SCJ2013 Data Structure & Algorithms

Linked List

Nor Bahiah Hj Ahmad & Dayang
Norhayati A. Jawawi

Course Objectives

At the end of the class, students are expected to be able to do the following:

- Describe linear list concepts using array and linked list.
- Lists variations of linked list and basic operations of linked lists.
- Explain in detail the implementation and operations of link lists using pointers.
- Write program that can implement linked list concept.

INTRODUCTION

Lists Definition

- Lists is a group of objects which is organized in sequence.
- List categories: linear list and nonlinear list.
- Linear list – a list in which the data is organized in sequence, example: array, linked list, stack and queue
- Non-Linear list – a list in which the data is stored not in sequence, example: tree and graph

Introduction to Linear List

- **Array and linked** lists are linear lists that doesn't have any restrictions while implementing operations such as, insertion, deletion and accessing data in the lists.
- The operations can be done in any parts of the lists, either in the front lists, in the middle or at the back of the lists.

Introduction to Linear List

Stack and queue is a linear lists that has restrictions while implementing its operations.

- **Stack** – to insert, delete and access data can only be done at the top of the lists.
- **Queue** - Insert data in a queue can be done at the back of the lists while to delete data from a queue can only be done at the front list.

Linear List Example : Array

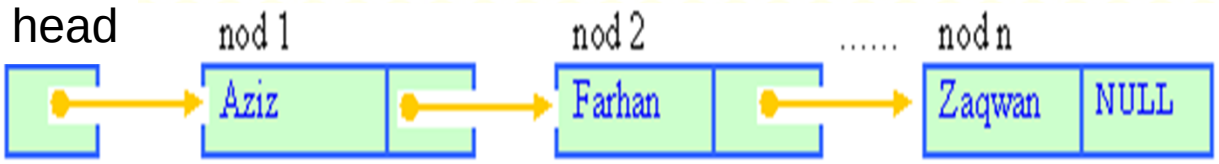
Pelajar

Indeks	Nama	Kursus	Tahun
[0]	Aziz Nabil	Sains Komputer	2
[1]	Boh Guan	Sains Komputer	1
[2]	Durrani Nukman	Kej. Elektrik	3
[3]	Mohd Saufi	Sains Pendidikan	2
[4]	Nafisah Nordin	Sains Komputer	1
[5]	Safinatun Najah	Pengurusan Komputer	3
[6]			
[7]			

- An array named Pelajar which contains attributes nama pelajar, kursus and tahun Pelajar.
- The array is sorted and can only be accessed based on the index or subscript of the array.
- Example: to access information for a student named Mohd Saufi, we can use:

```
Pelajar[3].Nama, Pelajar[3].Kursus and Pelajar[3].Tahun.
```

Linear List Example : Linked List



- Linked lists consists of several **nodes** which is sorted in ascending order.
- Each node must contain at least :
  - A piece of **data**
  - Pointer to the **next** node in the list
- Need a pointer variable, named ***head*** to point to the first node in the list.

Linear List Operations

Basic operations for linear lists:

- Insert new data in the lists.
- Delete data from a lists.
- Update data in the list.
- Sort data in the lists and
- Find data in the list.

Array as Linear List

- Sized is fixed during array declaration.
- Data insertion is limited to array size.
- In order to insert data, need to check whether the array is full or not. If the array is full, the insertion cannot be done.

Array as Linear List

- Data in the array can be accessed at random using the index of the array.
- To access information at index 3, for a student named Mohd Saufi taking Sains Pendidikan course and in year 2 are as follows:

```
cout << Pelajar[3].Nama << Pelajar[3].Kursus <<
                           Pelajar[3].Tahun;
```

- By random access, accessing data in an array can be done fast

Pelajar

Indeks	Nama	Kursus	Tahun
[0]	Aziz Nabil	Sains Komputer	2
[1]	Boh Guan	Sains Komputer	1
[2]	Durrani Nukman	Kej. Elektrik	3
[3]	Mohd Saufi	Sains Pendidikan	2
[4]	Nafisah Nordin	Sains Komputer	1
[5]	Safinatun Najah	Pengurusan Komputer	3
[6]			
[7]			

