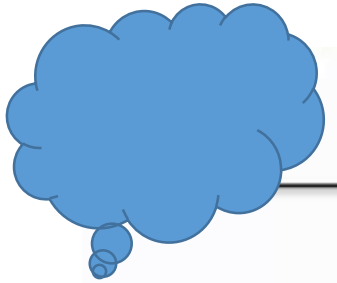
ساختمان داده ها والگوریتم ها

فصل پنجم

لیست پیوندی



مقدمه

فرض کنید آرایه مرتب شده ای داریم که عناصر زیر به ترتیب در آن قرار دارند:

Array={BAT, CAT, EAT, FAT, HAT, JAT, SAT, WAT}

برای تکمیل این لیست می خواهیم کلمه GAT را اضافه کنیم.

به دلیل این که این عناصر در آرایه ذخیره شده اند، برای اضافه کردن کلمه GAT باید همه عناصر بعد از آن را جا به جا کنیم.

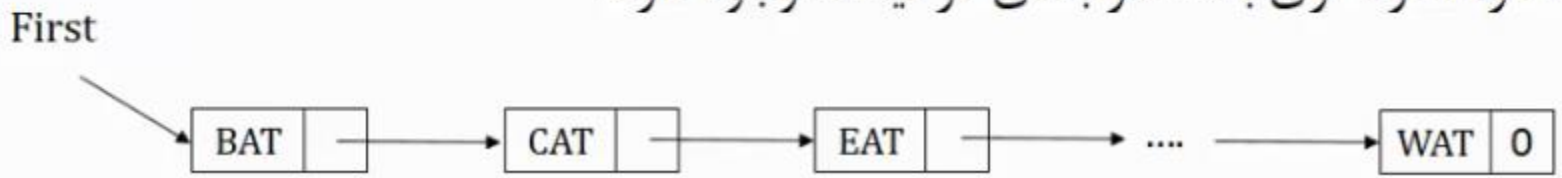
راه حل این مشکل استفاده از لیست پیوندی می باشد.

در نمایش ترتیبی با آرایه، عناصر به همان ترتیبی که در لیست مرتب شده قرار دارند در حافظه جای میگیرند. اما در لیست پیوندی لازم نیست که عناصر به همان ترتیب در حافظه قرار گیرند.

لیست پیوندی

❖ لیست پیوندی دارای عناصر مختلفی از حافظه پویا بوده که لزوما عناصر آن کنار هم قرار نگرفته اند.

❖ در لیست پیوندی برای هر عنصر، اشاره گری به عنصر بعدی در لیست وجود دارد.



	داده	اتصال
1	HAT	15
2		
3	CAT	4
4	EAT	8
5		
6	WAT	0
7	BAT	3
8	FAT	1
	.	.
	.	.

لیست پیوندی

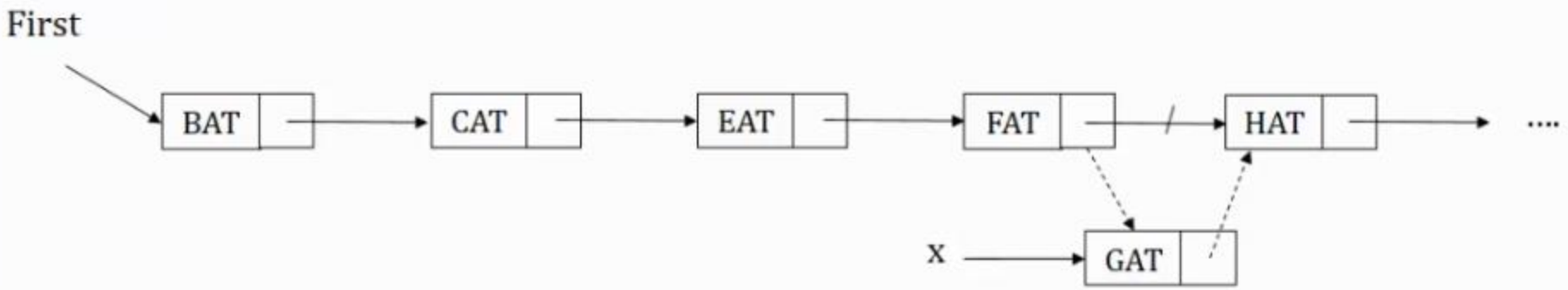
مراحل درج کلمه GAT بین FAT و HAT:

۱) گره ای جدید از نوع ساختار داده استفاده شده ایجاد می کنیم.

۲) فیلد داده (data) این گره را برابر GAT قرار دهید.

۳) فیلد اتصال (link) x را طوری مقداردهی کنید که به گره بعد از FAT یعنی HAT اشاره کند.

۴) فیلد اتصال (link) گره حاوی مقدار FAT را برابر x قرار دهید.



نمایش لیست ها در C++

نحوه دسترسی به فیلد های هر گره
First → data
First → link

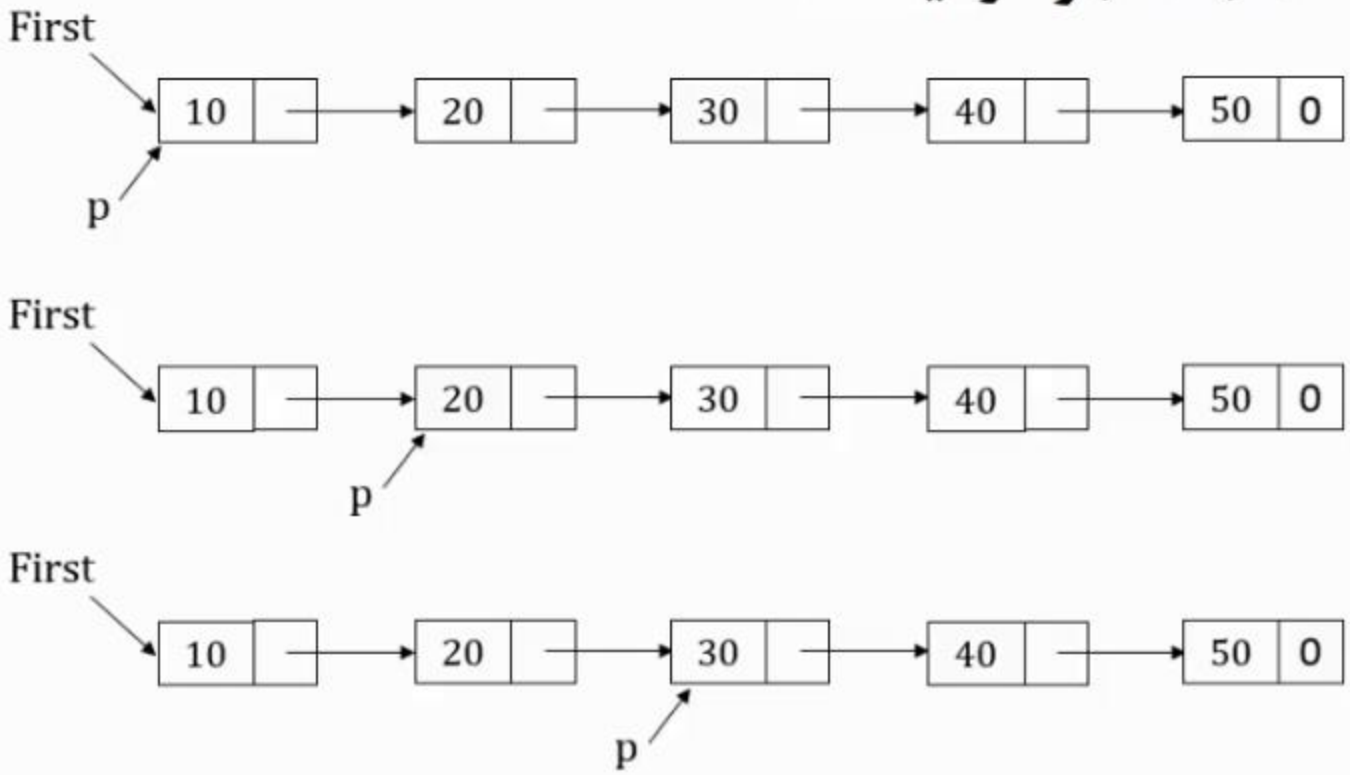
```
Struct node{
        type    data;
        node    *link;
};


//main
node *first;
first = new node;
```

