



How To Document C Source Code Using Doxygen For Beginners

Полное руководство по использованию Doxygen для документирования исходного кода С . . ! !

17 декабря 2019 г.

Как мы знаем, Doxygen - очень мощный инструмент для создания документации, но он больше ориентирован на объектно-ориентированные языки, такие как C ++ и Java. Мастер Doxy, он же графический интерфейс Doxygen, имеет множество опций, с которыми можно поиграть, и в этой статье я покажу вам, как оптимизировать настройки, чтобы использовать все возможности Doxygen для документирования исходного кода, написанного на С.

Я написал эту статью, имея в виду новичков. Сначала давайте посмотрим, как работает Doxygen, затем посмотрим, как писать комментарии, совместимые с Doxygen, а затем перейдем к тому, как установить Doxygen на свой компьютер и как настроить параметры для создания оптимизированной документации для кода С.

Если вы уже знакомы с некоторыми разделами, смело пропускайте их и переходите к интересующему разделу, используя приведенное ниже оглавление.

Итак, давайте начнем!

Содержание

- 1 Теги и символы Doxygen для использования в ваших комментариях
- 2 страницы Doxygen
- 3 Установка Doxygen
- 4 Графический интерфейс
- 5 Статей по теме

Как работает Doxygen? В Doxygen есть несколько специальных синтаксисов, которые вы можете использовать в своих комментариях к коду, и когда вы запустите этот инструмент, он проанализирует детали документации из комментариев, которые следуют специальному синтаксису Doxygen.

Теги и символы Doxygen для использования в ваших комментариях

Doxygen поддерживает несколько методов включения документации в комментарии. 2 самых простых метода для включения в исходный код C:

```
/**  
... комментарии...  
*/  
  
И  
  
/// Комментарии.
```

Обратите внимание на дополнительную звездочку (*) в строке 1 первого варианта и дополнительный символ косой черты (/), т.е. 3 вместо 2 косых черт. Эти синтаксические схемы используются, чтобы сообщить анализатору Doxygen, что это документация и ее необходимо извлечь.

Приведенные выше 2 синтаксических элемента должны быть размещены чуть выше *объекта*, чтобы анализатор Doxygen мог связать комментарии с этим конкретным *объектом*.

Что такое сущность в Doxygen? Сущность может быть глобальной переменной, объявлением структуры, перечислением или функцией.

Рассмотрим фрагмент кода ниже

```
1 #include <stdio.h>
2
3 /// This is a test string
4 char * hello = "Hello Embedded Inventors!";
5
6 /**
7  * This is the main function
8  *
9 */
10 int main()
11 {
12     printf("%s \n", hello);
13     return 0;
14 }
```

Здесь я использовал синтаксис с тремя косыми чертами для документирования глобальной переменной и синтаксис “`/**...*/`” для документирования функции. Ввод этого кода через Doxygen дает мне документацию, которая выглядит следующим образом.

```
#include <stdio.h>
```

Functions

```
int main()
```

Variables

```
char * hello = "Hello Embedded Inventors!"
```

This is a test string.

Function Documentation

Documented comments

◆ main()

```
int main( )
```

This is the main function.

Вам может показаться, что эти комментарии выглядят слишком простыми, поэтому давайте рассмотрим более сложный пример, чтобы увидеть всю мощь Doxygen. Взгляните на следующий фрагмент кода.

```

1  /// @file main.c
2  #include<stdio.h>
3
4  #define PI      3.1415    ///< Mathematical constant PI.
5  #define RADIUS_M 7.82     ///< Radius in meters
6
7 /**
8  * Calculates the Area of the circle.
9  * Formula: Area = PI*r^2
10 * @param[in] radius
11 * @param[out] area
12 */
13 float calculate_area(float radius)
14 {
15     float area;
16     area = PI * radius * radius;
17     return area;
18 }
19
20 /**
21 * Calculates the Perimeter of the circle.
22 * Formula: Perimeter = 2*PI*r
23 * @param[in] radius
24 * @param[out] perimeter

```

```

25 */
26 float calculate_perimeter(float radius)
27 {
28     float perimeter;
29     perimeter = 2 * PI * radius;
30     return perimeter;
31 }
32
33 /**
34 * Main entry point of the program.
35 */
36 int main()
37 {
38     float radius, area, perimeter;
39     radius = RADIUS_M;
40     area = calculate_area(radius);
41     perimeter = calculate_perimeter(radius);
42     printf("Area = %.2f sq.m \n", area);
43     printf("perimeter = %.2f m\n", perimeter);
44     return 0;
45 }
```

Это простая программа для вычисления площади и периметра с использованием радиуса. Теперь взгляните на документацию, созданную с помощью этого кода.

Macros

#define PI 3.1415	Mathematical constant PI.
#define RADIUS_M 7.82	Radius in meters.

В С считается хорошей практикой преобразовывать литеральные значения в макросы для улучшения читаемости, а некоторые комментарии, объясняющие значение, сделают это еще лучше.

Документация по макросам выглядит как на картинке выше. Если вы посмотрите на код, то, возможно, заметили, что вместо того, чтобы оставлять комментарии поверх объекта, я сохранил их в правой части определения макроса.

Также вместо “///”

Я использовал “///<”

Это еще один синтаксис, обычно используемый в Doxygen, если мы хотим иметь документацию в правой части объекта. При написании кода на С мы часто сталкиваемся с необходимостью размещать документацию справа. Это особенно верно для макросов и документации по элементам структуры.

Далее давайте рассмотрим часть документации, посвященную списку функций.

Functions

`float calculate_area (float radius)`

Calculates the Area of the circle. More...

`float calculate_perimeter (float radius)`

Calculates the Perimeter of the circle. More...

`int main ()`

Main entry point of the program.

Здесь, как вы можете видеть, первая строка комментариев размещена под названием функции в списке, а за ней следует “*Подробнее ...*” Ссылка.

Если вы нажмете на “*Подробнее*”, вы перейдете к подробному описанию функции, как вы можете видеть на следующем рисунке.

◆ `calculate_area()`

`float calculate_area (float radius)`

Calculates the Area of the circle.

Formula: $\text{Area} = \pi r^2$

Parameters

[in] `radius`

[out] `area`

Здесь вы можете увидеть как первую, так и вторую строку, содержащую используемую формулу.

В Doxygen существует 2 уровня описания сущности

- Краткое описание и
- Подробное описание

Настроив настройки Doxygen (я точно объяснил, как это сделать в следующих разделах), вы можете настроить анализатор Doxygen так, чтобы первое предложение комментария считалось кратким описанием, а остальные предложения - подробным описанием. Doxygen по умолчанию использует это краткое описание и помещает его в список функций, как показано выше, для удобства использования, и сохраняет подробное описание для раздела "Подробнее" документации.

Далее давайте посмотрим на документацию параметров. Здесь используется другой специальный синтаксис Doxygen, который является синтаксисом **@**.

Doxygen называет их **структурными командами**. Их несколько, и наиболее полезными для программистов С являются следующие.

<code>@file</code>	Имя файла должно присутствовать в заголовке файла для включения в процесс создания документации
<code>@param</code>	Документация по параметрам для функций
<code>@страница</code>	Название страницы Markdown
<code>@mainpage</code>	Главная страница уценки проекта
<code>@tableofcontents</code>	Создает "оглавление" для страницы markdown

Синтаксис, используемый в описании параметра, следующий

`@param[ввод / вывод] <имя переменной> <описание переменной>`

Поскольку имя переменной `radius` очевидно для читателя кода, я не добавлял описание. Пожалуйста, взгляните на мою другую статью, когда использовать, а когда избегать комментариев? рекомендации по комментированию.

Синтаксис, используемый в описании параметра, следующий

`@param[ввод / вывод] <имя переменной> <описание переменной>`

Поскольку имя переменной `radius` очевидно для читателя кода, я не добавлял описание. Пожалуйста, взгляните на мою другую статью, когда использовать, а когда избегать комментариев? рекомендации по комментированию.

Страницы Doxygen

Помимо синтаксиса и тегов, упомянутых выше, Doxygen также может использовать файлы markdown для создания HTML-веб-страниц как части вашей документации. Если вам нужно предоставить фрагменты кода, стандарты кодирования, информацию об архитектуре и т.д. В вашей документации, то лучший способ сделать это - через pages или файлы markdown.

Если вы не понимаете, что такое файлы markdown, я рекомендую вам посмотреть это короткое видео на YouTube (ссылка) ([youtube.com/watch?v=eJojC3lSkwg](https://www.youtube.com/watch?v=eJojC3lSkwg))

Doxygen классифицирует наши файлы markdown как 3 иерархические страницы

- Главная страница
- Страница и
- Подстраница

И на каждой странице также могут быть разделы и подразделы.

Главная страница

Давайте сначала посмотрим на пример главной страницы, а затем мы сможем рассмотреть теоретическую сторону вещей. Вот как должна быть создана главная страница проекта в файле markdown.

```
1 | @mainpage Project title
2 | This is the main page for the project.
3 |
4 | 1. Describe the project in general and the components/ modules
5 | 2. Explain each module using data flow and software architecture
6 | 3. Place links for each of the sub module's markdown pages
```

После запуска через Doxygen конечный HTML-файл будет выглядеть следующим образом

Project title

This is the main page for the project.

1. Describe the project in general and the components/ modules it has
2. Explain each module using data flow and software architecture diagrams
3. Place links for each of the sub module's markdown pages

Это главная страница проекта, на этой странице вы можете ввести такие детали, как требования, примечания к выпуску, общую архитектуру и т.д. Структурная команда Doxygen для использования - это “*@mainpage*”, как показано в примере выше. Этот тег в одном из наших файлов markdown сообщит анализатору Doxygen, что данный файл markdown является главной страницей проекта. Эта страница отображается при нажатии `index.html` из папки HTML, созданной Doxygen.

