ساختمان داده ها

دانشگاه صنعتی نوشیروانی بابل

دکتر حسام عمران پور

طراحان اسلاید:
زهرا ریحانیان و دانیال علیزاده

حل تمرین :
علی باقری

لینک کانال تلگرام اطلاع رسانی و حل تمرین:
t.me/ds_nit_4011

فصل ۴
لیست

لیست : دنباله ای از صفر یا چند عنصر هم نوع است.

<a1,a2,a3,…,an>

در لیست ترتیب مهم است.

n>=0: طول یا اندازه لیست
a1: عنصر ابتدای لیست
an: عنصر انتهای لیست
n == 0 : لیست خالی است

تفاوت های لیست و مجموعه

۱- نحوه نمایش: لیست <...> مجموعه {...}

۲- لیست عضو تکراری می پذیرد اما عضو تکراری در مجموعه بی معنی است.

مثال:

L1 = <1,2,2> L2 = <1,2> L1 ≠ L2

S1 = {1,2,2} S2 = {1,2} S1 = S2

۳- ترتیب اعضا در لیست مهم است اما در مجموعه فرقی ندارد.

مثال:

L1 = <1,5,3> L2 = <1,3,5> L1 # L2

S1 = {1,5,3} S2 = {1,3,5} S1 = S2

عملیات روی لیست از دیدگاه ADT

1- ایجاد لیست جدید:

create(ref L : List)

2- بررسی خالی بودن لیست:

isEmpty(L : List): Boolean

3- خالی کردن لیست:

makeNull(ref L : List)

4- برگرداندن آدرس عنصر اول:

first(L : List): Position

(لیست خالی باشد Null برمی گرداند)

5- آدرس عنصر بعد از p:

next(L : List ; p : Position):Position

(اگر p آدرس عنصر آخر باشد Null برمی گرداند)

6- آدرس عنصر ماقبل p: prev(L : List ; p : Position):Position

(اگر p عنصر ابتدا باشد Null برمی گرداند)

7- عنصر x را در آدرس بعد از p قرار می دهد(در کتاب های دیگر ممکن است در همان مکان p قرار دهند):

insert(ref L : List ; p : Position ; x : Element Type)

(اگر p Null باشد در خانه نخست قرار می گیرد)

8- حذف عنصر موجود در آدرس p: delete(ref L:List ; ref p:Position)

پس از این عملیات ، p به عنصر بعدی اشاره خواهد کرد. (فراخوانی با ارجاع)

9- مقدار عنصری که در آدرس p است را برمی گرداند:

retrieve(L : List ; p : Position):Element Type

بررسی کنید لیست L بعد از این دستورات به چه صورتی خواهد بود.

?

L : <2,5,7>

1. Insert (L, next(L, first(L)), 8)

L : <2,5,8,7>

2. Insert (L, null, 5)

L : <5,2,5,8,7>

تابعی بنویسید که مقدار عنصر آخر را در یک لیست برگرداند.

```
last (L : List) : Element Type
{
    if (isEmpty(L))
        return Error;

    else {
        p : position;
        p = first(L);

        while (next(L, p) != null)
            p = next(L, p);

        return retrieve(L, p);
    }
}
```

حل مثال قبل به فرم بازگشتی

```
last (L : List) : Element Type
{
    if (isEmpty(L))
        return Error;

    else if (next(L, first(L)) == null)
        ret retrieve (L, first(L));
    else {
        delete (L, first(L));
        return last(L);
    }
}
```

تابعی بازگشتی بنویسید که یک لیست به عنوان آرگومان ورودی دریافت کند و عناصر آن را از آخر به اول در خروجی چاپ کند.

?

```
invPrint (L : List)
{
    if (! isEmpty(L))
    {
        x : Element Type
        f : position
        f = first(L);
        x = retrieve(L, f);
        delete(L, f);
        invPrint(L);
        write(x);
    }
}
```

