

Types of Programming

Computer program – It is a sequence of statements intended to accomplish a certain task. It is a set of instructions for a computer to follow.

Programming – It is a process of planning and creating a program

Low-level Languages – These are the languages that deal with a computer's hardware components. There are two (2) common low-level languages: machine language and assembly language.

- **Machine Language** – It is the language that the computer can directly understand. It is the most basic set of instructions that the computer can execute. Machine language programs are written in binary codes (0, 1).
- **Assembly Language** – It is a symbolic form of machine language that is easier for people to read, such as *ADD AX DX*. This makes use of instructions in mnemonic form.
 - **Assembler** – a program that translates assembly language instructions into machine language.

High-level Languages – These are the programming languages that use natural languages, such as the English language. A high-level language has its own syntax.

- **Syntax** – rules of the language, for example, *print* or *write* is used to produce an output.
- **Commands** – these are program statements that carry out tasks that the program has to perform, for example, *print this word* or *add these two (2) numbers*.
- **Compiler** – it is a program that translates a program written in a high-level language into a low-level language before executing the program statements.
- **Interpreter** – this acts as a compiler, but it translates one (1) program statement at a time, this executes the statement as soon as it is translated.
- **Syntax errors** – these are errors that might be encountered during the process of translation. An example is a misspelled command.
- **Logical errors** – errors that occur when the syntax of the program is correct, but the expected output is not.
- **Debugging** – the process of locating and correcting the errors of a program.

Programming Cycle

Algorithm – It is a problem-solving technique used in solving programming problems. It is a step-by-step problem-solving process in which a solution is arrived at in a finite amount of time.

The problem-solving process in the programming environment involves the following steps:

1. **Problem Analysis:** Analyze the problem and outline the problem and its solution requirements.
2. **Algorithm Design:** Design an algorithm to solve the problem.
3. **Coding:** Implement the algorithm in a programming language.
4. **Execution:** Verify that the algorithm works.

Figure 1 summarizes the programming process.

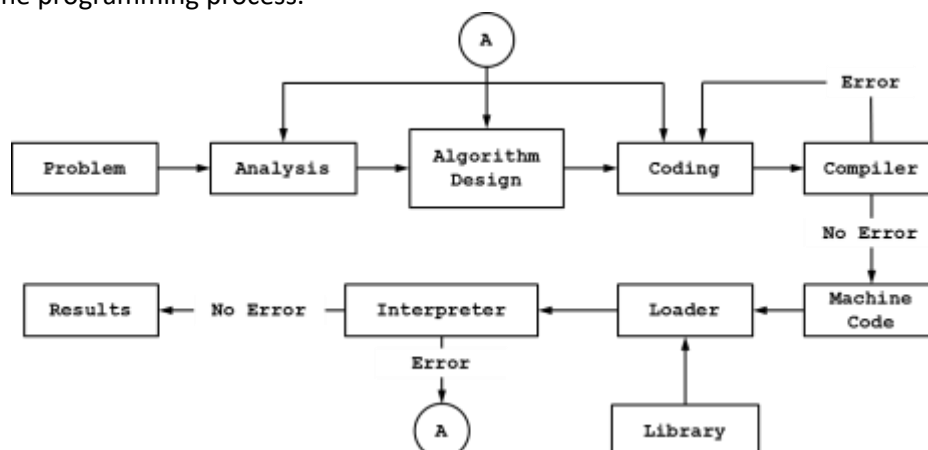


Figure 1. Problem analysis-coding-execution cycle (Malik, 2012)

To develop a program that solves a problem, start first by analyzing the problem, then outlining the problem and the options for a solution. Then design the algorithm, write the program instructions, and enter the program into a computer system.

Algorithm

An algorithm must be expressed completely in a natural language that anyone can follow, such as directions that can be written in the English language. The computer programmer lists down all the steps required to resolve a problem before writing the actual code.

Example:

Design an algorithm that finds and displays the volume of a rectangle. It is required to know the rectangle's length, width, and height, and the formula to know the rectangle's volume. The formula is $\text{volume} = \text{length} \times \text{width} \times \text{height}$.

The algorithm to find and display the volume of the rectangle is:

1. Get the length of the rectangle.
2. Get the width of the rectangle.
3. Get the height of the rectangle.
4. Find the volume using the formula: $\text{volume} = \text{length} \times \text{width} \times \text{height}$.
5. Display the computed volume.

There are two (2) commonly used tools in representing an algorithm:

- Pseudocode
- Flowchart

Pseudocode

Pseudocode – is a method of describing computer algorithms using a combination of natural language and programming language. It is a technique to show the programming steps.

The following are some rules that are frequently followed when writing pseudocode:

- Symbols are used for the following common operations:
 - Arithmetic operations (+, -, *, /)
 - Assignment (=)
 - Comparison (=, ≠, <, >, ≤, ≥)
 - Logical (and, or)
- Certain keywords can be used as a command, such as PRINT, WRITE, READ, SET, GO TO, etc.
- Indentation is used to indicate branches and loops of instructions.

