C Programming

Facilitator: Krishna Prasad Acharya
Mechi Multiple Campus
BCA Programme

Unit-2 Programming Technique

Contents

- Introduction to programming technique
- Structure programming language
- Modular design and programming
- Cohesion and coupling
- Top down and button up approach
- Deterministic and non deterministic Technique
- Iterative and recursive logic

Introduction to programming technique

- **Computer programming** is the process of designing and building an executable computer program for accomplishing a specific computing task.
- Programming involves tasks such as analysis, generating algorithms, profiling algorithms' accuracy and resource consumption, and the implementation of algorithms in a chosen programming language (commonly referred to as **coding**).
- The source code of a program is written in one or more programming languages. The purpose of programming is to find a sequence of instructions that will automate the performance of a task for solving a given problem.
- The process of programming thus often requires expertise in several different subjects, including knowledge of the application domain, specialized algorithms, and formal logic.
- Related tasks include testing, debugging, maintaining a program's source code, implementation of build systems, and management of derived artifacts such as machine code of computer programs.

Introduction to programming technique

- **Programming Technique** is an systematic approach to analysis the problem and write the corresponding code in any platform to solve the problems. There are different approach to write a program which include:

1. **Non/Unstructured Programming**

- In unstructured programming language, the program must be written as a single continuous, i.e. nonstop or unbroken block.
- This makes it a bit complicated as the whole program is taken as one unit. Also, it becomes harder to modify and to debug, such as if there is a bug in the program, which there always is, the programmer much check the code of the entire program,
- Unstructured Programming is historically the earliest type of programming that was capable of creating Turing-complete algorithms.
- In structured programming language allows a programmer to code a program by diving the whole program into smaller units or modules.

Structure vs unstructured programming

	Structured Programming Language	Unstructured Programming Language
Also known as	Modular Programming	Non-structured Programming
Subset of	Procedural Programming	None. It is the earliest programming paradigm.
Purpose	To enforce a logical structure on the program being written to make it more efficient and easier to understand and modify.	Just to code.
Programming	Divides the program into smaller units or modules.	The entire program must be coded in one continuous block.
Precursor to	Object-oriented programming (OOP)	Structured programming, specifically procedural programming and then object-oriented programming.
Code	Produces readable code	Producing hardly-readable ("spaghetti") code
For Projects	Usually considered a good approach for creating major projects	Sometimes considered a bad approach for creating major projects
Freedom	Has some limitations	Offers freedom to programmers to program as they want
Allowed data types	Structured languages allow a variety of data types.	Non-structured languages allow only basic data types, such as numbers, strings and arrays (numbered sets of variables of the same type).
Modify and debug	Easy to modify and to debug	Very difficult to modify and to debug
Languages	C, C+, C++, C#, Java, PERL, Ruby, PHP, ALGOL, Pascal, PL/I and Ada	early versions of BASIC (such as MSX BASIC and GW-BASIC), JOSS, FOCAL, MUMPS, TELCOMP, COBOL, machine-level code, early assembler systems (without procedural metaoperators), assembler debuggers and some scripting languages such as MS-DOS batch file language

Introduction to programming technique

- It extensively use unstructured control flow using goto statements or equivalent.
- Spaghetti (very long and hardly-readable) code is highly difficult to follow and to maintain, which is why many people don't prefer to use unstructured programming languages.
- Some languages commonly cited as being non-structured include JOSS, FOCAL, TELCOMP, assembly languages, MS-DOS batch files, and early versions of BASIC, Fortran, COBOL, and MUMPS.

2. Procedural Programming.

- It is a programming Technique, derived from structured programming, based upon the concept of the procedure call.
- Procedures, also known as routines, subroutines, or functions, simply contain a series of computational steps to be carried out. Any given procedure might be called at any point during a program's execution, including by other procedures or itself.
- The first major procedural programming languages first appeared circa 1960, including Fortran, ALGOL, COBOL and BASIC, Pascal.

Programing paradigms

1.Procedure orientated programming(POP):

- Program in a procedural language is a list of instruction where each statement tells the computer to do something. It focuses on procedure (function) & algorithm is needed to perform the derived computation.
- When program become larger, it is divided into function & each function has clearly defined purpose. Dividing the program into functions & module is one of the foundations of structured programming.