Drawbacks of Array Implementation

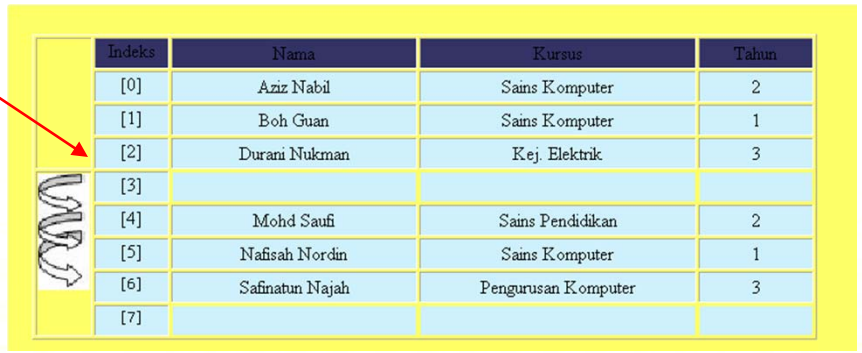
- Requires an estimate of the maximum size of the list
- insert and delete: slow
  - insert at position 0 (making a new element)
    - requires first pushing the entire array down one spot to make room
  - delete at position 0
    - requires shifting all the elements in the list up one
  - On average, half of the lists needs to be moved for either operation

Drawbacks of Array Implementation

- Need space to insert item in the middle of the list.
- Insert Fatimah Adam in between students named Durrani Nukman and Mohd Saufi.
- insert at index 3 : requires first pushing the entire array from index 3 down one spot to make room

Pelajar

Indeks	Nama	Kursus	Tahun
[0]	Aziz Nabil	Sains Komputer	2
[1]	Boh Guan	Sains Komputer	1
[2]	Durrani Nukman	Kej. Elektrik	3
[3]	Mohd Saufi	Sains Pendidikan	2
[4]	Nafisah Nordin	Sains Komputer	1
[5]	Safinatun Najah	Pengurusan Komputer	3
[6]			
[7]			



Drawbacks of Array Implementation

Indeks	Nama	Kursus	Tahun
[0]	Aziz Nabil	Sains Komputer	2
[1]	Boh Guan	Sains Komputer	1
[2]	Durani Nukman	Kej. Elektrik	3
[3]	Fatimah Adam	Kej. Awam	2
[4]	Mohd Saufi	Sains Pendidikan	2
[5]	Nafisah Nordin	Sains Komputer	1
[6]	Safinatun Najah	Pengurusan Kompuer	3
[7]			

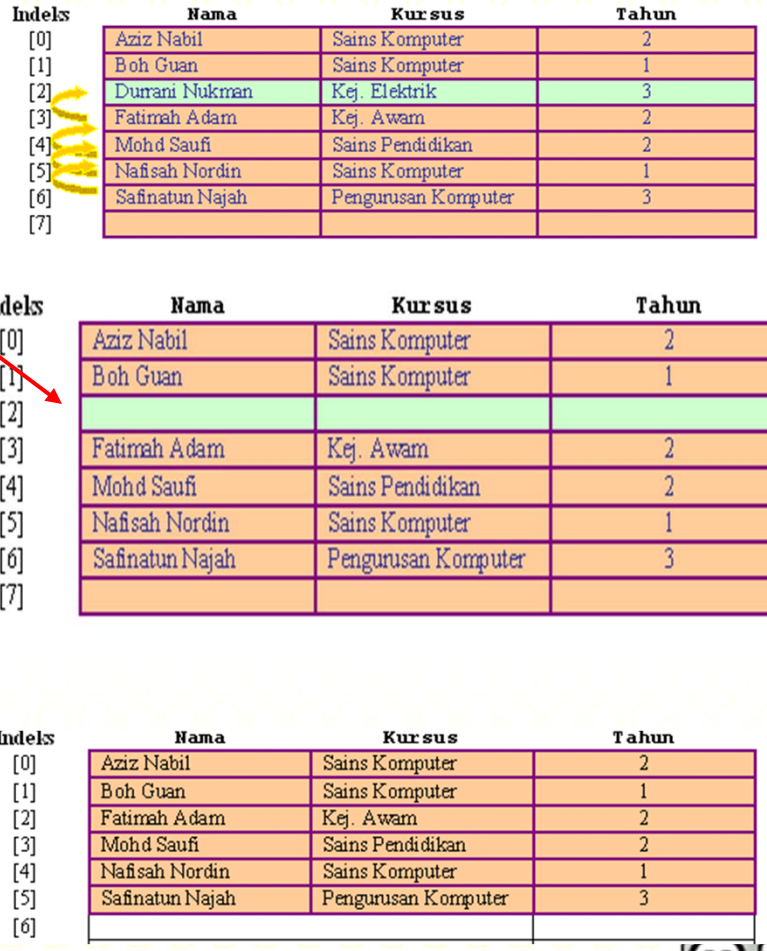
New item is inserted at index 3,
after shifting the data from index 3 onwards. .

