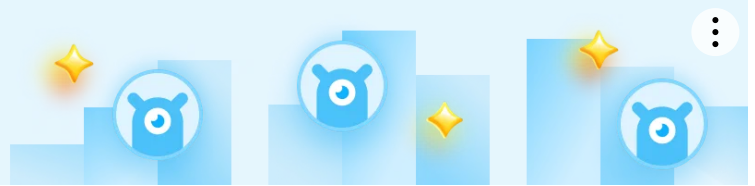


Топ-100
авторов Хабра



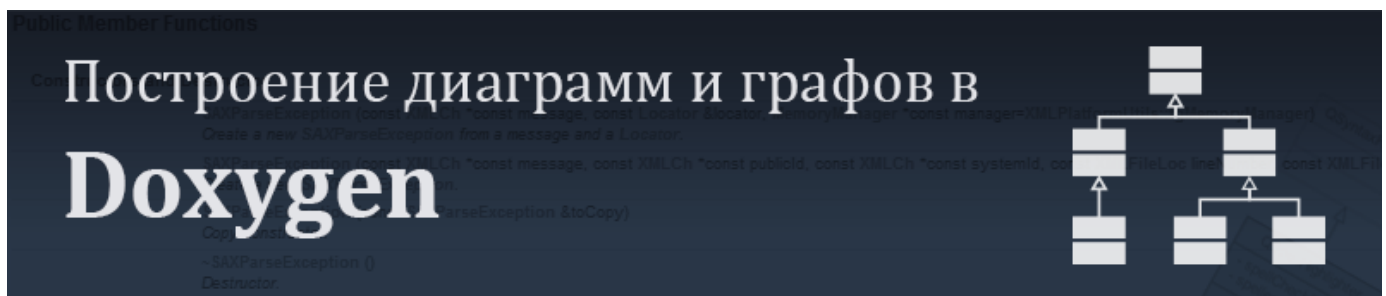
Norserium

17 мар 2015 в 15:18

Построение диаграмм и графов в Doxygen

🕒 9 мин 👁 60K

Программирование*, C++*, C*, C#*



Данная статья входит в получившийся цикл статей о системе документирования Doxygen:

1. Документируем код эффективно при помощи Doxygen
2. Оформление документации в Doxygen
3. Построение диаграмм и графов в Doxygen

Она завершает цикл статей о системе документации Doxygen. На этот раз статья посвящена построению различных диаграмм и графов в Doxygen. В ней мы рассмотрим основные их виды, различные способы их настройки и оформления, а также приведём ряд примеров и советов по их использованию.

Введение

Doxygen позволяет автоматически строить на основе исходного кода достаточно большое количество разнообразных диаграмм и графов, призванных проиллюстрировать те или иные элементы исходного кода, при этом любой график может служить также и способом навигации по коду и документации, поскольку Doxygen автоматически проставляет поверх самого графика ссылки на соответствующие разделы документации. В живую на результат его работы можно посмотреть, например, [здесь](#) и [здесь](#) (в последнем примере они для

компактности свернуты).

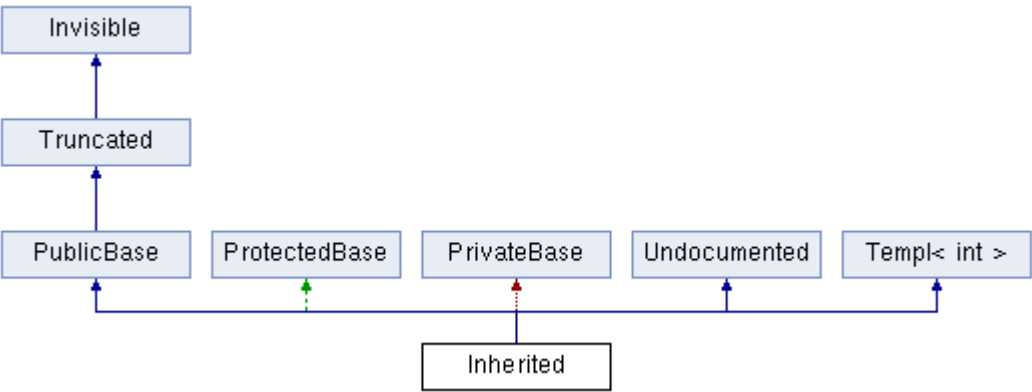
Основными типами диаграмм и графов, которые можно построить с его помощью, являются:

- Диаграмма наследования классов
- Диаграмма кооперации классов
- Диаграмма иерархии классов
- Граф вызовов
- Граф вызывающих функций
- Граф зависимостей (прямых и обратных)

Далее мы подробно рассмотрим каждый из этих типов, но прежде сделаем ряд общих замечаний и поговорим об основных настройках, которые нам пригодятся при их построении. Вообще, многое из того, что будет описано, вы можете найти в соответствующем разделе документации.

