

Автодокументация исходников посредством PasDoc

Иван Шихалев

<http://freepascal.ru/>

1 марта 2006 г.

Настоящая статья представляет собой описание основных возможностей утилиты PasDoc. Статья не претендует на полноту, но вполне может использоваться для начального знакомства с программой. Для настоящего освоения PasDoc лучше всего просто взять и поэкспериментировать, вооружившись оригинальной (на английском) Wiki-документацией с официального сайта: <http://pasdoc.sourceforge.net/>.

Содержание

Общие сведения	2
Статус	2
Принцип работы	2
Источники информации	3
Синтаксис комментариев	3
Блоки	3
PasDoc-теги	4
Теги-описания	5
Гиперссылки	6
Форматирование	8
Специальные	9
Расширенные возможности	9
Директивы компилятора	9
Условные директивы	10
{\$INCLUDE . . . }	10
Внешние файлы	10

Общие сведения

Описания	10
HTML-шапка и подвал	11
«Введение» и «Заключение»	11
Прочие вкусности	12
Поиск	12
GraphViz	12
Проверка орфографии	12
Синтаксис командной строки	12
Ключи	12
Заключение	16

Общие сведения

Программа PasDoc предназначена для того, чтобы формировать документацию по модулям Pascal, используя комментарии в исходных текстах. Поддерживается синтаксис как Delphi, так и Free Pascal¹. Сама программа также может быть скомпилирована как тем, так и другим. Впрочем, экспериментальная оболочка GUI должна собираться в Lazarus'e — форм в борландовском формате в исходниках не замечено.

Соответственно, PasDoc может работать на разных платформах — тех, которые поддерживаются FPC (а нас, собственно, они и интересуют).

PasDoc запускается из командной строки и формирует документацию в форматах HTML, HtmlHelp, L^AT_EX и т.д. из исходных файлов модулей. Что касается графической оболочки, на сегодняшний момент она носит статус экспериментальной и останавливаться мы на ней подробно не будем.

Статус

Текущая версия, использованная при написании данной статьи — 0.10.0. В последнее время проект довольно активно развивается, свежую версию см. на SourceForge.

Лицензия — GNU GPLv2.

Сайт — <http://pasdoc.sourceforge.net/>.

Принцип работы

PasDoc обрабатывает исходные тексты модулей, выполняя синтаксический разбор — в результате формируется общая структура документации, и разбирая

¹Различия в синтаксисе Delphi и FPC с точки зрения PasDoc совсем незначительны и мы их рассматривать не будем.

текст комментариев — из них формируются описания программных элементов. При этом могут разбираться как все комментарии, так и только помеченные специальными *маркерами* (например, “**”), что зачастую удобнее.

Описанием элемента считается *непосредственно предшествующий* ему комментарий, или *непосредственно следующий* за ним и начинающийся с символа “<”. Исключение может быть сделано для полей, свойств и т.д., которые удобнее описывать в комментарии родительского элемента (записи, класса...) используя специальный *pasdoc-тег* “@member”; для значений типа-перечисления — “@value”; а также для параметров подпрограмм — “@param”.

Специальные pasdoc-теги используются также для указания специальной информации (автор, краткое описание...), гиперссылок на другие программные элементы и т.д. Далее мы рассмотрим их подробнее.

Источники информации

Бинарный дистрибутив PasDoc документации не содержит (вероятно, она появится к релизу). Тем не менее, в архиве с исходниками присутствует руководство по программе в форматах HTML и PDF. Впрочем, содержание данного руководства несколько отстает от текущих возможностей программы, как это часто бывает в открытых проектах. Наиболее полную и свежую информацию можно получить на сайте PasDoc.

Синтаксис комментариев

Блоки

Рассмотрим, как PasDoc разбирает комментарии в программе... А разбирает он их *поблочно*. Блок в данном случае — единичный блочный комментарий — “{ ... }” или “(* ... *)”, или же последовательность строчных комментариев — “//”, между которыми нет значащих символов — допустимы переводы строки с пробелами. Если перед элементом расположены два блока комментария, то используется только последний.

