Как использовать doxygen.

Как использовать doxygen. 1
Как этим пользоваться. 1
В коде....1
Как сгенерировать chm?....1
Вы только что создали файл с документацией.....1
Вы хотите обновить документацию для файла, у которого она уже есть, но устарела. 2
Замечания по оформлению документации. 2
Документирование классов. 2
Документирование функций. 2
Документирование модулей.....3
Разные штуки 3
И еще....4

Doxygen – мощное средство документирования кода. Ведь гораздо удобнее, когда комментарии к коду лежат рядом с кодом – когда меняется код, с гораздо большей вероятностью комментарий тоже поменяется, если будет на виду во время изменения.

Doxygen позволяет из обычных сишных комментариев, записанных в специальном формате, генерировать документы chm (как вариант – HTML, LaTeX, rtf, pdf) с ссылками, диаграммами классов и прочими удобными наворотами.

Как этим пользоваться.

В коде

Комментарий в специальном формате. Существуют три варианта

```
/**
комментарий
*/
```

либо

```
/// комментарий
```

либо

```
//! комментарий
```

Все, что написано внутри комментариев, оформленных таким образом, выносится в chm. Для единообразия лучше использовать /// для коротких комментариев и /** */ для длинных

Кроме этого, есть некоторое количество служебных слов (они начинаются с символа @: `@main`, `@code` и др.), которые задают правила отображения документации (выделение в отдельный раздел, специальное форматирование и шрифт для выделения примера кода, вынесение текста на заглавную страницу и т.п.)

Как сгенерировать chm?

Вы только что создали файл с документацией.

1. Когда вы только создали файл с документацией, надо создать файл – хранилище опций. Для этого
   А) либо скопировать из уже существующего, например в CVS: CB5Comp/ThumbnailList/thumb_component
   Б). либо в командной строке:
   ```
   doxygen -g
   ```
   генерируется файл под названием Doxyfile (без расширения, текстовый, читать и редактировать можно в Far Manager либо Windows Commander по кнопочке F4). В этом файле хранятся опции – можно настроить имя chm-файла, папку куда будет складываться документация, генерацию

документации в формат LaTeX и кучу всего другого. Этот файл можно редактировать руками либо специальной утилитой doxywizard.

2. Если надо, отредактировать Doxyfile
   А) либо F4
   Б) либо утилитой с графическим интерфейсом – для этого в командной строке
   ```
   doxywizard Doxyfile
   ```
3. После этого запустить создание документации
   А) либо из командной строки командой:
   ```
   doxygen
   ```
   Б) либо утилитой с графическим интерфейсом по кнопочке, похожей на шестеренку – для этого в командной строке
   ```
   doxywizard Doxyfile
   ```

Вы хотите обновить документацию для файла, у которого она уже есть, но устарела.

В командной строке (в том каталоге, где лежит doxyfile):
```
doxygen
```

Замечания по оформлению документации.

Документировать следует классы, члены классов и функции со всеми параметрами. Кроме того, для отдельных модулей в целом следует документировать, как ими пользоваться.

Документирование классов.

```
/**
@class MyClass
Тестовый класс для примера документирования кода. Ничего особенного не
умеет, содержит конструктор, один член данных и один метод. Ваши классы
следует документировать подобным образом.
*/
class MyClass : public OurClass
{
public:

  /// Конструктор по умолчанию
  MyClass();

  /// Метод, делающий что-то. Он делает это таким-то образом.
  /// @param  in_iParameter1 – параметр, отвечающий за что-то
  /// @param  out_sParameter2 - параметр, отвечающий за что-то еще
  /// @return true если все хорошо, false если все плохо
  bool DoSmthng(int in_iParameter1, std::string out_sParameter2);

private:

  /// этот член хранит в себе какую-то информацию
  int m_iParameter1;

  /// этот член тоже хранит в себе какую-то информацию
  std::string m_sParameter2;
}
```

Документирование функций.

```
/**
@function MyFunction
Тестовая функция для примера документирования кода. Ничего особенного не
умеет, имеет два входных параметра. Ваши функции следует документировать
подобным образом.
```

```
 @param  in_iParameter1 – параметр, отвечающий за что-то
 @param  out_sParameter2 - параметр, отвечающий за что-то еще
 @return true если все хорошо, false если все плохо
 */
 bool MyFunction (int in_iParameter1, std::string out_sParameter2);
```

Документирование модулей.

В любом месте кода

```
 /**
 @mainpage Этот модуль следует использовать следующим образом:
сначала его нужно собирать в VisualStudio 6.0, а потом полученный dll
скопировать из Bin/BCB/ в папку с .exe
 */
```

Разные штуки

Текст на главной странице - @mainpage
Создание новой страницы - @page идентификатор название
    Например @page pagereq Требования к системе
Создание секции - @section идентификатор название
    Например @section common Общие требования
Вставка примера кода - @code
    Этот тэг парный, его закрывает @endcode
Краткое описание - @brief
    Можно настроить файл опций таким образом, чтобы в краткое описание выносилось первое предложение до точки подробного описания (по умолчанию описание считается подробным). Для этого нужно включить опцию JAVADOC_AUTOBRIEF – в утилите doxywizard она находится на первой странице.
Подробное описание - @full
Автор - @author
Описание к файлу – @file
Предупреждение - @warning
Заметка - @note
Именованная группа - @name имя
    После этого группа в скобках @{ … @}
    Например, таким образом можно группировать методы класса:

```
 /**
 @class MyClass
 */
 class MyClass : public OurClass
 {
 public:
   /// @name Конструктор/Деструктор
   /// @{
   MyClass();
   virual ~MyClass();
   /// @}

   /// @name Установка параметров
   /// @{
   /// Установливает параметр1
   SetParameter1(int);
   /// Возвращает параметр1
   int GetParameter1();
   /// @}

   /// Делает еще что-то
   void DoSmthng();

 private:
```

```
    /// этот член хранит в себе какую-то информацию
    int m_iParameter1;
}
```

И еще.

Doxygen создает документацию в виде html, и потом жмет ее в chm, поэтому для форматирования текста можно применять тэги html. Основные приведу сюда:

Название	Тэг	вид
списки	1. элемент списка – * - элемент * (можно не закрывать тэгом)	• элемент
	2. ненумерованные (unordered list) * * * первый пункт* * второй пункт* * третий пункт* * *	• первый пункт • второй пункт • третий пункт
	3. нумерованные (ordered list) * * * первый пункт* * второй пункт* * третий пункт* * *	1. первый пункт 2. второй пункт 3. третий пункт
Пре-форматирование	*<pre>* *тут форматирование* *и тут* *и тут* *</pre>* *а тут уже нет форматирования* *и тут нет* (обратите внимание, что переносы и табуляции внутри <pre> сохраняются, а вне - нет)	тут форматирование и тут и тут а тут уже нет форматирования и тут нет
bold	*this is bold*	**this is bold**
italic	*this is italic*	*this is italic*
ссылка	*The used algorithm idea is described here *	The used algorithm idea is described here
Перевод строки	*Line Line Line*	Line Line Line