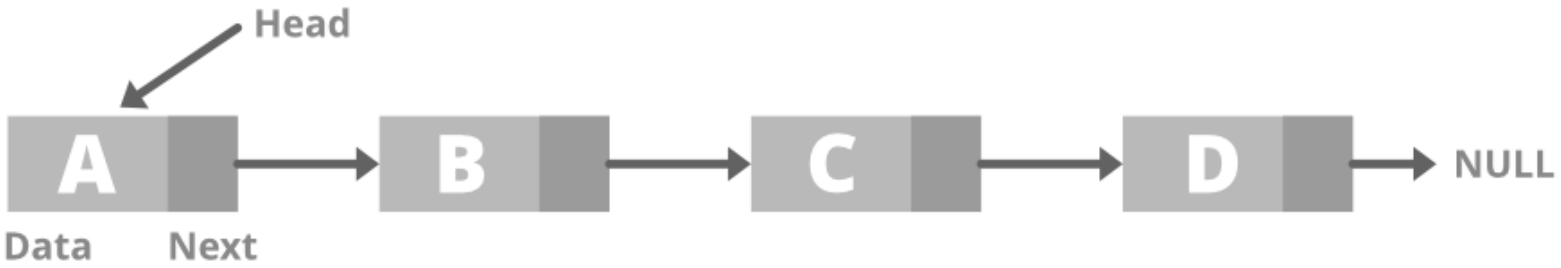
ساختمان‌های داده

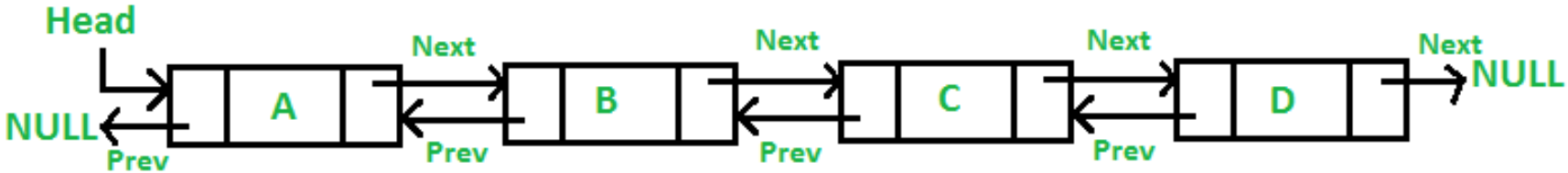
Data Structures

لیست های پیوندی (یکطرفه و دو طرفه)

Singly Linked Lists

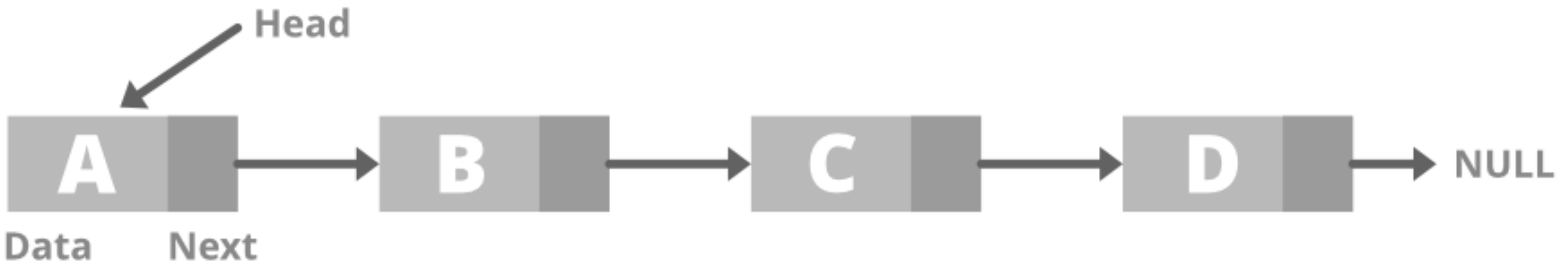
Doubly Linked Lists





لیست های پیوندی (یکطرفه)

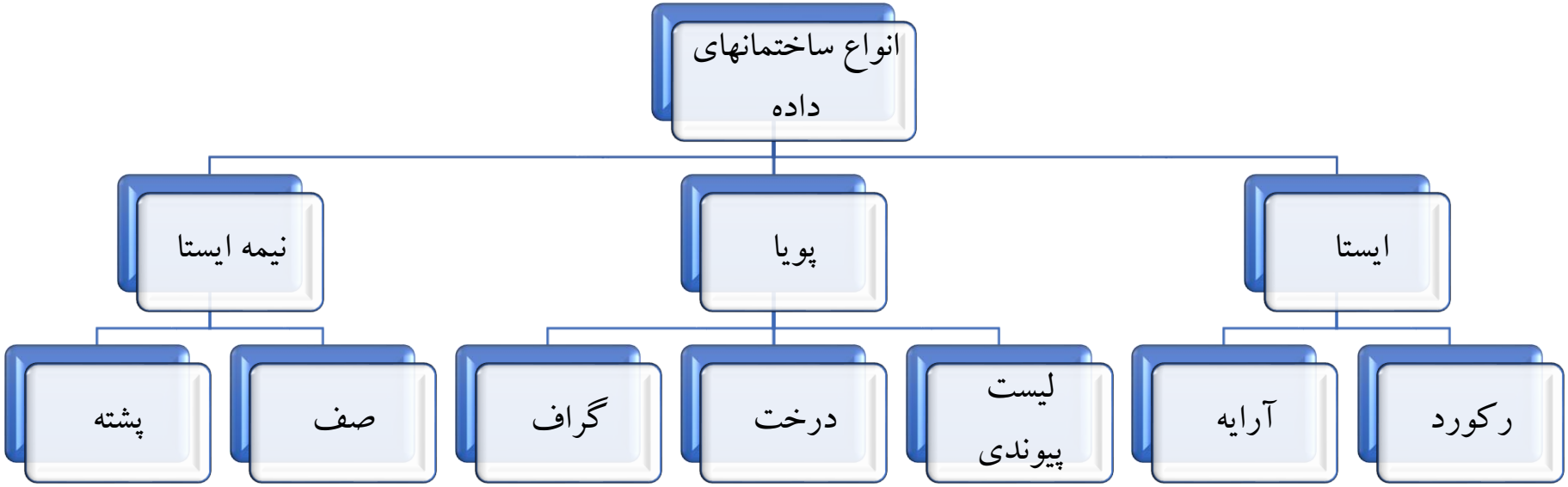
Singly Linked Lists



مفهوم لیست پیوندی

- غالبا برای ذخیره داده ها هم می توان از آرایه استفاده کرد و هم از لیست پیوندی.
- عناصر آرایه الزاما در حافظه پشت سرهم قرار گرفته اند ولی عناصر لیست پیوندی از نوع پویا بوده و عناصر آن الزاما در کنار یکدیگر نمی باشند
  - همین دلیل اعمال درج و حذف در لیست پیوندی ساده تر و خیلی سریعتر از آرایه انجام می گیرد.
  - بعضی از اعمال مثلا جستجو یا مرتب سازی ممکن است در آرایه سریعتر از لیست پیوندی باشد.
- هر عنصر یا گرهNode در لیست پیوندی حداقل از دو فیلد داده data و اشاره گری به گره بعدی Link تشکیل یافته است.

انواع ساختمان داده ها
(ساختارهای داده ای)



