فرادرس
فراتر از یک کلاس درس
www.faradars.org

فصل پنجم:

# لیست پیوندی

مدرس:

## فرشید شیرافکن

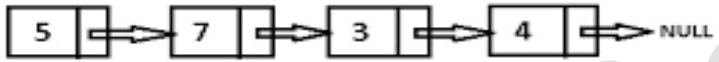
دانشجوی دکتری دانشگاه تهران
(کارشناسی و کارشناسی ارشد: کامپیوتر نرم افزار) (دکتری: بیو انفورماتیک)

# لیست پیوندی

لیست پیوندی، ساختمان داده ای پویا است که اشیاء با یک ترتیب خطی در آن قرار گرفته اند.

بر خلاف آرایه، که در آن ترتیب خطی توسط اندیس های آرایه تعیین می شود، ترتیب در لیست پیوندی بوسیله یک اشاره گر در هر شیء تعیین می‌گردد.

لیست پیوندی بر دو نوع یک طرفه و دو طرفه می باشد.

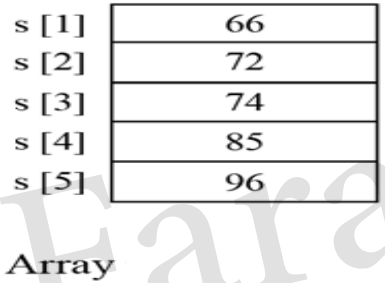


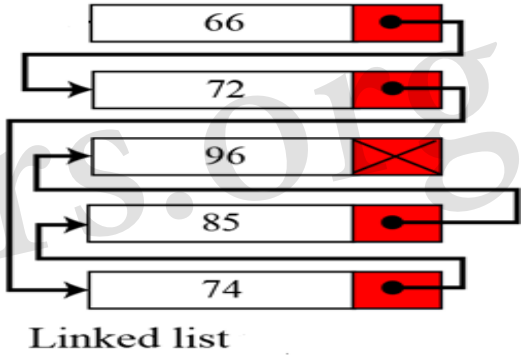

# مقایسه آرایه با لیست پیوندی

۱- دسترسی به عناصر آرایه نسبت به لیست پیوندی سریع تر است، چون اندیس هر عنصر در آرایه مشخص است.

۲- درج و حذف در لیست پیوندی نسبت به آرایه ساده تر است. چون نیاز به جابجایی فیزیکی عناصر در حافظه نمی باشد.

۳- امکان جستجوی دودویی در لیست پیوندی وجود ندارد.