Основы построения диаграмм и графов

По умолчанию возможности Doxygen для построения различного рода диаграмм и графов очень ограничены (доступна только диаграмма наследования классов), пример такой диаграммы приведён ниже:



Для того, чтобы строить более продвинутые диаграммы и графы потребуется Graphviz, специальное открытое программное обеспечение, предназначенное для построения различного рода графов. Скачать его можно [здесь](#).

После его установки необходимо выполнить некоторые изменения в файле настроек Doxygen. Для этого используются следующие команды:

Команда	Назначение
---------	------------

HAVE_DOT	Показывает, что для генерации диаграмм и графов необходимо использовать Graphviz (по умолчанию отключена)
DOT_PATH	Указывает путь к исполняемому файлу «dot.exe» (по умолчанию предполагается, что путь к нему имеется в переменной <i>path</i> ; как правило, так оно и есть).

Среди дополнительных настроек, которые могут пригодиться в дальнейшем, можно отметить следующие:

Команда	Назначение
DOT_GRAPH_MAX_NODES	Устанавливает максимальное количество узлов, которые будут отображаться в графе. Если число узлов станет больше установленного значения, то граф будет урезан, в результате чего родительский узел, чьи дочерние узлы были скрыты, станет выделен красным (если же число узлов было задано меньше, чем число детей корневого узла, то он вовсе не будет показан)
MAX_DOT_GRAPH_DEPTH	Устанавливает максимальную глубину графа (глубина равна числу граней от наиболее удаленного узла графа до корневого узла). Опять же, если глубина графа будет превышать это значение, то он будет обрезан, в результате чего родительский узел, чьи дочерние узлы были скрыты, станет выделен красным
DOT_IMAGE_FORMAT	Указывает формат, в котором будут создаваться диаграммы и графы (поддерживаются «png», «jpg», «svg», «gif»)
DOT_FONTNAME	Устанавливает гарнитуру используемую для отображения текста на графах и диаграммах
DOT_FONTSIZE	Указывает кегль (высота букв, измеряемая в пунктах) для текста на графах и диаграммах
DOT_FONTPATH	Указывает путь к папке со шрифтовыми файлами
HTML_DYNAMIC_SECTIONS	В случае установки этой опции, Doxygen будет сворачивать определенные элементы HTML документации (например, графы), которые пользователь по желанию может впоследствии раскрыть

Итак, теперь когда мы разобрались с основными настройками, мы готовы приступить к непосредственно основным типам диаграмм и графиков.

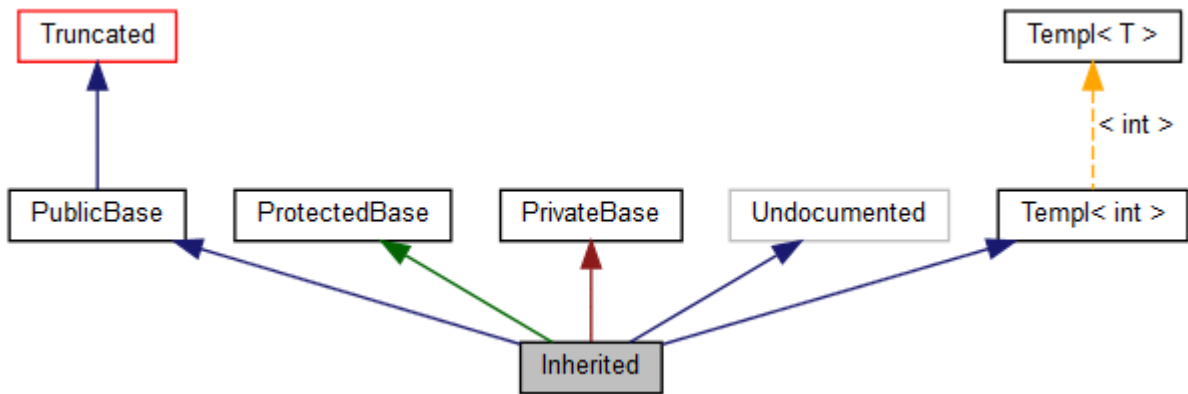
Диаграмма наследования классов

Диаграмма наследования классов (class graph) предназначена для графического отображения отношений наследования между классами (как «вверх», так и «вниз»).

Для того, чтобы сгенерировать такую диаграмму для всех документированных классов (при этом если в файле настроек установлена, например, опция `EXTRACT_ALL`, то к документированным классам относятся все классы) необходимо установить следующую опцию в файле настроек:

```
CLASS_GRAPH = YES
```





Ниже приведён пример построения такой диаграммы для тестового примера (он взят из легенды, создаваемой Doxygen), иллюстрирующего различные виды отношений между классами:



▸ [Исходный код примера и используемые настройки](#)

Теперь рассмотрим, что означают те или иные условные обозначения.

