**Scenario**

Shop-Now is an online Ecommerce Platform (B2C-Business to Customer). The company owns several sales outlets linked to a central warehouse where stocks are maintained and deliveries start from. The company requires a software system that has the following features:

1. The orders are collected locally at the sales outlets.
2. They are transmitted to the warehouse, where the delivery procedure should be managed.
3. An account of the delivery must be sent back to the sales outlets for following through the

Client’s order.

Shop-Now has published a notice asking for the proposal of developing the B2C Software System and In order to handover the project for the best one the Technical Lead of Shop-Now needs to know the interested company understands on how different data structures are used and manipulated and wants to know if the company is able to integrate best and optimized algorithms so as to develop efficient and accurate applications. I am an owner of Prativa Technology Pvt. Ltd and I am interested in developing the software system of Shop-Now. To show that our company understands on use of different ADTs, different sorting and searching algorithms, their applications and efficiency, we have to prepare the document. The document will have four parts which will have further divided sub tasks.

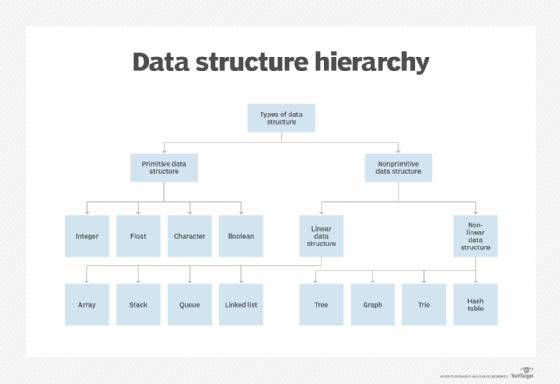
Let me start the documentation. At first, I will discuss about data, data structures and algorithms and will move towards what the clients of the application have asked us to do

Data can be defined as an elementary value or the collection of values, for example, student's name and its id are the data about the student. Data is a collection of facts and figures that can later be processed and can be used for extracting meaningful information. They are considered to be individual units of information which don’t give any meaning when taken individually. Data can be of two types; qualitative and quantitative data. Qualitative data which are explained in non-numeric words or values whereas qualitative data are the numeric values that further have two categories; discrete (whole numbers like 5, 100, 437 etc.) and continuous (decimal numbers like 2.43, 43.256, 40294.3435 etc.). In general, examples of data are; images, symbols, numbers, characters etc. Data processing converts data into meaningful information.

**Data Structures** Data structure is a specialized [format](https://whatis.techtarget.com/definition/format) for organizing, processing, retrieving and storing [data](https://searchdatamanagement.techtarget.com/definition/data). While there are several basic and advanced structure types, any data structure is designed to arrange data to suit a specific purpose so that it can be accessed and worked with in appropriate ways. Data Structure is a way of collecting and organizing data in such a way that we can perform operations on these data in an effective way. Data Structures is about rendering data elements in terms of some relationship, for better organization and storage. For example, we have some data which has, player's name "Virat" and age 26. Here "Virat" is of String data type and 26 is of integer data type.

We can organize this data as a record like Player record, which will have both player's name and age in it. Now we can collect and store player's records in a file or database as a data structure. For example: "Dhoni" 30, "Gambhir" 31, "Sehwag" 33 If you are aware of Object Oriented programming concepts, then a class also does the same thing, it collects different type of data under one single entity. The only difference being, data structures provides for techniques to access and manipulate data efficiently.

In simple language, Data Structures are structures programmed to store ordered data, so that various operations can be performed on it easily. It represents the knowledge of data to be organized in memory. It should be designed and implemented in such a way that it reduces the complexity and increases the efficiency.



**Types of data structure**

# Types of data structures

A data type is an attribute of data which tells the compiler (or interpreter) how the programmer intends to use the data.

**Primitive**: Primitive data structures are those which are predefined way of storing data by the system. And the set of operations that can be performed on these data are also predefined. Primitive data structures are char, int, float, double. Characters are internally considered as int and floats also falls under double and the predefined operations are addition, subtraction, etc.

