# ZADÁNÍ SEMESTRÁLNÍ PRÁCE
# GENERÁTOR DOKUMENTACE ZDROJOVÉHO KÓDU

## Zadání

Naprogramujte v ANSI C přenositelnou[1] **konzolovou aplikaci**, která prohledá vstupní soubor se zdrojovým kódem v programovacím jazyce ANSI C a na základě tam nalezených speciálních značek vygeneruje výstupní soubor s dokumentací vstupního zdrojového kódu ve formátu LaTeX. Vstupem aplikace bude tedy textový soubor se zdrojovým kódem v programovacím jazyce ANSI C opatřeným tzv. *dokumentačními komentáři* uvedenými speciálními značkami podle níže uvedené specifikace. Aplikace tyto značky vyhledá a zpracuje a následně z informací v nich obsažených vygeneruje dokumentaci předaného vstupního zdrojového kódu. Výstupem aplikace bude pak zdrojový soubor ve značkovacím jazyce TeX, který půjde přeložit typografickým systémem LaTeX. Úloha je velmi podobná (ačkoli výrazně zjednodušená) existujícím projektům **JavaDoc**, **Doxygen**, **PasDoc**, apod. Tyto projekty můžete v rámci přípravy k řešení semestrální práce prozkoumat a volně se jimi inspirovat.

Program se bude spouštět příkazem

`ccdoc.exe`[2] ⟨*vstupní soubor*[`.c`|`.h`]⟩ [⟨*výstupní soubor*[`.tex`]⟩]↵

Symbol ⟨*vstupní soubor*⟩ zastupuje povinný parametr – název vstupního souboru se zdrojovým kódem v jazyce ANSI C (může to být jak `.c`, tak hlavičkový soubor `.h`). Symbol ⟨*výstupní-soubor*⟩ pak představuje nepovinný parametr, kterým je jméno souboru s výstupem programu. Není-li druhý, nepovinný, parametr uveden, směrujte výstup programu do souboru pojmenovaného stejným jménem, jaký má vstupní soubor, ke kterému je připojen suffix `-doc` a přípona `.tex` a tento soubor bude umístěný ve stejné složce, kde se nachází vstupní soubor.

Úkolem Vámi vyvinutého programu tedy bude:

1. Načíst vstupní soubor se zdrojovým kódem v jazyce ANSI C.
2. Projít tento vstupní soubor, nalézt v něm připojení všech **lokálních** hlavičkových souborů (příkazy preprocesoru `#include "..."`) a projít i všechny takto připojené lokální hlavičkové soubory a jim odpovídající soubory zdrojové (tj. je-li připojen hlavičkový soubor `mylib.h`, je třeba hledat také odpovídající zdrojový soubor `mylib.c`; uvědomte si, že i tyto soubory mohou obsahovat příkazy preprocesoru `#include "..."`).
3. Ve všech vstupních souborech (i těch, které nejsou předané přímo na příkazové řádce, ale jsou importovány prostřednictvím příkazu preprocesoru `#include "..."`) nalézt dokumentační komentáře (specifikované níže).
4. Všechny nalezené komentáře zpracovat – zajistit jejich případné spojení (protože funkce může být okomentována dokumentačním komentářem jak v hlavičkovém souboru, tak také v souboru `.c`) – a vypsat jim odpovídající výstup do souboru ve formátu LaTeX podle pokynů uvedených dále v textu tohoto zadání.

Váš program může být během testování spuštěn například takto (v operačním systému Windows):

[1] Je třeba, aby bylo možné Váš program přeložit a spustit na PC s operačním prostředím Win32/64 (tj. operační systémy Microsoft Windows NT/2000/XP/Vista/7/8/10/11) a s běžnými distribucemi Linuxu (např. Ubuntu, Debian, Red Hat, atp.). Server, na který budete Vaši práci odevzdávat a který ji otestuje, má nainstalovaný operační systém Debian GNU/Linux 11 (bullseye) s jádrem verze 5.10.0-20-amd64 a s překladačem gcc 10.2.1.