Зачастую не все комментарии желательно вносить в описания — могут понадобиться комментарии и сугубо технические. PasDoc предлагает использовать механизм *маркеров* — специальных последовательностей символов, которые выделяют «документирующие» комментарии и позволяют игнорировать прочие. Подробнее см. далее описание параметров командной строки “--staronly” и “--marker”. При использовании маркеров ими следует помечать комментарии, которые должны рассматриваться как описания — вставлять маркер *сразу после* открывающих символов комментария. Например, если в качестве маркера выбрана последовательность “**” (именно это делает параметр “--staronly”), то в описания попадут только комментарии вида: “{** ... }”, “(** ... *)” и “//** ...”.

Синтаксис комментариев

Блок с описанием может располагаться и сразу после объявления идентификатора, но в этом случае следует непосредственно за открывающими символами комментария (и маркером, если таковые используются) поместить символ “<” (знак «меньше»). Следует отметить, что в этом случае строчные комментарии, идущие подряд, блоком уже не считаются.

Пример

```
unit Dummy;

{< Дурацкий модуль.

    Специально для примера — второй абзац описания.
}

interface

type
    (* Совершенно дурацкий тип-запись. *)
    TDummy = record
        A : Integer; //< Поле «А»
        B : Boolean; //< поле «В»
    end;

implementation

end.
```

В примере для простоты не использованы маркеры. Особо следует отметить пустую строчку в первом комментарии. Таким образом в описании разделяются абзацы (в случае выходного формата HTML — тег “<p> ... </p>”).

PasDoc-теги

Теперь о самом «вкусном»: PasDoc использует специальные ключевые слова, именуемые «тегами», для дополнительной разметки внутри комментариев-описаний. Все теги начинаются с символа “@” («собака»), а чтобы поместить сам этот символ в выводимый текст, его следует удвоить — “@@”.

Теги могут иметь параметры, заключаемые в круглые скобки. Как правило, параметр единственный и представляет собой некий текст, трактуемый так же, как и общий текст описания. В некоторых случаях параметр трактуется специальным образом: теги “@link”, “@code”, “@longcode”, “@html” и “@latex”. Кроме того, ряд тегов трактует первое слово как идентификатор, а последующий текст — как описание.

На самом деле, скобки вокруг параметров можно и не использовать: если RasDoc не находит открывающую скобку после тега, который предполагает параметры, то трактует как параметр весь текст от тега и до конца строки (или блока комментария). Понятно, что такая форма записи неприменима к параметрам, содержащим несколько строк, а также к параметрам вышеупомянутых специальных тегов.

Теги-описания

@abstract (описание)

Краткое описание используется в разного рода списках, например, на странице «Все модули» или «Все идентификаторы». В то же время, данное описание попадает и в основное описание элемента — первой строкой.

См. также ключ “--auto-abstract”.

@author (имя <e-mail>)

В основном используется внутри описания модуля, но может применяться и для классов, интерфейсов и object-типов. Очевидным образом содержит информацию об авторе. Например:

```
@author (Вася Пупкин <vasya@pupkin.gov>)
```

@created (дата создания) и **@lastmod** (дата изменения)

Позволяют указать дату создания и последнего изменения документа. Применяются только в тех же случаях, что и “@author”. Какой-либо специальный формат даты соблюдать не требуется.

@cvs (строка cvs)

Позволяет экстрагировать информацию об авторе и датах модификации из комментариев в CVS формате. Например:

```
@cvs ($Date: 2004/04/20 02:01:52 $)
```

@param (идентификатор описание), **@return** (описание),

@returns (описание) и **@raises** (идентификатор описание)

Эти теги используются в описаниях процедур и функций (в том числе — методов). Описывают соответственно параметры, возвращаемое значение и возможные исключения. “@return” и “@returns” — синонимы.