ساختارهای داده

❑مشکلات ساختارهای ایستا

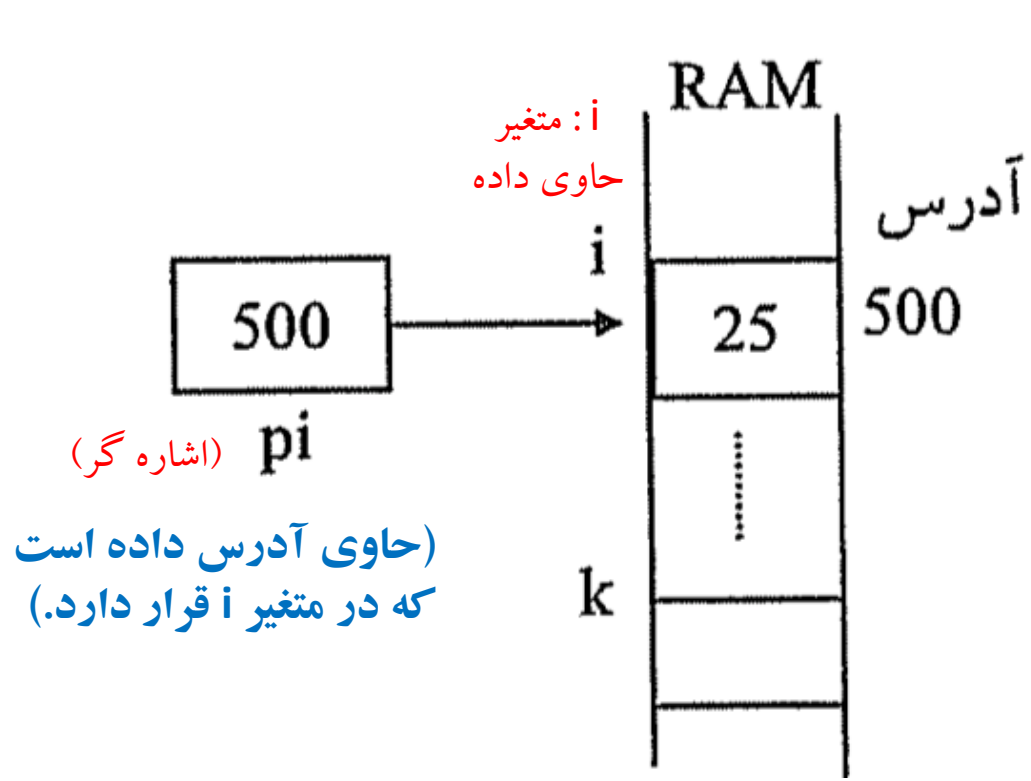
- اتلاف حافظه : تمام حافظه در ابتدا اختصاص پیدا می نماید.
- ذخیره سازی عناصر در حافظه بصورت پشت سرهم می باشد.

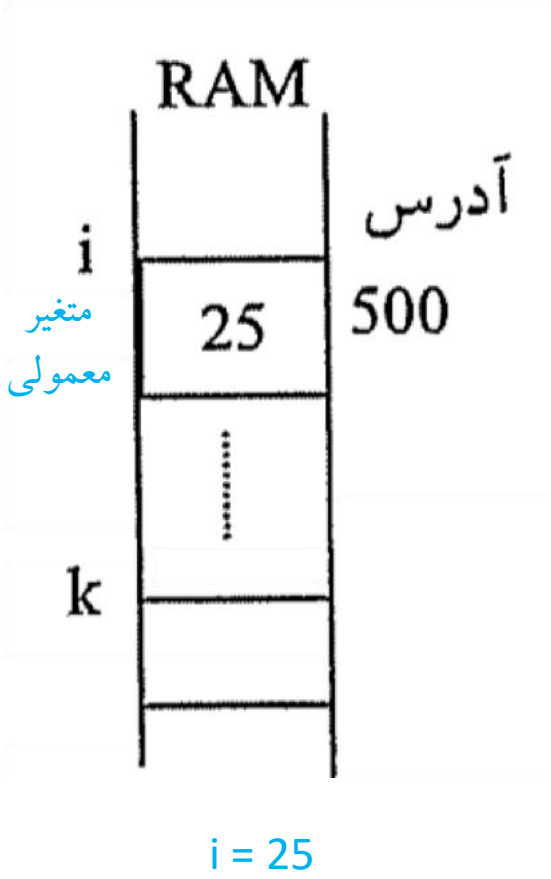
❑ساختارهای پویا

- هر جا نیاز باشد، حافظه اختصاص می یابد و اگر نیاز نبود به حافظه آزاد برگردانده می شود.
- از ساختاری به نام اشاره گر استفاده می شود.

اشاره گر ها

❑اشاره گر متغیری است که در آن آدرس یک خانه حافظه (داده ای دیگر) است.

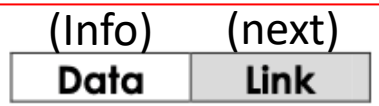




لیست پیوندی

- به لیستی از عناصر که به کمک اشاره گر به هم مرتبط شده اند، لیست پیوندی گفته می شود.
- هر نود (گره) شامل دو بخش است:
  - داده (Data or Info)
  - لینک (اشاره به عنصر بعدی)

$$node = \begin{cases} Data \\ Link \end{cases}$$



در قسمت link آدرس گره ای که منطقا بعد از گره فعلی قرار دارد ، باید مشخص شود.

- برای پیاده سازی به رکورد یا ساختار داریم :

C:

```
Struct Node{
        int info;          داده (موجود در این نود)
        Node *next;        لینک (به داده بعدی)
}
```

پاسکال:

```
Type
        nptr= ^node;
        node= Record
                info: integer;
                next: nptr;
        end;
```

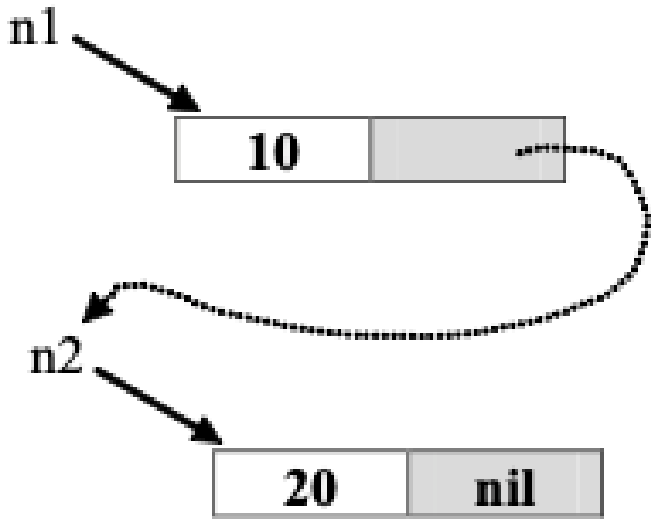
لیست پیوندی

❑مثال : ایجاد یک لیست پیوندی ساده

- به اشاره گر n1 که جزء گره ها داخلی نیست، اشاره گر خارجی (سر لیست) گفته می شود.
- از طریق آن به کل مجموعه دسترسی داریم.

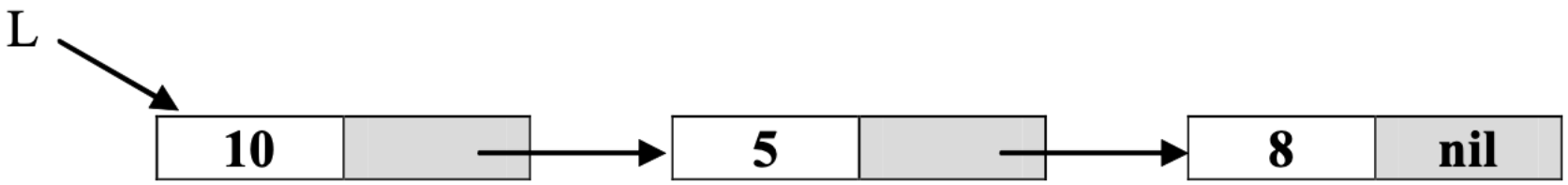
```
Var n1,n2: nptr;

new(n1);
n1^.info :=10 ;
new(n2);
n2^.info :=20 ;
n1^.next :=n2;
n2^.next :=nil ;
```



لیست پیوندی

- دسترسی به عناصر لیست پیوندی و پیمایش



داده گره اول: writeln(L^.info);

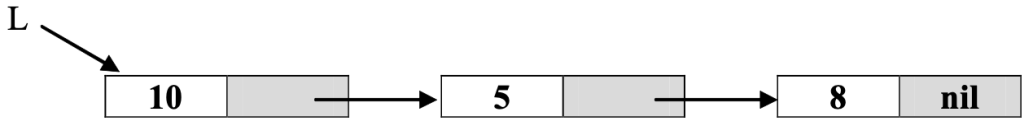
داده گره دوم: writeln(L^.next^.info);

اشاره گر برای پیمایش بین گره ها: P:=L;

P:=P^.next;

لیست پیوندی

❑مثال : تابعی بنویسید که با دریافت لینک خارجی به لیست، مجموع عناصر لیست را حساب کند.