★ مهم ترین و تنها چیزی که از یک لیست پیوندی در اختیار ماست، آدرس شروع لیست، یعنی first می باشد. بنابراین اگر آدرس اولین گره را از دست بدهیم، دیگر به هیچ یک از گره ها دسترسی نخواهیم داشت.

اعمال اصلی روی لیست های پیوندی

```
node *p;
p = first;
while ( p != null  &&  p→data !=x)
        p = p→link;
if (p==null)
        cout<< "not found";
else
{
...
}
```

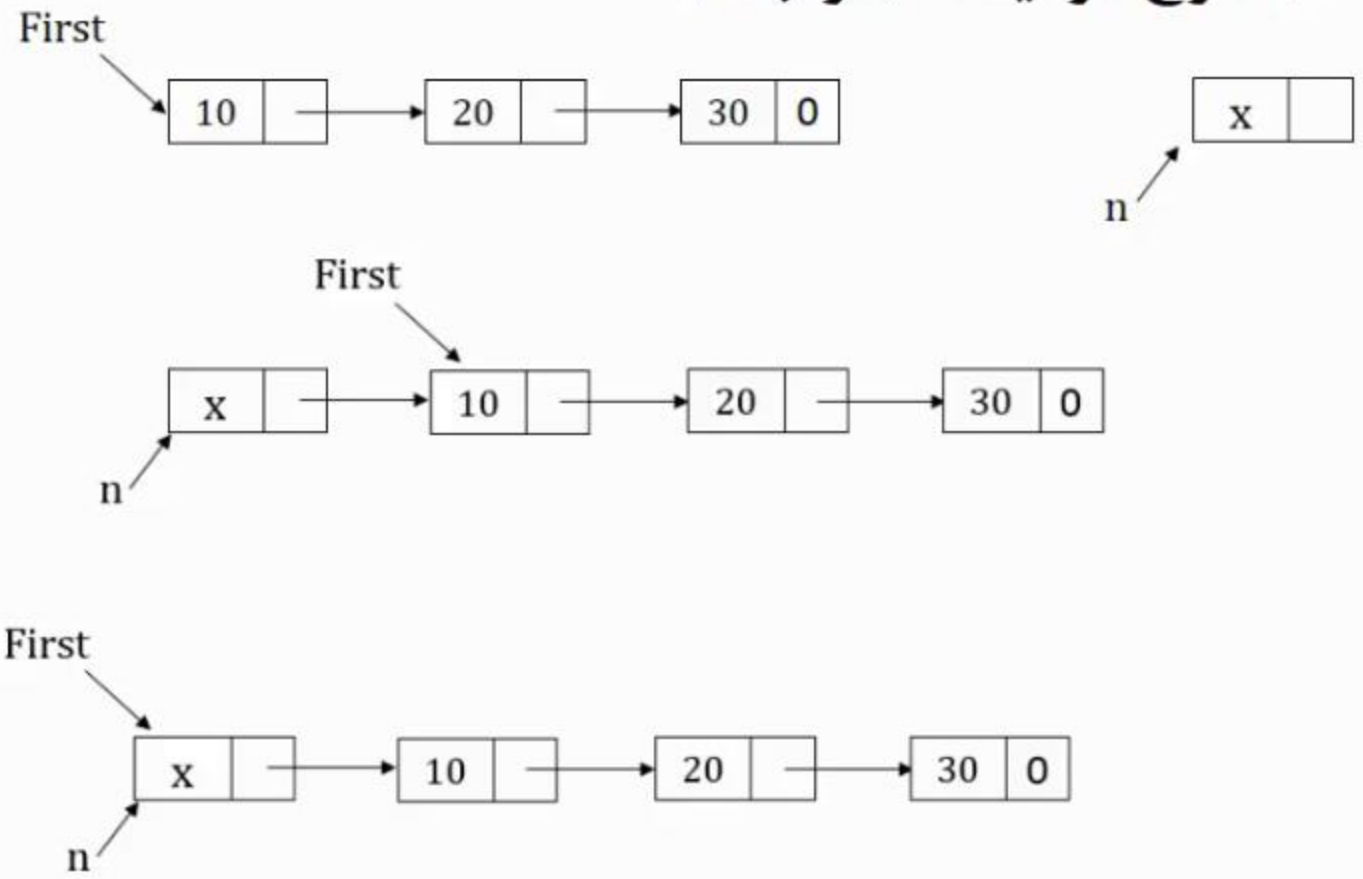
۱. جستجو در لیست:



اعمال اصلی روی لیست های پیوندی

```
node *n;
n = new node;
n → data = x;
n → link = first;
first = n;
```

