Abstract Data Type (ADT)

Data Abstraction & ADT

When you switch on the fan and it starts rotating at its desired speed, you don't care about the process of its internal working. Fan works unless there is a power cut. This real-world example highlights the programming concept of **data abstraction**, which allows a programmer to protect/hide the implementation of a process and only give the "keys" to other functions or user data. Similarly, in the case of programming when a user tries to find out some information, it is provided to him/her without showing the internal working of the functions in our program.

Data types created using the same concept are known as abstract data types. An **abstract data type** (or ADT) is a class that has a defined set of operations and values. If we consider the above example, then making the fan's switch as the entire abstract data type will prevent the user from knowing all of its internal working.

Features of ADT

- Prevents users from knowing the working of the program.
- Provides only specific data asked for if it's asking for an accurate set of data.

Methods of Abstraction

There are mainly three ways of data abstraction:

Procedural Abstraction

- Performs already stored repeated tasks, if asked for.
- Follows a fixed procedure of instructions provided.
- Generally, used in programs by invoking them. For example: While implementing a square root function in C++, we prefer to directly call the in-built **sqrt()** function and provide arguments to it, but we are not allowed to view the complete program written for this function.
- **Disadvantage:** Data is public, hence could be modified by any programmer using it.

Modular (File) Abstraction

- Contains both public and private sections and hence, can be used as per requirements.
- **Disadvantage:** Data can either be public or private at a single time.

Object-Oriented Programming

- The most efficient method of data abstraction.
- Data and procedures are instantiated together within an object.
- Multiple instantiations possible overcoming the disadvantage of modular abstraction.
- Accessing data can be controlled by the programmer.
- The program becomes more robust and readable using it.