اصول ساختمان داده ها در c++

ساختمان داده ها شاخه ای از مهندسی نرم افزار است که به مطالعه انواع ساختارهای داده و نیز الگوریتم های مربوط به آنها می پردازد.

گلزاری پور

آموزشکده فنی و حرفه ای شهید بهشتی کرج

رکورد[1]

رکورد ساختمانی است ترتیبی از عناصر که لزوما هم نوع نیستند به هر عنصر از رکورد یک فیلد گفته می شود.

در C++ رکورد را با کلمه کلیدی struct تعریف می شود

مثال

تعریف رکورد دانشجو با فیلدهای شماره دانشجویی ، نام و معدل ؟

```
Struct Student{
Int stno;
Char name [40];
Float avr ; } ;
Student s ;
```

برای دستیابی به هر فیلد از رکورد باید ابتدا نام متغیر رکورد و سپس نام فیلد را ذکر کنیم بین این باید علامت نقطه قرار دهیم.

نام فیلد . نام متغیر فیلد

```
S.stno
S. name
S. avr
```

اشاره گر[2]

اشاره گر متغیری است که می تواند آدرس ناحیه ای از حافظه را نگهداری نماید به عبارت دیگر می تواند آدرس متغیر دیگری در حافظه را نگهداری نماید .و به صورت زیر تعریف می شود :

< نوع > * < نام اشاره گر >;

```
Int * p ;
Float * q ;
Student * r ;
```

نکته

اگر p یک اشاره گر باشد p* محتوای خانه ای از حافظه را که این اشاره گر به آن اشاره می کند نتیجه میدهند و بر عکس اگر x یک متغیر باشد &x آدرس آن را در حافظه نتیجه می دهد .

مثال:

[1] Record
[2] Pointer

```
Int x=5 , y ;
Int *p ;
P = &x ; x آدرس
Y = *p ;// 5
Cout<<y;  // (5)
```

نکته اگر p یک اشاره گری به یک رکورد باشد از نماد (فلش) برای دستیابی به هر فیلد از آن رکورد استفاده می شود اشاره گری که به جایی اشاره نمی کند Null یا تهی نامیده می شود .

دستورات حافظه پویا[3]

برای کار با حافظه پویا در ++c از توابع Delete و New استفاده می کنیم .

تابع New

این تابع برای تخصیص حافظه به اندازه معین به کار میرود . تابع New پس از تخصیص حافظه آدرس ابتدای حافظه تخصیص داده شده را برمیگرداند . خروجی تابع در صورت عدم موفقیت مقدار Null خواهد بود.

تابع Delete

از این تابع جهت آزادسازی حافظه ای که قبلا تشخیص داده شده است استفاده می شود .

مثال

```
int *p ;
P = new int ;
.
.
Delete p ;
```

```
Student *s ;
S = new student ;
.
.
Delete s ;
```

```
Int *a ;
A = new int [10] ;
.
.
Delete [ ] a ;
```

[3] Dynamic memory alocation

لیست پیوندی : ساختارها

آرایه

پشته

صف

ترتیبی + منطقی و فیزیکی

معایب ساختارهای ترتیبی

۱. نیاز به حجم انتقال زیادی از داده ها در عملیات درج و حذف دارند .

۲. اغلب تخصیص حافظه در آنها ایستا ۱ است و باعث هدر رفتن حافظه می گردد.

لیست پیوندی[4]

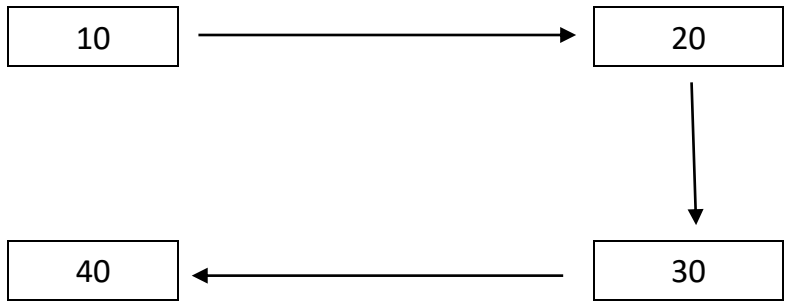
لیست پیوندی ساختاری است غیر ترتیبی از عناصر که ترتیب منطقی آن توسط اشاره گرها تامین می شوند .

گره[5]

به هر عنصر در لیست پیوندی اصطلاحا گره گفته می شود. هر گره از لیست پیوندی رکوردی با دو بخش داده و آدرس است . فیلدهای آدرس در واقع همان اشاره گرهایی هستند که برای ایجاد ارتباط منطقی بین گره های لیست مورد استفاده قرار می گیرد.