Features

- It focuses on process(function, procedure) rather than data.
- Large problems are divided into smaller programs known as functions.
- Most of the functions share global data.
- Data move openly around the system from function to function.
- Functions transfer data from one form to another.
- Employs top-down approach in program designing.
- In the cases of large program, bringing change is difficult and time consuming.
- Appropriate and effective techniques are unavailable to secure data of a function from others.

Introduction to programming technique

Advantages of Procedural Programming:

- Its relative simplicity, and ease of implementation of compilers and interpreters
- The ability to re-use the same code at different places in the program without copying it.
- An easier way to keep track of program flow.
- The ability to be strongly modular or structured.
- Needs only less memory.

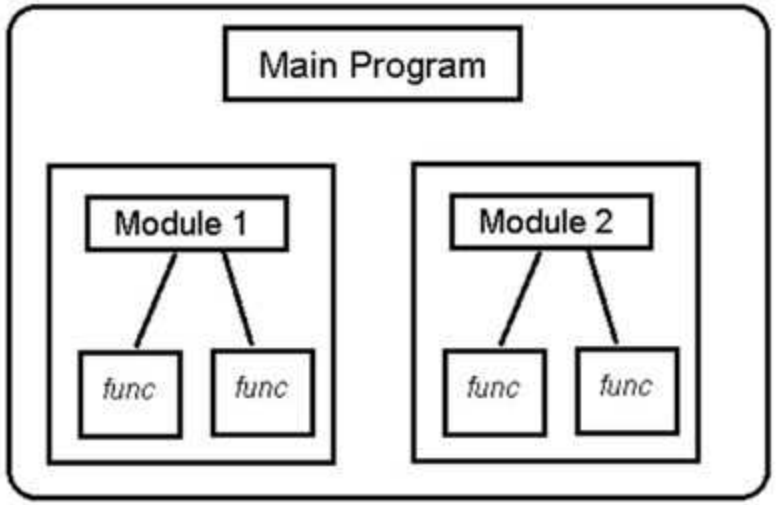
Disadvantages of Procedural Programming:

- Data is exposed to whole program, so no security for data.
- Difficult to relate with real world objects.
- Difficult to create new data types reduces extensibility.
- Importance is given to the operation on data rather than the data.

Introduction to programming technique

3. Modular Programming

- It is an approach to software development in which the individual functions of a program are separated into self-contained components called modules.
- Each module contains everything necessary to fulfill its own functionality, and can be edited and modified without the entire project being affected.
- Modular programming is accomplished by maintaining a consistent module interface that other software components use to access each module's functions.



Cohesion vs coupling

- When a software program is modularized, its tasks are divided into several modules based on some characteristics. As we know, modules are set of instructions put together in order to achieve some tasks.
- They are though, considered as single entity but may refer to each other to work together. There are measures by which the quality of a design of modules and their interaction among them can be measured. These measures are called **coupling** and **cohesion**.

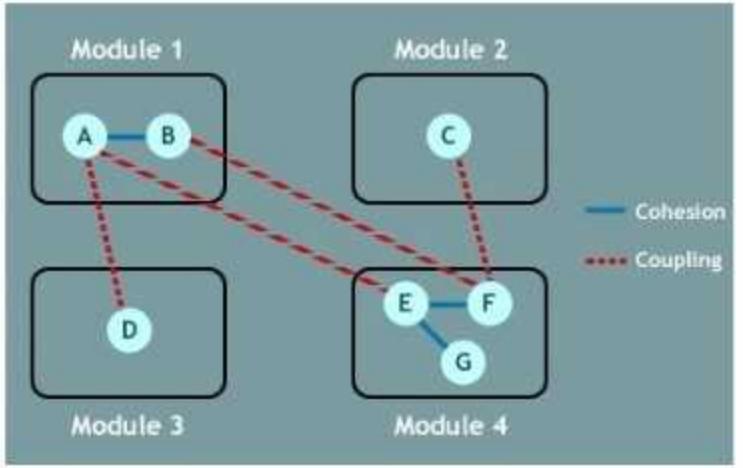
Characteristics of Good Design in modular programming:

- High cohesion
- Low coupling

Why is it required?

- Exception identification and handling
- Fault prevention and fault tolerance
- Design for change

1. Cohesion
2. Coupling



Coupling and Its Types

- The degree of interdependence between two modules.
- 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.
- Coupling is the measure of the interdependence of one module to another. Modules should have low coupling.
- Low coupling minimizes the "ripple effect" where changes in one module cause errors in other modules.