@member (идентификатор описание)

и **@value** (идентификатор описание)

Эти два тега позволяют описывать поля, методы и свойства классов, интерфейсов, объектов и записей, а также значения перечислений внутри описаний соответствующих типов. То есть, вместо:

```
type
{ Описание типа }
MyEnum = (
  { описание Val1 }
  Val1,
  { описание Val2 }
  Val2,
  . . .
);
```

Пишем так:

```
type
{ Описание типа
  @value(Val1 описание Val1)
  @value(Val2 описание Val2)
  . . .
}
MyEnum = (
  Val1,
  Val2,
  . . .
);
```

Что более читаемо внутри самих исходников. Точно также и с тегом “@member”, который применим как к полям, так и к методам и свойствам.

Гиперссылки

Вполне естественно, что PasDoc автоматически генерирует гиперссылки между описаниями программных элементов. Степень автоматизма может быть разной — см. “--auto-link”.

Для начала рассмотрим явные ссылки:

@link(идентификатор текст)

Вставляет непосредственную ссылку на описание идентификатора. Замещающий текст может отсутствовать, при этом ссылка будет представлять собой сам идентификатор, возможно — составной (типа «UnitName.Identifier») — в зависимости от состояния ключа “--link-look”.

@seealso(идентификатор текст)

Аналогично предыдущему создает ссылку на соответствующий элемент, но не в месте своего появления, а в отдельной секции «See also» (или «См. также») — аналогично, например, тегу “@raises”.

Если включен ключ “--auto-link”, PasDoc будет генерировать гиперссылки везде, где ему встретится объявленный идентификатор, что позволяет избежать явного указания “@link”, но не всегда удобно — идентификатор может совпадать с каким-то общеупотребимым словом, которое встречается в тексте само по себе². Чтобы и автоссылками воспользоваться и лишних ссылок избежать, следует обратить внимание на следующие теги, управляющими режимом автоматических ссылок.

@noAutoLink (ТЕКСТ)

Данный тег запрещает использовать автоматическое определение идентификаторов в тексте параметра. Внутри тега ссылки можно указать лишь явно.

@noAutoLinkHere

Здесь «here» следует понимать как «сюда», а не «здесь». Этот тег следует помещать в описание того элемента, на который не должно быть автоматических ссылок (т.е. объекта с идентификатором типа “Add”, “This” и т.д.). В этом случае данный идентификатор не определяется при автоматическом распознавании, но на него можно по прежнему сослаться явным образом — тегом “@link”.

Отдельно хотелось бы отметить ряд контекстно-зависимых ссылок — это теги без параметров, которые в тексте заменяются значениями (и ссылками), зависящими от того элемента, в описании которого они были употреблены.

@name и @classname

Эти теги не создают гиперссылки, а всего лишь вставляют вместо себя идентификатор текущего элемента и класса (интерфейса, записи и т.д.), к которому он принадлежит.

@inherited и @inheritedClass

Соответственно вставляют идентификатор (с гиперссылкой) унаследованного элемента и класса-предка. В описании класса тег “@inherited” действует так же, как “@inheritedClass”; в описании элемента он вставляет полный идентификатор перекрытого элемента (вида «ClassName.Item-Name») с соответствующей ссылкой.

Пример:

```
{ @name — это метод @classname, который перекрывает @inherited,  
  чтобы кое-что сделать иначе... }
```

²Само собой, это относится к комментариям на языках, использующих латиницу.

Форматирование

Следующие теги предназначены для форматирования и оформления описаний, генерируемых программой PasDoc.

@bold(текст) и **@italic(текст)**

Выделяют текст **полужирным** шрифтом и *курсивом* соответственно. Параметр этих тегов не может содержать конец абзаца — пустую строку. Теги могут быть вложенными.