۱۰	۲۰	۳۰	۴۰
p	P+1	P+2	P+3

نمایش ترتیبی لیست پیوندی

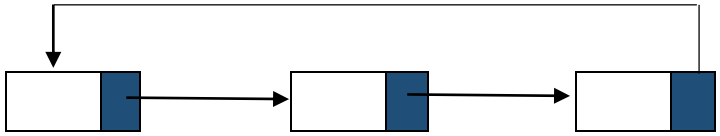


۱- لیست تک پیوندی[6](یک طرفه)

۲- لیست دوپیوندی[7](دو طرفه)

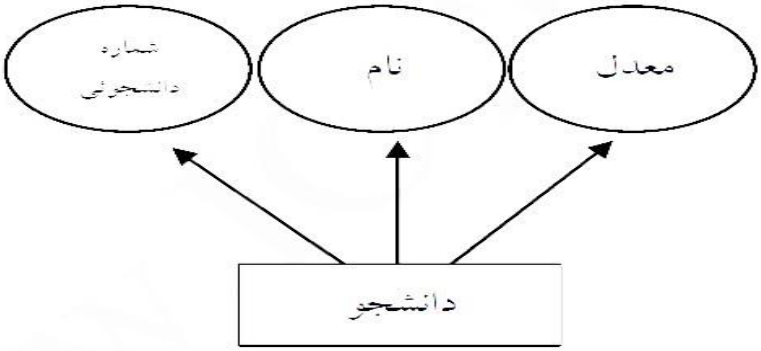
[4] Linked list
[5] Node
[6] Single linked list
[7] Doubly linked list

۳- لیست حلقه ای[8]



مثال

برای ذخیره سازی اطلاعات دانشجویان در لیست یک طرفه حلقوی و دوطرفه ساختارهای زیر تعریف می شوند : فرض کنید هر دانشجو دارای فیلدهای شماره دانشجویی ، نام و معدل می باشد :



شماره دانشجویی	نام	معدل
۱۰۰	رضا شیری	20
۱۰۱	حمید وحدت	20
۱۰۲	فرهاد کاظمیان	21

مثال: برای لیست تک پیوندی و حلقوی :

```
Struct student {
Int stno ;
Char name [40] ;
Float avr ;
Student *naxt ;
};
```

[8] Circular linked list

برای لیست دو طرفه :

```
Struct student{
Int stno ;
Char name [40] ;
Float avr ;
Student *next , *prev ;
};
```

توجه

در لیست پیوندی همیشه باید آدرس اولین گره از لیست را نگهداری نمود . چون در غیر این صورت لیست در حافظه گم می شود و قابل دستیابی نیست . معمولا آدرس اولین گره از لیست را به صورت سراسری در متغیری به نام first یا head نگهداری می کند واضح است که با داشتن آدرس اولین گره از لیست پیوندی بقیه گره ها نیز قابل دستیابی هستند به اشاره گرهایی که به صورت یک فیلد در گره های لیست پیوندی ذخیره نمی شوند ، اشاره گر خارجی می گویند . بنابراین اشاره گر first یک اشاره گر خارجی است .

