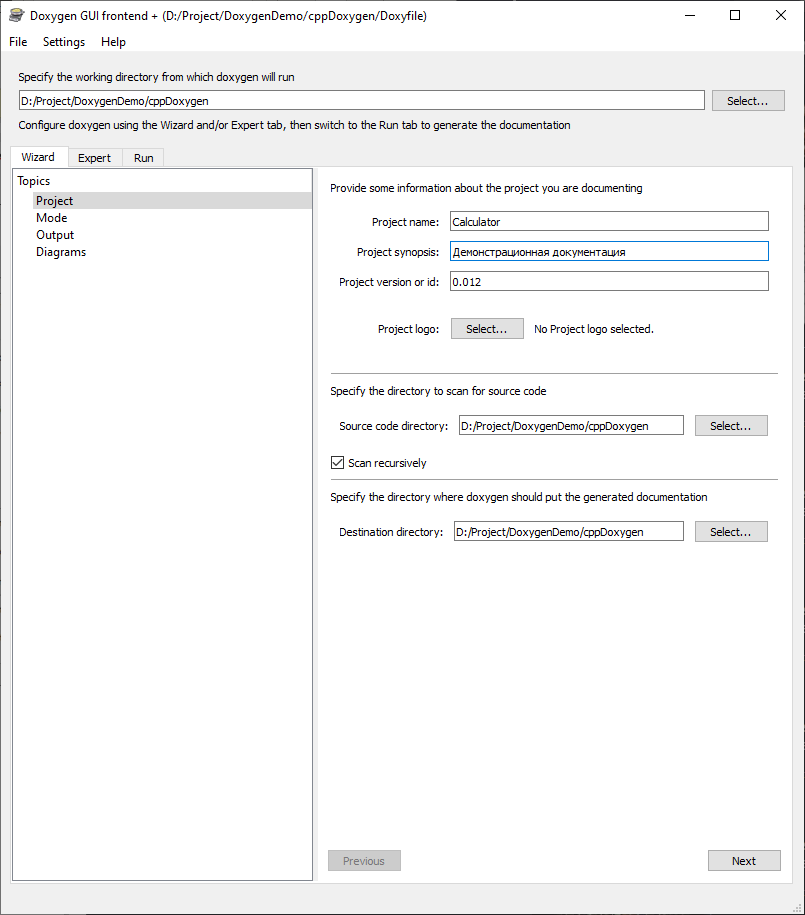
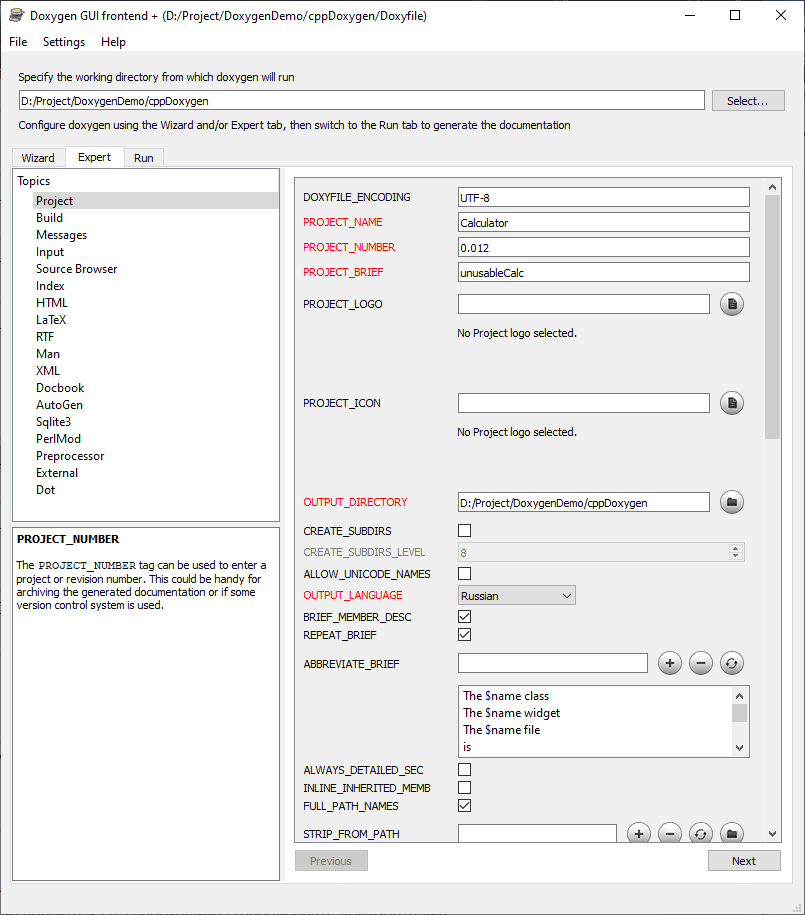
1. Для сборки файла с документацией необходимо установить ПО doxygen с их официального сайта: [тыц](https://doxygen.nl/index.html). Есть удобный GUI Doxywizard (для диаграмм понадобится также установить Graphviz). Хорошее описание работы с ним представлено [в этой серии статей](https://habr.com/ru/articles/252101/), а полный перечень команд можно найти по следующей [ссылке](https://doxygen.nl/manual/commands.html#cmdreturns)
2. Перед тем, как собирать проект, нужно создать его конфигурационный файл (гуишка сверху позволяет ускорить процесс и выглядит удобнее)



