UNIT 2
PROGRAMMING TECHNIQUE

LH – 5HRS

PRESENTED BY:

ER. SHARAT MAHARJAN

C PROGRAMMING

CONTENTS (LH – 5HRS)

2.1 Introduction to Programming Technique,

2.2 Top down & Bottom up Approach,

2.3 Cohesion and Coupling,

2.4 Structured Programming,

2.5 Deterministic and Non-deterministic Technique,

2.6 Iterative and Recursive Logic,

2.7 Modular Designing Programming

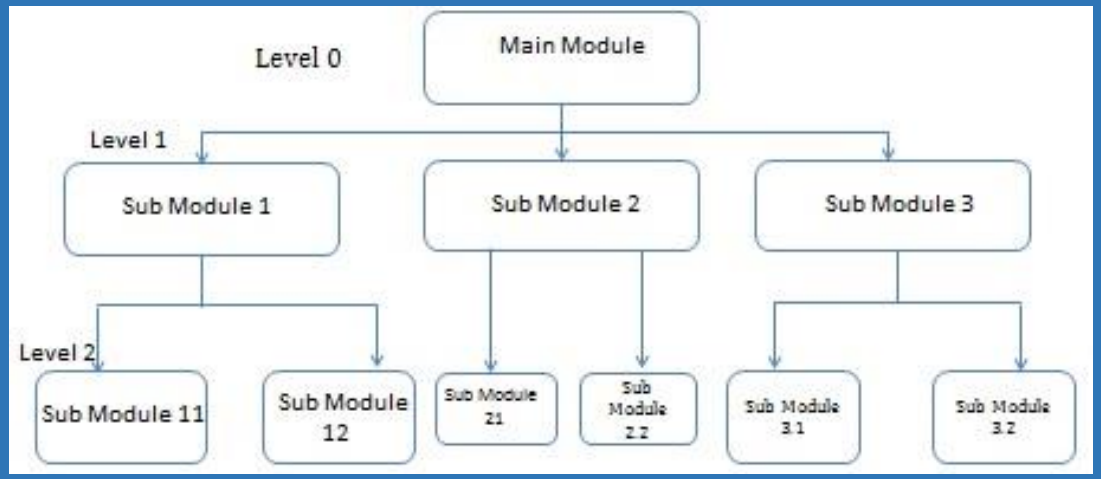
2.1 Introduction to Programming Technique

- A computer program is a set of instructions written in a programming language and organized in specific order to perform a specific task.
- A computer requires programs to accomplish any task using computer.
- When programmer writes such programs using different programming languages, he/she should follow specific approach/techniques/methodology of program development.
- These approaches are called programming techniques.
- There are many techniques that can be used in program development.

2.2 Top down & Bottom up Approach

1. Top-Down Strategy:

- The top-down strategy uses the modular approach to develop the design of a system. It is called so because it starts from the top or the highest-level module and moves towards the lowest level modules.
- In this technique, the highest-level module or main module for developing the software is identified. The main module is divided into several smaller and simpler submodules or segments based on the task performed by each module. Then, each submodule is further subdivided into several submodules of next lower level. This process of dividing each module into several submodules continues until the lowest level modules, which cannot be further subdivided, are not identified.
- C follows a top-down approach.



Advantages:

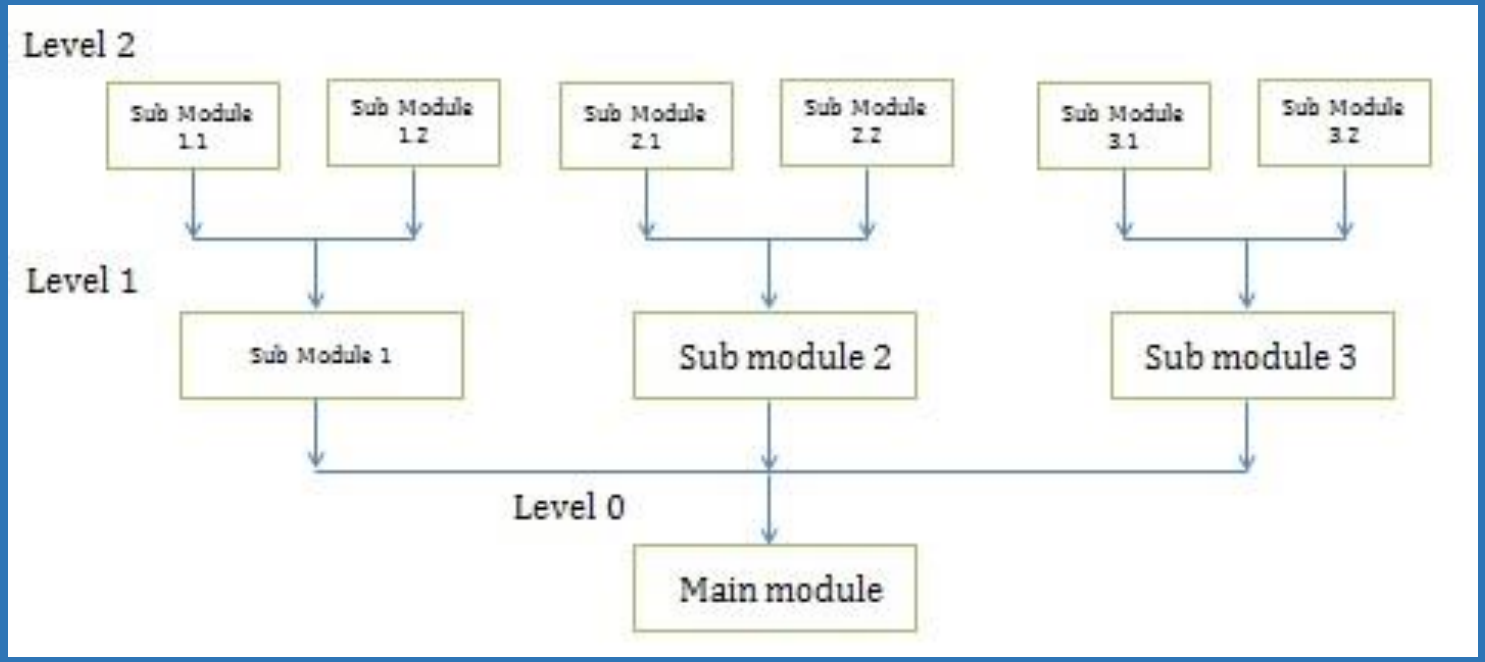
- Easy to decompose large program into a number of simpler modules.
- Easy to see the progress of program development by developer.
- Proper utilization of resources is possible on the basis of modules.
- Testing and debugging is easier and efficient.
- Implementation is smoother and shorter.

2. Bottom-Up Strategy:

- Bottom-Up Strategy follows the modular approach to develop the design of the system. It is called so because it starts from the bottom or the most basic level modules and moves towards the highest level modules.

In this technique,

- The modules at the most basic or the lowest level are identified.
- These modules are then grouped together based on the function performed by each module to form the next higher-level modules.
- Then, these modules are further combined to form the next higher-level modules.
- This process of grouping several simpler modules to form higher level modules continues until the main module of system development process is achieved.
- Object Oriented Programming languages like C++ or Java follows bottom-up approach.



Advantages:

- Make decisions about reusable low-level utilities then decide how there will be put together to create high-level construct.

Top-down Approach	Bottom-up Approach
1. The large task is decomposed into smaller subtasks.	1. The fundamental parts of the task are solved first and then combined those parts into a whole program.
2. Each sub-module is separately processed.	2. It implements the concept of data hiding through encapsulation.
3. The sub-modules don't require much communication.	3. Needs interaction between the separate fundamental modules to combine them later.
4. The procedural programming languages such as Fortran, COBOL and C follows a top-down approach.	4. OOP languages like C++, Java follows bottom-up approach.