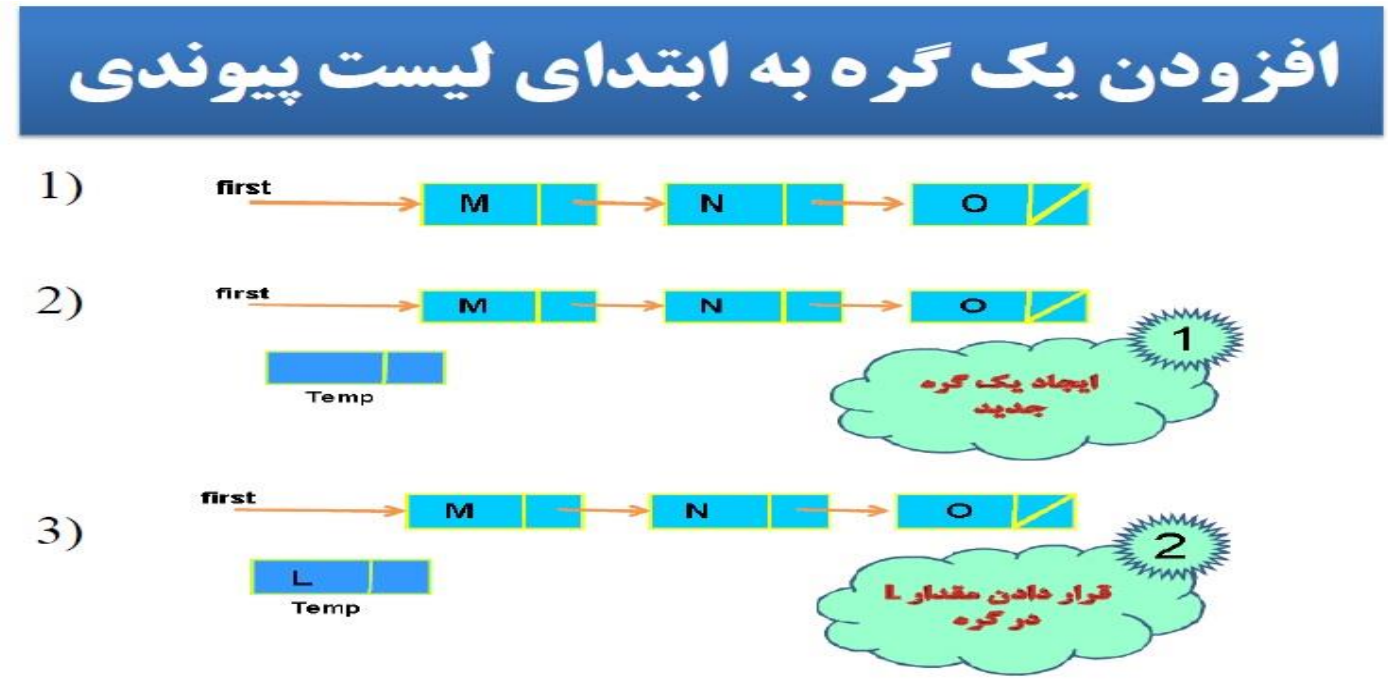
عملیات لیست پیوندی

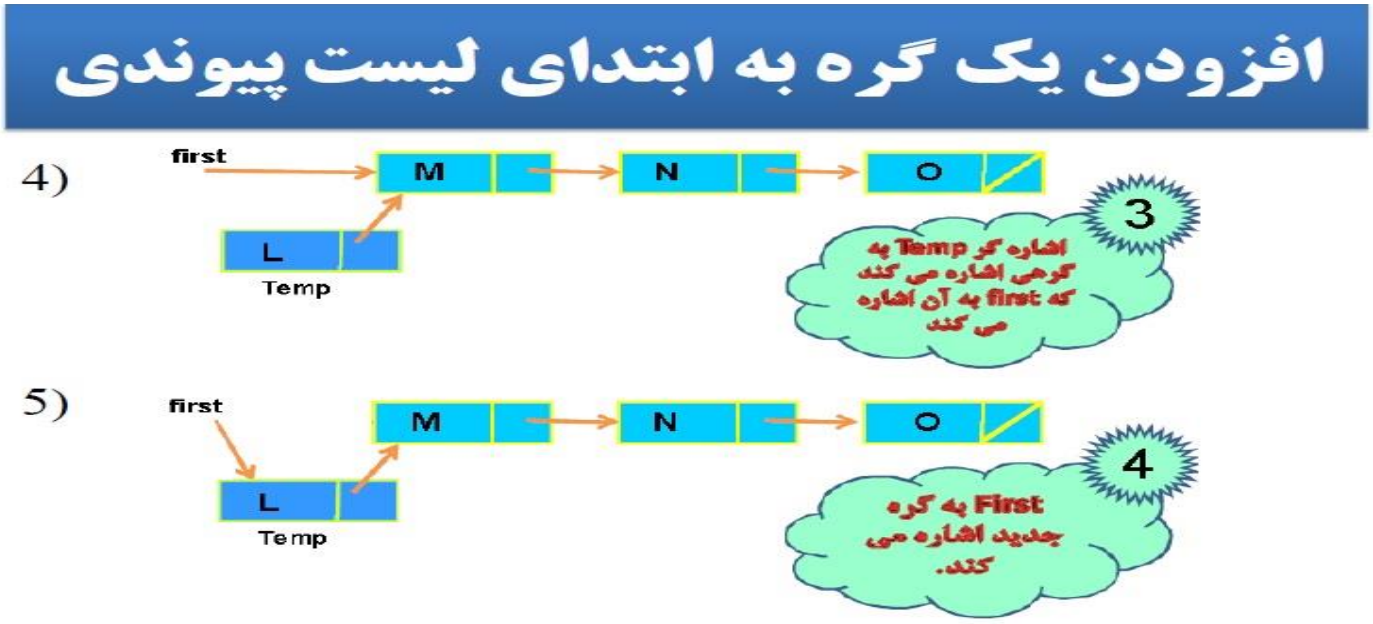
- ایجاد لیست
- درج گره در لیست
- حذف گره از لیست
- جستجو در لیست
- مرتب سازی لیست
- معکوس کردن لیست

...

درج به ابتدای لیست یک طرفه

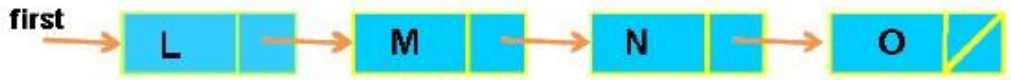
```
Struct node *temp=new struct node;
if (temp==0)
cout<<"Error Message";
else
temp->data=L;
temp->next=first;
first=temp;
```





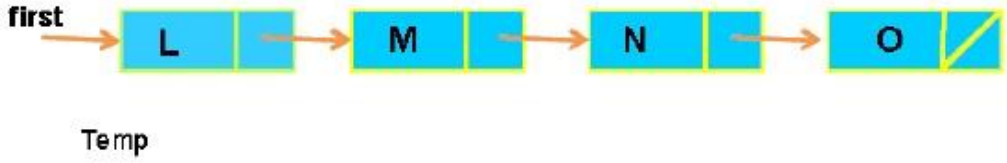
حذف یک گره از ابتدای لیست پیوندی

1)

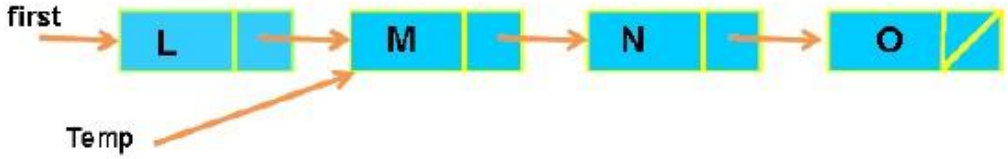


حذف با استفاده از یک متغیر کمکی انجام می شود.

