CS-218
DATA STRUCTURE

Dr. Hashim Yasin

National University of Computer and Emerging Sciences,

Faisalabad, Pakistan.

Course Details

- Course # = CS-218
- Credit Hours: 3 + 1
- Learn commonly used data structures.
- Focus on the concepts about costs and benefits for different types of data structures.
- Understand how to measure the cost of a data structure or program.
  - These techniques also allow to judge the merits of new data structures that you or others might invent.

Tentative Marks Distribution

Item Name	Marks (%)
Quizzes	10-15
Assignments / Project	10-20
Mid Exam1	15
Mid Exam 2	15
Final Exam	40-50

Recomended Books

Data Structures and Algorithm Analysis in C++

Mark Allen Weiss

C++ Plus Data Structure

Nell Dale

Data Structures, Revised 1st Edition

Seymour Lipschutz

Recomended Books

Data Structures Using C and C++

Y. Langsam,
M. J. Augenstein,
A. M. Tenenbaum

Introduction to Algorithms

Charles E. Leiserson,
Clifford Stein,
Ronald Rivest, and
Thomas H. Cormen

Contents

- Introduction
- Complexity Analysis
- Abstract Data Types and Arrays
- Linked List and its implementation
- Linked list (Doubly, circular)
- Elementary Data Structure: Stack
- Applications of stack (conversion (infix to ----), evaluation of postfix)

Contents

- Elementary Data Structure: Queue
- Queue applications and implementation
- Priority Queues
- Trees: Binary Search Tree (representation, insertion, searching, display, deletion)
- Trees: AVL Tree (Insertion, Implementation)
- Heaps: Heap Data Structure, Max and Min Heaps

Contents

- Graphs Data Structure: Adjacency Matrix and Adjacency list
- Graphs Searching: DFS and BFS
- Graphs: Minimum Spanning Trees (Prim's and Kruskal's Algorithm)
- Shortest Path Algorithms (Bellman Ford, Dijkstra)
- Hashing
- Sorting Techniques: Bubble Sort, Selection Sort, Insertion Sort, Merge Sort, Quick Sort

INTRODUCTION

Introduction

- Data structure is defined as a particular way of storing and organizing data in computer so that it can be used efficiently.
- Data structure refers to
  - the organization of data in computer memory or
  - the way in which data is efficiently stored, processed and retrieved.
- Data structure is a **structural representation** of logical relationships between elements of data

Introduction

- Data structure allows to
  - understand the relationship of one element with another,
  - organize them within the memory.

- Data structure is said to be the mathematical and logical model of a particular organization.

Introduction

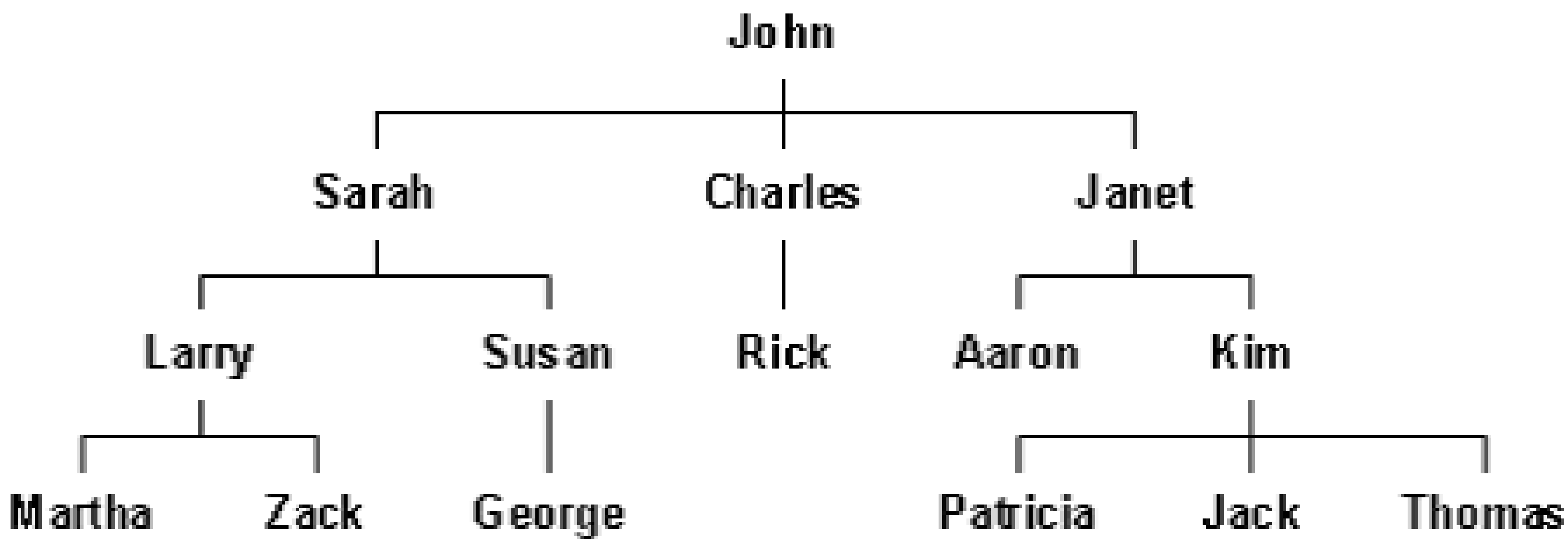
- All programs manipulate data
  - programs *process, store, display, gather*
  - data can be *information, numbers, images, sound*
- Each program must decide how to store data
- The choice influences program at every level
  - execution speed
  - memory requirements
  - maintenance (debugging, extending, etc.)

