THE ONLY DOCUMENT WITH NAMING CONV FOR ML

Variables

* Variable names should be in mixed case starting with lower case
* Variables with a large scope should have meaningful names. Variables with a small scope can have short names.
* The prefix n should be used for variables representing the number of objects.
* A convention on pluralization should be followed consistently.
* Variables representing a single entity number can be suffixed by No or prefixed by i.
* Negated boolean variable names should be avoided.
* Acronyms, even if normally uppercase, should be mixed or lower case.
* Avoid using a keyword or special value name for a variable name. (reserved)

Constants

* Named constants (including globals) should be all uppercase using underscore to separate words.
* Constants can be prefixed by a common type name.

Structures

* Structure names should begin with a capital letter.
* The name of the structure is implicit, and need not be included in a fieldname.

Functions

* Names of functions should be written in lower case.
* Functions should have meaningful names.
* Functions with a single output can be named for the output.

General

* Names of dimensioned variables and constants should usually have a units suffix.
* Abbreviations in names should be avoided.
* Consider making names pronounceable.
* All names should be written in English

Files and Organization

* Modularize
  + Two screen lengths
* Make interaction clear
* Partitioning, every function should hide something
* Subfunctions
* Format output for easy use.

Statements

* Consider documenting important variables in comments near the start of the file.
* Variables should not be reused unless required by memory limitation.
* Loop variables should be initialized immediately before the loop.
* The use of break and continue in loops should be minimized
* The end lines in nested loops can have comments
* Complex conditional expressions should be avoided. Introduce temporary logical variables instead. By assigning logical variables to expressions, the program gets automatic documentation. The construction will be easier to read and to debug.
  + if (value>=lowerLimit)&(value<=upperLimit)&~ismember(value,… valueArray) : end should be replaced by: isValid = (value >= lowerLimit) & (value <= upperLimit); isNew = ~ismember(value, valueArray);
* The usual case should be put in the if-part and the exception in the else-part of an if else statement.

General

* Use parentheses
* Comments should agree with the code, but do more than just restate the code.
  + A bad or useless comment just gets in the way of the reader. N. Schryer: “If the code and the comments disagree, then both are probably wrong.” It is usually more important for the comment to address why or how rather than what.
* Comments should usually have the same indentation as the statements referred to.
  + This is to avoid having the comments break the layout of the program. End of line comments tend to be cryptic and should be avoided except for constant definitions.

Documentation

* Formal documentation To be useful documentation should include
  + a readable description of what the code is supposed to do (Requirements),
  + how it works (Design),
  + which functions it depends on and how it is used by other code (Interfaces), and
  + how it is tested. For extra credit, the documentation can include
  + a discussion of alternative solutions and suggestions for extensions or maintenance

Java Programming Style Guidelines, Geotechnical Software Services Code Complete, Steve McConnel - Microsoft Press C++ Coding Standard, Todd Hoff