2)

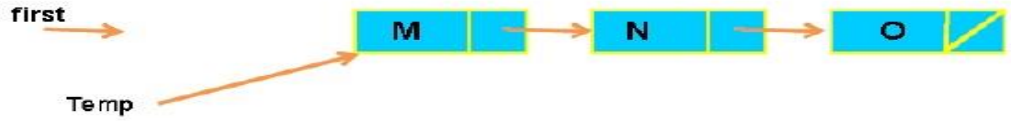


3)

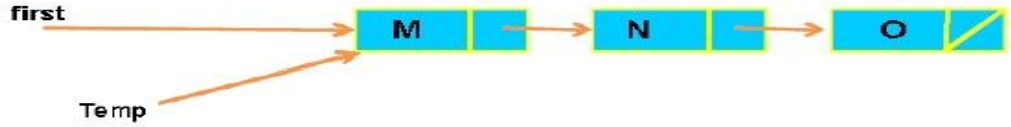


حذف یک گره از ابتدای لیست پیوندی

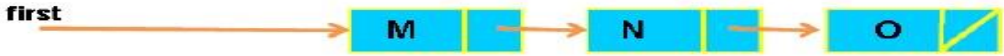
4)



5)

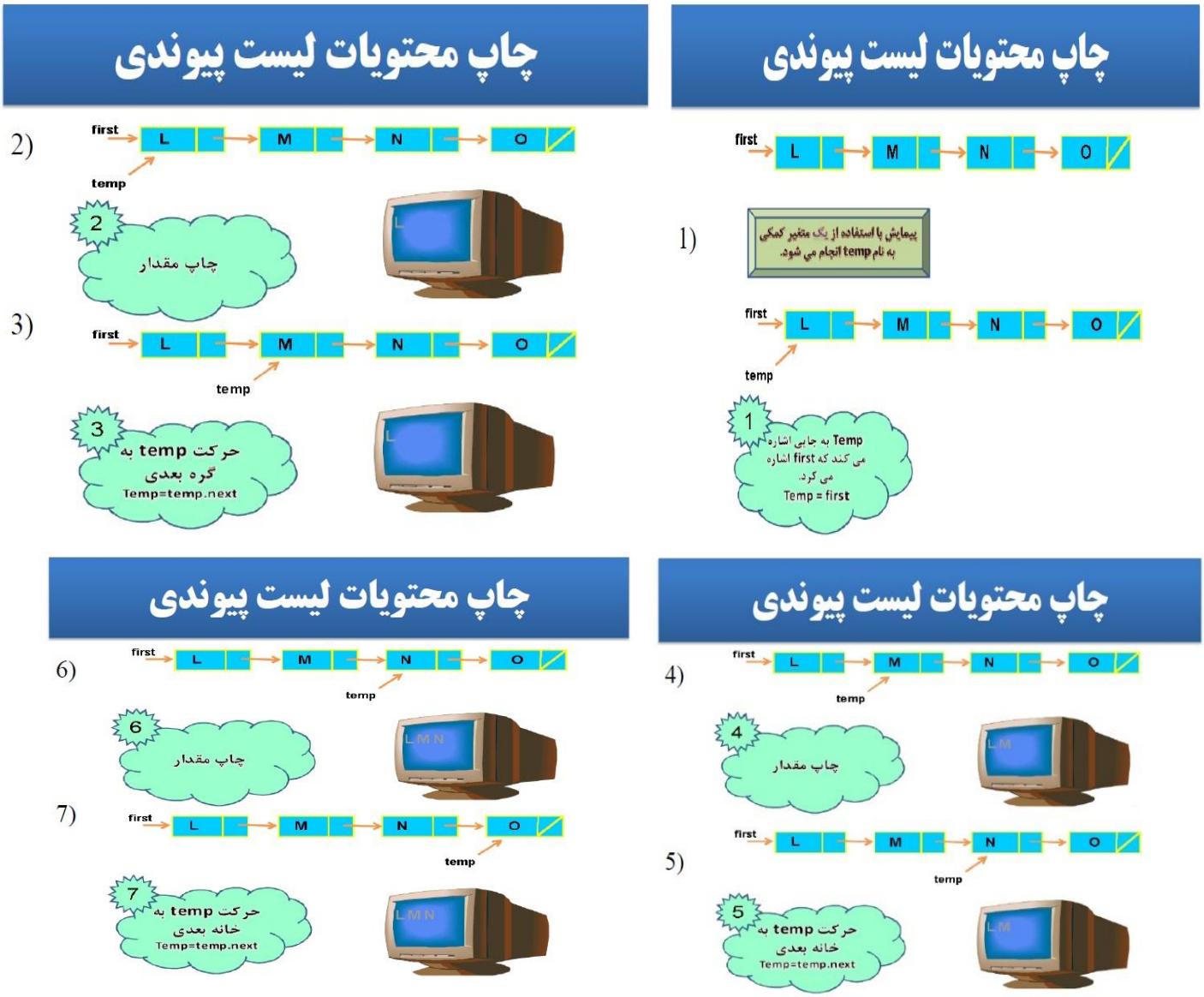


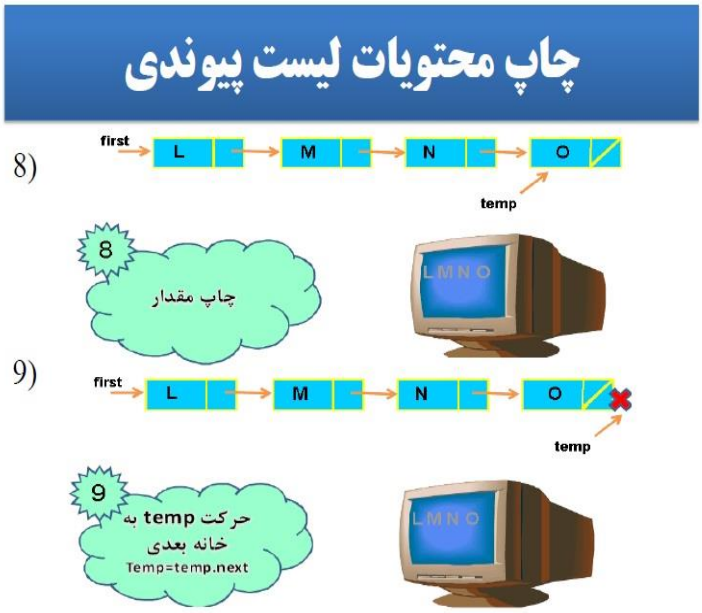
6)

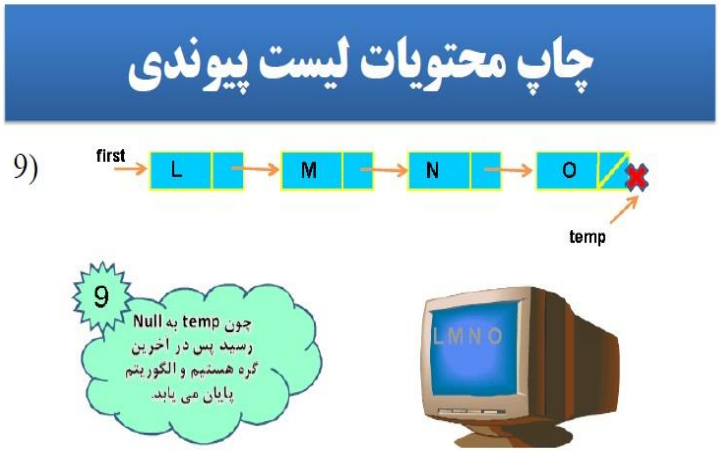


حذف یک گره از ابتدای لیست پیوندی

```
if (first==null)
cout<<"list in empty";
else
temp=first;
first=first-> next;
delete(temp);
