فصل چهارم: لیست ها

بخش ها

- لیست پیوندی
- لیست عمومی
- لیست حلقوی

لیست پیوندی(Linked List)

لیست: مجموعه ای خطی از اقلام داده ای

معایب آرایه ها:

- اضافه کردن و حذف عنصر در آرایه پر هزینه است
- بلوکی از حافظه را اشغال می کند و تغییر ابعاد آن ممکن نیست

راه حل:

ذخیره لیستی که هر عنصر آن شامل دو بخش؛ یکی داده و دیگری آدرس عنصر بعدی باشد

لیست پیوندی(Linked List)

لیست پیوندی:

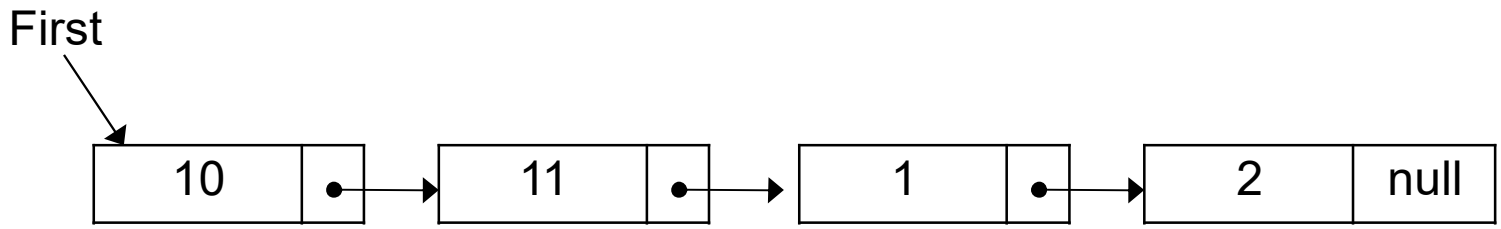
یک ساختمان داده پویاست. همانند پشته و صف از شامل مجموعه ای از عناصر است. به هر عنصر گره(Node) گفته می شود. هر گره از دو بخش اطلاعات(Info) و آدرس گره بعدی(اشاره گر)(Link) تشکیل شده است.

برای دسترسی به عناصر لیست پیوندی، از یک اشاره گر خارجی مانند First استفاده می شود که به اولین گره لیست اشاره می کند

لیست پیوندی(Linked List)

- برای دسترسی به عناصر لیست پیوندی، از یک اشاره گر خارجی مانند First استفاده می شود که به اولین گره لیست اشاره می کند.
- برای تشخیص انتهای لیست، آدرس آخرین گره لیست برابر تهی(null) می باشد.

مثال:



لیست پیوندی(Linked List)

- لیست فاقد گره را لیست خالی یا لیست تهی می گویند.
- در لیست تهی، اشاره گر First یک اشاره گر تهی است.
- برای به دست آوردن لیست تهی کافی است first=null قرار داده شود.

تعریف یک گره:

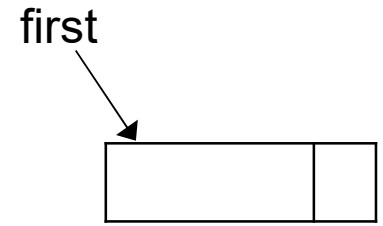
```
struct node
{
    int data;
    node *next;
};
```

لیست پیوندی(Linked List)

به دست آوردن یک گره جدید:

- برای تعریف اشاره گر First، اشاره گری از نوع گره لیست تعریف می کنیم
  node *first;
- سپس حافظه ای به اندازه ساختار node تخصیص داده می شود:

first = (node *) malloc(sizeof(node));



لیست پیوندی(Linked List)

مراجعه به گره های لیست:

- اگر first، اشاره گری به گره ای از لیست باشد، برای خواندن بخش آدرس first->next و برای خواند بخش داده first->info استفاده می شود.
- اگر first، اشاره گری به ابتدای لیست باشد و p نیز بخواهد به گره اول لیست اشاره کند می نویسیم: p=first
- برای بازگرداندن حافظه تخصیصی از تابع free() استفاده می شود. مثلا با دستور free(p) حافظه ای را که p به آن اشاره می کند، به مخزن حافظه برمی گرداند.

