فصل چهارم: لیست ها

بخش ها

- لیست پیوندی
- لیست عمومی
- لیست حلقوی

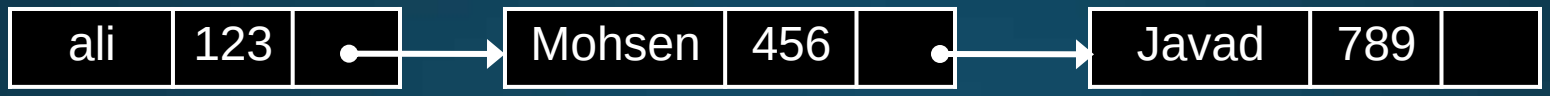
لیستهای پیوندی(Linked List)

- مشابه آرایه ها بوده با این تفاوت که عناصر آرایه در حافظه الزاما پشت سرهم قرار گرفته اند اما عناصر لیست پیوندی از نوع پویا بوده و الزاما در کنار یکدیگر نمی باشند
- در مقایسه با آرایه، اعمال درج و حذف در لیست پیوندی آسان تر و سریعتر است اما جستجو یا مرتب سازی ممکن است کندتر باشد.
- هر عنصر یا گره (Node) در لیست پیوندی حداقل از دو بخش داده(Data) و آدرس گره بعدی(اشاره گر)(Link) تشکیل شده است.

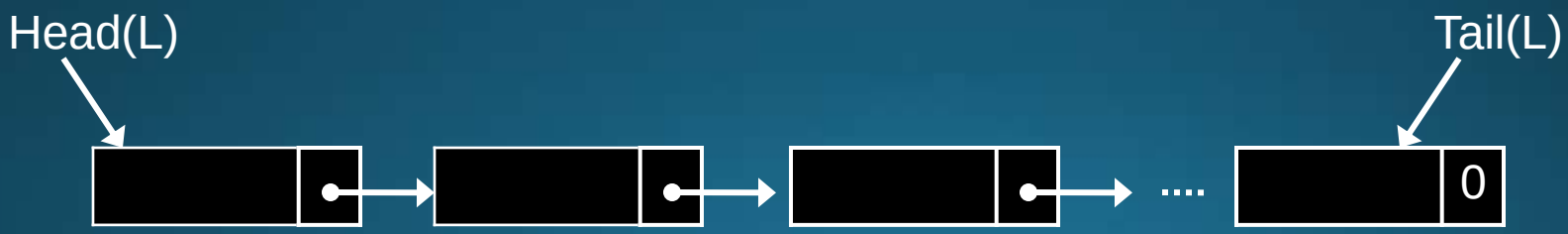
مثالی از لیستهای پیوندی

ali	123
Mohsen	456
Javad	789

- می خواهیم آرایه روبرو را به لیست تغییر دهیم



- دو متغیرhead و tail : head اشاره گر به عنصر ابتدایی وtail اشاره گر به عنصر انتهایی لیست است.



تعریف لیست پیوندی

برنامه ۱-۴

```
struct node {
      int data;
      node *next;
 };
class list {
       private:
              node *head, *tail;
       public:
              list() {
                     head=NULL;
                     tail=NULL;
               }
 };
```

نمایش لیست پیوندی

برنامه ۱-۴

```
void display()
   {
     node *temp=new node;
     temp=head;
     while(temp!=NULL)
     {
       cout<<temp->data<<"\t";
       temp=temp->next;
     }
   }
```

اضافه کردن گره (در انتهای) لیست پیوندی

برنامه ۱-۴

```
void createnode(int value)
    {
      node *temp=new node;
      temp->data=value;
      temp->next=NULL;
      if(head==NULL)
      {
        head=temp;
        tail=temp;
        temp=NULL;
      }
      else
      {
        tail->next=temp;
        tail=temp;
      }
    }
```

اضافه کردن به ابتدای لیست پیوندی

برنامه ۱-۴

```
void insert_start(int value)
  {
    node *temp=new node;
    temp->data=value;
    temp->next=head;
    head=temp;
  }
```

اضافه کردن در جایگاه مشخص لیست پیوندی

برنامه ۱-۴

```
void insert_position(int pos, int value)
  {
    node *pre=new node;
    node *cur=new node;
    node *temp=new node;
    cur=head;
    for(int i=1; i<pos; i++)
    {
      pre=cur;
      cur=cur->next;
    }
    temp->data=value;
    pre->next=temp;
    temp->next=cur;
  }
```

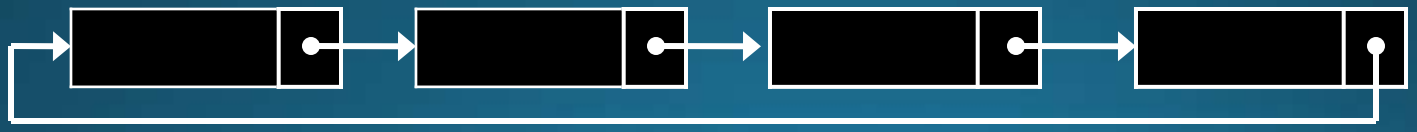
جستجو در لیست پیوندی

برنامه ۱-۴

```
void search(int s)
        {
                node *temp=new node;
                temp=head;
                int i=0;
                bool flgFound=false;
                while(temp!=NULL&&!flgFound)
                {
                        i++;
                        if(temp->data==s)
                        {
                                flgFound=true;
                        }
                        temp=temp->next;
                }
                if(flgFound)
                        cout<<"Found at "<<i<<"\n";
                else
                        cout<<"Not found"<<"\n";
        }
```

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.



