Linked Lists

Linked Lists

- list elements are stored, in memory, in an arbitrary order
- explicit information (called a link) is used to go from one element to the next

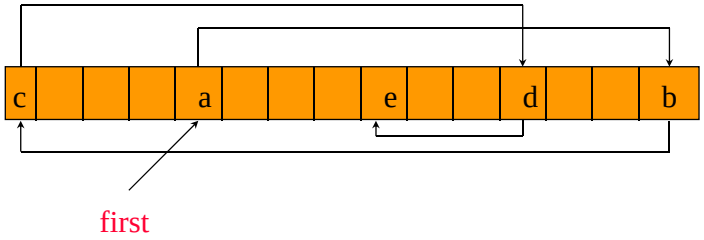
Memory Layout

Layout of L = (a,b,c,d,e) using an array representation.

(^denotes computer memory)

A linked representation uses an arbitrary layout.

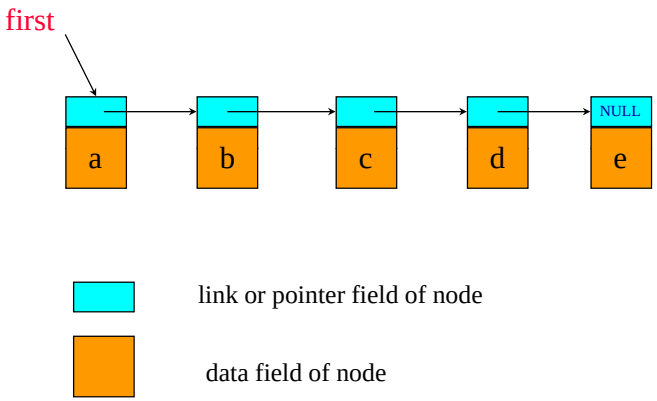
Linked Representation



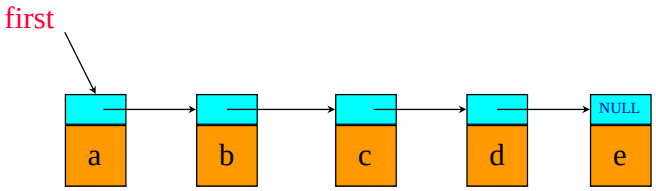
pointer (or link) in e is NULL

use a variable first to get to the first element a

Normal Way To Draw A Linked List



Chain



- A chain is a linked list in which each node represents one element.
- There is a link or pointer from one element to the next.
- The last node has a NULL (or 0) pointer.

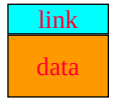
Operations on a linked list

- Possible operations
  - Get an element
  - Remove an element
  - Insert an element
- We first discuss the codes for represent an node, then discuss the operations.

Node and Chain Representation (See Program 4.6 @ P186)

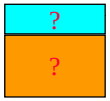
```
template <class T>
class ChainNode
{
  private:
    T data;
    ChainNode<T> *link;

    // constructors come here
};
template <class T>
class Chain
{
  ...
 private:
  ChainNode <T> *first;
}
```

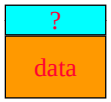


Constructors Of ChainNode

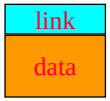
ChainNode() {}



ChainNode(const T& data)
{this->data = data;}

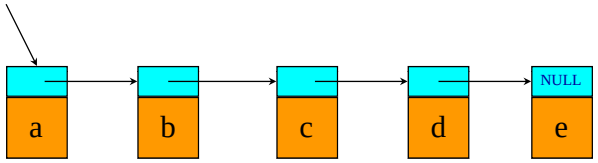


ChainNode(const T& data, chainNode<T>* link)
{this->data = data;
this->link = link;}



Get(0)

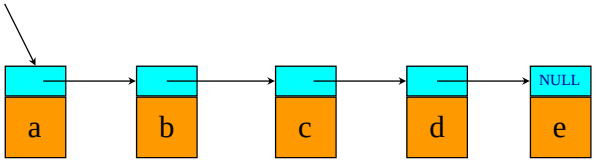
first



desiredNode = first; // gets you to first node
return desiredNode–>data;

Get(1)

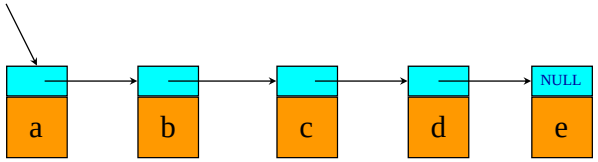
first



desiredNode = first–>link; // gets you to second node
return desiredNode–>data;

