Unit-II.

Introduction to Algorithm, Data structures & Analysis of algorithms

[7 Hrs]

A) Introduction to Data Structures: Concept of data, Data object, Data structure, Abstract Data Types (ADT), realization of ADT in 'C'.

B) Concept of Primitive and non-primitive, linear and Non-linear, static and dynamic, persistent and ephemeral data structures.


A) Introduction to Data Structures:

Few Important Concepts:

What is Data?

Data is nothing but value or set of values.

Whatever variable we will declare in computer programming is considered as Data.

E.g.

```
int x = 20;
char ch = 'a';
int arr[10] = {1,2,3,4,5,6,7,8,9,10};
```
Here, x is a variable which contains data or value 20.

ch is a variable which contains data or value ‘a’. arr is a array variable which contains data or set of values as 1,2,3,4,5,6,7,8,9,10.

**Data Item**: Data item is a single unit of values or smallest part of values. In above example x is a data item, ch is a data item, arr is a collection of data items.

**Group Item**: Data items which are divided into sub items are called as group items. The data item which is build of another data items is called as grouped Item.

  e.g. Name of a person gets divided into First name, Middle name, Last name.

```
Name
  ↓
First Name Middle Name Last Name
```

Here, Name field is divided into three sub data items.

**Elementary Items**: Data items which can not get sub divided in to another data items are called as elementary items. E.g. roll number of a student never gets divided into sub items.

**Data Object**: Data Object is a collection of data items.

**Data Structure**: Data means value or set of values.

  Structure means way of representing data (i.e. Collection of data and Operations performed on data).
**Definition:**

A data structure is a collection of Data objects (D), a set of functions (F) or operations performed on data (i.e. Add, Subtract, multiply, Divide etc) and a set of rules or Axioms (A).

For e.g.

**Data Structure** = {D, F, A}

D = Data objects.

F = Functions performed on Data objects

A = Axioms i.e. rules to implement operations on data

D = {1, 222.22, 333, "Hello"}

F = {ADD, MULT, SUB, DIV}

A = {set of rules}

"Data structure is a branch of computer science which gives the knowledge of how data should be organize (i.e. Systematic arrangement of data), how flow of data should be control, how data should be design and implement to reduce the complexity (i.e. time and space complexity) and increase the efficiency of algorithm".

- Data Structure provides a way of storing and retrieving data to and from computer memory effectively.
- An organization of information, usually in memory.
- Computer science concentrates on the study of how data organization and how flow of data is control in computer system.
- Data structure introduces concepts related to Object
Oriented Programming like Class, Abstract Data Type (ADT).

**Needs of Data Structure:**

1) Data Structure provides a efficient way for storing and retrieving data to and from computer memory.

2) To handle large amount of data efficiently.

3) Data structure gives the knowledge of –
   - How to organize data?
   - How to control flow of data?
   - How to design and implement efficient Data Structure?
   - How to reduce complexity (i.e. space complexity and Time complexity) of algorithm?
   - How to increase efficiency of algorithm?

There are so many data structures like Array, Stack, Queue, Link List, Trees and Graphs. Each data structure having there own efficiency, so programmer have to decide which data structure to use.

e.g. Array:

**Use:** Array is used to store data temporarily in computer memory for computer programming.

**Advantages of array:**

(i) Storing and retrieving data to/from array becomes faster.

(ii) Random accessing of data becomes possible.

**Disadvantages of array :**

i) Array is used to store fix amount of data. We can not store more data than array size.

ii) If we store less amount of data in to array than array size, then remaining memory allocated to array gets wasted.

Prepared By: Sandip M.Walunj, Lecturer in Computer Engg. Dept, Nashik
iii) Insertion and deletion of data elements to/from array is complex. It requires shifting of data.

iv) Array allocates contiguous memory locations. Due to that even though sufficient memory available to store data but it is not contiguous. So that memory will be wasted.

- Data structure helps you to understand concepts of data abstraction, information hiding and data encapsulation.