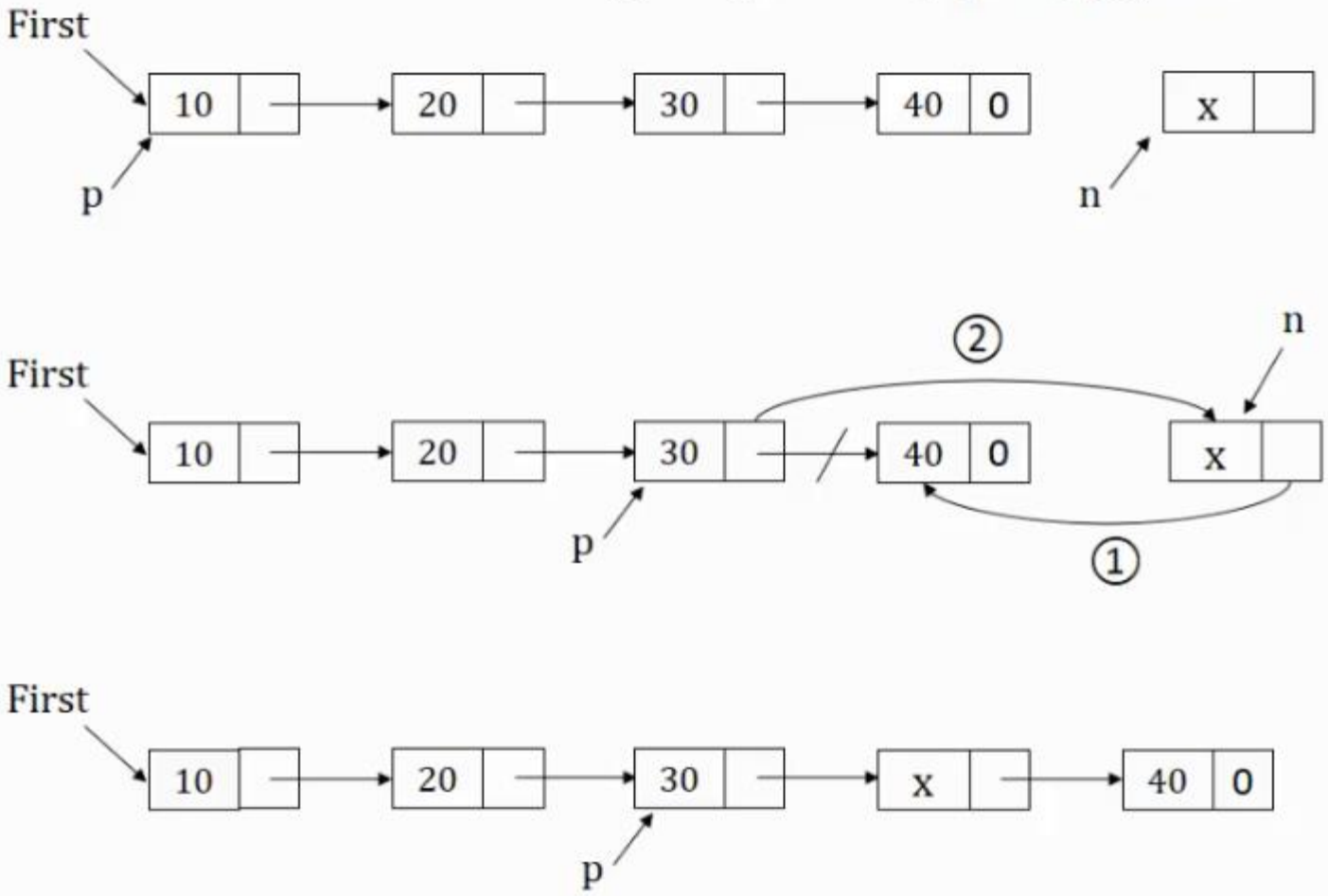
۲. درج در لیست(در ابتدا):



اعمال اصلی روی لیست های پیوندی

۳. درج بعد از یک عنصر خاص :

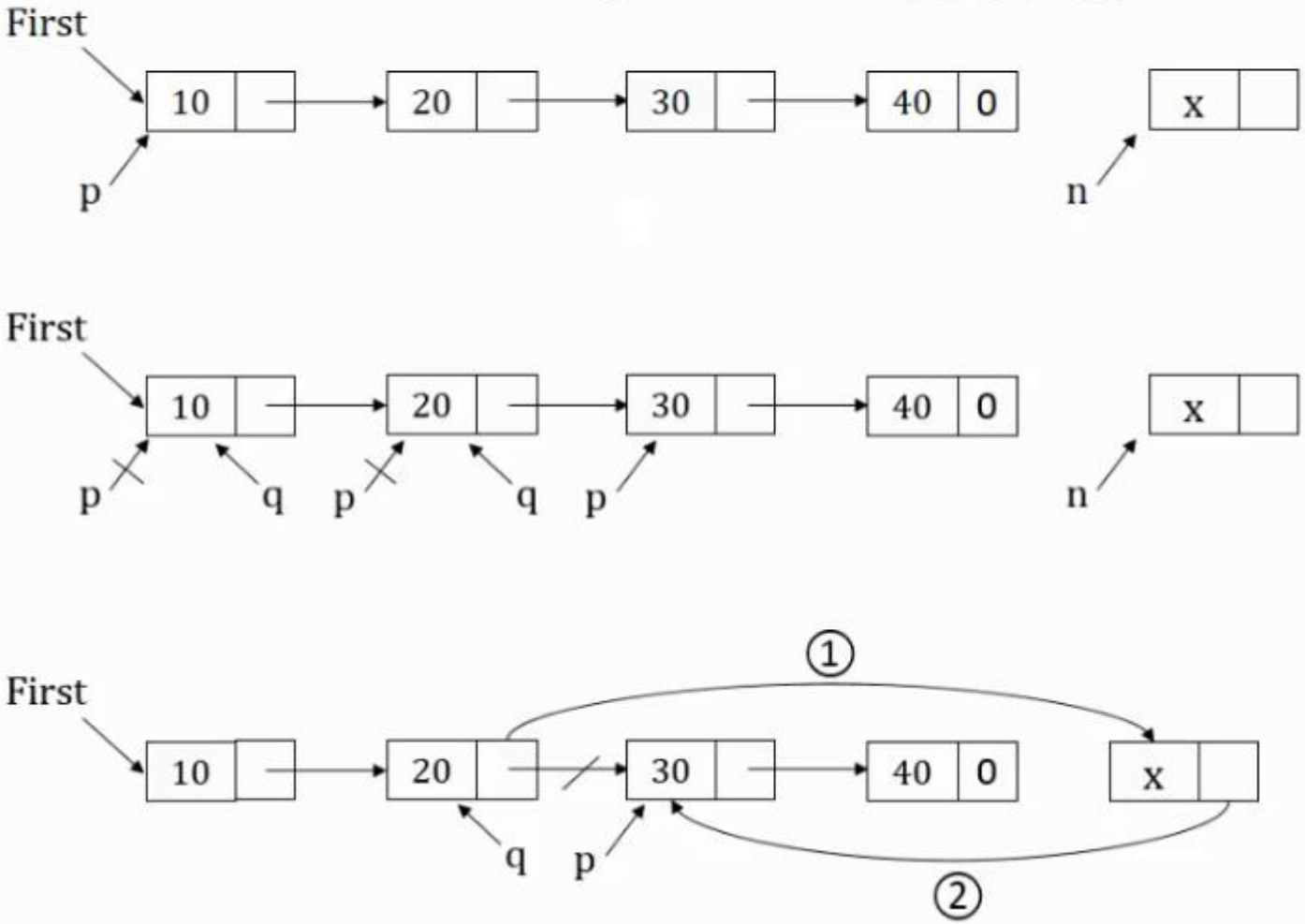
```
node  *n,*p;
n = new node;
n → data = x;
p = first;
while( p != null && p→data!=y)
        p = p→link;
n→link = p→link;
p→link = n
```



