

CHAPTER 1

INTRODUCTION






“Those who wise succeed must ask the right preliminary questions”. – Aristotle

In Computer Science, one of the core fields that belongs to its foundations, with the design, analysis, and implementation of algorithms for the efficient solutions of the problem concerned. The data structure is one of the subjects that intrinsically connected with the design and implementations of efficient algorithms.

The subject ‘Data Structure’ deals with the study of methods, techniques, and tools to organize or structure data in computer memory.

Now before defining Data Structure, we should know ~~what is data?~~ and ~~what is the difference between data and information?~~

KEY FEATURES

-  Data and Information
-  Data Structure
-  Data Type
-  Abstract Data Type
-  Classification of Data Structure

Data and Information

Data is a plural of datum, which is originally a Latin noun meaning ~~something given.~~

The Oxford dictionary meaning of data is:

- Facts or statistics used for reference or analysis.
- The quantities, characters or symbols operated by a computer.

For our purpose, the second meaning is more important. Therefore, we can say that:

The data represent quantities, characters, or symbols on which operations are performed by a computer, stored and recorded on either magnetic, optical, or mechanical recording media, and transmitted in the form of digital electrical signals.

Definition: Data is the basic entity or fact that is used in a calculation or manipulation process.

Data is commonly processed by some stages. Unprocessed data or raw data is a collection of numbers, characters, that may be considered as an input of a stage and processed data is the output of the stage.

There are two types of data, such as numerical and alphanumerical data. Data may be a single value or a set of values and it is to be organized in a particular fashion. This organization or structuring of data will have a profound impact on the efficiency of the program.

Most of the individuals consider that the terms "Data" and "Information" are interchangeable and mean the same thing. However, there is a distinct difference between the two words. Data are raw facts without context, whereas Information is data with context. Data are an abstraction of Information in a problem-solving system. Data requires interpretation to become an Information.

Data can be any character, text, words, number, pictures, sound, or video and if not put into context means nothing to a human or computer. For example, 10409 is a data, whereas information

may be 1/04/09-the date of birth of Avinaba, 10409 a zip code of somewhere or Rs. 10409 is the salary of someone.

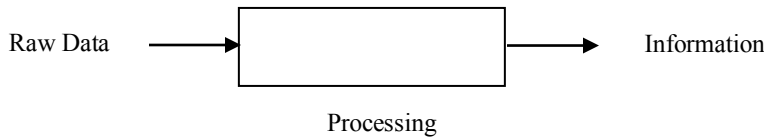


Figure 1.1: Data and Information

Representation of Data

Almost all the high-level languages, e.g. FORTRAN, COBOL, C, Pascal, C++, Java, etc. allow data to be represented in computer memory of mainly two types:

- i) Numerical Data
- ii) Alphanumeric Data

Most of the high-level languages classified numerical data into two types:

- i) Integer
- ii) Floating-point

In C language, the following Data types are used to represent numerical data.

Table 1.1: Ranges of numerical data types

Data types	Memory requirement	Ranges
char or signed char	1 byte	-128 to +127
unsigned char	1 byte	0 to 255
short signed int or short int or signed int or int	2 bytes	-32768 to +32767
long signed int or long int	4 bytes	-2147483648 to +2147483647
signed int or int (for 32 bit compiler)	4 bytes	-2147483648 to +2147483647
unsigned int	2 bytes	0 to 65,535
long unsigned int	4 bytes	0 to 4294967295
float	4 bytes	-3.4e38 to +3.4e38
double	8 bytes	-1.7e308 to +1.7e308
long double	10 bytes	-1.7e4932 to +1.7e4932

Integer

For small integer data, most of the high-level languages generally use two bytes to represent both positive and negative decimal integers. Negative integer numbers are represented by 2's complement notation. For large integer data, generally, four bytes are used to represent both positive and negative decimal integers.

Floating-Point

In the representation of small floating-point data, most of the high-level languages, e.g. C language, use four bytes for fractional decimal numbers. For large numbers, double and long double are used to accommodate in memory.

Alphanumeric data are classified into two types:

- i) Character
- ii) String

Character

The characters may be alphabets, digits, special characters and white spaces. In C language, characters are represented by char data type and one-byte memory space is used for storing the same. The ASCII format has been used in C language to represent the characters. While storing character `_C'` in the computer, ASCII value 67 is stored in memory.

String

The string is a sequence of characters may consist of any number and any combination of characters. The characters may be alphabets, digits, special characters and white spaces. In C language, the string can be defined as an array of character terminated with a null character.

