E- Lecture
Data Structures and Algorithms

By
H R Choudhary (Asstt. Professor)
Department of CSE
Engineering College Ajmer

Topics to be covered
Introduction to Data Structure

What do you mean by data ?

What is structures ?

What is Data structures ?

Need of Data structures.

Types of data Structures.

Different operations on data structures.

What is an Algorithm ?

What is Data ?

Data can be defined as a representation of facts, concepts, or instructions in a formalized manner, which should be suitable for communication, interpretation, or processing by human or electronic machine.

Data is represented with the help of characters such as alphabets (A-Z, a-z), digits (0-9) or special characters (+,-,/,*,<,>,= etc.)

What is Data ?

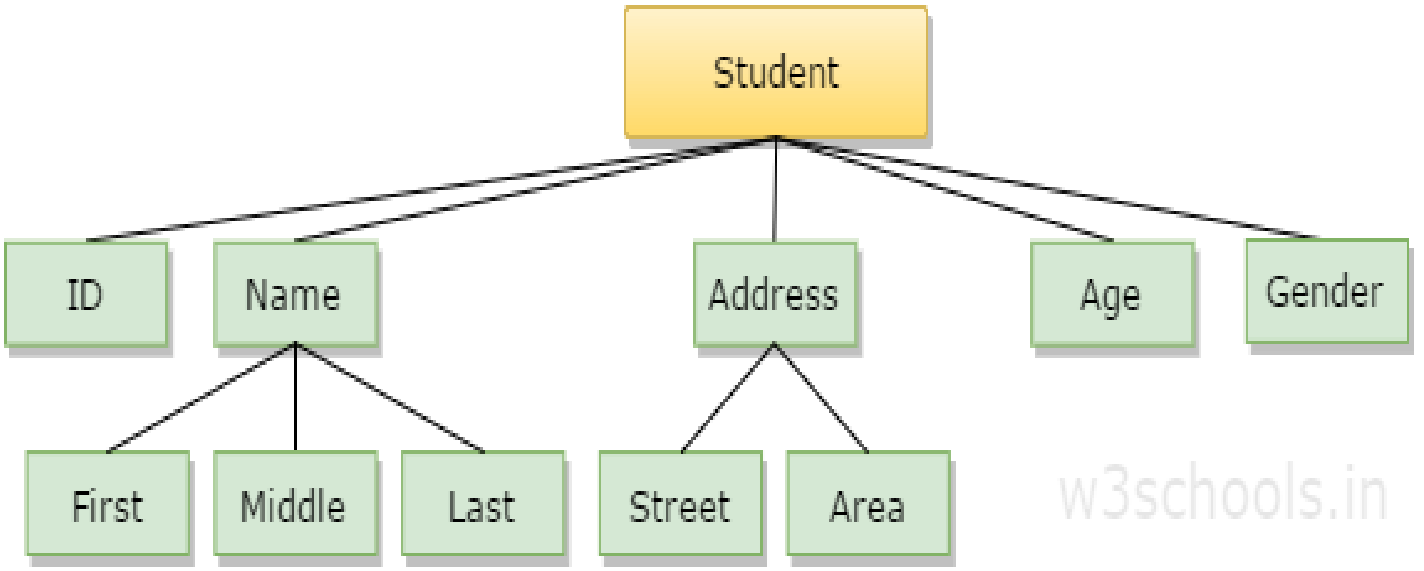
Data are just a collection of facts and figures, or you can say data are values or a set of values that are in a particular format.

A data item refers to a single set of values. Data items are then further categorized into sub-items, which are the group of items that are not being called a plain elementary form of items.

Let us take an example where the name of the student may be divided into three sub-items, namely: first name, middle name, and last name. But the ID that is assigned to a student would typically be considered as a single item.

What is Data ?

Example: In the example mentioned above, such as ID, Age, Gender, First, Middle, Last, Street, Area, etc. are elementary data items,whereas (Name, Address) is group data items.