اعمال اصلی روی لیست های پیوندی

```
node  *n,*p,*q;
n = new node;
n → data = x;
p = first;
while( p != null && p→data!=y){
        q=p;
        p = p→link;}
q→link = n;
n→link = p;
```

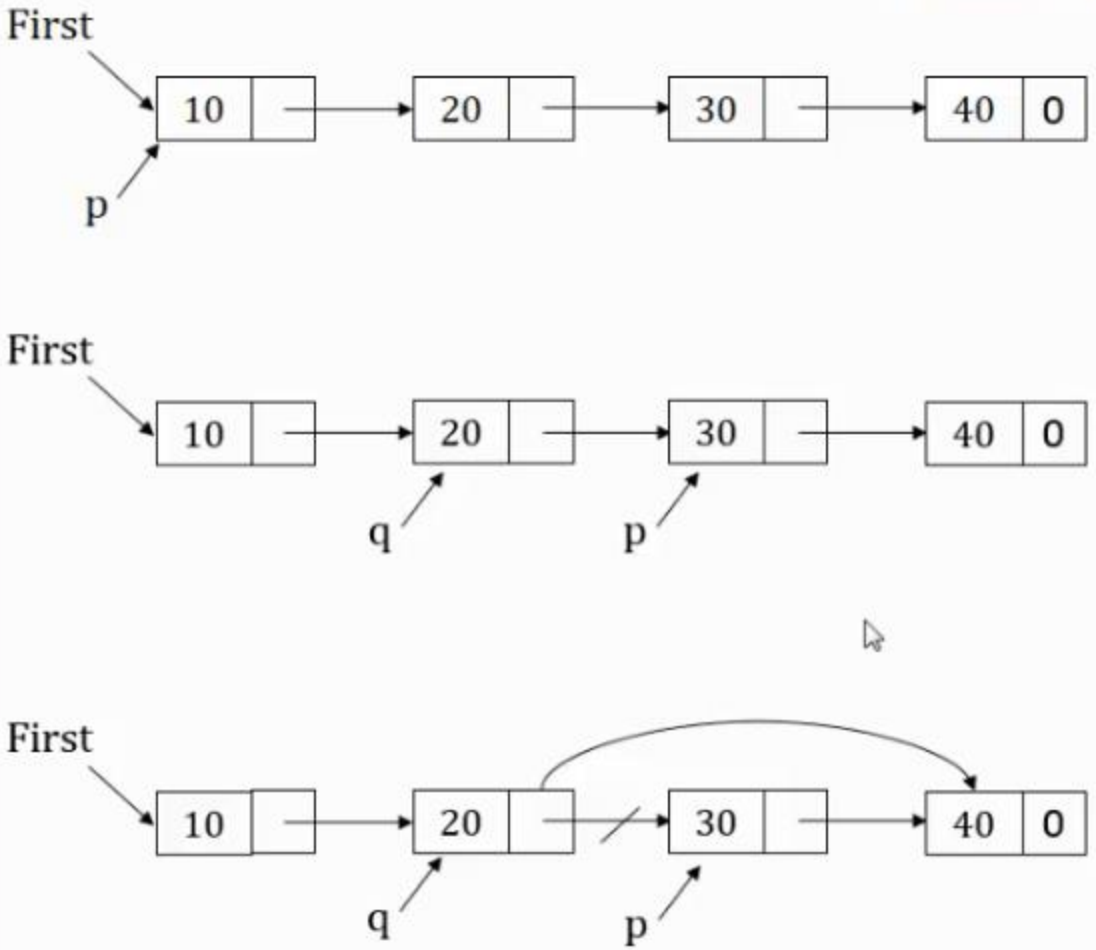
۴. درج قبل از یک عنصر خاص :



اعمال اصلی روی لیست های پیوندی

۵. حذف یک عنصر :

