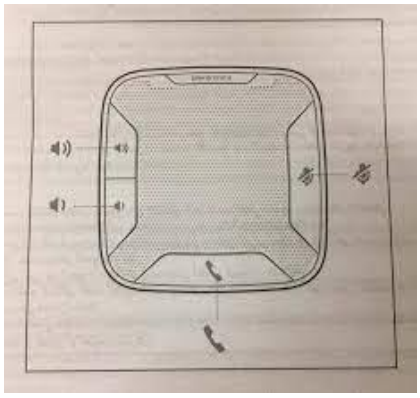
Модуль 3. Основи документування Лабораторна робота.

№8.1. Вступ до документації коду.

Как не стоит документировать код:



Doxygen документація:

Установка и настройка

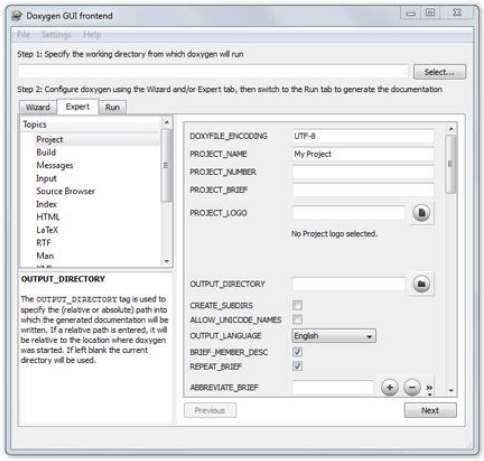
Скачать останню версію Doxygen можно на ліцензійном сайті, дистрибутива якої доступні для большості популярних операційних систем, крім того, вы можете користуватися вашим пакетним менеджером. для комфортной и полнофункциональной работы рекомендуется установить Graphviz.

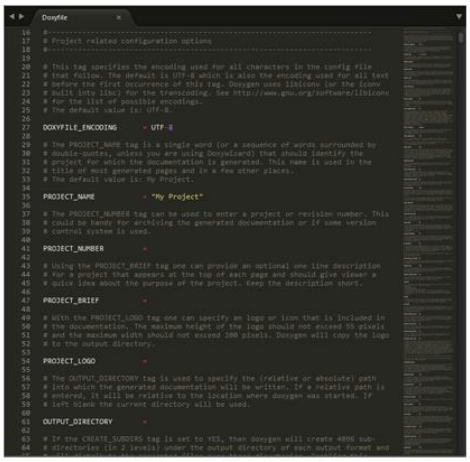
Далее работа с Doxygen весьма тривиальна: достаточно запустить программу, указав ей путь к файлу с настройками.

doxygen <config_file>

Но в этом файле и вся тонкость. Дело в том, что каждому проекту соответствует свой файл настроек, в котором может быть прописан путь до исходников проекта, путь, по которому должна быть создана документация, а также большое число других разнообразных опций, которые подробно описаны в документации, и которые позволяют максимально настроить документацию проекта под свои нужды.

В принципе, для редактирования данного файла и, вообще, работой с Doxygen, можно воспользоваться программой Doxywizard, которая чаще всего идёт вместе с Doxygen и которая позволяет чуть удобнее работать с файлом настроек (слева – Doxywizard; справа – файл открытый в текстовом редакторе):





Итак, приступим к созданию файла с настройками. Вообще, если вы используете Doxywizard, то он будет создан автоматически, в противном случае для создания этого файла необходимо запустить программу Doxygen с ключом *-g* (от generate):

```
doxygen -g <config_name>
```

Рассмотрим основные опции, которые могут вам пригодится, чтобы создать первую вашу документацию:

Тэг	**Назначение**	По умолчанию
DOXYFILE_ENCODING	Кодировка, которая используется для всех символов в данном файле настроек	UTF-8
OUTPUT_LANGUAGE	Устанавливает язык, на котором будет сгенерирована документация	English
PROJECT_NAME	Название проекта, которое может представлять собой единое слово или последовательность слов (если вы редактируете вне Doxywizard, последовательность слов необходимо поместить в двойные кавычки)	My Project
PROJECT_NUMBER	Данный тэг может быть использован для указания	—

	номера проекта или его версии	
PROJECT_BRIEF	Краткое однострочное описание проекта, которое размещается сверху каждой страницы и даёт общее представление о назначении проекта	—
OUTPUT_DIRECTORY	Абсолютный или относительный путь, по которому будет сгенерирована документация	Текущая директория
INPUT	Список файлов и/или директорий, разделенных пробелом, которые содержат в себе исходные коды проекта	Текущая директория
RECURSIVE	Используется в том случае, если необходимо сканировать исходные коды в подпапках указанных директорий	NO

После того, как мы внесли необходимые изменения в файл с настройками (например, изменили язык, названия проекта и т.п.) необходимо сгенерировать документацию.

Для её генерации можно воспользоваться Doxywizard (для этого необходимо указать рабочую директорию, из которой будут браться исходные коды, перейти на вкладку «Run» и нажать «Run doxygen») или запустив программу Doxygen, указав ей в качестве параметра путь к файлу с настройками:

```
doxygen <config_file>
```

Основи документації Doxygen

Теперь, когда мы разобрались с тем, как настраивать Doxygen и работать с ним, впору разобраться с тем, как необходимо документировать код, основными принципами и подходами.