@code(текст)

Данный тег форматировывает текст-параметра как фрагмент кода внутри строки — в случае выходного формата HTML это будет соответственно тег “<code> ... </code>”. Внутри тега нельзя использовать пустые строки, зато можно использовать прочие PasDoc-теги.

@preformatted(текст)

Сохраняет разбивку текста-параметра, а также все специальные символы внутри. В частности, внутри параметра данного тега все прочие теги PasDoc будут выведены, как обычные слова.

В параметре могут встречаться пустые строки — в результат они попадут сами по себе, а не как конец абзаца. При этом следует помнить об открывающих и закрывающих скобках — если не соблюден баланс, тег, как и любой другой, не будет закрыт или будет закрыт досрочно.

@longcode(маркер текст маркер)

Данный тег помимо того, что сохраняет все переводы строк и пробелы, как и предыдущий, еще и осуществляет подстветку ключевых слов и других синтаксических конструкций языка Паскаль. Подсветка может быть настроена посредством стилевого файла (для HTML).

Маркер в начале и в конце тега нужен для совместимости с предыдущими версиями. Им может быть любой символ, кроме скобок, но в документации рекомендовано использовать символ “#”. Данный маркер никакого отношения к маркеру PasDoc-комментариев (см. “--marker”) не имеет.

Несколько особняком стоят теги, вставляющие особым образом форматированные конструкции и специальные символы.

Тире и @-

PasDoc для вставки тире использует конструкции, позаимствованные из L^AT_EX: длинное тире («—» — “—”) — “---”; короткое тире («-» — “–”) — “--”; одиночный дефис “-” дефис («-») и обозначает. Если идущие подряд дефисы не следует превращать в тире, используется запись “@-”, которая всегда обозначает дефис, независимо от контекста.

@br

Тег вставляет принудительный разрыв строки.

@nil, @true и @false

Вставляют форматированные “Nil”, “True” и “False” соответственно.

Кроме описанного простого форматирования PasDoc позволяет создавать различные списки (упорядоченные, неупорядоченные и определенных), а также таблицы. Возможности их форматирования обширны и выходят за рамки обзорной статьи. Интересующиеся могут обратиться к документации на <http://pasdoc.sourceforge.net/>.

Специальные

Нижеприведенные теги используются для специальных целей и трудно поддаются классификации.

@deprecated

Помечает элемент, в описании которого используется, как нерекомендуемый³.

@exclude

Исключает элемент из генерируемой документации.

@html (*html-код*) и **@latex** (*latex-код*)

Эти два тега позволяют вставить некий код непосредственно в HTML-или L^AT_EX-вывод программы. Таким образом можно использовать произвольные конструкции, например — вставлять ссылки на внешние ресурсы. См. ключ командной строки “--format”.

Расширенные возможности

Директивы компилятора

Собственно, нас интересуют не все директивы, а только те, которые способны повлиять на состав документации. Это, во-первых, директивы условной компиляции, и во-вторых, директивы включения файла (`{ $INCLUDE ... }`).

³В настоящее время (версия 0.10.0) данная пометка не локализована.

Условные директивы

PasDoc поддерживает директивы условной компиляции частично — на том уровне, который был доступен еще в Turbo Pascal — `{IFDEF ...}`. Условия типа `{IFORT ...}` или `{IF ...}` с выражением пока не поддерживаются. Для управления символами условной компиляции используются ключи командной строки `--define` и `--conditionals`: первый позволяет определить символы непосредственно в командной строке, а второй — использовать файл, в котором они перечислены.

`{INCLUDE ...}`

PasDoc поддерживает включения файлов директивой `{INCLUDE ...}`, поиск файлов производится в каталоге исходников и каталогах, указанных ключом `--include`.

Внешние файлы

Описания

Описания программных элементов могут быть размещены не только в комментариях внутри исходников, но и во внешнем файле. Формат файла следующий:

```
#Item1
Описание Item1
#Item2
Описание Item2
. . . . .
```

Идентификаторы `Item1`, `Item2` и т.д. должны быть в полной форме — с указанием через точку модуля, класса и проч. (вида `#UnitName.ClassName.MethodName`). В тексте описаний — строках которые не начинаются с `#` — можно использовать PasDoc-теги точно так же, как и в описаниях-комментариях.