	Корневой узел
	Обычный узел, соответствующий документированному классу
	Узел, соответствующий недокументированному классу
	Родительский узел, чьи дочерние узлы были скрыты (например, в силу ограничения на число узлов в графе или его глубину)

	Указывает на открытое наследование
	Указывает на защищенное наследование
	Указывает на закрытое наследование
	Указывает, что класс является специализацией некоторого шаблонного класса, при этом подпись рядом со стрелкой указывает на то, какой шаблонный параметр был указан при специализации

Следует обратить внимание на следующие опции:

Опция	Значение	По умолчанию
HIDE_UNDOC_RELATIONS	Показывает, необходимо ли скрывать связь с некоторым классом, если он недокументирован или не является классом	YES
TEMPLATE_RELATIONS	Устанавливает, необходимо ли отображать связь, показывающую, что один класс является специализацией другого	NO

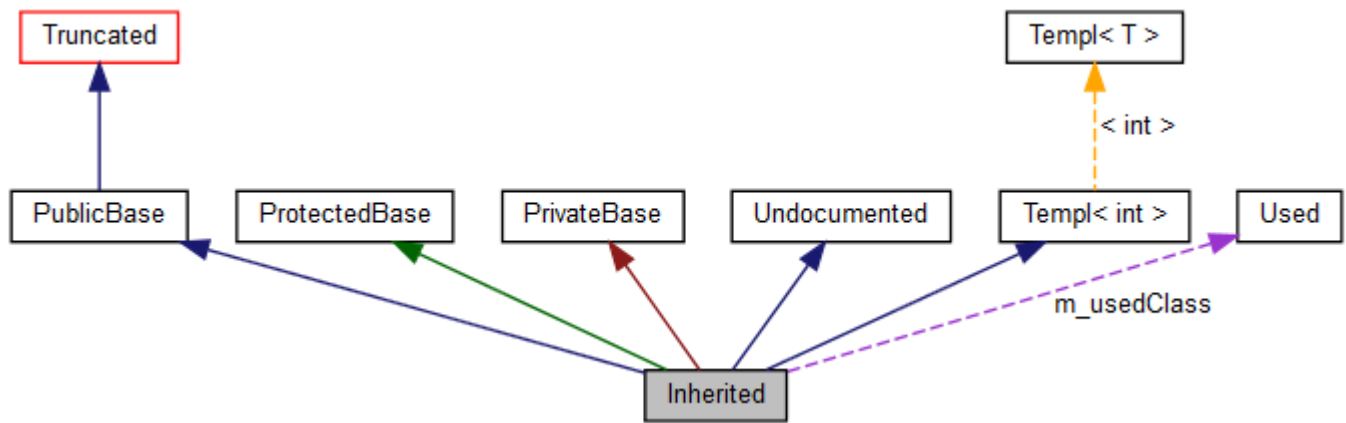
Диаграмма кооперации классов

Диаграмма кооперации классов (collaboration graph) предназначена для графического отображения отношений наследования между классами (только «вверх», то есть показаны будут только родительские классы), а также взаимодействие между ними, под которым в данном случае понимается то, что один класс содержит среди атрибутов объект другого класса.

Для того, чтобы включить диаграмму наследования классов, необходимо установить команду COLLABORATION_GRAPH:

COLLABORATION_GRAPH = YES

Ниже представлена диаграмма кооперации классов для того же примера, который был рассмотрен ранее:



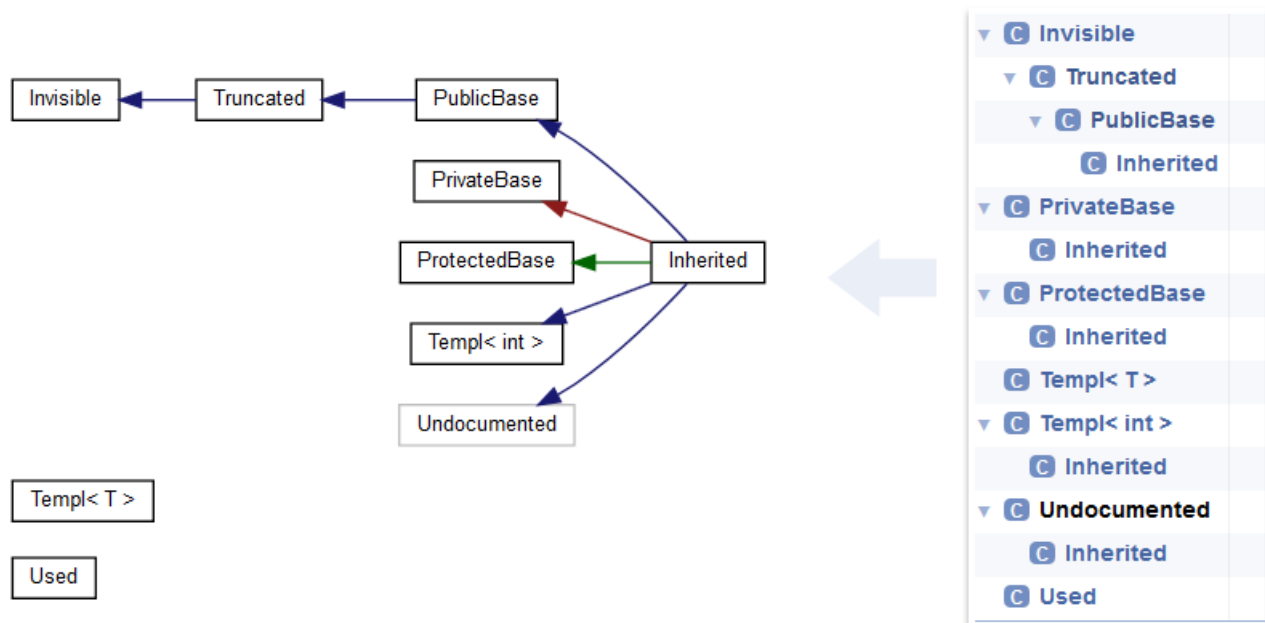
Следует отметить, что в данном случае отличие от диаграммы наследования классов состоит только в том, что здесь отображена связь между классами *Inherited* и *Used*, которая выражается в том, что первый класс имеет атрибут, являющийся экземпляром второго класса. Обозначение для такой связи следующее:

	Указывает, что один класс содержит атрибут, являющийся экземпляром другого класса, при этом подпись рядом со стрелкой указывает на имя соответствующего атрибута
--	--

Всё остальное, сказанное ранее про настройки диаграммы наследования классов остаётся справедливым и здесь.

Диаграмма иерархии классов

Диаграмма иерархии классов (graphical hierarchy) предназначена только для отображения отношений наследования между множеством разных классов. Фактически, она представляет собой графическое представление текстового описания иерархии классов, которое генерируется по умолчанию:



Граф вызовов

Прежде, чем перейти к описанию данного типа графов, договоримся, что всё сказанное далее справедливо и для функций и для методов классов, но для краткости мы будем употреблять слово «функции». Итак, граф вызовов (call graph), иллюстрирует то, какие функции вызываются внутри тела документируемой функции, а также во всех вложенных функциях (глубину можно настраивать, задавая максимальную глубину графа в файле настроек).

Для того, чтобы построить граф вызовов функции могут быть использованы два подхода: установка опции, которая вызовет построение графов вызовов для всех функций, и использование специальной команды, которая добавляется в описание функции (последнее может оказаться предпочтительнее, если вам надо проиллюстрировать конкретную функцию или ваш проект очень большой).

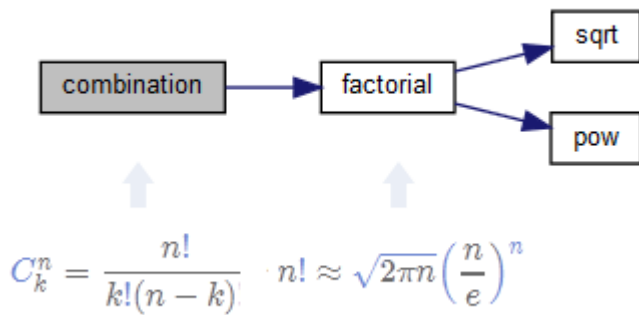
Для первого подхода необходимо установить следующую опцию:

```
CALL_GRAPH = YES
```

Для второго подхода используется следующая команда:

```
\callgraph
```

Пример графа вызова изображен ниже. Формулы на изображении ниже даны для примера (функция *combination* вычисляет число сочетаний, а для вычисления факториала использует формулу Стирлинга):



Граф вызывающих функций

Граф вызывающих функций (caller graph), иллюстрирует цепочку вызовов различных функций в результате которого вызывается документируемая функция, (глубина такой цепочки может быть настроена путём настройки максимальной глубины графа в файле настроек).

