Using Comments in Programming

In computer programming, a comment is a programming language construct used to embed information in the source code of a computer program. The syntax and rules for comments vary and are usually defined in a programming language specification. In most cases, when the source code is processed by a compiler or interpreter, comments are ignored.

Comments have a wide range of potential uses: from augmenting program code with basic descriptions, to generating external documentation. Comments are also used for integration with source code management systems and other kinds of external programming tools. Use of comments are described below.

Code description

Comments can be used to summarize code or to explain the programmer's intent. According to this school of thought, restating the code in plain English is considered superfluous; the need to re-explain code may be a sign that it is too complex and should be rewritten.

> *"Don't document bad code rewrite it."*

"Good comments don't repeat the code or explain it. They clarify its intent. Comments should explain, at a higher level of abstraction than the code, what you're trying to do."

Comments may also be used to explain why a block of code does not seem to fit conventions or best practices. This is especially true of projects involving very little development time, or in bug fixing.

Algorithmic description

Sometimes source code contains a novel or noteworthy solution to a specific problem. In such cases, comments may contain an explanation of the methodology.

Such explanations may include diagrams and formal mathematical proofs. This may constitute explanation of the code, rather than a clarification of its intent; but others tasked with maintaining the code base may find such explanation crucial. This might especially be true in the case of highly specialized problem domains; or rarely used optimizations, constructs or function-calls.

Resource inclusion

Logos, diagrams, and flowcharts consisting of ASCII art constructions can be inserted into source code formatted as a comment. Additionally, copyright notices can be embedded within source code as comments. Binary data may also be encoded in comments through a process known as binary-to-text encoding, although such practice is uncommon and typically relegated to external resource files.

Debugging

A common developer practice is to comment out a code snippet, so that it will not be executed in the final program. For example, one might write:

```
 if (opt.equals ("e"))
   opt_enabled = true;
 /*
  if (opt.equals ("d"))
   opt_debug = true;
 // */
 //*
 if (opt.equals ("v"))
    opt_verbose = true;
 // */
```

Automatic documentation generation

Developer tools sometimes store documentation and metadata in comments. These may include insert positions for automatic header file inclusion, commands to set the file's syntax highlighting mode, or the file's revision number. These functional control comments are also commonly referred to as annotations.

Keeping documentation within source code comments is considered as one way to simplify the documentation process, as well as increase the chances that the documentation will be kept up to date with changes in the code.

Necessity of comments

Technical commentators have documented varying viewpoints on whether and when comments are appropriate in source code. Some commentators assert that source code should be written with few comments, on the basis that the source code should be self-explanatory. Others suggest code should be extensively commented

In between these views is the assertion that comments are neither beneficial nor harmful by themselves, and what matters is that they are correct and kept in synch with the source code, and omitted if they are superfluous, excessive, difficult to maintain or otherwise unhelpful.

Level of detail

Depending on the intended audience of the code and other considerations, the level of detail and description may vary considerably. For example, the following Java comment would be suitable in an introductory text designed to teach beginning programming:

```
String s = "Wikipedia"; /* Assigns the value "Wikipedia" to the variable s. */
```

This level of detail, however, would not be appropriate in the context of production code, or other situations involving experienced developers. Such rudimentary descriptions are inconsistent with the guideline: "Good comments ... clarify intent." Additionally, for professional coding environments, the level of detail is ordinarily well defined to meet a specific performance requirement defined by business operations.

Offensive comments

Sometimes comments in source code are used as a way to relieve stress or to speak unfavorably about development tools, competitors, employers, working conditions, or even the quality of the code itself. Some commentators deem this highly inappropriate and recommend against including potentially offensive remarks in comments, especially if there is any possibility that the source code may later be viewed by anyone besides the original

developer responsible for it. The occurrence of this phenomenon can be easily seen from online resources that track profanity in source code.

Comments in web templates

Web development presents a special security challenge related to comments, because it is not uncommon for HTML comments to be viewable in plain text by any user of the application. Sections of code that are "commented out" in HTML templates may therefore present security vulnerability.

Styles

There are many stylistic alternatives available when considering how comments should appear in source code. For larger projects involving a team of developers, comment styles are either agreed upon before a project starts, or evolve as a matter of convention or necessity as a project grows. Usually programmers prefer styles that are consistent, non-obstructive, easy to modify, and difficult to break.

The following code fragments in C demonstrate just a tiny example of how comments can vary stylistically, while still conveying the same basic information:

```
/*
    This is the comment body.
    Variation One.
*/

/***************************\
*                           *
* This is the comment body. *
* Variation Two.            *
*                           *
\***************************/
```

Factors such as personal preference, flexibility of programming tools, and other considerations tend to influence the stylistic variants used in source code.

For example, Variation Two might be disfavored among programmers who do not have source code editors that can automate the alignment and visual appearance of text in comments.

Software consultant and technology commentator Allen Holub is one expert who advocates aligning the left edges of comments:

```
 /* This is the style recommended by Holub for C and C++.
  * It is demonstrated in "Enough Rope", in rule 29.
  */

 /* This is another way to do it, also in C.
 ** It is easier to do in editors that do not automatically indent the second
 ** through last lines of the comment one space from the first.
 ** It is also used in Holub's book, in rule 31.
 */
```