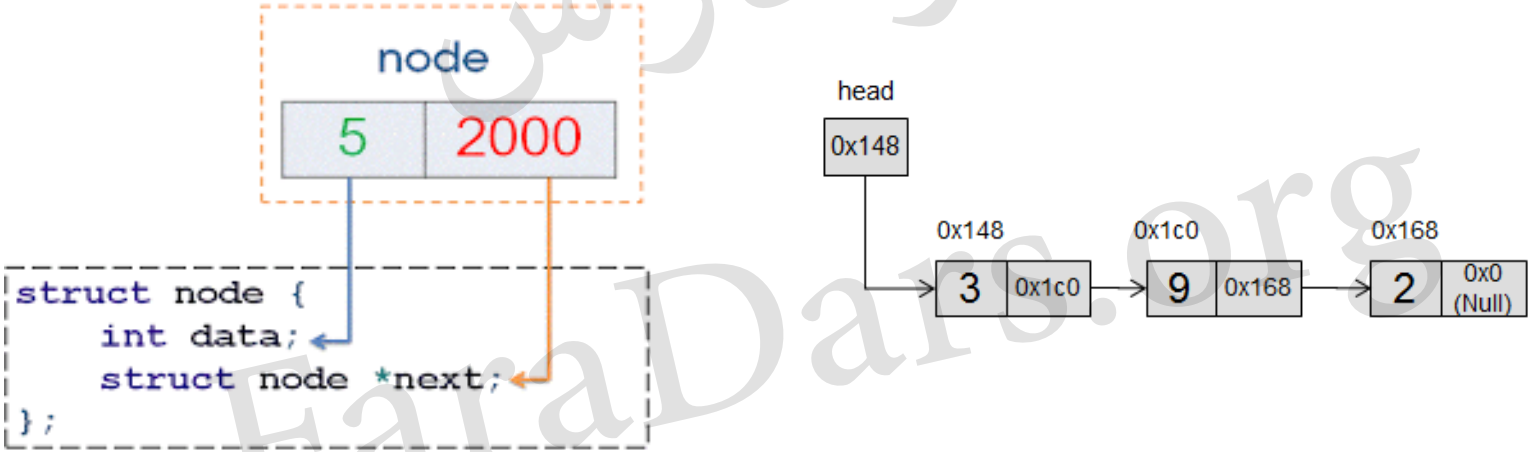


# لیست پیوندی یک طرفه

لیست پیوندی یک طرفه: لیستی که در آن، هر عنصر فقط آدرس عنصر بعدی را نگهداری می کند.

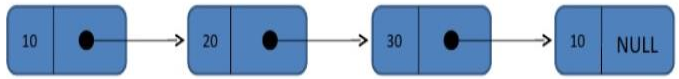
هر یک از گره های این لیست پیوندی از دو قسمت داده و آدرس تشکیل شده است.



# چاپ محتویات لیست

تابع زیر داده های لیست head را چاپ می کند.

```
print-list (head)
{
   p=head;
   for (    ;  p != NULL ;  p = p -> next )
         cout << p -> data;
}
```




تذکر: دو دستور زیر معادل هستند. اولی به زبان C و دومی شبه کد:

```
p = p -> next ;
p = next[p] ;
```

# چاپ محتویات لیست به صورت معکوس

تابع زیر داده های یک لیست با آدرس شروع p را به صورت معکوس چاپ می کند.

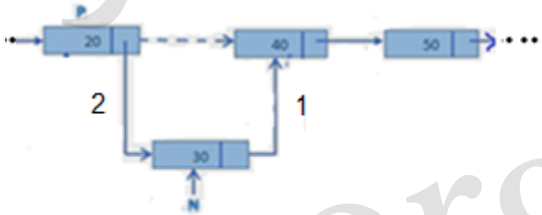
```
f (p)
{
    if(!p) return;
    f(p ->next);
    cout << p -> data;
}
```

Complexity: O(n)

# درج یک گره در لیست پیوندی یک طرفه

تابع زیر یک گره با داده item را بعد از یک گره با آدرس مشخص p درج می کند:

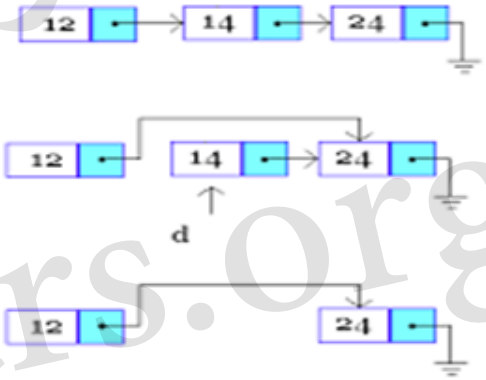
```
insert (head, p, item)
{
    n= malloc (sizeof(node) ) ;
    n -> data = item ;
    n -> next = p -> next;  (1)
    p -> next = n;          (2)
}
```



# حذف یک گره در لیست پیوندی یک طرفه

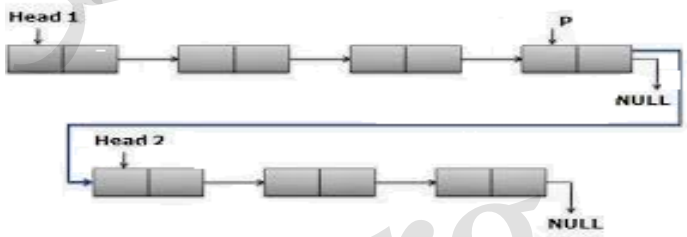
تابع زیر گره با آدرس d را حذف می کند.(گره **d** بعد از گره با آدرس مشخص p قرار دارد)

```
delete( first , p , d)
{
  if (p)
        p-> next = d -> next;
  else
        *first = (*first) -> next;
  free(d);
}
```



