ФАКУЛТЕТ ЗА ЕЛЕКТРОТЕХНИКА И ИНФОРМАЦИСКИ ТЕХНОЛОГИИ

Единечно поврзани листи

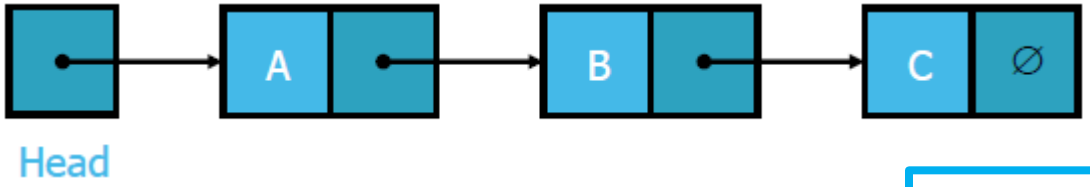
– Податочни структури и програмирање –

Низи vs. поврзани листи

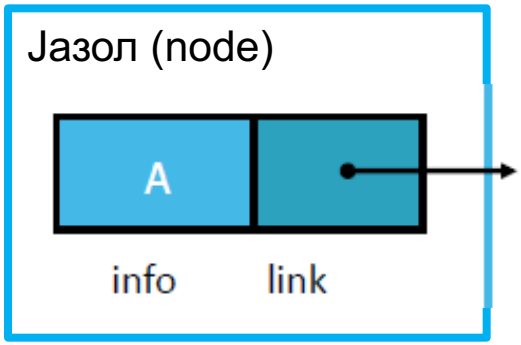
- **Низи:**
  - **+** Брз пристап до елементи
  - **–** Не може да им се менува големината
  - За да додадеме/избришеме елемент во низа, треба да ги поместуваме елементите
- **Поврзани листи:**
  - **– Покомплексни структури** од низите
  - **+** Лесно може да им се менува големината (зголемува или намалува) – **динамички структури**
    - Нема потреба да знаеме колку јазли ќе има во листата. Јазлите се креираат во меморија по потреба (ако има место)
  - **+** Лесно и **брзо додавање/бришење** на елемент
    - Со поврзана листа, нема потреба да движиме други јазли. Само треба да се ресетираат некои покажувачи (врски)

Единечно поврзана листа (1)

- **Апстрактен податочен тип**
  - Податочна структура во која секој елемент динамички се алоцира
  - Елементите покажуваат еден кон друг
- Поврзаната листа е **множество од поврзани јазли**

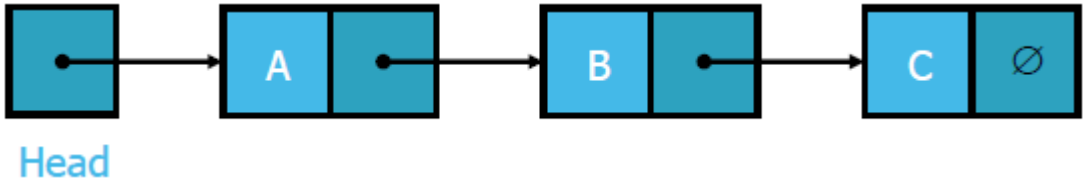


- **Секој јазол содржи**
  - Податок
  - Покажувач кон следниот јазол во листата
- **Head**: покажувач кон првиот јазол
- **Последниот јазол** покажува на **NULL**



Единечно поврзана листа (2)

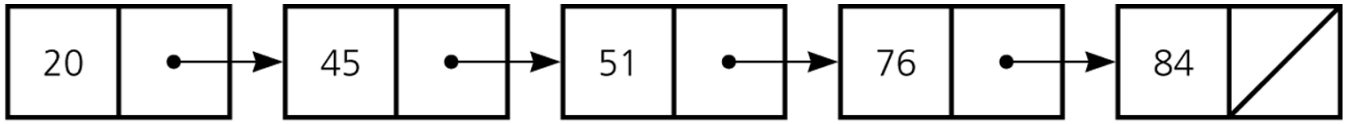
- **Основни операции:**
  - Креирање
  - Вметнување;
  - Наоѓање;
  - Бришење;
  - Печатење; итн.
- **Варијации на поврзани листи**:
  - Двојно-поврзани листи
  - Кружни поврзани листи (единечни или двојни)