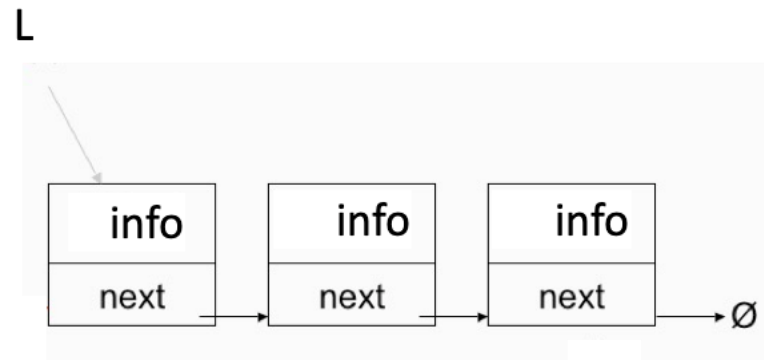


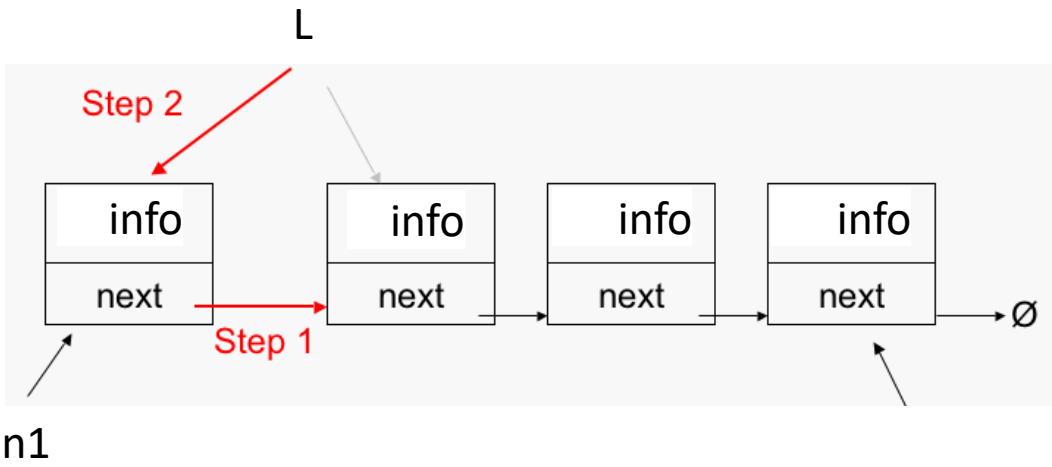

```
function sum(L:nptr):integer;
var
        s:integer;
        p:nptr;
begin
        s:=0; p:=L;
        while p<>nil do
        begin
                s:=s + p^.info;
                p:= p^.next;
        end
        writeln(s)
end;
```

لیست پیوندی

- اضافه نمودن نودی به ابتدای لیست

```
Var
        n1:nptr;
begin
        new(n1);
        n1^.info:=x;
        n1^.next:=L;
        L:=n1;
end
```

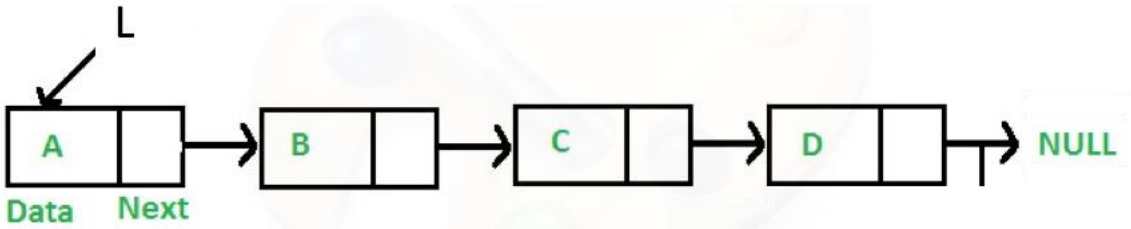


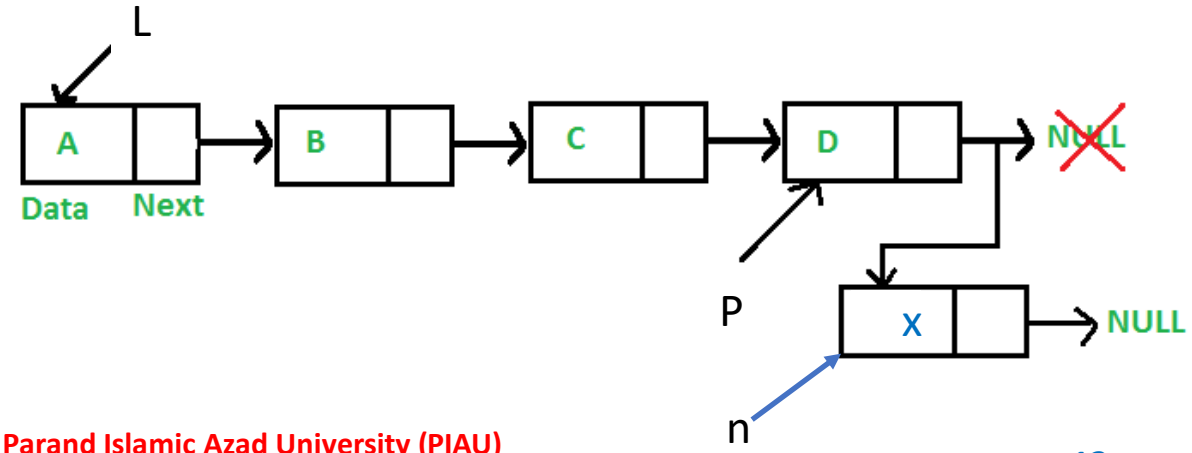


لیست پیوندی

- اضافه نمودن نودی به انتهای لیست

```
P=L;
while P^.next<>nil do                    //اشاره گر به گره آخر
        P:=P^.next;
new(n);
n^.info:=x;
n^.next=nil;
P^.next:=n;
```

