Linked List Implementation

SCSJ2013 Data Structures & Algorithms

Nor Bahiah Hj Ahmad & Dayang Norhayati A. Jawawi

Faculty of Computing

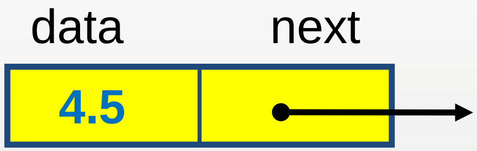
Linked List Implementation

There are 2 classes in linked list implementation:

1. Class `Node`
2. Classes `list`.

Declaration of Node

Declare **Node** class for the nodes which contains **data** and **next**, which is a pointer to the next node in the list.



```
class Node {
public:
   double data;    // data
   Node* next;     // pointer to next node
};
```

Declaring a node for class account

Create a node for class account using struct

```
struct nodeAccount {
   char accountName[20];
   char accountNo[15];
   float balance;
   nodeAccount *next;
};
```

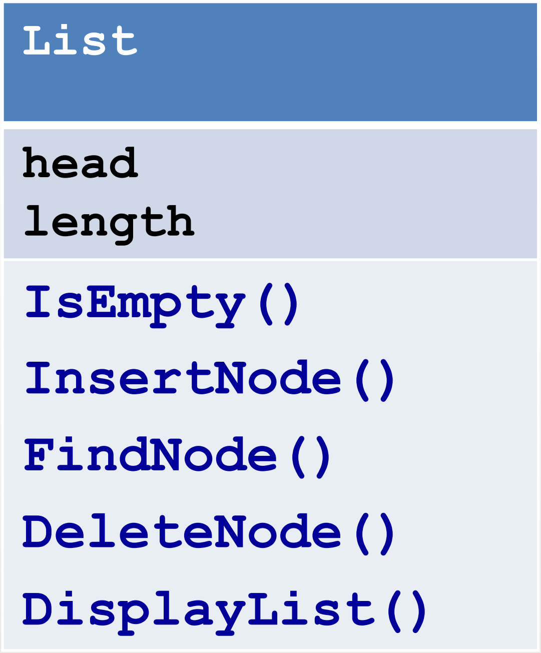
accountName	accountNo	balance	next
Ahmad Ali	1234567	10,000.00	→

Declaration of class List

Class List contains

- head: a pointer to the first node in the list.

  The list is initially empty, head is set to NULL
- length : number of nodes in the list
- Operations on List



List

head
length

IsEmpty()
InsertNode()
FindNode()
DeleteNode()
DisplayList()

Declaration of class List

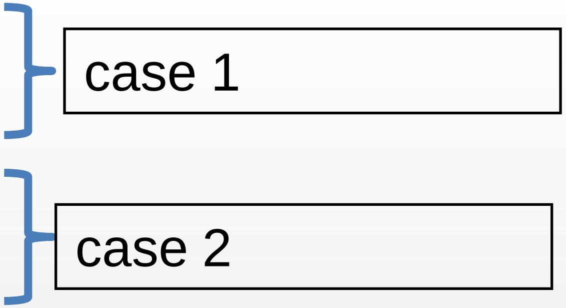
```
class List {
public:
     // constructor
    List(void) { head = NULL; length = 0;}
    // destructor
   ~List(void);

   bool IsEmpty() { return head == NULL; }
   void InsertNode(double x);
   int FindNode(double x);
   void DeleteNode(double x);
   void DisplayList(void);
private:
   Node* head;
   int length;
};
```

Insert a New Node to the List

Possible cases of `InsertNode`

1. **Insert into an empty list**
2. **Insert in front**
3. **Insert at back**
4. **Insert in middle**



case 1

case 2

Insert a New Node to the List

```
void InsertNode(double x)
```

- This function inserts a node with data equal to `x`.
- After insertion, this function generates a sorted list in ascending order.

