstup dokumentace. Od obyčejného textu jsou odlišeny pomocí únikové sekvence v podobě zpětného lomítka, které je následováno klíčovým slovem bez mezer.

2.2.4.1 Úvodní stránka

Vytvoření úvodní stránky lze provést klíčovým slovem *mainpage* na začátku souboru, za nímž může následovat název. Pokud je vynechán, bude použit název projektu. Také můžeme vytvořit nový soubor s příponou .dox, který neobsahuje žádný zdrojový kód, pouze dokumentační bloky a bude sloužit jako hlavní stránka. Syntaxe pro takto vytvořený soubor je naprosto totožná, jedinou podmínkou je použití vhodného souboru s nastavením, tak aby se přípona nacházela v seznamu FILE_PATTERNS.

2.2.4.2 Označení souboru

Každý soubor, který obsahuje komentující bloky, musí být označen klíčovým slovem *file*, které může být následováno názvem souboru. Explicitní udání názvu je nutné pouze pro externí komentování, čímž propojíme dokumentující soubor se zdrojovým. Tím, že neoznačíme soubor, pak žádný z jeho komentujících bloků není zahrnut do dokumentace. Další možností jak zahrnout soubor do dokumentace je použití \page *[název]*. Tento způsob se využívá pro soubory, které neobsahují zdrojový kód a zároveň se nejedná o úvodní stranu. Takový soubor může obsahovat například licenci. V případě existence takových souborů, je vytvořen jejich seznam a lze na ně odkázat pomocí reference (*ref)*.

2.2.4.3 Vizuální úprava

V následující tabulce (Tab. 1) jsou vypsány možnosti, jak upravit text po vizuální stránce. Pro aplikování vizuálních změn na skupinu slov, lze použít zápis ve druhém sloupci, který je inspirován jazykem HTML.

Tabulka 3 - Doxygen - klíčová slova pro vizuální úpravu

Zápis pro jedno slovo	**Zápis pro více slov**	**Význam**
\em *[slovo]*	*[více_slov]*	kurzíva
\b *[slovo]*	*[více_slov]*	ztučnění
\c *[slovo]*	<tt> *[více_slov]* </tt>	neproporcionální písmo
\ref *[reference]*		odkaz na prvek
\li *[text]*		řádek nečíslovaného seznamu
	\f$ *[vzorec]* \f$	vzorec uvnitř textu
	\f[*[vzorec]* \f]	vzorec na novém řádku, vycentrován na střed
\image *[výstupní_formát] [název]*		vložení obrázku
\diafile *[název]*		vložení diagramu

Speciálním případem je vkládání zdrojového kódu přímo do textu dokumentace. U vkládaného kódu je automaticky zvýrazněná syntaxe, která je podmíněná zadáním správné přípony programovacího jazyka ve složených závorkách. Následující ukázka je pro použití s jazykem C.

```
/**
 * \code{.c}
 * printf("Hello world");
 * \endcode
*/
```

2.2.4.4 Rozdělení do sekcí

Každá dokumentační strana je rozdělena do několika sekcí, vizuálně oddělených, pro snazší orientaci. Uvádění těchto informací je dobrovolné, stejně jako řazení do sekcí.

Tabulka 4 - Doxygen - klíčová slova pro rozdělení do sekcí

Zápis	**Význam**
\return *[popis]*	popis návratové hodnoty funkce
\param[out] *[název][popis]*	popis výstupních parametrů funkce
\param[in] *[název][popis]*	popis vstupních parametrů funkce
\sa *[reference]*	odkaz na další soubory
\note *[text]*	poznámka
\attention *[text]*	upozornění
\warning *[text]*	pozor
\todo *[text]*	plánované úpravy
\deprecated *[text]*	zastaralý prvek
\bug *[text]*	chyba
\since *[verze]*	od které verze je k dispozici
\version *[verze]*	verze
\author *[autor]*	autor
\copyright *[text]*	copyright
\date *[text]*	datum vytvoření

V případě označení plánovaných úprav, zastaralých prvků nebo chyb, následně jsou vygenerovány seznamy všech takových výskytů, které lze použít pro rychlé vyhledávání.