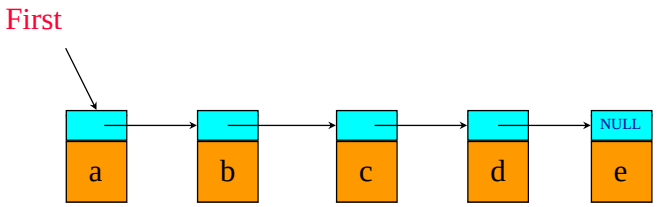
Get(2)

first



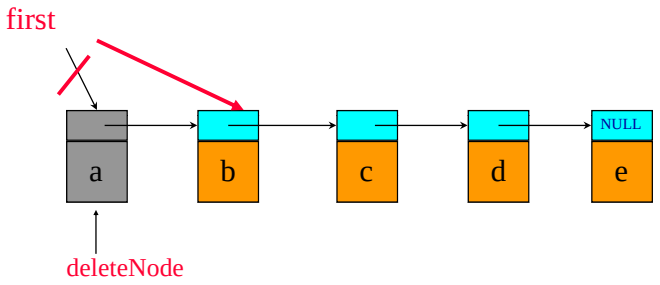
desiredNode = first–>link–>link; // gets you to third node
return desiredNode–>data;

Get(5)



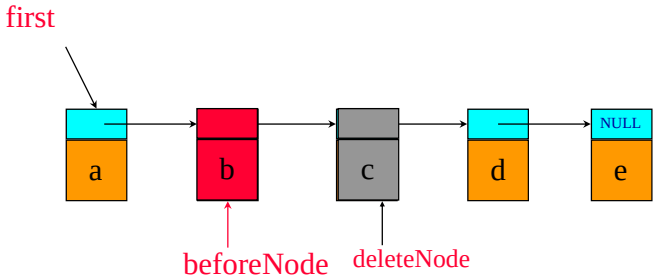

```
desiredNode = first–>link–>link–>link–>link–>link;
                              // desiredNode = NULL
return desiredNode–>data;   // NULL.element
```


Delete An Element (Delete (0))




```
deleteNode = first;
first = first–>link;
delete deleteNode;
```

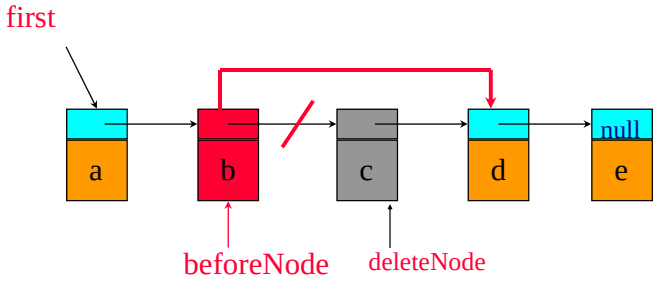

Delete(2)



first get to node just before node to be removed

```
beforeNode = first–>link;
```

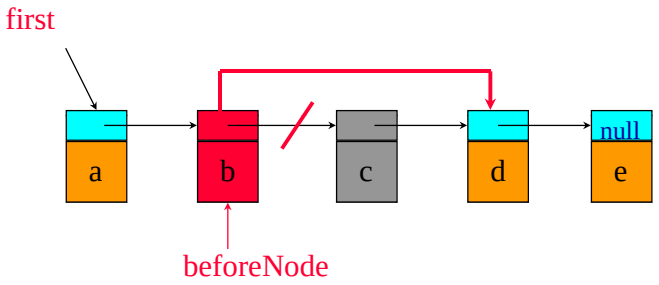

Delete(2)



save pointer to node that will be deleted

```
deleteNode = beforeNode–>link;
```

Delete(2)

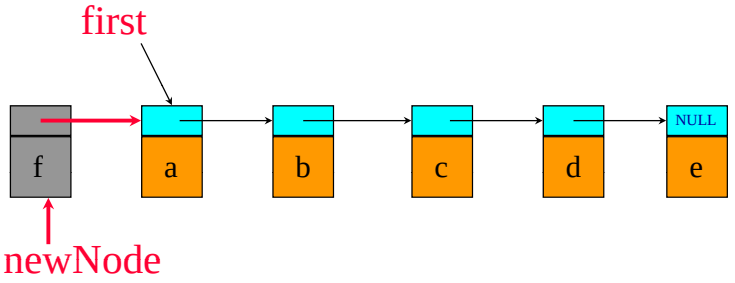


now change pointer in beforeNode

beforeNode–>link = beforeNode–>link–>link;

delete deleteNode;

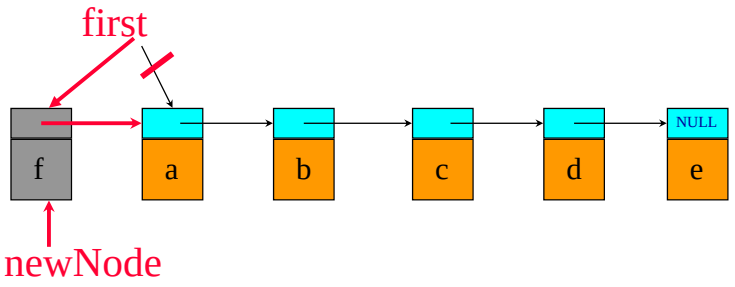
Insert(0,'f')



Step 1: get a node, set its data and link fields

newNode = new ChainNode<char>(theElement,
first);

