1.1 Concept of Data structure

- Concepts
  - group of Data elements.
  - predefined way of storage and organization.
  - storage are optimized for efficiency.

Some common examples of data structures are arrays, linked lists, queues, stacks, binary trees, hash tables tries and so on.

● Primitive and Non-primitive data structures

Primitive = fundamental data types, supported by almost all programming languages.
= also called basic data type, primitive data type.

For example - integer, real, character, boolean etc.

Non-Primitive = build by combining primitive data structures.

For example - linked lists, stacks, trees, and graphs.

● Linear and Non-Linear data structures

Linear = stored in a sequential order.
= means, in a sequential memory locations.
= means, has linear relationship between data elements.

For example - linked lists, stacks, queues are linear data structure.

Non-Linear = elements are not in sequential order.

For example - hash table, trees, graphs.

1.2 Abstract Data Type

Abstract = corresponds to the hiding of detail implementation.

Data Type = type representation of a variable, holds set of values.

ADT = focuses on what to work for, not how it works.

= separates the implementation details among multiple data stores.

Example: A stack has

= sequential storage.

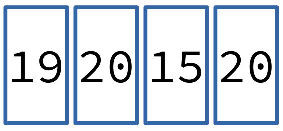
= push and pop operation attached.

= last in first out policy.

1.3 Arrays, Structure, Union, Class, Pointer

Arrays = similar elements, sequential.

```
int ages[4] = {19, 20, 15, 20};
```



16 bytes of information.

Structure = group of basic/user-defined elements.

```
struct Subject {
    char name[64];
    int mark;
};

struct Student {
    int crn;
    char name[100];
    Subject subjects[NO_OF_SUBJECTS];
};
```

Union = group of elements, shared common memory.

```
struct Account {
     char name[64];
     int type;
     union {
          struct Saving saving;
          struct Current current;
          struct Fixed fixed;
     } accountDetails;
 };
```

Pointer = dynamic allocation/call, operations achieved with address arithmetic.