مرتبه زمانی

$O(n)$

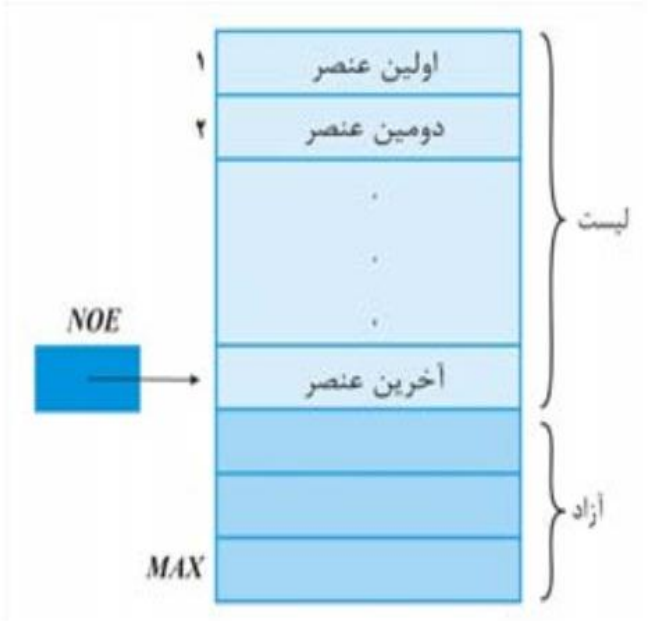
با استفاده از توابع پایع ، شبه کدی بنویسید که یک عنصر را در لیست جست و جو کند.

```
search( L : list; x : ElementType ) : Position
{
    p : Position;

    p = first(L);
    while p != null do
    {
        if retrieve(L, p) == x then
            return p;
        p = next(L, p);
    }
    return null;
}
```

پیاده سازی لیست با آرایه

ساختمان داده لیست را با استفاده از آرایه می توان به صورت زیر تعریف کرد :

```
Type List = Record
{
    Elements : Element Type[Max]
    NOE : integer
}

Type Position = integer

Const Null = 0
```



پیاده سازی آرایه ای توابع ADT لیست

1- ایجاد لیست جدید

```
Create ( ref L : List )
{
        L.NOE = 0;
}
```

2- خالی کردن لیست

```
MakeNull ( ref L : List )
{
        L.NOE = 0;
}
```

3- بررسی خالی بودن لیست

```
Empty ( L : List ) : Boolean
{
        if (L.NOE == 0)
                return True;

        return False;
}
```

هر جا که قرار است روی ورودی نیز تغییر ایجاد شود، فراخوانی با ارجاع است در غیر این صورت فراخوانی با مقدار خواهد بود.

4- مکان عنصر ابتدا لیست

```
First ( L : List ) : Position
{
        if (L.NOE == 0)
                return Null;

        return 1;
}
```

5- درج یک عنصر جدید

```
Insert ( ref L : List ; p : Position ; x : ElementType)
{
        if (L.NOE == Max)
                return Error("Overflow");

        for i = L.NOE downto (p + 1)
                L.Elements[i + 1] – L.Elements[i];

        L.Elements[p + 1] = x;

        L.NOE++;
}
```

6- مکان عنصر بعدی

```
Next ( L : List ; p : Position) : Position
{
        if ( p < L.NOE )
                return p+1;

        return Null;
}
```

عنصر بعدی آخرین عنصر، Null است.

7- مکان عنصر قبلی

```
Prev ( L : List ; p : Position) : Position
{
        if ( p > 1 )
                return p - 1;

        return Null;
}
```

8- حذف یک عنصر

```
Delete ( ref L : List ; ref p : Position)
{
        for  i = p to (L.NOE-1)
                L.Elements[i] = L.Elements[i+1];

        L.NOE --;

        if ( p > L.NOE )
                p = Null;
}
```

9- بازیابی یک عنصر

```
Retrieve ( L : List ; p : Position) : ElementType
{
        return L.Elements[p];
}
```

لیست پیوندی

اشاره گر لیست پیوندی

دستور new : یک رکورد از نوع آن اشاره گر در حافظه جدا می کند و آدرس آن را در یک متغیر اشاره گر قرار می دهد.(تخصیص فضا)

دستور dispose: رکوردی که این اشاره گر به آن اشاره می کرد را به کامپایلر (سیستم عامل) برمی گرداند.(آن فضا را آزاد می کند)

لیست پیوندی یک طرفه

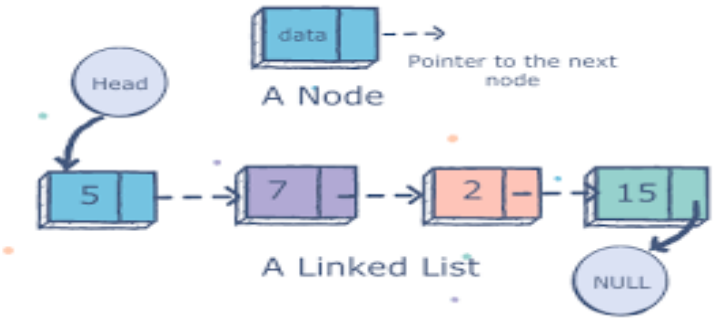
پیاده سازی لیست با لیست پیوندی یک طرفه

لیست پیوندی یک طرفه:

```
Type CellType = Record {
    Data : Element Type;
    Next : ^CellType;
}

Type List = ^CellType;
Type Position = ^CellType;
Const Null = 0;
```