# اتصال دو لیست پیوندی

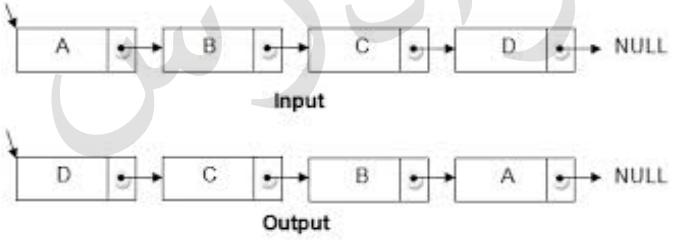
تابع زیر لیست head2 را به انتهای لیست head1 متصل می کند:




```
concate ( head1, head2) {
    if  (head1==NULL)
        return  head2;
    else {
        if (head2!= NULL)
        {
            for (p = head1; p -> next != NULL ; p = p -> next ) ;
            p -> next = head2;
        }
    return  head1;
}}
```

# وارون کردن لیست پیوندی

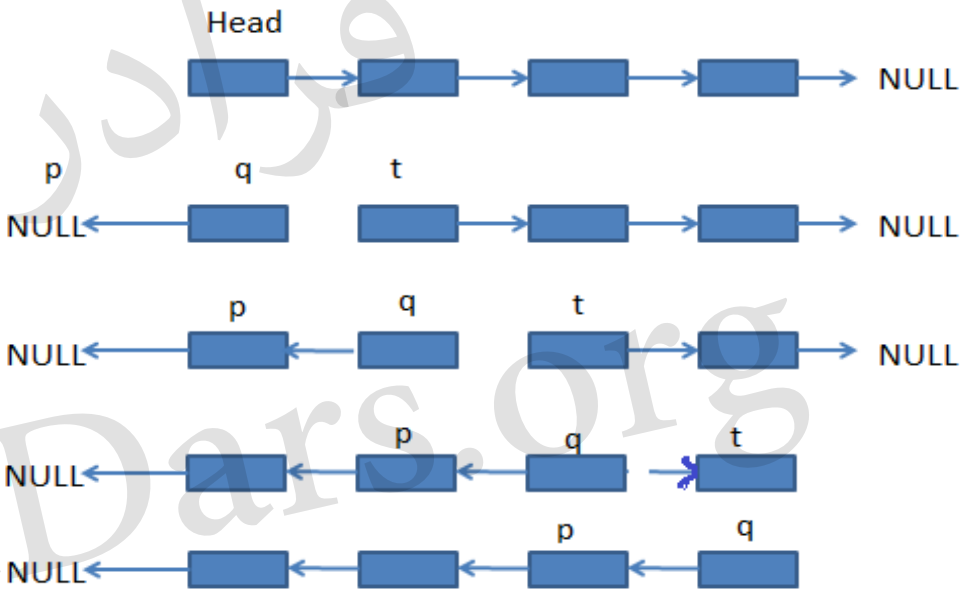
می توان یک لیست n عضوی را تنها با تغییر اشاره گرهای آن وارون کرد.




از سه اشاره گر p ، q و r استفاده می کنیم،که به سه عنصر متوالی اشاره می کنند. ابتدا p تهی است و q به عنصر اول و r به عنصر دوم لیست اشاره می کنند. در هر مرحله، مولفه next عنصر r را به عنصر q تغییر داده و هر سه اشاره گر را به جلو می بریم.
با تهی شدن r ، لیست معکوس شده است.

# تابع وارون

```
reverse(head) {
  if  (size(head)<=1) return head;
  p = null;
  q = head;
  t =  q -> next ;
  while ( t != null)
    {
          q -> next = p;
          p = q;
          q = t;
          t =  t -> next;
    }
  q ->next =  p;
  return   q;
}
```




این کار در زمان خطی انجام می شود.