  Data Structure introduces the concepts of object oriented Programming languages.

Example :- Concept of class in c++ is based on Abstract Data Type (ADT).

5) Data Structure helps you to analyze problem step by steps.

6) Develop algorithms to solve real world problems.

7) Data Structure provides different technique for sorting and searching, handling large amount of data.

  **Searching Techniques** : Linear search, Binary Search, Hashing

  **Sorting Techniques** : Bubble Sort, Selection Sort, Insertion Sort, Radix Sort, Heap Sort, Quick Sort, Merge Sort, Shell Sort, Binary tree sort.

- Data structure will maintain relationship of one data element with other data elements in memory, set of operation possible and set of rules to apply on data stored in data structure.

B) Concept of Primitive and non primitive, linear and Non-linear, static and dynamic, persistent and ephemeral data structures.
Data Types:

- User Defined Types:
  - Struct
  - Enum
  - Union
  - Class

- Built-in Type:
  - Integral Type
  - Void
  - Floating Type
    - Float (4 bytes) 3.4E-38 to 3.4E+38
    - Double (8 bytes) 1.7E-308 to 1.7E+308
    - Long double (10 bytes) 3.4E-4932 to 1.1E+4932

- Derived Type:
  - Array
  - Function
  - Pointer

- Integer Types:
  - signed short int (1 byte) (Range from -128 to 127)
  - signed int (2 bytes) (Range from -32768 to 32767)
  - signed long int (4 bytes) (Range from -2,147,483,648 to 2,147,483,647)
  - unsigned short int (1 byte) (Range from 0 to 255)
  - unsigned int (2 bytes) (Range from 0 to 65535)
  - unsigned long int (4 bytes) (Range from 0 to 4,294,967,295)

Fig. (d) Data Types

Prepared By: Sandip M. Walunj, Lecturer in Computer Engg. Dept, Nashik
Definitions :-

- Data type is a term used to describe the information type that can be processed by computer and which is supported by programming languages.

- A term which refers to the kind of data that variables may hold in a programming languages.

- A method of interpreting a bit pattern is called a data type.

- A data type is a collection of values and operations on values.

- A data structure is an abstract concept defined by a set of logical properties.

Why need of data types?

Suppose, we can think of a universal data type that may hold any value like character, integer, float or any complex number.

Disadvantages of using such data type:

1) Large amount of memory will be occupied for small size of data.

2) Different type of data requires different interpretation of bit strings.
Data Structures are classified in following main types:

1) Primitive and non primitive
2) Linear and Non-linear
3) Static and dynamic
4) Persistent and ephemeral data structures.

1) **Primitive and non primitive**:

   **Primitive Data Structures**:

   1. Primitive data structures are the data structures provided by a programming language as basic building blocks. Primitive data structure are also known as built-
in data structures or basic data structures or fundamental data structures.

2. Operations performed on primitive data structures are faster than non primitive data structure.

3. Example :- Primitive data structures are int, char, float, double etc.

Non - Primitive Data Structures :

- The data structures which are derived from the primitive data structures are called as non-primitive data structures.

- Non Primitive data structure concentrates on making collection of or structuring of group of similar or different data items.

- Example: Non primitive data structures are Lists and files.

Lists are divided in to linear list or linear data structure and non linear lists or non linear data structures.

2) Linear and Non-linear:

Linear Data Structures:

A data structure is said to be linear if the data elements stored in these data structure forms a sequence. e.g. Array, Linked list, queue, stack, strings etc. Linear data structure is one in which, while traversing sequentially, we can reach to only one element directly from another. e.g. Link List, Array.

In Linear data structure a relationship among the data elements is one dimensional. Such data structures are good to represent one to one relationship among data elements but difficult to represent one to many relationship. So it is clear that to represent complex data which is having one to many or many to many relationship using linear data structures. In link list each node has a link which points to another node.