۱- ایجاد لیست جدید

```
Create ( ref L : List )
{
        L = Null;
}
```

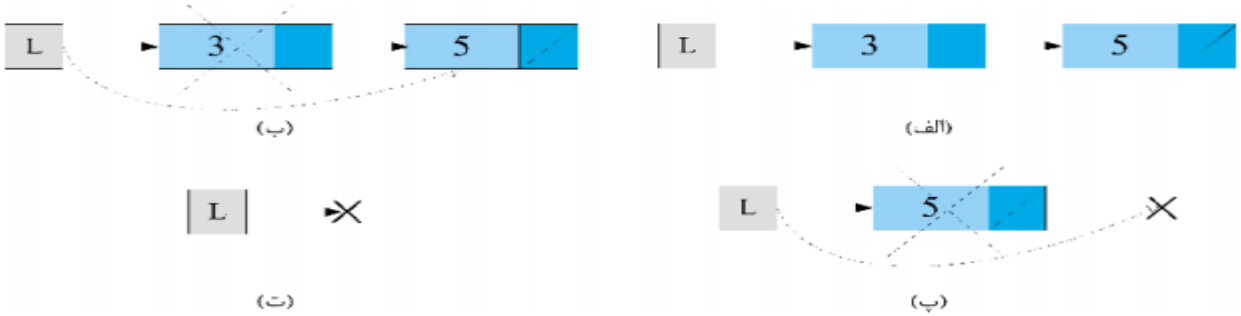


۲- خالی کردن لیست

```
MakeNull ( ref L : List )
{
        p : Position

        while (L != Null)
        {
                p = L^.Next
                dispose(L)
                L = p
        }
}
```

بازگشتی ⟹

```
if (L != Null) {
    MakeNull(L^.Next)
    Dispose(L)
    L = Null
}
```



۳- بررسی خالی بودن لیست

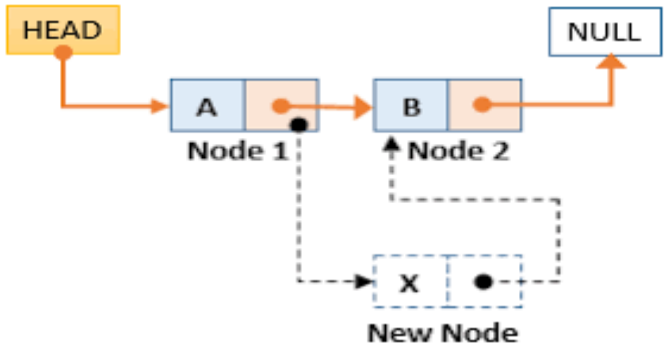
```
Empty ( L : List ) : Boolean
{
        if L == Null
                return True;

        return False;
}
```

۴- آدرس عنصر ابتدا

```
First ( L : List ) : Position
{
        return L;
}
```

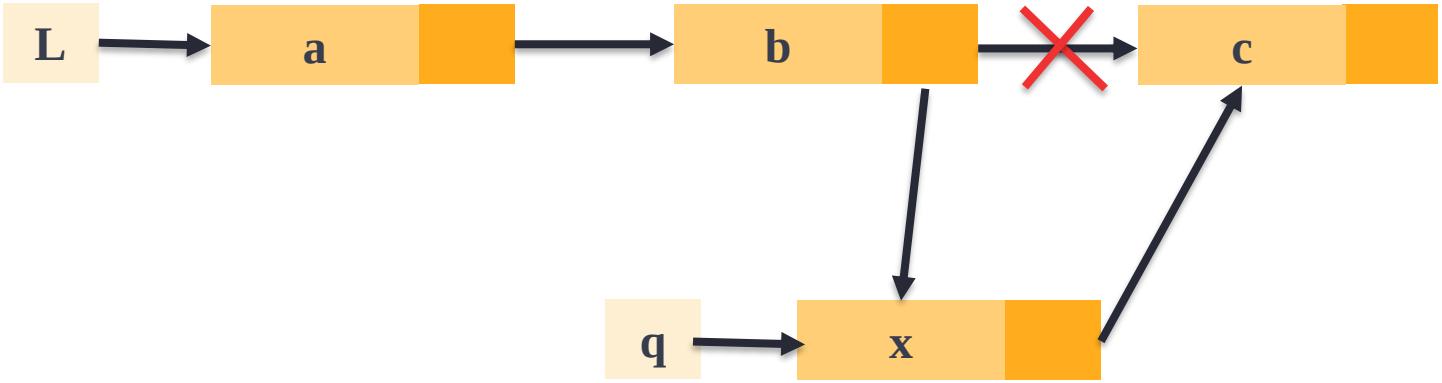
۵- درج یک عنصر جدید

```
Insert ( ref L : List ; p : Position ; x : ElementType )
{
        q = ^cellType
        new(q)
        q^.Data = x
        if p == Null
        {
                q^.Next = L
                L = q
        }
        else
        {
                q^.Next = p^.Next
                p^.Next = q
        }
}
```



