Linked List

SCSJ2013 Data Structures & Algorithms

Nor Bahiah Hj Ahmad & Dayang Norhayati A. Jawawi

Faculty of Computing

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

What is Lists?

- 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.

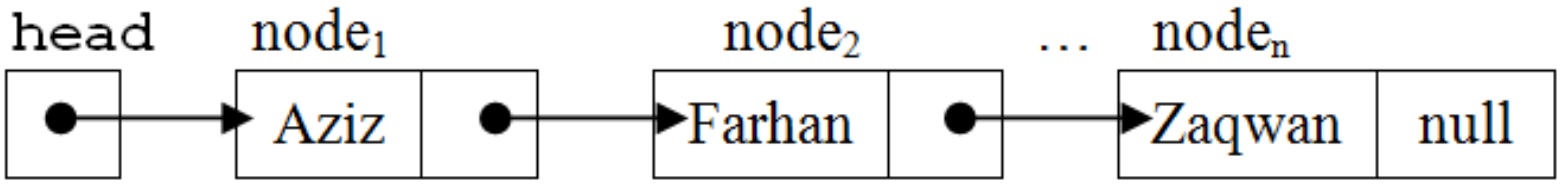
- The operations can be done in any parts of the lists, either in the front lists, in the middle or at the back of the lists.

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 : Linked List



- Linked lists consists of several **nodes** which is sorted into ascending order.
- Each node must contain at least 2 parts:
  - 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.

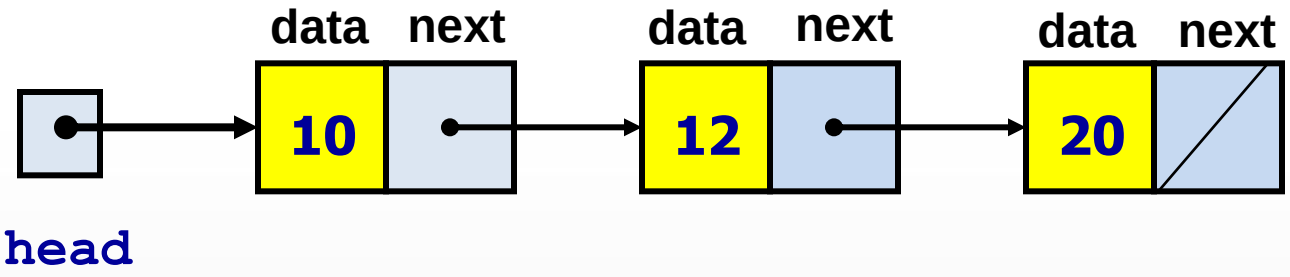
Pointer Implementation (Linked List)

- **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
  - û 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

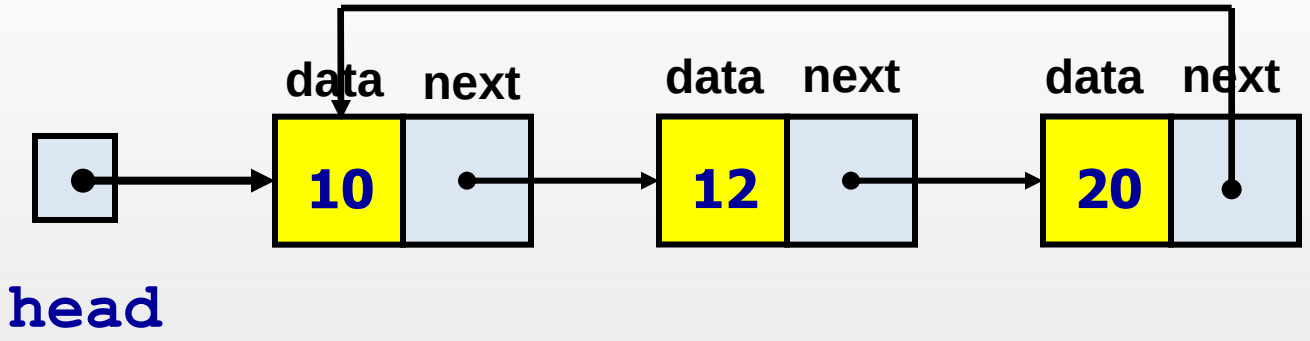
Singly Linked Lists



- 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

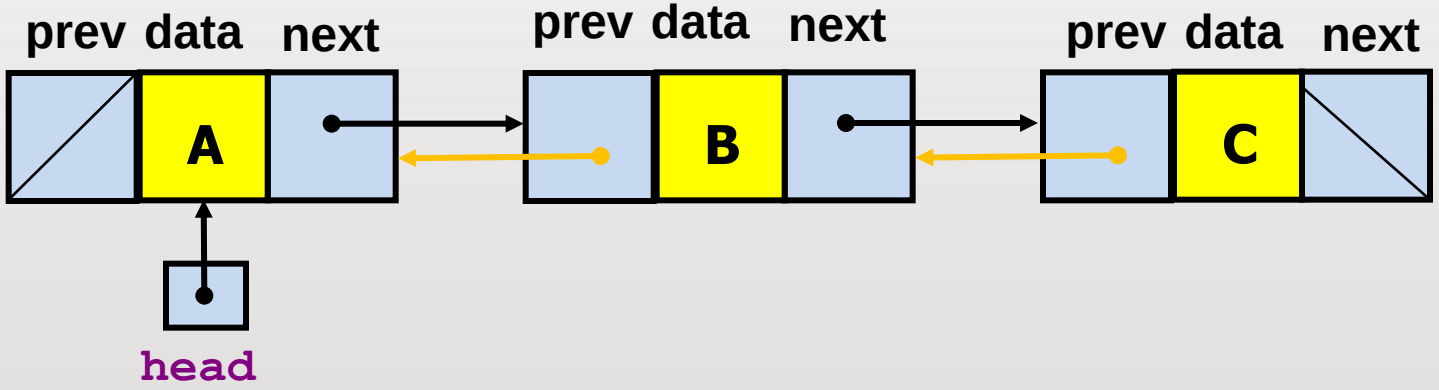
Circular linked list contains a series of connected nodes with the last node points to the first node of the list.



