TTK4235 - Tilpassede datasystemer
Vår 2026

Norges teknisk-naturvitenskapelige universitet
Institutt for teknisk kybernetikk

Øving 5
Dokumentasjon med Doxygen

Revisjonshistorie

År	Forfatter
2020	Kolbjørn Austreng
2021	Kiet Tuan Hoang
2022	Kiet Tuan Hoang
2023	Kiet Tuan Hoang
2024	Terje Haugland Jacobsson Tord Natlandsmyr

I Introduksjon - Praktisk rundt filene

I denne øvingen får dere utlevert noen `.c` og `.h`-filer i `example` og `document_this`-mappene. Tabellen under lister opp alle filene som kommer med i `source`-mappene i de respektive hovedmappene samt litt informasjon om dere skal endre på filene eller om dere skal la dem bli i løpet av øvingen.

Filer	**Skal filen(e) endres?**
`example/source/main.c`	Nei
`example/source/memory_library.c`	Nei
`example/source/memory_library.h`	Nei
`document_this/source/main.c`	Nei
`document_this/source/calculations.c`	Nei
`document_this/source/calculations.h`	Ja
`document_this/source/system.c`	Nei
`document_this/source/system.h`	Jøss, ja!
`Main/*`	Nei
`.github/*`	Nei

II Introduksjon - Praktisk rundt øvingen

Dokumentasjon er viktig for videre utvikling av kode. Uten dokumentasjon blir det vanskelig for andre å forstå og bruke koden som har blitt utviklet. Dokumentasjon

er særlig viktig når man jobber i en stor gruppe, hvor man som oftest ikke har tid eller trenger å sette seg inn i koden for å kunne implementere og bruke den.

Ulempen med dokumentasjon er at det ikke er særlig gøy. Dermed, hvis kode og dokumentasjon er separate arbeidsoppgaver, vil dokumentasjonen gjerne glemmes litt bort.

Doxygen er en minste motstands vei - løsning på dette problemet. Med Doxygen, kan man med spesielt formaterte kommentarer generere dokumentasjon automatisk. Fordelene med å kombinere koding og dokumentasjon med Doxygen er at det da er mer sannsynlig at dokumentasjonen:

1. Skrives i utgangspunktet
2. Oppdateres hver gang koden endrer seg

Til å starte med skal dere få en grunnleggende innføring i bruk av Doxygen i seksjon III med `.c` og `.h`-filene i `example`-folderen. Deretter gjør dere oppgavene i seksjon 1 med `.c` og `.h`-filene i `document_this`-folderen for å få godkjent øvingen.

III Introduksjon - Innføring i Doxygen

Dette kapitlet gir en innføring i bruk av Doxygen med et eksempelprosjekt. Dere har fått utlevert en mappe som heter `example`. I mappen kan dere finne en `memory`-modul, som definerer enkle operasjoner på lister av heltall. Denne modulen er implementert i `memory_library.h` og `memory_library.c`. Prosjekttreet ser slik ut:

```
example
├── Makefile
└── source
    ├── main.c
    ├── memory_library.h
    └── memory_library.c
```

hvor `memory`-modulen består av to funksjoner: `memory_reverse_copy` og `memory_multiply_elements`:

- `memory_reverse_copy` tar inn to buffere, og kopierer fra det første inn i det andre, samtidig som rekkefølgen på elementene blir reversert.
- `memory_multiply_elements` tar inn et buffer som blir skalert med en skaleringsfaktor.