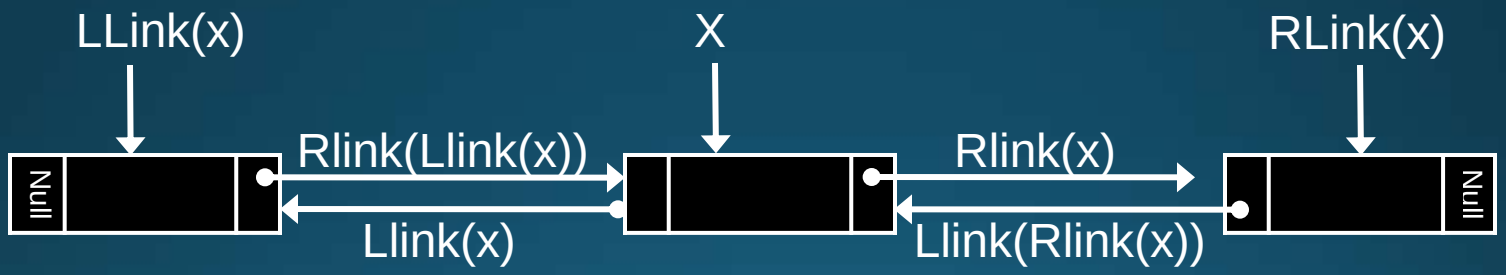
لیست پیوندی دو طرفه

- 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 کزد

لیست پیوندی دو طرفه

- در این لیست رابطه زیر همواره برقرار است:

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



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 راست y:

Llink(Rlink(y)) = x

Rlink(x) = Rlink(y)

Rlink(y) = x

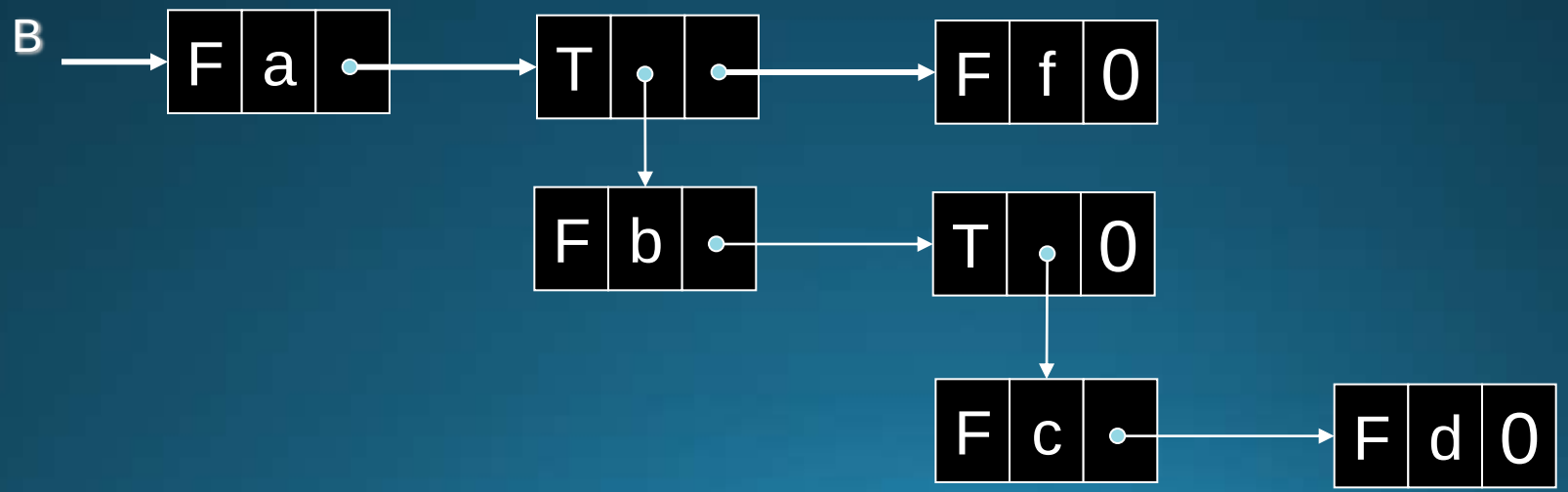
Llink(x) = y

لیستهای عمومی General List

- لیستی که هر گره آن بتواند خود یک زیرلیست باشد

Typetag {True,False}	Data	Link
	DLink	

B=(a,(b,(c,d)),f)



نمایش لیست عمومی(بازگشتی)

```
void display(node *temp)
{
    if(temp!=NULL)
    if(! temp->tag)
        cout<<temp->data<<"\t";
    else
        display(temp->data)
    display(temp->link)
  }
```

کاربرد لیست پیوندی

- پیاده سازی پشته
- Push

```
a=new(stack);
a->data=temp;
a->link=top;
top=a;
```

- Pop

```
a=top->link;
temp=top->data;
top->link=NULL;
delete  top;
top=a;
```

کاربرد لیست پیوندی

- پیاده سازی صف
- ADD

```
queue * p;
  p=new (queue);
  p->data =temp;
  p->link=null;
  if (front ==null)
    front=p;
  else
     rear->link=p;
```