# لیست پیوندی دوطرفه

در لیستهای پیوندی دو طرفه(دو سویه)، هر عنصر علاوه بر مولفه های لیست یک طرفه ، مولفه prev هم دارد که به عنصر قبلی اش در لیست اشاره می کند.

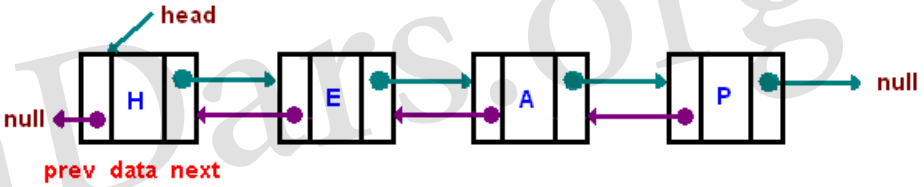
تذکر: به جای Prev از Llink و به جای next از Rlink نیز استفاده می شود.

با استفاده از این اشاره‌گرها امکان حرکت به هر دو طرف وجود دارد.

بنابراین با داشتن آدرس یک گره، به کلیه گره‌های لیست می‌توان دسترسی پیدا کرد.

تعریف نود:

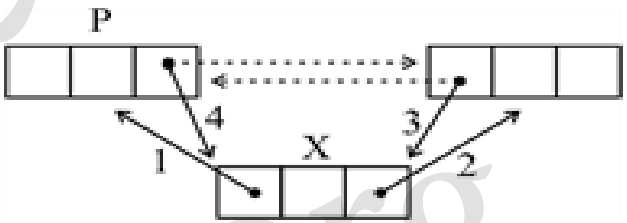
```
struct   node{
        int       data;
        struct  node    *next;
        struct  node    *prev;
};
```



# اضافه کردن گره در لیست پیوندی دو طرفه

اضافه کردن گره x به سمت راست گره با آدرس مشخص p

```
insert ( p , x) {
    x -> prev = p;              (1)
    x -> next = p -> next;      (2)
    p -> next -> prev = x;      (3)
    p -> next = x;              (4)
}
```

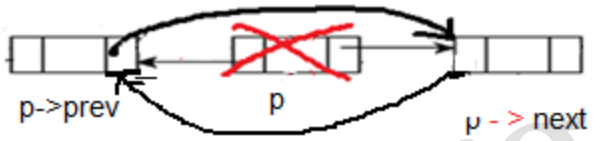



برای اضافه کردن یک گره به یک لیست پیوندی دو طرفه، به ۴ تغییر اشاره گر نیاز است.

# حذف گره از لیست پیوندی دو طرفه

تابع زیر گره با آدرس مشخص p را از لیست پیوندی دو طرفه حذف می کند. برای این کار کافی است اشاره گر next گره قبل از p ، به گره بعد از p اشاره کند و اشاره گر prev گره بعد از p ، به گره قبل از p اشاره کند.

```
delete(p)
{
    p -> prev -> next = p -> next;
    p -> next -> prev = p -> prev;
    free(p);
}
```



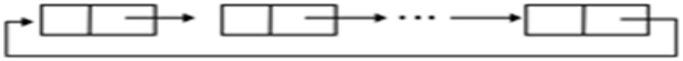

برای حذف یک گره از یک لیست پیوندی دو طرفه، به ۲ تغییر اشاره گر نیاز است.

# لیست پیوندی حلقوی

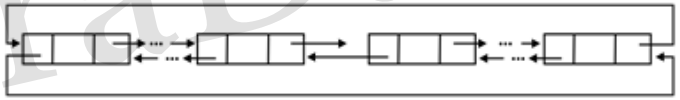
لیست حلقوی(چرخشی)، نوعی لیست یک طرفه است که در آن اشاره‌گر آخرین گره، به جای مقدار دهی با NULL به ابتدای لیست اشاره می‌کند.

در لیست چرخشی با داشتن آدرس هر گره می‌توان به کلیه گره‌ها دسترسی داشت.

مزیت لیست یکطرفه چرخشی نسبت به لیست یکطرفه غیرچرخشی در این است که به گره قبلی دسترسی داریم.