لیست پیوندی(Linked List)

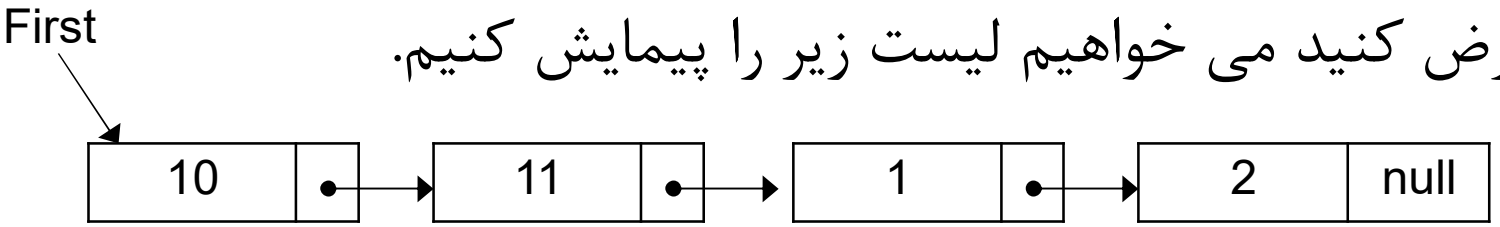
پیمایش لیست:

- منظور از پیمایش دستیابی به تمام عناصر لیست و در صورت لزوم، پردازش آنها است.
- برای این کار باید به غیر از اشاره گر first، اشاره گر دیگری مانند p را با عمل p=first تعریف کنیم تا آن نیز به اول لیست اشاره کند.
- استفاده از خود first برای پیمایش باعث می شود ابتدای لیست پیوندی را از دست بدهیم.

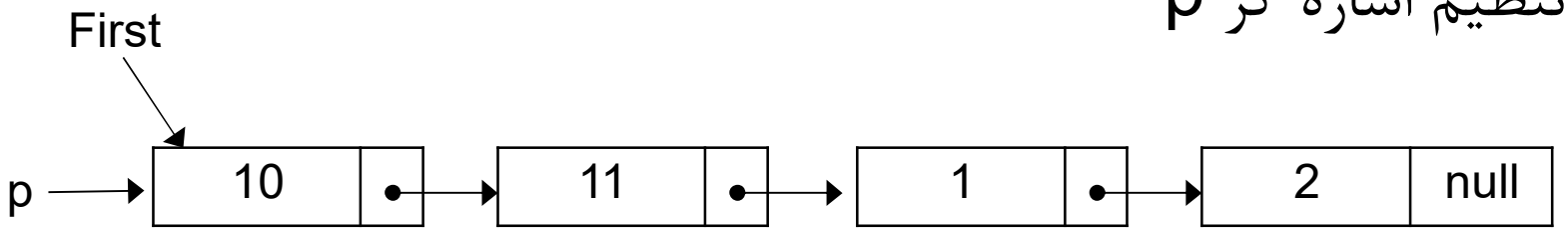
لیست پیوندی(Linked List)

مثال:

- فرض کنید می خواهیم لیست زیر را پیمایش کنیم.



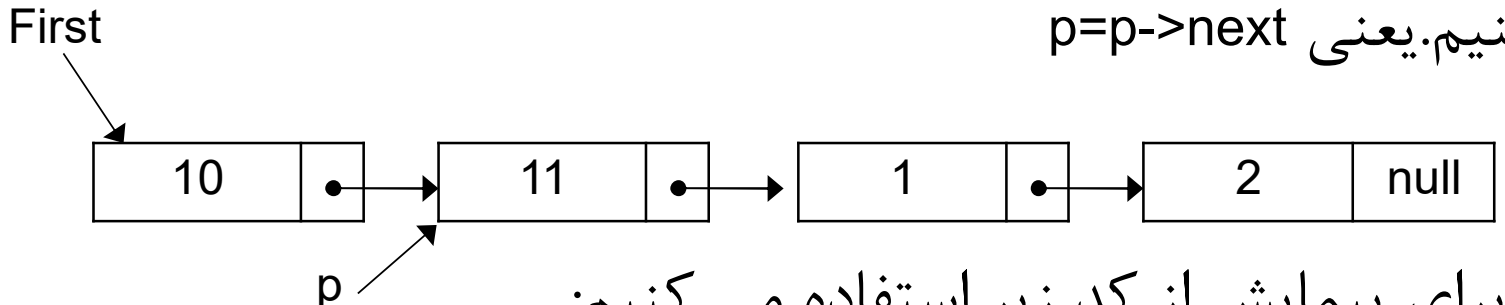
- تنظیم اشاره گر p



لیست پیوندی(Linked List)

مثال:

- برای دستیابی به گره بعدی با محتوای ۱۱ باید بخش آدرس را دنبال کنیم.یعنی p=p->next



- برای پیمایش از کد زیر استفاده می کنیم:

```
temp=first;
while(temp!=null){
        /////////  پردازش بخش داده(temp->info  /////////
         temp=temp->next;
}
```

لیست پیوندی(Linked List)

جستجو در لیست پیوندی:

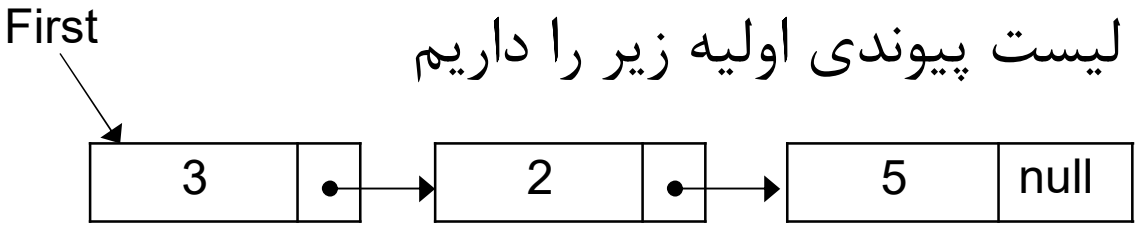
- مشابه پیمایش است با این تفاوت که در بخش پردازش باید مقدار مورد جستجو را با داده فعلی مقایسه کنیم

```
int i=0;
bool flgFound=false;
temp=first;
while(temp!=NULL&&!flgFound){
                    i++;
                    if(temp->info==s){
                              flgFound=true;
                              }
                    temp=temp->next;
          }
```

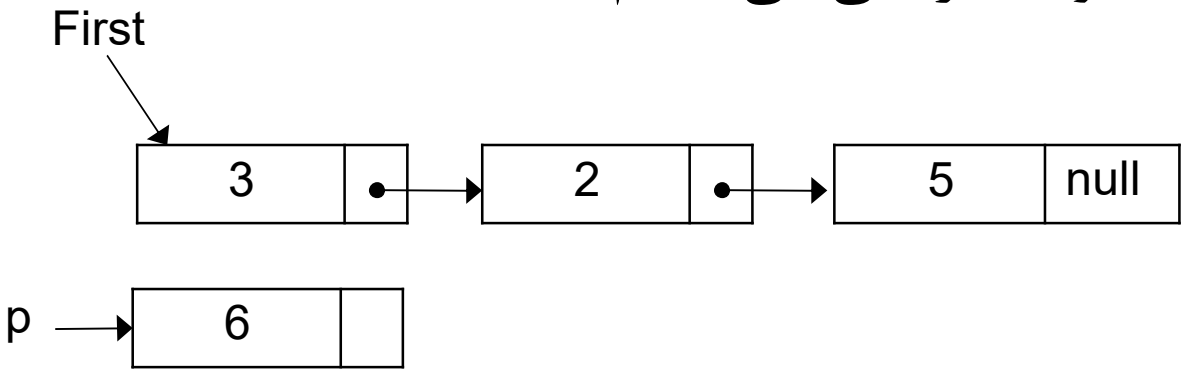
درج و حذف گره از لیست پیوندی

درج گره در ابتدای لیست:

- فرض می کنیم لیست پیوندی اولیه زیر را داریم



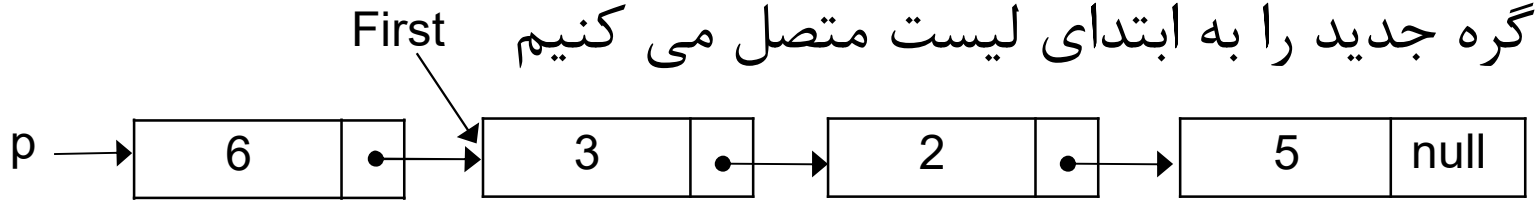
- گره جدید را ایجاد و مقداردهی می کنیم



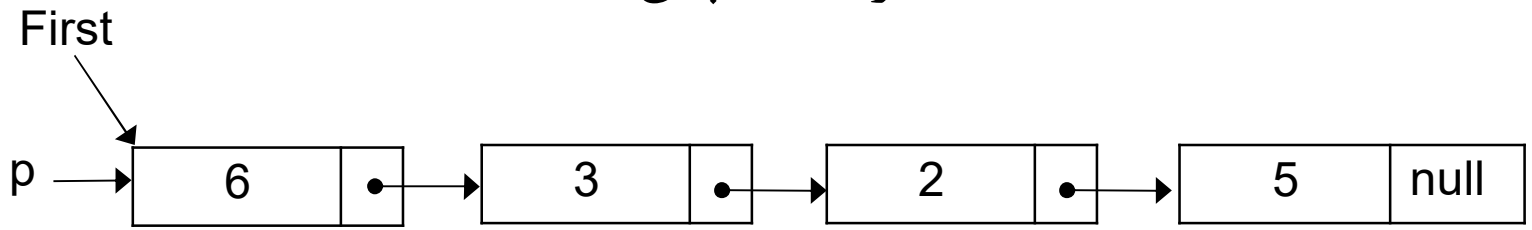
درج و حذف گره از لیست پیوندی

درج گره در ابتدای لیست:

- گره جدید را به ابتدای لیست متصل می کنیم



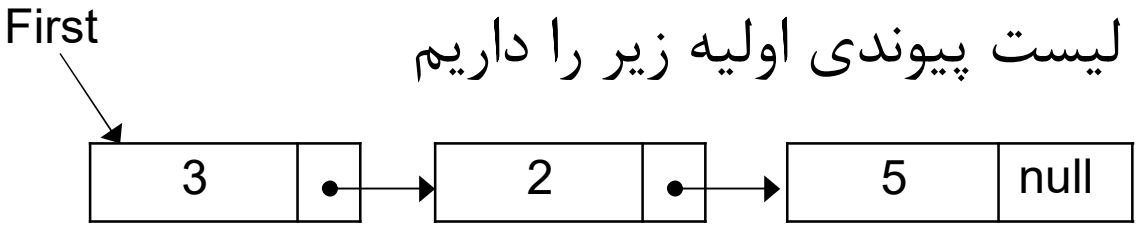
- first باید به ابتدای لیست اشاره کند پس first=p



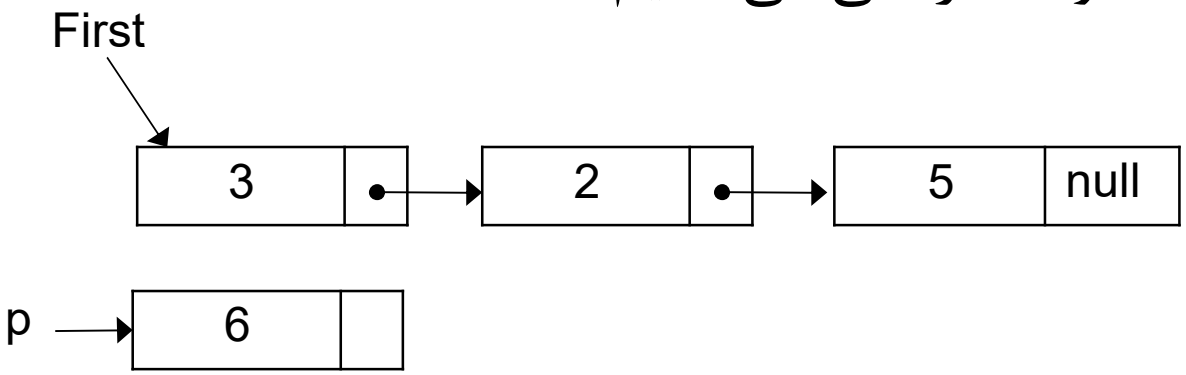
درج و حذف گره از لیست پیوندی

درج گره در مکان مشخصی در لیست:

- فرض می کنیم لیست پیوندی اولیه زیر را داریم



- گره جدید را ایجاد و مقداردهی می کنیم

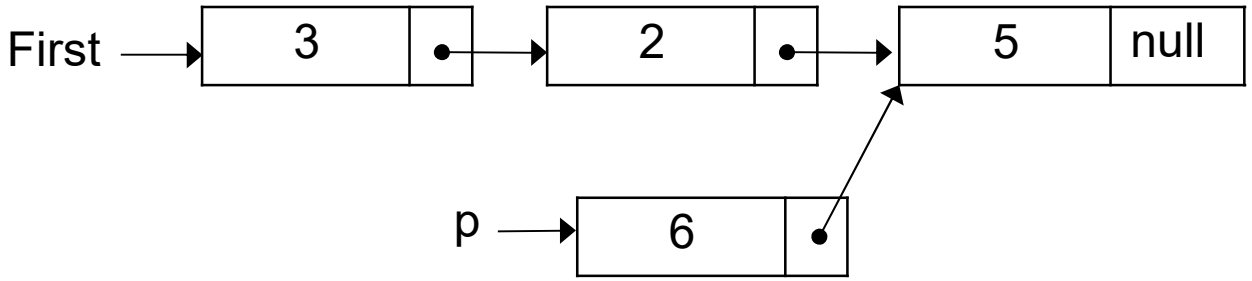


درج و حذف گره از لیست پیوندی

درج گره در مکان مشخصی در لیست:

- از عنصر ابتدایی لیست را تا محل تعیین شده پیمایش می کنیم مثلا بعد از گره دوم با داده ۲(گره جاری در cur قرار می گیرد)

