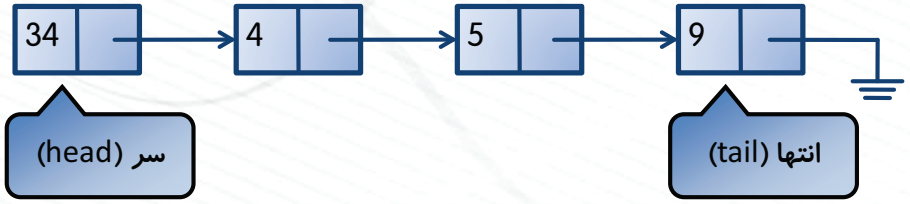
فهرست پیوندی (Linked List)

قاسم مهدور (mahdevar@ibb.ut.ac.ir)

فهرست پیوندی(linked list)

- یک فهرست پیوندی مجموعه‌ای از گره‌ها (nodes) است و
  - گره‌ها آن در حافظه مجاور هم نیستند،
  - حاوی عناصری جهت اشاره (point) به گره‌های بعدی (successor) است، و
  - زمان خطی اضافه نمودن و حذف نمودن وجود ندارد!!! (چرا و چه وقت؟)



فهرست مطالب

نحوه ذخیره سازی فهرست پیوندی در حافظه

مقدار	موقعیت حافظه
34	2342
2346	2344
4	2346
4200	2348
.	.
.	.
.	.
5	4200
5126	4202
.	.
.	.
.	.
9	5126
0	5128

- در این مثال برای ذخیره سازی
  - اعداد دو بایت و
  - اشاره‌گرها نیز دو بایت

  نیاز داریم (همگی integer هستند).

- با داشتن کدام یک از آدرس‌های حافظه می‌توان از فهرست پیوندی استفاده کرد؟
- در یک کامپیوتر با حافظه 256MB هر اشاره‌گر چند بایت است؟

پیاده سازی ساختار گره در C

```
struct Node
  {
      int Data;
      .
      .
      .
      Node *Next;
  };
void main()
  {
      Node A, B;
      A.Next=&B;
  }
```

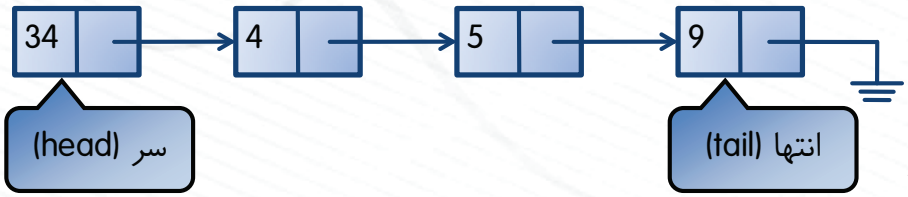
هر گونه و هر تعداد داده‌ای در یک گره می‌تواند باشد.

از این عنصر ساختار جهت اشاره به عضو بعدی فَهرست استفاده می‌شود.

الگوریتم درج گره جدید در فهرست پیوندی

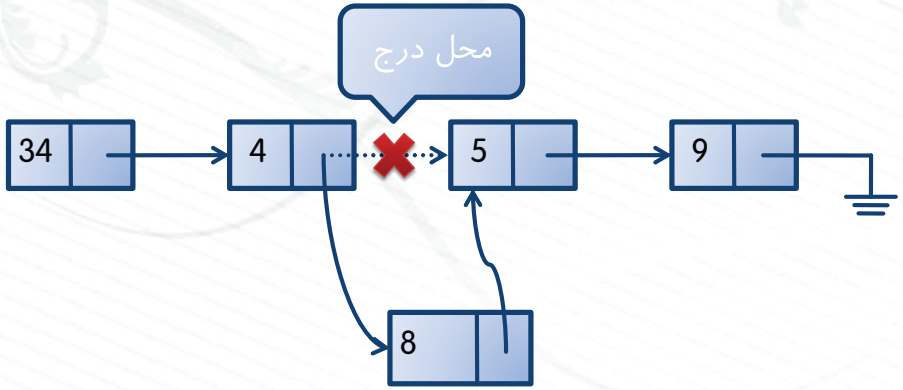
```
void Insert(Node *P, int a)
  {
      Node *t = new Node;
      t->Data = a;
      t->Next = P->Next;
      P->Next=t;
  }
```

- به ازای کدام گره‌ها این الگوریتم درست عمل می‌کند؟
  پاسخ: گره قبل از محل درج



چگونگی درج گره جدید در فهرست پیوندی

- برای اضافه نمودن گره‌ای جدید باید
  1. گره‌ای جدید ساخت، به داده‌های آن مقدار داد،
  2. اشاره‌گر گره جدید به گره بعد از محل درج اشاره کند،
  3. اشاره‌گر گره قبل از محل درج به گره جدید اشاره کند.