Основни операции

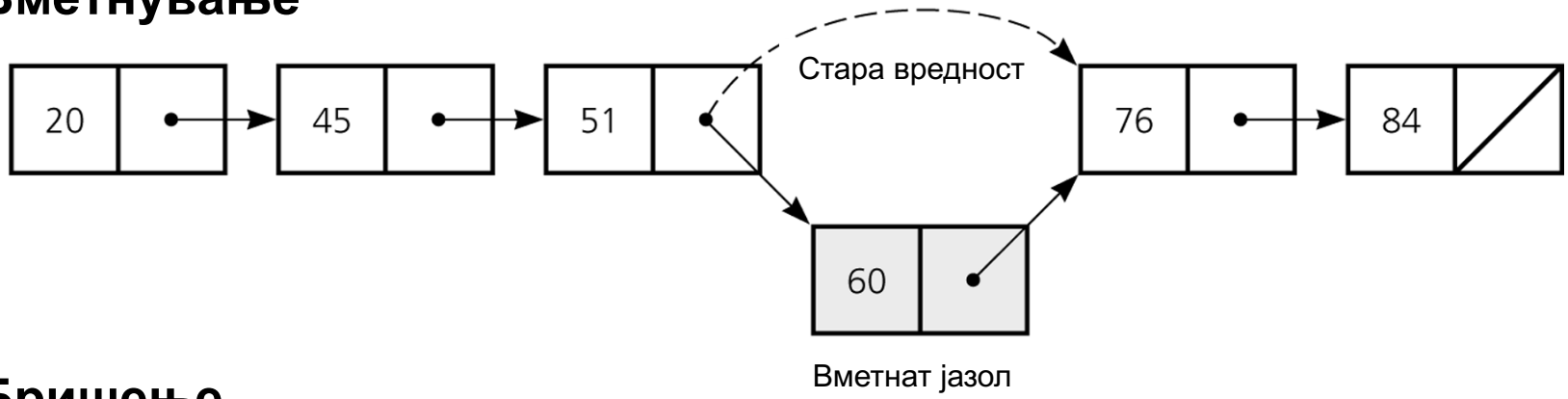
- **Креирање**

А)



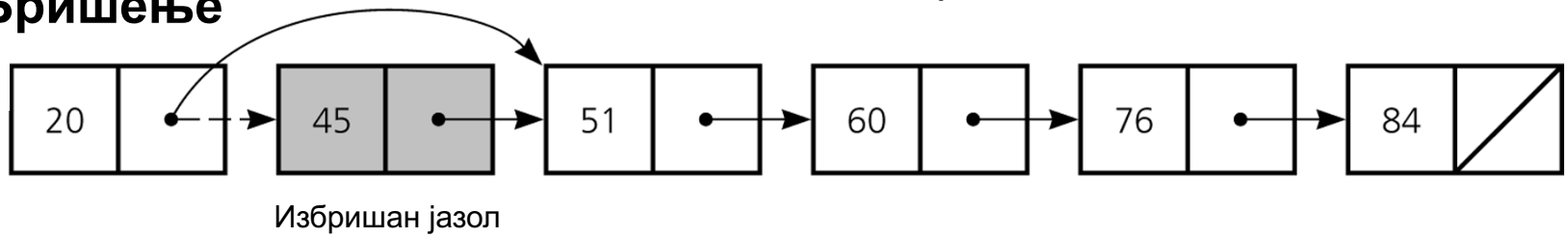
- **Вметнување**

Б)



- **Бришење**

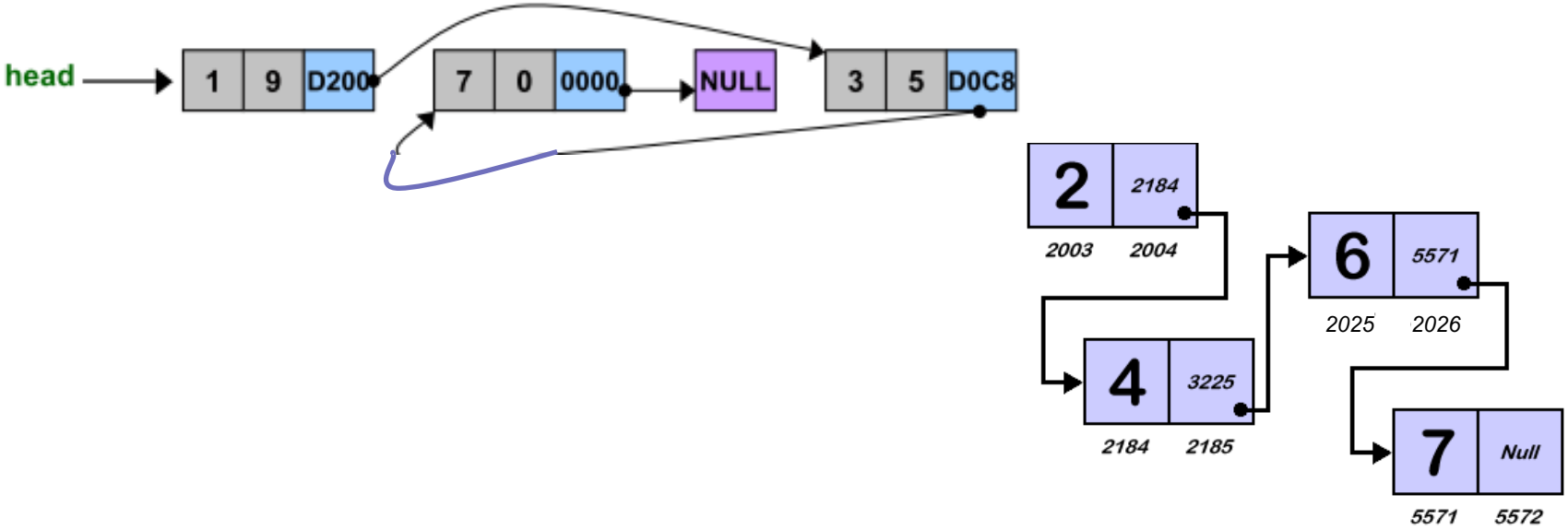
В)