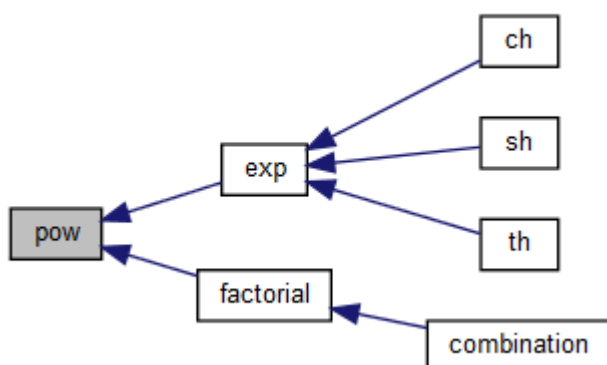
Опять же есть два способа построения таких графов. Первый способ – это установка соответствующей опции в файле настроек:

```
CALLER_GRAPH = YES
```

Второй способ – это использование специальной команды:

```
\callergraph
```

Простой пример графа вызывающих функций приведён ниже:



Кратко поясним его: для вычисления факториала по формуле Стирлинга используется функция возведения в степень (*pow*), для вычисления экспоненты также используется данная функция, а вычисление экспоненты может быть использовано для вычисления

значений гиперболических функций.

Граф зависимостей

Граф зависимостей идейно близок к графу вызовов, только он иллюстрирует не вызовы, а зависимости между различными файлами исходного кода.

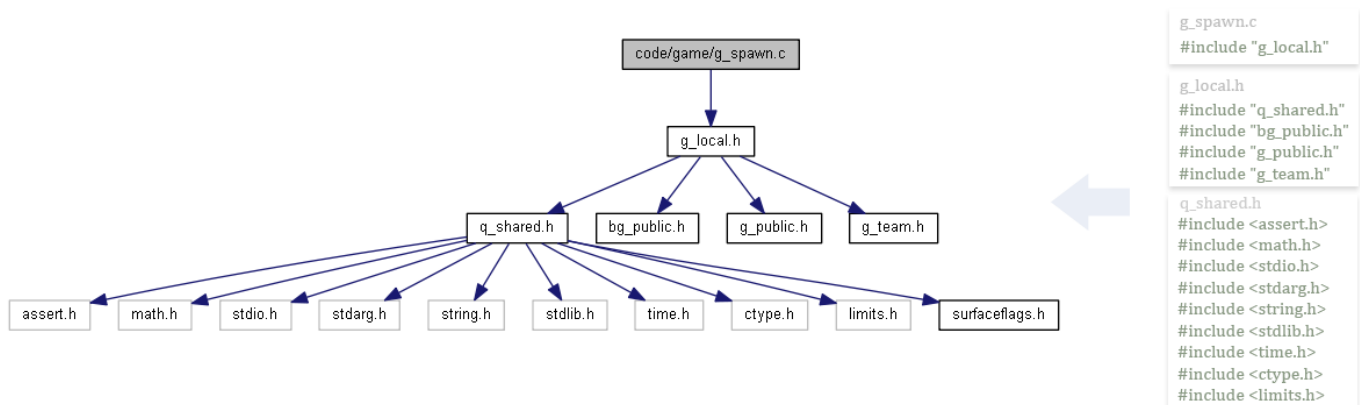
Прямой граф зависимостей

Прямой граф зависимостей (`include graph`) иллюстрирует цепочку зависимостей вплоть до документируемого файла, т.е. то, какие файлы были подключены к документируемому файлу, затем какие файлы, в свою очередь, были подключены к тем файлам и т.д.

Для построения такого рода графов необходимо установить следующую опцию в файле настроек:

```
INCLUDE_GRAPH = YES
```

Пример приведён ниже:



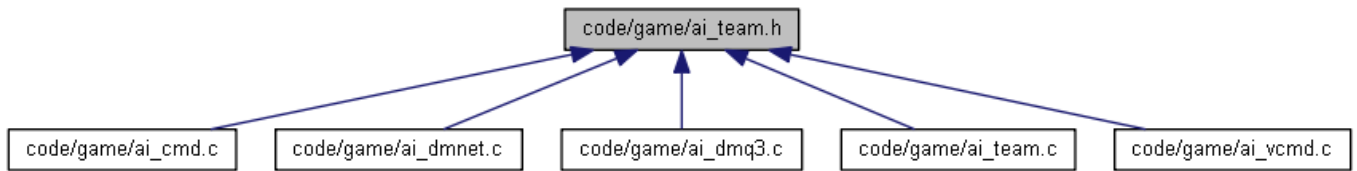
Обратный граф зависимостей

Обратный граф зависимостей (`included by graph`) иллюстрирует цепочку зависимостей от документируемого файла, т.е. то какие файлы включают в себя документируемый файл, затем в какие файлы включены данные файлы и т.д.