برای درج یک عنصر جدید بعد از عنصر y ، ابتدا آدرس عنصر **بعدی** y یعنی p به آن داده می شود. اگر p برابر null باشد این عنصر به ابتدای لیست اضافه می شود.

- درج عنصر جدید پس از دومین عنصر در یک لیست پیوندی یک طرفه با ۳ عنصر



مرتبه زمانی

O(1)

۶- آدرس عنصر بعدی

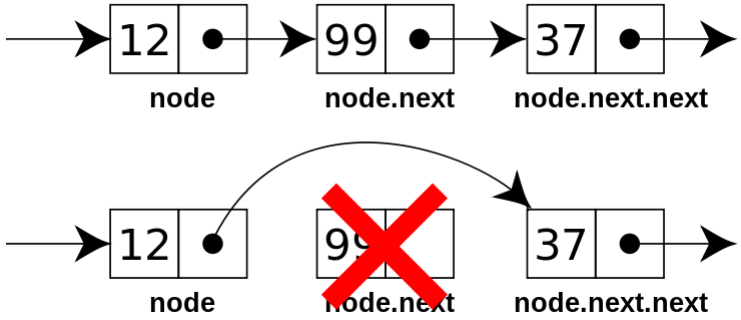
```
Next ( L : List ; p : Position ) : Position
{
        if p != Null
                return p^.Next;

        return L;
}
```

۷-آدرس عنصر قبلی

```
Prev ( L : List ; p : Position ) : Position
{
        q = Position
        if p != L
        {
                q = L
                while ( q^.Next != p )
                        q = q^.Next;

                return q;
        }

        return Null;
}
```

۸- حذف یک عنصر

```
Delete ( ref L : List ; ref p : Position )
{
     q = Position
     q = L;
     if p != L
    {
          while q^.Next != p
               q = q^.Next;
          q^.Next = p^.Next;
          dispose(p);
          p = q^.Next;
     }
     else
     {
          L = L^.Next;
          dispose(p);
          p = L;
     }
}
```



برای حذف یک عنصر از لیست ، باید ابتدا اشاره گر عنصر قبلی آن یعنی p به عنصر معتبر بعدی اشاره کند و سپس خود آن عنصر از حافظه پاک شود. همچنین اگر این عنصر اولین عنصر لیست باشد، به جای اشاره گر عنصر قبلی ، باید متغیر لیست اصلاح گردد .

مرتبه زمانی O(n)

برنامه بازگشتی بنویسید که یک لیست جدید کپی از لیست ورودی بسازد.

```
Copy (L : List) : List
{
        if L == Null
                return Null;

        else
        {
                q , L1 : Position
                q = Copy(L^.Next);
                new(L1)
                L1^.Data = L^.Data;
                L1^.Next = q;
                return L1;
        }
}
```

از آخر به اول

یا در قسمت else می توان نوشت :

```
...

else
{
        new(q)
        q^.Data = L^.Data
        q^.Next = Copy(L^.Next)
        return q;
}
```

از اول به آخر

برنامه بازگشتی بنویسید که داده های زوج و فرد یک لیست را در دو لیست جداگانه ذخیره کند.

```
Split (L : List , ref L1, ref L2 : List) {
        if (L == Null)
                L1 , L2 = Null;
        else {
                if (L^.Data % 2 == 0) {
                        New L1;
                        L1^.Data = L^.Data;
                        Split(L^.Next, L1^.Next, L2);
                }
                else {
                        New L2;
                        L2^.Data = L^.Data;
                        Split(L^.Next, L1, L2^.Next);
                }
        }
}
```

برنامه بازگشتی بنویسید که داده های مکان زوج و فرد یک لیست را در دو لیست جداگانه ذخیره کند.

