Coding Style and Standards

"Suit the action to the word, the word to the action." -- William Shakespeare

Style refers to structure, form, organization, look-and-feel and layout. Style is more important in poetry than it is in prose. Poetry has greater structure of layout; words are selected very carefully, each line is spaced properly and indentation could be important. Prose, on the other hand, is viewed as one long stream; it is often broken anywhere (including the middle of a word) to begin another line.

Programming is more like writing poetry than prose, except that code usually does not rhyme. Programming style is an art; it does not consist of "sacred" rules, but of guidelines. It takes style to know when to break the guidelines.

Many ways of laying out program code are possible; some are better than others. Proper layout is important for humans to read, understand, analyze, test and maintain code. It reflects quality of the product and professionalism of the programmer.

Style in coding refers not only to guidelines, but also incorporates standards, conventions, and common practices. Standards may seem arbitrary, but they often come from experience, which has shown them to be useful. Even if sometimes they seem arbitrary, the resulting consistency is in itself significant.

The following guidelines are recommended, but not required. They should help you define your own style.

Spacing, especially of the "white areas", is particularly useful for readability. Horizontal spacing, or indentation, shows the levels of structure. Usually 2 to 5 spaces are used; 3 spaces are a very common compromise.

Vertical spacing, or gaps, should show logically separate units; they split parts that should be apart, so joining things that should be together.

Comments should be few, and used only when necessary. They should be brief, describing What is being done rather than How it is being done; the code describes the How.

Names, or identifiers, of boxes, constants, methods, classes, etc should be chosen carefully to reflect the meaning. Names should not normally be too short (such as a, B, z) nor too long; a length of "7 plus or minus 2" is a guideline. Class names should begin with capital letters; boxes (variables) should begin with lower-case letters. Names consisting of compound words (such as `bodyTemperature` or `countOfSheep`) should be written with each "sub-word" beginning with a capital letter.

Names of functions should be nouns (such as `maximum`); routines should be verbs (such as `sort`), boolean queries should be adjectives (such as `isEmpty`, why not `Empty?`). The names of methods (functions, routines) should be separated from the parenthesized list of slots (parameters or arguments) by a space (as in `minimum (a, b)`).

Constants could be written entirely in uppercase (such as `MAX_SIZE`) with subwords separated by underscores. They could also be written less obtrusively with beginning caps (such as `Max_Size`).

Binary operators, and assignment operators, should be separated from operands by a single space (such as `y = m * x + b`, rather than `y=m*x+b`). Also, extra parentheses can help to better group parts making expressions more readable, and less prone to error.