**Composite:**  composite data type or compound data type is any data type which can be constructed in a program using the programming language's primitive data types and other composite types. It is sometimes called a structure or aggregate data type, although the latter term may also refer to arrays, lists, etc. The act of constructing a composite type is known as composition. Composite data types are often contrasted with scalar variables.

**Abstract:** Abstract data types are mathematical models of a set of data values or information that share similar behavior or qualities and that can be specified and identified independent of specific implementations. Abstract data types, or ADTs, are typically used in algorithms. An abstract data type is defined in term of its data items or its associated operations rather than by its implementation.

**Linked List** A linked list is a sequence of data structures, which are connected together via links. Linked List is a sequence of links which contains items. Each link contains a connection to another link. Linked list is the second most-used data structure after array.

**Types of Linked List**

Following are the various types of linked list

**Simple Linked List** − Item navigation is forward only.

**Doubly Linked List** − Items can be navigated forward and backward.

**Circular Linked List** − Last item contains link of the first element as next and the first element has a link to the last element as previous.

**Basic Operations**

Following are the basic operations supported by a list.

**Insertion** − Adds an element at the beginning of the list.

**Deletion** − Deletes an element at the beginning of the list.

**Display** − Displays the complete list.

**Search** − Searches an element using the given key.

**Delete** − Deletes an element using the given key.

**Array:** An array is the data structure that contains a collection of similar type data elements. Arrays a kind of data structure that can store a fixed-size sequential collection of elements of the same type. An array is used to store a collection of data, but it is often more useful to think of an array as a collection of variables of the same type.

# Basic Terminology

Data structures are the building blocks of any program or the software. Choosing the appropriate data structure for a program is the most difficult task for a programmer. Following terminology is used as far as data structures are concerned

**Data:** Data can be defined as an elementary value or the collection of values, for example, student's name and its id are the data about the student.

Group Items: Data items which have subordinate data items are called Group item, for example, name of a student can have first name and the last name.

Record: Record can be defined as the collection of various data items, for example, if we talk about the student entity, then its name, address, course and marks can be grouped together to form the record for the student.

File: A File is a collection of various records of one type of entity, for example, if there are 60 employees in the class, then there will be 20 records in the related file where each record contains the data about each employee.

Attribute and Entity: An entity represents the class of certain objects. it contains various attributes. Each attribute represents the particular property of that entity.

Field: Field is a single elementary unit of information representing the attribute of an entity.

# Need of Data Structures

As applications are getting complexes and amount of data is increasing day by day, there may arise the following problems:

**Processor speed:** To handle very large amount of data, high speed processing is required, but as the data is growing day by day to the billions of files per entity, processor may fail to deal with that much amount of data.

**Data Search**: Consider an inventory size of 106 items in a store, If our application needs to search for a particular item, it needs to traverse 106 items every time, results in slowing down the search process.

**Multiple requests:** If thousands of users are searching the data simultaneously on a web server, then there are the chances that a very large server can be failed during that process.

In order to solve the above problems, data structures are used. Data is organized to form a data structure in such a way that all items are not required to be searched and required data can be searched instantly.

# Advantages of Data Structures

* Data structures allow information storage on hard disks.
* Provides means for management of large dataset such as databases or internet indexing services.
* Are necessary for design of efficient algorithms.
* Allows safe storage of information on a computer. The information is then available for later use and can be used by multiple programs. Additionally, the information is secures and cannot be lost (especially if it is stored on magnetic tapes).
* Allows the data use and processing on a software system.
* Allows easier processing of data.
* Using internet, we can access the data anytime from any connected machine (computer, laptop, tablet, phone, etc.)

# Disadvantages:

* Only advanced users can make changes to data structures
* Any problem involving data structure will need an expert's help, i.e. basic users cannot help themselves

.