Types:

- Content coupling: When a module can directly access or modify or refer to the content of another module, it is called content level coupling. One module directly references contents of the other
  - if one refers to local data in another module.
  - if one branches into a local label of another.

Coupling and Its Types

Types:

- Common coupling: access to global data. modules bound together by global data structures.
- Control coupling: passing control flags (as parameters or global) so that one module controls the sequence of processing steps in another module.
- Stamp coupling: When multiple modules share common data structure and work on different part of it, it is called stamp coupling.
  - data structure is passed as parameter, but called module operates on only some of individual components
- Data coupling: use of parameter lists to pass data items between routines.

Cohesion

- It 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.
- "The measure of the strength of functional relatedness of elements within a module".
- Cohesion refers to the dependence within and among a module's internal elements (e.g., data, functions, internal modules)

Types of Cohesion

1. Coincidental cohesion
2. Logical cohesion
3. Temporal cohesion
4. Procedural cohesion
5. Communicational cohesion
6. Functional cohesion
7. Informational cohesion

Types of Cohesion

1.Coincidental cohesion: Coincidental strength exists if there is no meaningful relationship between the parts in a module

Coincidental strength exists if there is no meaningful relationship between the parts in a module

2.Logical cohesion : similar functions such as input, error handling, etc. put together. Functions fall in same logical class. May pass a flag to determine which ones executed.

3.Temporal cohesion: Elements of a component are related by timing.

4.Procedural Cohesion: Elements of a component are related only to ensure a particular order of execution.

5.Communicational cohesion: Module performs a series of actions related by a sequence of steps to be followed by the product and all actions are performed on the same data.

6.Functional cohesion: Every essential element to a single computation is contained in the component.

7.Informational cohesion: Module performs a number of actions, each with its own entry point, with independent code for each action, all performed on the same data.

Cohesion vs compiling

Cohesion	Coupling
Cohesion is the indication of the relationship within **module**.	**Coupling** is the indication of the relationships between modules.
Cohesion shows the module's relative **functional** strength.	Coupling shows the relative **independence** among the modules.
Cohesion is a degree (quality) to which a component / module focuses on the **single** thing	Coupling is a degree to which a component / module is connected to the **other** modules.

Introduction to programming technique

Advantages of modular programming

- Less code has to be written.
- A single module can be developed for reuse, eliminating the need to retype the code many times.
- Programs can be designed more easily because a small team deals with only a small part of the entire code.
- Modular programming allows many programmers to collaborate on the same application.
- The code is stored across multiple files.
- Code is short, simple and easy to understand.
- Errors can easily be identified, as they are localized to a subroutine or function.
- The same code can be used in many applications.
- The scoping of variables can easily be controlled.

Disadvantages of modular programming:

- Modular programming requires extra time and memory.
- The combining of modules together is a difficult task.
- There is need of careful documentation as it may affect the other parts of the program.
- Testing and debugging of separate modules may be time consuming, thus reducing the efficiency of a program.

Introduction to programming technique

4. Object oriented programming Technique

- In the object-oriented programming, program is divided into a set of objects. The emphasis given on objects, not on procedures. All the programming activities revolve around objects.
- An object is a real-world entity. It may be airplane, ship, car, house, horse, customer, bank Account, loan, petrol, fee, courses, and Registration number etc. Objects are tied with Method(functions).
- Objects are not free for walk without leg of method. One object talks with other through earphone of method. Object is a boss but captive of method.

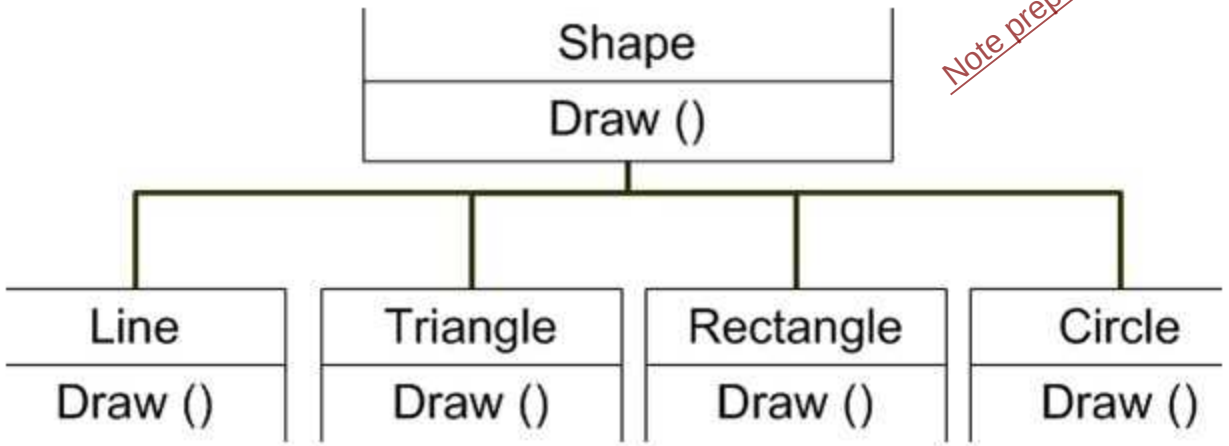
Features:

- It emphasis in own data rather than procedure.
- It is based on the principles of inheritance, polymorphism, encapsulation and data abstraction.
- It implements programs using the objects.
- Data and the functions are wrapped into a single unit called class so that data is hidden and is safe from accidental alternation.
- Objects communicate with each other through functions.
- New data and functions can be easily added whenever necessary.

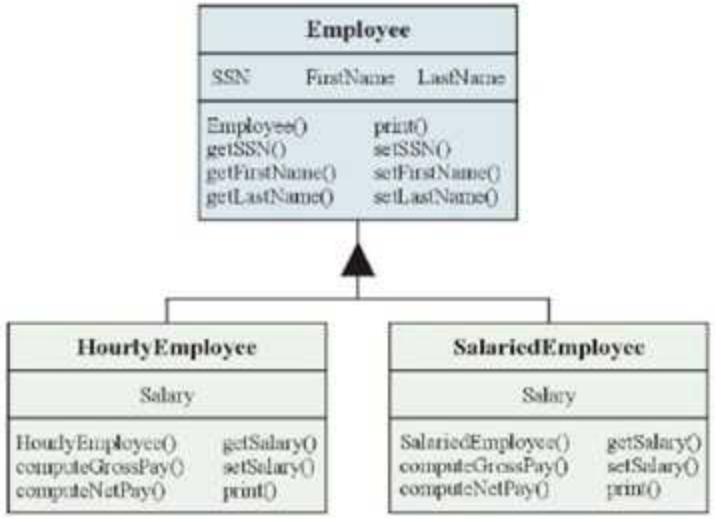
Polymorphism

- It is a mechanism where we can to derive a class from another class for a hierarchy of classes that share a set of attributes and methods.

- The word polymorphism is a combination of two words poly and morphism. Poly means many and morphism means form. A function use in different way is called polymorphism.



Inheritance Concepts - Example



Inheritance

- Inheritance is a hierarchy of class in which some properties of base class is transferred to derived class. The main types of inheritance are:

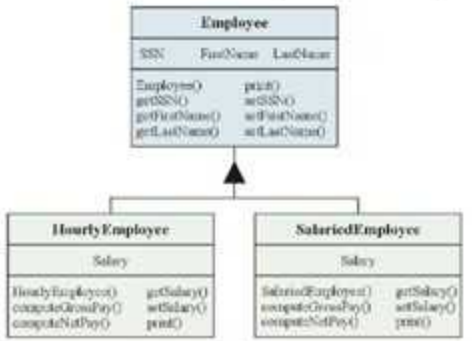
(a) Single Inheritance: A derived class (child class or sub class) of single base (super or parent) class is called single Inheritance.

(b) Multiple Inheritances: A derived class of multiple base classes is called Multiple Inheritance.

(c) Multilevel Inheritance: When derived class is derived from another derived class, such type of inheritance is called Multilevel Inheritance.

The word polymorphism is a combination of two words poly and morphism. Poly means many and morphism means form. A function use in different way is called polymorphism.

Inheritance Concepts - Example



Encapsulation

- The encapsulation is a very striking feature of OOP in which data and function is bound into single unit
- The data and function are encapsulated into class. External world or external function cannot access the data. It can be accessed by its own function or method encapsulated with it into class. It hides private elements of objects behind public interface.

Abstraction: The abstraction is an important property of OOP. The use of essential features over less essential features is called abstraction. The following examples will help to understand abstraction.

