Lecture 9.1

Topics

1. A (Simplified) Problem Solving Process in Computer Programming

---

1. A (Simplified) Problem Solving Process in Computer Programming

In general, the program development can be depicted in the following figure.

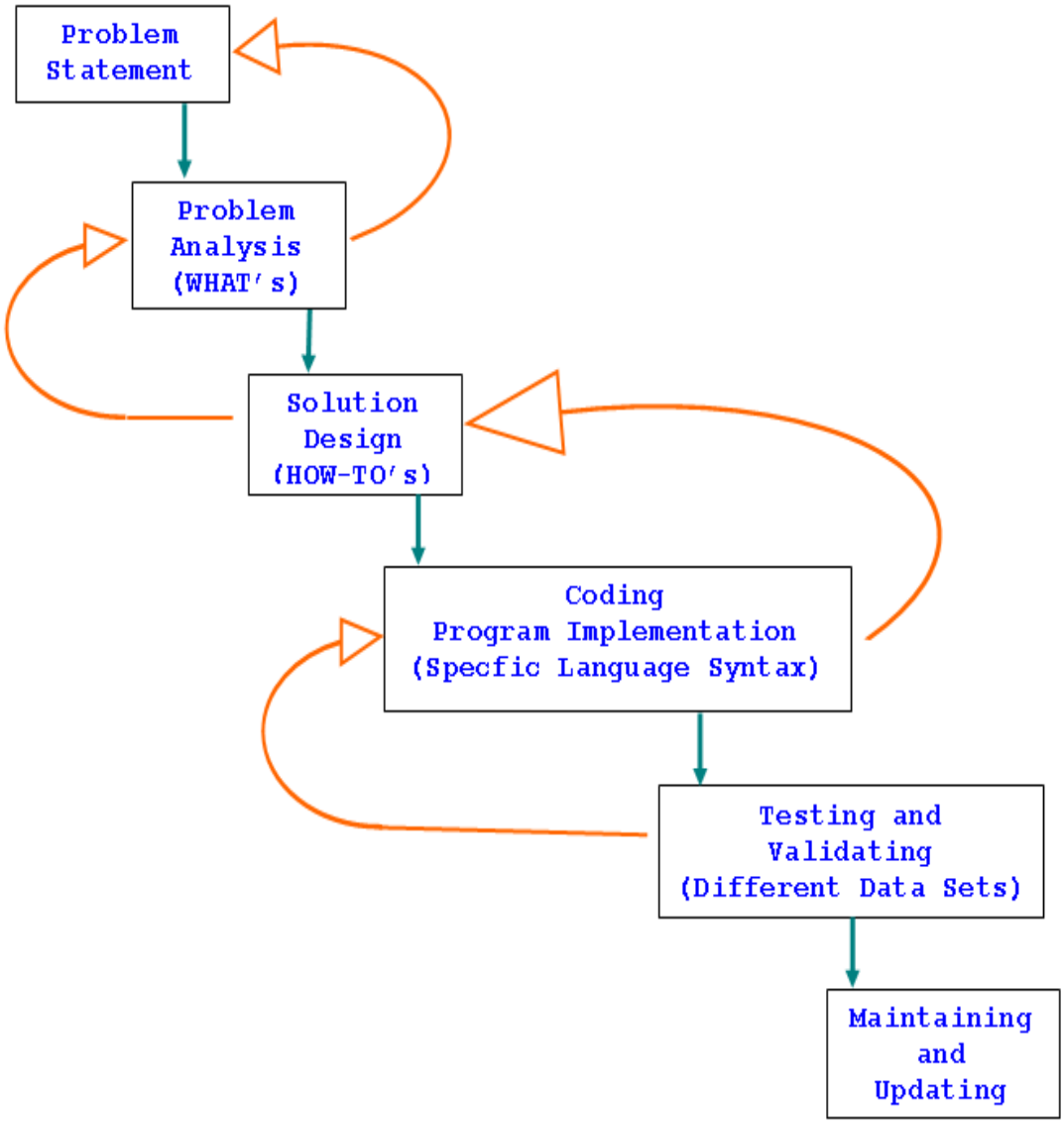

Figure 1 Program Development Process

A simplified version or description (with the first 4 Steps) is used and explained in the following digression.

When working on some given tasks (e.g., doing/coding your assignments), the following steps and points will be identified and considered:

(1) Problem Statement

Each task must be written or presented in a problem statement (from the assignment sheet or rephrasing the task properly). The problem statement may be precise or "a bit" vague based on the complexity of the problem or how it was presented the first time.

(2) Analysis

A rough analysis must be then followed. It must conform to the nature of problem statement, the underlined assumptions and hypotheses.

The analysis will provide the answers for the questions of "what we have/what are the input", "what are required or to have", "what are the assumptions/constraints", and "what are the objectives/output".

(3) Algorithm & Design

The design step will detail HOW-TO do or perform the tasks (i.e., the tasks of "WHAT-NEED-TO-BE-DONE" that are obtained through the analysis step).

A set of logical steps (or simpler TASKS) leading toward the desired solution may also be called the algorithm.

There are several ways to describe a design or an algorithm:

- A flowchart,
- Structure chart (with execution flow for each function), or
- Other functional/object-oriented diagrams (UML).