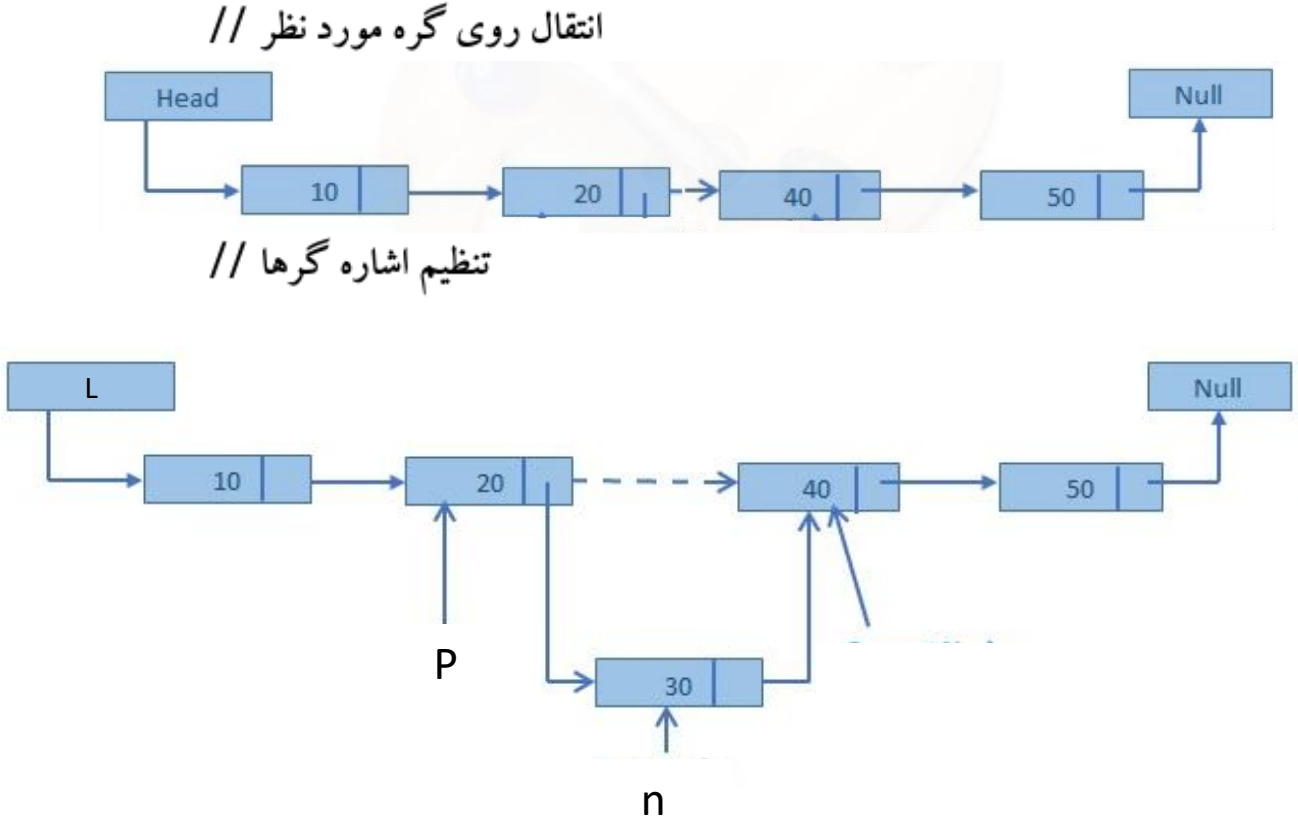




لیست پیوندی

- اضافه نمودن نودی بعد از نود nام در لیست

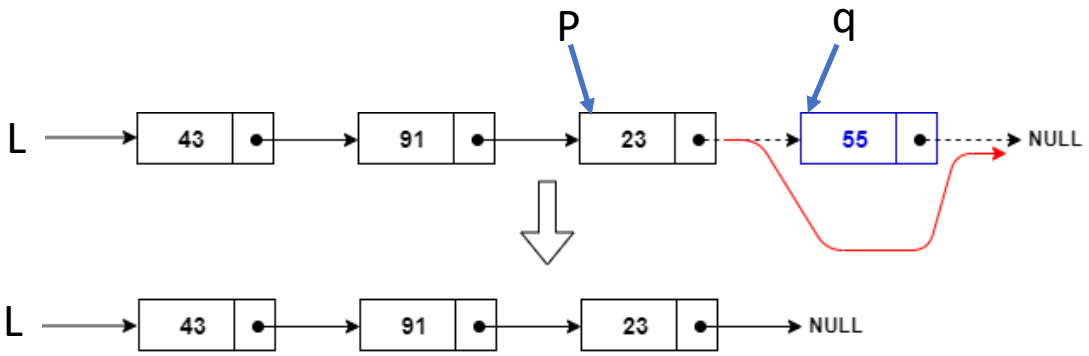
```
P:=L;
for i:=1 to n-1 do
        P:=P^.next;
new(n);
n^.info:=X;
n^.next :=P^.next
P^.next:=n;
```



لیست پیوندی

- حذف گره انتهایی از لیست L و برگرداندن مقدار آن

```
function RE(var L:nptr):integer;
var
        P:nptr;
        S:integer;
begin
        P:=L;
        While P^.next^.next<>nil do
                P:=P^.next;
        q:=P^.next;
        P^.next=nil;
        S:=q^.info;
        dispose(q);
        RE:=S;
end
```

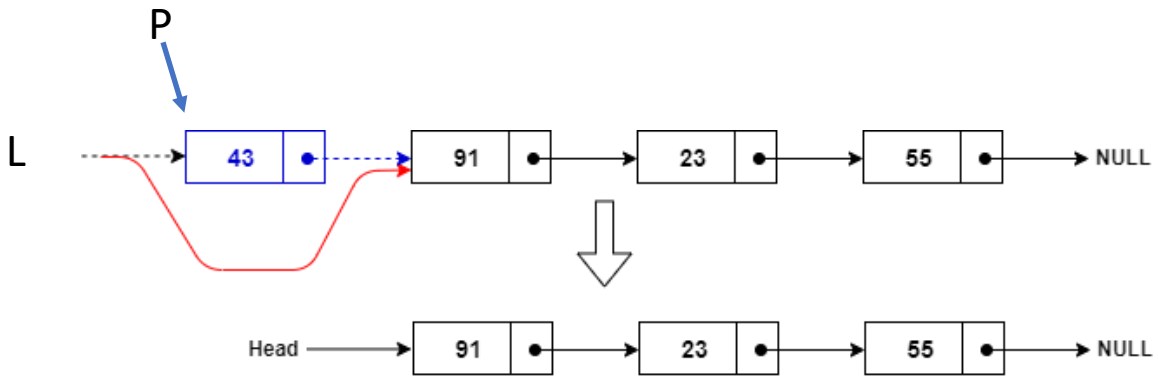


Delete the Last Node of a Linked List

لیست پیوندی

- حذف گره ابتدایی از لیست L

```
P:=L
L:=L^.next;
P^.next=nil;
dispose(P);
```

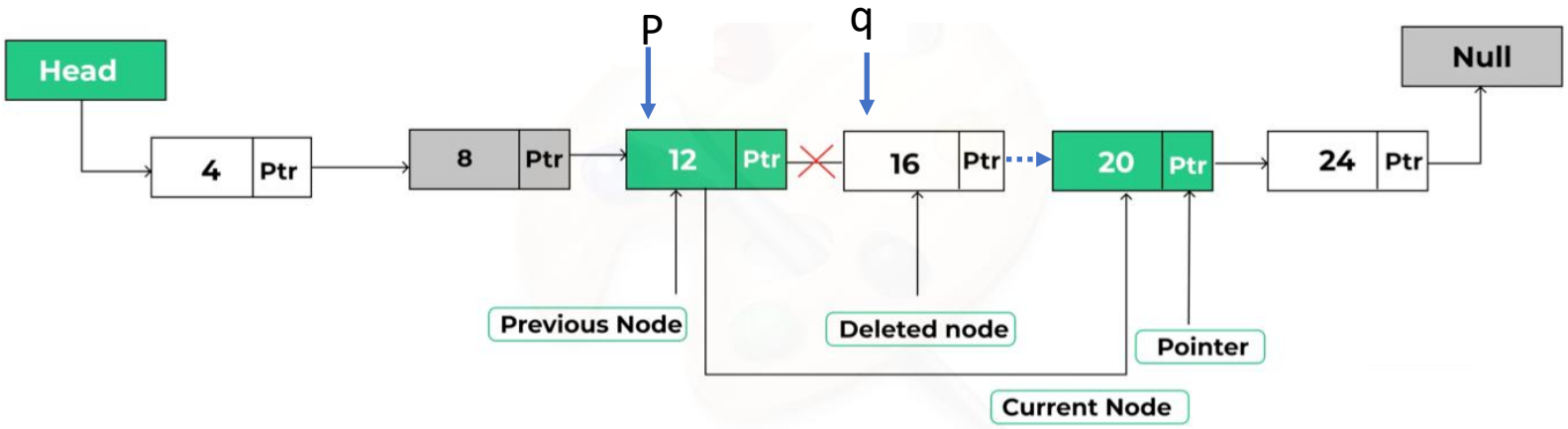


لیست پیوندی

- حذف گره kام از لیست L

```
P:=L;
for i:=1 to k-2 do
        P:=P^.next;
q:=P^.next;
P^.next:=q^.next;
q^.next=nil;
dispose(q);
```

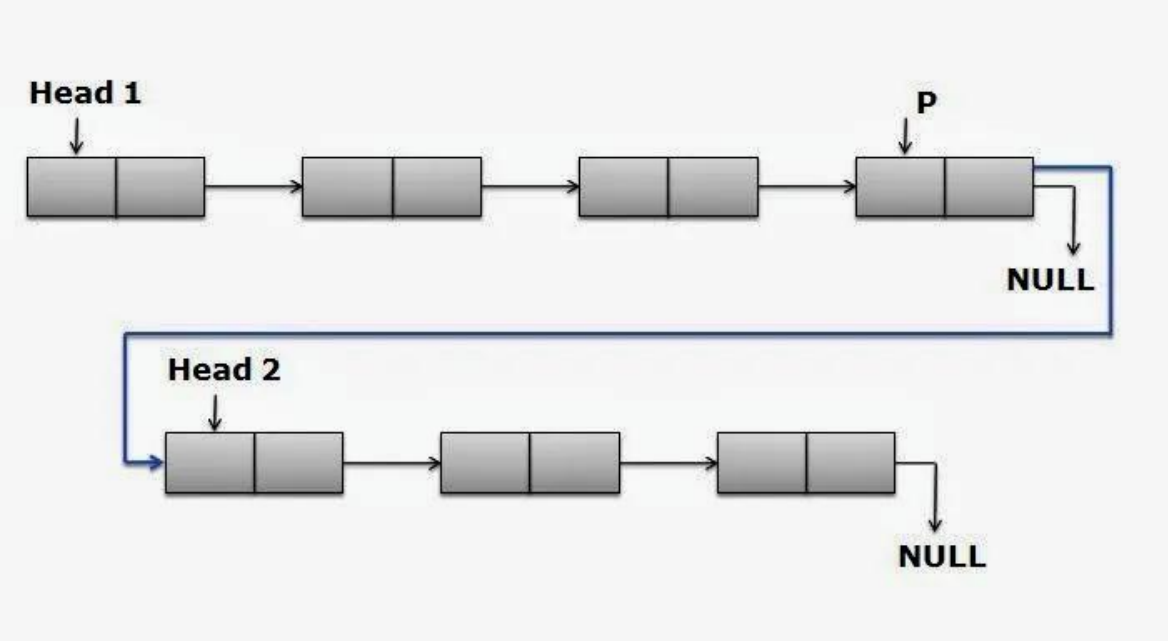
L



لیست پیوندی

- الحاق (Concatenate) دو لیست بهم

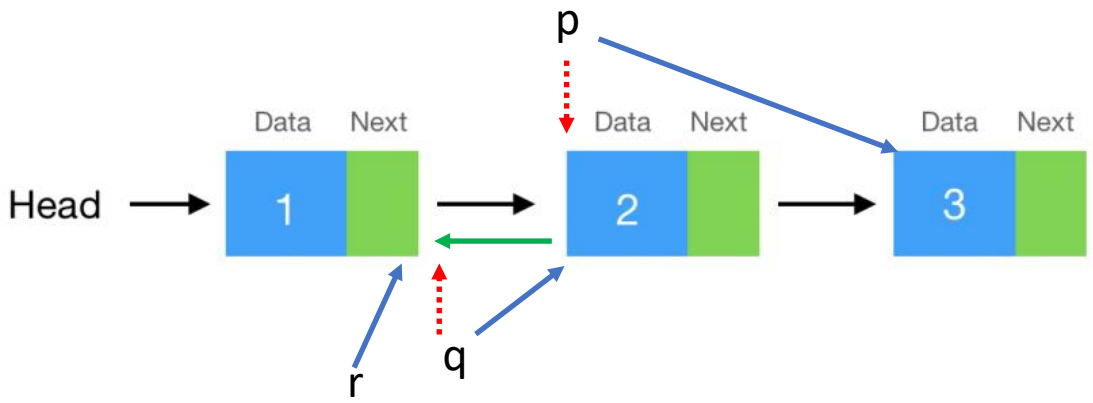
```
P:=L1;
while p^.next<>nil do
        P=P^.next;
P^.next:=L2;
L2:=nil;
```