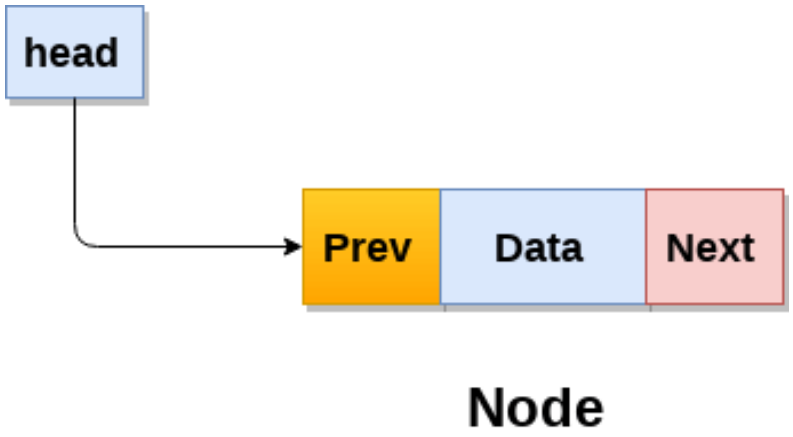
```
Split2 (L : List ; ref L1, ref L2 : List) {
        if  (L == Null)  L1, L2 = Null;
        else {
                if (L^.Next == Null){
                        new L1;
                        L1^.Data = L^.Data;
                        L1^.Next = Null;
                        L2 = Null;
                }
                else {
                        new L1;
                        new L2;
                        L1^.Data = L^.Data;
                        L2^.Data = L^.Next^.Data;
                        Split2(L^.Next^.Next , L1^.Next, L2^.Next);
                }
        }
}
```

به جای این دو خط می توان نوشت:
Split2(L^.Next, L1^.Next, L2)

لیست پیوندی دوطرفه

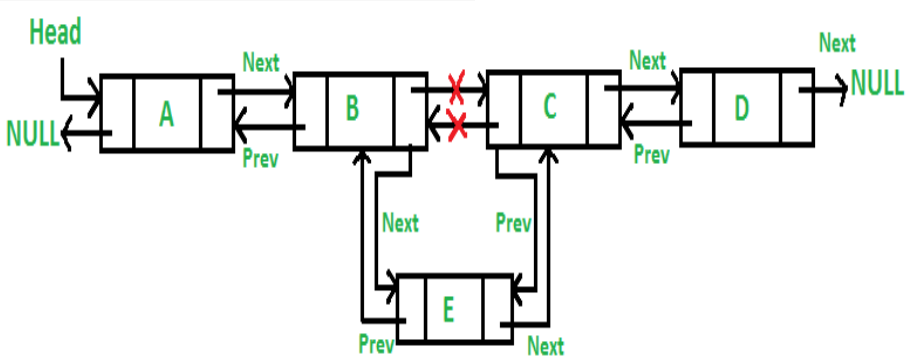
لیست پیوندی دو طرفه

```
Type CellType = Record
{
        Data : ElementType;
        Next, Prev : ^CellType;
}

Type List = ^CellType;
Type Position = ^CellType;
Const Null = 0;
```



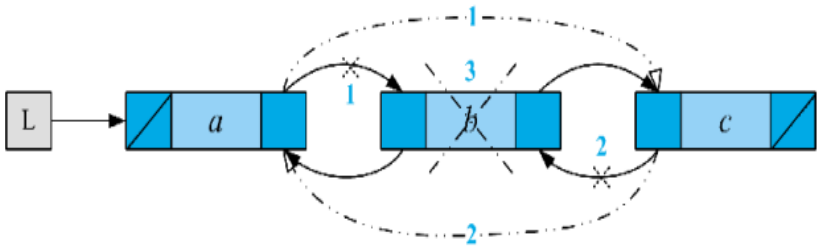
-درج یک عنصر جدید :

```
Insert (ref L : List ; p : Position ; x : ElementType)
{
    q = ^CellType
    new(q)
    q^.Data = x

    if p != Null
    {
        q^.Next = p^.Next
        q^.Prev = p
        if p^.Next != Null
            p^.Next^.Prev = q
        p^.Next = q
    }
    else
    {
        q^.Next = L
        L^.Prev = q
        q^.Prev = Null
        L = q
    }
}
```



حذف یک عنصر :