Поврзана листа

- Јазолот се чува **било каде во меморискиот простор** (онаму каде што има слободно место)
- **Редоследот го определуваат само** врските/линковите

Address	D000	D004	D008	..	D0C8	D0CC	D0D0	..	D200	D204	D208
Value	1	9	D200	..	7	0	0000	..	3	5	D0C8



Единечно поврзана листа: Декларирање

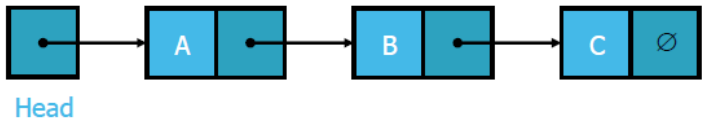
```
struct jazol
{
      int info; //податок
      jazol *link; //покажувач кон следен јазол
};
      jazol *list;
```

- Поврзаната листа не е јазол, туку **покажувач кон јазол**!!!
- Со покажувач кон првиот јазол се идентификува листата

Единечно поврзана листа: Декларирање (2)

```
struct Lista
{
      jazol *head; //покажувач кон првиот јазол
      void init() {head=NULL;}
      void kreirajLista(int nov);
      void dodadiPrv(int prv);
      void dodadiPosleden(int posleden);
      void pechatiLista();
      void brishiPrv();
      void brishiPosleden();
      void brishiLista();
};
```

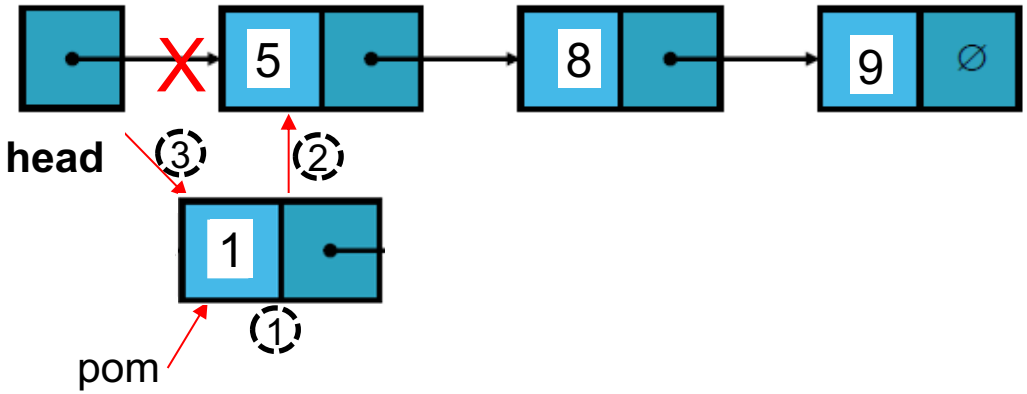
Единечно поврзана листа: Креирање/Додавање




```
void Lista::kreirajLista(int nov)
{
        head = new jazol;
        head->info = nov;
        head->link =  NULL;
}
```

```
void Lista::dodadiPrv(int prv)
{
        jazol *pom = new jazol;
        pom->info = prv;
        pom->link = head;
        head=pom;
}
```