Страница

Это обычные страницы. Они будут отображаться на первом уровне страниц в левой части древовидного представления. В более позднем разделе этой статьи я показал вам, как включить “TreeView”, который, на мой взгляд, лучше, чем просмотр документации C по умолчанию.

Структурная команда, используемая для страницы Doxygen, - это `@page`, как показано в файле markdown ниже

```

1 | @page module_name
2 | **Author:** Author name
3 | **Date:** Date
4 | ## Module's Role
5 | Explain the module's role in the system in general

```

После запуска через Doxygen конечный результат будет выглядеть следующим образом

My Project
Project title
module_name
▶ Data Structures
▶ Files

Author: Author name**Date:** Date

Module's Role

Explain the module's role in the system in general

Subpage

Эти страницы не будут находиться на первом уровне, скорее они будут размещены под другой страницей.

Структурная команда, используемая для подстраницы Doxygen, - это @subpage . Эта @subpage должна использоваться на странице модуля, для которой данная страница является подстраницей, чтобы это выглядело так.

```

1 | @page module_name Module Name
2 | @subpage sub_page_name
3 | **Author:** Author name
4 | **Date:** Date
5 | ## Module's Role
6 | Explain the module's role in the system in general

```

Файл markdown для подстраницы будет выглядеть следующим образом

```

1 | @page sub_page_name Sub module Name
2 | some text here explaining the role of the sub module

```

My Project
Project title
Module Name
Sub module Name
▶ Data Structures
▶ Files

Module Name

Sub module Name**Author:** Author name**Date:** Date

Module's Role

Explain the module's role in the system in general

Как вы можете видеть на рисунке выше, выходные данные Doxygen добавили подстраницу “**Название подмодуля**” под страницей “**Название модуля**” в древовидном представлении слева и добавили ссылку на нее на нашей странице.

Сама подстраница является обычной страницей и выглядит так, как показано на изображении ниже.



Обычно я избегаю разделов и подразделов, поскольку это усложняет процесс документирования. Вместо этого мне нравится выполнять их с помощью параметров форматирования Markdown (даже если из-за этого будет сложно связать один раздел с другим). Я оставляю за вами право самим решать, нужны ли они вам в вашем конкретном проекте или нет.

Установка Doxygen

Установка Doxygen максимально проста. Просто перейдите по этой [ссылке](#) и прокрутите вниз до раздела “Исходные тексты и двоичные файлы” и загрузите версию, поддерживающую вашу конкретную операционную систему, будь то Linux, Mac или Windows. Затем установите, как вы устанавливали бы любое другое обычное приложение!

Sources and Binaries

Latest release

The latest version of doxygen is 1.8.16 (release date August 8th 2019).

The **source** distribution

[doxygen-1.8.16.src.tar.gz](#) (5.2MB)

A binary distribution for **Linux x86-64**

[doxygen-1.8.16.linux.bin.tar.gz](#) (42.5MB)

Compiled using Ubuntu 16.04 with kernel 4.4.0 and gcc 5.4.0 and statically linked again libclang. This archive includes the HTML version of the manual, but not the GUI frontend.

A binary distribution for **Windows**. All versions of Windows since Vista are supported.

[doxygen-1.8.16-setup.exe](#) (46.4MB)

This is a self-installing archive that includes the HTML and compressed HTML versions of the manual and the GUI frontend. It bundles 32-bit and 64-bit versions of doxygen.exe, and will install the right one based on the OS.

If you are allergic to installers and GUIs, haven't sufficient bandwidth, or don't have administrator privileges you can also download the 32-bit doxygen binary in a zip (18.3MB) or the 64-bit version (22.2MB).

A binary distribution for **Mac OS X 10.9 and later**

[Doxygen-1.8.16.dmg](#) (125.8MB)

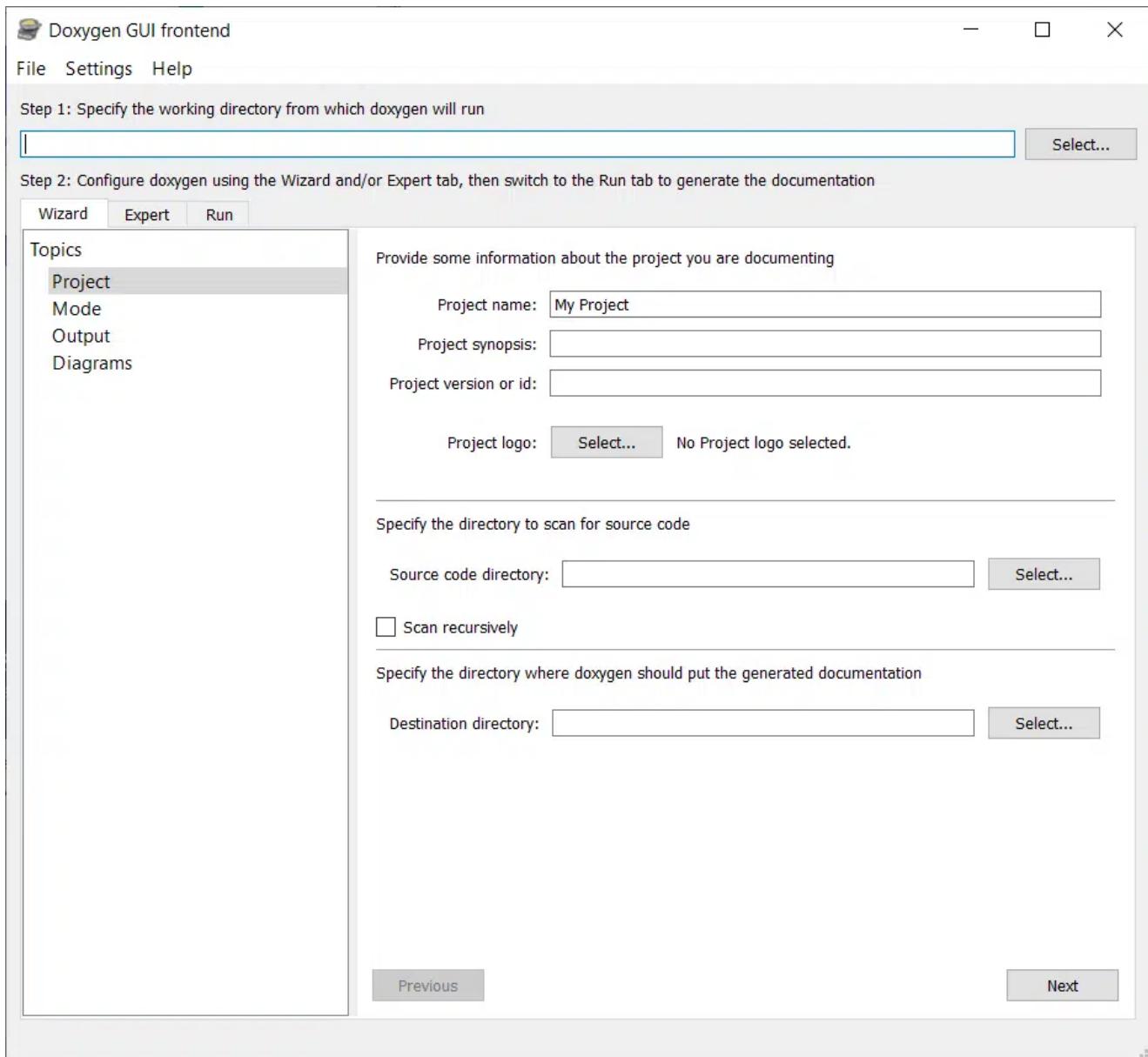
This is a self-contained disk image, which contains the GUI frontend. The binaries support the whole range of Intel CPUs (both 32 and 64 bit).

Alternatively you can download the files from [SourceForge](#).

[Страница загрузки Doxygen](#)

Графический интерфейс

После успешной загрузки и установки Doxygen откройте интерфейс Doxygen с графическим интерфейсом под названием Doxywizard. Графический интерфейс в Windows выглядит примерно так



Как вы можете видеть, здесь есть 3 вкладки с названиями

- Мастер
- Эксперт и
- Выполнить

Вкладка Мастера

Эта вкладка предназначена для новичков в использовании этого инструмента и содержит все основные необходимые опции для создания документации с использованием Doxygen

На вкладке мастера есть 4 “Раздела” или, как мне нравится называть это, “Вложенных вкладки” с названиями