Doubly Linked Lists

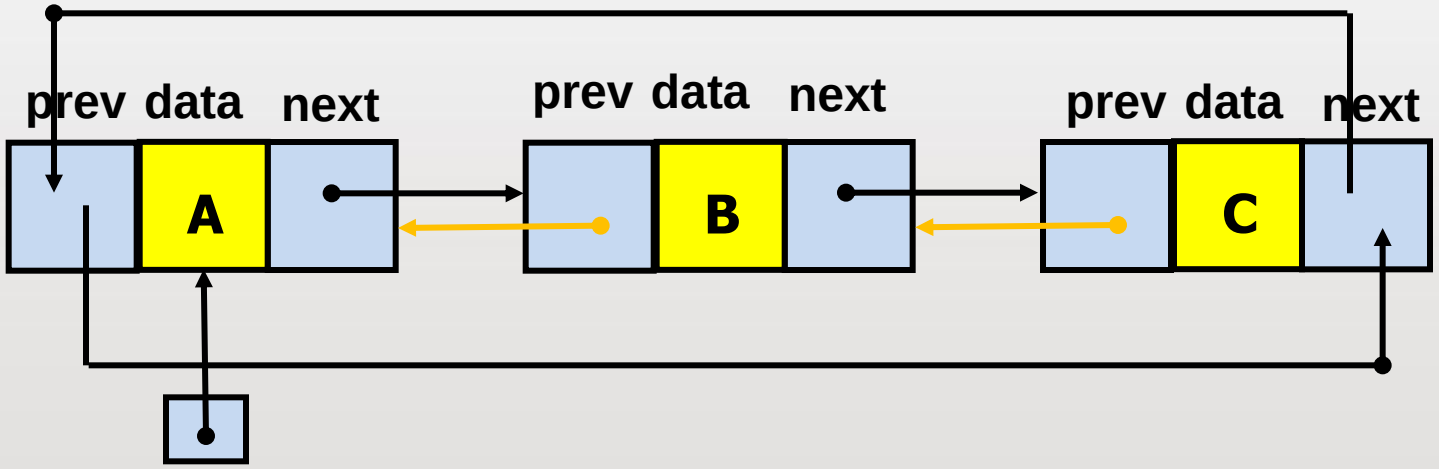
Each node in doubly linked list has 2 pointers that point not only to the successor but the predecessor

- 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

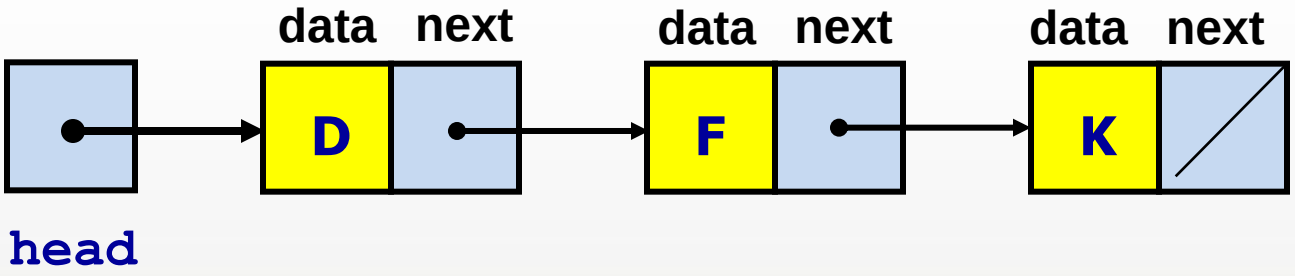
- Circular doubly linked list doesn't has `NULL` value at the first and last nodes in the list
- Advantage : Convenient to traverse lists backwards and forwards



Variations of Linked Lists

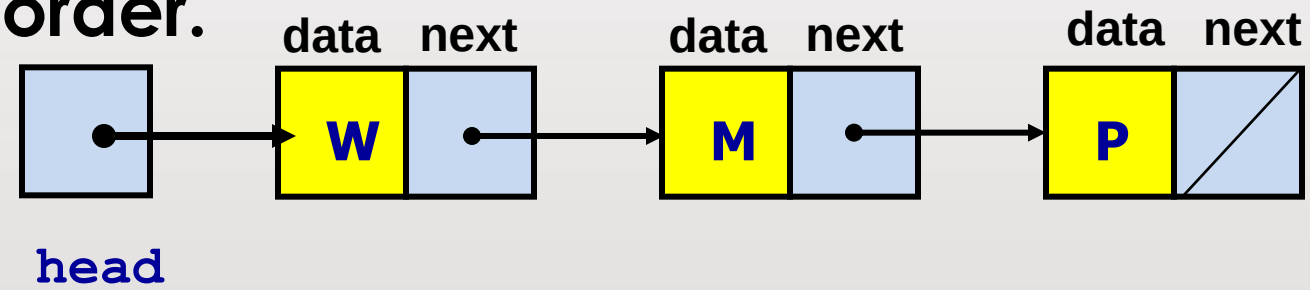
Sorted Linked list :

The nodes in the lists is sorted in certain order.



UnSorted Linked list :

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




Thank
You