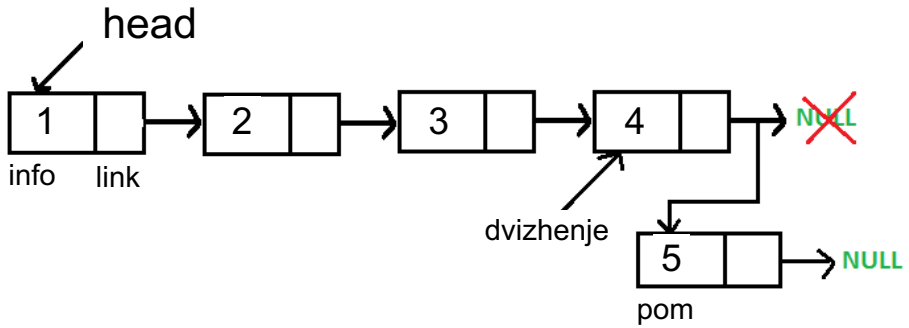
```
//Повик во main():

Lista L1;
L1.init();
L1.kreirajLista(5);
```



Единечно поврзана листа: Додавање на последен јазол (на крај)

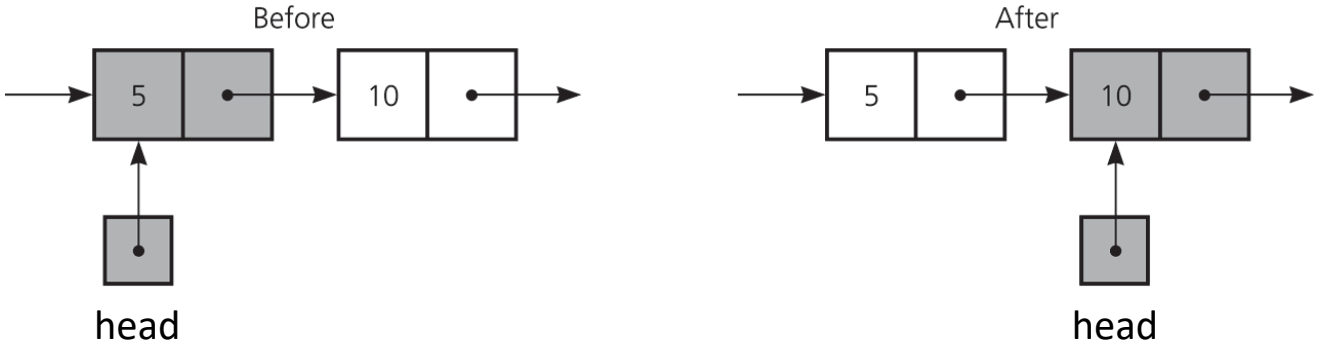
```
void Lista::dodadiPosleden(int posl)
{
        jazol *dvizhenje = head;
        jazol *pom= new jazol;
        pom->info = posl;
        if (head == NULL) head = pom;
        else
        {
                while (dvizhenje->link != NULL)
                        dvizhenje = dvizhenje->link;
                dvizhenje->link = pom;
        }
        pom->link = NULL;
}
```

```
//Повик во main():

Lista L1;
L1.init();
L1.kreirajLista(5);
L1.dodadiPosleden(7);
```



Единечно поврзана листа: Печатење

- Се пристапува до info полето на секој елемент и се печати неговата вредност



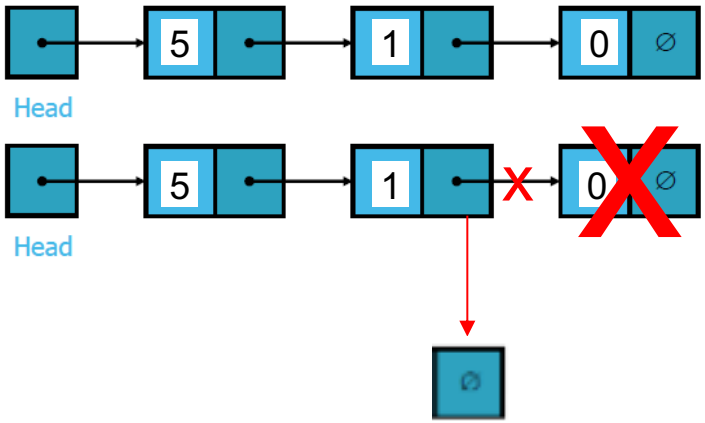
- Се итерира до крај на листата

Ефектот на head=head->link

```
void Lista::pechatiLista() {
        cout << "Elementite vo listata se: ";
        for (jazol *pom = head; pom != NULL; pom = pom->link)
                cout << pom->info << '\t’;
        cout << endl;
}
```