Data Type

Generally, computer programs do exist for a single purpose: how to process data. The type of data, the format of data that is going to be returned and the correctness of the processing are the primary concerns of a computer program. When a program is written, how the computer handles the data internally is usually a secondary concern.

Definition: A data type refers to the type of data that variables hold.

Now, depending on the representation of different forms of data, different data types are used. The data types are names given to a set of variables, which have common properties. A data type refers to the type of data that variables hold.

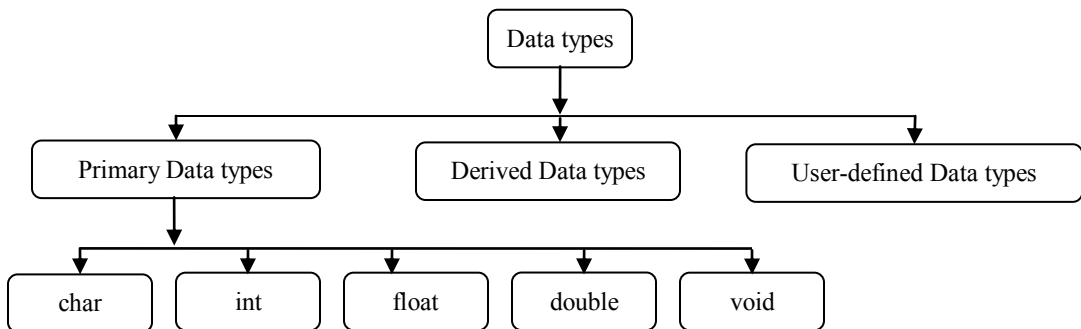


Figure 1.2: Data Types

The C language supports three classes of data types:

- i) Primary or Basic data types
- ii) Derived data types
- iii) User-defined data types

Basic or primary data types are language built-in. The C language supports five primary or basic data types. They are

- i) char
- ii) int
- iii) float
- iv) double
- v) void

Data Structure

Data Structure is the representation of the logical relationship between individual elements of data. In Computer Science, Data Structure is defined as a mathematical or logical model of organizing the data items into computer memory in such a way that they can be used efficiently. The purpose of studying

data structure is to learn how to organize data into memory so that it can be accessed quickly and conveniently.

Data Structure refers to the study of data and representation of data objects within the program; i.e., the implementation of structured relationships among different data objects.

Example: lists, stacks, queues, heaps, search trees, hash tables, etc.

Different sets of operations can be performed on different data structures. The operations that can be performed on different data elements within a data structure are accessing, traversing, inserting, deleting, modifying etc. While writing a program, a minimal data structure must be chosen that supports all the operations as per need.

Data Structure can be used for the following purpose:

- i) Organizing the data – How data items are organized in the main memory?
- ii) Accessing methods – How data items can be accessed?
- iii) Specifying the degree of associativity – How data items are interrelated?
- iv) Processing alternatives for data – How many different ways are there in which these data items can be processed?

A program is a set of instructions, which involve a computer performing some kind of computation or algorithm. Data Structure affects the design of both structural and functional aspects of a program. To implement a program of an algorithm, we should select an appropriate data structure for that algorithm. Therefore, the programs are inherited by an algorithm and its associated data structure.

Algorithm + Data Structure = Program

Programming languages provide facilities for representatives of algorithm and data. High-level Programming Language, like C, facilitates structured and modular programming by providing algorithm structures.

Classifications of Data Structure

Data Structure can be classified into two categories:

- i) Primitive data structure
- ii) Non-Primitive data structure

Primitive Data Structure

The primitive data structures are defined that can be manipulated or operated by the machine instruction. There are numerous types of data structures, generally built upon simpler primitive data types, called Primitive data structures, which are represented in computer memory.

Example: Integer, floating point, characters, pointer, boolean, etc. are some of the different primitive data structure.

In C language, the different primitive data structures are defined using the data type such as int, char, float, double etc.

Non-primitive Data Structure

The non-primitive data structures are a data structure that cannot be manipulated or operated directly by the machine instructions. These are more sophisticated data structures. Non-primitive data structures are derived from the primitive data structure. These are a group of homogeneous or heterogeneous data items.

Example: Arrays, structure, stack, queues, linked lists etc.