III .1 Konfigurasjonsfil

Før man kan bruke Doxygen, må man opprette en konfigurasjonsfil med kommandoen `doxygen -g doxconfig`. Dette er en fil som bestemmer ulike parametre og innstillinger for prosjektet. I dette eksemplet endrer man på disse parametrene:

```
PROJECT_NAME = "Memory Library Example"
OPTIMIZE_OUTPUT_FOR_C = YES
INPUT = source/
SOURCE_BROWSER = YES
```

Prosjekttreet ser nå slik ut:

```
example
├── Makefile
├── doxconfig
└── source
    ├── main.c
    ├── memory_library.h
    └── memory_library.c
```

III .2 Syntaks og kommentering av kode

Doxygen genererer dokumentasjon basert på spesielt formaterte kommentarer som består av kommandoer som definerer hvordan kommentaren skal tolkes. For å starte kommandoer i Doxygen bruker man `@`.

Det første Doxygen gjør er å se igjennom kodene etter en `@file`-kommando. Denne `@file`-kommandoen blir definert helt øverst i en fil (**Ingen `@file` øverst i koden resulterer i tom dokumentasjon!**) og melder til Doxygen om at det skal genereres dokumentasjon for denne filen:

```
/**
* @file
* @brief A simple library for doing operations on memory
* buffers consisting of integers
*/
```

I tillegg til `@file`, har man også `@brief` som kort gir en oppsummering av filens funksjon. Doxygen støtter også \ istedenfor `@` for å definere kommandoer. For `C` og `C++` anbefales `@` fordi tegnet ikke kolliderer med *escape characters* (som `\n` eller `\t`) i koden, og det blir enklere å søke etter Doxygen-kommentarer senere.

III .2.1 `int memory_reverse_copy(...)`

Til og starte med, dokumenterer vi funksjonen `int memory_reverse_copy`. For å dokumentere denne funksjonen, skriver man følgende kommentar inn rett over `memory_reverse_copy` i `.h`-filen:

```
/**
* @brief Copy a list of integers from one buffer to another,
* reversing the order of the output in the process.
*
* @param[in] p_from Source buffer.
* @param[out] p_to Destination buffer.
* @param[in] size Number of integers in the buffer.
*
* @return 0 on success, 1 if either @p p_from or @p p_to
* is a @c NULL pointer.
*/
```

I tillegg til `@brief` som gir en kort oppsummering av hva funksjonen gjør har man `@param` som forteller hva en funksjonsparameter sin oppgave er. Det er også vanlig å legge ved `[in]`, `[out]`, og `[in,out]`. Dette er valgfritt og forteller om parameteren blir modifisert av funksjonen eller ikke. Å bruke `[in]`, `[out]`, og `[in,out]` er spesielt nyttig når man bruker pekere som argument i `C/C++`. Følgende konvensjon brukes:

- `[in]`: *Input* - parameter som leses og brukes i funksjonen, men som ikke får endret verdi (dvs. blir skrevet til) i funksjonen.
- `[out]`: *Output* - parameter som funksjonen kan endre verdien på men ikke leser og bruker direkte. Dette brukes bare for parametere hvis verdi forblir endret på utsiden av funksjonen etter at denne har returnert (dvs. som regel pekere).
- `[in,out]`: *Input* og *Output* - parameteren som lese og brukes direkte i funksjonen, og som kan være endret på utsiden av funksjonen når den returnerer.

Kommandoen `@return` dokumenterer hva funksjonen returnerer. Denne trengs ikke om man har en `void` - funksjon.

For å legge inn ekstra informasjon i koden bruker man `@c` og `@p`. `@c` er et hint til Doxygen om at det neste ordet er et kodeord. I tilfellet over betyr det at `NULL` blir markert som kode i dokumentasjonen. Likeså, er `@p` et hint til Doxygen om at det neste ordet er en parameter.