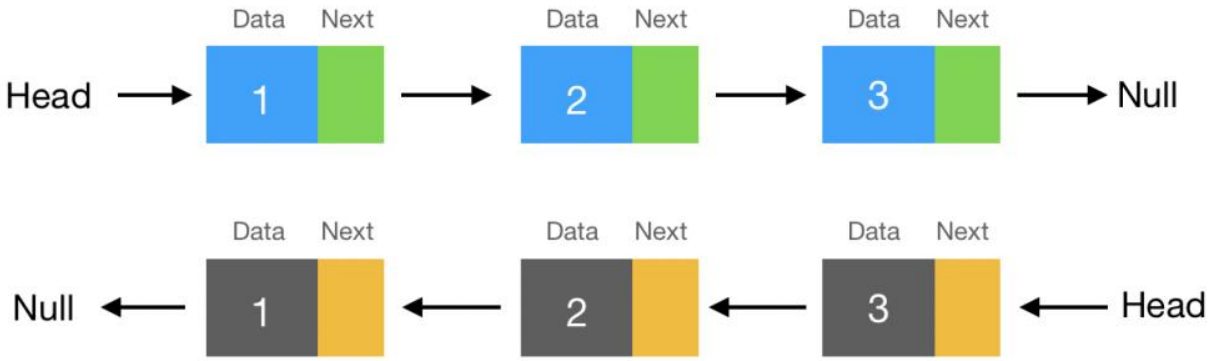


لیست پیوندی

❑معکوس کردن یک لیست پیوندی

```
p:=L;
q:=nil;
while p<>nil do
begin
        r:=q;
        q:=p;
        p:=p^.next;
        q^.next:=r;
end;
L:=q;
```

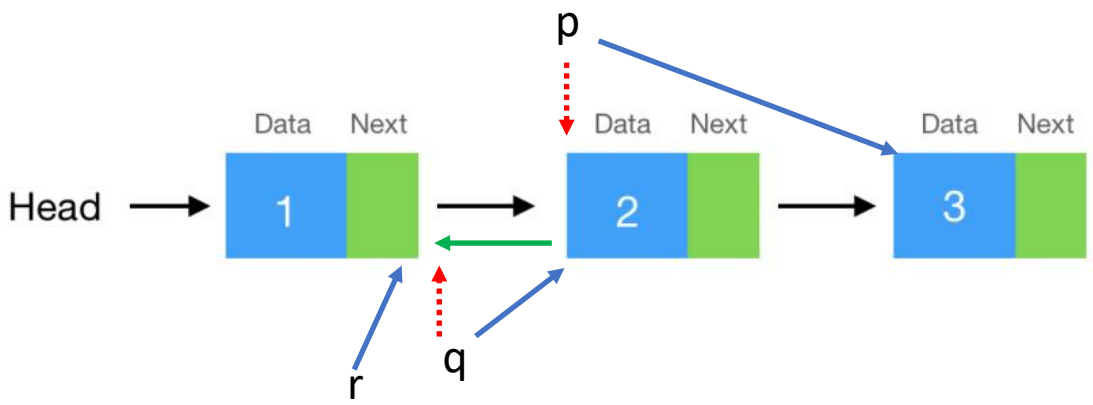




لیست پیوندی

❑معکوس کردن یک لیست پیوندی

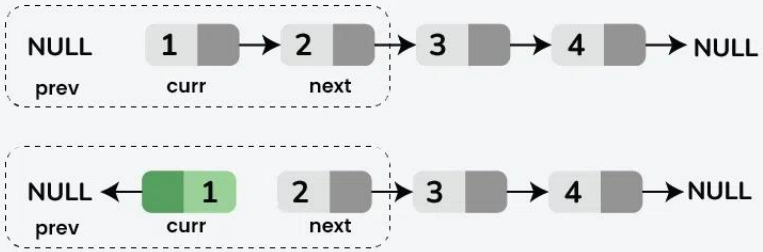
```
p:=L;
q:=nil;
while p<>nil do
begin
        r:=q;
        q:=p;
        p:=p^.next;
        q^.next:=r;
end;
L:=q;
```



01 Step | Input Linked List having 4 nodes

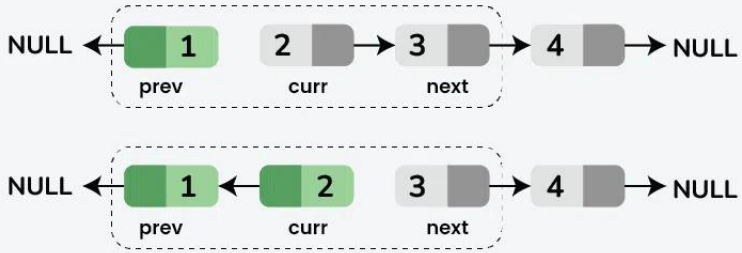
Reverse Linked List Using Iterative Method

02 Step | Initialize prev pointer = NULL
Store next = curr.next
Update curr.next = prev



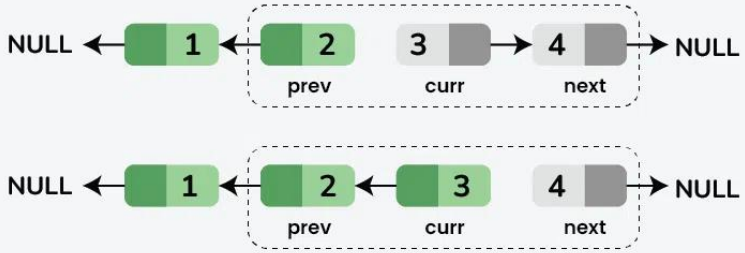
Reverse Linked List Using Iterative Method

03 Step | Update prev = curr and curr = next
Store next node 3 as next
Update next pointer of current node 2 to previous node 1.



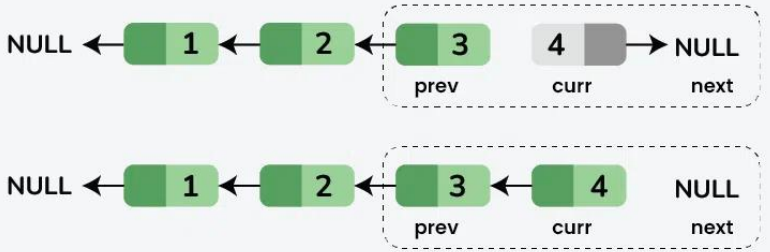
Reverse Linked List Using Iterative Method

04 Step | Update prev = curr and curr = next
Store next node 4 as next
Update next pointer of current node 3 to previous node 2.



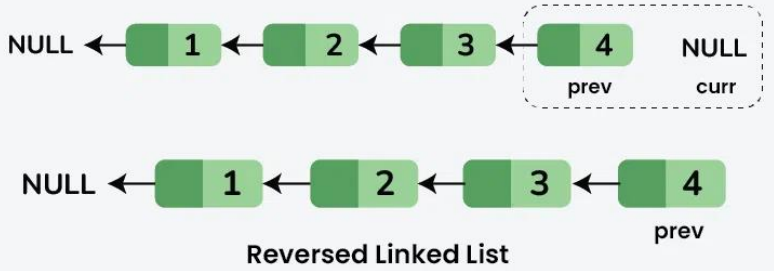
Reverse Linked List Using Iterative Method

05 Step | Update prev = curr and curr = next
next pointer points to NULL
Update next pointer of current node 4 to previous node 3.