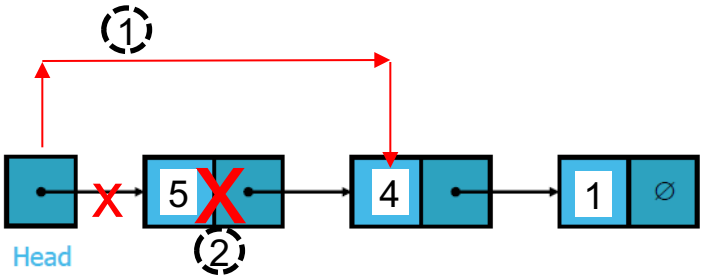
Единечно поврзана листа: Бришење последен елемент од листа

```
void Lista::brishiPosleden()
{
        if(head != NULL)
        {
                if(head->link == NULL)
                {
                        delete head;
                        head = NULL;
                }
                else
                {
                        jazol *pom = head, *brishi;
                        while (pom->link->link != NULL)
                                pom = pom->link;

                        brishi = pom->link;
                        pom->link = NULL;
                        delete brishi;
                }
        }
}
```



Единечно поврзана листа: Ослободување меморија/ бришење листа и бришење прв елемент

```
void Lista::brishiLista()
{
        while (head != NULL)
                brishiPosleden();
}
```




```
void Lista::brishiPrv()
{
        if(head != NULL)
        {
                if(head->link == NULL)
                {
                        delete head;
                        head = NULL;
                }
                else
                {
                        jazol *pom = head;
                        head = head->link;
                        delete pom;
                }
        }
}
```

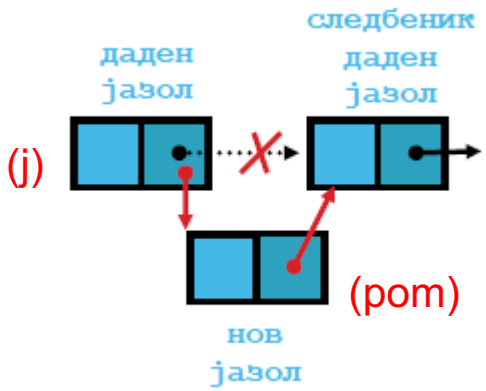
Единечно поврзана листа: Должина на листа

```
int Lista::list_length(){
    int n = 0;
    jazol *pom = head;

    while(pom!=NULL){
        pom = pom->link;
        n++;
    }
    return n;
}
```

Единечно поврзана листа: Вметнување на јазол после даден јазол

- **Чекори**:
  - Алоцирај меморија за новиот јазол
  - Новиот јазол нека покажува кон следбеникот на даденият јазол
  - Даденият јазол нека покажува кон новиот јазол
- Мора да се внимава да не се изгуби предвреме адресата на било кој јазол

```
void insert_inside(jazol *j, int data)
{
      jazol *pom= new jazol;
       pom->info = data;
       pom->link = j->link;
       j->link = pom;
}
```

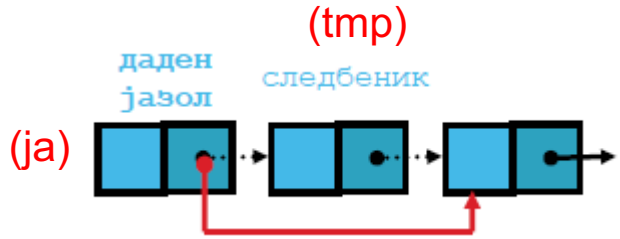


Внимавајте по кој редослед ги креирате новите врски!!!

Единечно поврзана листа: Бришење на јазол после даден јазол

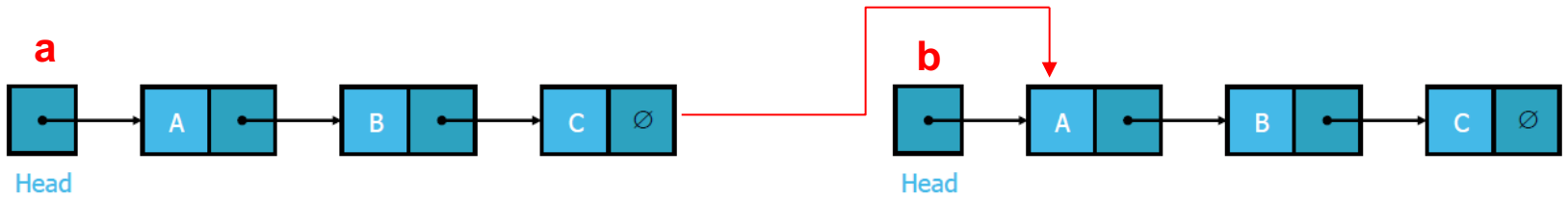
- **Чекори**:
  - Ако листата се состои само од еден елемент, врати
  - Дадениот јазол нека покажува кон следбеникот на неговиот следбеник
  - Ослободи ја меморијата за новиот јазол

```
void delete_after(jazol *ja)
{
        if (ja->link == NULL) return;
        jazol *tmp = ja->link; //tmp е следбеник
        ja->link = tmp->link;
        delete tmp;
}
```



