# **Data and Data Types**

**Data**

Strictly speaking, data is the plural of **datum**, a single piece of information. In practice, however, people use data as both the singular and plural form of the word.

Data can exist in a variety of forms as numbers or text on pieces of paper, as bits and bytes stored in electronic memory, or as facts stored in person’s mind.

**Data Types**

Data types are the category in which the variables are listed to hold the specific value. A data type tends to mean a primitive data type. Primitive data are built-in data types, such as integers, characters and Booleans. They are basic constructs of the language (that is, they are built into the language).

Primitive data type also tends to be of a strict data type, meaning you can't treat characters like integers or Booleans like integers, etc., although some languages will support implicit casting of primitive data types (for example, will treat Booleans like integers if you use a Boolean in an arithmetic operation).

A data type or simply type is a classification identifying one of various types of data, such as

* real-valued, integer or Boolean, that determines the possible values for that type,
* the operations that can be done on values of that type,
* the meaning of the data and
* the way values of that type can be stored.

# **Data Structure**

A data structure is a way of organizing data that considers not only the items stored, but also their relationship to each other. Advance knowledge about the relationship between data items allows designing of efficient algorithms for the manipulation of data.

Data may be organized in many different ways: the logical or mathematical model of a particular organization of data is called data structure. Data model depends on two things. First, it much be rich enough in structure to mirror the actual relationship of the data in the real world. On other hand, the structure should be simple to execute the process the data when necessary.

Data are also organized into more complex types of structures. The study of such data structure includes the following three steps:

1. Logical or mathematical descriptions of the structure.
2. Implementations of the structure on a computer.
3. Quantitative analysis of the structure, which include determining the amount of memory needed to store the structure and the time required to process the structure.

**Operations on Data Structure**

Data are processed by means of certain operations which appearing in the data structure. Data has situation that depends largely on the frequency with which specific operations are performed. Some of the operations on data structures are:

1. **Traversing:** Accessing each records exactly once so that certain items in the record maybe processed.
2. **Searching:** Finding the location of a particular record with a given key value, or finding the location of all records which satisfy one or more conditions.
3. **Inserting:** Adding a new record to the structure.
4. **Deleting:** Removing the record from the structure.
5. **Sorting:** Managing the data or record in some logical order(Ascending or descending order).
6. **Merging:** Combining the record in two different sorted files into a single sorted file.

**Types of Data structure**

Data structures are classified into two types such as linear or non-linear.

* **Linear:** A data structure is said to be linear if its elements form a sequence. The element of linear data structure represents by means of sequential memory locations. The other way is to have the linear relationship between the elements represented by means of pointers or links. E.g. Array and Link List.
* **Non-linear:** A data structure is said to be non-linear if its elements a hierarchical relationship between elements such as trees and graphs. All elements assign the memory as random form and you can fetch data elements through random access process.

**Importance of data structure**

Data structure is a particular way of storing and organizing information in a computer so that it can be retrieved and used most productively.

Different kinds of data structures are meant for different kinds of applications, and some are highly specialized to specific tasks.

1. Data structures are important for the following reasons:
2. Data structures are used in almost every program or software system.
3. Specific data structures are essential ingredients of many efficient algorithms, and make possible the management of huge amounts of data, such as large integrated collection of databases.

Use of appropriate data structure enables a computer system to perform its task more efficiently, by influencing the ability of computer to store and retrieve data from any location in its memory. Different kind of data structures is suited to different computer applications and tasks.

# **DS vs ADT**

We need to be clear about the logical and implementation level of data. Let us take an example of a simple built-in data type integer.

* At the logic level, we application programmers know only about what are the operations an integer data type can perform, i.e., addition, subtraction etc., But we are no way aware of how the data type is actually implemented.
* At the implementation level, it is about how the data type integer is implemented in the machine level, i.e., it could either of binary-coded decimal, unsigned binary, sign-and-magnitude binary, One's complement and Two's complement notation.

Now for understanding the ADT and data structure, we need to assume a higher level abstraction where we have the built-in types at the implementation level.

To put it simple, ADT is a logical description and data structure is concrete. ADT is the logical picture of the data and the operations to manipulate the component elements of the data.

Data structure is the actual representation of the data during the implementation and the algorithms to manipulate the data elements.

ADT is in the logical level and data structure is in the implementation level. ADT is implementation independent. For example, it only describes what a data type List consists (data) and what are the operations it can perform, but it has no information about how the List is actually implemented.

Whereas data structure is implementation dependent, as in the same example, it is about how the List implemented i.e., using array or linked list. Ultimately, data structure is how we implement the data in an abstract data type

**Data Structure**

* A data structure is just a bunch of data grouped together for some reason. All one can do with a data structure is access/modify the data within the structure.
* Data structures as being implementations of a specific type of algorithm.
* Data structures are the way in which the values and variables are stored in the memory.
* A data structure is a gathering together of many different data types. For example, objects and arrays are data structures.
* Data structures usually can contain information of many different types (such as strings, integers, Booleans) at the same time, and in more complex structures -- namely, classes -- can contain specific methods, properties and events to manipulate that data, change its type, etc.
* A data structure is a particular way of storing and organizing data in a computer so that it can be used efficiently.

**Abstract Data Type**

* An abstract data type is a collection of data plus the operations that can be performed on the data.
* Abstract data types having a specific set of properties which can be implemented in any number of ways.
* Abstract data type is the user defined data type model designed by programmer to store complex data.
* Abstract data types are generally constructed by the user or by a higher level language. For example, you might create a currency data type, which generally acts like a float but always has a precision of 2 decimal places and implements special rules about how to round off fractions of a cent.
* Abstract data types also often contain the ability to either be treated as a specific type of primitive data in certain circumstances (for example, many languages allow you to treat strings as character arrays); or contain certain rules / methods to manipulate their data (such as a programming language allowing you to cast a float as an integer).
* An abstract data type (ADT) is a mathematical model for a certain class of data structures that have similar behavior; or for certain data types of one or more programming languages that have similar semantics. An abstract data type is defined indirectly, only by the operations that may be performed on it and by mathematical constraints on the effects (and possibly cost) of those operations.
* Abstract data types are purely theoretical entities, used to simplify the description of abstract algorithms, to classify and evaluate data structures, and to formally describe the type systems of programming languages. However, an ADT may be implemented by specific data types or data structures, in many ways and in many programming languages; or described in a formal specification language.