(4) Implementation

The implementation should have the ***synopsis*** for each function/method/routine – ***What the function does***, and the relevant ***Input and Output***. One might want to specify/indicate the mathematical expressions or techniques being used.

Now, the resulting program should work and produce the results as required. However, in addition to executable code or statements, you may want to provide explanatory comments as needed about some interesting logics next to their implementations.

After all of your efforts, if your program DOES NOT WORK then you must provide some insight and hint on why it does not work. To show a "GOOD" effort, you should propose a possible explanation for fixingto each of the logical steps or bugs of the program.

For my class(es), at the very least, you must be consistent with the coding style when writing code. You will get the hints from me, from books, or from some other resources, and please keep the style consistent throughout the program.

Hint! Read the article on "Programming Convention" that was handed out in class.

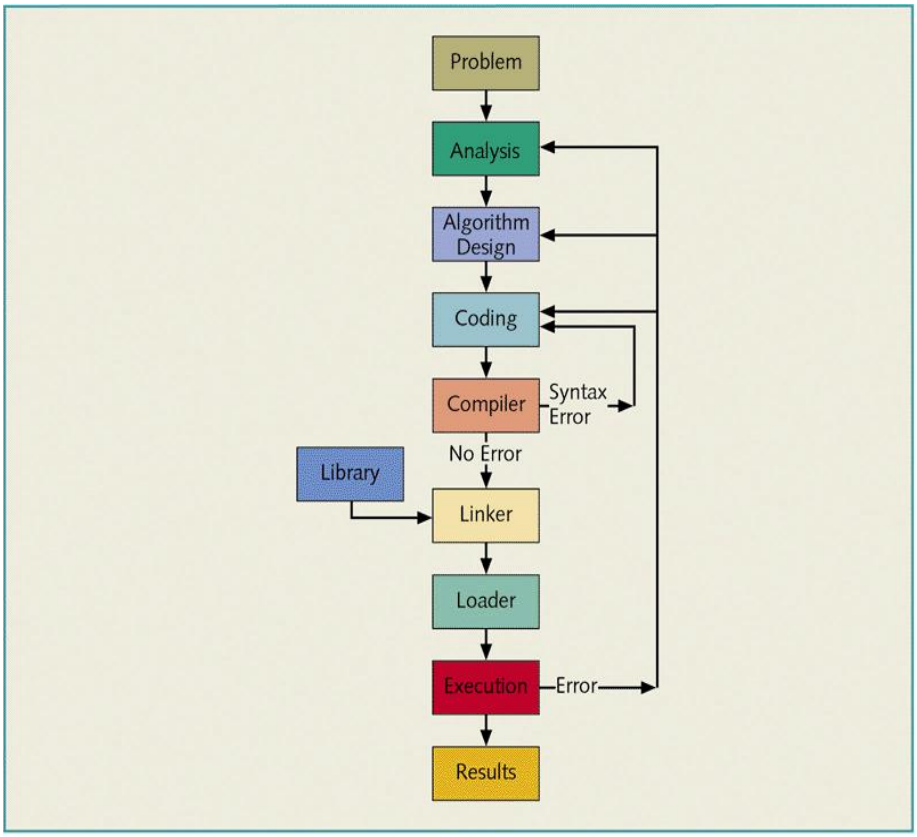

Figure 2 Development process — Problem analysis-design-coding-execution cycle

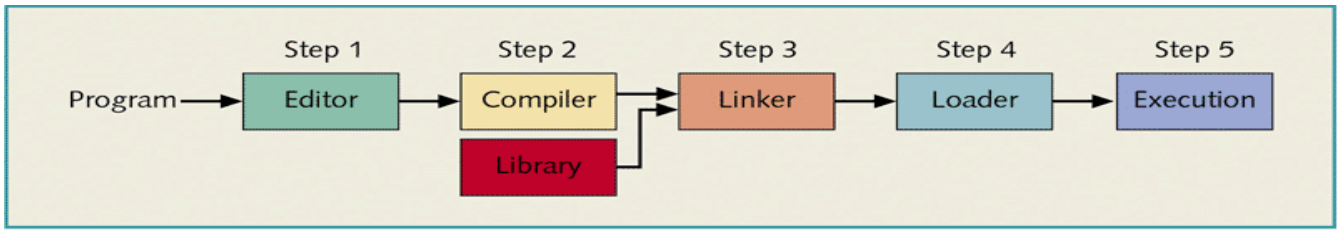

Figure 3 Creating and processing a high-level language program

Example – Finding the Smallest Value

Problem Statement

Write a program to determine the smallest value from a given sequence of inputting values. As the user enters the values, the program will keep track of the (smallest) value entered so far. The program will print this smallest value when the user enters **`eof (ctrl+z)`**.

Analysis:

What do we have?

A sequence of values

What do we want?

The smallest value

What do we assume about the input values?
To be numeric and entered from keyboard

Design:

How to get the smallest value?
Read in first value
Assume this first value is the smallest one

Read in next value
Compare with the previous smallest value
Swap, if necessary, and store new smallest value

Read in another value
Compare with the previous smallest value
Swap, if necessary, and store new smallest value

Keep repeating the process of read/compare/update until `ctrl+z`.