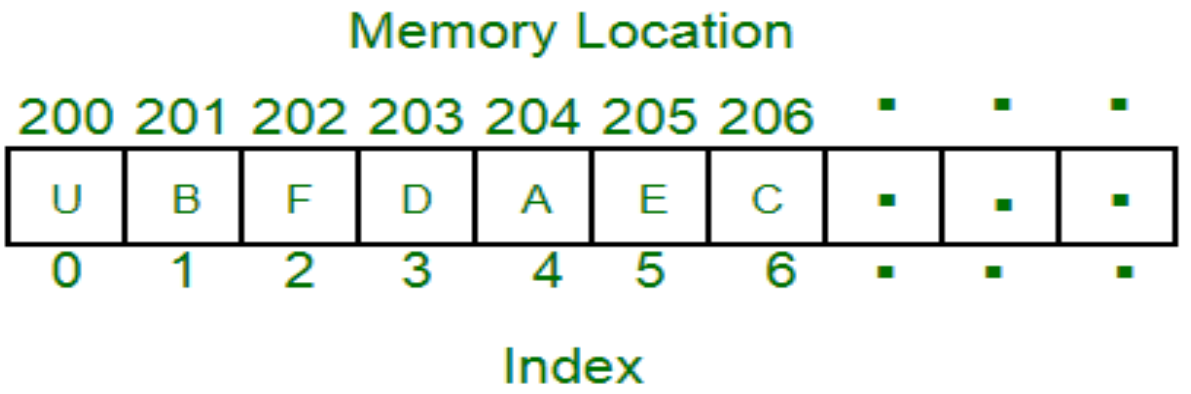
What is Structure?

A structure is an arrangement and organization of interrelated elements in a material object or system, or the object or system so organized.

What is Data structures

A data structure is a particular way of organizing data in a computer so that it can be used effectively.

For example, we can store a list of items having the same data-type using the array data structure.



What is Data structures

Other definitions:

- A data structure is a data organization, management, and storage format that enables efficient access and modification.
- A data structure is a collection of data values, the relationships among them, and the functions or operations that can be applied to the data.
- A data structure is a specialized format for organizing, processing, retrieving and storing data.
- Data Structure can be defined as the group of data elements which provides an efficient way of storing and organising data in the computer so that it can be used efficiently

What is Data structures

WHAT IS DATA STRUCTURE?

- Data structure is a systematic way to organize data in computer memory so that it can be used efficiently.
- Examples - Array, Lists, Stack, Queue, Tree etc.

- Every data structure has two main forms –
  - Interface - It provides the basic operations a data structure can support.

    For example , An array , supports are iteration and accessing a particular data element stored at particular index value .

    String array = {"Suresh", "Dinesh" , "Monali" }

    0^{th} element is Suresh, 1^{st} is Dinesh
  - Implementation – Implementation provides how the data structure is internally stored in computer memory.

    For example array is linear data structure. Values are stored one after another. Again all memory locations are consecutive.

What is Data structures

Examples:

Data Structures are int, float, char, arrays, Linked List, Stack, Queue,Tree and Graph etc.

Need of Data structures.

WHY DATA STRUCTURE NEEDED ?

- Every computer program acts upon data. What if this data is not organized or structured properly ?
- Where do you store data in a computer program ? A variable.

```
int a=123;
```

- What if I want to store list of numbers, employees, movies, online tickets ? And I want to do processing on it say process employee payroll / book tickets.
- We can not store everything in a simple variable, we need more sophisticated structure to hold this data
- That is called DATA STRUCTURES !!

Types of data Structures

There are two types of Data Structures:

- Primitive data structures
- Non-primitive data structures

Primitive data structures: Primitive Data Structures are the basic data structures that directly operate upon the machine instructions.They have different representations on different computers.eg Integers, Floating point numbers, Character constants, String constants and Pointers.

Non-primitive data structures: Non-primitive data structures are more complicated data structures and are derived from primitive data structures.

They emphasize on grouping same or different data items with relationship between each data item.eg Arrays, Lists and Files.