[2] Přípona `.exe` je povinná i při sestavení v Linuxu, zejm. při automatické kontrole validačním systémem.

```
ccdoc.exe "E:\My Work\Test\test01.c" test01-doc.tex↵
```

nebo takto:

```
ccdoc.exe "E:\My Work\Test\test01.c" doc\test01-doc.tex↵
```

nebo pouze takto:

```
ccdoc.exe work_mdl.c↵
```

Druhý z uvedených příkladů spuštění by měl vést k vytvoření výstupního souboru `work_mdl-doc.tex` v tomtéž adresáři, kde se nachází zdrojový soubor `work_mdl.c`.

Spuštění v UNIXových operačních systémech (GNU/Linux, FreeBSD, macOS, atp.) může vypadat např. takto:

```
ccdoc.exe /home/user/work/test01.c test01-doc.tex↵
```

Uvažujte všechny možné specifikace jak vstupního, tak výstupního souboru, které jsou přípustné v daném operačním systému. Pokud nebude na příkazové řádce uveden alespoň jeden argument (tj. jméno souboru se zdrojovým kódem), vypište chybové hlášení a stručný návod k použití programu (v angličtině).

Hotovou práci odevzdejte v jediném archivu typu ZIP prostřednictvím automatického odevzdávacího a validačního systému. Postupujte podle instrukcí uvedených na webu předmětu. Archiv nechť obsahuje všechny zdrojové soubory potřebné k přeložení programu, **makefile** pro Windows i Linux (pro překlad v Linuxu připravte soubor pojmenovaný `makefile` a pro Windows `makefile.win`) a dokumentaci ve formátu PDF vytvořenou v typografickém systému TEX, resp. LATEX. Bude-li některá z částí chybět, kontrolní skript Vaši práci odmítne.

## Popis činnosti programu

Program projde vstupní zdrojový soubor, jehož jméno bylo předáno argumentem na příkazové řádce, a nalezne všechny připojené hlavičkové soubory (podporované přípony jsou `.c` nebo `.h`) ve všech připojených souborech (minimální úroveň zanoření vkládaných souborů příkazem preprocesoru `#include`, kterou vytvořený program musí zvládnout zpracovat, je 4). Analyzujte však pouze **lokální hlavičkové soubory**, tedy ty, které jsou připojené příkazem preprocesoru `#include` se souborem uvedeným v uvozovkách, tj. např. `#include "myheader.h"`. Hlavičkové soubory z knihovny překladače (ty uvedené v lomených závorkách) ignorujte, stejně k nim nelze (jednoduše) zjistit cestu.

Ve všech nalezených souborech bude program vyhledávat řetězce znaků (viz část Specifikace hledaných řetězců) odpovídající formátem dokumentačním komentářům. Nalezené řetězce (dokumentační komentáře) program přeformátuje do notace TEXu a vypíše je do výstupního souboru (viz část Specifikace výstupu programu). Jak již bylo uvedeno výše, některé komentáře bude třeba **sloučit**, protože určité části kódu je možné okomentovat jak v hlavičkovém souboru, tak v příslušném souboru se zdrojovým kódem modulu. Zamyslete se tedy nad vhodnou vnitřní reprezentací komentářů, aby bylo takové slučování snadno proveditelné.

## Specifikace hledaných řetězců

Program bude vyhledávat pouze ty komentáře ve zdrojových souborech, které budou uvedeny třemi znaky '`/**`' nebo '`/*!`' – stejný způsob používá Doxygen (vizte URL `https://www.doxygen.nl/manual/index.html`). Řádka v souboru těmito znaky nemusí začínat, ale od výskytu sekvence znaků '`/**`' nebo '`/*!`' již zbytek řádky neobsahuje jiné znaky než bílé nebo konec řádky. Každá

další řádka může na začátku obsahovat bílé znaky a prvním viditelným znakem bude '*'. Následuje další text komentáře ke zpracování. Např. takto:

```
/**
 * A brief history of JavaDoc-style (C-style) comments.
 *
 * This is the typical JavaDoc-style C-style comment. It starts with two
 * asterisks.
 *
 * @param theory Even if there is only one possible unified theory. it is just a
 *               set of rules and equations.
 */
```

Kromě obvyklého textu bude program v komentáři vyhledávat **speciální dokumentační značky**. Značky budou vždy pouze jednořádkové a označují úseky dokumentačního textu se specifickou sémantickou rolí. Text mimo značky je prostý popis dokumentovaného úseku kódu (funkce, struktury, apod.).