- Проект
- Режим

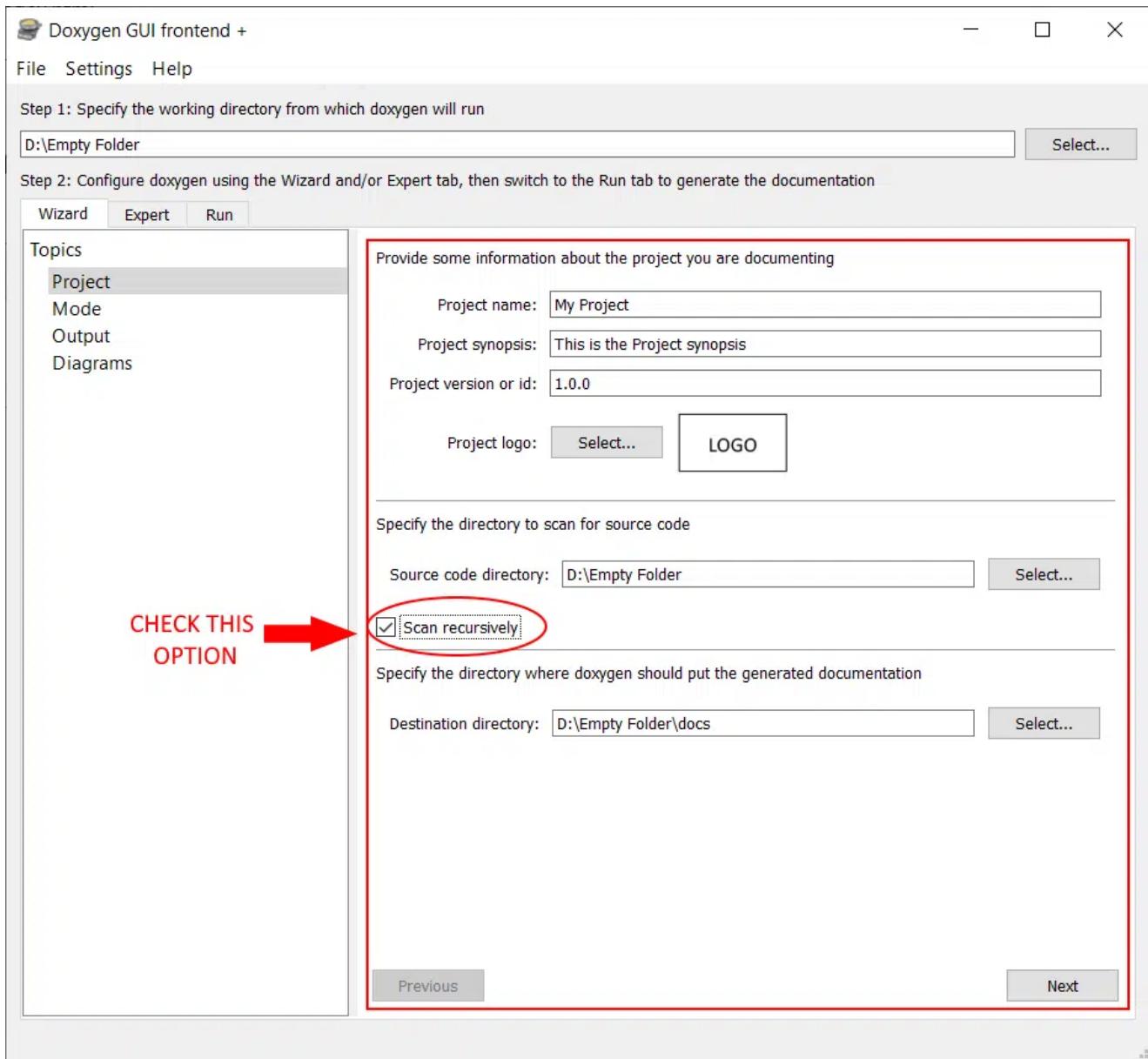
- Вывод и
- Схемы

Мастер-> Проект

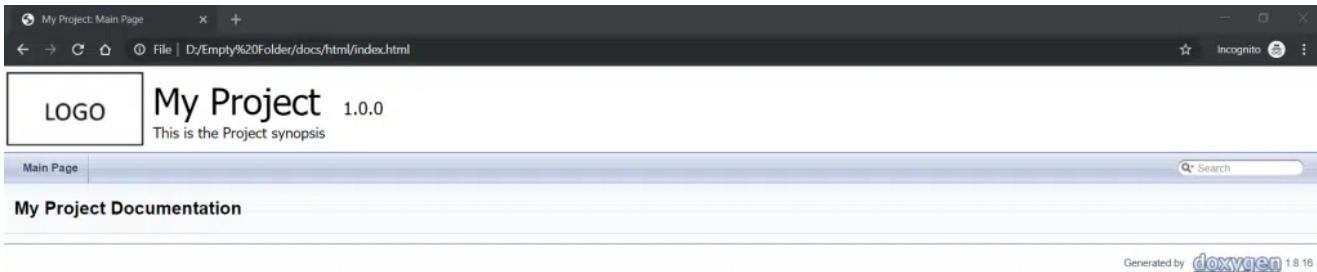
На картинке выше вложенная вкладка Project уже выбрана, и в ней вы получаете следующие 3 опции

- Сведения о проекте: Здесь вы можете ввести такие сведения, как заголовок проекта (название), подзаголовок (краткое описание), версию проекта и логотип проекта
- Расположение источника: здесь вы можете ввести корневой каталог вашего проекта, другими словами, путь к папке вашего проекта
- Место назначения: Здесь вы можете ввести папку, в которую вы хотите сохранить документацию, созданную Doxygen.

Обязательно установите флагок Сканировать рекурсивно, чтобы Doxygen выполнял поиск по всем вложенным папкам.



Если вы запустите Doxygen с приведенными выше настройками, предоставив ему пустую папку в качестве исходного каталога, вы получите что-то вроде этого.



Как вы можете видеть на картинке выше, логотип (который я создал для этого руководства), название проекта, версия и краткий обзор показаны в верхней части сгенерированного HTML-кода.

Оставьте высоту логотипа равной 100 пикселям, чтобы обеспечить хорошую подгонку, поскольку Doxygen недостаточно умен, чтобы изменить его размер за нас.

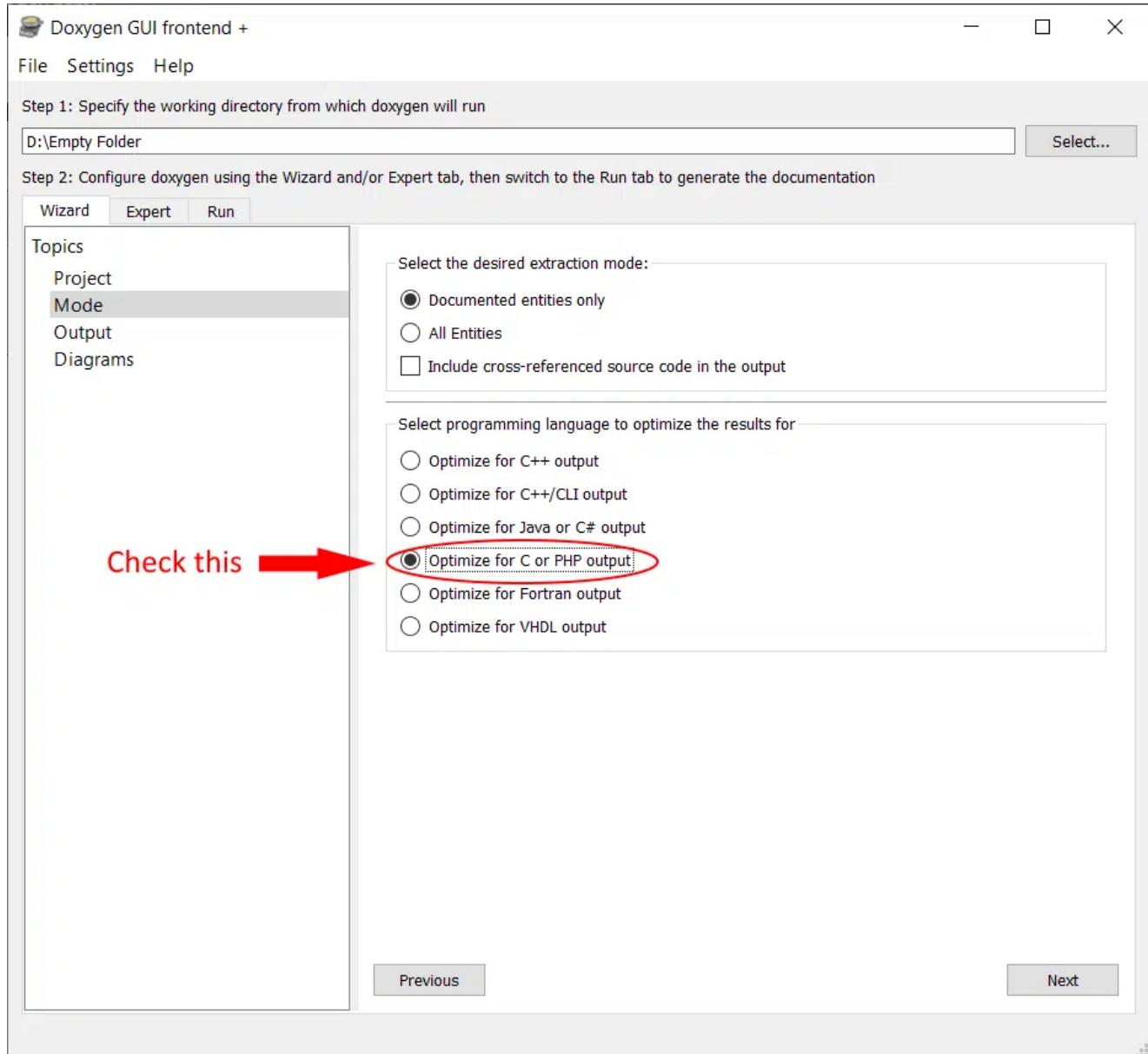
Мастер-> Режим

Здесь вы можете выбрать 2 варианта

1. **Режим извлечения:** позволяет вам решить, хотите ли вы использовать недокументированный код для генерации HTML или нет.
2. **Язык программирования:** позволяет вам выбрать свой язык программирования, чтобы Doxygen мог использовать соответствующий шаблон для создания документации.