Types of data Structures

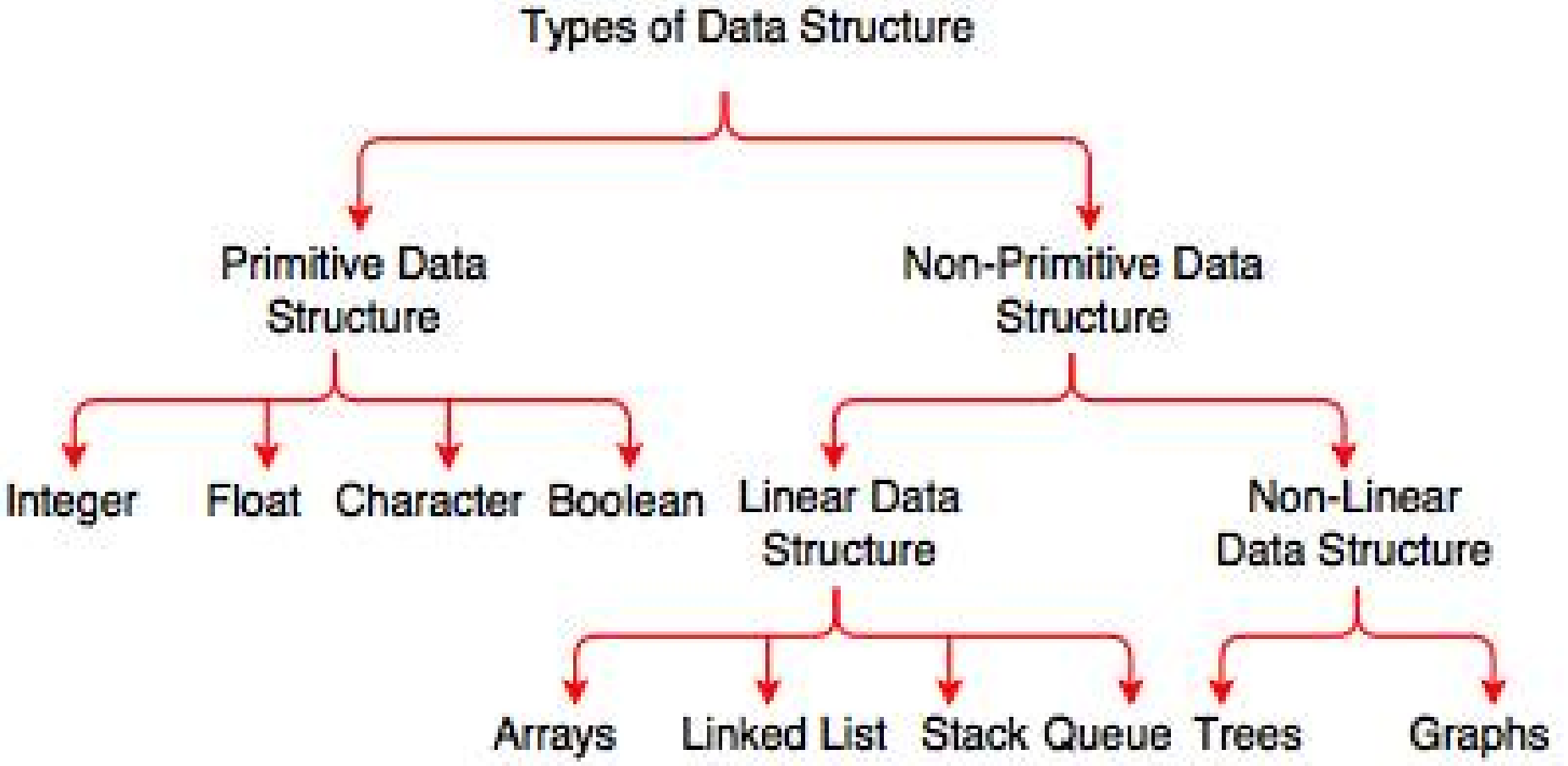


Fig. Types of Data Structure

Different operations on data structures

Basic Operations

The data in the data structures are processed by certain operations. The particular data structure chosen largely depends on the frequency of the operation that needs to be performed on the data structure.

- Traversing
- Searching
- Insertion
- Deletion
- Sorting
- Merging

Different operations on data structures

Data Structure Operations

- The data appearing in data structures are processed by means of operations. The following are operations are major operations:

  a) **Traversing:** Accessing each record exactly once so that certain items in the record may be processed.

  b) **Searching:** Finding the location of the record with a given key value, or finding the locations of all records which satisfy one or more conditions.

  c) **Inserting:** Adding a new record to the structure.

  d) **Deleting:** Removing a record from the structure.

Following two are special operations:

a) **Sorting:** Arranging the records in some logical order.

b) **Merging:** Combining the records in two different sorted files into a single sorted file.

What is an Algorithm

An algorithm is a finite set of instructions or logic, written in order, to accomplish a certain predefined task. Algorithm is not the complete code or program, it is just the core logic(solution) of a problem, which can be expressed either as an informal high level description as pseudocode or using a flowchart.

Every Algorithm must satisfy the following properties:

Input- There should be 0 or more inputs supplied externally to th algorithm.

Output- There should be atleast 1 output obtained.

Definiteness- Every step of the algorithm should be clear and well defined.

Finiteness- The algorithm should have finite number of steps.

Correctness- Every step of the algorithm must generate a correct output.

What is an Algorithm..