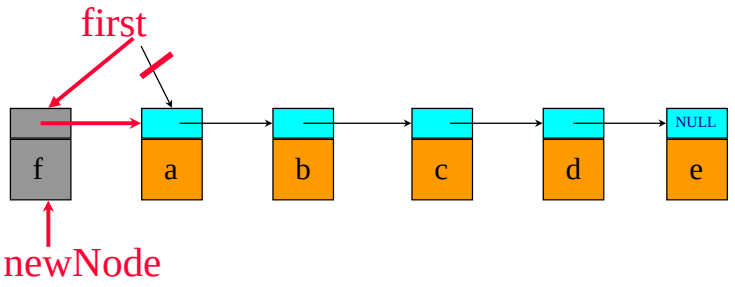
Insert(0,'f')



Step 2: update first

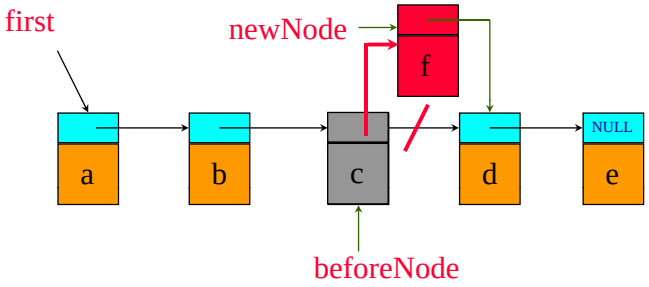
first = newNode;

One-Step Insert(0,'f')



first = new chainNode<char>('f', first);

Insert(3,'f')



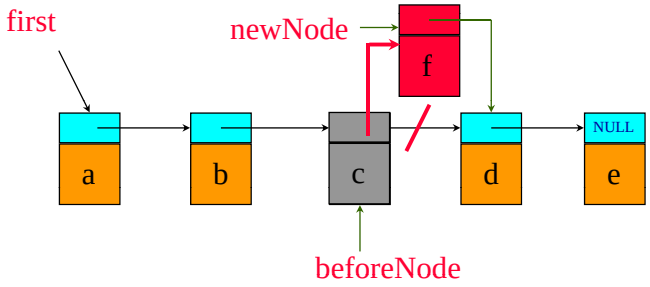
- first find node whose index is 2
- next create a node and set its data and link fields

ChainNode<char>* newNode = new ChainNode<char>('f',
beforeNode–>link);

- finally link beforeNode to newNode

beforeNode–>link = newNode;

Two-Step Insert(3,'f')



beforeNode = first–>link–>link;
beforeNode–>link = new ChainNode<char>
('f', beforeNode–>link);

- A chain is a linked list in which each node (chain node) represents one element。

- 而一個Chainnode 是由一個link和一個data所組成

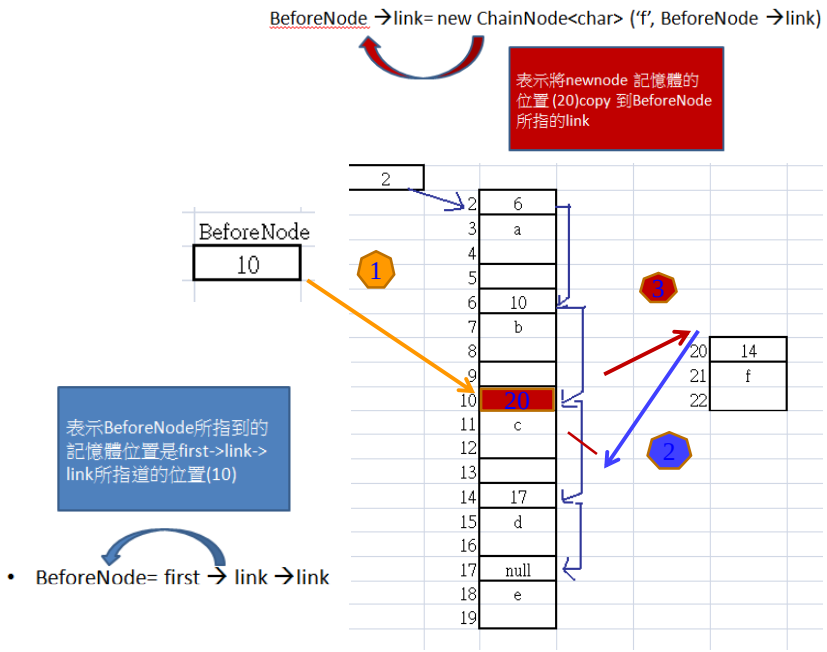
Insert (3, 'f') 步驟分析

表示BeforeNode所指到的記憶體位置是first->link-> link所指到的位置(10)

表示newnode 的link要指到Beforenode→link指到的記憶體位置 (14)

- BeforeNode= first → link →link
- BeforeNode →link= new ChainNode<char> ('f', BeforeNode →link)

表示將newnode 記憶體的位置 (20)copy 到BeforeNode所指的link



In Class Exercise

- Write down the pseudo code for the function insert(int n, char c)