Unit: II
Lecture: 3
Modularization

Modularity has become an accepted approach in all engineering disciplines as a modular design reduces complexity, facilitates change (a critical aspect of software maintainability) and results in easier implementation by encouraging parallel development of different parts of a system.

Modularization is a technique to divide a software system into multiple discrete and independent modules which are expected to be capable of carrying out tasks independently. Designers tend to design modules such that they can be executed and/or compiled separately and independently.

Advantages of modularization:

- Smaller components are easier to maintain
- Program can be divided based on functional aspects
- Desired level of abstraction can be brought in the program
- Concurrent execution can be made possible.

Functional Independence:

Functional independence is achieved by developing modules with "single-minded" function and with a tendency to very less interaction with other modules. In other words, a software is designed in such a way that each module addresses a specific subfunction of requirements and has a simple interface when viewed from other parts of the program structure.

- Software with effective modularity i.e. independent modules is easier to develop because function may be compartmentalized and interfaces are simplified.
- Independent modules are easier to maintain (and test) because secondary effects caused by design or code modification are limited, error propagation is reduced and reusable modules are possible.

Functional independence is a key to good design and design is the key to software quality.

Independence is measured using two qualititative criteria:

- **cohesion**
  - is a measure of the relative functional strength of a module.
- **coupling**
  - is a measure of the relative interdependence among modules.

Cohesion:

Cohesion is a measure that defines the **degree of intra-dependability within elements** of a module.

The greater the cohesion, the better is the program design → we always strive for high cohesion.

Coupling:

Coupling is a measure that defines the **level of inter-dependability among modules of a program**. It tells at what level the modules interfere and interact with each other.

The lower the coupling, the better the program. → we strive for lowest possible coupling.

There are five levels of coupling:

- **Content Coupling:** When a module can directly access or modify or refer to the content of another module, it is called content level coupling.

- **Common Coupling:** When multiple modules have read and write access to some global data, it is called common or global coupling.

- **Control Coupling:** Two modules are called control-coupled if one of them decides the function of the other module or changes its flow of execution.

- **Stamp Coupling:** When multiple modules share common data structure and work on different part of it, it is called stamp coupling.

- **Data Coupling:** Data coupling is when two modules interact with each other by means of passing data (as parameter). If a module passes data structure as parameter, then the receiving module should use all its components.