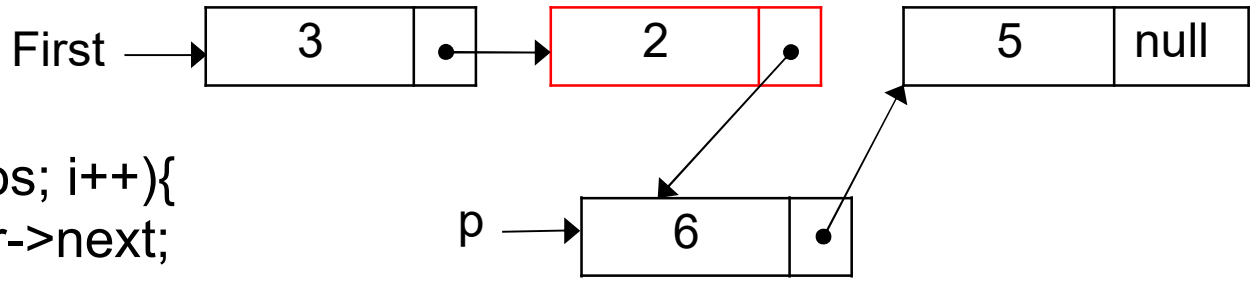
- p باید به گره بعد از محل درج اشاره کند پس p->next=cur->next



درج و حذف گره از لیست پیوندی

درج گره در مکان مشخصی در لیست:

- گره جاری باید به p اشاره کند.

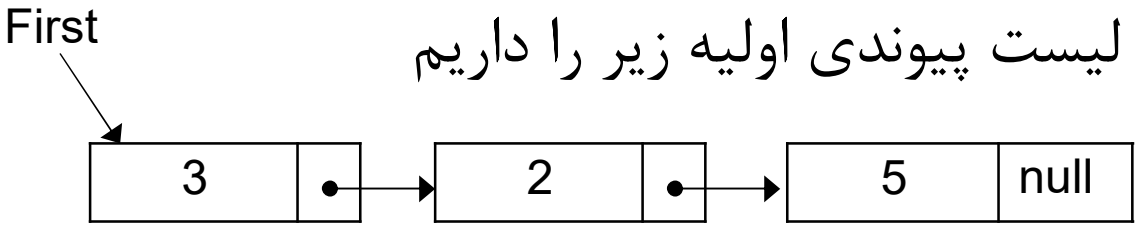



```
for(int i=1; i<pos; i++){
        cur=cur->next;
  }
p->next=cur->next;
cur->next=p;
 }
```

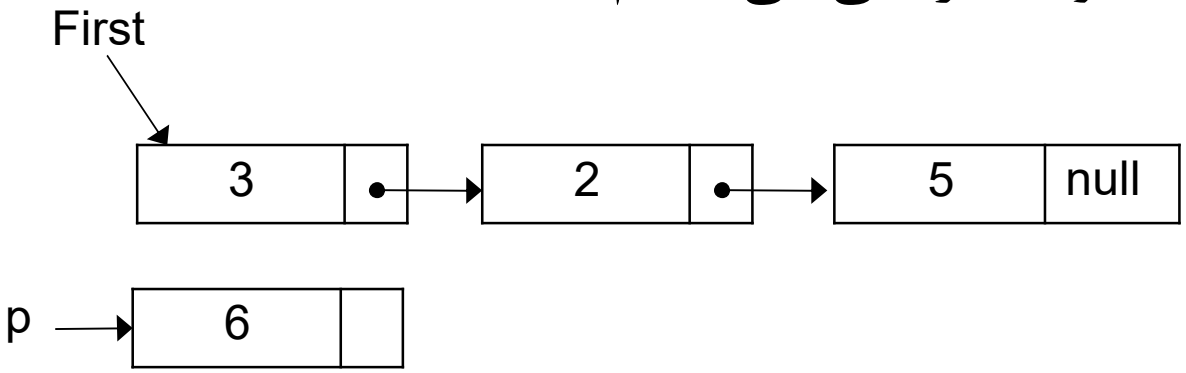
درج و حذف گره از لیست پیوندی

درج گره در انتهای لیست:

