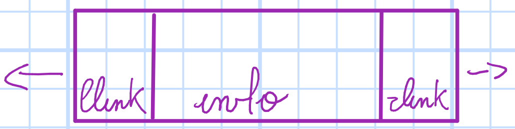
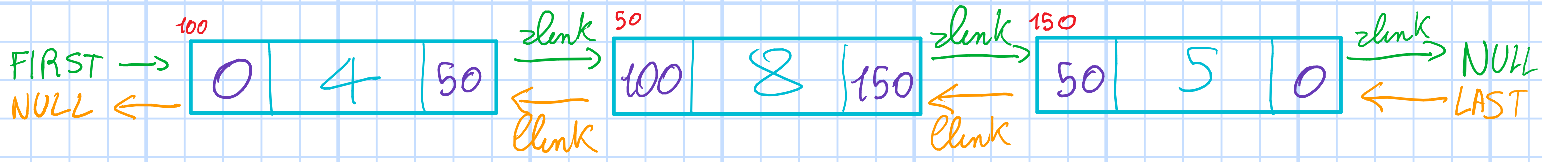
Liste dublu înlănțuite

Fiecare element



rlink | adresa următorului element
llink | adresa elementului precedent



```
struct NOD2 {
    T info;
    NOD2* rlink;
    NOD2* llink;
}
NOD2* FIRST, *LAST;
```

OPERAȚII DE BAZĂ

1) Accesarea unui element

↳ Căutăm elementul cu info "a" în listă } → Funcția returnează adresa elementului găsit sau NULL, în caz contrar

De la început

```
iter = FIRST;
while iter != NULL && iter->info != a {
        iter = iter -> rlink;
}
return iter;
```

De la sfârșit

```
iter = LAST ;
while iter != NULL && iter->info != a {
        iter = iter -> llink;
}
return iter;
```

2) Inserarea unui element

2.1) La început → Alocă zonă de memorie p= new NOD2;

```
if p=NULL {
    COUT<<"OVERFLOW!"; exit(0);
}
p->info = a;
p-> llink = NULL;
p-> rlink = FIRST;
FIRST = P;
```

```
→ if (LAST == NULL) {
      LAST = P;
  }
```

2.2) La sfarșit → Alocă zonă de memorie p= new NOD2;

```
if p=NULL {
     COUT<<"OVERFLOW!";
     exit(0);
}
p->info=a;
p->rlink =NULL;
p->llink=LAST;
LAST->rlink=p;
LAST=p;
if (FIRST==NULL) {
     FIRST=P;
}
```

2.2) După un element dat q

```
if (q==NULL){
    cout << "POZ INVALIDĂ"; exit(1);
}
if (q==LAST){
    insert_LAST(a);
} else {
    if (p==NULL) {
        COUT<<"OVERFLOW!";
        exit(0);
    }
    p->info=a;
    p->rlink= q->rlink;
    p->llink=q;
    p->rlink->llink=p;
    q->rlink=p;

}
```

3) Stergerea unui element

3.1) Primul element

```
if (FIRST==NULL && LAST==NULL) {
        cout << "UNDERFLOW";
        exit(0);
}
temp = FIRST;
FIRST = temp ->rlink;
temp->rlink->llink = NULL;
cout << temp ->info<<endl;
delete temp;
if (FIRST==NULL){
     LAST=NULL;
}
```

3.2) Ultimul element

```
if (FIRST==NULL && LAST==NULL) {
        cout << "UNDERFLOW";
        exit(0);
}
temp = LAST ;
LAST = temp ->llink ;
temp->llink->rlink =NULL;
cout << temp ->info<<endl;
delete temp;
if (LAST ==NULL){
     FIRST=NULL;
}
```

3.3) un element dat q

```
if (q==NULL){
    cout << "poz INVALIDĂ";
    exit(1);
}
if (FIRST==NULL && LAST==NULL){
     cout<< "UNDERFLOW";
     exit ;
}
if (q==FIRST){
     delete_first ();
}else if (q==LAST){
     delete_LAST();
}else{
     q->llink->rlink = q->rlink;
     q->rlink->llink = q->llink;
     cout << q->info;
     delete q;
}
```

GRAFURI

- Neorientate

↳ Matricea de adiacență → $A=(a_{ij})$;
n= numar de varfuri.

$a_{ij}=\begin{cases}1 & \text{dacă } (i,j)\in E\\ 0 & \text{altfel}\end{cases}$ (i,j) = muchie

$a_{ij}=a_{ji}$ pentru grafuri neorientate!