Я предпочитаю сохранить значение по умолчанию для первого варианта, который является “Только документированные объекты”, поскольку это позволяет мне вести менее загроможденную документацию. Если вы выберете “Все объекты”, то все функции, глобальные переменные и файлы markdown, даже если они не задокументированы, будут использоваться с использованием специального синтаксиса, показанного в первом разделе этой статьи. Но некоторые люди предпочитают документировать все объекты, поскольку это указывало бы на функции, которые не задокументированы, как индикатор

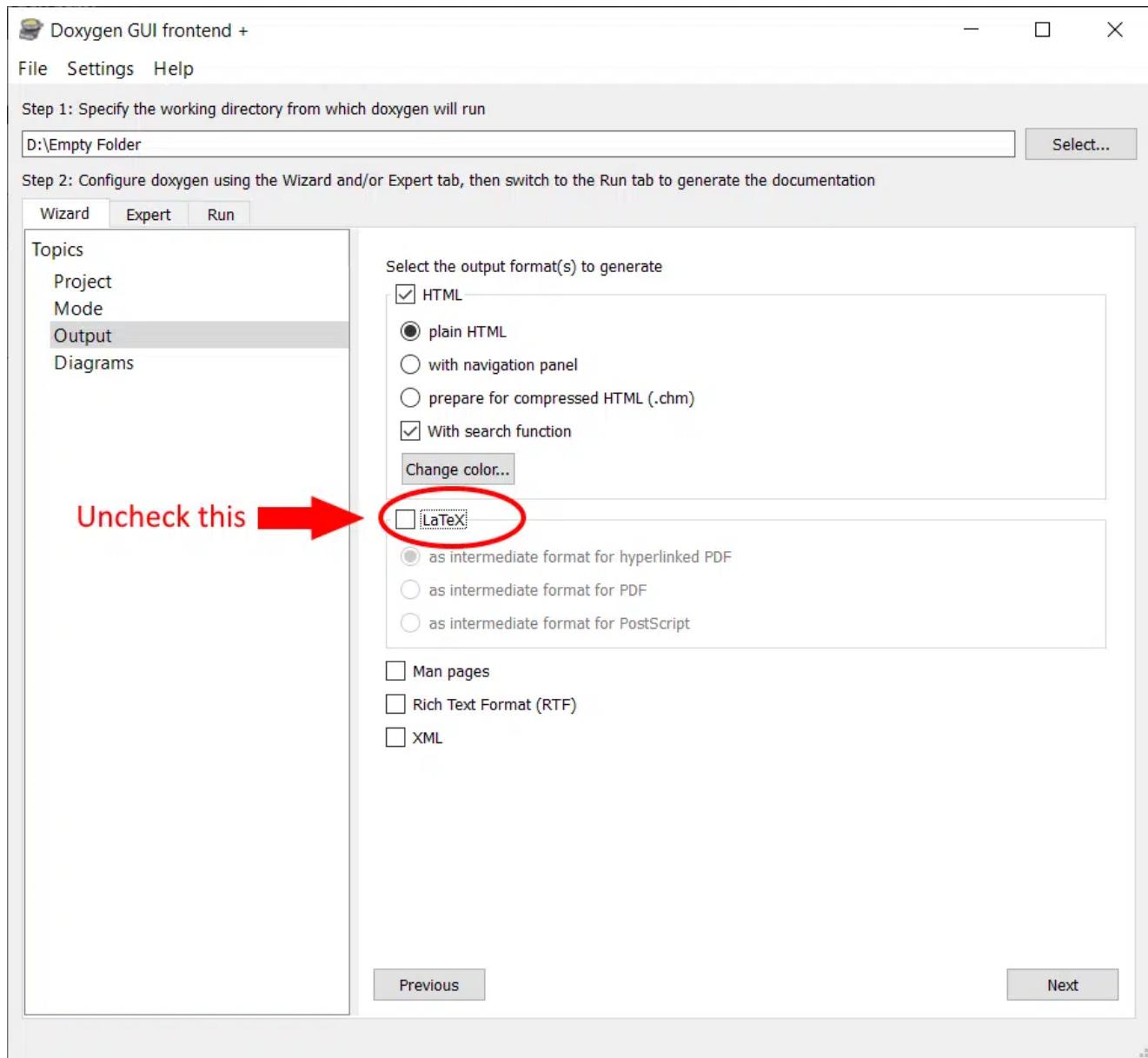
ТОГО, ЧТО ОН НУЖДАЕТСЯ В ДОКУМЕНТАЦИИ. Я ОСТАВЛЯЮ ЗА ВАМИ ПРАВО ПОЭКСПЕРИМЕНТИРОВАТЬ С ОБОИМИ ВАРИАНТАМИ И ВЫБРАТЬ ОДИН.



Убедитесь, что вы выбрали “Оптимизировать для вывода на С или PHP” для второго варианта, как показано выше.

Мастер-> Вывод

На этой вкладке вы можете выбрать нужный вам тип вывода. Лично я никогда не использую Latex, поэтому я снял этот флагок, а в остальном сохранил настройки по умолчанию, как показано на рисунке ниже.

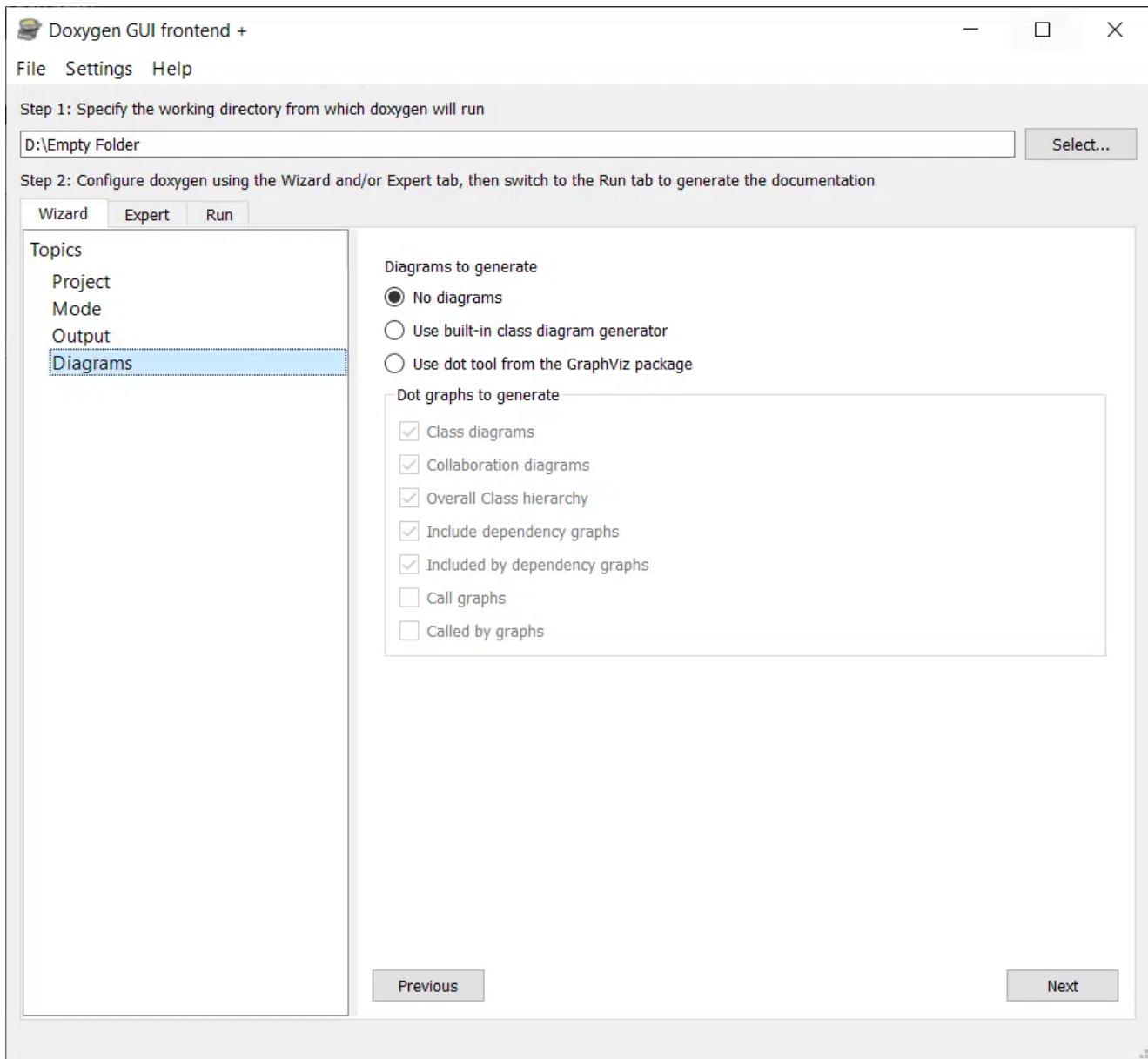


Мастер-> Схемы

Я мало экспериментировал с параметрами генерации диаграмм в Doxygen, поскольку большинство из них ориентированы на объектно-ориентированные языки.

Я пробовал call graphs и called by graphs и не нашел их очень полезными.

Лично я предпочитаю только диаграммы архитектуры, которые я создаю сам для своих проектов, поэтому я оставляю опцию диаграмм без диаграмм, как показано ниже.