The Non-primitive data structures are classified into two categories:

- i) Linear data structure
- ii) Non-linear data structure

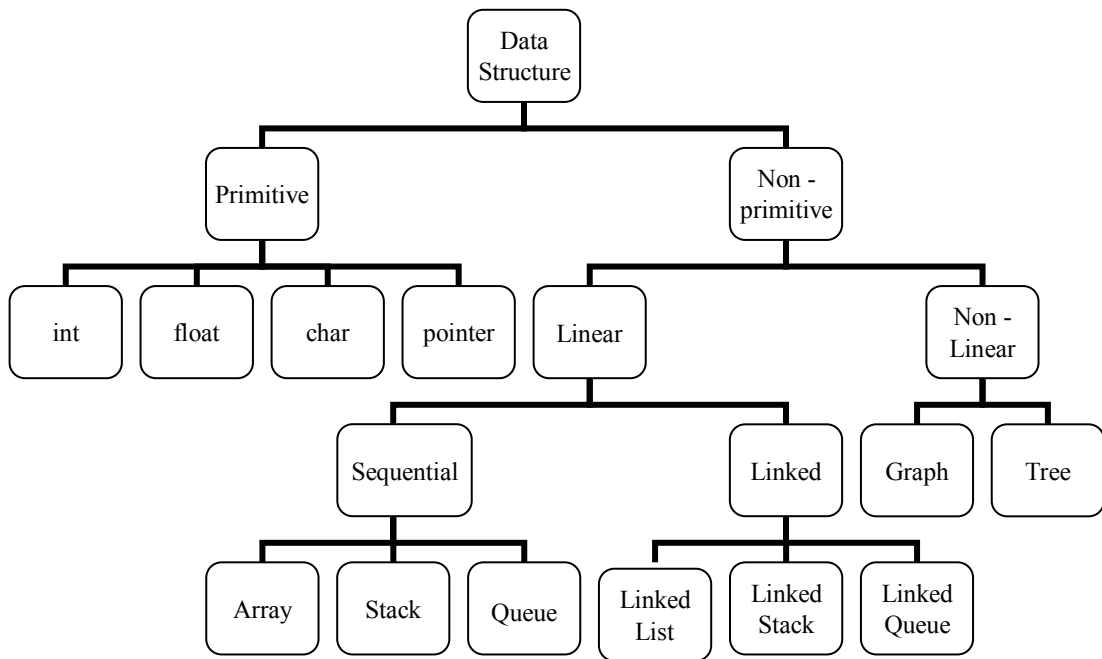


Figure 1.3: Classification of Data Structure

Linear Data Structures

The data structure is linear if every data item is related to its previous and next data items (e.g., array, linked list). In the linear data structure, data items are arranged in memory in a linear sequence and data items are accessed linearly. The traversing of the linear data structure is exactly once.

Linear data structures are two types:

- i) Sequential
- ii) Linked

Sequential Data Structures are based on arrays where objects are stored in a sequence of consecutive memory locations.

Example: Arrays, Stacks, Queues

Linked Data Structure is a data structure, which consists of a set of nodes linked together and organized with links.

Example: Linked Lists, Linked Stacks, Linked Queues

Non-linear Data Structures

A data structure is non-linear if every data item attaches to many other data items in specific ways to reflect relationships (e.g., tree). In non-linear data structures, the data elements are not in sequence, i.e., insertion and deletion are not possible in a linear manner. The traversing of the non-linear data

structure is always more than one.

Example: Graphs, Trees

Static Data Structure

The static data structure is a kind of data structure, in which once memory space is allocated it cannot extend, i.e. the memory allocation for the data structure takes place at compile-time that cannot be changed afterwards.

Example: Array

Dynamic Data Structure

Dynamic Data Structure is another kind of data structure, which can be extended or shrink during the execution, i.e., the memory allocation as well as memory de-allocation for the data structure takes place at run-time and allocates memory as required amount at any time.

Example: linked list, stack, queue, tree

Applications of Data Structure

Different Data Structure is used in real life, such as the representation of an image in the form of a bit-map, implement printer spooler so that jobs can be printed in the order of their arrival, store information about the directories and files in a system, etc. Data Structure is used in various fields of Computer Science, such as:

- Compiler Design
- Operating System
- Database Management System
- Statistical Analysis Package
- Numerical Analysis
- Graphics
- Artificial Intelligence
- Simulation

Different kinds of data structures are suitable for different kinds of applications. Some data structures are highly specialized for specific tasks. For example, databases use B-tree indexes for small percentages of data retrieval, and compilers and databases use dynamic hash tables as lookup tables, operating systems use queues for process management, I/O request handling.

Abstract Data Type

A data type refers to the type of data that variables hold. Thus, integer, real, characters are referred to as primitive data types.