نحوه حذف یک گره از فهرست پیوندی

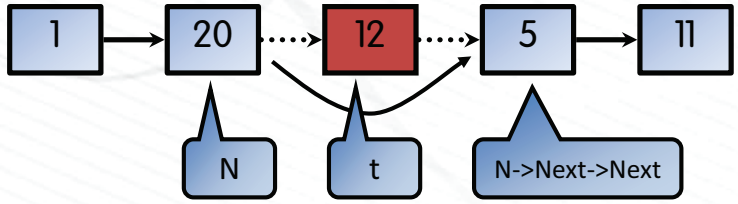
- تابع حذف اشاره‌گر گره قبل از محل حذف را باید داشته باشد.
- برای حذف یک گره باید
  - نشانی گره مورد نظر را ذخیره نمود،
  - گره قبلی به گره بعدی اشاره کند
  - گره از حافظه پاک شود.

الگوریتم حذف یک عنصر از فهرست پیوندی

```
void Delete(Node *N)
{
    Node *t;
    t = N->Next;
    N->Next = N->Next->Next;
    free(t);
}
```

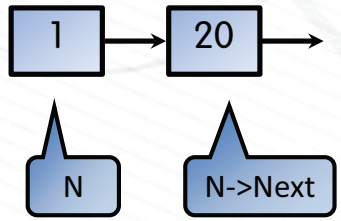
این الگوریتم نمی‌تواند گره اول را حذف کند.

دقت شود که t->Next همان N->Next->Next است!



الگوریتم حذف گره اول از یک فهرست پیوندی

```
void DeleteFirst(Node *N)
{
    Node *t;
    t = N;
    N=N->Next;
    free(t);
}
```



مثال: برنامه‌ای برای برای ذخیره سازی و بازیابی ۱ تا ۱۰۰

```
#include <stdio.h>
#include <stdlib.h>
struct Node
{
   int Data;
   Node *Next;
};
void Insert(Node *P, int a)
{
   Node *t = new Node;
   t->Data = a;
   t->Next = P->Next; P->Next=t;
}
void Delete(Node *N){
   Node *t;
   t = N->Next;
   N->Next = N->Next->Next;
   free(t);
}
```


مثال: برنامه‌ای برای برای ذخیره سازی و بازیابی ۱ تا ۱۰۰

```
void main(){
   Node First;
   Node Last;
   Node *t;
   First.Data=1;
   First.Next=&Last;
   Last.Next=NULL;
   t=&First;
   for(int i=2; i<100; i++){
       Insert(t, i);
       t=t->Next;
   }

   t=&First;
   while(t->Next!=NULL){
       printf(" %d", t->Data);
       t=t->Next;
   }
}
```

جمع بندی

- نشانی گره اول را باید داشت و از طریق آن می‌توان به باقی گره‌ها دسترسی پیدا نمود.
- برای اضافه و حذف نمودن احتیاجی به جابجای اعضا در حافظه نیست.
- حافظه هدر نمی‌رود.

یافتن عضو k ام در یک فهرست پیوندی

```
Node *Retrieve (Node *N, int k)
{
  int i;
  Node *t=N;
  for(i=0; i<k; i++)
  {
     t=t->Next;
  }
  return t;
}
```


الگوریتم درج عضوی جدید بعد از عضو k ام

```
void InsertAt(Node *N, int i, int Val)
{
  Node *t;
  t=Retrieve (N, i);
  Insert(P, Val);
}
```


الگوریتم حذف عضو k ام

```
void DeleteFrom(Node *N, int i)
{
  Node *t;
  t=Retrieve (N, i - 1);
  Delete(t);
}
```

یافتن عضوی خاص در یک فهرست پیوندی

```
Node *Find(Node *N, int k)
{
  Node *t=N;
  while (t->Data != k && t!=NULL)
  {
    t=t->Next;
  }
  return t;
}
```


فهرست پیوندی – زمان مورد نیاز اعمال مختلف

1. ایجاد (Create) ← O(1)
2. چاپ (Print)، ← O(n)
3. اضافه نمودن یک عضو (Insert)، ← O(1) *
4. اضافه نمودن یک عضو (InsertAt)، ← O(n)
5. حذف یک عضو (Delete)، ← O(1) *
6. حذف یک عضو (DeleteFrom)، ← O(n)
7. جستجو برای یافتن یک عضو (Find)، ← O(n)
8. یافتن عضو بعدی (Next)، ← O(1)
9. یافتن عضو قبلی (Previous)، و ← O(n) **
10. یافتن عضو K ام (Retrieve). ← O(n)

* البته باید موقعیت گره را داشته باشیم!!!