2.3 Cohesion and Coupling

1. Cohesion:

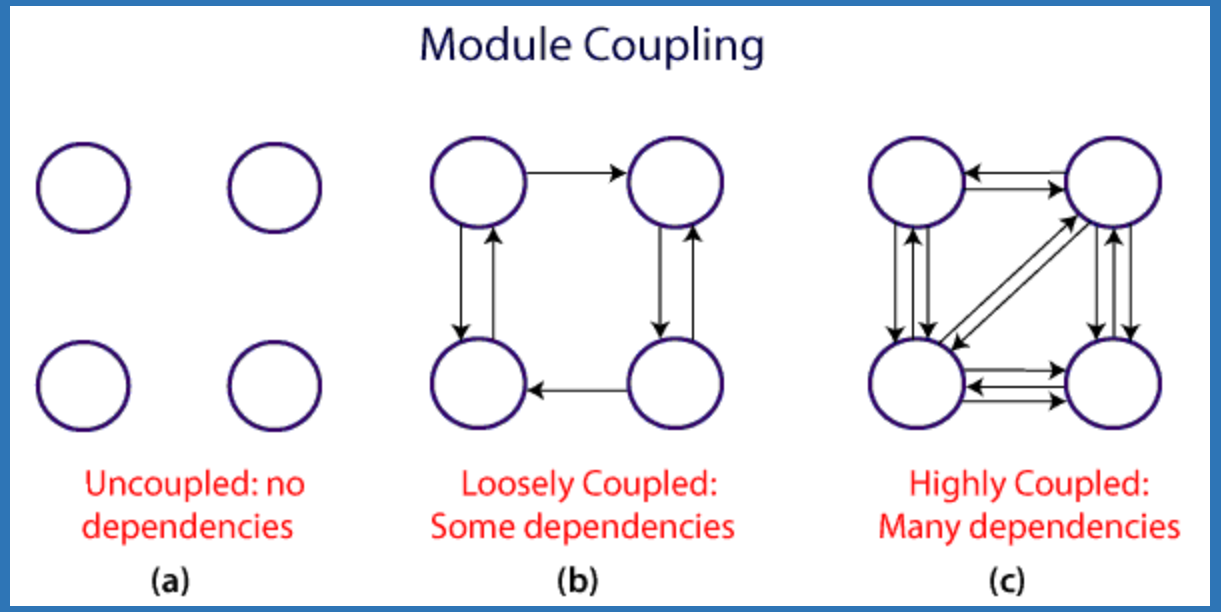
- In computer programming, cohesion defines to the degree to which the elements of a module belong together.
- Thus, cohesion measures the strength of relationships between pieces of functionality within a given module.
- A good software design will have high cohesion.
- It is intra-module concept.

Types of Cohesion:

1. **Functional Cohesion:** Functional Cohesion is said to exist if the different elements of a module, cooperate to achieve a single function.
2. **Sequential Cohesion:** A module is said to possess sequential cohesion if the element of a module form the components of the sequence, where the output from one component of the sequence is input to the next.
3. **Communicational Cohesion:** A module is said to have communicational cohesion, if all tasks of the module refer to or update the same data structure, e.g., the set of functions defined on an array or a stack.
4. **Procedural Cohesion:** A module is said to be procedural cohesion if the set of purpose of the module are all parts of a procedure in which particular sequence of steps has to be carried out for achieving a goal, e.g., the algorithm for decoding a message.
5. **Temporal Cohesion:** When a module includes functions that are associated by the fact that all the methods must be executed in the same time, the module is said to exhibit temporal cohesion.
6. **Logical Cohesion:** A module is said to be logically cohesive if all the elements of the module perform a similar operation. For example Error handling, data input and data output, etc.
7. **Coincidental Cohesion:** A module is said to have coincidental cohesion if it performs a set of tasks that are associated with each other very loosely, if at all.

2. Coupling:

- Coupling in program development measures the degree of interdependence between the different modules.
- A good program has low coupling.
- It is inter-module concept.



Types of Coupling:

1. **Data Coupling:** When data of one module is passed to another module, this is called data coupling.
2. **Stamp Coupling:** Two modules are stamp coupled if they communicate using composite data items such as structure, objects, etc. When the module passes non-global data structure or entire structure to another module, they are said to be stamp coupled. For example, passing structure variable in C or object in C++ language to a module.
3. **Control Coupling:** Control Coupling exists among two modules if data from one module is used to direct the structure of instruction execution in another.
4. **External Coupling:** External Coupling arises when two modules share an externally imposed data format, communication protocols, or device interface. This is related to communication to external tools and devices.
5. **Common Coupling:** Two modules are common coupled if they share information through some global data items. **Content Coupling:** Content Coupling exists among two modules if they share code, e.g., a branch from one module into another module.

