DOUBLY LINKED LISTS

EMAN T.MAHDI
COLLEGE OF C.S. &I.T.

OVER VIEW

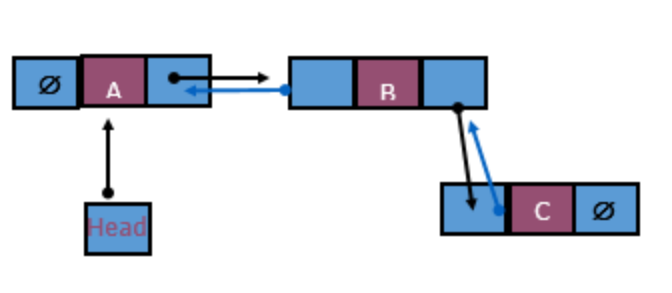
- Doubly linked lists
- General definition type
- Create first node
- Add new nodes
- Display double linked list by different manners
- Compute no. of odd number in D.L.L.
- Delete node form D.l.l. depending on value x.
- Insert node before specific information.
- Insert node after specific information

DOUBLY LINKED LISTS

Each node points to not only successor but the predecessor.

There are two NULL: at the first and last nodes in the list

Advantage: given a node, it is easy to visit its predecessor. Convenient to traverse lists backwards

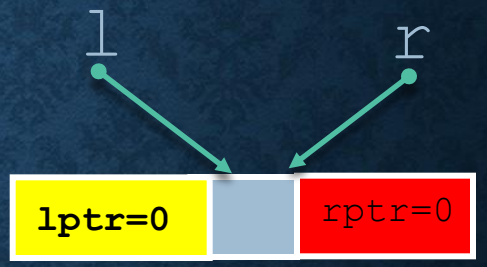


GENERAL DEFINITION TYPE

```
#include<iostream.h>
#define n 5
struct nod{
int      info;
nod    *rptr;
nod    *lptr;
 };
 nod *r,*l,*p,*q;
```

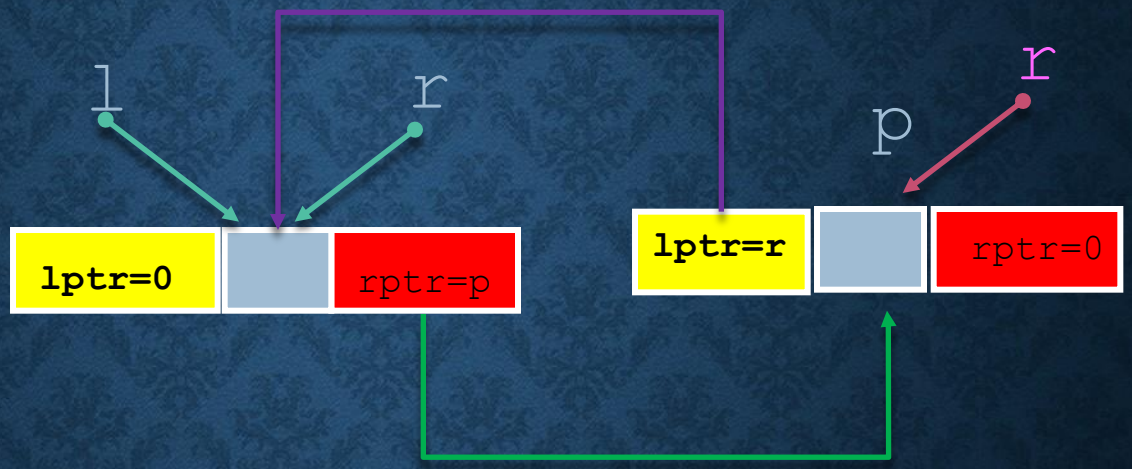
CREATE FIRST NODE

```
q=new nod;
cout<<" enter the information :";
cin>>q->info;
q->rptr  = q->lptr =0;
l=r=q;
```



ADD NEW NODES

```
for(int i=1;i<n;i++)
 { q=new nod;
  cin>>q->info;
  q->lptr=r;
  q->rptr=0;
  r->rptr=q;
  r=q;
 }
```



DISPLAY DOUBLE LINKED LIST BY DIFFERENT MANNERS

```
p=l;
cout<<"\n The DLL from left to right : ";
while(p!=0)
{  cout<<p->info;      p=p->rptr;   }
p=r;
cout<<"\n The DLL from right to left: ";
while(p!=0)
 { cout<<p->info;   p=p->lptr; }
```

COMPUTE NO. OF ODD NUMBER IN D.L.L.

```
p=l;
int od=0;
 while(p!=0)
 {
            if(p->info%2!=0)  od++;
            p=p->rptr;
 }
 cout<<"\n the no. of odd =  "<<od;
```

COMPUTE NO. OF PRIME NUMBER IN D.L.L.

```
p=l;  int prime=0;
while(p!=0)
{ f=0;
 for(int j=2;j<p->info;j++)
            if(p->info%j==0) f=1;
            if(f==0) prime++;
           p=p->rptr;
}
cout<<"\n the no. of prime =  "<<prime;
}
```

DELETE NODE FORM D.L.L. DEPENDING ON VALUE X

```
p=l;
while(p->info!=x)     p=p->rptr;
if(p==l)  { l=l->rptr;
            l->lptr=0;
            p->rptr=0;
            delete p;  }
else  if(p==r) { r=r->lptr;
            r->rptr=0;
           p->lptr=0;
           delete p; }
 else { p->lptr->rptr=p->rptr;
        p->rptr->lptr=p->lptr;
        delete p; }
```

INSERT NODE BEFORE SPECIFIC INFORMATION

```
p=l;
  while(p->info!=x)    p=p->rptr;
if(p->info==x)
{
 t=new nod ;
cin>>t->info;
  t->rptr=t->lptr=0;
        t->rptr=l;
         l->lptr=t;
          l=t;
     }
else cout<<“not found”;
```

INSERT NODE AFTER SPECIFIC INFORMATION

```
p=l;
  while(p->info!=x)    p=p->rptr;
if(p->info==x)
{ t=new nod ;
cin>>t->info;
  t->rptr=t->lptr=0;
t->lptr=r;
r->rptr=t;
r=t;}
else cout<<“not found”;
```