Implementation

Implementation is the process of translating OO design into code.

Why is implementation an issue? Most projects are too large for a single person to implement – <u>team work</u>.

Communication is essential when considering:
- Translation of certain objects into code
- usage of behavior and interfaces
- changes in the code

Choice of programming language:
- consider the risks each language choice represents: compiler availability, staff expertise, problem domain support by the language, etc.
- the language for which the <u>overall</u> risk is the smallest should be selected.

Good Programming Practices

<u>Use of consistent and meaningful variable names</u>
- meaningful from the perspective of a future maintenance programmer
- same concepts in a program should have the same name
- keep the components of a name in the same order, e.g., maxFreq, minFreq, NOT FreqMin.

<u>Documenting code</u>
- it is important that the code is easily understood by all programmers who have to read the code
- prologue comments
  - brief description of what the module does
  - date when created
  - developer name
  - module arguments
  - list of variable name with brief explanations
  - names of files accessed (if any)
  - names of files modified (if any)
  - module input/output
  - module error handling capabilities
- inline comments
  - insert inline comments to assist others in understanding the code
  - insert when code is written in a <u>non-obvious way</u>
  - insert when subtle aspects of the programming language are used

- never code numeric constants into your code
    - always parameterize your program over what you might think are constants, e.g., maximum number of records, tax rate, error returns, etc.
- layout code for increased readability
    - no more than one statement should appear on a line
    - indentation
        - highlight block structure of your code with indentation

          ```
          while (C) {
                  do
                   something
                   really
                   interesting
          }
          ```

    - blank lines
        - code communicates concepts, just as prose, use blank lines to highlight transitions from one concept to another, e.g., from declaration to computation

          ```
          int x;
          int y;
          int z;

          for (int I;…….) {
                   …
          }

          printf("Done!\n");
          ```

- coding standards
    - insure that your code can be read and understood by anyone in the development organization
    - insures that the code for a particular product has some unity – similar concepts, similar expression.