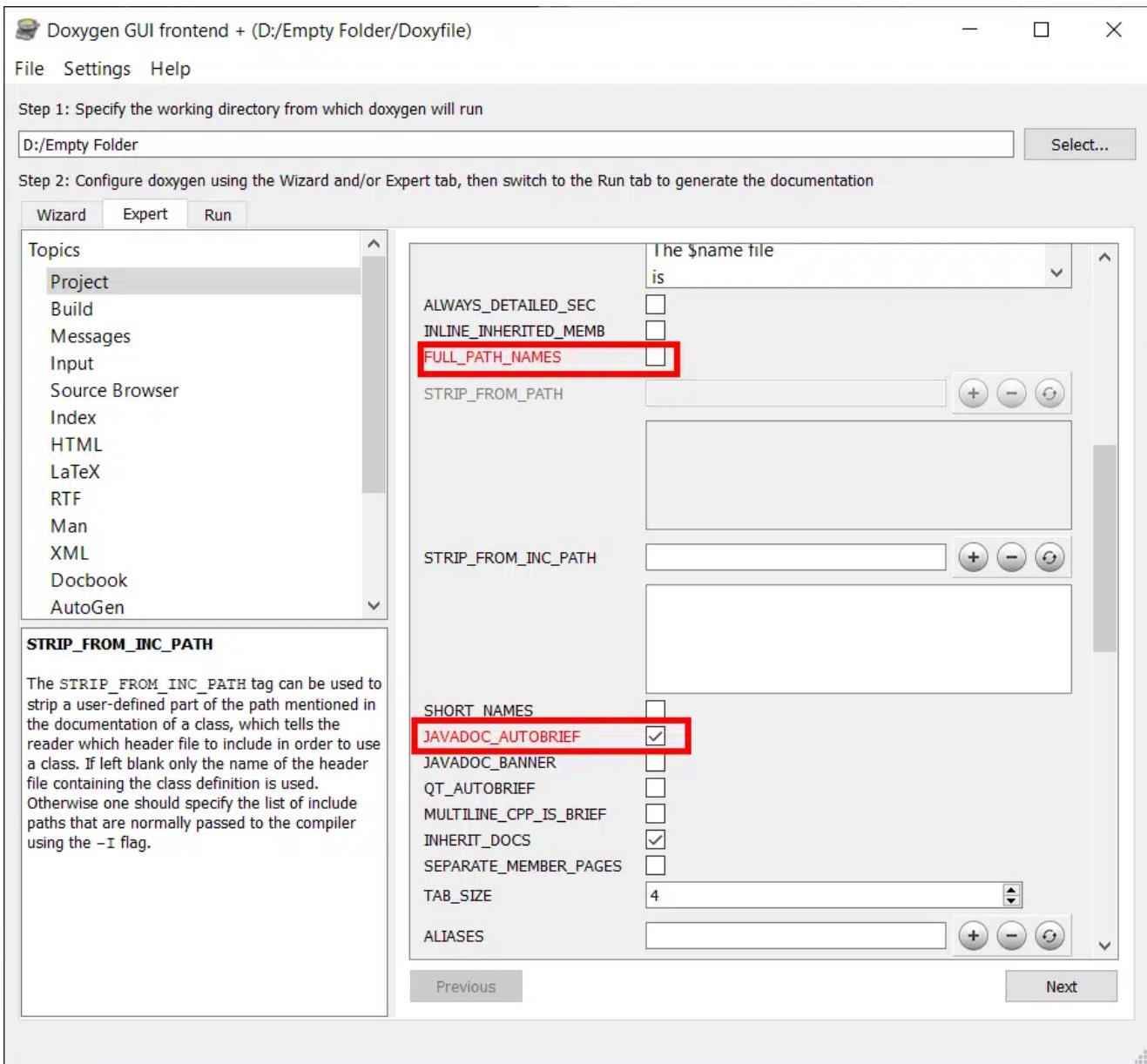


Вкладка Expert

Вкладка “Эксперт” содержит параметры конфигурации, недоступные на вкладке “Мастер”. Давайте изменим параметры на этой вкладке, чтобы дать нам больше контроля над выводом и сделать его адаптированным специально для исходного кода С.

Эксперт-> Проект

Вам не нужно много менять здесь, поскольку все параметры, которые необходимо изменить, уже были изменены, когда мы переходили на вкладку мастера. 2 простых изменения, которые можно выполнить в этом разделе, показаны на изображении ниже (красные прямоугольники показывают внесенные изменения).



Параметр `FULL_PATH_NAMES` был снят, чтобы конечный результат не включал полное имя пути, например `"C:/User/user_name/Desktop?project_directory/sub_directory/file_name"`. Вместо этого мы можем получить выходные данные для запуска из корневого каталога самого проекта. В приведенном выше случае выходной HTML-код будет отображать только **"подкаталог /имя файла"**

Как мы видели выше, существует 2 уровня описания для любой заданной сущности: краткое описание и подробное описание. Выбран следующий параметр `JAVADOC_AUTOBRIEF`, чтобы мы могли разделить краткое и подробное описание объектов, используя простую пунктуацию с точкой, чтобы первое предложение было кратким описанием, а все предложения после него были частью подробного описания.

```

1  /**
2   * Calculates the Area of the circle.
3   * Formula: Area = PI*r^2

```

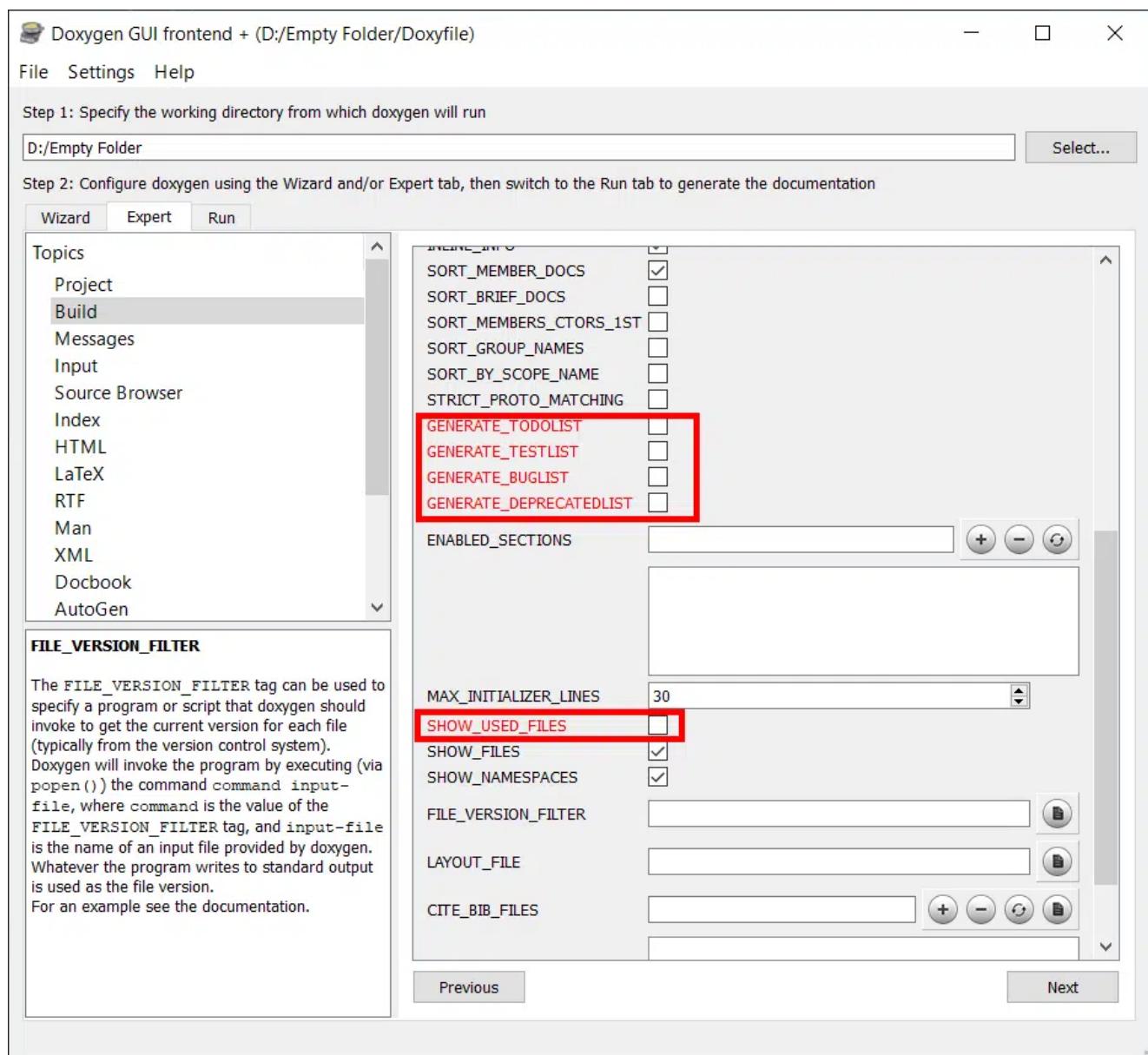
```

4 * @param[in] radius
5 * @param[out] area
6 */
7 float calculate_area(float radius)
8 {
9     float area;
10    area = PI * radius * radius;
11    return area;
12 }

```

В фрагменте кода выше первая строка "Вычисляет площадь окружности" является кратким описанием, а "Формула ..." - подробным описанием.

Эксперт-> Сборка



На вкладке сборка вы можете снять флагки со следующих параметров

1. GENERATE_TODOLIST
2. GENERATE_TESTLIST
3. GENERATE_BUGLIST

4. GENERATE_DEPRECATEDLIST

5. SHOW_USED_FILES

Первые 4 варианта, упомянутые выше, говорят сами за себя. Эти текущие тесты, ошибки и список устаревших не относятся к коду, скорее они должны быть частью вашего инструмента отслеживания исходного кода и управления им.

