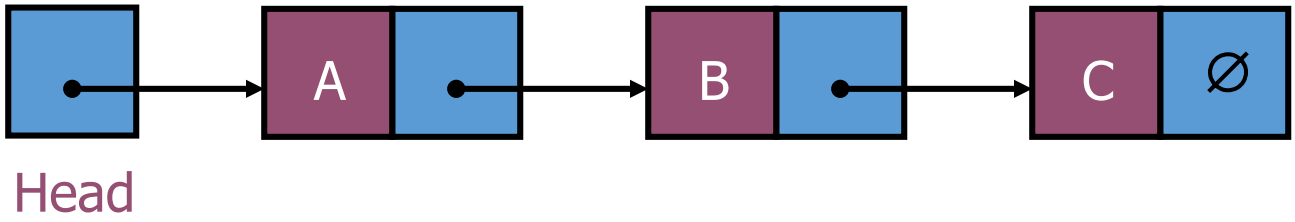
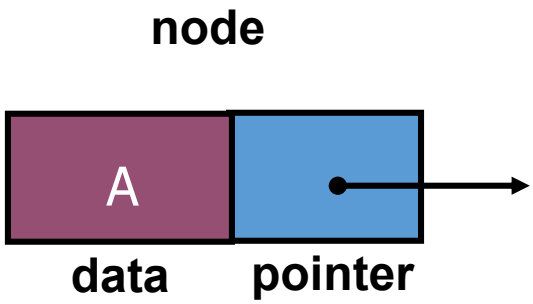
❖ Linked List



A linked list is a series of connected nodes ,each node contains at least a piece of data (any type) and Pointer to the next node in the list

Head: pointer to the first node

The last node points to NULL



A Simple Linked List Class

We use two classes: Node and List

Declare Node class for the nodes

data: double-type data in this example

next: a pointer to the next node in the list

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

Declare List, which contains

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

Since the list is empty initially, head is set to NULL

Operations on List

```
class List {
public:
        List(void) { head = NULL; }     // constructor
        ~List(void);                    // destructor
        bool IsEmpty() { return head == NULL; }
        Node* InsertNode(int index, double x);
        int FindNode(double x);
        int DeleteNode(double x);
        void DisplayList(void);
private:
        Node* head;
};
```

Operations of List

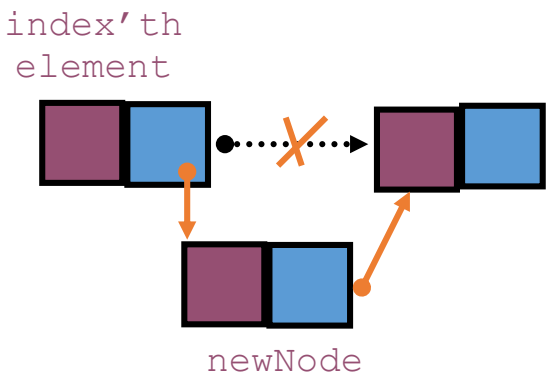
* IsEmpty: determine whether or not the list is empty
* InsertNode: insert a new node at a particular position
* FindNode: find a node with a given value
* DeleteNode: delete a node with a given value
* DisplayList: print all the nodes in the list

Inserting a new node

* Node* InsertNode(int index, double x)
    - Insert a node with data equal to x after the index'th elements.
        - (i.e., when index = 0, insert the node as the first element;
        - when index = 1, insert the node after the first element, and so on)
    - If the insertion is successful, return the inserted node.

      Otherwise, return NULL.

      (If index is < 0 or > length of the list, the insertion will fail.)
* Steps
    1. Locate index'th element
    2. Allocate memory for the new node
    3. Point the new node to its successor
    4. Point the new node's predecessor to the new node



* Possible cases of InsertNode
  - Insert into an empty list
  - Insert in front
  - Insert at back
  - Insert in middle

* But, in fact, only need to handle two cases
  - Insert as the first node (Case 1 and Case 2)
  - Insert in the middle or at the end of the list (Case 3 and Case 4)

```
Node* List::InsertNode(int index, double x) {
        if (index < 0) return NULL;

        int currIndex       =       1;
        Node* currNode      =       head;
        while (currNode && index > currIndex) {
                currNode    =       currNode->next;
                currIndex++;
        }
        if (index > 0 && currNode == NULL) return NULL;

        Node* newNode       =       new   Node;
        newNode->data       =       x;
        if (index == 0) {
                newNode->next       =       head;
                head                =       newNode;
        }
        else {
                newNode->next       =       currNode->next;
                currNode->next      =       newNode;
        }
        return newNode;
}
```