Файл описаний подключается параметром командной строки `--description`.

Такие внешние описания можно использовать, например, для генерации документации на нескольких языках, хотя в этом отношении более удачным выбором представляется утилита FPDoc, идущая в составе дистрибутива FPC и поддерживающая синхронизацию, обновление, объединение и т.д. своих файлов описаний (но не поддерживающая получение описаний из комментариев, и существенно более сложная в использовании).

HTML-шапка и подвал

Если выходной формат — HTML или HtmlHelp, в начало и конец каждой страницы можно добавить определенные фрагменты (с копирайтом, ссылкой на сайт и чем угодно еще). Данные фрагменты оформляются в виде отдельных файлов и подключаются ключами “--header” и “--footer”. Содержимое этих файлов должно представлять собой фрагменты непосредственно html-кода, которые будут размещены сразу после “<body>” и сразу перед “</body>” соответственно⁴.

«Введение» и «Заключение»

PasDoc позволяет добавить к документации в целом «главы» «Introduction» и «Conclusion» — «Введение» и «Заключение» соответственно. Они представляют собой текстовые файлы, которые подключаются посредством ключей “--introduction” и “--conclusion”. Синтаксис данных файлов тот же, что у блоков комментариев-описаний, за исключением нескольких дополнительных тегов, которые в обычных описаниях встречаться не должны.

@anchor (идентификатор)

Устанавливает метку-«якорь», на которую можно ссылаться из других частей документации по идентификатору (посредством тегов “@link” и “@seealso”). Идентификатор должен соответствовать требованиям языка Паскаль.

@section (уровень идентификатор заголовков)

Создает подраздел введения/заключения (а также подраздел и т.д.). Вложенность раздела определяется числом в первом параметре тега — данный уровень должен быть больше или равен 1. Уровень 1 соответствует “<h2> ... </h2>” в HTML и “\section{...}” в L^AT_EX. Второй параметр — идентификатор — позволяет ссылаться на данный подраздел посредством тега “@link”. Третий параметр — отображаемый заголовок.

@tableOfContents (уровень)

Данный тег вставляет содержание, формируемое тегами “@section” в текст. Необязательный параметр — уровень — позволяет ограничить глубину вложенности элементов содержания. Данный тег работает только при выходном формате HTML.

@title (текст) и @shorttitle (текст)

Первый тег устанавливает заголовок вводной или заключительной части, а второй используется при формате вывода HTML в фрейме навигации.

⁴Строго говоря, фрагмент, вставляемый параметром “--footer”, не всегда размещается непосредственно перед “</body>” — это зависит от наличия «информации о генераторе» — см. ключ “--exclude-generator”.

Синтаксис командной строки

Для того, чтобы сослаться на введение или заключение откуда-то из текста описаний следует использовать тег “@link” с именем файла, содержащего введение/заключение, без расширения.

Прочие вкусности

Поиск

PasDoc позволяет встроить в html-результаты поиск, построенный на «Tipue Search Engine» — JavaScript-движке для локального поиска. Ищет достаточно адекватно (в том числе — на русском), но при больших объемах могут быть и тормоза.

См. ключ “--use-tipue-search”.

GraphViz

GraphViz — это такая программа, которая по текстовым данным, предоставляемым, например, утилитой PasDoc, строит графическое представление некоего графа. PasDoc умеет использовать такое представление для иерархии классов и зависимостей модулей. Подробности см. в документации.

Проверка орфографии

PasDoc позволяет подключить проверку орфографии посредством aspell. Подробности см. в документации.

