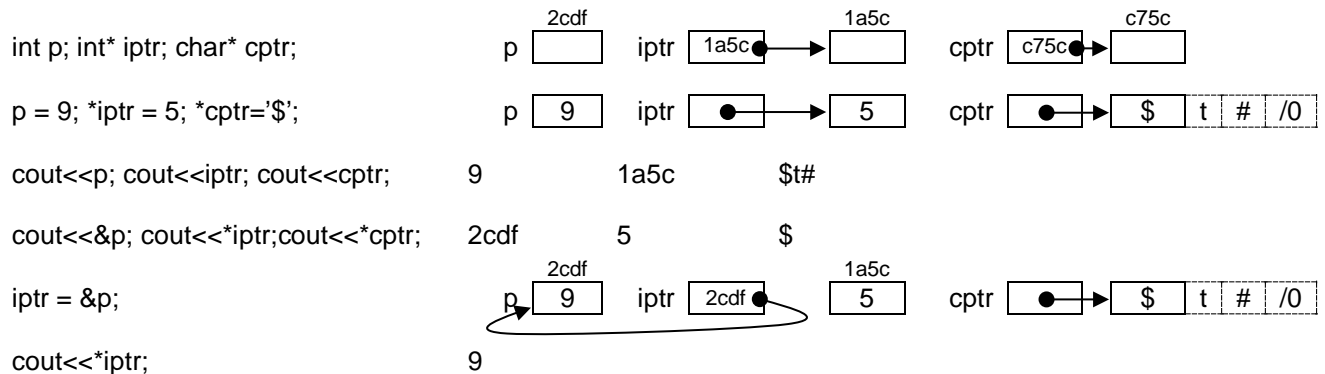


3. Linked List Data Structure

What is Pointer?

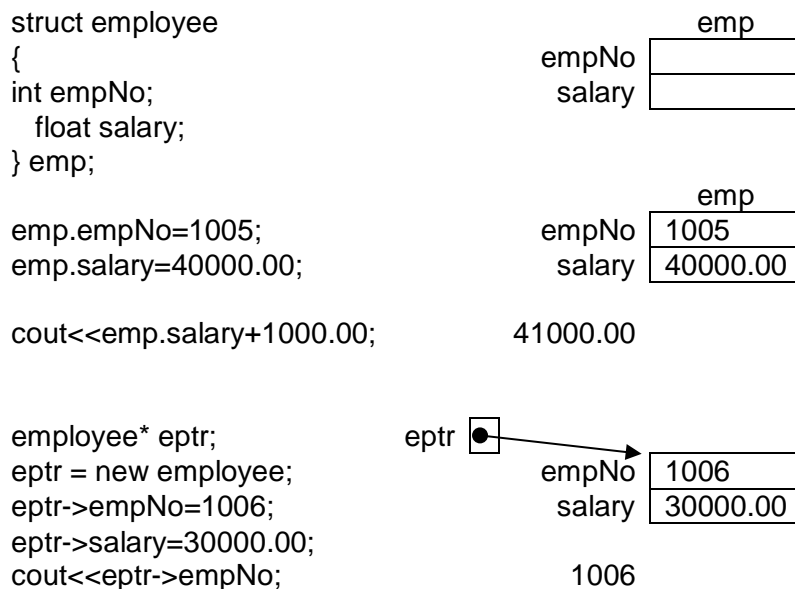
Pointer contains memory address of a particular type of data.

In C++, we use * to declare a pointer.



What is Structure (in C++)?

It is a group of data elements which may have different types of data.



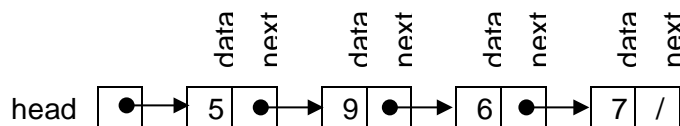
What is Linked List?

A linked list consists of nodes of data which are connected to other nodes.

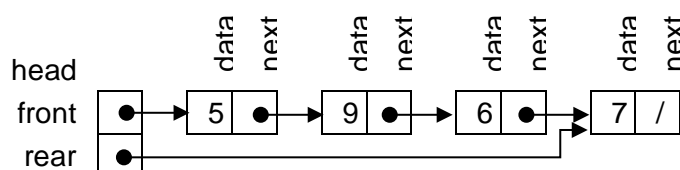
Linked lists can be linear and non-linear data structures however arrays are linear data structures.

There are several types of linked lists.

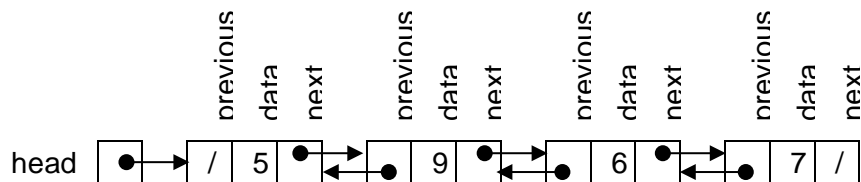
E.g1:



E.g2:



E.g3:



Common operations of Linked List are:

initializeList() - initializes the list as empty list.

insertFirstElt(int elt) - inserts a new element into an empty list.

insertAtFront(int elt) - inserts an element at the beginning of the list.

insertAtEnd(int elt) - inserts an element at the end of the list (appendElt or appendNode).

insertAfter(int oldElt, int newElt) - inserts an element after a specified element.

deleteElt(int elt) - deletes a specified element.

displayList() - displays all the elements in the list

isEmpty() - returns true if the list has no elements, false otherwise.

isFull() - returns false if the list is full, false otherwise.