2.2.4.5 Definování jazyku bloku

Pokud vytváříme více jazykových variant dokumentace, lze tyto popisy umístit do jednoho zdrojového kódu. Pro rozlišení mezi nimi se používá na začátku komentujícího bloku znak tilda následovaná názvem jazyka. V případě, že chceme část dokumentace používat pro všechny jazykové výstupy, název jazyka vynecháme.

2.2.4.6 Definování typu bloku

Určení typu je důležité v případě, že komentující blok se nenachází těsně nad komentujícím prvkem, případně na stejném řádku, v těchto situacích je typ automaticky rozpoznán.

Tabulka 5 - Doxygen - klíčová slova pro definování typu bloku

Zápis	**Význam**
\class *[jméno]*	označení třídy
\def *[jméno]*	označení makra
\enum *[jméno]*	označení výčtového typu
\fn *[deklarace funkce]*	označení funkce
\headerfile *[jméno]*	označení hlavičkového souboru
\interface *[jméno]*	označení rozhraní
\namespace *[jméno]*	označení jmenného prostoru
\overload *[deklarace funkce]*	označení přetížené funkce
\struct *[jméno]*	označení struktury
\typedef *[deklarace datového typu]*	označení uživatelského datového typu
\union *[jméno]*	označení typu union
\var *[deklarace proměnné]*	označení proměnné

2.2.4.7 Rozdělení do modulů

Separaci kódu do skupin lze provést pomocí tzv. modulů. Tento způsob je často řešením problému, kdy používáme stejný název globální proměnné u více souborů, Doxygen automaticky předpokládá, že jde o tutéž paměťovou adresu, a proto dokumentace tohoto prvku spojí. Ve skutečnosti se však může jednat o dva zdrojové kódy, které se kompilují nezávisle na sobě. Kromě umožnění stejných názvů proměnných nebo funkcí jsou moduly také řešením pro dosáhnutí přehlednosti. Všechny moduly jsou při generování dokumentace seřazeny do přehledného seznamu. Důležité je, abychom rozdělovali pouze části, které nejsou na sobě přímo závislé. Syntaxe je následující.

```
/**
\addtogroup nazevModulu
 * @{  */
 Zdrojový kód
/** @} */
```

2.2.4.8 Výběr výstupních formátů

Komentující blok můžeme specifikovat, pro který výstupní formát se má použít (Tab. 4).

Tabulka 6 - Doxygen - klíčová slova pro omezení výstupu podle formátu

Zápis	**Význam**
\htmlonly *[text]* \endhtmlonly	použít pouze pro HTML výstup
\latechonly *[text]* \endlatechonly	použít pouze pro Latex výstup
\manonly *[text]* \endmanonly	použít pouze pro MAN pages výstup
\rtfonly *[text]* \endrtfonly	použít pouze pro RTF výstup
\xmlonly *[text]* \endxmlonly	použít pouze pro XML výstup
\docbookonly *[text]* \enddocbookonly	použít pouze pro DocBook výstup

Další možností, jak omezit výstup je uzavření komentářů mezi klíčové slova \cond *[název_sekce]* a \endcond. Poté bude výstup záviset pouze na položce ENABLED_SECTIONS v souboru s nastavením. Pokud zde bude uveden název, který jsme udali za klíčovým slovem, pak bude obsažen dokumentaci. Stejným názvem můžeme označit několik míst a poté generovat dvě odlišné verze, podle toho, pro koho je výstup určen.

2.2.4.9 Podpora markdown syntaxe

Od verze 1.8.0 je možné využít formátující syntaxi markdown. Povolením v souboru s nastavením získáváme plnou podporu v jakémkoliv komentujícím bloku. Hlavním cílem je zvýšit čitelnost nahrazením formátujících slov speciálními znaky [151].

2.3 Vygenerování dokumentace

Pro závěrečné generování dokumentace postačí následující příkaz

doxygen [název_souboru_s_nastavením]

Jakmile je proces dokončen, v terminálu nalezneme výpis informací o exportu (lze potlačit v nastavení). Výstupní soubory lze nalézt ve složkách, rozdělené podle typu výstupu. Nejsme-li s něčím spokojeni, můžeme kdykoliv provést úpravu a vygenerovat dokumentaci znovu.