Последний параметр SHOW_USED_FILES был снят, чтобы предотвратить ненужный беспорядок в документации. В основном он печатает строку в конце страниц, в которой структурирован список файлов, использованных для создания документации. Для примера рассмотрим следующий заголовочный файл

```
1 // @file main.h
2 #ifndef MAIN_H_
3 #define MAIN_H_
4 /**
5  * Stores the calculated parameters of the circle.
6  * It stores the area and perimeter of the circle.
7 */
8 struct circle_params
9 {
10     float area; // < Area of the circle
11     float perimeter; // < Perimeter of the circle
12 };
13#endif
```

Он имеет простое определение структуры.

Теперь добавление этого файла и запуск его через Doxygen дает результат, который выглядит следующим образом.

Stores the calculated parameters of the circle. [More...](#)

```
#include <main.h>
```

Data Fields

float area

Area of the circle.

float perimeter

Perimeter of the circle.

Detailed Description

Stores the calculated parameters of the circle.

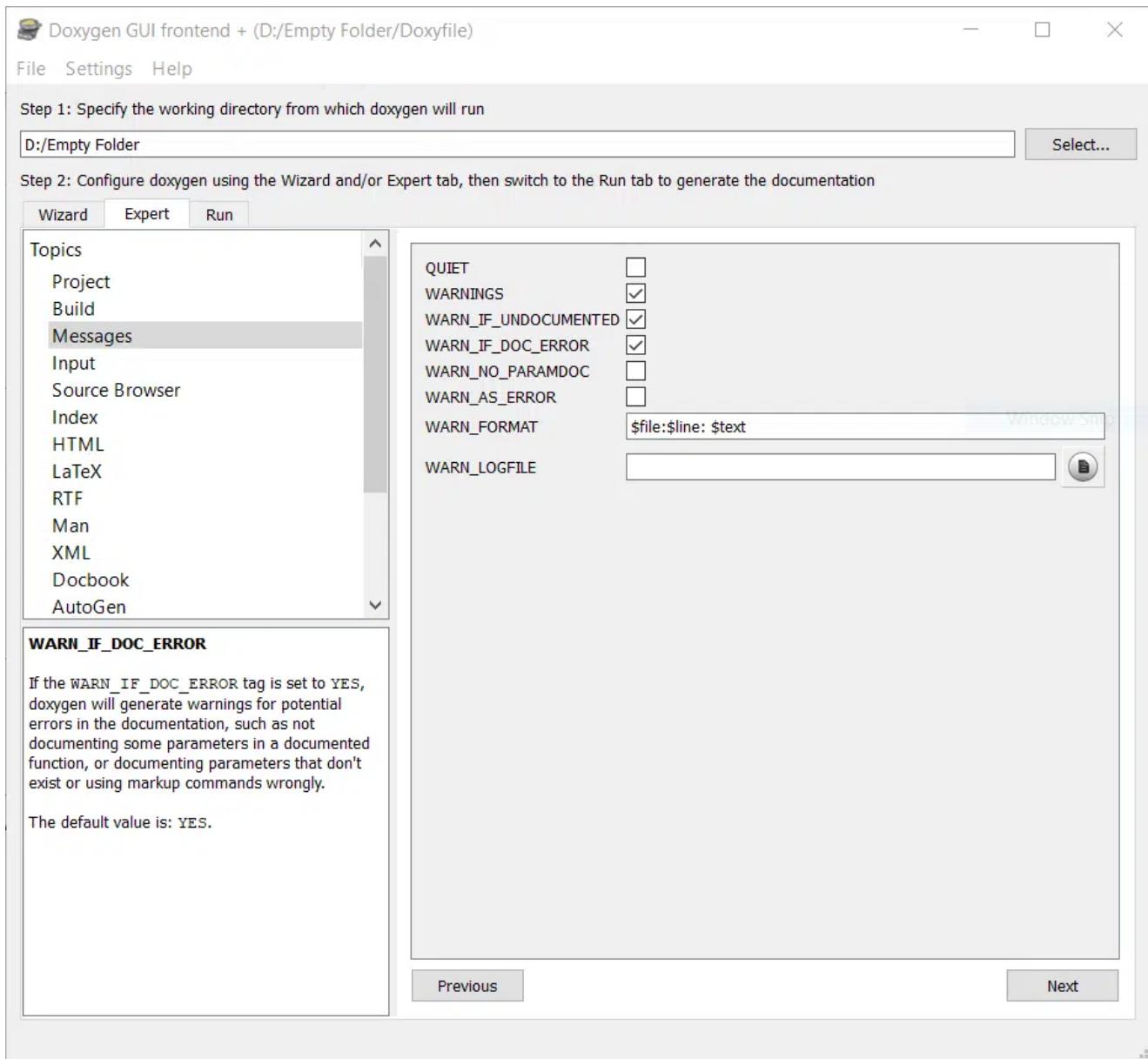
It stores the area and perimeter of the circle.

The documentation for this struct was generated from the following file:

- [main.h](#)

По сути, мы пытаемся исключить эту часть документации, сняв флагок с этой опции. Для простого примера кода эта часть может показаться приемлемой, но для проекта с реальным текстом беспорядок может накапливаться и мешать реальной полезной документации. Я предлагаю вам поэкспериментировать самим и выяснить, нужна ли вам эта часть или нет.

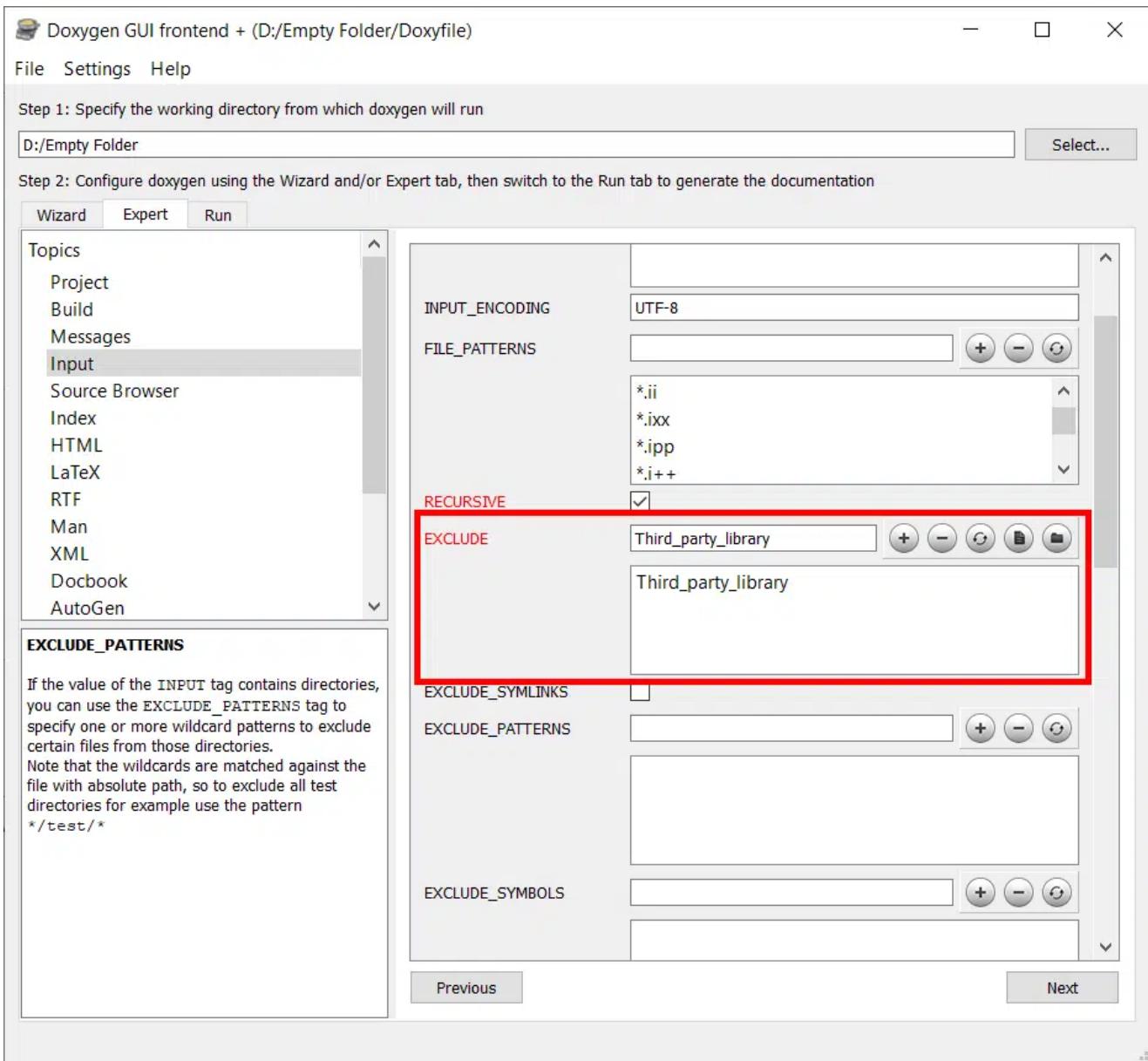
Эксперт-> Сообщения



Здесь вам не нужно ничего менять. В основном это управляет журналом, печатаемым при запуске Doxygen, и не имеет ничего общего с содержимым сгенерированного HTML.