** آیا می‌توان آن را به O(1) کاهش داد؟ چگونه؟

چاپ فهرست پیوندی (تابع Print)

```
void Print(Node *P){
  Node *t;
  t=P;
  while(t){
    printf(" %d", t->Data);
    t=t->Next;
  }
}
```

حالت فشرده t != NULL است.

این تابع را می‌توان اینگونه نیز نوشت

```
void Print(Node *P){
  while(P){
    printf(" %d", P->Data);
    P=P->Next;
  }
}
```


زمان اجرای اعمال مختلف روی آرایه و فهرست پیوندی

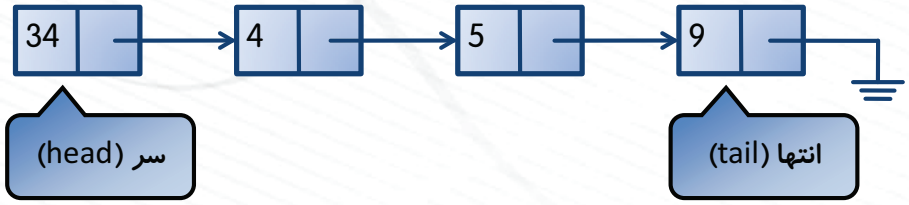
عمل	آرایه	فهرست پیوندی
Create	O(n)	O(1)
Print	O(n)	O(n)
Insert	O(n)	O(1)
InsertAt	O(n)	O(n)
Delete	O(n)	O(1)
DeleteFrom	O(n)	O(n)
Find	O(n)	O(n)
Next	O(1)	O(1)
Previous	O(1)	O(n) [↓O(1)]
Retrieve	O(1)	O(n)

روش‌های بهبود فهرست پیوندی

- اشاره‌گر به انتها (Tail Pointer)
- فهرست حلقوی (Circular Lists)
- فهرست دو پیوندی حلقوی (Doubly Linked Circular List)

اشاره‌گر به انتها (Tail Pointer): مزیت

- معمولاً ما به انتهای یک فهرست عضوی جدید می‌افزاییم؛
- با داشتن اشاره‌گر به آخرین گره می‌توان برای افزودن به انتهای فهرست از تابع Insert به جای InsertAt استفاده نمود و پیچیدگی را از O(n) به O(1) کاهش داد.



اشاره‌گر به انتها (Tail Pointer): پیاده سازی تابع Insert

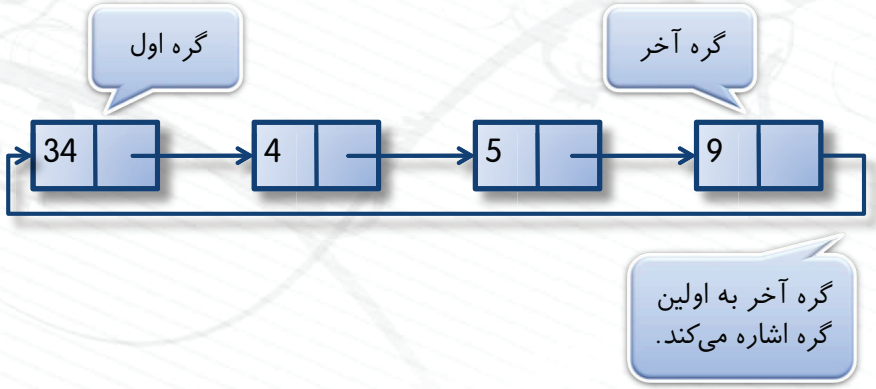
```
void Insert(Node *P, int a)
{
  Node *t = new Node;
  t->Data = a;
  t->Next = P->Next;
  P->Next=t;
  if(Tail==P)
     Tail=t;
}
```


اشاره‌گر به انتها: پیاده سازی تابع Delete

```
void Delete(Node *N)
 {
    Node *t;
    t = N->Next;
    N->Next = N->Next->Next;
    if(Tail==t)
         Tail=t;
    free(t);
 }
```

فهرست حلقوی (Circular Lists)

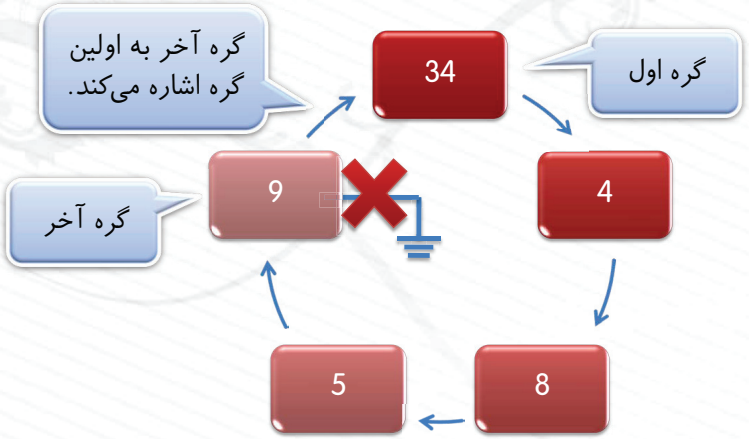
- در این نوع فهرست آخرین گره (که به NULL اشاره می‌کرد) را به اولین گره متصل می‌کنیم.