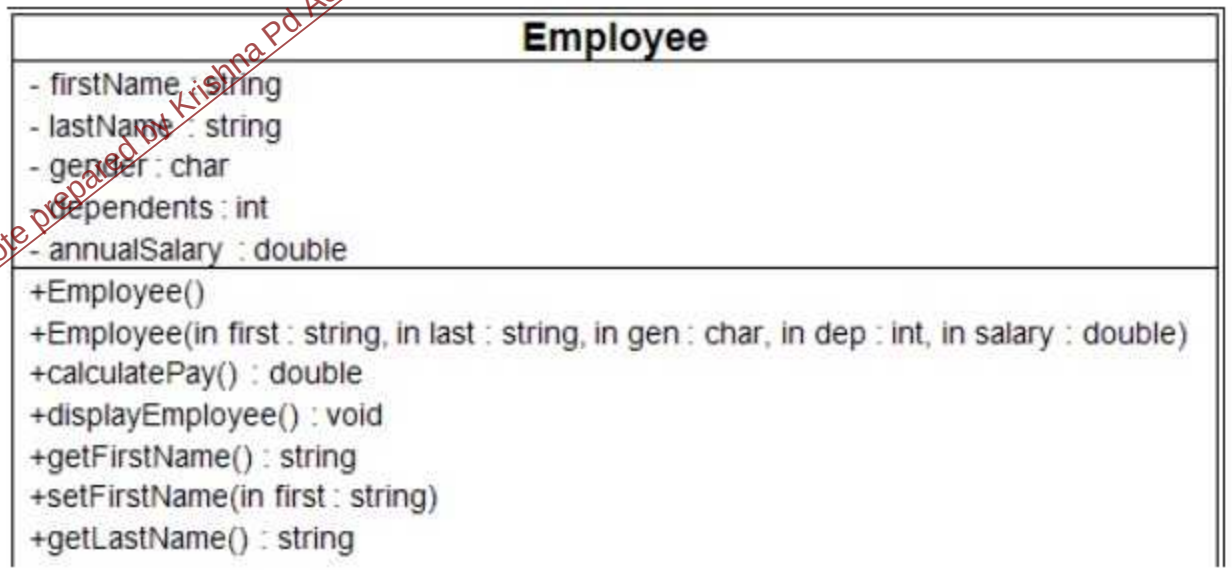
- Example: The computer operators know only to operate computer, but they are unaware to internal organization of computer.

Class

A class is a collection of similar objects. Objects are members of class. Once a class is declared, its many members can be easily created in programs. The class binds attributes (data and functions or methods) of member objects.

Examples:

```
Class employee
{
char name[30];
float basic;
void getdata();
void show();
};
```

Programing paradigms

OOP

1. In OOP, program is divided into parts called objects.
2. OOP has access specifiers named Public, Private, Protected, etc.
3. OOP provides Data Hiding so provides more security.
4. In OOP, overloading is possible in the form of Function Overloading and Operator Overloading.
5. In OOP, data can not move easily from function to function, it can be kept public or private so we can control the access of data.
6. Example of OOP are : C++, JAVA, VB.NET, C#.NET.

POP

1. In POP, program is divided into small parts called functions.
2. POP does not have any access specifier.
3. POP does not have any proper way for hiding data so it is less secure.
4. In POP, Overloading is not possible.
5. In POP, Most function uses Global data for sharing that can be accessed freely from function to function in the system.
6. Example of POP are : C, VB, FORTRAN, Pascal

Recursion vs Iteration

- An Iterative algorithm will use looping statements such as for loop, while loop or do-while loop to repeat the same steps while a Recursive algorithm, a function calls itself again and again till the base condition (stopping condition) is satisfied.
- An Iterative algorithm will be faster than the Recursive algorithm because of overheads like calling functions and registering stacks repeatedly. Many times, the recursive algorithms are not efficient as they take more space and time.
- Recursive algorithms are mostly used to solve complicated problems when their application is easy and effective. For example, Tower of Hanoi algorithm is made easy by recursion while iterations are widely used, efficient and popular.

Recursion vs Iteration

- **Approach**: In recursive approach, the function calls itself until the condition is met, whereas, in iterative approach, a function repeats until the condition fails .
- **Programming Construct Usage**: Recursive algorithm uses a branching structure, while iterative algorithm uses a looping construct.
- **Time & Space Effectiveness**: Recursive solutions are often less efficient in terms of time and space, when compared to iterative solutions
- **LOC(line of code)**: Unlike Iteration, Recursion simplifies a source code by shortening it.
- **Termination Test**: Iteration terminates when the loop-continuation condition fails; recursion terminates when a base case is recognized.
- **Infinite Call**: An infinite loop occurs with iteration if the loop-continuation test never becomes false; infinite recursion occurs if the recursion step does not reduce the problem in a manner that converges on the base case.