2.4 Structured Programming

- In structured programming, programs are broken into different functions, called modules, subprogram, subroutines or procedures.
- Each function is design to do a specific task with its own data and logic.
- Information can be passed from one function to another function through parameters.
- A function can have local data that can’t be accessed outside the function’s scope.
- The result of this process is that all the other different functions are synthesized in another function which is known as main function.
- C programming language is an example of structured programming.

Features of structured programming:

- Supports modular programming.
- Programs are divided into individual function each function has the precisely defined interface to other functions in the program.
- Emphasis is given on the algorithm rather than on data.
- Maintenance of program is more straightforward than a procedural program.

Advantages:

- Reduced complexity: Modularity allows the programmer to think problems in logical way which ensures clarity of flow of control in program.
- Increased productivity: Modularity allows multiple programmers to work on project at the same time. Modules can be re-used many times, which saves time and reduces complexity but also increases reliability.
- Easy to maintain and update: It is also easier to update or fix the program by replacing individual modules rather than larger amounts of code.

Disadvantages:

- A high level language has to be translated into the machine language by translator and thus a price in computer time is paid.
- The object code generated by a translator might be inefficient compared to an equivalent assembly language program.
- Data types are frequently used in function. When some changes occur in those data types, the corresponding change must be made to every location that acts on those data types within the program which is time consuming task if the program is very large.

2.5 Deterministic and Non-deterministic Technique

- In deterministic technique, the computer program will always produce the same output going through the same states for a given particular input.
- In non-deterministic technique, the computer program may produce different output in different runs for same input.
- As the non-deterministic techniques can show different behaviors for the same input on different execution, there is a degree of randomness to it.

Deterministic Technique	Non-deterministic Technique
1. For a particular input, the computer program gives always same output.	1. For a particular input, the computer program may produce different output on different execution.
2. Can solve the problem in polynomial time.	2. Can’t solve the problem in polynomial time.
3. Can determine the next step of execution.	3. Can’t determine the next step of execution due to more than one path the algorithm can take.

2.6 Iterative and Recursive Logic

- In computer programming, iteration is used to describe a situation in which a sequence of instructions can be executed multiple times (e.g loops). One pass through the sequence is called iteration. If the sequence of instructions is executed repeatedly, it is called a loop and we say that the computer iterates through the loop.
- The process in which a function calls itself directly or indirectly is called recursion and the corresponding function is called as recursive function.
- Recursion and iteration both repeatedly executes the set of instructions.

Recursion	Iteration
1. A function is called from the definition of the same function to do repeated task.	1. It uses loop to do repeated task.
2. It follows top-down approach to problem solving.	2. It follows bottom-up approach.
3. A function calls to itself until some condition will be satisfied.	3. A function does not call to itself.
4. Problem could be written or defined in terms of its previous result to solve a problem using recursion.	4. It is not necessary to define a problem in terms of its previous result to solve using iteration.
5. It is always applied to function.	5. It is applied to set of instructions.
6. It uses stack to store recursive call.	6. It doesn’t require stack.
7. It reduces the size of program code.	7. It makes a code longer relative to recursion.

2.7 Modular Designing Programming

- It is a software design technique that focuses on separation of the functionality of a program into independent, inter-changeable modules.
- A module is a separate software component to perform a particular task.
- It can often be used in a variety of applications and functions with other components of the system.
- The breaking of large programs into small problems in modular programming techniques increases maintainability and readability of the code and makes the program simple to make any changes in future or to correct the errors.

Advantages:

1. **Ease of use**: It uses concept of software modules and makes ease in debugging the code and prone to less error.
2. **Reusability**: It allows the user to reuse the functionality with a different interface without typing the whole program again.
3. **Increased in productivity**: It helps in less collision at the time of working on modules, helping a team to work with proper collaboration while working on a large application.

THANK YOU FOR YOUR ATTENTION