Синтаксис командной строки

Общий вид командной строки следующий:

```
pasdoc [<опции>] <файлы>
```

Где <файлы> — произвольное количество имен исходных файлов и масок (например, *.pp), а <опции> — комбинация из нижеперечисленных ключей.

Ключи

```
-? --help
```

Показывает краткую справку по параметрам командной строки.

```
--version
```

Показывает версию PasDoc.

Синтаксис командной строки

- v --verbosity=(0-6) [2]
Устанавливает степень подробностей выводимой в процессе информации.
- D --define=<список>
Определяет символы условной компиляции. Символы должны располагаться за ключом через запятую без пробелов.
- R --description=<файл>
Читает описания из файла.
- d --conditionals=<файл>
Читает из файла символы условной компиляции. Каждый символ в файле должен располагаться в отдельной строке.
- I --include=<путь>
Определяет пути поиска включаемых файлов.
- S --source=<файл>
Берет имена исходных файлов из данного указанного.
- html-help-contents=<файл>
Указывает файл .cnt для использования в качестве содержания справки в формате HtmlHelp (.chm). Если данный параметр не указан, то .cnt-файл прекрасно генерируется автоматически.
- F --footer=<файл>
Указывает файл подвала (для выходного формата HTML).
- H --header=<файл>
Указывает файл шапки (для выходного формата HTML).
- N --name=<имя> [docs]
Указывает имя выходного файла без пути и расширения (кроме формата HTML, в котором никакого эффекта не производит).
- T --title=<заголовок>
Указывает заголовок для документации.
- O --format=<HTML|HtmlHelp|LaTeX|LaTeX2RTF> [HTML]
Выходной формат документации. Для форматов, отличных от HTML, PasDoc формирует только исходные файлы, не запуская компилятор (hhc, latex и т.д.).
- E --output=<путь>
Каталог, куда помещаются выходные файлы.

Синтаксис командной строки

-X --exclude-generator

Если этот ключ *не* указан, то PasDoc помещает в конец каждой генерируемой html-страницы информацию о себе:

«Generated by PasDoc <версия> on <дата> <время>».

-L --language=<язык[.кодировка]> [en]

Указывает язык и кодировку документации. Заметим, что данная кодировка относится как к исходным, так и к выходным файлам.

Список всех допустимых значений см. в документации. Отметим лишь: “en” — английский, “ru.1251” — русский в Windows-кодировке, “ru.866” — русский в DOS-кодировке, и “ru.KOI8” — русский в кодировке KOI8-r, распространенной в *nix-системах. UTF-8, к сожалению, пока не поддерживается.

--staronly

Если указан данный ключ, то обрабатываются только комментарии, начинающиеся с “(***”, “{**” или “//**”.

--marker=<маркер>

Если указан данный ключ, то обрабатываются только комментарии, начинающиеся с “(<маркер>”, “{<маркер>” или “//<маркер>”. Маркером может быть произвольная последовательность символов без пробелов.

Таким образом, предыдущий ключ (“--staronly”) — это короткий эквивалент “--marker=**”.

--marker-optional

Употребляется вместе с ключом “--marker” (или “--staronly”). В этом случае обрабатываются все комментарии, но <маркер> в начале комментария игнорируется и не попадает в описание (если просто не указывать ключ “--marker”, то любые символы будут считаться началом собственно текста).

--numericfilenames

Если указан данный ключ, PasDoc генерирует цифровые имена html-файлов документации.

-M --visible-members=<список>

Ключ указывает список типов видимости, которые будут отражены в документации. Список может состоять из следующих значений (через запятые без пробелов): “private”, “protected”, “public”, “published”, “automated”, “implicit”. Последнее — особый тип видимости, введенный PasDoc’ом для специальных целей (см. далее ключ “--implicit-visibility”).

Синтаксис командной строки

По умолчанию в документацию попадают все элементы классов, кроме типов видимости “private” и “implicit”.

`--write-uses-list`

Если указан данный ключ, в описания модулей добавляется перечень используемых.