**Algorithms**

An algorithm is defined as a step-by-step procedure or method for solving a problem by a computer in a finite number of steps. Steps of an algorithm definition may include branching or repetition depending upon what problem the algorithm is being developed for. While defining an algorithm steps are written in human understandable language and independent of any programming language. We can implement it in any programming language of our choice. Mostly, data structures and algorithms are taken together during discussions. This is because, where data structure comes, it has to be passed through one of many algorithms to use them for system development and data definition tasks in the application/system development phases. In data and data structures’ field, algorithms can be used to define the steps to be carried out for inserting a new data, searching or sorting a list of items, or any other kind of data manipulation tasks in computing system. It is a very important tool in data handling process. In data processing systems, we use different types of sorting algorithms like; quick sort, merge sort, bubble sort, insertion sort, selection sort etc. are used. (Programiz, n.d.)

**Design specification for data structures**

Before moving into design specification, let’s first look into how the system works. At first, the orders are collected locally at the sales outlets (retail store). The orders are then transmitted to the warehouse, where the delivery procedure should be managed. After that, an account of the delivery must be sent back to the sales outlets for following through the client's order. In general, a customer places order online using the e-commerce site. The order information is sent to stockroom. Delivery procedure is managed in the warehouse. The information of delivery is sent back to the sales outlet. All these things which require information regulation are done with the use of Shop-Now online Ecommerce Platform. The main focus of the system are information of order and delivery.

We are going to use ASN.1 formal notation for design specification of this system. ASN.1 stands for Abstract Syntax Notation one. It is a joint standard from IEC (International Electrotechnical Commission), ITU-T (International Telecommunication Union) and ISO (International organization for Standardization). It is used by different bodies to describe data transmitted by telecommunications protocols. We get to see ASN.1 in different phases of our lives but they generally come to us in hidden form. Mostly, this is used in health and genetics, biometrics, banking, security authentication and cryptography, banking, aviation etc. Along with this, VOIP (Voice over Internet Protocol), LDAP (Lightweight Directory Access Protocol), SNMP (Simple Network Management Protocol), X.509 which include HTTPS (Hypertext Transfer Protocol Secure) and SSL (Secure Sockets Layer) etc. use ASN.1 widely.

Now, let’s design a formal notation for order, sales and delivery of products in Shop-Now application

**Design specification using ASN.1 formal notation for Shop-Now online ecommerce platform:**

Shop-Now DEFINITIONS AUTOMATIC TAGS ::= BEGIN

Order::= SEQUENCE {

customer Customer,

product Product,

delivery Delivery

}

Customer::= SEQUENCE {

cus\_name PrintableString (SIZE (1..50)),

cus\_address Address,

cus\_phone NumericString (SIZE (7..12)),

cus\_email PrintableString (SIZE (5..30)),

}

Address::= SEQUENCE {

street PrintableString (SIZE (1..50)) OPTIONAL,

city PrintableString (SIZE (1..30)),

zipCode NumericString (SIZE (5..9))

}

Product ::= SEQUENCE {

prod\_name PrintableString (SIZE (1..50)),

cost\_price NumericString (SIZE (1..15)),

manu\_date DATE OPTIONAL,

exp\_date DATE OPTIONAL,

category pro\_Category }

pro\_Category ::= CHOICE{

cat1 PrintableString ::= “Electronics”,

cat2 PrintableString ::= “Clothes”,

cat3 PrintableString ::= “Groceries”,

cat4 PrintableString ::= “Other”

}

Sales ::= SEQUENCE{

product\_details Product,

customer Customer,

payment\_method Pay,

delivery\_person DeliveryPerson,

delivery\_status BOOLEAN,

payment\_status BOOLEAN,

}

Pay ::= CHOICE {

cheque NumericString(SIZE (15)),

credit\_card Credit\_card,

cash NULL,

PayPal NULL,

}

Credit\_card ::= SEQUENCE {

type Card\_type,

number NumericString(SIZE (20)),

expiry-date DATE

}

Card\_type ::= ENUMERATED {

VISA(0), AMERICAN EXPRESS(1), MASTERCARD(2),

}

Delivery ::= SEQUENCE{

delivery\_person DeliveryPerson,

delivery\_address Address,

delivered\_date DATE

}

DeliveryPerson ::= SEQUENCE{

del\_name PrintableString (SIZE (1..50)),

del\_phone NumericString (SIZE (7..12))

}

END