Example:

Using the example problem in the algorithm.

The pseudocode to find and display the volume of the rectangle is:

```

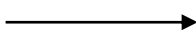
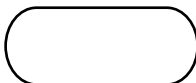
READ length
READ width
READ height
SET volume to 0
COMPUTE volume as length * width * height
PRINT volume
  
```



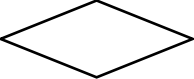


Flowchart

Flowchart – is a visual representation of an algorithm. It contains shapes describing how an algorithm or program operates. Each command is placed in an appropriate shape, and arrows are used to direct program flow.

Flowchart shapes represent various operations. These shapes are connected by directed lines to indicate the flow of data or control from one (1) point to another. *Table 1* shows the often-used shapes in a flowchart.

Table 1: Flowchart shapes and functions

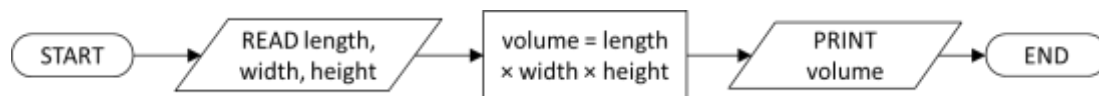
Flowchart Shape	Function
	Flow lines: Used to indicate the direction of the process flow by connecting other shapes. Arrows should not cross each other.
	Terminal block: Used to represent the beginning or end of a program.

Flowchart Shape	Function
	Process: Used to represent a process step or activity, such as computation, initialization, etc.
	Data: This represents the data used as inputs or outputs, such as user input and display text.
	Decision block: This indicates a decision operation in the process, where there are two (2) alternatives: true and false.
	Connector: Used as a connector to combine flow lines by indicating an identifier, such as letters. This is used for complex algorithms.
	Predefined Process: Indicates the use of an algorithm specified outside the program, such as methods or functions.

Example:

Using the example problem in the algorithm

The flowchart to find and display the volume of the rectangle is:



Programming Methodologies

Programming methodology – is the approach to analyzing such complex problems by planning software development and controlling the development process. There are two (2) popular approaches to writing computer programs:

Procedural Programming – in this approach, the problem is broken down into functions that perform one (1) task each. This approach is suitable only for small programs that have a low level of complexity.

Object-Oriented Programming – in this approach, programs are organized around objects rather than actions, and data rather than logic. The solution revolves around entities or objects that are part of the problem. It deals with how to store data related to the entities, how the entities behave, and how they interact with each other to give the desired result.

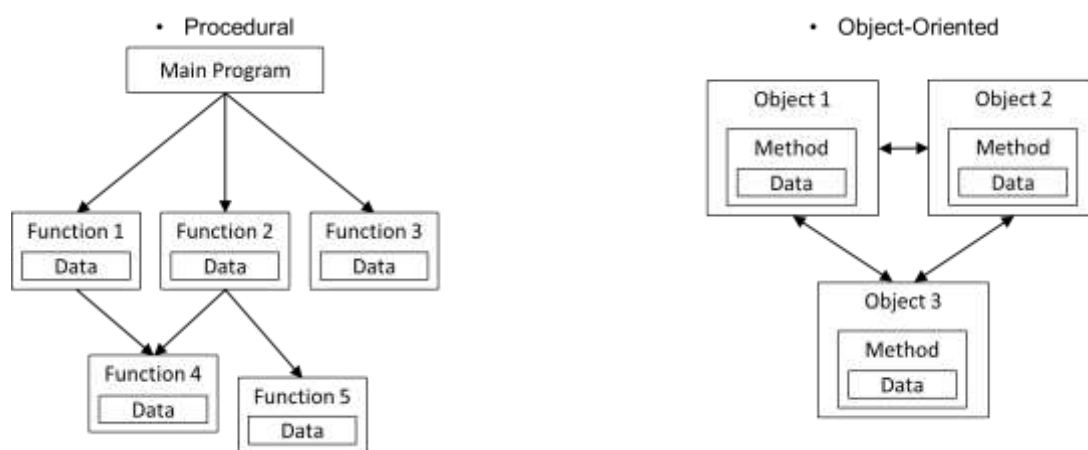


Figure 2. Representation of Procedural and Object-Oriented programming

REFERENCES:

- Baesens, B., Backiel, A., & Broucke, S. (2015). *Beginning java programming: The object-oriented approach*. Indiana: John Wiley & Sons, Inc.
- Farrell, J. (2014). *Java programming, 7th edition*. Boston: Course Technology, Cengage Learning.
- Savitch, W. (2014). *Java: An introduction to problem solving and programming, 7th edition*. California: Pearson Education, Inc.