Эксперт-> Ввод

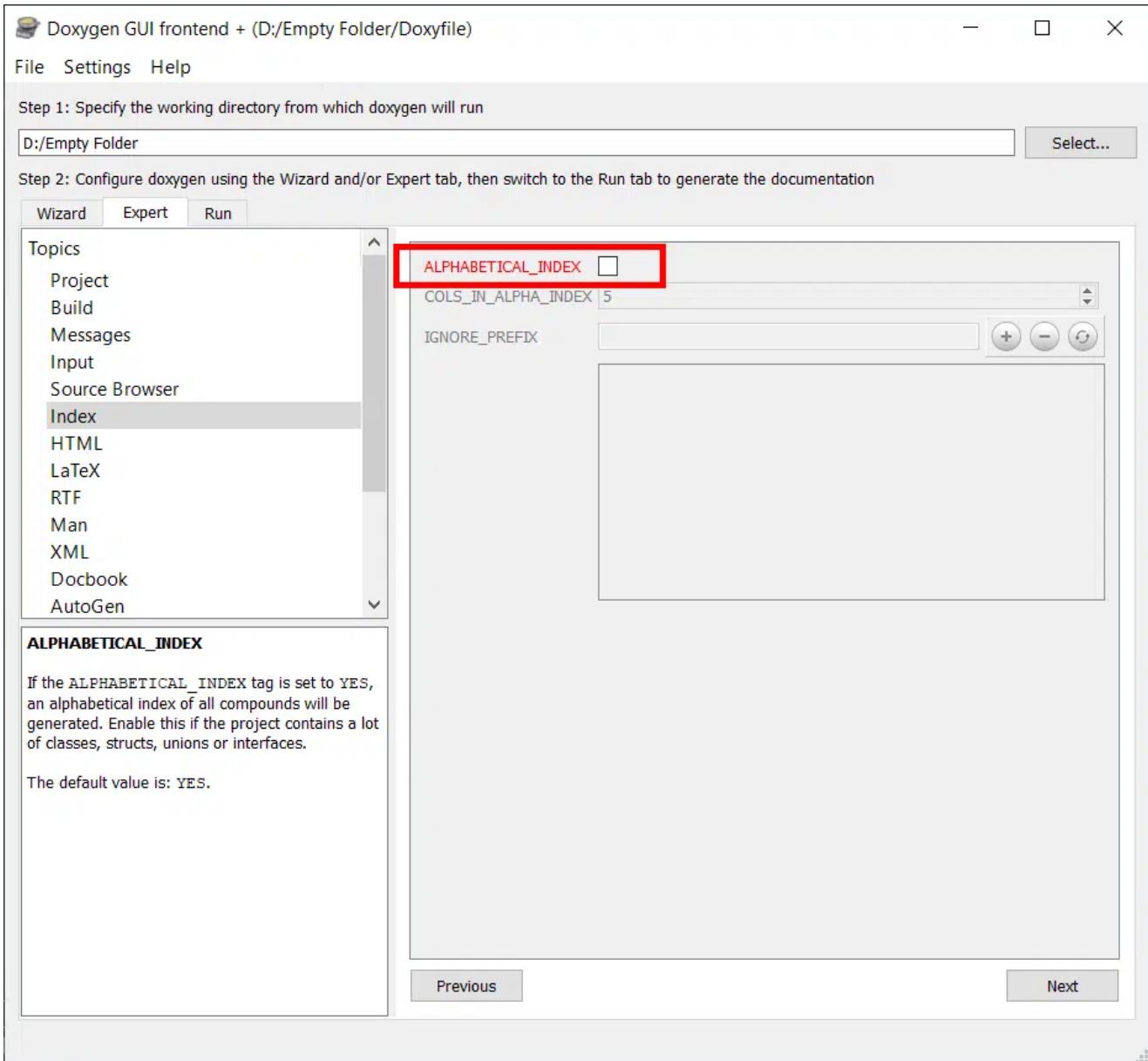
Это руководство содержит опции для управления файлами, которые используются для создания документации. При написании кода на С мы часто сталкиваемся с использованием библиотек сторонних производителей. Эти библиотеки обычно содержат большое количество файлов, и это может привести к ненужному загромождению документации. Если у вас есть папки, которые необходимо исключить из документации, вы можете добавить их на вкладку исключить, как показано ниже.



Эксперт-> Браузер исходного кода

Здесь никаких изменений вносить не нужно.

Эксперт-> Индекс

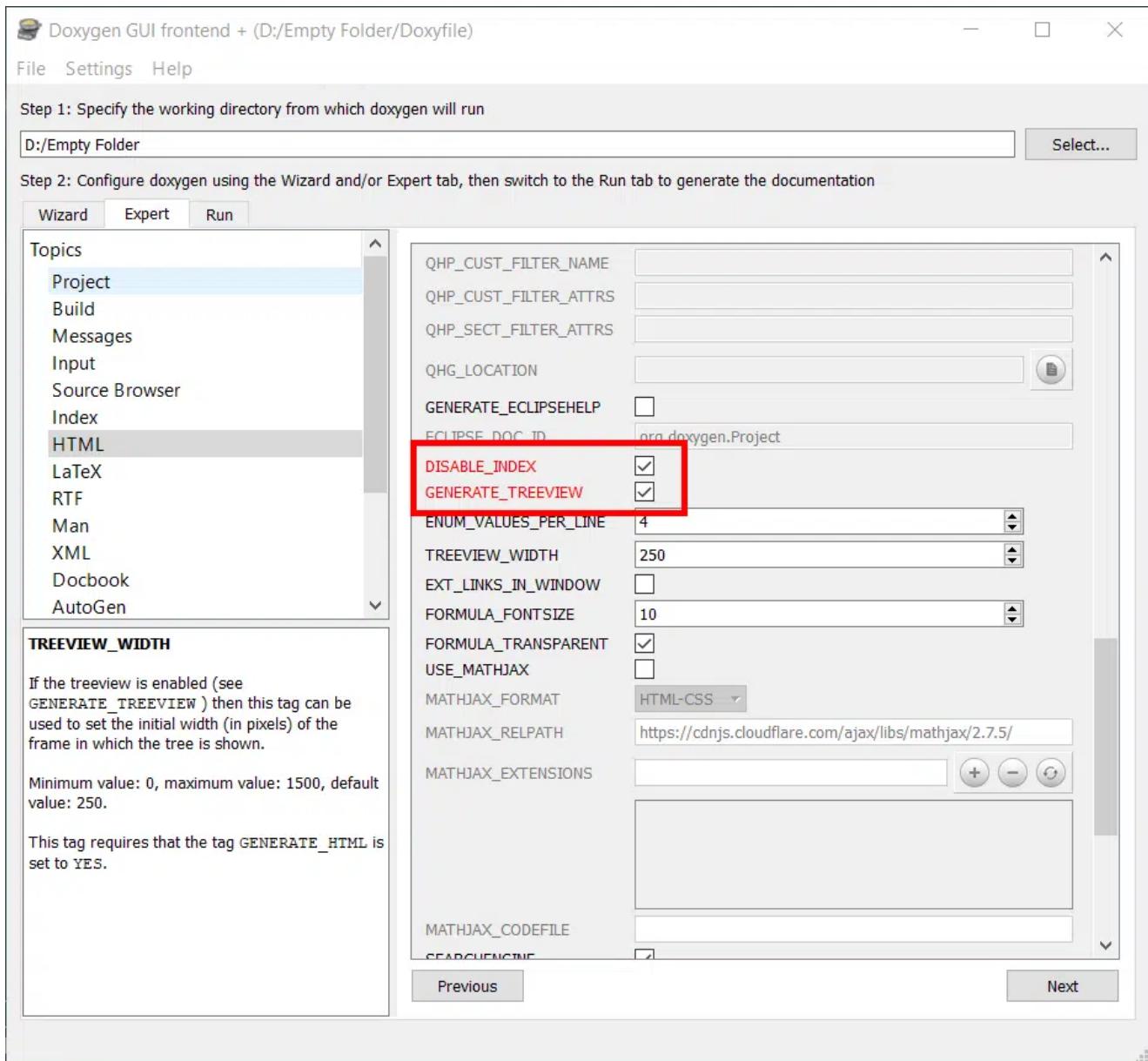


Я предлагаю вам отключить эту опцию ***ALPHABETICAL_INDEX***, иначе в итоге вы получите страницы, подобные этой. Обычно я запоминаю модули, в которые помещена конкретная структура, а не ее фактическое название, поэтому я снимаю этот флагок, чтобы исключить подобную документацию. Вы можете поэкспериментировать и выяснить сами, если вам это нужно.