Drawbacks of Array Implementation

To delete item in the middle of the array will leave a blank space in the middle.

It requires shifting all the elements in the list up one in order to eliminate the space..

Example: when information about Durrani Nukman is deleted, all elements below it, is shifted up.



Indeks	Nama	Kursus	Tahun
[0]	Aziz Nabil	Sains Komputer	2
[1]	Boh Guan	Sains Komputer	1
[2]	Durrani Nukman	Kej. Elektrik	3
[3]	Fatimah Adam	Kej. Awam	2
[4]	Mohd Saufi	Sains Pendidikan	2
[5]	Nafisah Nordin	Sains Komputer	1
[6]	Safinatun Najah	Pengurusan Komputer	3
[7]			

Indeks	Nama	Kursus	Tahun
[0]	Aziz Nabil	Sains Komputer	2
[1]	Boh Guan	Sains Komputer	1
[2]			
[3]	Fatimah Adam	Kej. Awam	2
[4]	Mohd Saufi	Sains Pendidikan	2
[5]	Nafisah Nordin	Sains Komputer	1
[6]	Safinatun Najah	Pengurusan Komputer	3
[7]			

Indeks	Nama	Kursus	Tahun
[0]	Aziz Nabil	Sains Komputer	2
[1]	Boh Guan	Sains Komputer	1
[2]	Fatimah Adam	Kej. Awam	2
[3]	Mohd Saufi	Sains Pendidikan	2
[4]	Nafisah Nordin	Sains Komputer	1
[5]	Safinatun Najah	Pengurusan Komputer	3
[6]			

Pointer Implementation (Linked List)

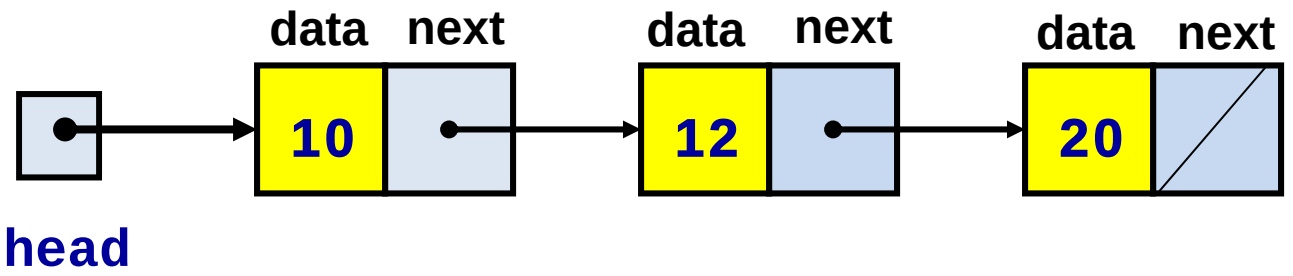
- Ensure that the list is not stored contiguously
  - use a linked list
  - a series of structures that are not necessarily adjacent in memory

- Each node contains the element and a pointer to a structure containing its successor
  - the last cell's next link points to NULL
- Compared to the array implementation,
  - the pointer implementation uses only as much space as is needed for the elements currently on the list
  - but requires space for the pointers in each cell

Linked List Variations

- Singly linked list
- Doubly linked list
- Circular linked list
- Circular doubly linked list
- Sorted linked list
- Unsorted linked list