Внешний вид Doxywizard (краткое описание конфига для проекта)



Внешний вид Doxywizard (расширенное описание)

1. Предпочтительный формат документов – html для возможности просмотра документации на любых носителях + предоставляет большие возможности для кастомизации документа.
2. Ключевые опции доксигена для конфига:
   1. DOXYFILE\_ENCODING – кодировка, которая используется для всех символов в файле настроек;
   2. OUTPUT\_LANGUAGE – язык, на котором будет сгенерирована документация
   3. PROJECT\_NAME – название проекта.
   4. PROJECT\_NUMBER – номер проекта и его версия
   5. PROJECT\_BRIEF – краткое описание проекта, размещающееся сверху
   6. OUTPUT\_DIRECTORY – путь, по которому будет генерироваться документация
   7. INPUT – список файлов (разделитель – пробел), которые содержат в себе исходники проекта.
   8. RECURSIVE – флаг выставляется в случае необходимости сканировать исходные коды в подпапках указанных директорий
   9. INPUT\_ENCODING – для корректного отображения комментариев на русском необходимо внести в конфиг CP1251
3. Стили написания doxygen-блоков:
   1. JavaDoc стиль:

/\*\*

\* ... первая строчка ...

\* ... вторая строчка ...

\*/

или

/\*\*

... первая строчка ...

... вторая строчка ...

\*/

* 1. Qt стиль:

/\*!

\* ... первая строчка ...

\* ... вторая строчка ...

\*/

6. Хороший пример оформления документации можно увидеть у [KDE](https://api.kde.org/kcgroups/html/classes.html) . Это довольно близко к тому, что автоматически генерирует доксиген, просто накатили свой стиль.

Перечень основных тегов для документирования:

**Работа со страницами:**

- \mainpage – указание главной страницы документации;

- \page – именованная страница документации;

*Примечание:* для оформления документации используется язык разметки markdown.

**Работа с файлами:**

- \file – указание на отношение блока к файлу. Пример:

/\*!

\file calculator.h

\*/

- \author \authors – указание автора. Пример:

/\*!

\ author Кочергин О.Б.

\*/

- \bug – сообщение об ошибке. Создает список ошибок. Пример:

/\*!

\bug При запуске ПО все вокруг взрывается

\*/

- \version – версия ПО, класса или функции

- \date – дата создания класса, функции, ect

- \warning – предупреждение для использования

**Работа с функциями, модулями, ect:**

- \brief – краткое многострочное описание элемента. Пример:

/\*!

\brief Краткое описание и

его продолжение.

Подробное описание

\*/

- /// //! – краткое однострочное описание элемента. Пишется НАД элементом.

- ///< – краткое однострочное описание элемента. Пишется ПОСЛЕ элемента.

- \details – подробное описание. Вместо него можно сделать отступ. Пример:

/\*!

\brief Краткое описание

\details Подробное описание

\*/

- \param – описание параметров функции. Имеет следующий вид:

\param [<направление>] <имя\_параметра> {описание\_параметра}

Направление – это опциональный атрибут, который показывает назначение параметра и может иметь три значения "[in]", "[out]", "[in,out]";

Имя параметра – это имя, под которым данный параметр известен в документируемом коде;

Описание параметра представляет собой простое текстовое описание используемого параметра.

- \return – описание возвращаемого значения.

- \throw – описание исключения, возникающего при ошибке. Пример:

\throw std::bad\_alloc В случае возникновения ошибки при выделении памяти

- \example – пример того, как использовать элемент ПО. Применяется, когда примеры находятся в отдельном файле (в настройках необходимо указать путь к папке с примерами: EXAMPLE\_PATH = путь\_к\_директории). Будет создан раздел «Примеры», в котором добавятся ссылки на задокументированные функции. Пример:

\example <имя\_файла>

- \code и \endcode – применяются для добавления кода в документацию. Пример:

/\*!

\brief Алгоритм Евклида

\param a,b Два числа, чей наибольший делитель мы хотим найти

Данная функция реализует алгоритм Евклида, при помощи которого

находится наибольшее общее кратное у двух чисел.

Код функции выглядит следующим образом:

\code

int gcd(int a, int b) {

int r;

while (b) {

r = a % b;

a = b;

b = r;

}

return r;

}

\endcode

\*/

int gcd(int a, int b);

В Doxygen поддерживается автоматическая вставка тел документируемых элементов. Для этого в настройках необходимо выставить флаг:

INLINE\_SOURCES = YES

- \todo – команда для описания тех изменений, которые необходимо будет внести. Пример:

/\*\*

\* \brief Сравнение чисел

\* \todo Реализовать функцию сранения чисел

\* \returns большее число, если mode = MORE; меньшее число, если mode = LESS

\*/

int diff(diffMode\_t mode);

**Работа с группами:**

**-** \defgroup – Создание модуля, содержащего в себе элементы ПО, которые можно объединить по какому-либо принципу (как пример – сделать группу из родительского класса и дочерних).

- \addtogroup – добавление элемента в группу. Пример:

/\*\*

\* \defgroup someGroup Отдельная группа

\* \brief Описание отдельной группы

\*/

/\*\*

\* \addtogroup someGroup Comparator

\* @{

\*/

Какой-то код

///@}

- Создание подгруппы осуществляется следующим образом:

/\*! \defgroup main\_module Главный модуль \*/

/\*! \defgroup second\_module Вложенный модуль

\ingroup main\_module

\*/