Structural Diagrams of Linked List Operations:

1. initializeList() head

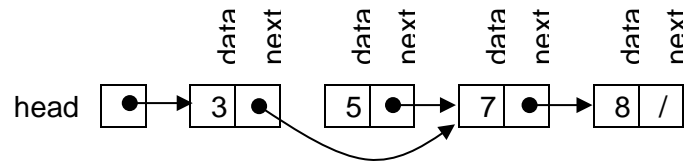
2. insertFirstElt(5) data next
head

3. insertAtFront(3) data next
head

4. insertAtEnd(8) data next data next
head

5. insertAfter(5,7) data next data next data next
head

6. deleteElt(5):



Implementation of Linked List Operations:

```
#include<iostream.h>
```

```
#include<conio.h>
```

```
class LinkedList
```

```
{
```

```
private:
```

```
    struct listNode
```

```
    {
```

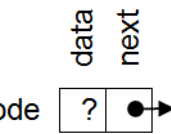
```
        int data;
```

```
        listNode* next;
```

```
    };
```

```
    listNode* head;
```

```
    head
```



```
public:
```

```
    LinkedList();
```

```
    void initializeList();
```

```
    void insertFirstElt(int elt);
```

```
    void insertAtFront(int elt);
```

```
    void insertAtEnd(int elt);
```

```
    void insertAfter(int oldElt, int newElt);
```

```
    void deleteElt(int elt);
```

```
    void displayList();
```

```
    int isEmpty();
```

```
    int isFull();
```

```
}
```

```
LinkedList::LinkedList() //Constructor
```

```
{
```

```
    head=NULL;
```

```
}
```

```
void LinkedList::initializeList()
```

```
{
```

```
    head=NULL;
```

```
}
```

```
head
```



```
void LinkedList::insertFirstElt(int elt)
```

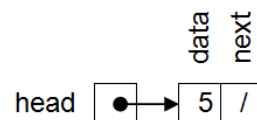
```
{
```

```
    head=new listNode;
```

```
    head->data=elt;
```

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

```
}
```



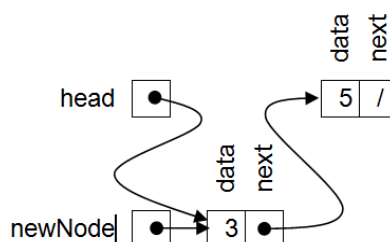
```
void LinkedList::insertAtFront(int elt)
```

```
{
```

```
    listNode *newNode;
```

```
    newNode=new listNode;
```

```
    newNode->data=elt;
```



```

    newNode->next=head;
    head=newNode;
}

```

```

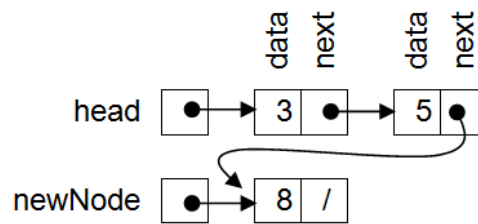
void LinkedList::insertAtEnd(int elt)
{

```

```

    listNode *newNode, *curNode;
    newNode=new listNode;
    newNode->data=elt;
    newNode->next=NULL;
    if (!head)
        head=newNode;
    else
    {
        curNode=head;
        while (curNode->next != NULL)
            curNode = curNode->next;
        curNode->next = newNode;
    }
}

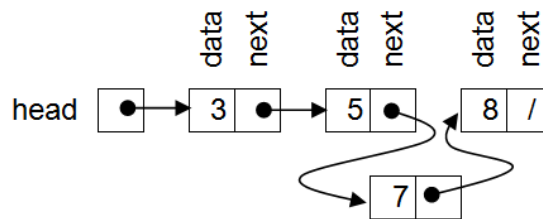
```



```

void LinkedList::insertAfter(int oldElt, int newElt)
{

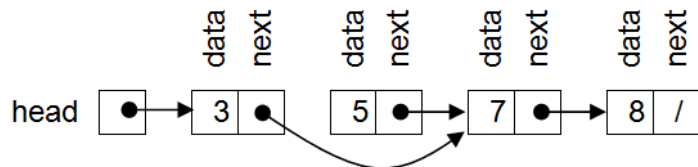
```



```

void LinkedList::deleteElt(int elt)
{

```



```

void LinkedList::displayList()
{

```

```

    listNode* curNode;
    curNode=head;
    while (curNode)
    {
        cout<<curNode->data<<" ";
        curNode=curNode->next;
    }
}

```

```

int LinkedList::isEmpty()
{

```

```

    if (head== NULL)
        return 1;
    else
        return 0;
}

```

```

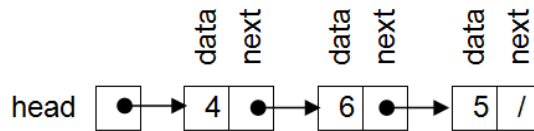
int LinkedList::isFull() //It always returns false.

```

```
{  
    return 0;  
}
```

```
void main()  
{
```

```
    clrscr();  
    LinkedList lst;  
    lst.insertAtEnd(4);  
    lst.insertAtEnd(6);  
    lst.insertAtEnd(5);  
    lst.displayList();  
}
```

**Advantages of Linked List:**

Easy to insert and delete elements.

Unlike array, memory space is not wasted in linked list.

Disadvantages of Linked List:

Slow in searching.