```



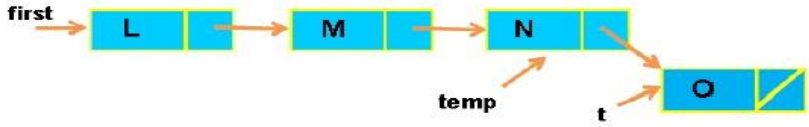





```
Temp=first;
While(temp!=null)
{
cout<<temp -> data;
temp=temp -> next;
}
```

سوال: برای جستجوی یک عنصر در لیست چه باید کرد ؟؟

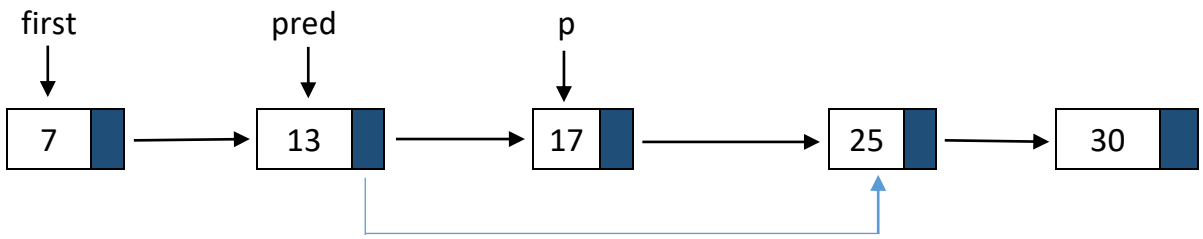
افزودن یک گره به انتهای لیست پیوندی




```
Temp=first;
if(temp==null) {
first=t;
t->next=null;}
else   {
            while(temp->next!=null)
             temp=temp->next;   }
t->next=null;
```

temp->next=t;

