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 – team work.

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 overall risk is the smallest should be selected.

Good Programming Practices

Use of consistent and meaningful variable names
- 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.

Documenting code
- 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 non-obvious way
  - 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

    ```c
    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

    ```c
    int x;
    int y;
    int z;
    for (int I,…….) {
      ...
    }
    ```

    ```c
    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.