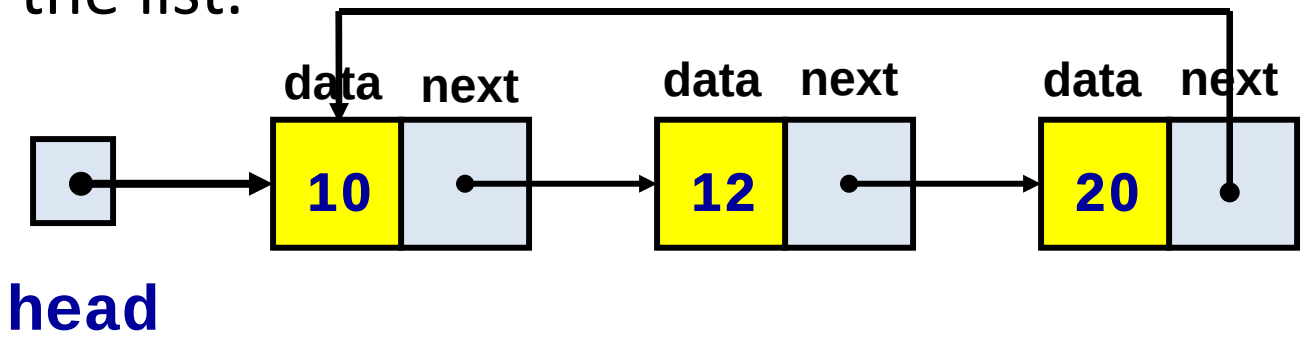
Singly Linked Lists



- A linked list is a series of connected nodes
- Each node contains at least
  - A piece of data of any type
  - Pointer to the next node in the list
- Head is a pointer that points to the first node
- The last node points to **NULL**

Circular Linked Lists

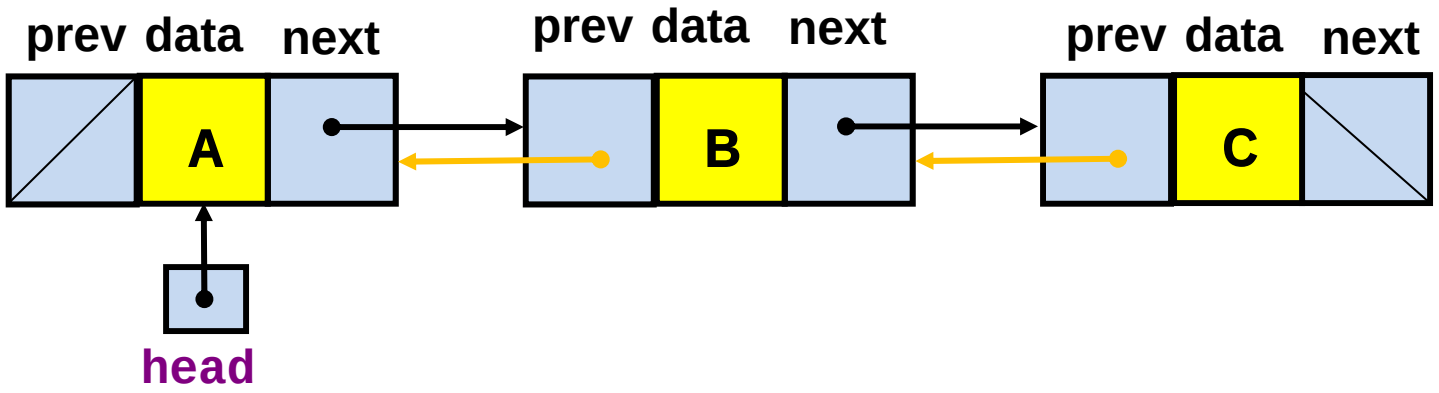
Circular linked list contains a series of connected nodes with the last node points to the first node of the list.