Program bude rozpoznávat a zpracovávat minimálně následující **speciální dokumentační značky** (vychází z Doxygenu, takže v jeho dokumentaci je případně možné najít detailní popis):

- `@brief` ⟨*stručný popis*⟩
- `@details` ⟨*méně stručný popis*⟩
- `@param` ⟨*typ*⟩ ⟨*název parametru funkce*⟩ ⟨*popis parametru funkce*⟩
- `@return` ⟨*typ návratové hodnoty funkce*⟩ ⟨*popis návratové hodnoty funkce*⟩
- `@author` ⟨*jméno autora kódu*⟩
- `@version` ⟨*číslo verze kódu*⟩

Pro zjednodušení můžete předpokládat, že text následující za speciální dokumentační značkou, který je obsahem příslušného dokumentačního elementu, končí znakem konce řádky '`\n`'.

Pokud program nenalezne žádné hledané komentáře nebo komentáře nebudou obsahovat žádný text, program dokumentaci stejně **vytvoří**, ovšem s chybovým hlášením, že užitný obsah potřebný k vytvoření dokumentace nebyl ve zdrojových souborech nalezen.

## Specifikace výstupu programu

Výstup programu (potřebný pro interakci s uživatelem) bude směrován na konzoli. Pokud program neobdrží požadovaný počet argumentů na příkazovém řádku či pokud argumenty jsou chybné (neexistující soubory, nepovolené znaky v jejich názvech, apod.), měl by vypsat srozumitelné chybové hlášení v angličtině a **skončit nenulovým návratovým kódem**: Návratový kód `1` znamená nesprávný počet nebo tvar zápisu argumentů, kód `2` chybný argument (např. neexistující soubor nebo soubor, který nelze otevřít pro čtení či zápis), kód `3` pak znamená, že dokumentace byla vytvořena za okolností, kdy jeden nebo více dokumentačních komentářů ve zdrojovém kódu neodpovídalo požadovanému formátu jejich zápisu (jinými slovy byly zapsané špatně). Ostatním chybovým stavům můžete přiřadit hodnoty dle vlastního uvážení.

Výsledkem činnosti programu bude **zdrojový soubor programátorské dokumentace v makrojazyce TEXu**, uložený v kódování UTF-8 (abychom nemuseli řešit různé kódování akcentovaných českých znaků ve Windows a v Linuxu), přeložitelný typografickým systémem LaTeX do formátu PDF. Úprava dokumentu bude následující:

```
\section{Programátorská dokumentace}
\subsection{Modul \texttt{main.c}}
\subsubsection{Funkce \texttt{main(int argc, char *argv[])}}
\textbf{Argumenty:}
\verb"int argc" -- Počet parametrů, předávaných programu na příkazové řádce.
```

```
\verb"char *argv[]" -- Pole řetězců odpovídajících jednotlivým předaným parametrům.\\
\par\noindent
\textbf{Návratová hodnota:} \verb"int" -- Hodnota, předávaná operačnímu systému při
ukončení programu.\\
\par\noindent
\textbf{Popis:} Tato funkce představuje hlavní ...
```

Jednoduše řečeno, každé speciální dokumentační značce odpovídá jeden odstavec textu, začínající tučně vysázeným českým označením sémantické hodnoty (významu) příslušného bloku textu. Jde-li tedy o značku '`@author`', bude mít odpovídající LaTeXový příkaz tvar '`\textbf{Autor}:`' a následovat bude text za značkou do konce řádky. Všechny ostatní značky analogicky. Výslednou podobu části dokumentu ukazuje rámeček níže:

# 1 Programátorská dokumentace

## 1.1 Modul `main.c`

### 1.1.1 Funkce `main(int argc, char *argv[])`

**Argumenty:** `int argc` – Počet parametrů, předávaných programu na příkazové řádce. `char *argv[]` – Pole řetězců odpovídajících jednotlivým předaným parametrům.

