PRACTICAL-9

Aim: Design coding standards and guidelines for a given project(C, C++, Java, HTML etc.)

Software Required: None

Knowledge Required: Basics of Software Engineering and Knowledge about coding convention.

Theory:

Coding standards are a set of guidelines, best practices, programming styles and conventions that developers adhere to when writing source code for a project.

Instead of each developer coding in their own preferred style, they will write all the code to the standards outlined in the document.

Importance of Coding Guidelines:

- Coding guidelines are which help in writing the software code efficiently and with minimum errors. These guidelines, known as coding guidelines, are used to implement individual programming language constructs, comments, formatting, and so on. These guidelines, if followed, help in preventing errors, controlling the complexity of the program, and increasing the readability and understandability of the program.
- There are numerous programming language for writing software codes, each having different features and capabilities, coding style guidelines differ from one language to another. However, there are some basic guidelines which are followed in all programming languages. These include naming conventions, commenting conventions, and formatting conventions.

Different Coding Guidelines:

There are certain rules for naming variables, functions and methods in the software code. These naming conventions help software developers in understanding the use of a particular variable or function. The guidelines used to assign a name to any variable, function, and methods are listed below:

- All the variables, functions, and methods should be assigned names that make the code more understandable to the reader. By using meaningful names, the code can be self-explanatory, thus, minimizing the effort of writing comments for variables. For example, if two variables are required to refer to ‘sales tax’ and ‘income tax’, that should bs assigned names such as ‘sales Tax’ and ‘income Tax’.
- For names, a full description in a commonly spoken language (for example, English) should be used. In addition, the use of abbreviations should be avoided. For example, variable names like ‘contact Number’ and ‘addresses should be used of ‘cno’ and ‘add’.

- Short and clear names should be assigned in place of long names. For ‘example, ‘multiply The Two Numbers’ can be shortened to ‘multiply Numbers’ as it is clear and short enough to be expressed in reasonable length.
- As with variables and constants, there are some guidelines that should be followed while naming functions in the software code. These conventions are listed below.
- The names of the functions should be meaningful and should describe the purpose of the function with clarity and briefness, like variables, the names should be self-explanatory so that no additional description about the task of that function is required.
- The function name should begin with a verb. For example, the verb ‘display’ can be used for the function that displays the output on the screen. In case the verb itself is not descriptive, an additional noun or adjective can be used with the verb. For example, the function names ‘add Marks’ should be used to clarify the function and its purpose.
- In case the function returns a Boolean value, the helping verbs ‘is’ and ‘has’ should be used as prefixes for the function name. For example, the function name ‘is Deposited’ and ‘has Deposited’ should be used for functions that return true or false values.

Implementing Coding Guidelines:

- All the codes should be properly commented before being submitted to the review team.
- All curly braces should start from new line.
- All class names should start with abbreviation of each group. For example, CU and PU can be used instead of common user and privileged user, respectively.
- Errors should be mentioned in the following format: [error code]: [explanation]. For example, 0102: null pointer exception, where 0102 indicates the error code and null pointer exception is the name of the error.
- Every if statement should be followed by a curly braces even if there exists only a single statement.
- Every file should contain information about the author of the file, modification date, and version information.

Things to remember while creating a code:

- The developed code should be easy to be read.
- Try to bifurcate different sections of the developed.
- Code by segmenting blocks of code into aa certain paragraph.
- Try to make use of indentation for indicating the beginning and end of the control structures with a clear mentioning of where an exact code between them.
- There should be consistency in the naming convention of the variables throughout the developed code.
- Name the functions based on what they perform.
- They developed code should be easily understood even after returning to it after some time.
- Follow a specific method for commenting on the work.

- The language functions which are complex or the structure which is difficult to be comprehended should be avoided.

Coding Standards:

Proper and consistent indentation is important in producing easy to read and maintainable programs. Indentation should be used to:

- Emphasize the body of a control statement such as a loop or a select statement.
- Emphasize the body of a conditional statement.
- Emphasize a new scope block.
  A minimum of 3 spaces shall be used to indent. Generally, indenting by three or four spaces is considered to be a adequate. Once the programmer chooses the number of spaces to indent by, then it is important that this indentation amount be consistently applied throughout the program. **Tabs shall not be used for indentation purposes.**
  Examples:

```
/* Indentation used in a loop construct. Four spaces used for indentation. */
for (int i = 0 ; i<number_of_employees ; ++i )
{
total_wages += employee [i] wages;
}
// Indentation used in the body of a mentod.
package void get_vehicle_info ()
{
System.out.ptintln("VIN:" + vin);
System.out.ptintln("Make:" + make);
System.out.ptintln("Model:" + model);
System.out.ptintln("Year:" + year);
}
/* Indentation used in a conditional statement. */
IF (IOS .NE> 0)
WRITE(* , 10) IOS
END IF
10 FORMAT("Error opening log file:" ,14)
```

Inline Comments:

Inline comments explaining the functioning of the subroutine or key aspects of the algorithm shall be frequently used. See section 4.0 for guidance on the usage of inline comments.