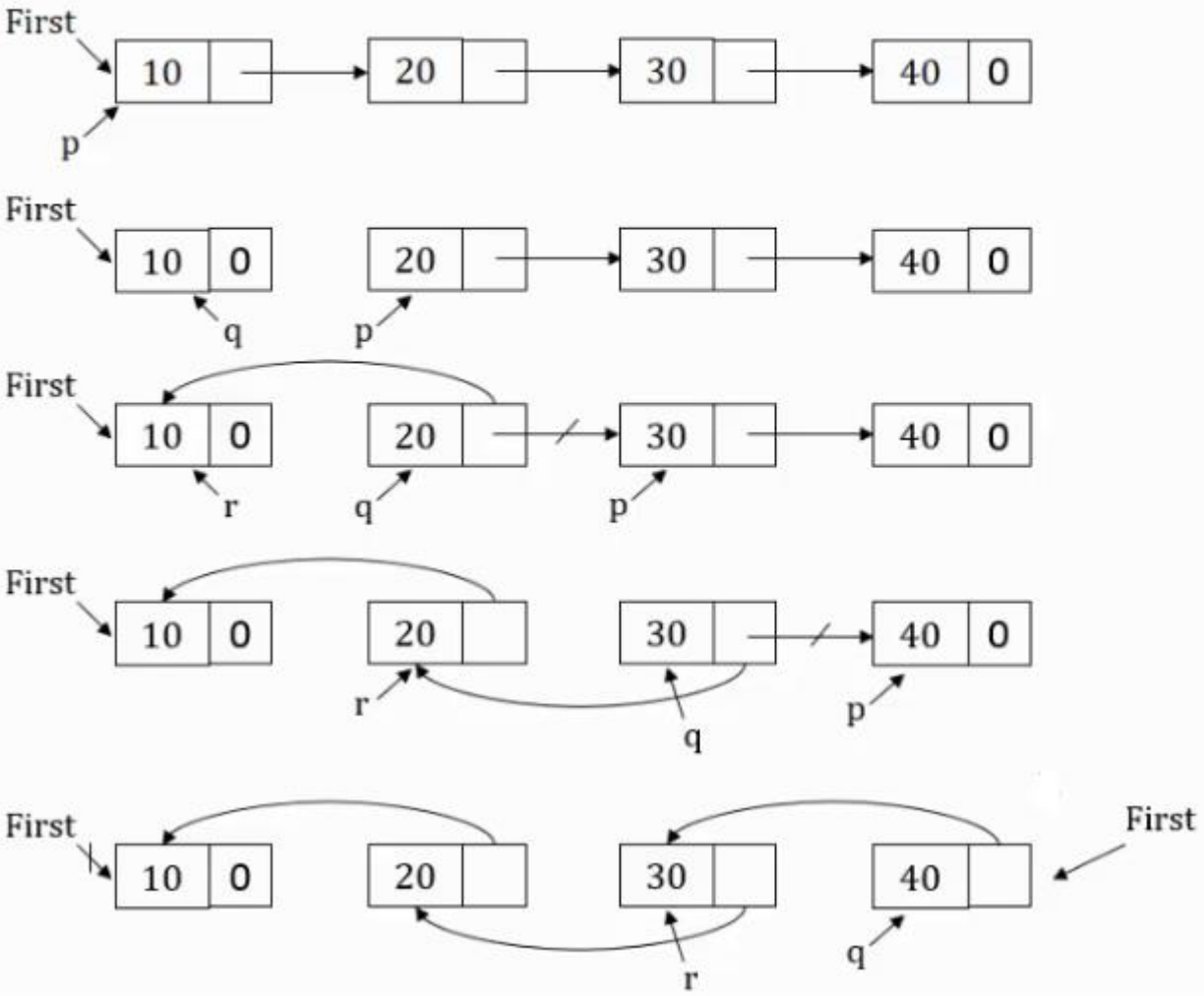
```
node  *p ,*q;
type    item;
p = first;
while( p != null && p→data!=y){
            q=p;
            p = p→link;}
q→link = p→link;
Item = p→data;
delete   p;
```



اعمال اصلی روی لیست های پیوندی

۶. معکوس کردن لیست پیوندی:

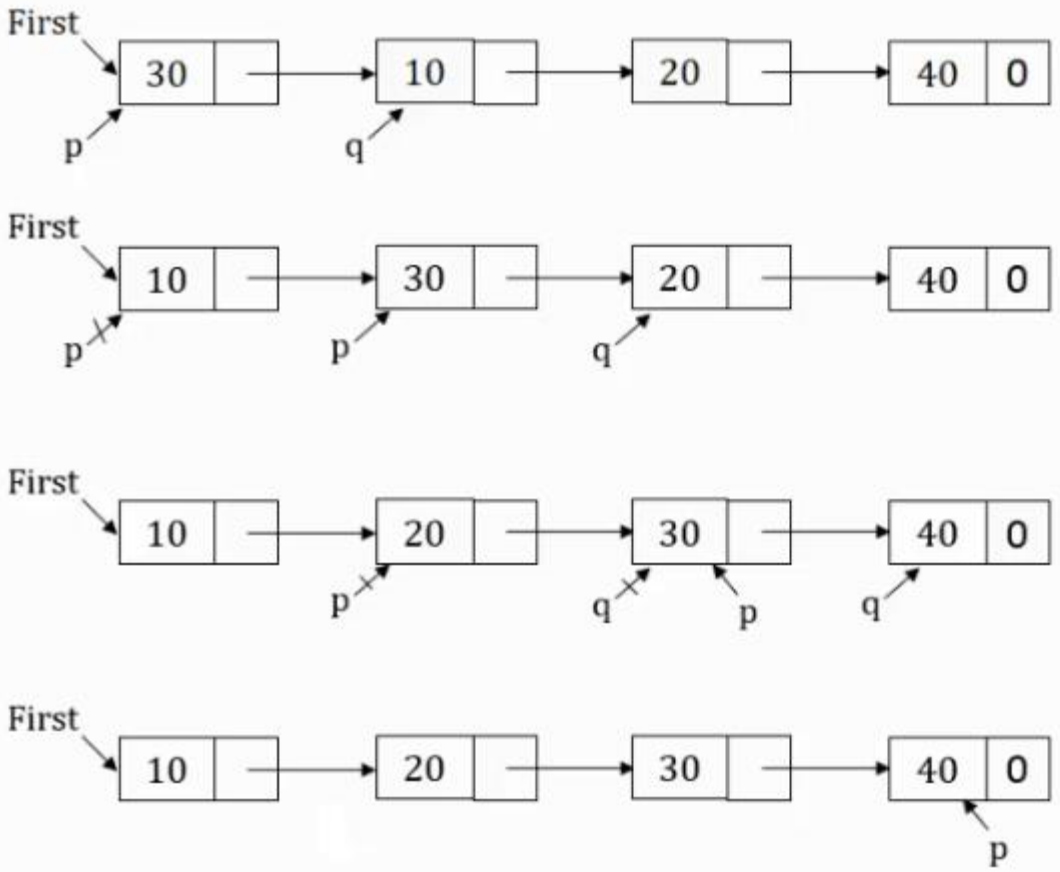
```
node  *p ,*q,*r;
p = first;
q = null;
while( p != null){
        r = q;
        q = p;
        p = p→link;
        q→link=r;}
First = q;
```



اعمال اصلی روی لیست های پیوندی

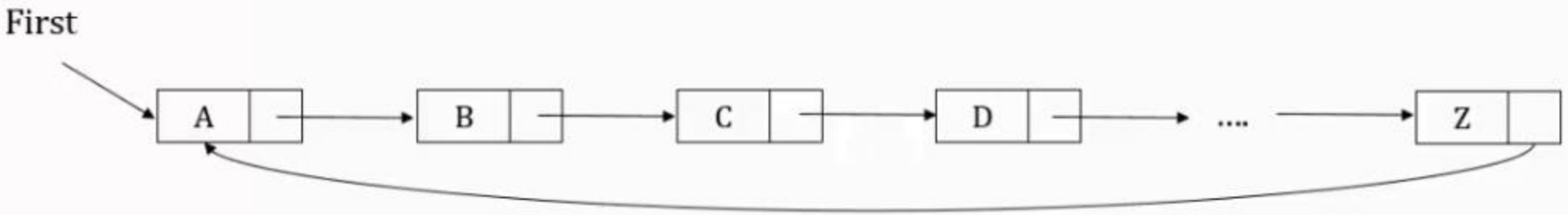
۷. مرتب سازی حبابی:

```
node *p ,*q;
p = first
while( p != null)
{
        q = p→link;
        while (q!=null)
{
                if( q→data<p→data )
                        swap(q,p)
                q = q→link;
}
        p = p→link;}
```



لیست پیوندی حلقوی

اگر در لیست پیوندی یک طرفه، اشاره گر link آخرین گره به اولین گره اشاره کند، به لیست حاصل شده لیست پیوندی حلقوی گویند.