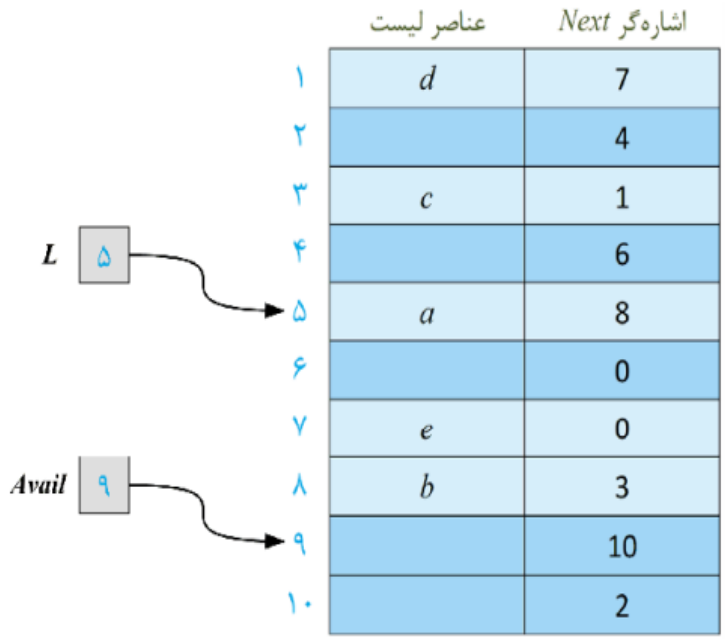
```
Delete (ref L : List ; ref p : Position)
{
    q = Position
    q = p^.Next

    if p != L
    {
        p^.Prev^.Next = p^.Next
        if p^.Next != Null
            p^.Next^.Prev = p^.Prev
    }
    else
    {
        L = L^.Next
        L^.Prev = Null
    }
    dispose(p)
    p = q
}
```



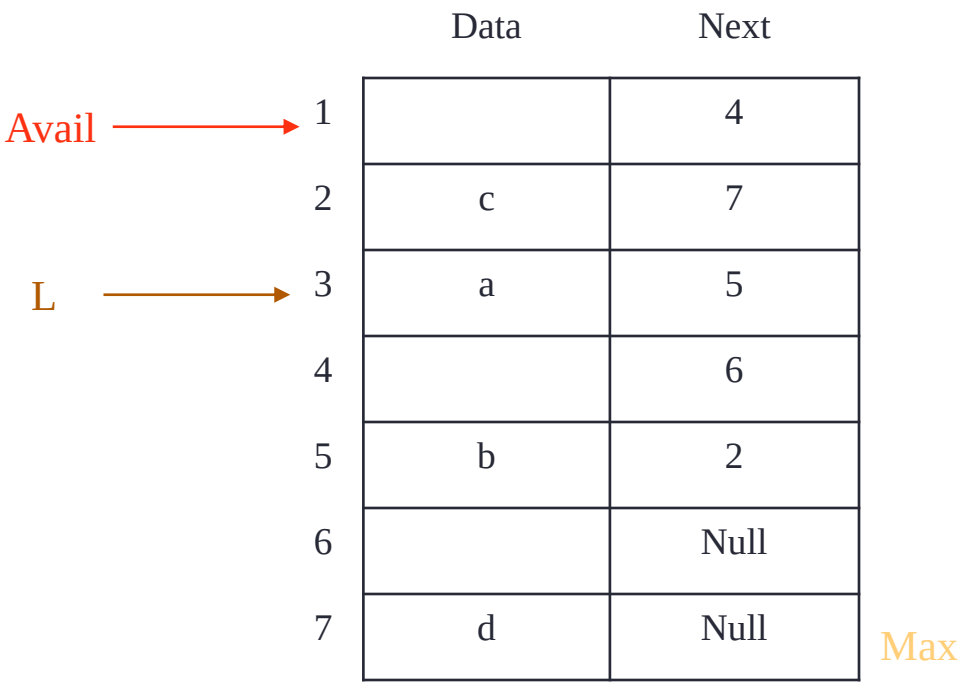
لیست با اشاره گر های اندیسی

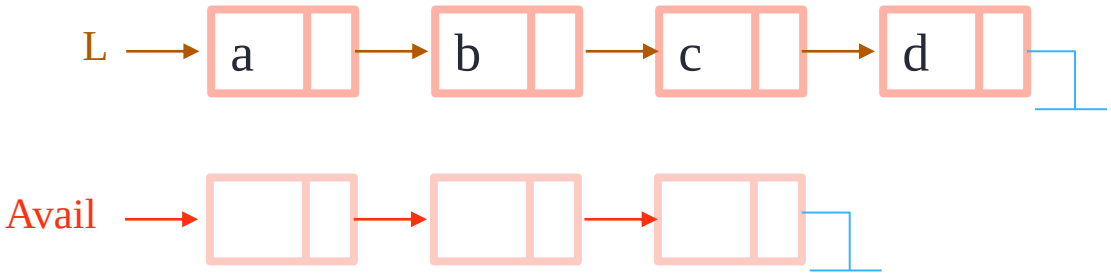
لیست اندیسی

در برخی از زبان های برنامه نویسی مانند فورترن اشاره گر حافظه وجود ندارد. بنابراین در این زبان ها برای پیاده سازی لیست با استفاده از اشاره گر ها، از اندیس عناصر در آرایه برای شبیه سازی اشاره گر ها استفاده می شود. اشاره گر ها اندیس بوده و اندیس عناصر را در آرایه نگه داری می کنند.