**Návratová hodnota:** `int` – Hodnota, předávaná operačnímu systému při ukončení programu.

**Popis:** Tato funkce představuje hlavní ...

V případě, že se nebude jednat o dokumentační komentář funkce, ale např. struktury či proměnné (to lze ze zdrojového kódu velmi dobře poznat, protože gramatika jazyka ANSI C bezpodmínečně vyžaduje přítomnost znaků '`(`' a '`)`' v hlavičce každé funkce, i té, která nemá žádné argumenty), je třeba příslušným způsobem upravit uvozující text v příkazu `\subsubsection`.

# Užitečné techniky a odkazy

Níže uvedené techniky je možné (ale nikoliv nezbytně nutné) využít při řešení úlohy. Protože se jedná o postupy v informatice víceméně standardní, lze k nim nalézt velké množství dokumentace např. za pomoci vyhledávače Google:

1. lexikální analýza, např. nástroje **Lex/Yacc**
2. **JavaDoc** – `https://www.oracle.com/java/technologies/javase/javadoc-tool.html`
3. **Doxygen** – `https://www.doxygen.nl`
4. **spojový seznam**
5. **zásobník**
6. **cyklický graf**

**Řešení úlohy je zcela ve vaší kompetenci** – zvolte takové algoritmy a techniky, které podle vás nejlépe povedou k cíli.

# Příloha: Ukázka vstupního zdrojového kódu a odpovídajícího výstupu

Aplikace analyzuje tento vstupní soubor se zdrojovým kódem v jazyce ANSI C opatřený dokumentačními komentáři:

```
/**
 * @brief Výpis celé databáze.
 *
 * Tato funkce vypisuje do konzole obsah pole, které udržuje ukazatele
 * na všechny instance položek tvořících databázi.
 *
 * @param entry * data[] Pole ukazatelů na jednotlivé položky databáze.
 * @param int dsize Velikost pole udržujícího databázi.
 * @return int Indikátor úspěchu (0 = neúspěch, 1 = úspěch).
 *
 * @author \textcopyright{} Jan Naj
 * @version 1.1.3
 */
int dump(entry *data[], int dsize) {
  ...
}
```

Výsledkem činnosti aplikace je následující zdrojový kód dokumentace v LaTeXu:

```
\section{Programátorská dokumentace}
\subsection{Modul \texttt{db.c}}
\subsubsection{Funkce \texttt{int dump(entry *data[], int dsize)}}
\textbf{Argumenty:}
\verb"entry * data[]" -- Pole ukazatelů na jednotlivé položky databáze.
\verb"int dsize" -- Velikost pole udržujícího databázi.\\
\par\noindent
\textbf{Návratová hodnota:} \verb"int" -- Indikátor úspěchu (0 = neúspěch,
1 = úspěch).\\
\par\noindent
\textbf{Popis:} Tato funkce vypisuje do konzole obsah pole, které udržuje
ukazatele na všechny instance položek tvořících databázi.\\
\par\noindent
\textbf{Autor:} \textcopyright{} Jan Naj\\
\par\noindent
\textbf{Verze:} 1.1.3
```

Po přeložení LaTeXem vznikne takovýto dokument:

# 1 Programátorská dokumentace

## 1.1 Modul `db.c`

### 1.1.1 Funkce `int dump(entry *data[], int dsize)`

**Argumenty:** `entry * data[]` – Pole ukazatelů na jednotlivé položky databáze. `int dsize` – Velikost pole udržujícího databázi.

**Návratová hodnota:** `int` – Indikátor úspěchu (0 = neúspěch, 1 = úspěch).

**Popis:** Tato funkce vypisuje do konzole obsah pole, které udržuje ukazatele na všechny instance položek tvořících databázi.

**Autor:** © Jan Naj

**Verze:** 1.1.3