Eventuelle kommandoer som Doxygen støtter kan fås ved å ta en titt på
https://www.doxygen.nl/manual/commands.html#cmdp

III .2.2 `void memory_multiply_elements(...)`

For å dokumentere denne funksjonen, skriver man følgenede kommentar inn rett over `memory_multiply_elements` i `.h`-filen:

```
/**
* @brief Multiply all the elements in @p p_buffer, of size
* @p size with the supplied @p factor.
*
* @param[in,out] p_buffer Buffer of integers to be multiplied
* with @p factor.
*
* @param[in] factor Factor to multiply each of the
* elements in @p p_buffer with.
*
* @param[in] size Size of @p p_buffer.
*
* @warning If @p p_buffer is @c NULL, the function will
* abruptly terminate the program with exit code @c 1.
*/
```

I tillegg til de overnevnte kommandoene, inkluderer man for denne funksjonen en `@warning` som brukes for å advare brukeren mot oppførsel som ikke er åpenbar.

III .3 Automatisk generering av dokumentasjon

Dokumentasjon blir automatisk generert ved at man kaller `doxygen doxconfig` i samme mappe som konfigurasjonsfilen. `doxygen doxconfig` generer dokumentasjon både i HTML- og LaTeX-format, i hver sin mappe. Prosjekttreet ser nå slik ut:

```
memory
├── Makefile
├── doxconfig
├── source
│   ├── main.c
│   ├── memory_library.h
│   └── memory_library.c
├── html
│   ├── index.html
│   └── [...]
└── latex
    ├── index.tex
    └── [...]
```

`GENERATE_LATEX` og `GENERATE_HTML` definert i konfigurasjonsfiguren bestemmer hvilke format Doxygen skal generere dokumentasjon på. I tillegg kan man også spesifisere hvor Doxygen lagrer den genererte dokumentasjonen ved å endre på parameteren `OUTPUT_DIRECTORY`.

Dersom man nå åpner `html/index.html` i en egnet nettleser ved å enten kalle `nettleser html/index.html` fra terminalen, eller trykke `Ctrl + O` inne i nettleseren og åpne `index.html` derfra blir man møtt av en ganske tom side (se figur 1). Dette er et resultat av at `"related configuration options"` i konfigurasjonsfilen har blitt ignorert i denne øvingen. Opp til venstre hjørne er det en lenke til `Files`, som kan bli fulgt for å få en oversikt som illustrert i figur 2 (se også figur 3, 4, og 5 for dokumentasjon som genereres av doxygen for `memory_library.h`, `int memory_reverse_copy`, og `void memory_multiply_elements`).

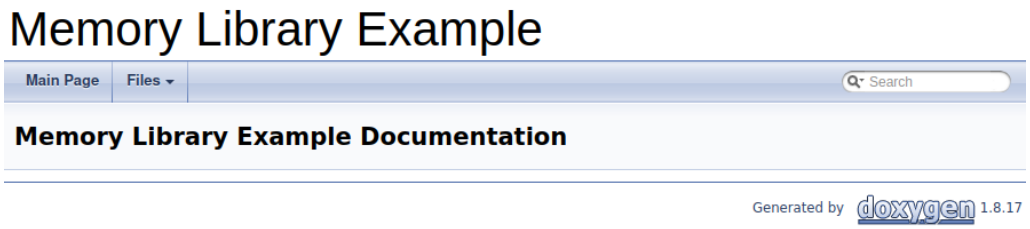


Figure 1: Prosjektfilens hovedside.