Эксперт-> HTML

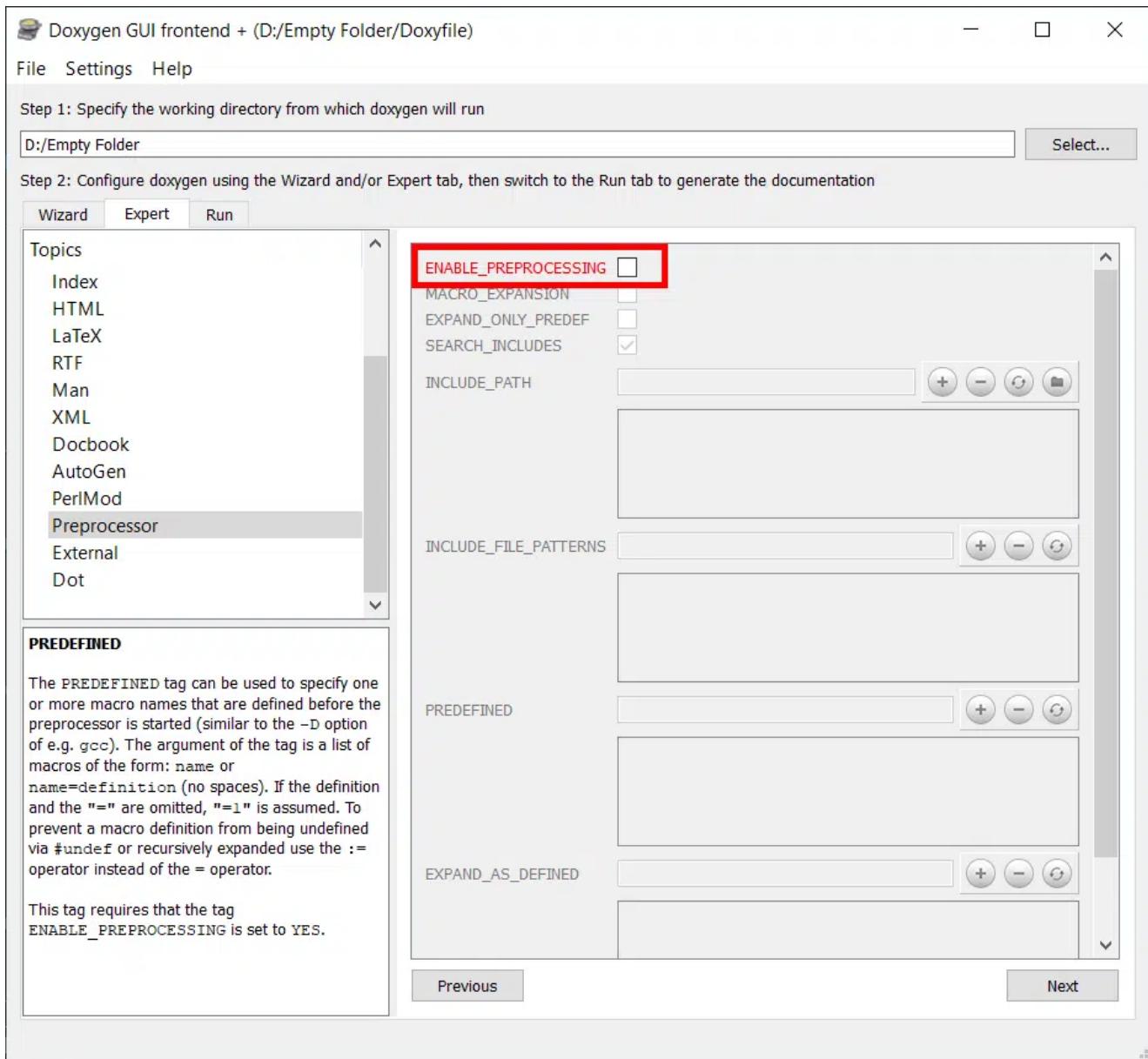
Здесь проверьте параметры ***DISABLE_INDEX*** и ***GENERATE_TREEVIEW***, как показано на рисунке ниже, чтобы получить тот же удобный вид дерева слева, кото мы видели на протяжении всей этой статьи.



Эксперт-> Latex, RTF, Docbook, AutoGen, PerlMod

Здесь нет изменений.

Эксперт-> Препроцессор

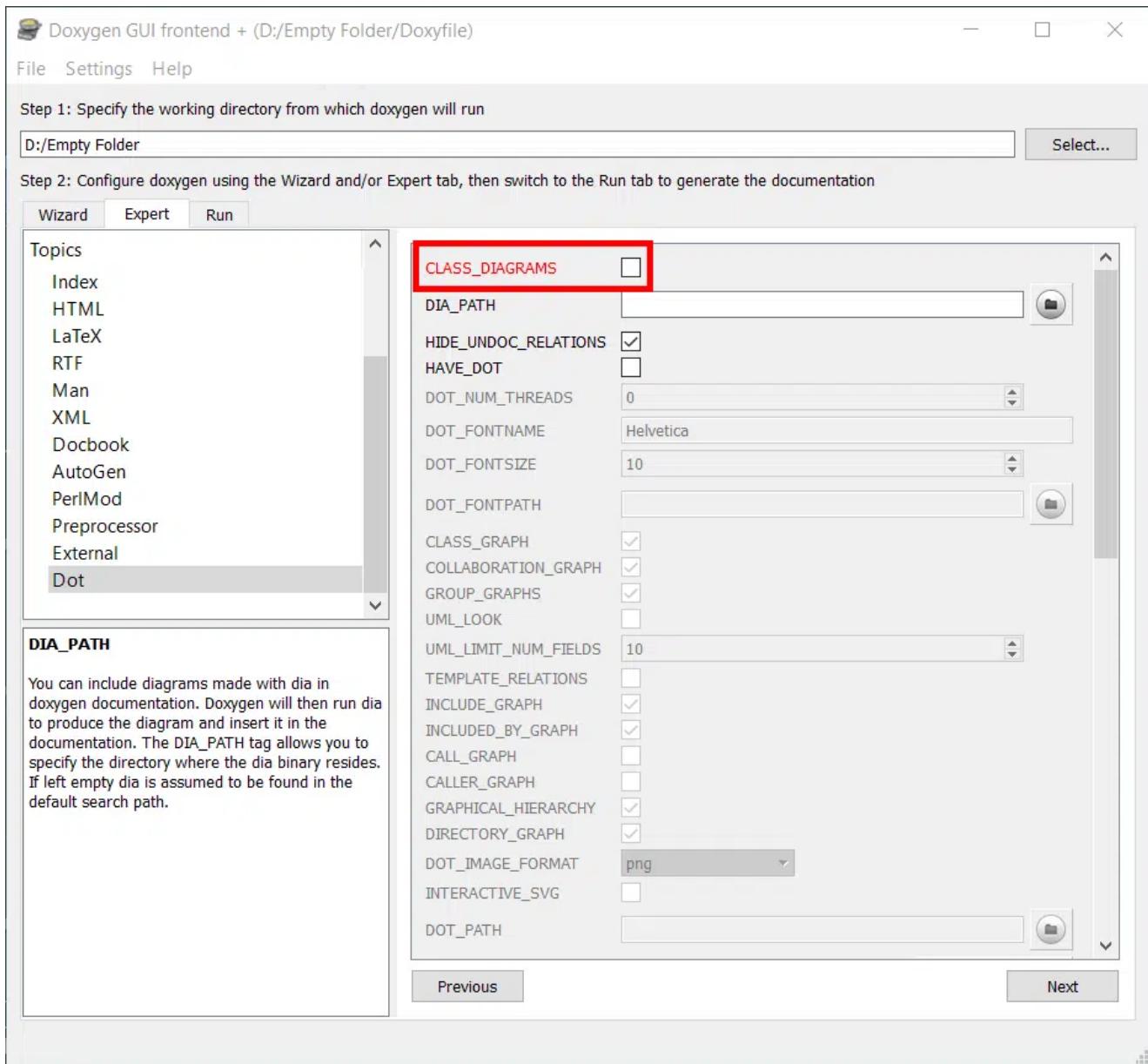


Мне нравится снимать флажок с опции предварительной обработки, чтобы я мог видеть код таким, каким я его написал, а не измененным в результате предварительной обработки, выполняемой стороной Doxygen, поэтому лучше снять флажок. Опять же, вы можете поэкспериментировать сами, чтобы увидеть и решить, имеет ли для вас смысл предварительная обработка.

Эксперт-> Внешний

Здесь никаких изменений вносить не нужно.

Эксперт-> Точка



Диаграммы, создаваемые Doxygen, в большей степени ориентированы на объектно-ориентированное программное обеспечение, и, следовательно, вы можете снять флаги с диаграмм классов. Если вы хотите, вы также можете поэкспериментировать с использованием "Dot", который является генератором диаграмм для Doxygen. Но мне нравится сохранять свои диаграммы на уровне архитектуры, а не на уровне кода, и поэтому я предпочитаю не использовать его.

Файл Doxyfile

Вы можете загрузить сгенерированный doxyfile, перейдя по этому руководству по [этой ссылке](#), и использовать его в своих собственных проектах!

Я надеюсь, что вы, ребята, узнали что-то из этого поста и надеюсь, что он был вам полезен.

Вы можете [отправить нам электронное письмо](#) или связаться с нами по этой [ссылке](#), если у вас есть какие-либо вопросы или предложения.

Если вам понравился пост, не стесняйтесь поделиться им со своими друзьями и коллегами!

Статьи по теме

[Как правильно использовать возможности комментариев в вашем коде?](#)

[C: макрофункция против обычной функции против встроенных функций](#)



Редактор

Баладжи Гунасекаран

Баладжи Гунасекаран - старший инженер-программист со степенью магистра наук в области мехатроники и степенью бакалавра в области электротехники и электроники. Он любит писать о технологиях и написал более 300 статей. Он также опубликовал книгу "Cracking the Embedded Software Engineering Interview". Вы можете подписаться на него на [LinkedIn](#)

Категории

Select Category



Категории

Select Category ▾

Поиск ...

Об авторе



Баладжи Гунасекаран

Баладжи Гунасекаран - старший инженер-программист со степенью магистра наук в области мехатроники и степенью бакалавра в области электротехники и электроники. Он любит писать о технологиях и написал более 300 статей. Он также опубликовал книгу "Cracking the Embedded Software Engineering Interview". Вы можете подписаться на него на [LinkedIn](#).

[Политика конфиденциальности](#)

[Политика Возврата](#)

[Условия использования](#)

[О компании EmbeddedInventor.com](#)

[Связаться с нами](#)

© 2024 embeddedinventor.com