Reverse Linked List Using Iterative Method

06 Step | Update prev = curr and curr = next
Finally, curr becomes NULL and prev stores the head of the reversed linked list



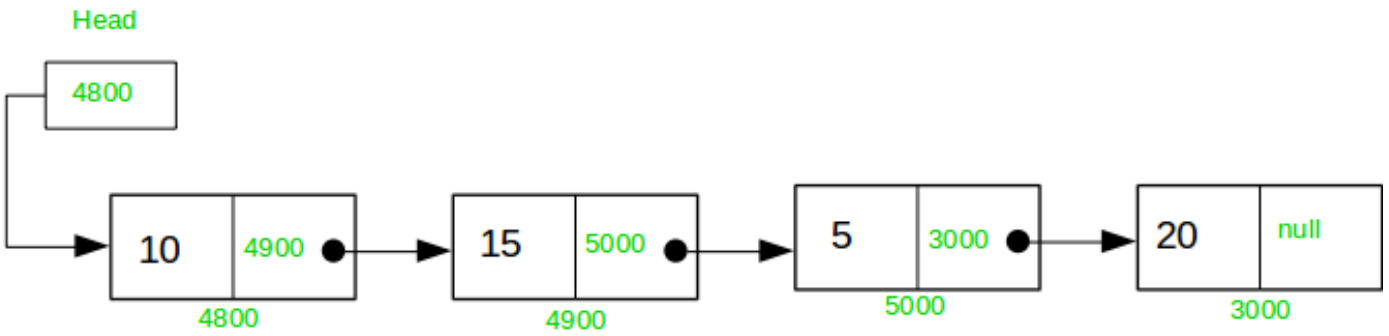
Reverse Linked List Using Iterative Method

لیست پیوندی

- چاپ محتویات لیست (یکطرفه) به روش بازگشتی

```
procedure Travers (x : listpointer);
begin
        if (x <> nil) then begin
                write (x^. data);
                Travers (x^.link);
        end;
end;
```

```
void    Travers (listpointer x)
{
   if (x! = NULL) {
    printf("%d", x → data),
     Travers (x → link);
   }
}
```



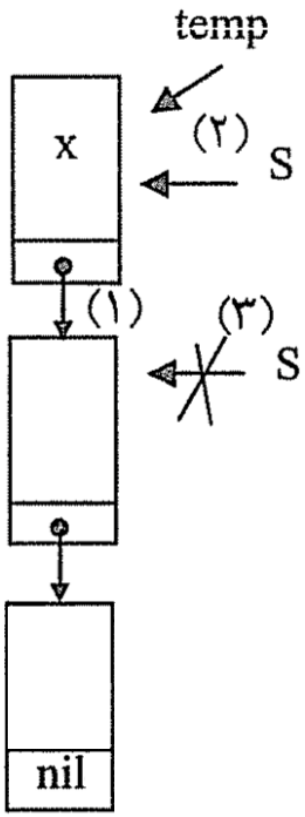
پیاده سازی پشته با لیست پیوندی

- در پیاده سازی پشته، اولین گره لیست، عنصر بالای پشته است و همواره اشاره گری داریم که به عنصر بالای پشته (همان اشاره گر به اول لیست) اشاره می کند.
- در الگوریتم بعدی اشاره گر S همواره به ابتدای پشته اشاره می کند و حذف فقط از یک سر لیست انجام می شود.

پیاده سازی پشته با لیست پیوندی

- عمل Push(S, x) : مقدار x را در بالای پشته S قرار می دهد.

پاسکال	C++
new (temp) ; temp^.data:= x; temp^.link:=S; ⇒ رسم فلش (۱) S:=temp; ⇒ رسم فلش (۲) و حذف فلش (۳)	temp=new NODE; temp → data = x; temp → link = S; S = temp;



پیاده سازی پشته با لیست پیوندی

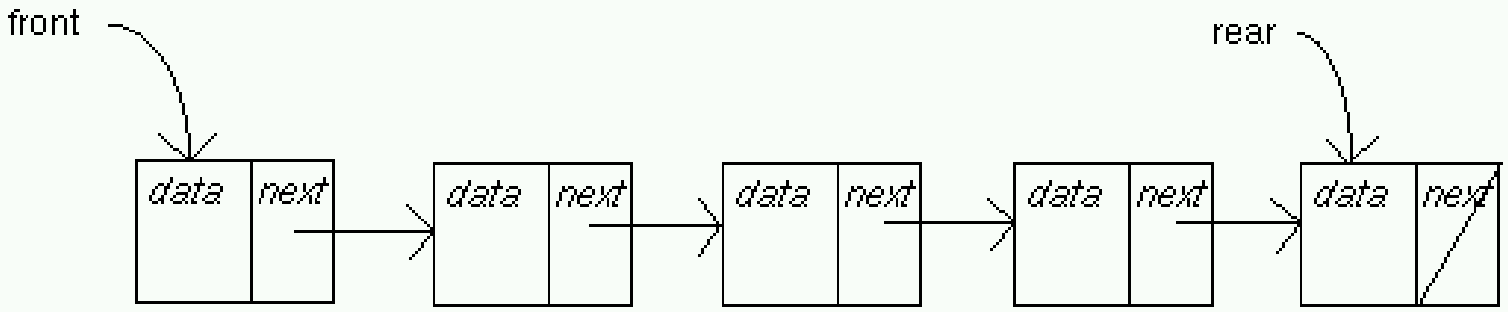
- عمل Pop(S, x) : مقدار x را از بالای پشته S حذف می کند.

پاسکال	C++

temp:=S;
S:=temp^.link;
x:=temp^.data;
dispose(temp);
	temp = S;
	S = temp → link;
	x = temp → data;
	delete (temp);

پیاده سازی صف با لیست پیوندی

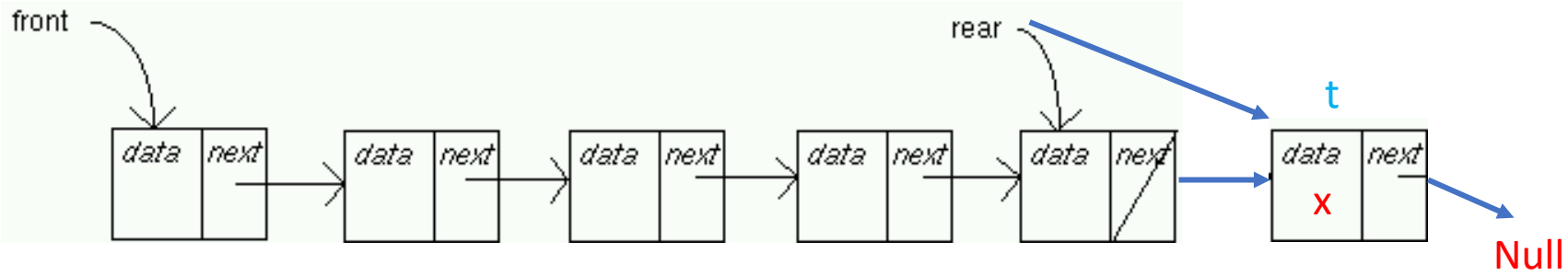
- اضافه کردن گره (داده) به انتهای صف (Rear) و حذف گره (داده) از ابتدای صف (Front) انجام می گیرد.



اضافه کردن به صف با لیست پیوندی

- اضافه کردن گره (داده) به انتهای صف (Rear)

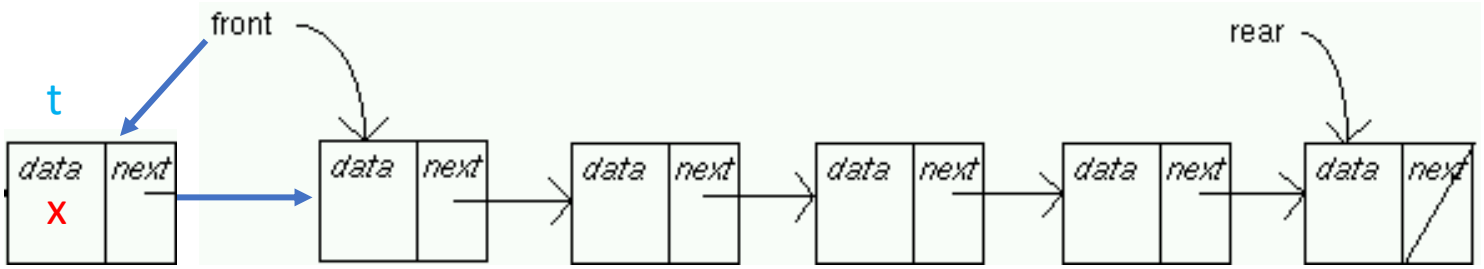
```
t = new   NODE;
t → data = x;
t → link = NULL;
if(front = = NULL) front = t;
else  rear → link = t;
rear = t;
```