An algorithm is said to be efficient and fast, if it takes less time to execute and consumes less memory space. The performance of an algorithm is measured on the basis of following properties :

- Space Complexity
- Time Complexity

Space Complexity:

Its the amount of memory space required by the algorithm, during the course of its execution. Space complexity must be taken seriously for multi-user systems and in situations where limited memory is available.

What is an Algorithm..

An algorithm generally requires space for following components :

Instruction Space: Its the space required to store the executable version of the program. This space is fixed, but varies depending upon the number of lines of code in the program.

Data Space: Its the space required to store all the constants and variables(including temporary variables) value.

Environment Space: Its the space required to store the environment information needed to resume the suspended function.

But while calculating the Space Complexity of any algorithm, we usually consider only Data Space and we neglect the Instruction Space and Environmental Space.

What is an Algorithm..

Calculating the Space Complexity

For calculating the space complexity, we need to know the value of memory used by different type of datatype variables, which generally varies for different operating systems, but the method for calculating the space complexity remains the same.

Type Size

Type	Size
bool, char, unsigned char, signed char, __int8	1 byte
int16, short, unsigned short, wchar_t, __wchar_t	2 bytes
float, __int32, int, unsigned int, long, unsigned long	4 bytes
double, __int64, long double, long long	8 bytes

What is an Algorithm..

Now let's learn how to compute space complexity by taking a few examples:

```
{
   int z = a + b + c;
   return(z);
}
```

In the above expression, variables a, b, c and z are all integer types, hence they will take up 4 bytes each, so total memory requirement will be (4(4) + 4) = 20 bytes, this additional 4 bytes is for **return value**. And because this space requirement is fixed for the above example, hence it is called **Constant Space Complexity**.

What is an Algorithm..

Let's have another example:

```
// n is the length of array a[]
int sum(int a[], int n)
{
int x = 0;  // 4 bytes for x
for(int i = 0; i < n; i++) // 4 bytes for i
{
    x  = x + a[i];
}
return(x);
}
```

- In the above code, 4*n bytes of space is required for the array a[] elements.
- 4 bytes each for x, n, i and the return value.

What is an Algorithm..

Hence the total memory requirement will be (4n + 12), which is increasing linearly with the increase in the input value n, hence it is called as Linear Space Complexity.

Similarly, we can have quadratic and other complex space complexity as well, as the complexity of an algorithm increases.

But we should always focus on writing algorithm code in such a way that we keep the space complexity minimum.

What is an Algorithm..

Time Complexity:

Time Complexity is a way to represent the amount of time required by the program to run till its completion. It's generally a good practice to try to keep the time required minimum, so that our algorithm completes it's execution in the minimum time possible.

For any defined problem, there can be N number of solution. This is true in general. If I have a problem and I discuss about the problem with all of my friends, they will all suggest me different solutions. And I am the one who has to decide which solution is the best based on the circumstances.

What is an Algorithm..

Similarly for any problem which must be solved using a program, there can be infinite number of solutions. Let's take a simple example to understand this. Below we have two different algorithms to find square of a number(for some time, forget that square of any number n is n*n):

One solution to this problem can be, running a loop for n times, starting with the number n and adding n to it, every time.

What is an Algorithm..

```
/* we have to calculate the square of n
*/
for i=1 to n
    do n = n + n
// when the loop ends n will hold its square
return n
```

Or, we can simply use a mathematical operator * to find the square.

```
/*
    we have to calculate the square of n
*/
return n*n
```

What is an Algorithm..

In the above two simple algorithms, you saw how a single problem can have many solutions. While the first solution required a loop which will execute for n number of times, the second solution used a mathematical operator * to return the result in one line. So which one is the better approach, of course the second one.

The time complexity of algorithms is most commonly expressed using the big O notation. It's an asymptotic notation to represent the time complexity.

Now the most common metric for calculating time complexity is Big O notation. This removes all constant factors so that the running time can be estimated in relation to N, as N approaches infinity. In general we can think of it like this :

```
Statement;
```

What is an Algorithm..

Above we have a single statement. Its Time Complexity will be Constant. The running time of the statement will not change in relation to N.

```
for(i=0; i < N; i++)
{
   statement;
}
```

The time complexity for the above algorithm will be **Linear**. The running time of the loop is directly proportional to N. When N doubles, so does the running time.

What is an Algorithm..

```
for(i=0; i < N; i++)
{
    for(j=0; j < N;j++)
    {
    statement;
    }
}
```

This time, the time complexity for the above code will be **Quadratic**. The running time of the two loops is proportional to the square of N. When N doubles, the running time increases by N * N.

Thank You