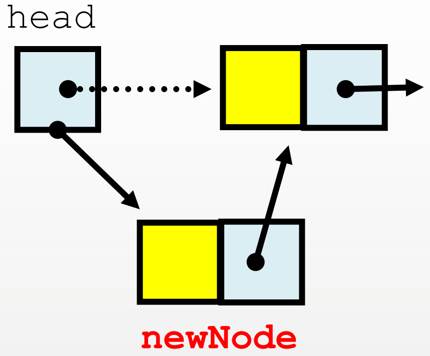
Steps to insert a node in linked list

- Find the location of the value to be inserted so that the value will be in the correct order in the list.
- Allocate memory for the new node
- Insert the new node to the list.

Insert a New Node to the List

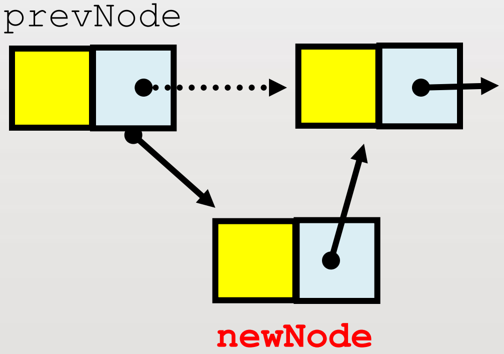
```
void InsertNode(double x)
```

- Insert at front or empty list : point head to the new node



```
newNode->next = head;
head = newNode;
```

- Insert in the middle or back list : point the new node predecessor to the new node



```
newNode->next = prevNode->next;
prevNode->next = newNode;
```

Delete Node

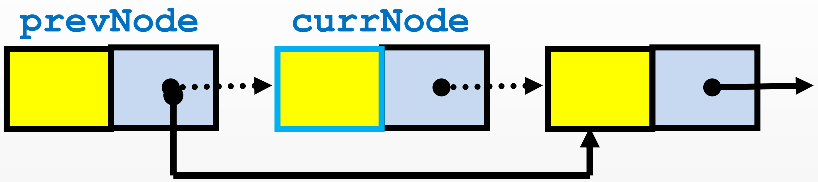
`void DeleteNode(double x)`

- Delete a node with the value equal to `x` from the list.

- Steps
  - Find the node to be deleted .
  - Release the memory occupied by the found node.
  - Set the pointer of the predecessor of the found node to the successor of the found node.
- Like `InsertNode`, there are two special cases
  - Delete first node.
  - Delete the node in middle or at the end of the list.

Delete Node

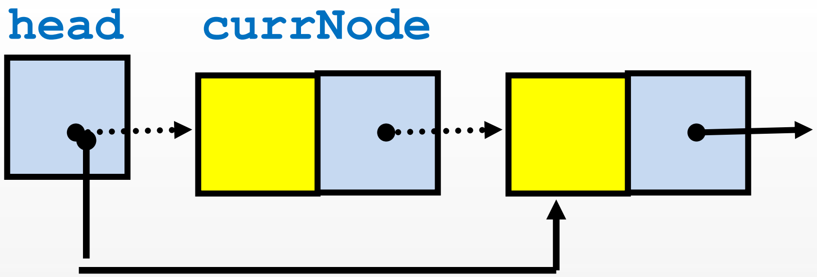
Delete in the middle or at the back of the list



```
prevNode->next = currNode->next;
delete currNode;
currNode = NULL;
```

Delete Node

Delete at the front of the list



```
head   = currNode->next;
delete currNode;
currNode = NULL;
```

Print All Elements in the List

`void DisplayList()`

- Print the data of all the elements and
- Print the number of the nodes in the list

```
void List::DisplayList()
{
   int num = 0;
   Node* currNode = head;
   while (currNode != NULL){
   cout << currNode->data << endl;
   currNode = currNode->next
   }
}
```

Summary

Implementation

- Linked lists implementation need 2 classes to be declared, which are class node and class list.

List Size

- No need to know in advanced how many nodes will be in the list. Linked list can easily grow and shrink in size dynamically.
- However, the size of a C++ array is fixed at compilation time, therefore the number of elements in the list are limited to the size.

Summary

Insertions and deletions

- To insert or delete an element in an array, need to make room for new elements or close the gap caused by deleted elements.
- With a linked list, no need to move other nodes. Only need to reset some pointers. Linked list is easier and faster to delete node in the list.

Accessing element

- In array, elements can be access at random, while in linked list item can only be accessed sequentially.

Thank You

http://comp.utm.my/