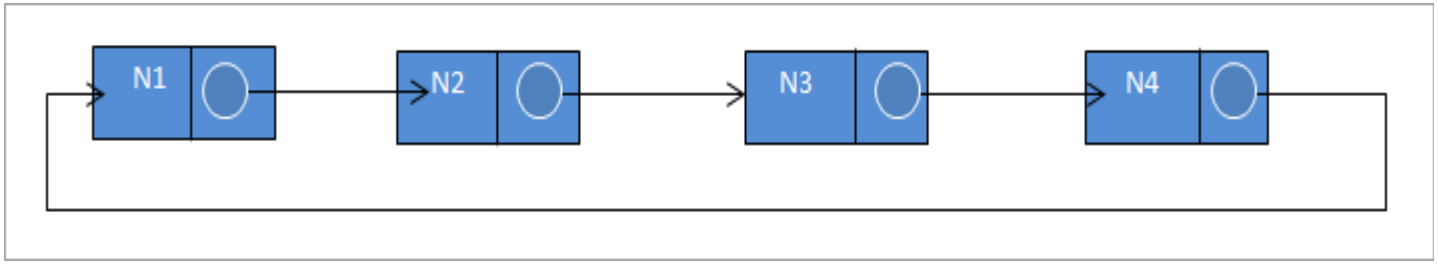
Единечно поврзана листа: Поврзување на две единечни листи - со рекурзија

```
void spojuvanje(jazol *a, jazol *b)
{
        if ((a!= NULL)&&(b!= NULL))
         {
              if (a->link== NULL)
                    a->link = b;
              else spojuvanje(a->link, b);
         }
        else cout<<“Ili prvata ili vtorata niza se NULL";

}
```



Единечни кружни поврзани листи

- **Кружните поврзани листи се варијација на единечни поврани листи**
  - Последниот јазол не покажива на NULL, туку на првиот јазол (во неговото link поле ја има адресата на првиот јазол).



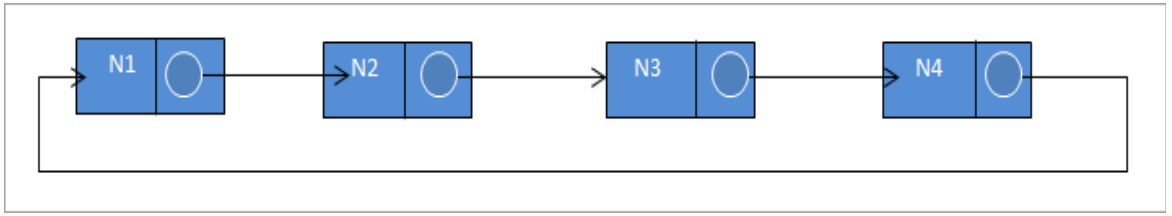
Единечни кружни поврзани листи: Декларирање

```
struct KruznaLista
{     jazol *head; //покажувач кон првиот јазол
      jazol *end; //покажувач кон последниот јазол
      int brojelementi;

      void init() {head=end=NULL; brojelementi=0;};
      void kreirajLista(int nov);
      void dodadiPrv(int prv);
      void dodadiPosleden(int posleden);
      void pechatiLista();
      void brishiPrv();
      void brishiPosleden();
      void brishiLista();
};
```

Единечни кружни поврзани листи: Креирање

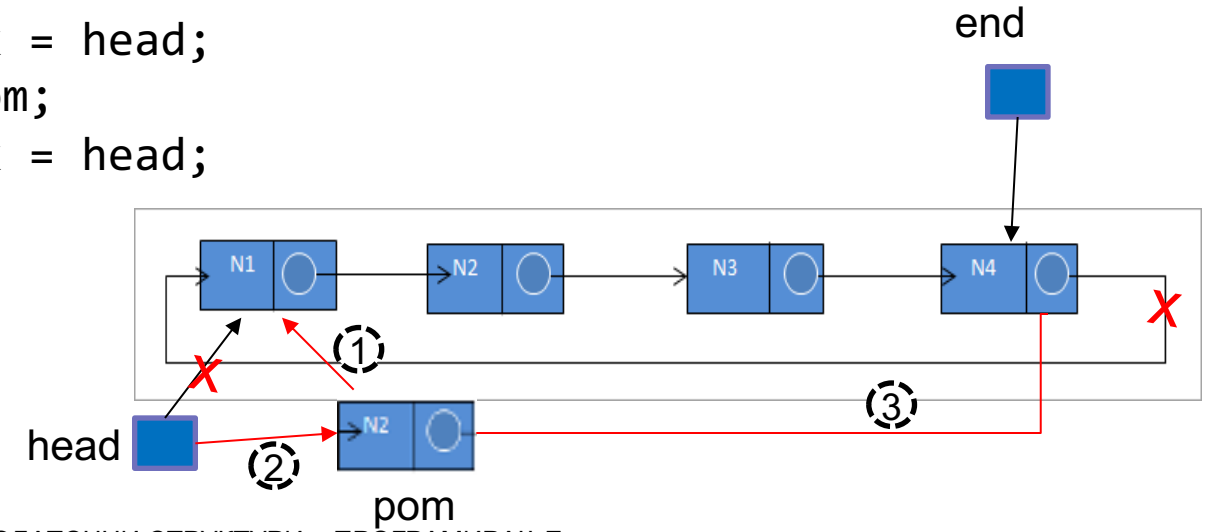
```
void KruznaLista:: kreirajLista(int nov){
    jazol *pom = new jazol;
    pom->info = nov;
    pom->link = pom;
    end = head = pom;
    brojelementi = 1;
}
```

```
//Повик во main():

KruznaLista L1;
L1.init();
L1.kreirajLista(5);
```