حذف کردن از صف با لیست پیوندی

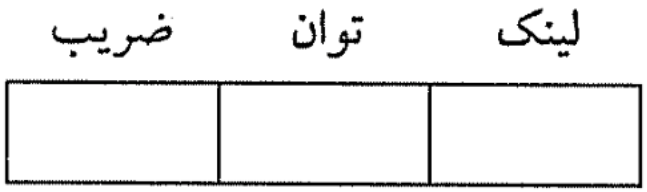
- حذف (خواندن) از صف و قرار دادن مقدار آن گره در x

```
if (front = = NULL) queueempty( );
else {
    t = front ;
    front = t → link;
    x = t → data ;
    if (front = = NULL) rear = NULL;
    delete (t);
}
```



کاربرد لیست پیوندی

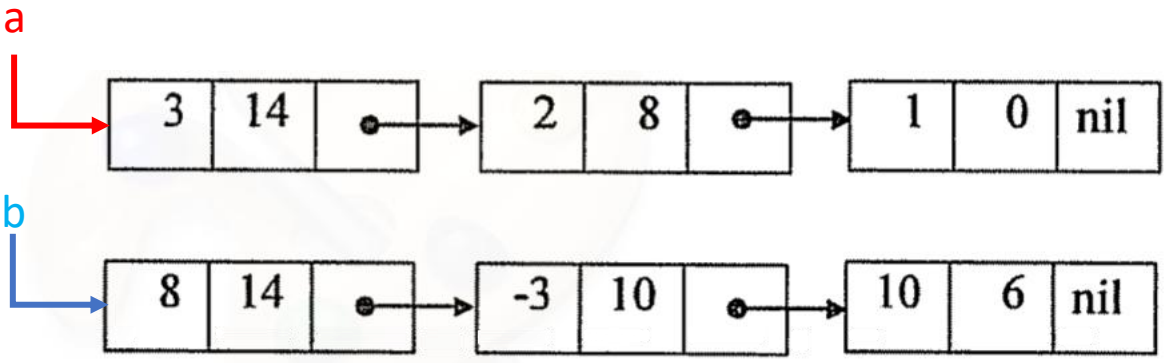
- نمایش چند جمله‌ای‌ها
- برای این منظور می توانیم از لیست خطی که هر گره آن سه فیلد زیر را دارد استفاده کرد:



- مثال : نمایش چند جمله‌ای های زیر با لیست پیوندی:

$a = 3x^{14} + 2x^{8} + 1$

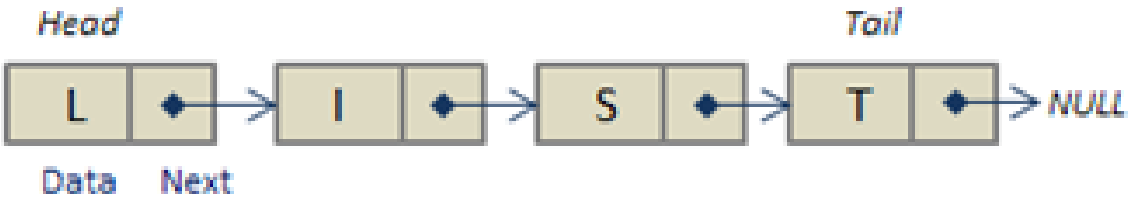
$b = 8x^{14} - 3x^{10} + 10x^{6}$



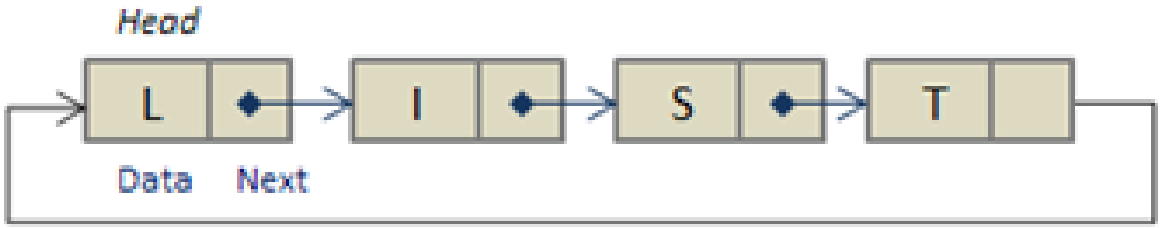
لیست پیوندی حلقوی(چرخشی)

- به جای اینکه انتهای لیست به Null ختم شود، به ابتدای لیست اشاره خواهد داشت.

Singly Linked List:



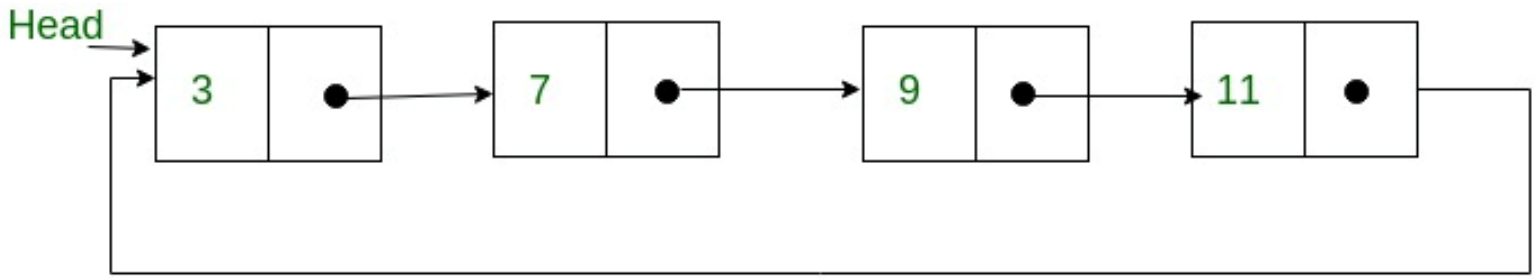
Circularly Linked List:



لیست پیوندی چرخشی

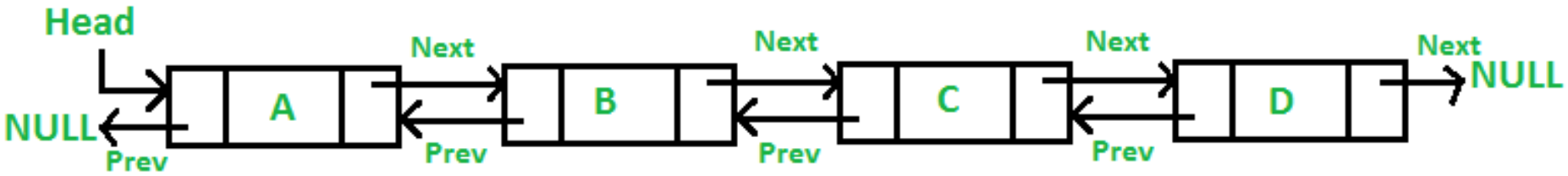
- محاسبه مجموع مقادیر گره های لیست چرخشی

```
P:=L;
Repeat
        S:=S+P^.info;
        P:=P^.next;
Until P=L;
```



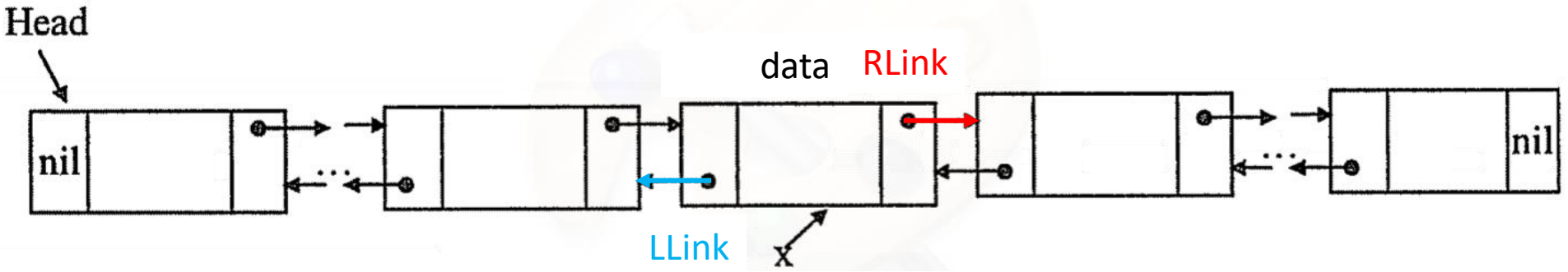
لیست های پیوندی (دوطرفه)

Doubly Linked Lists



مقدمه

- در لیست پیوندی دو طرفه، در هر گره دو اشاره گر وجود دارد.
  - یک اشاره گر به گره بعدی (Next = RLink)
  - یک اشاره گر به گره قبلی (Previous = LLink)
- به این ترتیب می توان لیست را در هر دو جهت پیمایش نمود.