- **Avail** : از نوع **Position** است و به اولین سلول آزاد اشاره می کند.
- آخرین سلول آزاد و آخرین سلول لیست به اندیس صفر اشاره می کنند.


Data
Next
Avail
L
1
2
3
4
5
6
7
4
c
7
a
5
6
b
2
Null
d
Null
Max


L
a
b
c
d
Avail

ساختمان داده ی لیست با استفاده از اشاره گر های اندیسی

```
Type CellType  = Record
{
        Data : ElementType;
        Next : integer;
}
Type Position = integer;
Type List = Position;
var Space : CellType[MAX];
var Avail : Position;
Const null = 0;
```

برای شبیه شدن توابع این نوع لیست به توابع لیست های پیوندی، توابعی مشابه با توابع new و dispose طوری نوشته می شوند که عملیات تخصیص سلول و آزاد کردن سلول را انجام دهند :

```
MyNew (ref p : Position )
{
    if Avail == null then
        p = null;
    else
    {
        p = Avail;
        Avail = Space[Avail].Next;
    }
}
```

```
MyDispose ( p : Position )
{
    Space[p].Next = Avail;
    Avail = p;
}
```

مساله ای که باقی می ماند ، مقداردهی اولیه اشاره گر های آرایه هنگام ایجاد لیست جدید است، به طوری که هر سلولی به سلول آزاد بعدی اشاره کند. پس تابع create برای لیست های اندیسی به صورت زیر خواهد بود :

```
Create (ref L : List )
{
        Avail = 1;
        for i = 1 to (MAX – 1) do
                Space[i].Next = i + 1;
        Space[MAX].Next = 0;
        L = 0;
}
```

تابع insert در لیست های با اشاره گر اندیسی بنویسید.

```
Insert (ref L : List ; p : Position ; x : ElementType)
{
        if (Avail == Null)
                return Error;

        q = Avail;
        Avail = space[Avail].Next;
        space[q].Data = x;
        space[q].Next = space[p].Next;
        space[p].Next = q;
}
```

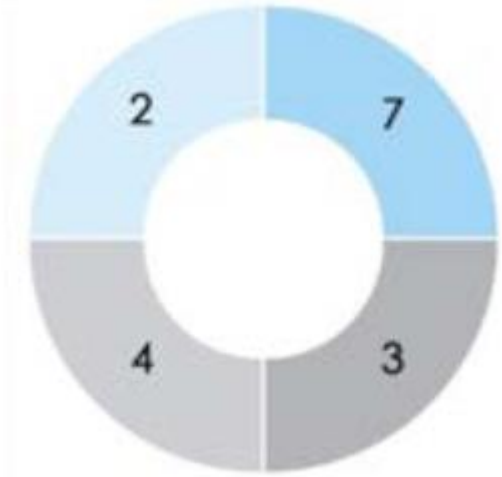
تمرین: حالت های خاص را برای درج یک عنصر جدید بررسی کنید.

لیست چرخشی(دوار)

لیست چرخشی(دوار)

این نوع لیست،عنصر ابتدا و انتها ندارد و عناصر به صورت چرخشی پشت سر هم قرار می گیرند.به عبارتی این نوع لیست را می توان یک لیست خطی فرض کرد که در پیمایش آن،پس از عنصر انتهای لیست دوباره عنصر ابتدا قرار دارد.