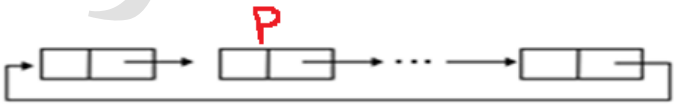


لیست پیوندی دو طرفه حلقوی



# محاسبه طول یک لیست حلقوی

```
length (p){
      c=0;
      if (p) {
         t = p;
         do{
               c++;
               t = t -> next;
        } while(t != p);
      }
   return  c;
}
```



# مسئله ژوزفوس

ژوزفوس یکی از ۴۱ یهودی ای بود که به وسیله ی رومیان در یک غار محاصره شده بودند. به جای تسلیم، این گروه تصمیم گرفتند که همگی خودکشی کنند. آنها قرار گذاشتند تا با شروع از نفر اول و به صورت حلقوی، هر بار نفر دوم زنده ها خود را بکشد و نوبت به نفر زنده ی بعدی برسد، تا این که هیچکس باقی نماند. ولی ژوزفوس زرنگ‌تر از آنها بود و مکان نشستن آخرین فردی را که باید خودکشی می کرد را محاسبه کرد و از ابتدا در آن مکان نشست و جان سالم در برد.

**مسئله در حالت کلی**: اگر n نفر با شماره های 1 تا n دور دایره ای قرار بگیرند و با شروع از شماره ی 1 و در جهت ساعت گرد، هر بار دومین نفر زنده خودش را بکشد، آخرین نفر چه شماره ای دارد؟

ترتیب خودکشی برای n=5:

1 , 2 , 3 , 4 , 5 ⇒ 1 , 3 , 4 , 5 ⇒ 1 , 3 , 5 ⇒ 3 , 5 ⇒ 3

ترتیب خودکشی برای n=6:

1 , 2 , 3 , 4 , 5 , 6 ⇒ 1 , 3 , 4 , 5 , 6 ⇒ 1 , 3 , 5 , 6 ⇒ 1 , 3 , 5 ⇒ 1 , 5 ⇒ 5

# رابطه بازگشتی مسئله ژوزفوس

رابطه بازگشتی مسئله ژوزفوس به صورت زیر است:

$$f(2n) = 2f(n) - 1 \qquad n > 1$$
$$f(2n+1) = 2f(n) + 1 \qquad n > 1$$
$$f(1) = 1$$

جواب این رابطه : $2(n-k)+1$ که $k = 2^{\lfloor \log n \rfloor}$ .

مثال: مقدار f(10) را به کمک رابطه بازگشتی بالا محاسبه کنید.

$$f(10) = 2f(5) - 1 = 2 \times 3 - 1 = 5$$
$$f(5) = 2f(2) + 1 = 2 \times 1 + 1 = 3$$
$$f(2) = 2f(1) - 1 = 2 \times 1 - 1 = 1$$

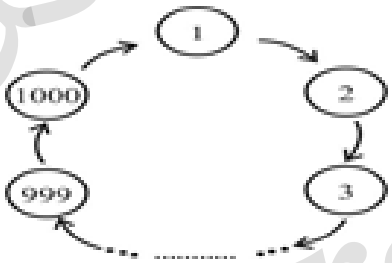
روش دوم:

اگر در جواب رابطه به ازای n مقدار 10 و به ازای k مقدار 8 (بزرگترین عدد توان دو کوچکتر از 10) را قرار دهیم، جواب 5 خواهد شد.

# مثال

با فرض اجرای تابع f بر روی لیست پیوندی حلقوی شکل زیر، مقدار خروجی چقدر است؟

```
int   f (L) {
   if (L -> next == L)
         return   L -> data;
   L -> next = L -> next -> next;
   return   f (L -> next);
}
```




حل:

با قرار دادن 1000 به جای n و 512 (بزرگترین عدد توان دو کوچکتر از 1000 ) به جای k در رابطه $2(n-k)+1$ ، مقدار 977 حاصل می شود.