Structured Programming:

Structured (or modular) programming techniques shall be used. GO TO statements shall not be used as they lead to "spaghetti" code, which is hard to read and maintain, except as outlined in the FORTAN Standards and Guidelines.

Classes, Subroutine, Functions, and Methods

Keep subroutines, functions, and methods reasonably sized. This depends upon the language being used. For guidance on how large to make software modules and methods, see section 4.0. a good rule of thumb for module length is to constrain each module to one function or action (i.e. each module should only do one "thing"). If a module grows too large, it is usually because the programmer is trying to accomplish too many actions at one time.

The names of the classes, subroutines, functions, and methods shall have verbs in them.

That is the names shall specify an action, e.g. "get_name", "compute_temperature".

Source Files:

The names of the source file or script shall represent its function. All of the routines in a file shall have a common purpose.

Variable Names:

Variable shall have mnemonic or meaningful names that convey to a casual observer, the intent of its use. Variables shall be initialized prior to its first use.

Example:

The variable names should be in camel case letters starting with a lower case letter. For example use 'total Amount' instead of 'Total Amount'.

Use of Braces:

In some languages, braces are used to delimit the bodies of conditional statements, control constructs, and blocks of scope of programmers shall use either of the following bracing styles:

```
For (int j = 0; j<max_iterations; ++j)
{
/* Some work I done here. */
}
```

Or the Kernighan and Ritchie style

```
For (int j = 0; j <max_iterations; ++j) {
/* Some work I done here. */
}
```

It is felt that the former brace style is more readable and leads to neater-looking code than the latter style, but either use is acceptable. Whichever style is used, be sure to be consistent throughout the code. When editing code written by another author, adopt the style of bracing used.

Braces shall be used even when there is only one statement in the control block. For example:

Bad:

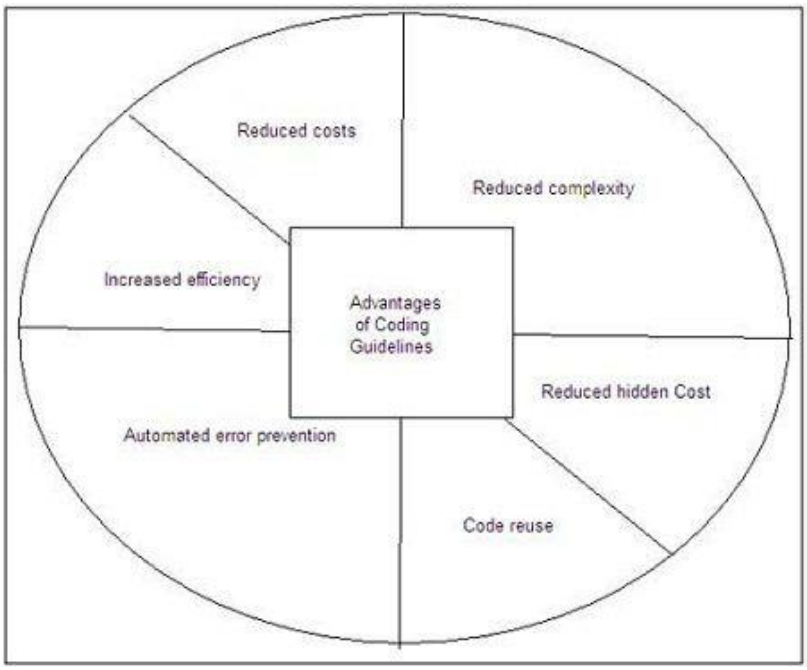
```
if(j==0)
printf("j is zero.\n");
```

Better:

```
if(j==0)
{
printf ("j is zero. \n");
}
```

Advantages of Coding Guidelines:

- Enhanced Efficiency
- Risk of project failure is reduced.
- Easy to maintain
- Bug to rectification
- Cost efficient



Conclusion:

From this practical I have learned how to maintain coding standards and guidelines for given project.