Example

Assume that you are given a task by company XYZ to organize all of their records into a computer database.

Name	Position
Aaron	Manager
Charles	VP
George	Employee
Jack	Employee
Janet	VP
John	President
Kim	Manager
Larry	Manager
Martha	Employee
Patricia	Employee
Rick	Secretary
Sarah	VP
Susan	Manager
Thomas	Employee
Zack	Employee

Example


John
Sarah
Charles
Janet
Larry
Susan
Rick
Aaron
Kim
Martha
Zack
George
Patricia
Jack
Thomas

Example

- In one of the data structures, data is organized into a list.
  - Useful for keeping the names of the employees in alphabetical order so that we can locate the employee's record very quickly.
- However, this structure is not very useful for showing the relationships between employees.
- A tree structure is much better suited for this purpose.

Applications

1. Compiler design

2. Data management system

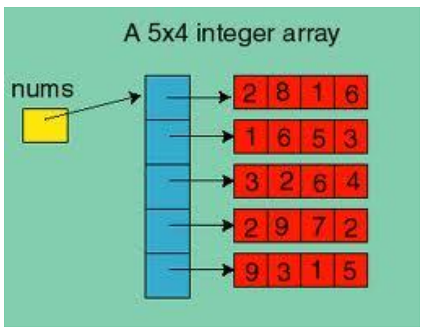
3. Simulation

4. Operating system

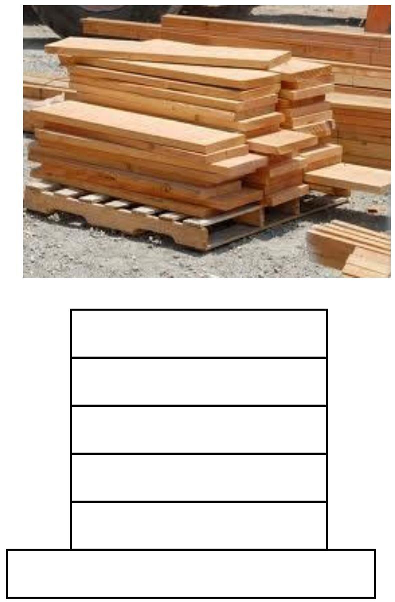
Types of Data Structures

1. Array
2. Ordered array
3. Stack
4. Queue
5. Linked list
6. Trees
7. Hash Table
8. Heap
9. Graph

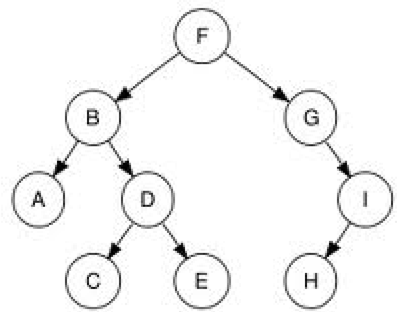