2-D array, though seems to be non-linear, is actually linear data structure. This is because memory is single
dimensional and when it is stored in the memory it is stored as a single dimension array in either row-major or column-major format. Similarly all multi-dimensional arrays are also linear, for the same reason.

There are two ways of representing linear data structures in memory. One way is to have the linear relationship between the elements by means of sequential memory locations. Such linear structures are called arrays. The other way is to have the linear relationship between the elements represented by means of links. Such linear data structures are called linked list.

**Non Linear Data Structures:**

A data structure is said to be non linear if the data elements stored in these data structures do not form a sequence. Elements stored in a nonlinear data structure do not form a sequence. A non linear data structure is one in which, while traversing sequentially, we can reach more than one element directly from another.

**e.g.** Hash tables, Trees (Binary tree, B+ tree, B trees, AVL tree) and graphs etc. We can represent non linear data structure using array and link lists.

**3) Static and dynamic:**

**Static Data Structure:**

The data structures to which memory gets allocated during the time of program execution, such data structures are called as static data structures. **e.g. Array**

**Dynamic Data Structures:**

The data structures to which memory gets allocated during the time of program execution is called dynamic data structures.

**4) Persistent and ephemeral data structures:**

**Persistent Data Structures:**

A data structure is called as persistent data structure because a change to the data structure does not destroys the old
version, it creates new version and maintains previous version. Persistent data structure allows access to any version, old or new, at any time.

We shall call a data structure persistent if it supports access to multiple versions. The data structure is partially persistent if all versions can be accessed but only the newest version can be modified, and fully persistent if every version can be both accessed and modified.

**Ephemeral data structures:**

A data structure is called as ephemeral data structure because a change to the structure destroys the old version, leaving only the new version available for use.

**C) Algorithm:**

**Definition:**

"An algorithm means writing a sequence of instructions which acts on input data and produce desired output in a finite number of steps for solving any problem."

To solve any problem it is necessary to write algorithm.

- A way of solving problem step by step is called algorithm.
- Algorithm is a step by step execution of particular problem. It works on given data and produces desired output within finite number of steps.
- Algorithm gives you a stepwise representation of how to solve problem.
- There are two types of algorithms.
  1) **Iterative algorithm or Repetitive**: Iterative algorithm make use of loops and conditional statements.
  2) **Recursive algorithm**: Recursive algorithm uses a Divide and Conquer strategy. As per this strategy the recursive algorithm breaks down a large
problem into small pieces and then applies the algorithm to each of these small pieces.

Recursive algorithm is small, straightforward and simple to understand. When a function calls itself in its body then such function is called as recursive function.

Example:

```c
void display()
{
    display();
}
```

here, display function is a recursive function because it calls itself in its body.

Properties/Characteristics of Algorithm:

Every algorithm must have following five properties:

**Input** :- Algorithm must have some input.

**Output** :- Algorithm must produce desired output.

**Finiteness** :- Algorithm must terminate after finite or fixed number of steps.

**Definiteness** :- Steps to be performed in the algorithm must be clear and unambiguous.

**Effectiveness** :- Algorithm should be efficient (having less time and space complexity).

Prepared By: Sandip M.Walunj, Lecturer in Computer Engg. Dept, Nashik
Analysis of algorithm:

Relation between Data Structure and algorithm:

Abstract Data Type (ADT):
- ADT provides useful guidelines to implement data structure.
- ADT is a useful tool for specifying the logical properties of data structure.
- An ADT is defined as a mathematical model of the data objects that make up a data type as well as the functions that operate on these objects.
- ADT is the specification of logical and mathematical properties of data type or data structure.
- The term Abstract Data Type (ADT) refers to the basic mathematical concept that defines data type.
- Defining ADT is not concern with implementation or coding details of data type.
- For defining ADT we are not concern with Space and Time complexity.
- Abstract Data type is a collection of values and set of operations on those values.

The important steps in defining ADT are as follows:

i) State what data & describe the way in which data elements are related to each other.
ii) State possible operations on that data type or data structure.