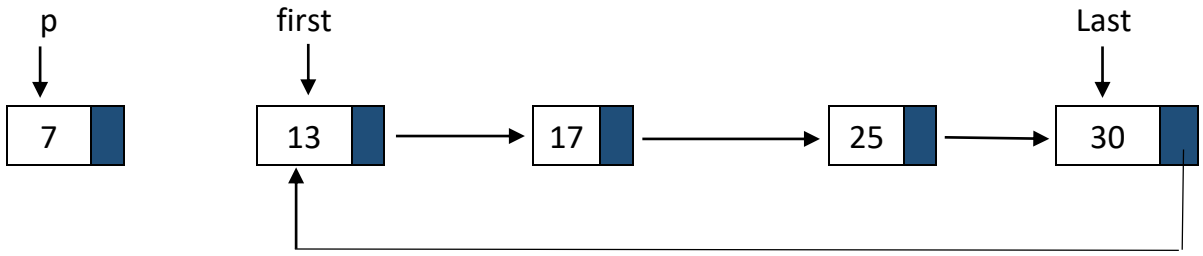
حذف از لیست پیوندی یک طرفه (عنصر بعد از pred)



1- p=pred->next
2- Pred->next = p->next
3- delete(p);

درج در ابتدای لیست پیوندی حلقوی

برای درج در لیست حلقوی نیاز است که آدرس آخرین گره لیست پیدا شود برای این کار باید لیست از ابتدا پیمایش شود

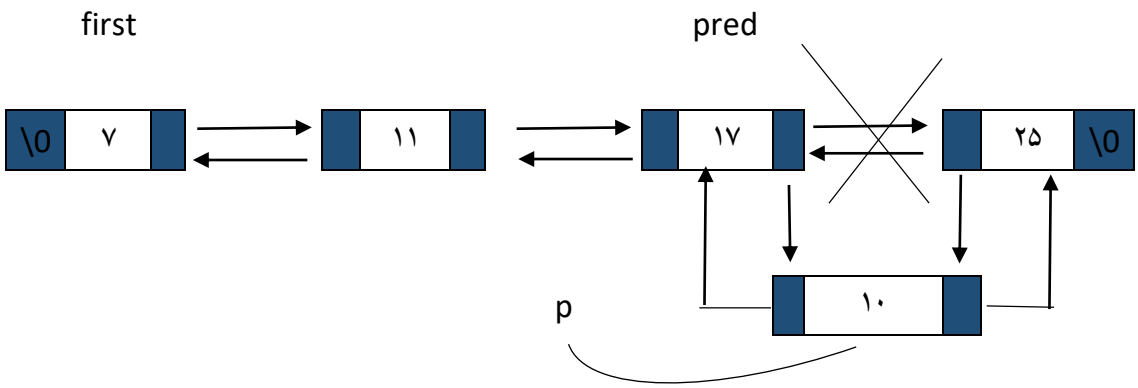


مثال : درج ۷ به ابتدای لیست

1- P->next = first
2- Last ->next =p
3- First=p

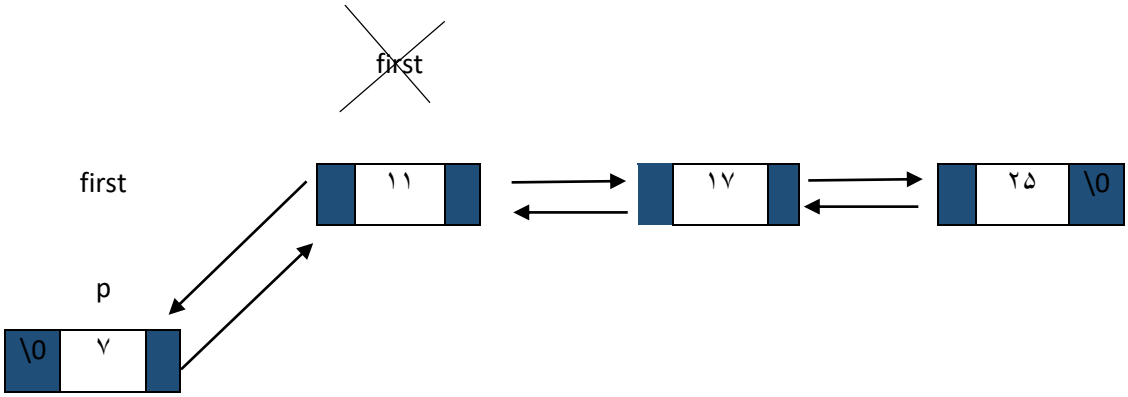
زمان درج در ابتدای لیست برابر با O(n) است