مزیت لیست پیوندی حلقوی بر لیست پیوندی خطی:

در لیست حلقوی بدون نیاز به هیچ حافظه اضافی با داشتن آدرس یک گره دلخواه امکان دسترسی به گره قبلی وجود دارد، اما در لیست خطی یک طرفه این امکان وجود نداشت و دسترسی به گره های قبل از یک گره فقط از طریق آدرس شروع لیست وجود دارد.

لیست پیوندی حلقوی

پیمایش لیست خطی:

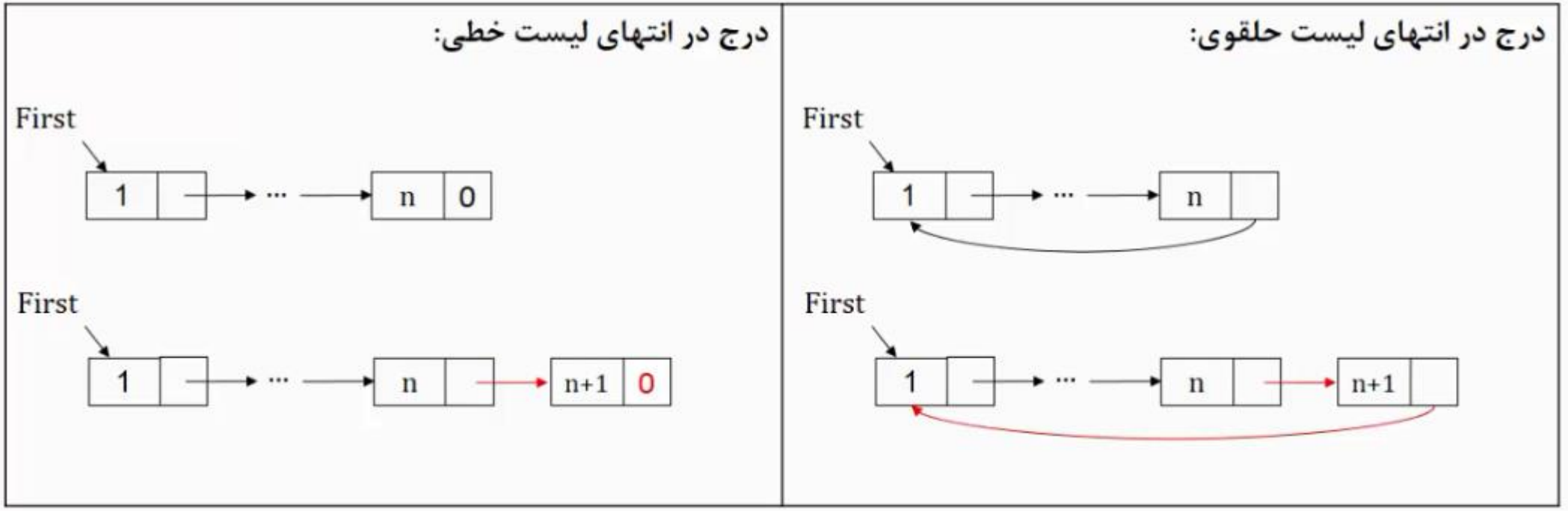
```
node *p;
p = first;
while(p!=null){
        ...
        p = p→link;
        ....
        }
```

پیمایش لیست حلقوی:

```
if( first != null)}
        node *p;
        p = first;
        do{
                ...
                p = p→link;
                ...
        }while(p!=fisrt)  }
```

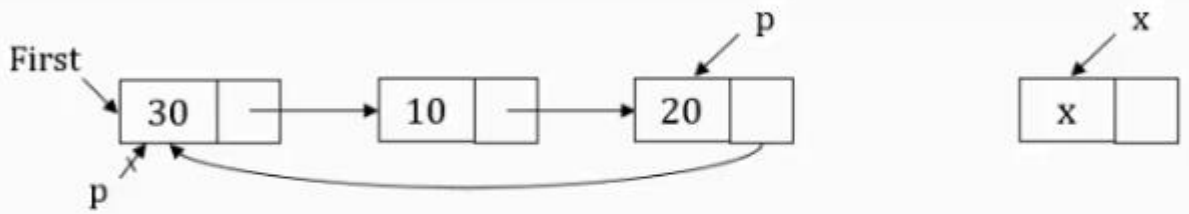
عملیات در لیست های حلقوی

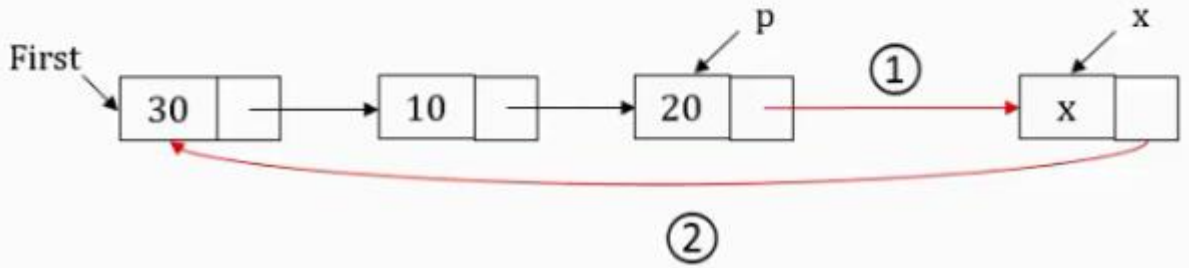
★ عملیات در لیست های حلقوی مانند عملیات در لیست های خطی است با این تفاوت که باید **شرط پایان حلقه ها** و **نحوه اصلاح اشاره گر گره آخر** با توجه به حلقوی بودن لیست تغییر می کند.