Data Object represents an object having a data. The study of classes of objects whose logical behavior is defined by a set of values and a set of operations.

Definition: An Abstract Data Type (ADT) describes the data objects, which constitute the data structure and the fundamental operations supported on them.

An ADT promotes data abstraction and focuses on what a data structure does, rather than how it implements (does).

- An ADT is a conceptual model of information structure.
- An ADT specifies the components, their structuring relationships and a list of operations that

are allowed to perform.

- It is just a specification, no design or implementation information is included.
- Specification involves the “what” the operations, not the “how”.
- ADT’s are generalizations of primitive data types.

A data structure is the **design representation** of an ADT.

- The same ADT may be represented by several data structures.
- There are many data structures corresponding to the ADT “set”.

Operations Perform on Data Structure

Data are processed by means of certain operations, which appear in the data structure. The particular data structure is chosen largely depends on the frequency of the operation that needs to be performed on the data structure. Different kinds of operations are to be performed on data structures.

Table 1.2: Operations on Data Structures

Operation	Description
Creation	Allocation of memory for the data structure, the creation of data structure may take place either during compile-time or during run-time.
Insertion	Insert a data item in the data structure.
Deletion	Delete a data item from the data structure.
Traversing	Accessing and processing each data item of the data structure exactly once.
Searching	Find the location of the key value within the data structure.
Sorting	Arranging all the data items in a data structure either in ascending or in descending order or in lexicographical order (for Strings).
Merging	Combining the data items of two different sorted lists into a single sorted list.

Overview of Different Data Structures

Different data structures can be defined as follows:

Array

An array is a collection of the same type of data items, which are stored in consecutive memory locations under a common name. In arrays, there is always a fixed relationship between the addresses of two consecutive elements as all the items of an array must be stored contiguously.

Stack

A stack is a collection of elements into which new elements may be inserted and from which elements may be deleted only at one end called the top of the stack. Since all the insertion and deletion in a stack is done from the top of the stack, the last added element will be first to be removed from the stack. That is the reason why stack is also called Last-In-First-Out (LIFO) data structure.

Queue

A queue is a homogeneous collection of elements in which deletions can take place only at the front end, known as dequeue and insertions can take place only at the rear end, known as enqueue. The element, which inserts in the queue first, will delete the queue first. In this order, a queue is called First-In-First-Out (FIFO) system.

Linked List

A linked list is an ordered collection of finite homogeneous data elements called node where the linear order is maintained by means of links or pointers. In linked list, data items may be scattered arbitrarily all over the memory. In a linked list, there is no relationship between the addresses of elements; each element of a linked list must store explicitly the address of the element, next to it.

Tree

The tree is a non-linear data structure. A Tree may be defined as a non-empty finite set of nodes, such that

- i) There is a specially designated node called the root,
- ii) The remaining nodes are partitioned into zero or more disjoint trees $T_1, T_2, \dots T_n$ are called the subtrees of the root R.

Graph

The graph is another non-linear data structure. A Graph G is defined as an ordered set $G = (V, E)$, consists of finite non-empty set of objects V , where $V(G) = \{v_1, v_2, v_3, \dots v_n\}$ called vertices (or nodes or points) and another set E where $E(G) = \{e_1, e_2, e_3, \dots e_m\}$ whose elements are called edges, that connects these vertices.

Summary

- Data is the basic entity or fact that is used in the calculation or manipulation process.
- Data are raw facts without context, whereas information is data with context.
- Data Structure is defined as a mathematical or logical model of the particular organization of data items in computer memory so that it can be used efficiently.
- An Abstract Data Type (ADT) describes the data objects, which constitute the data structure, and the fundamental operations supported on them.

Exercises

1. What are the differences between linear and non-linear data structures?
2. What are the operations can be performed on data structures?
3. What is an Abstract Data Type? What do you mean by a Dynamic Data Structure?
4. Choose the correct alternatives for the following:
 - i) Which of the following data structure is a linear data structure?
a) Trees b) Graphs c) Arrays d) None of these
 - ii) The operation of processing each element in the list is known as
a) Sorting b) Merging c) Inserting d) Traversal
 - iii) Finding the location of the element with a given key in the list is known as
a) Traversal b) Searching c) Sorting d) None of these
 - iv) Representation of data structure in memory is known as
a) Recursion b) Abstract data type c) Storage structure d) File structure
 - v) An ADT is defined to be a mathematical model of a user-defined type along with the collection of all _____ operations on that model
a) Cardinality b) Assignment c) Primitive d) Structured