درج در لیست پیوندی دو طرفه (بعد از pred)



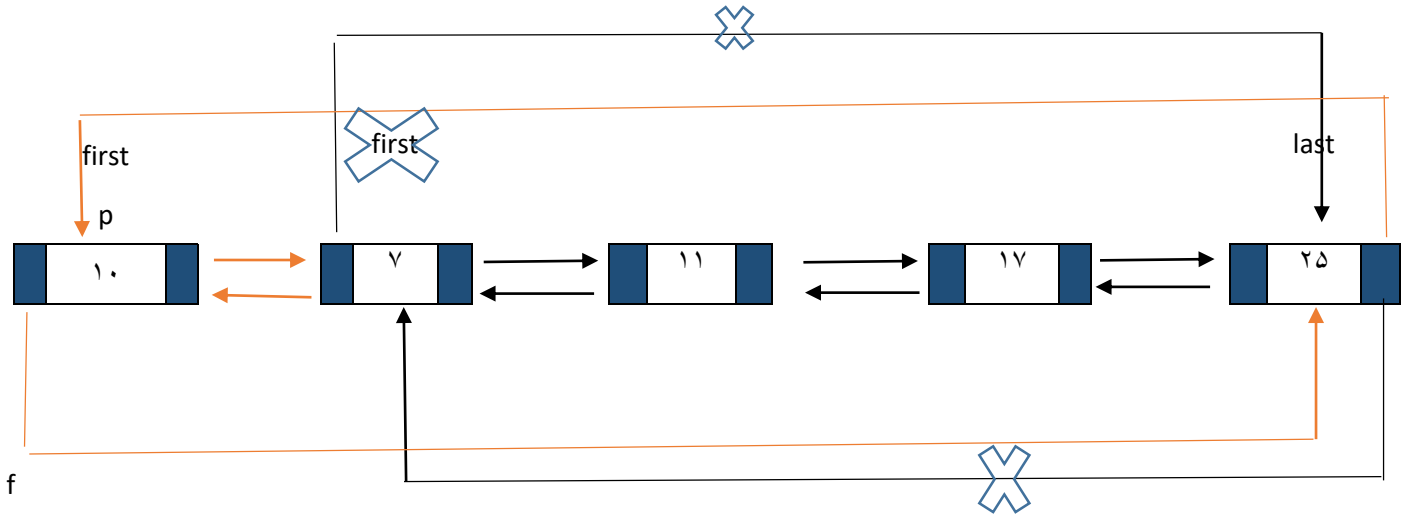
1- p->prev=pred ; p->next =pred-> next ;
2- pred->next->prev=p;
3- pred-> next =p;

سوال : برای درج در ابتدای لیست چه باید کرد



1- p->next =first;
2- first->prev =p;
3- p->prev =null;
4- first =p ;

درج در ابتدای لیست پیوندی دو طرفه حلقوی

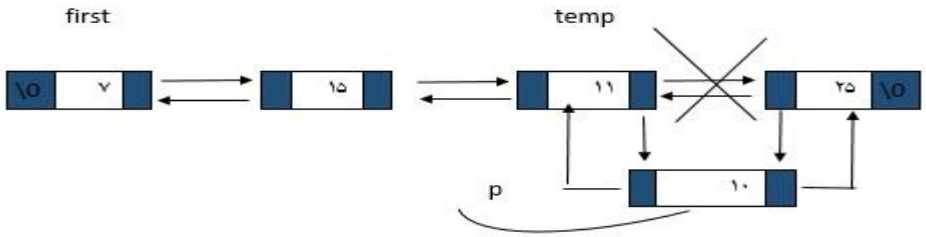


1- P->next=first;
2- P->prev = last ;
3- First->prev =p;
4- Last ->next =p;
5- First =p ;

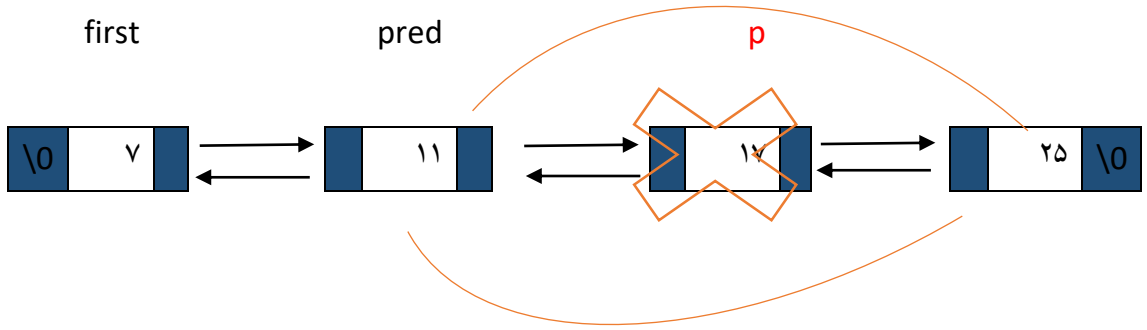
سوال درج در انتهای لیست پیوندی دو طرفه را با رسم شکل توضیح دهید ؟