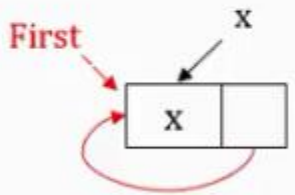


عملیات در لیست های حلقوی

```
if (first != null){
node  *p;
p = first
do{
        p=p→link;
}while(p→link != first)
p→link = x;
x→link = first;}//end of if
else{
first = x;
x→link = x;}//end of else
```







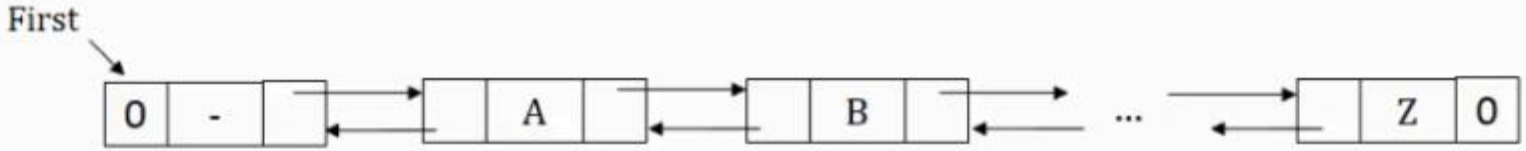
اشاره گر first در لیست حلقوی

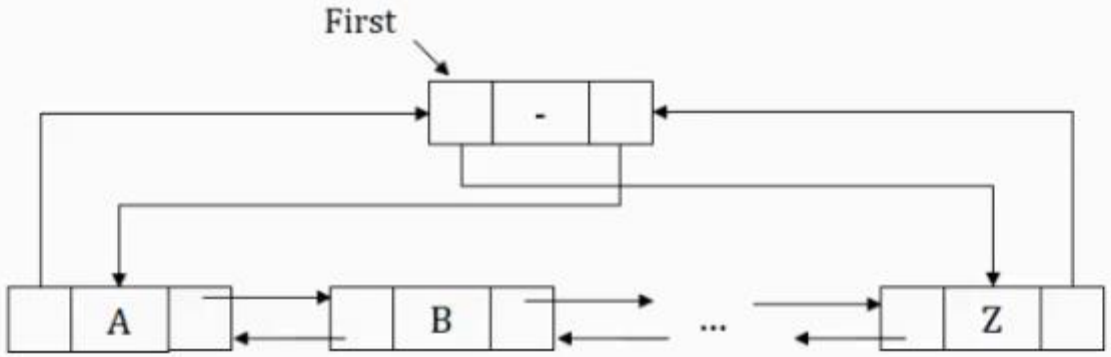
اگر اشاره گر first به اولین یا آخرین گره لیست حلقوی اشاره داشته باشد، وضعیت های متفاوتی از نظر مرتبه زمانی برای بعضی از عملیات ها به وجود می آید:

اشاره به اولین گره		اشاره به آخرین گره	
عملیات	مرتبه زمانی	عملیات	مرتبه زمانی
درج در ابتدا	O(n)	درج در ابتدا	O(1)
حذف از ابتدا	O(n)	حذف از ابتدا	O(1)
درج در انتها	O(n)	درج در انتها	O(1)
حذف از انتها	O(n)	حذف از انتها	O(n)

لیست های پیوندی دو طرفه

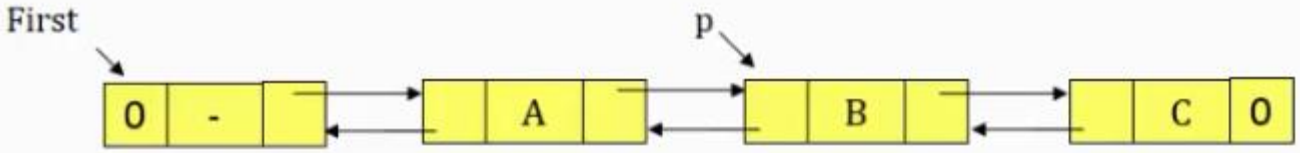
در لیست های پیوندی دو طرفه هر گره به کمک دو اشاره گر left و right به عناصر قبل و بعد خود اشاره می کند.

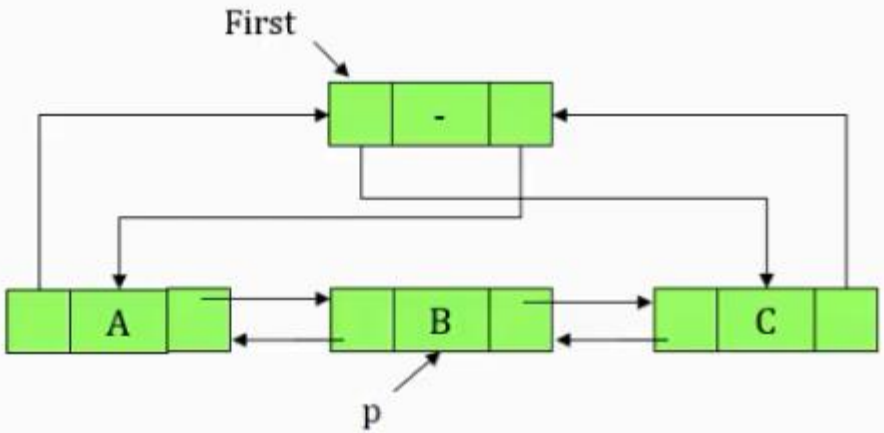




حذف از لیست پیوندی دو طرفه

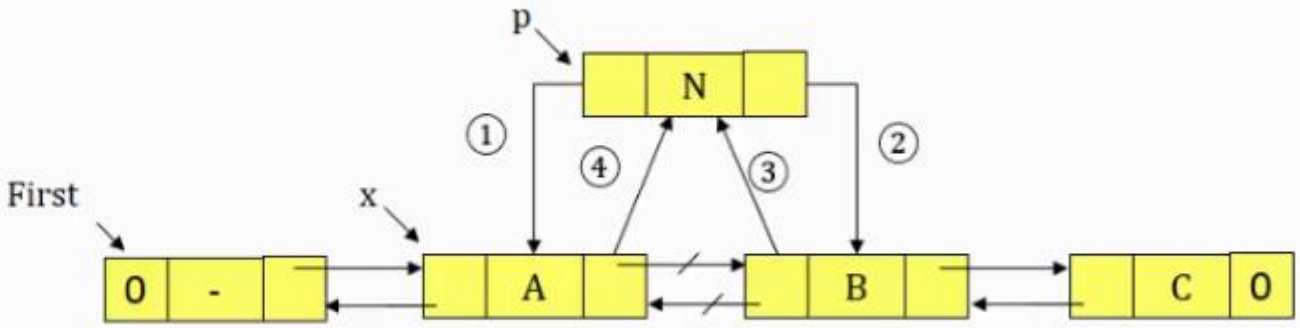
```
void delete(node p, node head)
{
if ( p == head ){
        cout<<"warning";
        exit(1);}
if (p→Rlink==0){
        p→Llink→Rlink=p→Rlink;
        delete p;}
else{
p → Llink → Rlink = p→Rlink;
p → Rlink → Llink = p→Llink;
delete p;}
}
```



