UNIT-I

DATE / /
PAGE No.

DATA STRUCTURE

It is a particular way of organizing and storing data in a computer. So that it can be accessed and modified efficiently. A data structure will have a collection of data and functions or operations that can be applied on the data.

Data type

- *Primary
  - → Integer
  - → Character
  - → Boolean
  - → Floating Point
  - → Double floating Point
  - → Void
  - → Wide character
- *Derived
  - → Function
  - → Array
  - → Pointer
  - → Reference
- *User defined
  - → class
  - → Structure
  - → Union
  - → Enum
  - → Typedef

1) Primitive Data Types :→ These data types are built in or predefined data types and can be used directly by the user to declare variables. example : int, char, float, bool, etc. Primitive data types available in C++ are:

- Integer
- Character
- Boolean
- Floating Point
- Double floating Point

- Valueless or Void
- Wide character

2) Derived Data Types :→ Derived data types that are derived from the primitive or built-in datatypes are referred to as Derived Data Types. These can be of four types namely:
- Function
- Array
- Pointer
- Reference

3) Abstract or User-Defined Data Types :→ Abstract or User-Defined data types are defined by the user itself. Like, defining a class in C++ or a structure. C++ provides the following user-defined datatypes:
- Class
- Structure
- Union
- Enumeration
- Typedef defined Datatype

* Need for data structure :→ Data structures are essential for two main reasons: they make the code more efficient and they make the code easier to understand.

When it comes to efficiency, data structures help the computer to run the code faster by organizing the data in a way that is easy for the computer to process. For example, a linked list is a data structure that can store a list of items. The computer can quickly add or remove items form a linked list without searching through an entire array of data.

Data structures also make code easier to understand because they provide a way to organize data logically. By using data structures, developers can create easy-to-read and debug code. For example, developers can use an array to store a list of items in a specific order. This makes it easy for other developers to see how the data is used and what operations are performed.

* Types of Data Structure :→

1) Primitive and Non-Primitive Data structure
2) Linear and Non-Linear Data structure
3) Static and Dynamic Data structure
4) Sequential and Direct Data Structure

1) Primitive and Non-Primitive Data structure

Data structure defines a set of primitive

elements that do not involve any other elements as its subparts. They are generally built in data type in programming languages. Eg → Integers, Characters etc
Non-Primitive Data Structures are that defines a set of derived elements such as Arrays, Structures and Classes.

2) Linear and Non-Linear Data Structures:→

A Data Structure is said to be linear, if its elements form a sequence and each element have a unique successor and predecessor. Eg → Stack, Queue etc
Non-linear Data Structures are used to represent data that have a hierachical relationship among the elements. In non-linear Data Structure every element has more than one predecessor and one successor. Eg → Trees, Graphs.

3) Static and Dynamic Data structure:→

A data structure is refferred as static Data structure, if it is created before program execution. i.e, during compilation time. The variables of static Data structure have user specified name. Eg → Array
Data structures that are created at run time are called as Dynamic Data structure. The variables of this type are known always referred by user defined name instead using their addresses through

pointers. Eg → Linked List

4) Sequential and Direct Data Structures :→ This classification is with respect to access operation associated with the data types. Sequential access means the locations are accessed in a sequential form. Eg → To access n^{th} element, it must access preceding $n-1$ data elements. Eg → Linked List

Direct Access means any element can access directly without accessing its predecessor or successor ie, n^{th} element can be accessed directly. Eg → Array