- فرض می کنیم لیست پیوندی اولیه زیر را داریم



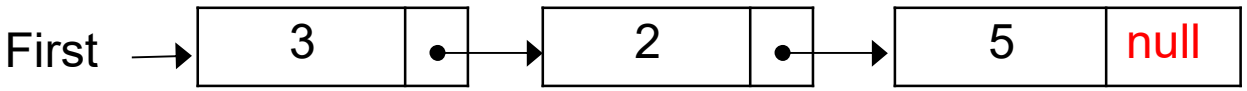
- گره جدید را ایجاد و مقداردهی می کنیم



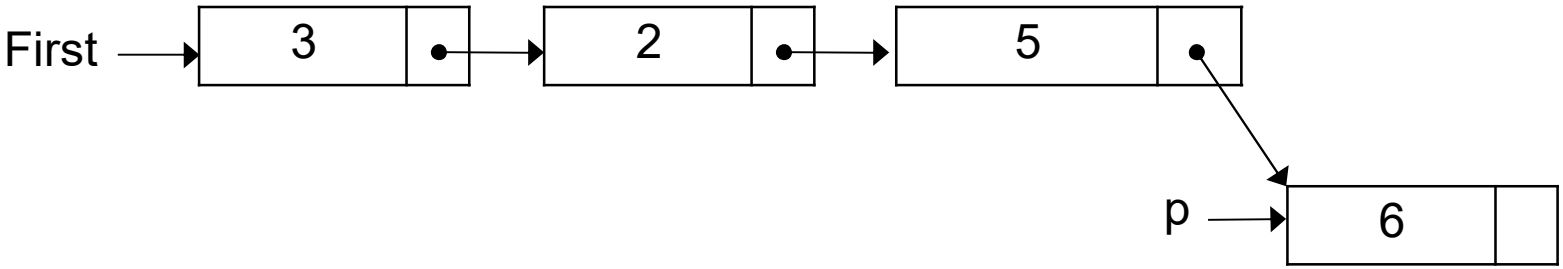
درج و حذف گره از لیست پیوندی

درج گره در انتهای لیست:

- از عنصر ابتدایی لیست را تا انتها را پیمایش می کنیم



- گره آخر باید به p اشاره کند پس cur->next=p



درج و حذف گره از لیست پیوندی

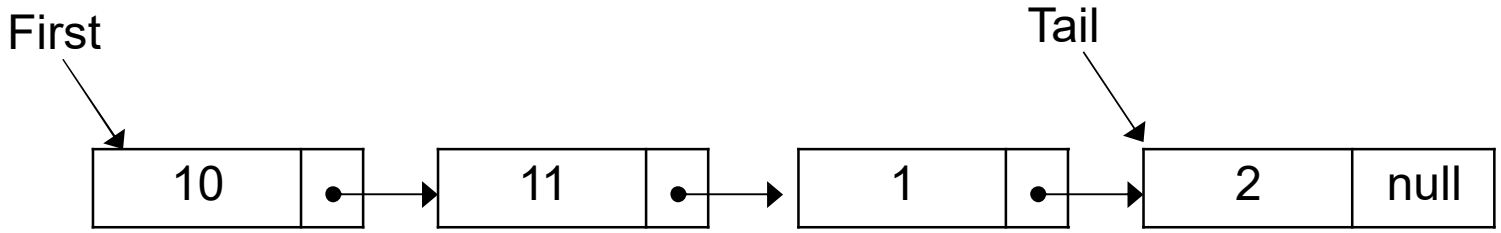
درج گره در انتهای لیست:

- گره جاری باید به p اشاره کند.

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

درج و حذف گره از لیست پیوندی

- گاهی برای سادگی، اشاره گر دیگری به انتهای لیست در نظر گرفته می شود یعنی:

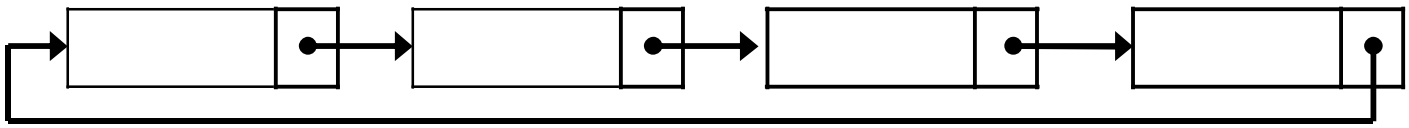


- در این صورت عملیاتی که نیاز به رسیدن به انتهای لیست را دارند سریع تر و ساده تر انجام می شوند.
- مثلا برای افزودن گره در انتهای لیست داریم:

```
Tail->next=p;
```

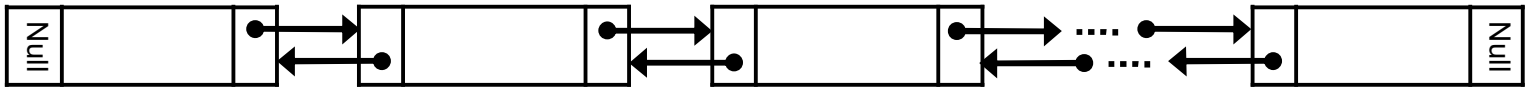
لیست پیوندی چرخشی

- مشابه لیست پیوندی خطی است با این تفاوت که اشاره گر آخرین گره به جای NULL به ابتدای لیست اشاره می کند.
- در لیست خطی همواره باید آدرس ابتدای لیست را داشته باشیم ولی در لیست چرخشی با داشتن آدرس هر گره دلخواه می توان به همه گره ها دسترسی داشت.
- کلیه الگوریتم های لیست چرخشی و خطی مشابه یکدیگر هستند، فقط شرط پایان حلقه تکرار و نحوه اصلاح اشاره گر گره پایانی آنها با یکدیگر تفاوت دارد.



