Evolution of Programming Paradigms

Like the computer hardware, programming languages have been passing through evolutionary phases or generations. This shift in programming paradigm is categorized into the following:

- Monolithic Programming
- Procedural Programming
- Structured Programming
- Object Oriented Programming

Program and data are two basic elements of any computation. Among these, data plays an important role and it can exist without a program. But a program has no relevance without data.

Monolithic Programming:

In monolithic programming whole problem is solved as a single block. All the data are global and there is no security. To share the codes jumps are allowed and so contain lot of goto statements. The complexity of program increases with the size of code and difficult to follow and correct errors and is suitable only for small problem.

Structured programming

Structured programming (sometimes known as modular programming) is a subset of procedural programming. Structured programming frequently employs a top-down design model, in which developers map out the overall program structure into separate subsections. A defined function or set of similar functions is coded in a separate module or submodule, which means that code can be loaded into memory more efficiently and that modules can be reused in other programs. After a module has been tested individually, it is then integrated with other modules into the overall program structure.

Structured programming is a method of writing a computer program that uses

1. Top-down analysis for problem solving,
2. Modularization for program structure and organization, and
3. Structured code for the individual modules.

Top-down analysis

A program is written to tell a computer what to do. But what do you want it to do? What is the job you want it to perform for you? This "job" is more formally called the problem. However, before you can tell the computer what to do, you have to "solve" the problem yourself. In other words, you have to state every step necessary in order to accomplish the job. This activity on your part is called problem solving or problem analysis. For big problems, developing a solution can be very complicated. Where do you start? Top-down analysis is a method of problem solving. It tells you how to start and guides you through the entire process. The essential idea is to subdivide a large problem into several smaller tasks or parts. Top-down analysis, therefore, simplifies or reduces the complexity of the process of problem solving.

Modular Programming

Programs generally require many instructions for the computer. Modular programming is a method of organizing these instructions. Large programs are broken down into separate, smaller sections called modules, subroutines, or subprograms. Each module has a specific job to do and is relatively easy to write. Thus, modular programming simplifies the task of programming by making use of a highly structured organizational plan. There is, of course, a direct correlation between the subdivisions of the problem obtained through a top-down analysis and these

modules: each subdivision will correspond to a module in the program. Modular structure also simplifies programming by greatly reducing the need for the GOTO statement.

Structured Coding

If programs are broken down into modules, into what are modules subdivided? Obviously, each consists of a set of instructions to the computer. But are these instructions organized in any special way? That is, are they grouped and executed in any clearly definable patterns? In structured programming they are. They are organized within various control structures. A control structure represents a unique pattern of execution for a specific set of instructions. It determines the precise order in which that set of instructions is executed.

Each control structure represents a different pattern of execution, but each of these patterns in tum represents one of three basic types of execution. The component statements within a specific control structure are executed, sequentially, conditionally, or repetitively.