Для построения такого рода графов необходимо установить следующую опцию в файле настроек:

```
INCLUDED_BY_GRAPH = YES
```

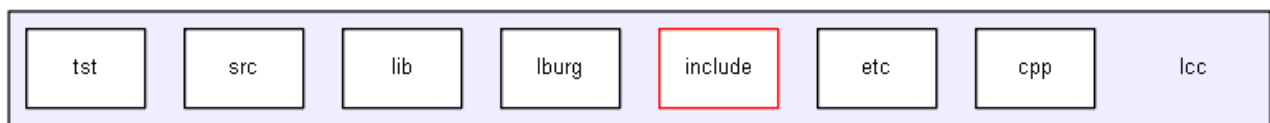
Пример приведён ниже:



Граф директорий

Граф директорий (directory graph) предназначен для графического представления рассматриваемой директории. Для построения такого графа используется следующая опция:

```
DIRECTORY_GRAPH = YES
```



Directories

directory	cpp
directory	etc
directory	include
directory	lburg
directory	lib
directory	src
directory	tst

Обратите внимание на то, что один из элементов графа имеет красную обводку, это означает, что в данную директорию вложены другие директории.

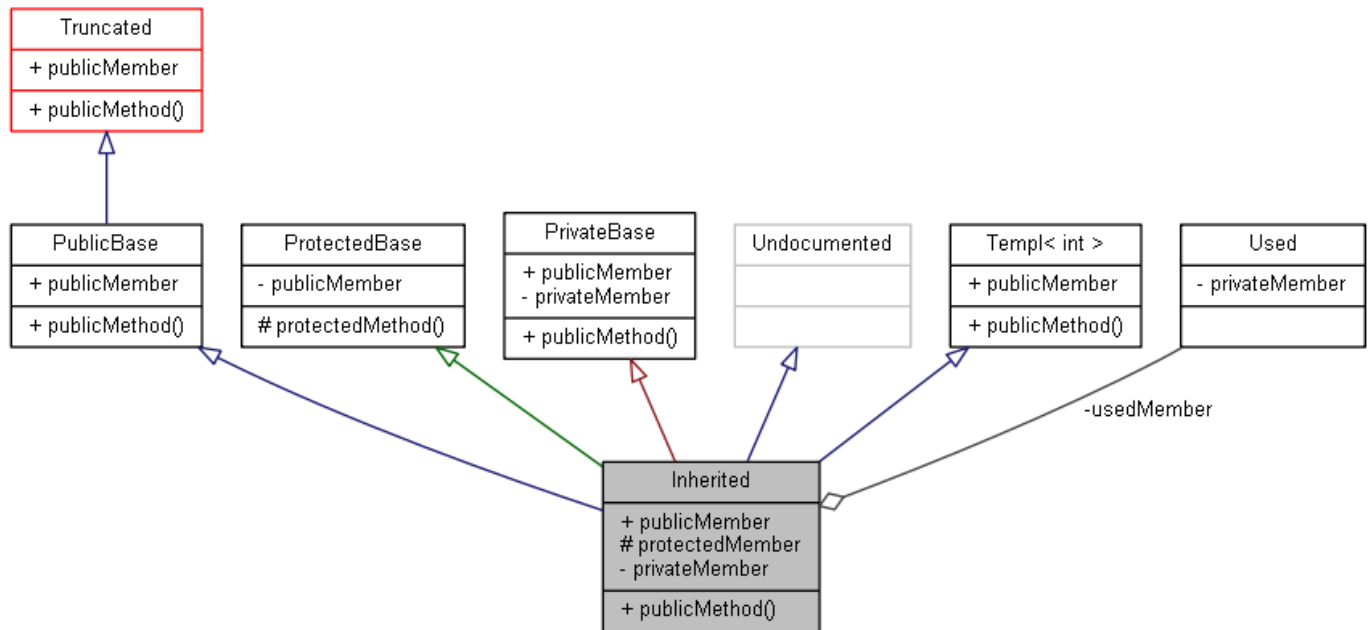
UML и Doxygen

Doxygen позволяет создавать диаграммы «в стиле, схожем с Unified Modeling Language». Разработчики очень аккуратно выражаются в этом плане, поскольку эта возможность не подразумевает то, что на основе кода будет построена аккуратная и грамотная UML-схема, для этой задачи следует использовать другие инструменты, да и они зачастую дают сбои, поскольку в UML в первую очередь выражается идея, пусть и порой достаточно близкая к конкретной реализации.