- مزیت: با داشتن نشانی هر گره می‌توان تمام فهرست را پیمود.

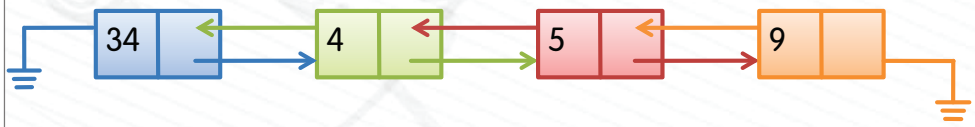
فهرست حلقوی: تغییرات

- تغییرات در پیاده سازی:
  1. دانستن اینکه آیا گره آخرین گره است؟
- پاسخ:
- `if(P->Next==First)`



فهرست دو پیوندی (Doubly Linked List)

- در این نوع فهرست پیوندی
  1. هر گره دو اشاره‌گر دارد، یکی به گره بعدی و دیگری به گره قبلی، و
  2. ماقبل گره اول و مابعد گره آخر NULL است.



پیاده سازی گره فهرست دو پیوندی در C

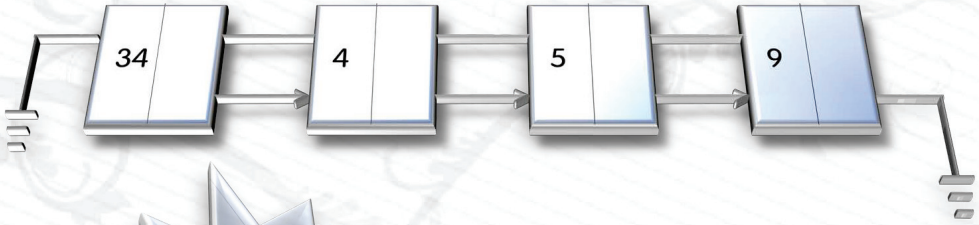
```
struct Node
{
    int Data;
    .
    .
    .
    Node *Next;
    Node *Previous;
};
```

هر گونه و هر تعداد داده‌ای در یک گره می‌تواند باشد.

از این عنصرِ ساختار جهت اشاره به عضو بعدی در فهرست استفاده می‌شود.

از این عنصرِ ساختار جهت اشاره به عضو قبلی در فهرست استفاده می‌شود.

توابع Delete و Insert یک فهرست دو پیوندی

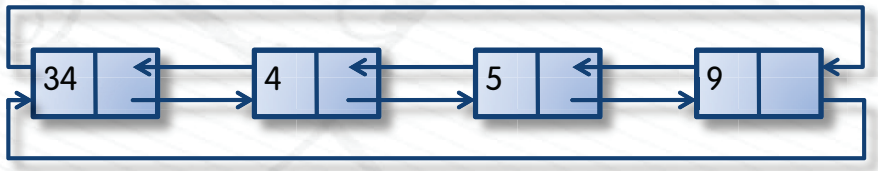


- راهنمایی: Insert و Delete را به گونه‌ای باز نویسی کنید که علاوه بر اشاره‌گرهای Next اشاره‌گرهای Previous گره‌ها نیز همواره معتبر باشند.

29

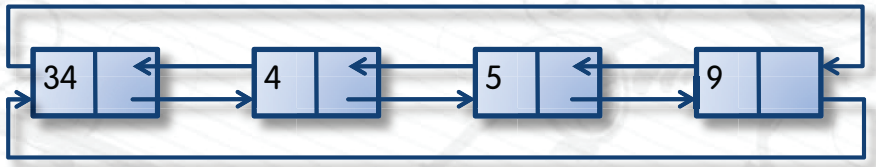
فهرست دو پیوندی حلقوی (Doubly Linked Circular List)

- در این نوع فهرست دو پیوندی گره قبل از گره اول، گره آخر است و گره بعد از گره آخر همان گره اول است.



30

توابع Delete و Insert یک فهرست دو پیوندی حلقوی



- راهنمایی: Insert و Delete را به گونه‌ای باز نویسی کنید که علاوه بر اشاره‌گرهای Next اشاره‌گرهای Previous نیز همواره معتبر باشند.

31