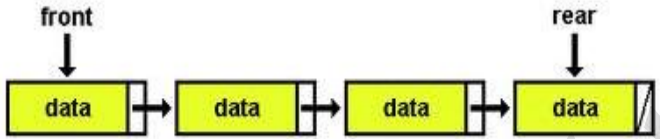
# پیاده سازی پشته به کمک لیست پیوندی


پشته ، لیستی است که درج و حذف از ابتدای آن انجام می گیرد.

```
add(top , item)
{
   n = malloc(sizeof (stack)) ;
   n -> data= item ;
   n -> next = top ;
   top = n ;
}
```

```
delete(top ) {
    d = top ;
    if  (d ==NULL)  exit ( ) ;
    else {
            item = d -> data ;
            top = d -> next ;
            free(d) ;
            return  item ;
} }
```

# پیاده سازی صف به کمک لیست پیوندی




صف: لیستی که درج به انتهای آن و حذف از ابتدای آن انجام می گیرد.
اولین گره لیست، عنصر ابتدای صف است.

```
addq(front , rear, item) {
    n = malloc(sizeof(queue));
    n -> data= item;
    n ->next = NULL;
    if (front)
            rear -> next = n ;
    else
            front = n ;
    rear = n ;
}
```

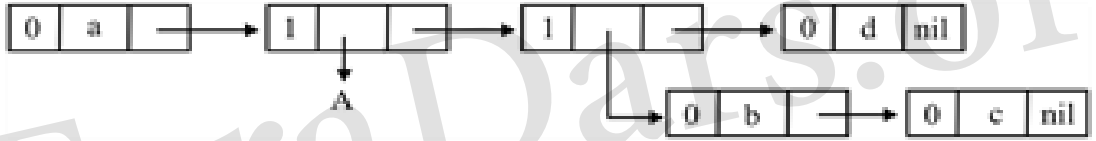
```
deleteq(front)
{

    if ( front == NULL)
            exit( );
    d = front ;
    item = d -> data ;
    front = d -> next ;
    free(d);
    return  item;
}
```

# لیست عمومی

لیست عمومی، رشته محدودی از n عنصر است، به نحوی که هر عنصر، یک اتم یا یک لیست است.
اتم ها با حروف کوچک و لیست ها با حروف بزرگ نمایش داده می‌شوند.
هر نود لیست عمومی از سه قسمت تشکیل شده است:
قسمت اول یک فلگ است. اگر این فلگ صفر باشد، قسمت دوم، یک اتم خواهد بود. اگر فلگ یک باشد، قسمت دوم، اشاره‌گر به یک لیست است. قسمت سوم برای اشاره به نود بعدی استفاده می‌شود.

مثال:

**L=(a , A , (b,c) , d)**




head : عنصر اول

tail : همه عناصر به غیر از عنصر اول

# تمرین

۱- الگوریتم وارونه کردن لیست پیوندی یک طرفه را به صورت بازگشتی بنویسید.

۲- الگوریتمی بنویسید که عناصر تکراری در لیست پیوندی یک طرفه را حذف کند.

۳- در مسئله ژوزفوس اگر هر بار **k** امین نفر زنده خودکشی کند، رابطه بازگشتی چگونه خواهد شد.

۴- الگوریتمی بنویسید که لیست پیوندی یک طرفه را به صورت **k** تایی معکوس کند. به طور نمونه با فرض **k=2** لیست **1,2,3,4,5,6** را به شکل **2,1,4,3,6,5** تبدیل کند.

# تمرین

۵-عملکرد تابع زیر چیست؟

```
what(head) {
    prev = NULL;
    last =head ;
    while ( last ->next != NULL) {
        prev = ptr ;
        last = last ->next ;
    }
    prev ->next = NULL ;
    last -> next = head ;
    head = last ;
}
```

**این اسلاید ها بر مبنای نکات مطرح شده در فرادرس**
**«مجموعه فرادرس های ساختمان داده ها»**
**تهیه شده است.**

**برای کسب اطلاعات بیشتر در مورد این آموزش به لینک زیر مراجعه نمایید.**

**faradars.org/fvds9402**