Types of Data Structures



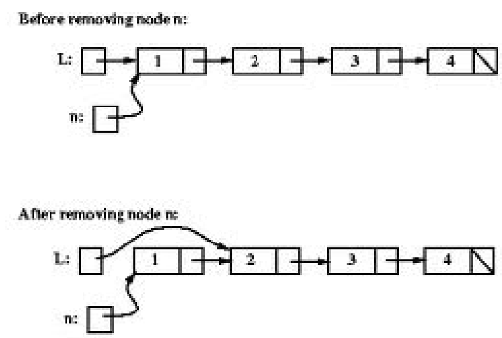
Arrays



Stack

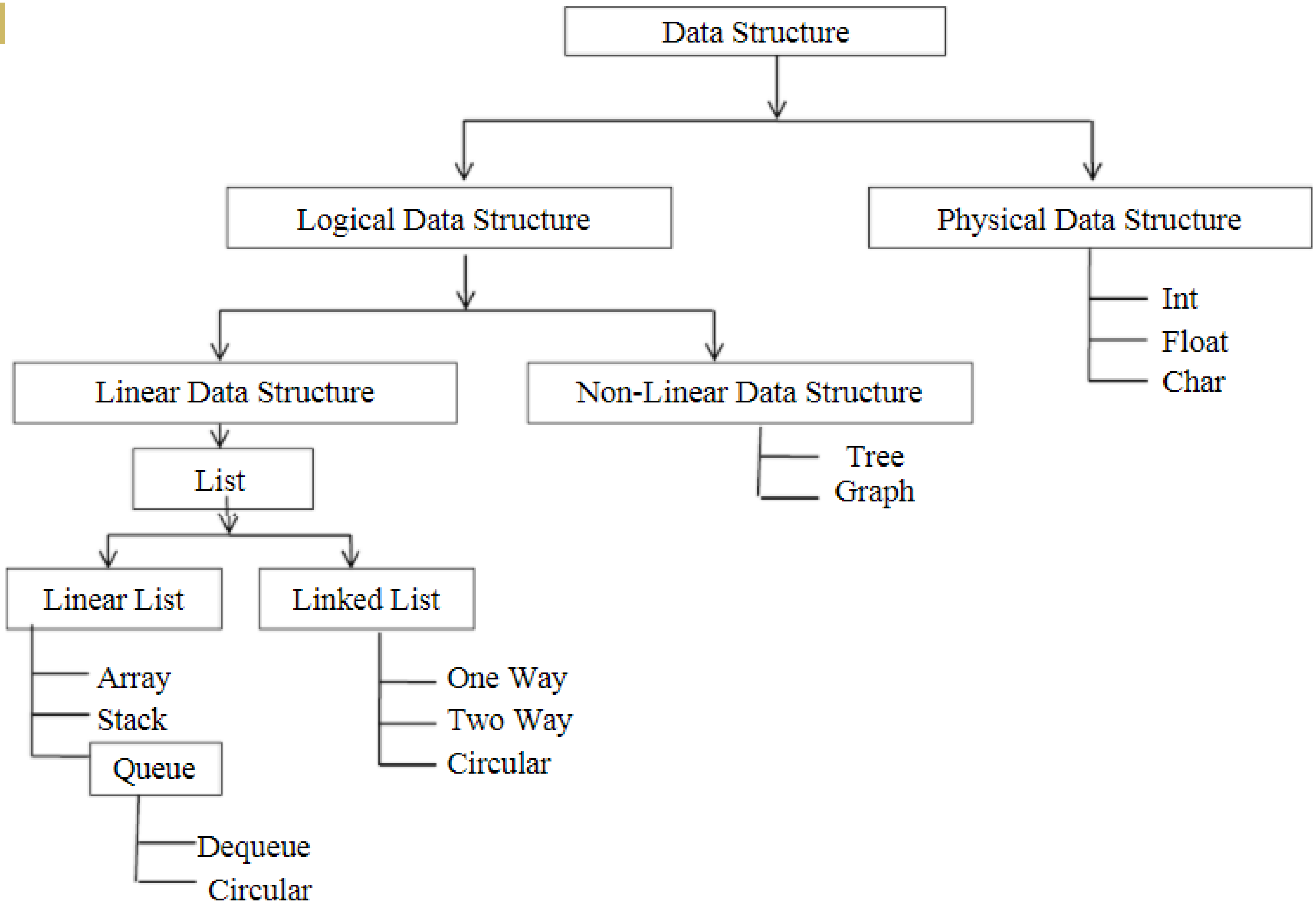


Tree



Linked List

Types of Data Structures



Types of Data Structures

Data structures are divided into two types:

- Primitive data structures.
- Non-primitive data structures.

- **Primitive Data Structures:** are the basic data structures that directly operate upon the machine instructions. They may have different representations on different computers.
- **Examples:** Integers, floating point numbers, character constants, string constants, and pointers etc.

Types of 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.
- **Examples:** Arrays, lists and files etc.

Arrangement of Data Structure

1. Nature of size

- Static Data Structure (S.D.S)
- Dynamic Data Structure (D.D.S)

- **Static Data Structure:** It is said to be static when we can store data to a fixed number, e.g., arrays.
- **Dynamic Data Structure:** It's a type of dynamic data that can allow the programmer change its size during execution to add or delete data., e.g., linked list, trees, graph.