Итак, для того чтобы строить диаграммы в UML стиле, необходимо установить следующую опцию:

```
UML_LOOK = YES
```

Вновь обратимся к используемому ранее примеру, только дополним его открытыми, закрытыми и защищенными атрибутами и методами:



К сожалению, Doxygen зачастую не в состоянии отразить многие более сложные концепции, хотя бы просто потому, что нельзя указать, какие конкретно классы необходимо включить в диаграмму. Поэтому в тех случаях, когда необходимо проиллюстрировать, например, примененный шаблон проектирования или какое-то иное решение, иногда проще и эффективнее построить схему в ином редакторе и просто импортировать её. Благо, что для одного из очень распространенных специализированных редакторов PlantUML у Doxygen есть встроенная поддержка (для его использования необходимо указать путь к в опции).

В первую очередь необходимо убедиться в том, что Java установлена на вашем компьютере и корректно настроена и затем указать корректный путь к *jar* файлу при помощи соответствующей опции:

```
PLANTUML_JAR_PATH = путь_к_jar_файлу
```

После этого для вставки диаграммы PlantUML необходимо использовать соответствующие команды `\startuml` и `\enduml`:

```
\startuml [{file}] ["caption"] [<sizeindication>=<size>]
...
\enduml
```

Все параметры в данном случае необязательны. Параметр *file* указывает имя для сгенерированного файла, если его не указать, то оно будет выбрано автоматически; параметр *caption* устанавливает подпись к схеме; последний параметр используется для

указания размера схемы.

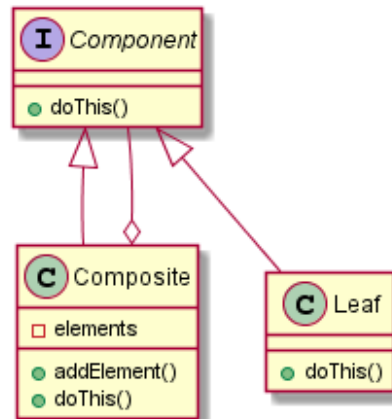
Теперь перейдём непосредственно к примеру (далее изображена диаграмма для шаблона проектирования «компоновщик»):

```
/*! \file
Компоновщик (англ. Composite pattern) – структурный шаблон проектирования, объединя
\startuml
    interface Component {
        +doThis()
    }
    class Composite {
        -elements
        +addElement()
        +doThis()
    }
    class Leaf {
        +doThis()
    }
    Component <|-- Leaf
    Component <|-- Composite
    Composite o-- Component
\enduml
*/
```

Результат представлен ниже:

Подробное описание

Компоновщик (англ. Composite pattern) — структурный шаблон проектирования, объединяющий объекты в древовидную структуру для представления иерархии от частного к целому. Компоновщик позволяет клиентам обращаться к отдельным объектам и к группам объектов одинаково. Ниже представлена иллюстрация данного шаблона при помощи UML:



Заключение

Наконец, следует отметить, что схемы можно строить в любом предпочитаемом вами редакторе или при помощи других генераторов, а затем результат импортировать в Doxygen. Вообще, всё зависит от ваших требований и тех задач, которые стоят перед вами; в некоторых случаях тех возможностей, которые предоставляет вам Doxygen, хватит с головой, и тогда я, надеюсь, эта статья окажется для вас приятной находкой; а иногда их может оказаться недостаточно и тогда потребуется применение других инструментов и других решений.

Спасибо за внимание!

