Course Code:	**CSC 313**
Course Title:	**Data Structure and Algorithms**
Course Unit:	**3**

Course Developer/Writer: **A. J. Ikuomola**
&
Dr. A.T. Akinwale

Department of Computer Science
College of Natural Science
University of Agriculture Abeokuta,
Ogun State, Nigeria

UNIT 1: MATHEMATICAL NOTATION AND FUNCTION

Summation Symbol (Sum)

$\sum$ Called Summation (Sigma)

Consider a sequence of a_1, a_2, a_3,… Then the sums

$a_1 + a_2 + a_3 + ... + a_n$ and $a_{m1} + a_{m+1} + \ldots + a_n$

will be denoted respectively by

$$\sum_{j=1}^{n} a_j \quad \text{and} \quad \sum_{j=m}^{n} a_j$$

Example:

(1) $$\sum_{i=1}^{n} a_i = a_1 + a_2 + a_3 + a_4 + ... + \ldots + a_n$$

(2) $$\sum_{i=1}^{n} a_i b_i = a_1 b_1 + a_2 b_2 + a_3 b_3 + a_4 b_4 + ... + \ldots + a_n b_n$$

(3) $$\sum_{j=2}^{5} j^2 = 2^2 + 3^2 + 4^2 + 5^2 = 4 + 9 + 16 + 25 = 54$$

(4) $$\sum_{j=1}^{n} j = 1 + 2 + 3 + 4 + ... + n$$

PIE (Product)

$$\prod_{i=1}^{n} x_i = x_1 . x_2 . x_3 ... x_n$$

Floor Function

Let x be any real number, then x lies between two integers called the floor and the ceiling of x. Specifically,

$\lfloor x \rfloor$, called the floor of x denotes greatest integer that does not exceed x.

Examples:

(1) $\lfloor 3.14 \rfloor$ = 3

(2) $\lfloor \sqrt{5} \rfloor$ = 2.23 = 2

(3) $\lfloor -8.5 \rfloor$ = -9

(4) $\lfloor 7 \rfloor$ = 7

Ceiling Function

The symbol for ceiling function is $(\lceil\ \rceil)$ called the ceiling function of x denotes the least integer that is not less than x.

Example:

(1) $\lceil 3.14 \rceil$ = 4

(2) $\sqrt{5}$ = 2.23 = 3

(3) $\lceil -8.5 \rceil$ = - 8

(4) $\lceil 7 \rceil$ = 7

Remainder Function: Modular Arithmetic

Let K be any integer and let M be a positive integer. Then

$$k \pmod{M}$$

(read k modulo M) will denote the integer remainder when k is divided by M. More exactly k (mod M) is the unique integer r such that

$$k = Mq + r \quad \text{when } 0 < r < M$$

When k is positive, simply divide k by M to obtain the remainder r.

Example:

(1) 25(mod7)

25/7 = 3 r 4

25(mod7) = 4

(2) 25(mod5)

25/5 = 5 r 0

25(mod5) = 0

(3) 35(mod11)

35/11 = 3 r 2

35(mod11) = 2

(4) 3(mod8)

3/8 = 3 r 4

3(mod8) = 3 (note that 3 = 8 . 0 + 3 = 3) when q= 0

UNIT 2: DATA STRUCTURE

Introduction

Data structure is a particular way of storing and organizing data in a computer so that it can be used efficiently. Data structure is the logical arrangement of data element with the set of operation that is needed to access the element. The logical model or mathematical model of the particular organization of data is called a data structure. It is defined as a set of rules and constraint which shows the relationship that exist between individual pieces of data which may occur.

Basic Principle

Data structures are generally based on the ability of a computer to fetch and store data at any place in its memory, specified by an address – a bit string that can be stored in memory and manipulated by the program. Thus the record and array data structures are based on computing the addresses of data items with arithmetic operations; while the linked data structures are based on storing addresses of data items within the structure itself. Many data structures use both principles.

The choice of a data structure for a particular problem depends on the following factors:

1) Volume of data involved
2) Frequency and ways in which data will be used.
3) Dynamic and static nature of the data.
4) Amount of storage required by the data structure.
5) Time to retrieve an element.
6) Ease of programming.

Classification of Data Structure

(1) Primitive and non – primitive: primitive data structures are basic data structure and are directly operated upon machine instructions. Examples are integer and character. Non-primitive data structures are derived data structure from the primitive data structures. Examples are structure, union and array.

(2) Homogenous and Heterogeneous: In homogenous data structures all the elements will be of the same type. Example is array. In heterogeneous data structure the elements are of different types. Example: structure

(3) Static and Dynamic data structure: In some data structures memory is allocated at the time of compilation such data structures are known as static data structures. If the allocation of memory is at run-time then such data structures are known as Dynamic data structures. Functions such as malloc, calloc, etc. are used for run-time memory allocation.

(4) Linear and Non – linear data structure: Linear data structure maintains a linear relationship between its elements. A data structure is said to be linear if its elements form a sequence or a linear list. Example, array. A non-linear data structure does not maintain any linear relationship between the elements. Example: tree.

Linear structure can be represented in a memory in 2 basic ways:

i) To have the linear relationship between the element represented by mean of sequential memory location. These linear structures are called ARRAY.
ii) To have the linear relationship between the elements represented by means of points or links. These linear structures are called LINKLIST.

Data Structure Operation

The following operations are normally performs on any linear structure, whether is an array or a linked list.

- Transversal (Traversing)
- Search (Searching)
- Inserting
- Deleting
- Sorting
- Merging

Transversal/Transversing: accessing each element or record in the list exactly only, so that certain items in the record may be processed. This accessing and processing is sometimes called "visiting" the record.

Search (Searching): finding the location of the record with a given key value or finding the location of all records which satisfy one or more conditions.

Inserting: adding a new record to the structure

Deleting: removing an element from the list of records from the structure.

Sorting: arranging the record in some logical order (e.g. alphabetically according to some NAME key or in numerical order according to some NUMBER key such as social security number, account number, matric number, etc.)

Merging: combining the records in two different sorted file into a single sorted file.

Characteristics of Data Structures

Data Structure	Advantages	Disadvantages
Array	Quick inserts Fast access if index know	Slow search Slow deletes Fixed size
Ordered Array	Faster search than unsorted array	Slow inserts Slow deletes Fixed size
Stack	Last-in, first-out access	Slow access to other items
Queue	First-in, first-out access	Slow access to other items
Linked List	Quick inserts Quick deletes	Slow search
Binary Tree	Quick search Quick inserts Quick deletes (if the tree remains balanced)	Deletion algorithm is complex
Red-Black Tree	Quick search Quick inserts Quick deletes (Tree always remains balanced)	Complex to implement
2-3-4 Tree	Quick search Quick inserts Quick deletes (Tree always remains balanced) (Similar trees good for disk storage)	Complex to implement
Hash Table	Very fast access if key is known Quick inserts	Slow deletes Access slow if key is not known Inefficient memory usage
Heap	Quick inserts Quick deletes Access to largest item	Slow access to other items

Graph	Best models real-world situations	Some algorithms are slow and very complex