لیست پیوندی دوطرفه

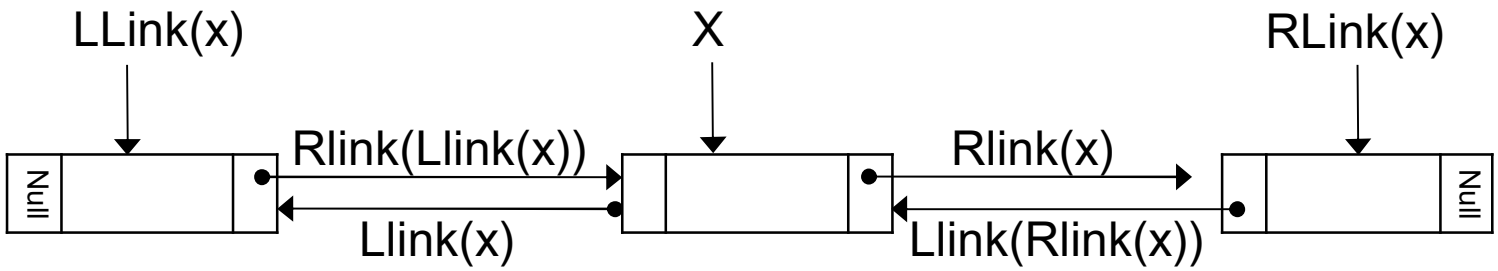
- در لیست پیوندی دو طرفه در هر گره دو اشاره گر وجود دارد که یکی به گره بعدی و دیگری به گره قبلی در لیست اشاره می کند.
- به کمک اشاره گرهای سمت راست (RLink) و سمت چپ (LLink)می توان در هر دو طرف لیست حرکت کرد.
- بنابراین با داشتن آدرس یک گره، می توان به همه گره ها دسترسی داشت.
- این لیست را به صورت چرخشی نیز می توان پیاده سازی کرد



لیست پیوندی دوطرفه

- در این لیست رابطه زیر همواره برقرار است:

$$Rlink(Llink(x))=Llink(Rlink(x))=x$$



لیست پیوندی دوطرفه

مهمترین اعمال لیست پیوندی دو طرفه:

- الگوریتم حذف گره x :

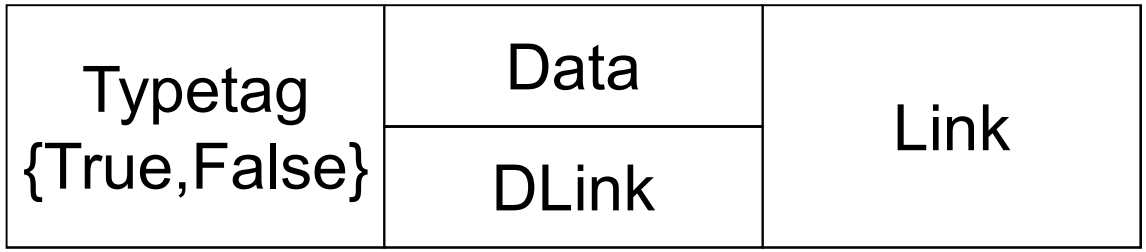
Rlink(Llink(x)) = Rlink(x)
Llink(Rlink(x)) = Llink(x)

- الگوریتم افزودن گره x به سمت راستy:

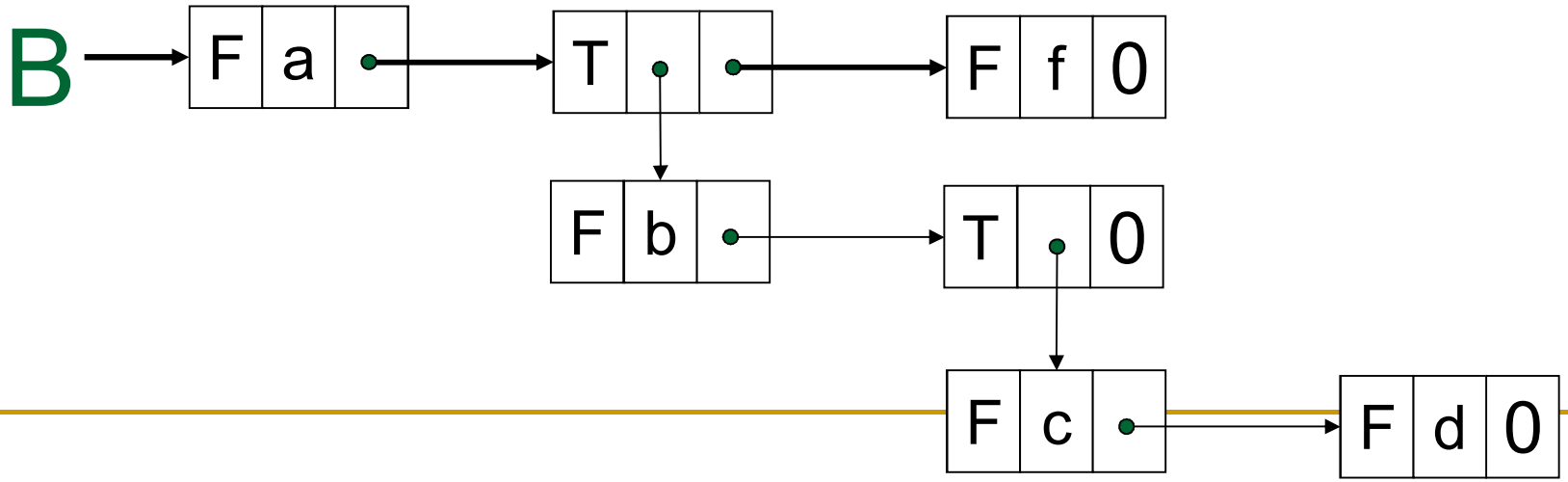
Llink(Rlink(y)) = x
Rlink(x) = Rlink(y)
Rlink(y) = x
Llink(x) = y

لیست های عمومی (General List)

- لیستی که هر گره آن بتواند خود یک زیرلیست باشد



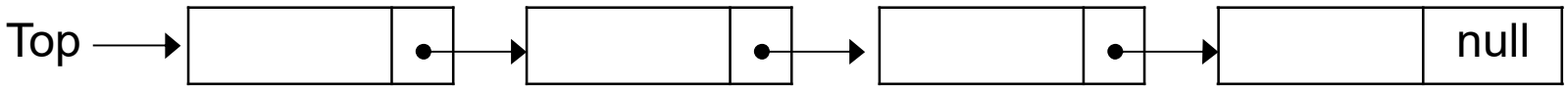
- مثال: B=(a,(b,(c,d)),f)



کاربرد لیست پیوندی

پیاده سازی پشته با لیست پیوندی:

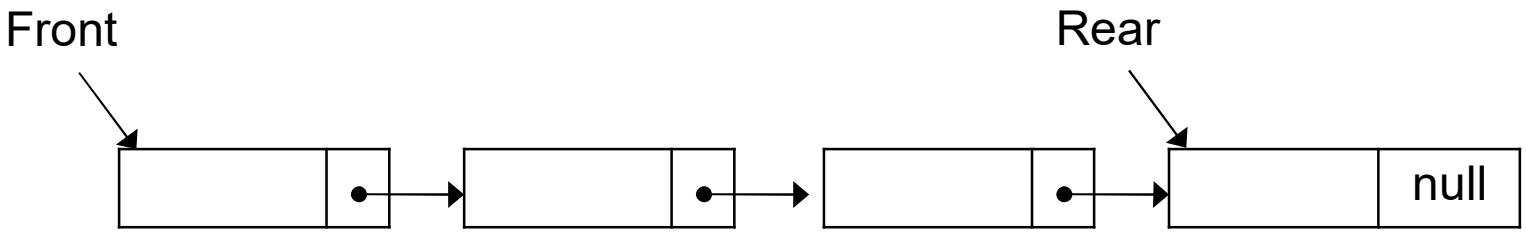
- یکی از معایب پیاده سازی پشته و صف با استفاده از آرایه این است که اندازه ثابت آرایه، اندازه پشته و صف را محدود می کند. استفاده از لیست پیوندی محدودیت رشد را از بین می برد و بدون هدر دادن حافظه اندازه آن کوچک می شود.
- با توجه به ساختار پشته، عملیات تنها از سمت بالای پشته انجام می شود. بنابراین افزودن عنصر به ابتدای لیست و حذف نیز از همین سمت انجام می شود.



کاربرد لیست پیوندی

پیاده سازی صف با لیست پیوندی:

- مشابه پشته است تنها به دو اشاره گر ابتدا و انتهای لیست نیاز دارد.
- با توجه به ساختار صف، افزودن عنصر به انتهای لیست و حذف از ابتدای آن انجام می شود.



کاربرد لیست پیوندی

معایب پیاده سازی پشته و صف با لیست پیوندی:

- چون علاوه بر داده باید آذرس نیز ذخیره شود، در مقایسه با آرایه حافظه بیشتری اشغال می کند
- مدیریت عناصر مستلزم صرف زمان بیشتری است. هر عمل افزودن و حذف نیاز به اجرای الگوریتم های عنوان شده دارد.