Process to Define Abstract Data Type:

An ADT consists of two parts:

a) Value Definition Part:

It tells what value is stored in data structure and what relationship between data elements. It contains following two things.

i) Definition Clause: In definition clause we have to declare data with data type which gets stored in data structure. This clause is required compulsorily.
ii) Condition Clause: In condition clause we have to write any
primary condition if required. This clause is optional.

b) Operator Definition Part:
It only tells what operations or functions to be performed on data stored in data structure. It does not tell how these operations takes place.
It contains following three sections,

i) Header: Header specifies name, data type of data passed and return data type of function.

ii) Precondition:
Precondition tells, is there any primary condition required or not. It is optional.

iii) Post condition: Final result generated.

Example 1: Write Abstract Data Type for Rational number.

Solution:
Rational number means a number which is in the form of Numerator and Denominator i.e. a/b. For writing ADT of any data structure we have to give specification of data and list of operations performed on data stored in the data structure. Here you have to write whatever possible operations on data. Such as makerational, addition of two rationals, multiplication of rationals, division, subtraction and any more. We can write ADT for rational number as follows:

```c
/* Value definition */
abstract typedef <integer , integer> Rational;
Condition : Rational [1] != 0;
/*Operator definition */
abstract Rational makerational(a,b)
int a,b;
precondition : b !=0
Postcondition : makerational [0] = = a;
              makerational [1] = =b;
/*Perform addition of two rational number suppose a & b are two
```
rational numbers.
Here, \( a = \frac{a_0}{a_1} \) and \( b = \frac{b_0}{b_1} \). Now we can add two rational numbers:

\[
\left( \frac{a_0}{a_1} \right) + \left( \frac{b_0}{b_1} \right) = \frac{(a_0*b_1) + (a_1*b_0)}{(a_1*b_1)}
\]

abstract Rational add(a,b)
Rational a,b;
Postcondition:

\[
\text{add}[0] = \frac{a[0] * b[1] + a[1] * b[0]}{1}; \\
\text{add}[1] = \frac{a[1] * b[1]}{1};
\]

/* Multiplication of two rationals i.e. \( a * b \). Here \( a = \frac{a_0}{a_1} \) & \( b = \frac{b_0}{b_1} \).

\[
\left( \frac{a_0}{a_1} \right) \times \left( \frac{b_0}{b_1} \right) = \frac{(a_0*b_0)}{(a_1*b_1)}
\]

abstract Rational mult(a,b)
Rational a,b;
Postcondition:

\[
\text{mult}[0] = \frac{a[0] * b[0]}{1}; \\
\text{mult}[1] = \frac{a[1] * b[1]}{1};
\]

Example 2: Define ADT for Strings.

/* value definition */
abstract typedef < <char> > String; // here String is a abstract data type of

// variable length of characters

/* operator definition */
abstract length(s)               //here length function
calculates length of string

String s;
Postcondition : length = = len(s);

/* operator definition */
abstract  String concat(s1,s2)
String s1,s2;
Postcondition :
            concat = = s1 + s2;

Explanation of example 2:

Here, keyword abstract typedef introduce a value
definition. In operator definition section each operator is
defined as an abstract function with three parts a header,
onoptional precondition and post condition.

Example 3: Abstract Data Type for Stack (ADT)

We know that a stack is a finite set of elements, where
insertion or removal of an element is allowed from one end only.
Any new element join the set is kept at top and only the element
at the topmost position can be taken out.

abstract typedef<1,2,3,........,n,top> STACK;
Condition : top == NULL

Abstract STACK  PUSH( data)
int data;
Precondition : top != n
Postcondition :
            insert(data)
            STACK[top] =  data

Prepared By: Sandip M.Walunj, Lecturer in Computer Engg. Dept, Nashik
/*top points to most recently inserted data */

abstract pop()

precondition : top ! = NULL

postcondition :

    Remove(top)

    Data = STACK[top]