Документация кода в Doxygen осуществляется при помощи документирующего блока. При этом существует два подхода к его размещению:

1. Он может быть размещён перед или после объявления или определения класса, члена класса, функции, пространства имён и т.д.;
2. Либо его можно располагать в произвольном месте (и даже другом файле), но для этого потребуется явно указать в нём, к какому

элементу кода он относится. Мы не будет рассматривать этот подход, поскольку даже разработчики рекомендуют его избегать, но если интересно, то подробнее о нём можно прочитать в документации.

Структурно, любой документирующий блок является комментарием, просто оформленным специальным образом, поэтому естественно, что его вид зависит от используемого языка (подробнее об этом можно прочитать в соответствующем разделе документации). Поэтому далее мы остановимся на рассмотрении синтаксиса для C-подобных языков (C/C++/C#/Objective-C/PHP/Java).

Сразу отметим, что, вообще, всего существует два основных типа документирующих блоков: многострочный блок и однострочный блок.

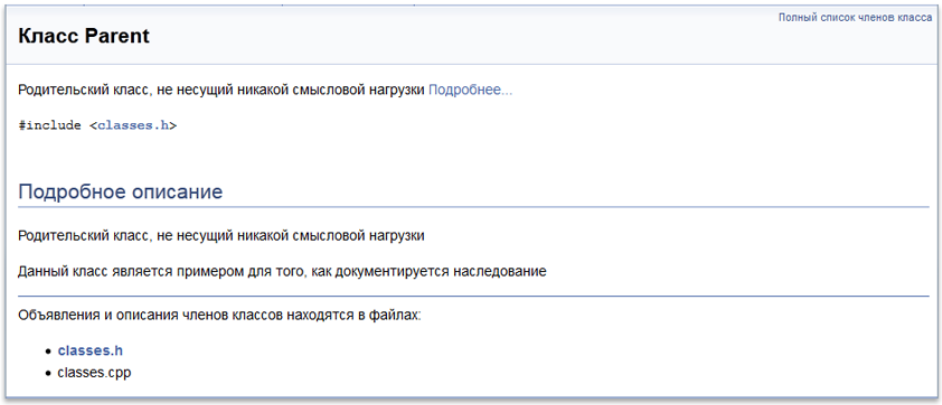
Разница между ними чуть более сильная, чем между однострочным и многострочным комментарием. Дело в том, что текст, написанный в однострочном блоке относится к краткому описанию документируемого элемента (сродни заголовку), а текст, написанный в многострочном блоке относится к подробному описанию. Про эту разницу не следует забывать.

Пример документации

Теперь рассмотрим то, как это будет выглядеть на практике. Ниже представлен документированный код некоторого класса в соответствии с теми правилами, которые мы рассматривали ранее.

```
 /*!

\brief Родительский класс, не несущий никакой смысловой нагрузкиДанный класс имеет только одну простую цель: проиллюстрировать то,как Doxygen документирует наследование */

class Parent

{public:Parent();

~Parent();

};
```

В итоге Doxygen сформирует на основе данных комментариев следующую красиво оформленную страничку (здесь приведена вырезка из неё):



Теперь, когда мы научились основам, пришла пора познакомиться с тем, как можно детализировать документацию. Инструментом для этого являются команды.

Команды

С несколькими из команд в Doxygen мы успели познакомиться (речь идёт о *\brief* и *\details*), однако на самом деле их значительно больше. Полный их список приведён в официальной документации.

Вообще, любая команда в Doxygen представляет собой слово на английском языке предваренное символом "\" или "@" (обе записи тождественны) и таких команд очень много, порядка двухсот. Приведём для примера несколько таких команд:

Команда	**Значение**
\authors	Указывает автора или авторов
\version	Используется для указания версии
\date	Предназначена для указания даты разработки
\bug	Перечисление известных ошибок
\warning	Предупреждение для использования
\copyright	Используемая лицензия
\example	Команда, добавляемая в комментарий для указания ссылки на исходник с примером (добавляется после команды)
\todo	Команда, используется для описания тех изменений, которые необходимо будет сделать (TODO).

Стандарты оформления кода

https://www.kernel.org/doc/html/v4.10/process/coding-style.html

`clang-format -i` - ваш друг

Constants - ALL_CAPICAL_CASE: MAX_ITERATIONS, PI

Имена всех переменных, функций, структур – snake_case: student_name

- Все аббревиатуры – в нижнем регистре: export_html, read_dvd

Имена всех множеств записываются в множественном числе:

- int grades[];
- ?? students[];

Названия всех идентификаторов однозначно определять их назначение (~~idx~~, index, numerator, denominator, matrix)

- Исключением являются счетчики циклов – для них могут использоваться следующие имена: i,j,k,m,n

Слова *get* та *set* должны быть использованы везде, где происходит прямой доступ к атрибуту:

- *get_matrix_element(matrix, 2, 4)*;
- *set_matrix_element(matrix, 2, 4, value)*;

тандарты оформления кода. Условные операторы

```
if (condition) {
    //...
} else if (condition) {
    //...
} else {
    //...
}
```

```
switch (condition) {
case 1:
    //...
    // відсутній "break"
case 2:
    //...
break;
case 3:
    //...
break;
default:
    //...
break;
}
```

Складних умовних виразів слід уникати. Краще замість цього використовувати булеві змінні:

```
bool is_finished = (element_no < 0) || (element_no > max_element);
bool is_repeated_entry = element_no == last_element;
if (is_finished || is_repeatedentry) {
//...
}
```