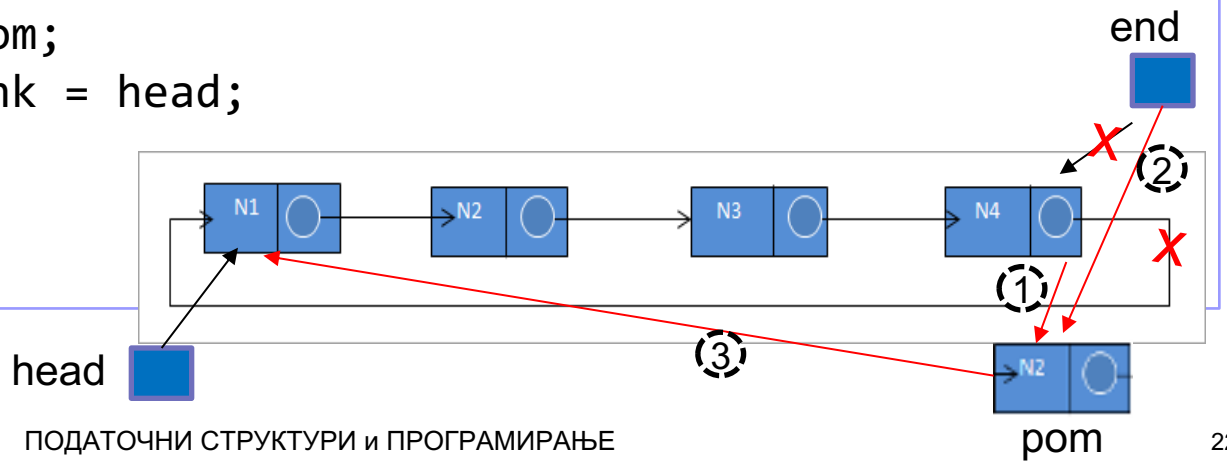
Единечни кружни поврзани листи: Додавање

```
void KruznaLista::dodadiPrv(int prv)
{
        jazol *pom = new jazol;
       pom->info = prv;
       if (head == NULL) //ako listata ne sodrzhi elementi
       {
              pom->link = pom; //jazolot pokazhuva sam kon sebe
              end = head = pom;
       }
       else
       {
              pom->link = head;
              head = pom;
              end->link = head;
       }
       brojelementi++;
}
```



Единечни кружни поврзани листи: Додавање

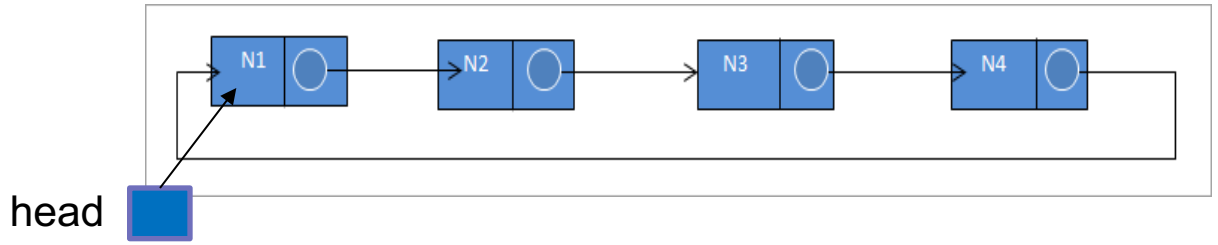
```
void KruznaLsta::dodadiPosleden (int posleden)
{
        jazol *pom = new jazol;
        pom->info = posleden;
        if (head == NULL)
        //ako listata ne sodrzhi elementi
        {
                pom->link = pom; //jazolot pokazhuva sam kon sebe
                end = head = pom;
        }
        else
        {
                end->link = pom;
                end = pom;
                end->link = head;
        }
        brojelementi++;
}
```