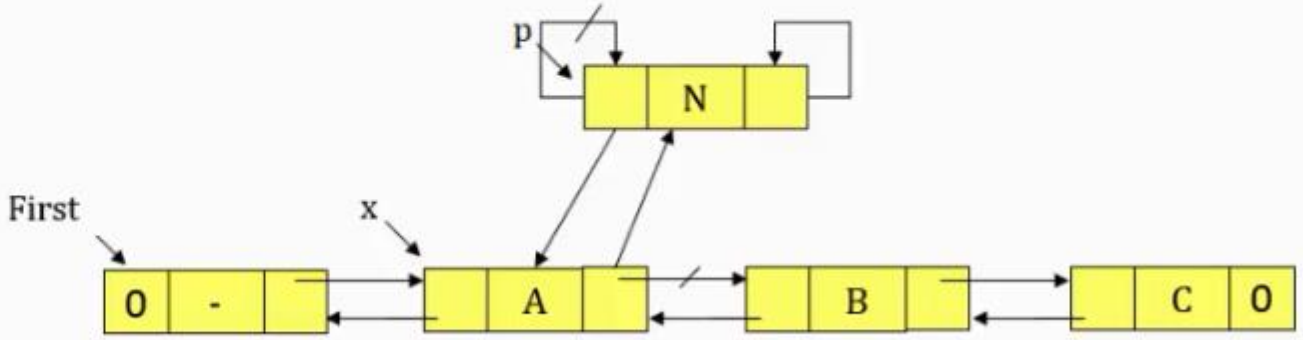


درج در لیست پیوندی دو طرفه

```
void insert(node p, node x)
{
① p→Llink = x ;
② p→Rlink = x→Rlink ;
③ x→Rlink→Llink = p ;
④ x→Rlink = p ;
}
```



④ → ③ → ② → ①



پیاده سازی پشته و صف با استفاده از لیست پیوندی

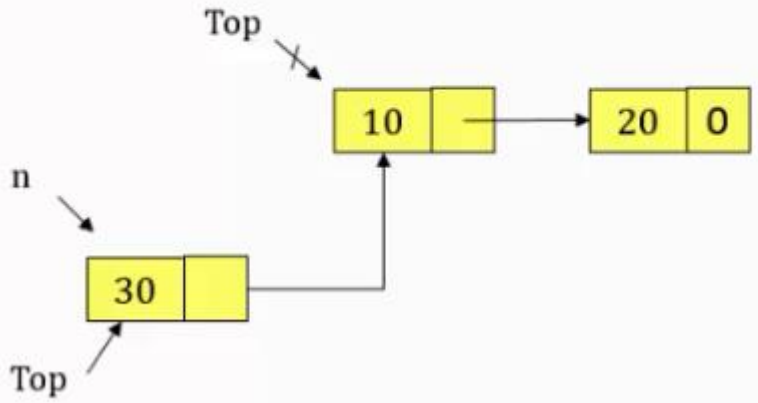
پشته پیوندی:

- برای پیاده سازی از لیست پیوندی یک طرفه معمولی استفاده می کنیم.
- اشاره گر شروع لیست پیوندی همان Top پشته خواهد بود.
- شرط خالی بودن پشته این است که Top=Null باشد.
- شرط پر بودن وجود ندارد. (مگر اینکه به صورت صریح تعداد عناصر پشته محدود گردد)
- Push کردن و pop کردن معادل درج در ابتدای پشته و حذف از ابتدای پشته می باشد.

پیاده سازی پشته و صف با استفاده از لیست پیوندی

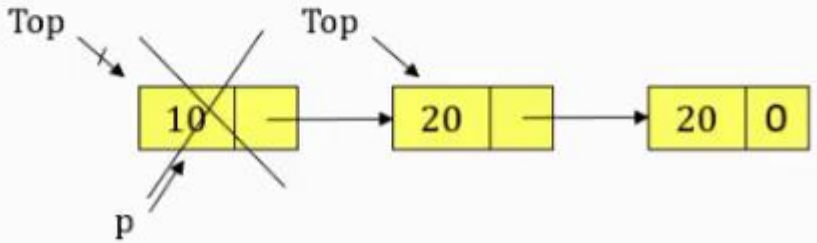
:push

```
node *n;
n = new node;
n→data = data;
n→link = Top;
Top = n;
```



:pop

```
if( Top == Null)
        cout<<“stack is empty;
else{
        x=top→data;
        p = top;
        Top = Top→link;
        delete p;}
```



پیاده سازی پشته و صف با استفاده از لیست پیوندی

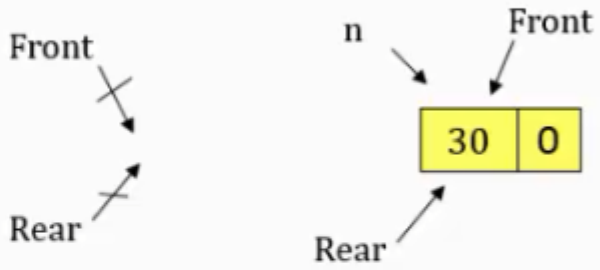
صف پیوندی:

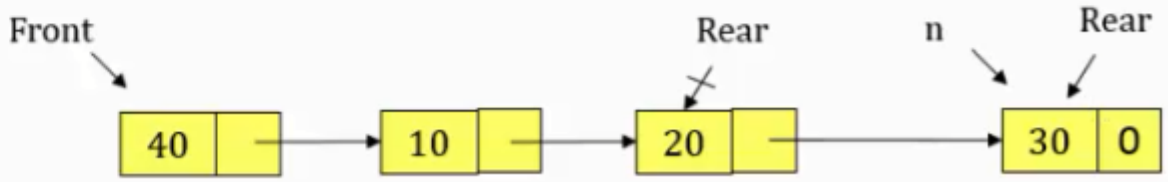
- برای پیاده سازی از لیست پیوندی یک طرفه معمولی استفاده می کنیم.
- دو اشاره گر به نام های front و rear که به ترتیب به ابتدا و انتهای صف اشاره می کنند
- شرط خالی بودن صف این است که front=rear=Null باشد.
- شرط پر پودن وجود ندارد. (مگر اینکه به صورت صریح تعداد عناصر صف محدود گردد)
- درج در صف معادل درج در انتهای لیست و حذف از آن معادل حذف از ابتدای لیست می باشد.

پیاده سازی پشته و صف با استفاده از لیست پیوندی

:insert

```
node *n;
n = new node;
n→data = data;
if ( front == null && rear == null)
        front = n;
else
        rear→link = n;
n→link = null;
rear = n
```





پیاده سازی پشته و صف با استفاده از لیست پیوندی

:delete

```
if ( front == null )
        cout<<“queue is empty”;
else{
        x = front→data;
        t = front;
        front = front→link;
        delete t;
}
```