**Introduction**

Generally, any problem that has to be solved by the computer involves the use of data. If data is arranged in some systematic way, then it gets a structure and becomes meaningful. This meaningful or processed data is called information. It is essential to manage data in such a way so that it can produce information.

There can be many ways in which data may be organized or structured. To provide an appropriate structure to your data, you need to know about data structures. Data structures can be viewed as a systematic way to organize data so that it can be used efficiently. The choice of a proper data structure can greatly affect the efficiency of our program.

**1. Need of Data Structure**

Data structures are essential because:

* **Efficient Data Storage**: Helps in organizing data in memory efficiently.
* **Fast Access**: Improves data retrieval speed.
* **Data Processing**: Makes it easier to perform operations like search, insertion, deletion, sorting, and traversal.
* **Memory Management**: Optimizes memory usage by choosing suitable data structures.
* **Code Reusability**: Promotes reusable and modular code through Abstract Data Types (ADTs).
* **Solves Complex Problems**: Crucial for algorithm design and solving problems in AI, DBMS, OS, etc.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**2. Basic Terminology**

**a. Data Item**

* A single unit of values (e.g., an integer, character, or string).
* Example: 5, 'A'

**b. Field**

* A collection of data items that represent one piece of information.
* Example: A studentName field may store a string of the student's name.

**c. Record**

* A collection of fields.
* Example: A Student record may contain fields: Name, RollNo, Marks.

**d. File**

* A collection of related records stored together.
* Example: StudentRecords.txt containing all student information.

**e. Entity**

* A real-world object with an identity.
* Example: A Student, Employee.

**f. Attributes**

* Properties of an entity.
* Example: For Student – Name, Age, ID.

**Definition of Data Structure:**

A **data structure** is a **way of organizing and storing data** in a computer so that it can be **accessed and modified efficiently**.

It defines the **relationship between data elements**, the **operations** that can be performed on the data, and how data is **stored and retrieved**.

**🔹 Formal Definition:**

“A data structure is a specialized format for organizing, processing, retrieving, and storing data.”

**a. Data and Information**

* **Data**: Raw facts (e.g., numbers, characters).
* **Information**: Processed data that is meaningful.

**b. Data Type**

A data type defines a domain of allowed values and the operations that can be performed on those values. For example, in C, an int data type can take values in a range, and operations that can be performed are addition, subtraction, multiplication, division, bitwise operations, etc.

Similarly, the float data type can take values in a particular range, and operations allowed are addition, subtraction, multiplication, division, etc. (Note: % operation and bitwise operations are not allowed for floats). These are called built-in data types or primitive data types.

If an application needs to use a data type other than the primitive types (e.g., a date type), then it is the programmer’s responsibility **to** define and implement such a type, since it is not natively available in the language.

* A classification that tells the compiler/interpreter what type of value a variable holds.
* Example: int, float, String in Java.

**c. Data Object**

* An instance of a data type.
* Example: int a = 10; → a is a data object.

**d. Abstract Data Type (ADT)**

An Abstract Data Type (ADT) is a mathematical model thatdefines a data type logically. It specifies:

* A set of data, and
* A collection of operations that can be performed on that data.

ADT only tells what operations are to be performed, not how they will be implemented.

It does not specify:

* How data will be stored in memory.
* What algorithms will be used.

This separation is known as abstraction, and the details are hidden from the user.

Users use data types like int, float, and char knowing only what they do — not how they are implemented. Similarly, ADT allows users to work with data logically.

We can think of an ADT as a black box — it hides the internal structure and implementation details.

* A model for data types where only the behaviour is defined, not the implementation.
* Example: Stack, Queue, List, Tree.

**e. Data Structure**

* A way to store and organize data for efficient use.
* Can be Linear or Non-Linear.

**4. Types of Data Structures**

| **Category** | **Types** | **Examples** |
| --- | --- | --- |
| **Primitive** | Built-in types | int, float, char, boolean |
| **Non-Primitive** | - **Linear** - **Non-Linear** | Array, Linked List Tree, Graph |
| **Static** | Size is fixed | Array |
| **Dynamic** | Size can grow/shrink | Linked List, Stack, Queue |

**What is Algorithmic Notation?**

**Algorithmic Notation** is a way to write algorithms in a structured, clear, and human-readable format that is **independent of any programming language**.

It allows us to:

* Express logic without worrying about syntax.
* Focus on the **steps** and **flow** of the algorithm.
* Communicate ideas clearly to others before actual coding.

**🔹 Key Features of Algorithmic Notation**

* Uses **simple English-like language**.
* Contains **control structures** like IF, WHILE, FOR, etc.
* Uses symbols like ← for assignment.

**🔹 Common Symbols in Algorithmic Notation**

| **Symbol** | **Meaning** | **Example** |
| --- | --- | --- |
| ← | Assignment | x ← 10 (means x = 10) |
| IF | Decision making | IF x > 0 THEN |
| ELSE | Alternate condition | ELSE print "Negative" |
| WHILE | Loop while condition is true | WHILE i < 5 DO |
| FOR | Loop with counter | FOR i ← 1 TO 10 DO |
| // | Comment (ignore during run) | // This is a comment |

**🔹 Example Algorithm in Algorithmic Notation**

**Problem**: Find the sum of first 5 natural numbers.

text

CopyEdit

Algorithm Sum\_Natural\_Numbers

Step 1: Initialize sum ← 0

Step 2: FOR i ← 1 TO 5 DO

sum ← sum + i

END FOR

Step 3: Print sum

End Algorithm

**🔹 Why Use Algorithmic Notation?**

* Easy to understand.
* Helps in **designing algorithms** before coding.
* Good for **planning and documentation**.
* Platform- and language-independent.

**Big O Notation: Explained Simply**

**Big O Notation** is used in **computer science** to describe the **efficiency** of an algorithm, especially how it **scales** with increasing input size.

**💡 Why Use Big O?**

To understand:

* How **fast** or **slow** an algorithm runs
* How much **memory** it uses
* How it behaves as **input size (n)** grows

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**6. Control Structure**

Directs the flow of program execution:

* **Sequence**: Instructions are executed in order.
* **Selection**: Decisions (e.g., if, switch)
* **Iteration**: Loops (e.g., for, while, do-while)