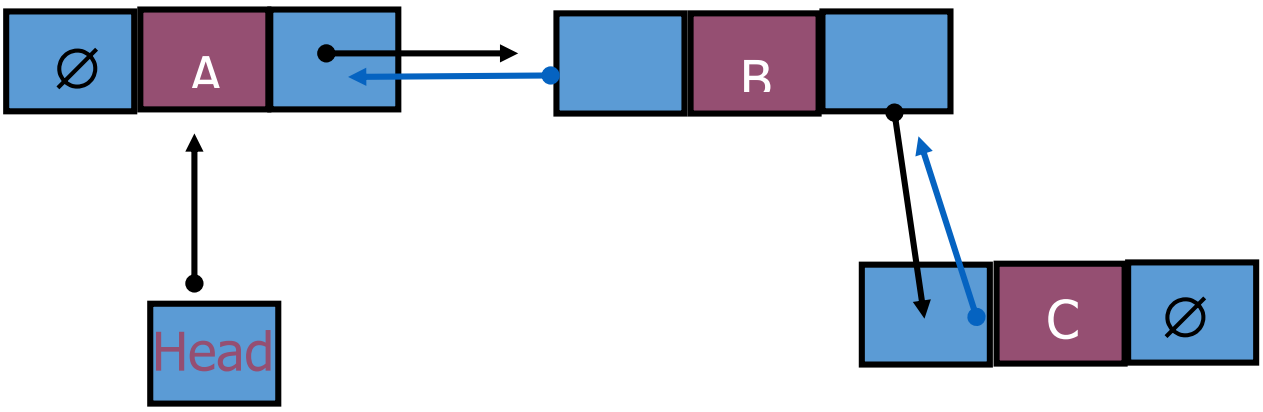
Variations of Linked Lists

Doubly linked lists

* Each node points to not only 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. Convenient to traverse lists backwards



Array versus Linked Lists

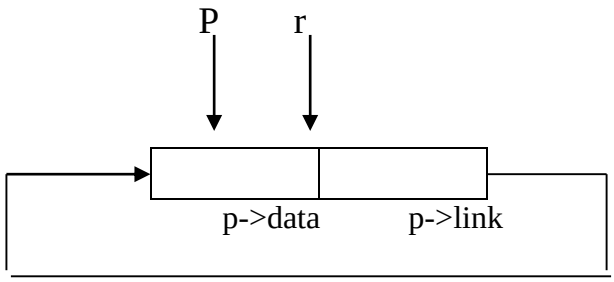
Linked lists are more complex to code and manage than arrays, but they have some distinct advantages

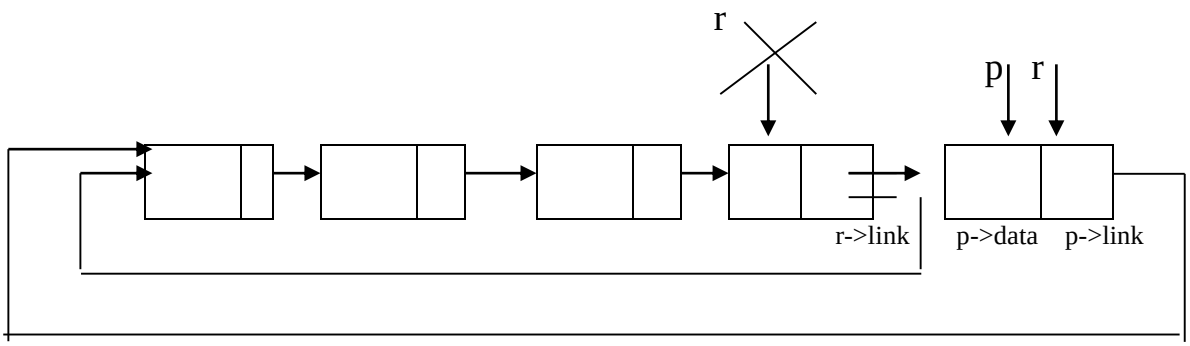
✓ Dynamic: a linked list can easily grow and shrink in size.
* We don't need to know how many nodes will be in the list. They are created in memory as needed.
* In contrast, the size of a C++ array is fixed at compilation time.

✓ Easy and fast insertions and deletions
- To insert or delete an element in an array, we need to copy to temporary variables 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.

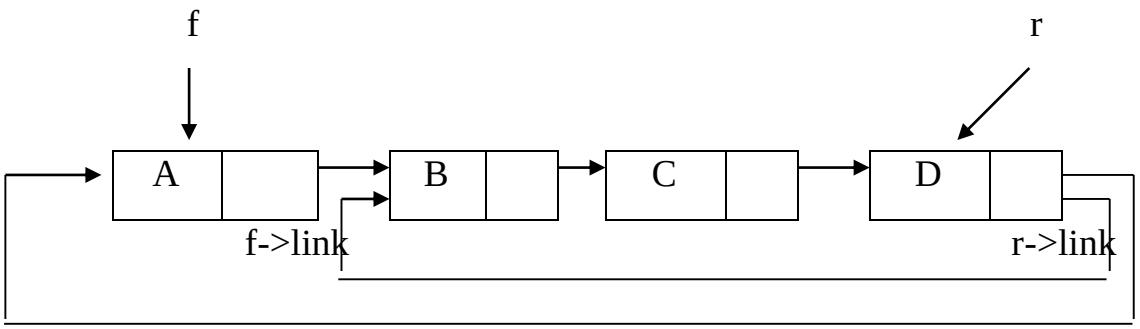
```
#include<iostream.h>
#include<conio.h>
#include<stdlib.h>
Struct node {    int data;
          struct node*link;
         }*start,*p,*r,*f,*q;
void addcll()
{
 p=new node;
 cout<<"input new element"<<endl;
 cin>>p->data;
 if(r==NULL)
 p->link=p;
 else
   {
    p->link=r->link;
    r->link=p;
    }
 r=p;
}
```





p->link=r->link

```
void deletcll()
{
 int item;
 if(r==NULL)
 cout<<"error..theC.L.L is empty"<<endl;
 else
   {
    f=r->link;
    item=f->data;
    if(r==f)
    r=NULL;
    else
    r->link=f->link;
    delete(f);
    }
}
```



r->link=f->link