Arrangement of Data Structure

2. Occurrence

- Linear Data Structure L.D.S
- Non-Linear Data Structure N.L.D.S

Liner Data Structure:

- Data is stored in a consecutive data structure (sequential form).
- Every element in the structure has a unique predecessor or successor.
- Examples are stack, queues, etc.

Arrangement of Data Structure

2. Occurrence

- Linear Data Structure L.D.S.
- Non-Linear Data Structure N.L.D.S.

Non-Liner Data Structure:

- Data is stored in a non-consecutive memory location; i.e., are NOT in sequential form.
- N.L.D.S. is used to represent data containing a hierarchical relationship between elements.
- Examples are B-Tree, Graph, etc.

ARRAYS

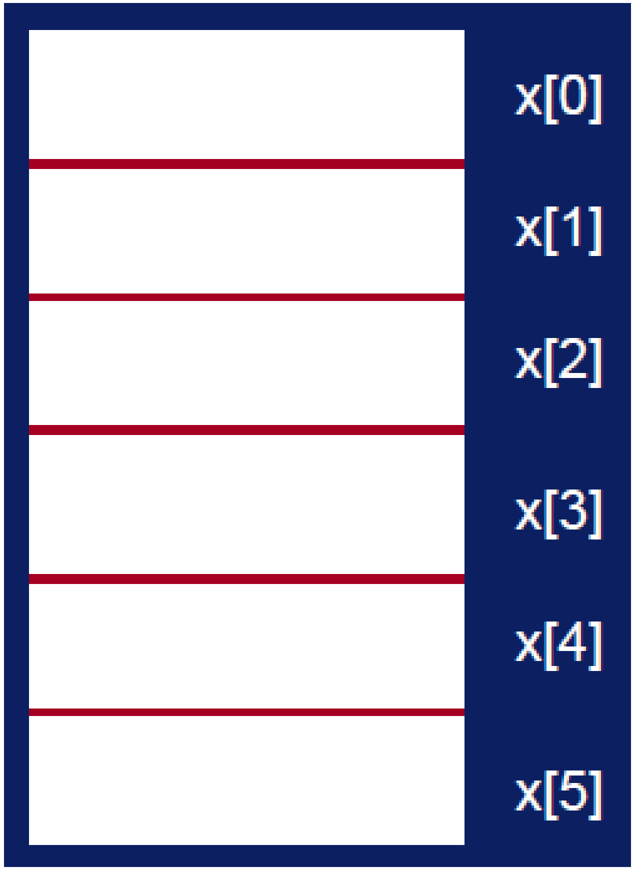
Arrays

- Array declaration: int x[6];
- An array is collection of cells of the same type.
- The collection has the name 'x'.
- The cells are numbered with consecutive integers.
- An array of 'n' elements will have an index of zero for the first element up to index (n-1) for the last element.
- To access a cell, use the array name and an index:

x[0], x[1], x[2], x[3], x[4], x[5]

Array Layout

- Array cells are contiguous in computer memory
- The memory can be thought of as an array



Array

- ‘x’ is NOT an l-value (locator value), “l-value” refers to memory location which identifies an object”.

```
int x[6];
int n;

x[0] = 5;
x[1] = 2;

x = 3;          // not allowed
x = a + b;      // not allowed
x = &n;   // not allowed
```

Array

- In previous example, we can only store integers in this array.

- We CANNOT put *int* in first location, *float* in second location and *double* in third location.

Dynamic Array

- You would like to use an array data structure, but you do not know the size of the array at compile time.
- You find out when the program executes that you need an integer array of size n=20.
- Allocate an array using the new operator:

```
int* y = new int[100]; // or int* y = new int[n]
y[0] = 10;
y[1] = 15; // use is the same
```

Dynamic Array

- 'y' is a l-value; it is a pointer that holds the address of 20 consecutive cells in memory.
- It can be assigned a value. The new operator returns as address that is stored in y. We can write:

```
y = &x[0];
y = x; // x can appear on the right
       // y gets the address of the
       // first cell of the x array
```

Dynamic Array

- We must free the memory we got using the new operator once we are done with the *y* array.

```
delete[ ] y;
```

- We would not do this to the *x* array because we did not use new to create it.

Example


```
#include <iostream>
using namespace std;
int main(){
        int x[6];
        x[0] = 5;
        x[1] = 2;
        x[2] = 1000;
        int* y = new int[20];
        y[0] = 10;
        y[1] = 15;
        cout <<"y[0]="<<y[0];
        cout <<"\ny[1]="<<y[1];
        y = &x[1];
        cout << "\nLook at the difference ";
        cout <<"\ny[0]="<<y[0];
        cout <<"\ny[1]="<<y[1];
}
```

What would be the output?

Example

y[0]=10

y[1]=15

Look at the difference

y[0]=2

y[1]=1000