`--graphviz-uses, --graphviz-classes,`

`--link-gv-uses, --link-gv-classes`

Данные параметры относятся к использованию программы «GraphViz». Подробности см. в документации.

`--abbreviations=<файл>`

Указывает файл аббревиатур. Файл должен содержать строки вида “[name] value”, строки, не начинающиеся с “[”, игнорируются.

`--aspell=<язык>`

Подключает проверку орфографии посредством aspell. В настоящее время работает только с выходным форматом HTML.

`--spell-check-ignore-words=<файл>`

Указывает файл со словами-исключениями для aspell. Файл должен содержать по одному слову на строку.

`--cache-dir=<каталог>`

Каталог кэширования.

`--link-look=<default|full|stripped> [default]`

Данный ключ управляет отображением составных идентификаторов: “default” означает, что ссылка вида “@link(UnitName.Identifier)” будет выглядеть как текст «UnitName.Identifier», ссылающийся на соответствующий элемент; в случае “full” будет «UnitName» со ссылкой на модуль, точка, а затем «Identifier» со ссылкой на элемент; если указать “stripped”, то будет показан только «Identifier».

`--full-link`

Синоним “--link-look=full”.

`--css=<файл>`

Позволяет заменить css-файл, используемый по умолчанию, на любой другой. Файл копируется в каталог вывода.

`--auto-abstract`

Автоматически формирует краткое описание (вместо тега “@abstract”). В качестве такового берется первое предложение описания (то есть — до точки).

Заклучение

`--use-tipue-search`

Использовать JavaScript-поиск в документации. См. выше «Поиск».

`--sort=<список>`

Указывает категории программных элементов, которые в документации будут отсортированы по алфавиту. Неперечисленные войдут в документацию в порядке появления в коде.

Список может состоять из следующих значений: “structures”, “constants”, “functions”, “types”, “variables”, “uses-clauses”, “record-fields”, “non-record-fields”, “methods” и “properties”.

`--introduction=<файл>,`

`--conclusion=<файл>`

Эти ключи указывают файлы введения и заключения соответственно. См. выше «Введение» и «Заклучение».

`--latex-head=<файл>`

Указывает файл, который будет помещен в преамбулу документа при выходном формате L^AT_EX.

`--implicit-visibility=<public|published|implicit> [public]`

Определяет, как будут трактоваться элементы классов, видимость которых явно не указана — то есть тех элементов, которые расположены непосредственно после ключевого слова “class”.

“Public” означает видимость “public”, если не включена RTTI, и “published” в противном случае; “published” указывает трактовать элементы как “published” в любом случае; “implicit” — считать такие элементы относящимися к специальному типу видимости “implicit” — эта возможность используется ключом “--visible-members”.

`--no-macro`

Отключает поддержку макросов FPC.

`--auto-link`

Включает режим автоматических ссылок. См. соответствующие теги.

Заклучение

Общее впечатление от программы PasDoc — более чем благоприятное. Свою основную задачу — формирование документации из комментариев в исходниках — она выполняет вполне успешно.

Пока писалась данная статья, на «PasDoc Wiki» появилась информация о новых тегах, которые будут доступны в следующем релизе — “@image” и “@include”.

Заключение

Первый служит для вставки картинок, а второй, соответственно, для вставки в описание текста из внешнего файла. Поскольку в текущем релизе они еще недоступны, в материал статьи я их решил не включать. В общем — следите за обновлениями на официальном сайте.

Есть некоторые недоработки в русской локализации. . . Напомню, что проект открытый и принять посильное участие может каждый желающий. Полагаю, поддержкой русского языка должно озаботиться именно русскоязычное сообщество. . .

В общем и целом, для быстрого создания документации из откомментированных исходников PasDoc — оптимальный выбор. Если же потребности немного иные — многоязычность, версияльность, более гибкое оформление — есть FPDoc, входящий в состав дистрибутива Free Pascal. О нем я надеюсь написать в ближайшее время.