Recursion vs Iteration

Sl.No.	Iteration	Recursion
1.	Iteration explicitly user make a repetition structure.	Recursion achieves repetition throught repeated function calls.
2.	Iteration terminates when the loop continuation condition fails.	Recursion terminates when a base case is reached.
3.	Iteration keeps modifying the counter until the loop continuation condition fails.	Recursion keeps producing simple versions of the original problem until the base case is reached.
4.	An infinite loop occurs when the loop step continuation test never becomes false.	An infinite loop occurs if the recursion doses not reduce the problem each time in a manner that converges the base case.
5.	Iteration normally occurs within a loop so extra memory assignment is omitted.	Recursion causes another copy of the function & hence a considerable memory space is occupied.
6.	It reduces the processor operating time.	It increases the processor operating time.

Deterministic and Non-deterministic

- Deterministic functions always return the same result any time they are called with a specific set of input values and given the same state of the database.
- Nondeterministic functions may return different results each time they are called with a specific set of input values even if the database state that they access remains the same.
- For example, the function AVG always returns the same result given the qualifications stated above, but the GETDATE function, which returns the current datetime value, always returns a different result.

Top down vs bottom up approach

- The large program is divided into many small module or subprogram or function or procedure from top to bottom.
- At first supervisor program is identified to control other sub modules. Main modules are divided into sub modules, sub-modules into sub-sub modules. The decomposition of modules is continuing whenever desired module level is not obtained.
- Top module is tested first, and then sub-modules are combined one by one and tested.

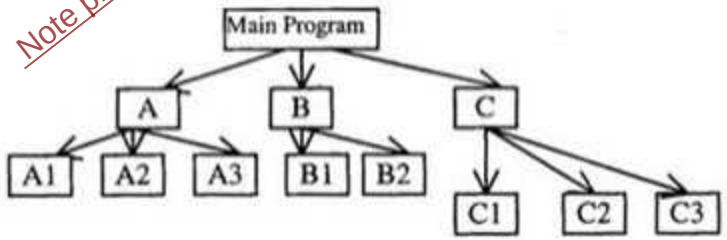


Fig: Top down Approach

- Example: The main program is divided into sub-program A, B, and C. The A is divided into subprogram A1, A2 and A3. The B is into B1, and B2. Just like these subprograms, C is also divided into three subprogram C1, C2 and C3. The solution of Main program is obtained from sub program A, B and C.

Top down vs bottom up approach

- In bottom up approach designing is started from bottom and advanced stepwise to top. So, this approach is called Bottom up approach.
- At first bottom layer modules are designed and tested, second layer modules are designed and combined with bottom layer and combined modules are tested. In this way, designing and testing progressed from bottom to top.
- In software designing, only pure top down or bottom up approach is not used. The hybrid type of approach is recommended by many designers in which top down and bottom up, both approaches are utilized.

Assignment

1. What do you mean by programing technique? Explain procedural programming approach.
2. Differentiate structure and non-structure programing approach.
3. Explain modular programming approach in detail.
4. What is cohesion? Explain different types of cohesion.
5. What is coupling? Explain different types of coupling.
6. What do you mean by object oriented approach? Explain inheritance and polymorphism with example.
7. Differentiate between recursion and iteration.
8. Differentiate between top down and bottom up programming approach.

Top down vs Bottom up approach

Top down

- Emphasis is on doing things (algorithms)
- Large program are divided into smaller programs known as function
- Most of functions share global data
- Data move openly around the system from function to function
- Functions transform data from one form to another
- This is mainly function oriented
- No data security
- Not extensible
- It follows procedure oriented programming
- Simple user defined data types are created.
- Programming language like c, Pascal follow this approach

Bottom up

- Emphasis is on data rather than function
- Large program are divided into what are known as Object
- Rare use of share global data
- Function that operate on the date are tied together in the data structure called class
- Data is hidden and cannot be accessed by external functions.
- Object may communicate with each other through functions
- New data and function can be easily added whenever necessary.
- This is mainly data(object) oriented
- Data security is exist.
- Extensible
- It fallows object oriented programming approach
- Complex user data types can be created.
- Programming language like c++ ,java use this approach.