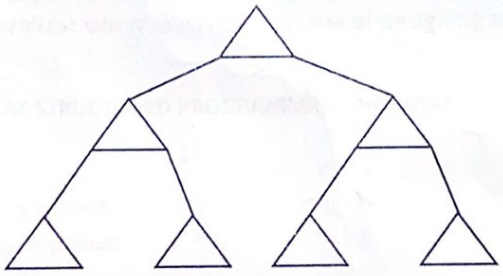
## COMPARATIVES LOOK AT STRUCTURED PROGRAMMING AND OOP

- Top-down approach
- Bottom-up approach
- Process
- Data
- Languages
- Etc.

**Introduction:**

**Top-down** and **bottom-up** are both an algorithm design methods or strategies of information processing and knowledge ordering, used in a variety of fields including humanistic and scientific theories, management and organization, and **software** development. Top-down design was promoted in the 1970s by IBM researchers Harlan Mills and Niklaus Wirth to aid program design. Program design is a process of converting a set of requirements into a collection of commands or a program that can be executed on a computer system.


TOP-DOWN DESIGN APPROACH

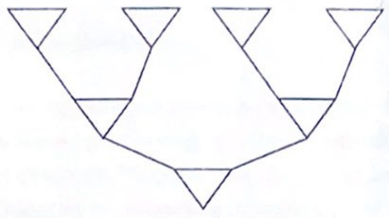
**Top down:**
Top-down program design also known as decomposition or stepwise refinement is a conventional approach to program design that start with the general concept and repeatedly breaks the complex module down into its component parts or submodules. It decomposes the system from high-level specification to low-level specification.

**Common features with top-down design:**

- The programming style supports procedural languages
- The approach starts with the Big Picture, and continue by breaking down the system to gain insight into its compositional subsystems (decomposition)
- specifying complex pieces and then dividing them into successively smaller pieces.
- The technique for writing a program using top–down methods is to write a main procedure that names all the major functions it will need.
- The top-down approach emphasizes on the isolation of the submodules while ignores the identification of communication and reusability concept
- **Support Specialization**: the process of designing subclasses from "top down"

**Bottom up**
Bottom up is the exact inverse of top-down approach. It starts with the component part or elementary modules and repeatedly combines them up to achieve the general concept. It starts with

the specific and continually combines it until it reaches the abstract. In the bottom-up approach, information hiding, and reusability are the prominent factors.



Features of Bottom-up approach

- Reusability of code is one of the main benefits of the bottom-up approach
- Bottom-up emphasizes coding and early testing, which can begin as soon as the first module has been specified
- the individual base elements of the system are first specified in detail.
- The approach resembles a "seed" model, where the beginnings are small, but eventually grow in complexity and completeness.
- The design style supports Object-oriented programming (OOP)
- **It supports generalization**: which is the process of designing subclasses from "bottom up"

**A COMPARATIVES LOOK AT STRUCTURED PROGRAMMING AND OOP**

**with respect to**

- Top-down approach
- Bottom-up approach
- Process
- Data
- Security
- Reusability
- Ability to handle complexity
- Etc.

**Structured Programming**

1. Structured Programming is designed which focuses on **process** (logical structure and then data required for that process).
2. Structured programming follows **top-down approach.**
3. In Structured Programming, Programs are divided into small self-contained **functions or procedures.**
4. Structured Programming provides **less reusability**, more function dependency.
5. Less abstraction and less flexibility.
6. Structured Programming is also known as **Modular Programming** and a subset of **procedural programming language.**
7. Structured Programming is **less** secure as there is no way of **data hiding.**
8. Structured Programming can solve **moderately** complex programs.
9. Structured Programming provides **less reusability**, more function dependency.

10. Languages that support this paradigm is "C".

**Object Oriented Programming**

1. Object Oriented Programming is designed which focuses on **data itself**.
2. Object oriented programming follows **bottom-up approach**.
3. In Object Oriented Programming, Programs are divided into small entities called **objects**
4. Object Oriented Programming provides more reusability, less function **dependency**.
5. More abstraction and more **flexibility**.
6. Object Oriented Programming supports **inheritance, encapsulation, abstraction**, **polymorphism**, etc.
7. Object Oriented Programming is more secure as having data hiding feature.
8. Object Oriented Programming can solve any **complex** programs.
9. Object Oriented Programming provides more reusability, less function **dependency**.
10. Languages that support this paradigm are C++, Java etc.