سوال برای درج گره p بعد از گره ای که مقدار ۱۱ دارد در لیست پیوندی دو طرفه چه باید کرد ؟

```
Temp=first;
while(temp->data! =11)
        temp=temp->next;
p->next =temp->next;
temp ->next ->prev =p;
p->prev=temp;
temp->next=p;
```



حذف گره بعد از pred در لیست پیوندی دوطرفه



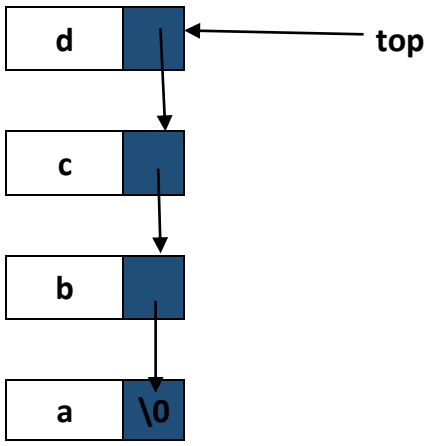

```
P=pred->next ;
Pred->next=p->next;
p->next->prev=pred;
delete(p);
```

چند نکته مهم

- سادگی عمل حذف و درج عنصر در لیست پیوندی مهمترین مزیت لیست پیوندی نسبت به آرایه است ولی بعضی از اعمال مانند جستجو یا مرتب سازی ممکن است در آرایه سریعتر از لیست پیوندی باشد.
- در لیست یک طرفه دوار امکان دسترسی به گره قبلی وجود دارد ولی در لیست یک طرفه غیردوار نمی توانیم به عنصر قبلی دسترسی داشته باشیم.
- حذف گره در لیست دوطرفه راحت تر از لیست یک طرفه است.
- روش جستجو در یک لیست پیوندی یک طرفه با n گره که داده های آن به ترتیب صعودی مرتب شده اند از جستجوی ترتیبی است.
- در لیست یک طرفه خطی نمی توان از جستجوی دودویی استفاده کرد زیرا آدرس وسط لیست را نداریم.
- جستجوی ترتیبی در لیست یک طرفه خطی از مرتبه O(n) می باشد.
- عملیات درج گره در ابتدای لیست از مرتبه O(1) است.
- عملیات درج گره در انتهای لیست از مرتبه O(1) است.
- عملیات درج گره در آرایه از مرتبه O(n) است.
- عملیات حذف گره در آرایه از مرتبه O(n) است.
- عملیات حذف گره از انتهای لیست از مرتبه O(n) است.

پشته پیوندی :شکل زیر یک پشته پیوندی را نمایش داده است

جهت اتصال در پشته پیوندی به گونه ای است که به راحتی می توان گره ای را به بالای پشته اضافه کرد و یا گره ای را از بالای پشته حذف کرد top اشاره گری است که به بالاترین گره از از پشته اشاره میکند .

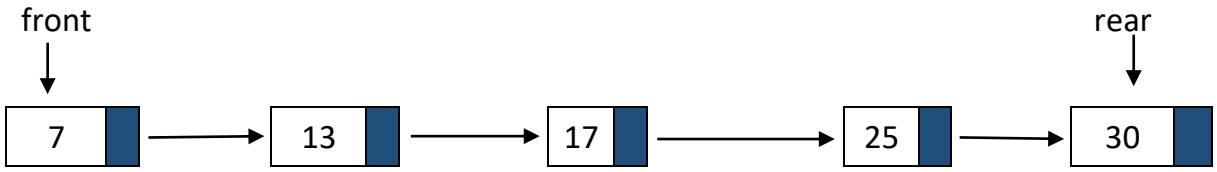



```
Void push (int x)
{     node  *p=new node;
p->data=x;
if (top==null) (top=p  ;  top->next =null;)
else (p->next = top ; top =p ;)   }
void pop (int   &x)
{
if  (top ==null )          cout << "linked stack is empty.";
     else (       node *p =top ;
                x=top->data;
                  top = top ->next; delete( p)  ;)   }
```

صف پیوندی : در شکل زیر صف پیوندی نمایش داده شده است

در این صف به راحتی میتوان گره ای را به انتهای صف اضافه و یا گره ای را از ابتدای صف حذف کرد .

اشاره گر front به گره ی ابتدایی صف و اشاره گر rear به انتهای صف اشاره میکند



شرط تهی بودن صف پیوندی این است که اشاره گر front برابر null باشد الگوریتم های درج و حذف در صف پیوندی به صورت زیر است .

```
void add (const int x)
{
        node *p =new node;
        p->data =x; p->next =null;
        if (front ==null) front = rear =p;
        else {      rear->next=p;
                     rear =p;
              }
}

void delete (int &x)
{    if (front ==null)
     cout << "linked queue is empty.";
     else   {      node *p =front;
                   X=front -> data;
                   front =front-> next;
                    delete p;
              }
 }
```