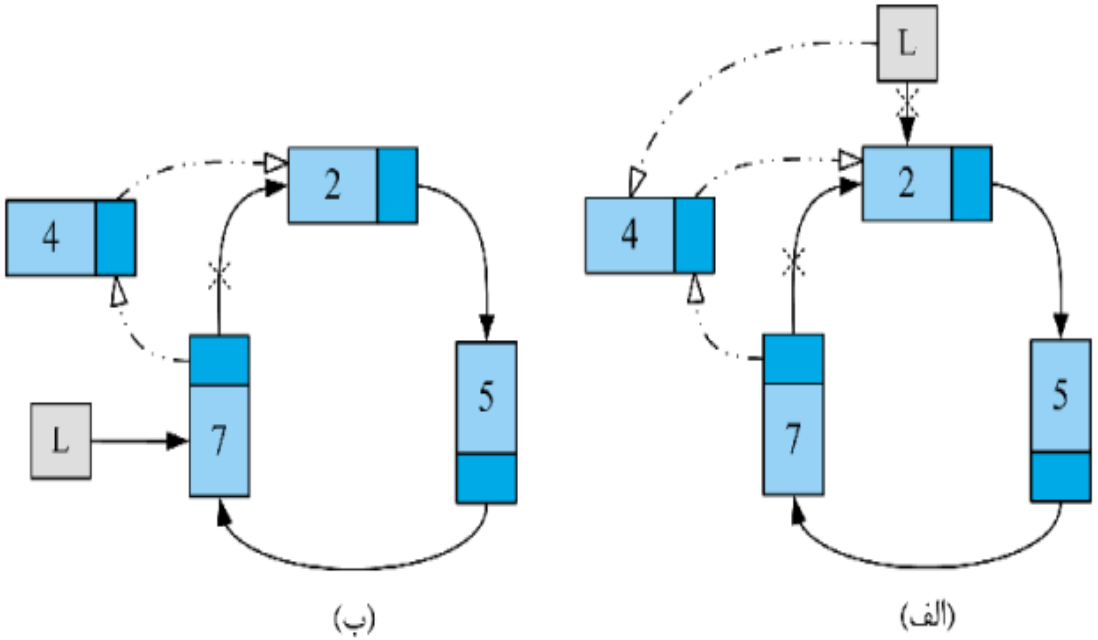
مثال: شکل زیر لیست چرخشی با ۴ عنصر <2,7,3,4> را نشان می دهد.



درج در ابتدای لیست پیوندی یک طرفه چرخشی

الف) اشاره گر لیست به عنصر ابتدای لیست اشاره کند.

ب) اشاره گر لیست به عنصر انتهای لیست اشاره کند.



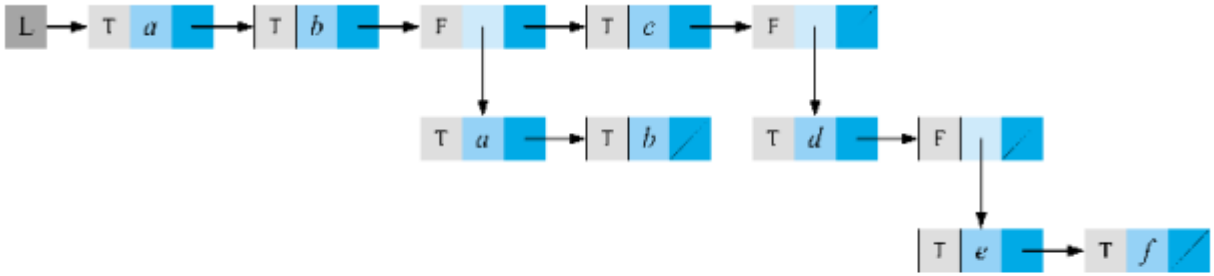
با فرض اینکه در یک لیست پیوندی یک طرفه ی چرخشی، اشاره گر لیست به عنصر انتهای لیست اشاره کند ، شبه کد درج یک عنصر در انتهای لیست را بنویسید .

```
AddLast ( ref L : List ; x : ElemetType )
{
    var q : Position;
    new(q);
    q^.Data = x;
    if L == null then
    {
        q^.next = q;
        L = q;
    }
    else
    {
        q^.next = L^.next;
        L^.next = q;
        L = q;
    }
}
```

لیست پیوندی عمومی

پیاده سازی لیست عمومی

```
Type CellType = Record
{
        Link : ^CellType;
        case Tag : boolean
        {
                True : ( Dtata : ElementType );
                False : (Dlink : ^CellType );
        }
}
Type GList = ^CellType;
Type Position = ^CellType;
```



شکل ۲۰-۴. نمایش لیست عمومی A = <a, b, <a, b>, c, <d, <e, f>>> .

شبه کد پیمایش و چاپ تمام عناصر ساده ی یک لیست عمومی را بنویسید.

```
Traverse ( L : Glist )
{
        if L == null then
                return;
        if L^.Tag == True then
                Write( L^.Data );
        else
                 Traverse ( L^.Dlink );
         Traverse ( L^.Link );
}
```

یک تابع بنویسید که تعداد عناصر لیست عمومی را return کند.

```
Sum (L : List) : integer
{
        if L == Null
                return 0;

        return sum(L^.sublist) + sum(L^.Next) + 1;
}
```


پایان