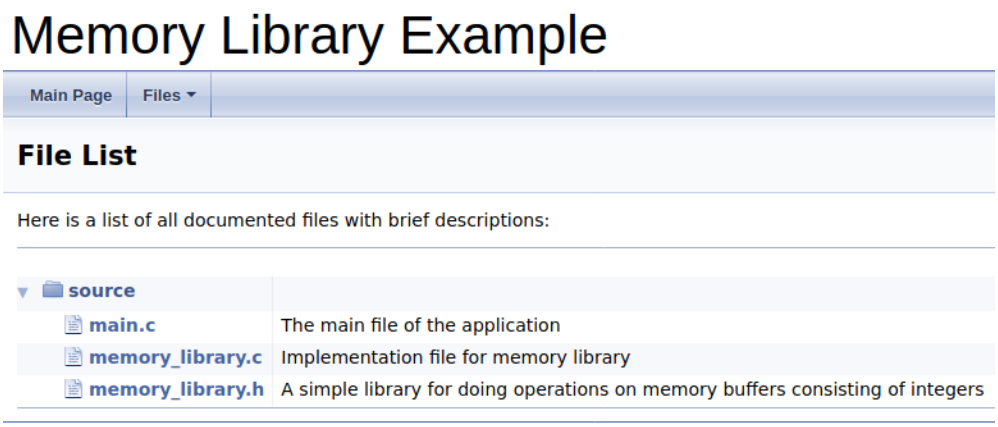


Figure 2: Oversikt over hvilke filer prosjektet består av.

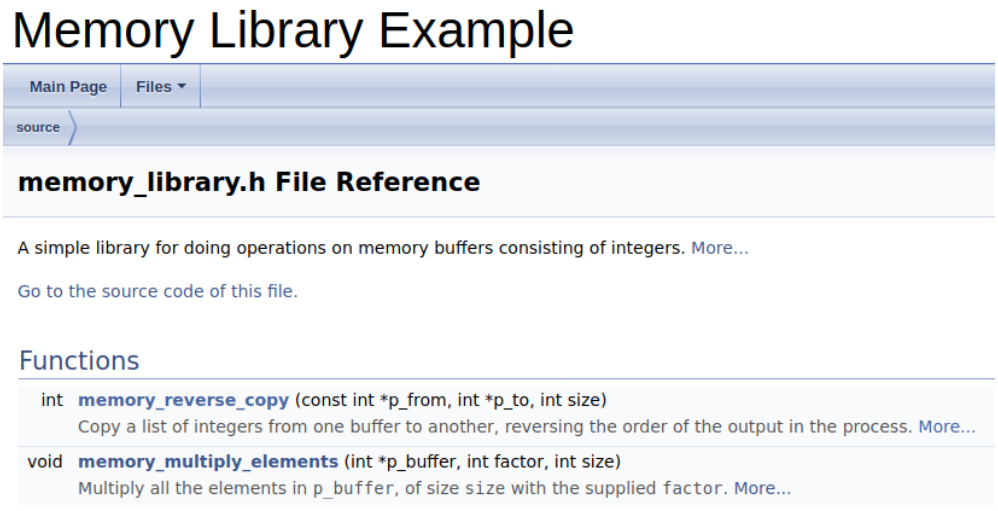
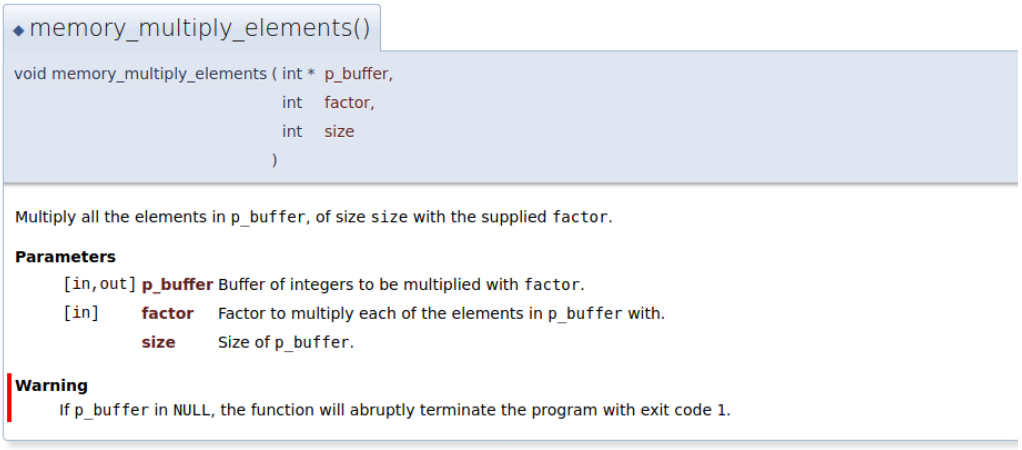


Figure 3: Dokumentasjon for `memory_library.h`



◆ memory_multiply_elements()

```
void memory_multiply_elements ( int *  p_buffer,
                                int     factor,
                                int     size
                              )
```

Multiply all the elements in `p_buffer`, of size `size` with the supplied `factor`.