Теги: документация, документация кода, doxygen, диаграммы, графы

Хабы: Программирование, C++, C, C#

Редакторский дайджест

Присылаем лучшие статьи раз в месяц

Электронная почта



52

Карма

0

Рейтинг

@Norserium


Пользователь

Подписаться




Github

HYBRID.AI

 **РУВИКИ**

ЕГЭ/ОГЭ
готовься с РУВИКИ




Мы собрали для тебя
1451 статью для подготовки
к экзаменам

из ru.wikipedia.ru, АНО «Интернет-энциклопедия «Рувикс». ИНН 9714009164

Комментарии 2




Публикации

ЛУЧШИЕ ЗА СУТКИ ПОХОЖИЕ


-  Cloud4U


18 часов назад



Как запустить Windows 95 на одноразовом вейпе

 Простой  15 мин  8.2K

Тutorial Перевод




 +79

 30


 25
-  pokrovsk


19 часов назад



Как я тестировал российские фоторедакторы, полжизни проработав в Фотошопе

 Простой  9 мин  15K

Обзор




 +55

 32

 75
-  sendelust

18 часов назад

Как начать писать на Java в VSCode

 Простой  11 мин  5.1K

[Тutorial](#) +51 120 34**Stefano**

19 часов назад

Как потреблять API с ограничением по RPS в .NET приложениях

**Простой**

11 мин



2.2K

[Tutorial](#) +42 49 1**ru_vds**

15 часов назад

Нельзя предполагать, что все используют UTF-8

**Средний**

6 мин



4K

[Мнение](#)[Перевод](#) +34 18 25**S1oNN**

22 часа назад

Как Яндекс создал свою шину данных, чтобы передавать сотни гигабайт в секунду

**Простой**

7 мин



11K

[Рoadмэп](#) +33 50 19**techno_mot**

18 часов назад

Шаг за шагом: разработка 3D-игры в Godot 4.2 для начинающих



7 мин



2.4K

 +32 26 3**pkolt**

21 час назад

Метеостанция на ионисторе

 Простой  10 мин  4.9K

Из песочницы

 +31

 40

 24



MaFrance351

20 часов назад

Оживляем раритетный домофон с магнитным ключом

 Простой  9 мин  2.3K

Обзор

 +29

 9

 8



aso1ovye24

17 часов назад

Удаленка. Мой путь к выгоранию (и обратно)

 7 мин  5.8K

 +21

 33

 29

Проблемы со связью и боязнь облаков: а при чём тут DevOps?

Турбо

Показать еще

МИНУТОЧКУ ВНИМАНИЯ



Гиперскейлер в коробке и другие новинки с конференции GoCloud



IT-события, которые ты ищешь, тоже ищут тебя



Отечественный DevOps: экспедиция в поисках йети

ВОПРОСЫ И ОТВЕТЫ

Как удобно вести матрицу межсерверных взаимодействий?

Документация · Средний · 0 ответов

Документация по процессору Intel Pentium 4 socket 478?

Документация · Средний · 1 ответ

Документация по процессору Intel 4004?

Документация · Средний · 1 ответ

Документация инфраструктура, как?

Документация · Простой · 2 ответа

Где найти документацию и мануал для IP-камеры?

Документация · Простой · 4 ответа

Больше вопросов на Хабр Q&A



ЧИТАЮТ СЕЙЧАС

Попросил нейросети собрать игровой ПК за 100 000 рублей. Вот что из этого получилось

143K 122

Как я тестировал российские фоторедакторы, полжизни проработав в Фотошопе

15K 75

В Steam стартовал первый фестиваль российских игр

3K 6

Число пользователей Tog в РФ снизилось двукратно

11K 21

Как Яндекс создал свою шину данных, чтобы передавать сотни гигабайт в секунду

 11K

 19

Проблемы со связью и боязнь облаков: а при чём тут DevOps?

Турбо

ИСТОРИИ



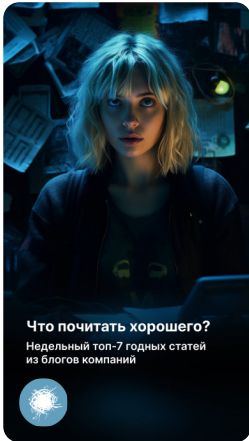
Как приходят идеи крутых статей



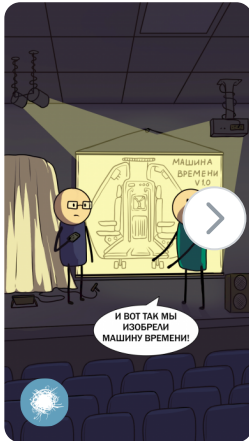
Зовём участвовать в UX-тестах



Активность найма в 1 квартале 2024



Годнота из блогов компаний



Как продвинуть машину времени?

РАБОТА

Программист C++
125 вакансий

QT разработчик
9 вакансий

Программист C# удаленно
100 вакансий

Программист C
38 вакансий

Все вакансии

БЛИЖАЙШИЕ СОБЫТИЯ

Открытый вебинар

oTus

Go-to-market стратегия:
как вывести IT-продукт
на рынок

Выход
на рынок

Онлайн-курс Product Marketing Manager в IT

Вебинар «Go-to-market
стратегия: как вывести IT-
продукт на рынок»

📅 13 мая

🕒 20:00

📍 Онлайн

Подробнее в календаре

Открытый вебинар

oTus

MySQL: оптимизация
производительности

Онлайн-курс Базы данных

Вебинар «MySQL:
оптимизация
производительности»

📅 14 мая

🕒 19:00

📍 Онлайн

Подробнее в календаре

Открытый вебинар

oTus

Конфигурация
PostgreSQL

Онлайн-курс PostgreSQL
баз данных и разработки

Практический у
конфигурации PostgreSQL

📅 14 мая

🕒 20:00

📍 Онлайн

Подробнее в календаре



Ваш аккаунт	Разделы	Информация	Услуги
Войти	Статьи	Устройство сайта	Корпоративный блог
Регистрация	Новости	Для авторов	Медийная реклама
	Хабы	Для компаний	Нативные проекты
	Компании	Документы	Образовательные
	Авторы	Соглашение	программы
	Песочница	Конфиденциальность	Стартапам



Настройка языка

Техническая поддержка

© 2006–2024, Habr