مقدمه

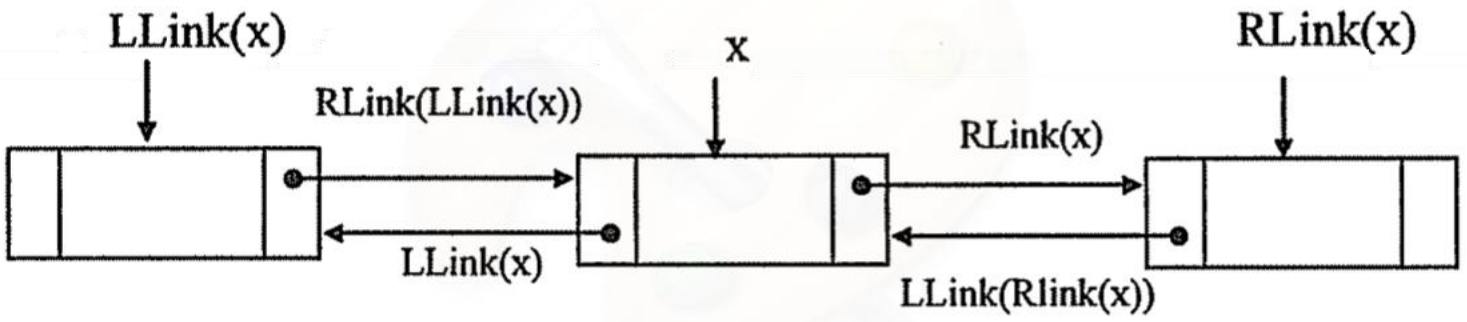
- در لیست پیوندی دو طرفه، رابطه زیر برقرار است:

RLink(LLink(x)) = LLink(RLink(x)) = x

- **رابطه بالا به زبان پاسکال و سی:**

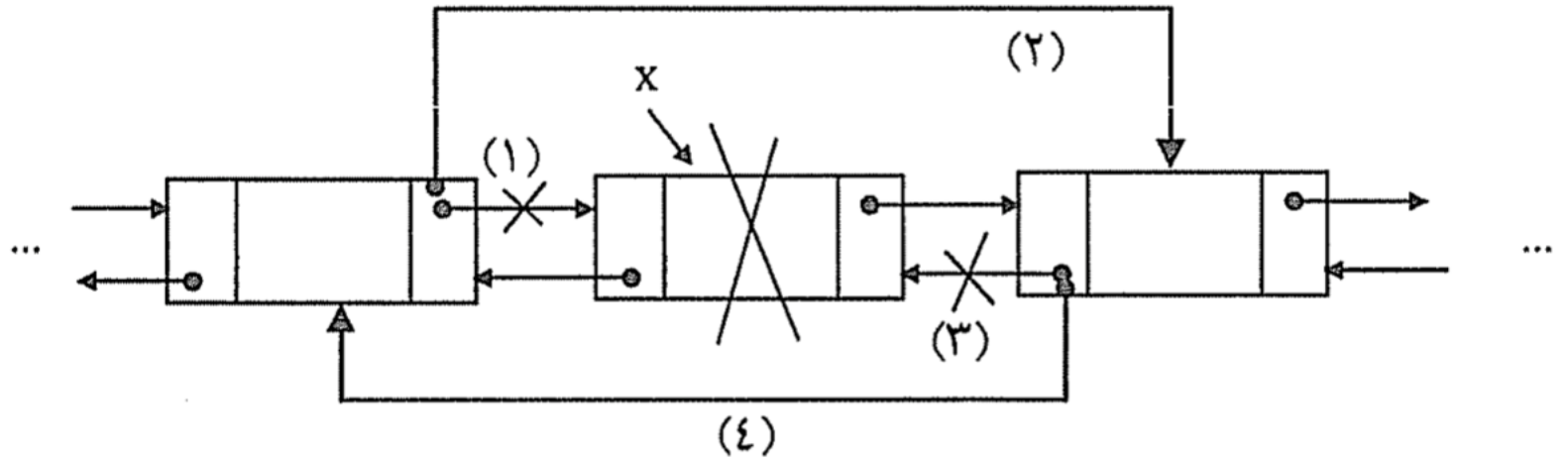
- **x^.LLink^.RLink = x^.RLink^.LLink = x**

- **x->LLink->RLink = x->RLink->LLink = x**



حذف گره از لیست پیوندی دو طرفه

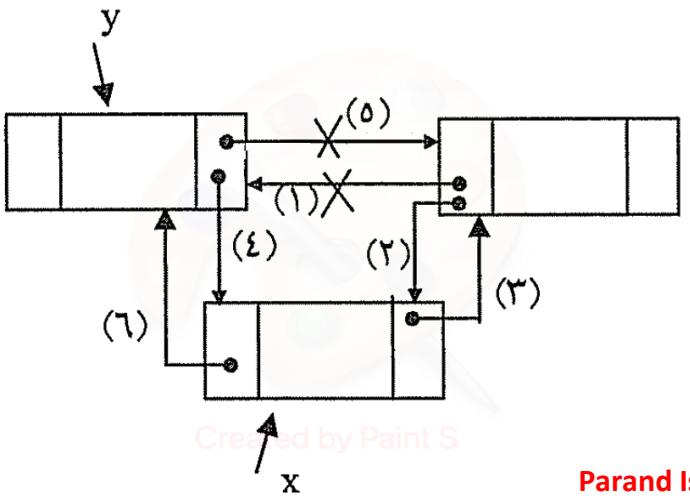
زبان فرضی	C++
Rlink (Llink(x)) := Rlink (x) ; ⇒ حذف فلش (۱) و ترسیم فلش (۲)	x → Llink → Rlink = x → Rlink;
Llink (Rlink(x)) := LLink (x);⇒ حذف فلش (۳) و ترسیم فلش (٤)	x → Rlink → Llink = x → Llink;
dispose(x) ;	delete(x)



اضافه کردن گره به لیست پیوندی دو طرفه

- برای اضافه کردن گره x به سمت راست گره y :

زبان فرضی	C++
new (x) ;	x = new NODE;
LLink (RLink(y)): = x; ⇒ (۲) حذف فلش (۱) و ترسیم فلش	y → Rlink → Llink = x;
Rlink(x) := Rlink(y); ⇒ (۳) رسم فلش	x →Rlink = y → Rlink;
Rlink(y):= x; ⇒ (٥) رسم فلش (٤) و حذف فلش	y → Rlink = x;
LLink(x):=y; ⇒ (٦) رسم فلش	x → Llink = y ;



تمرین

۱) قطعه برنامه زیر چه عملی انجام میدهد؟

پاسکال	C , C++

x,y,z, p:pointer;
p:=x;
z:=x;
while p^.link <> nil do
p:=p^.link;
p^.link:=y;
	pointer x,y,z,p;
	p = x;
	z = x;
	while (p → link !=NULL)
	p = p → link;
	p → link = y;

تمرین

۲) تابع زیر چه عملی انجام میدهد؟

پاسکال

```
PROCEDURE f(VAR Start: Nodeptr);
BEGIN
        IF Start ^.Link=Nil THEN
        BEGIN
                Dispose (Strart);
                Start:= Nil;
        END
        ELSE
        f(Start^.Link)
END;
```

C++

```
void   f(Nodeptr   Start)
{
  if (Start → Link = = NULL)
    {
       delete (Start);
       Start = NULL;
    }
   else   f(Start → link);
}
```

تمرین

۳) تابع زیر چه عملی انجام میدهد؟

پاسکال

```
Function  M (L : List ; a : integer) : List ;
Var       x: integer ;
begin
                  x : = x ↑ . Link ;
        while (x <> Nil) and (x ↑ . data <> a)
                  x : = x ↑ . LINK ;
        M : = x ;
end .
```

C++ , C

```
List  M(List  L,  int a)
{    int x;
     x = x → link;
     while  ((x != NULL) && (x → data != a))
                  x = x → link;
      return(x);
}
```

تمرین

۴) تابع زیر چه عملی انجام میدهد؟ (first اشاره به ابتدای لیست دارد)

پاسکال

```
procedure  x(first : pointer);
begin
     if  first  <> nil  then
       begin
          x(fist ↑ .link);
          writeln (frist ↑ .data);
       end;
end;
```

C++

```
void  x(pointer  first)
{
    if(first  != NULL)
    {
        x (first → link);
        cout << first → data;
    }
}
```

تمرین

۵) مزیت لیست یکطرفه چرخشی نسبت به لیست یکطرفه ساده کدام است؟