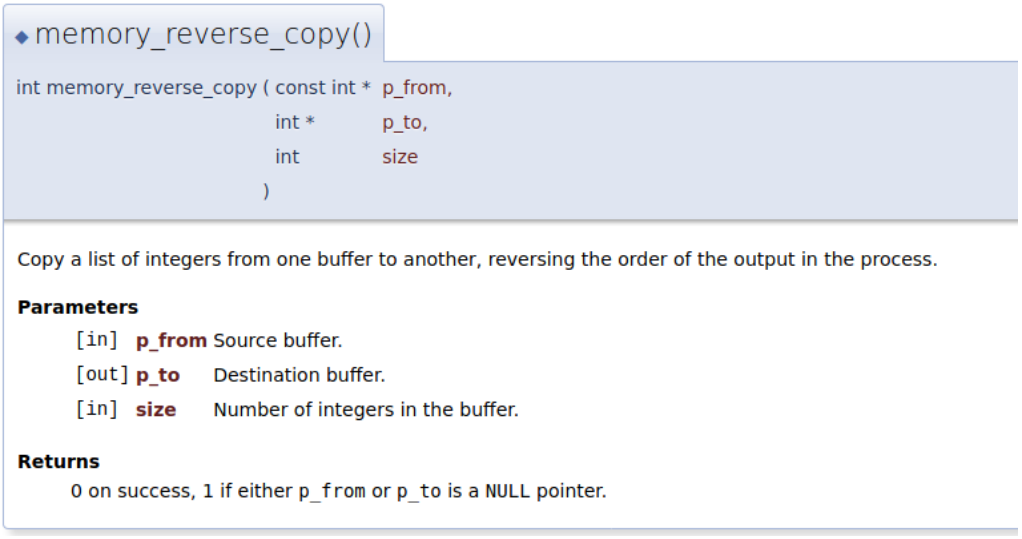
Parameters

`[in,out]`	**p_buffer**	Buffer of integers to be multiplied with `factor`.
`[in]`	**factor**	Factor to multiply each of the elements in `p_buffer` with.
	size	Size of `p_buffer`.

Warning

If `p_buffer` in `NULL`, the function will abruptly terminate the program with exit code 1.

Figure 4: Dokumentasjon for `int memory_reverse_copy`.



◆ memory_reverse_copy()

```
int memory_reverse_copy ( const int *  p_from,
                          int *        p_to,
                          int          size
                        )
```

Copy a list of integers from one buffer to another, reversing the order of the output in the process.

Parameters

`[in]`	**p_from**	Source buffer.
`[out]`	**p_to**	Destination buffer.
`[in]`	**size**	Number of integers in the buffer.

Returns

0 on success, 1 if either `p_from` or `p_to` is a `NULL` pointer.

Figure 5: Dokumentasjon for `void memory_multiply_elements`.

1 Oppgave (100%) - Grunnleggende Doxygen

Dere har fått utlevert en mappe som heter `document_this`. `document_this` består av to biblioteker: `system.h` og `calculations.h`. Deres oppgave er å:

a Opprette en konfigurasjonsfil for Doxygen.

b Få en oversikt over implementasjonsfilene (`.c`). Det kan være lurt på dette stadiet å lage en simplifisert Makefile for å se hva programmet faktisk gjør.

c Dokumenter headerfilene (`.h`).

d Bruke Doxygen for å generere HTML-dokumentasjon.

Når dere er ferdige, viser dere en nettside (`.html`) med dokumentasjon til en studass for godkjenning.