Doubly Linked Lists

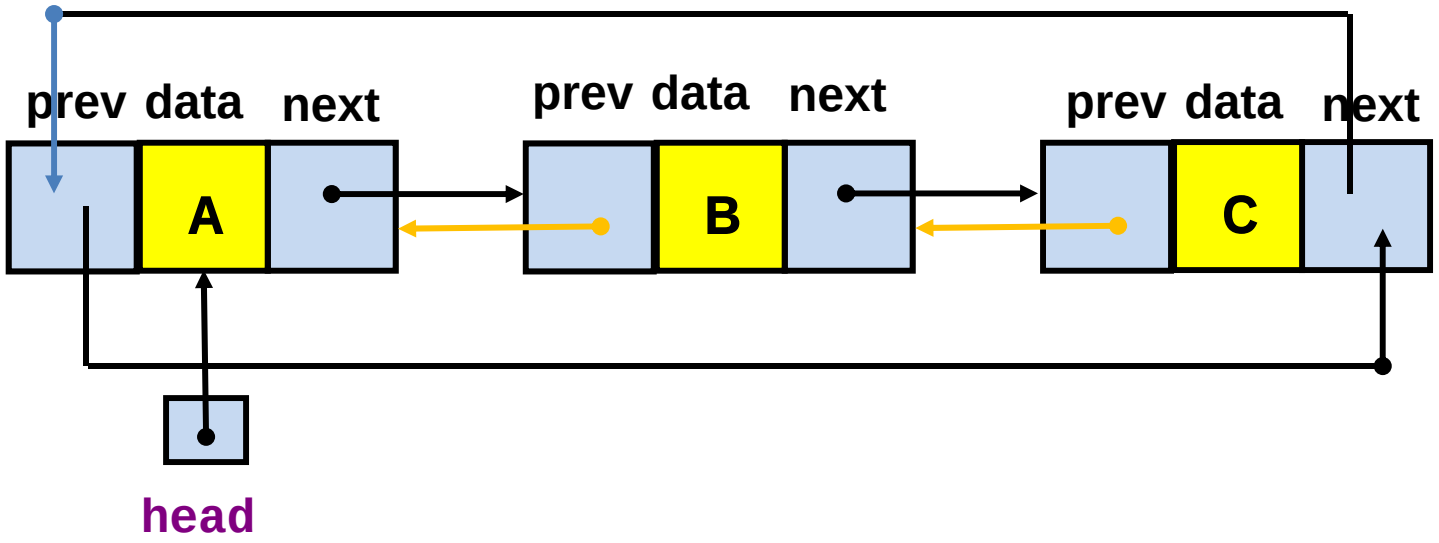
Each node in doubly linked list has 2 pointers that point not only to the successor but the predecessor

- There are two NULL : at the first and last nodes in the list
- Advantage: given a node, it is easy to visit its predecessor and convenient to traverse lists backwards.



Circular Doubly Linked Lists

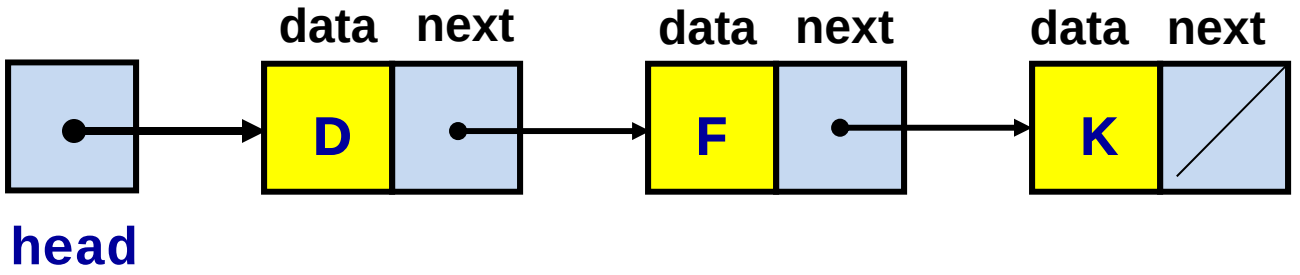
- Circular doubly linked list doesn't has NULL value at the first and last nodes in the list
- Advantage : Convenient to traverse lists backwards and forwards



Variations of Linked Lists

Sorted Linked list :

The nodes in the lists is sorted in certain order.



UnSorted Linked list :

The nodes in the lists is not sorted in any order.