Единечна кружна поврзана листа: Печатење

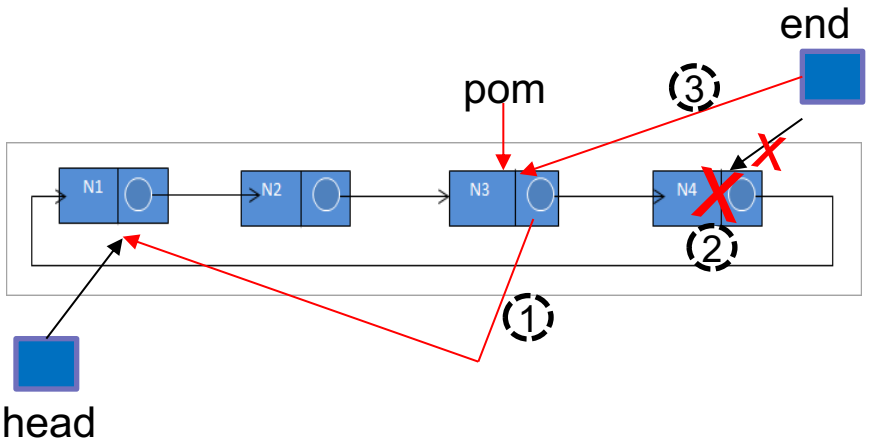
- Се пристапува до info полето на секој елемент и се печати неговата вредност, и се итерира до крај на листата (опашот)

```
void KruznaLista::pechatiLista()
{
        jazol *pom = head;
        cout << "Elementite vo listata se: ";

        if (head == NULL) cout << “Listata e prazna";
        else
        {
                cout << pom->info << '\t’;
                pom = pom->link;
                for ( ; pom != head; pom = pom->link)
                        cout << pom->info << '\t’;
                cout << endl;
        }
}
```



Единечна кружна поврзана листа: Бришење последен елемент од листа

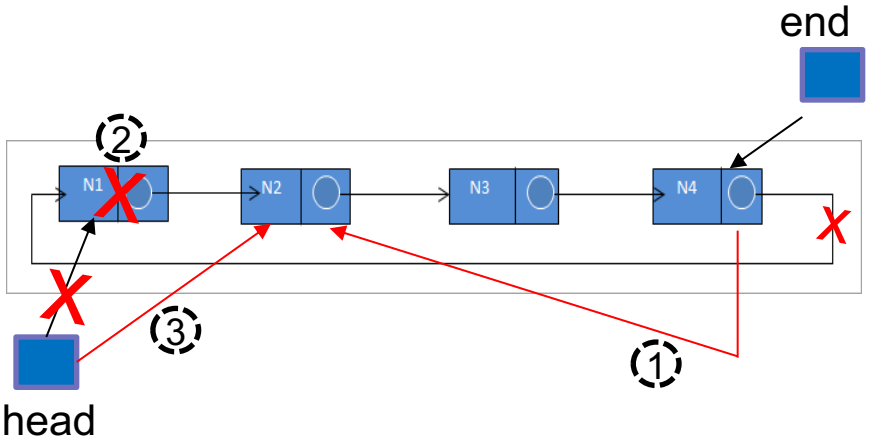
```
void KruznaLista::brishiPosleden()
{
        if(head->link == head)
        {
              delete head;
              head =end = NULL;
        }
        if(head != NULL)
        {
            jazol *pom = head;
            while (pom->link!= end) //stop na pretposleden jazol
                pom = pom->link; //postavi na posleden jazol

            pom->link=head; //pretposeleden jazol go povrzuvame so prviot
            delete end; //osloboduvanje memorija
            end=pom; //pretposeleden jazol e end
        }
        brojelementi--;
}
```



Единечна кружна поврзана листа: Ослободување меморија/ бришење листа

```
void KruznaLista::brishiLista()
{
      while (head != NULL)
            brishiPosleden();
      brojelementi=0;
}
```

Единечна кружна поврзана листа: Ослободување меморија - бришење прв елемент




```
void KruznaLista::brishiPrv()
{
    if(head->link == head)
    {
        delete head;
        head =end= NULL;
    }
    if(head!= NULL)
    {
        end->link = head->link;//krajot na listata pokazhuva kon vtoriot jazol

        delete head; //osloboduvame memorija za prviot jazol
        head = end->link; //pochetok stanuva vtoriot jazol
    }
    brojelementi--;
}
```