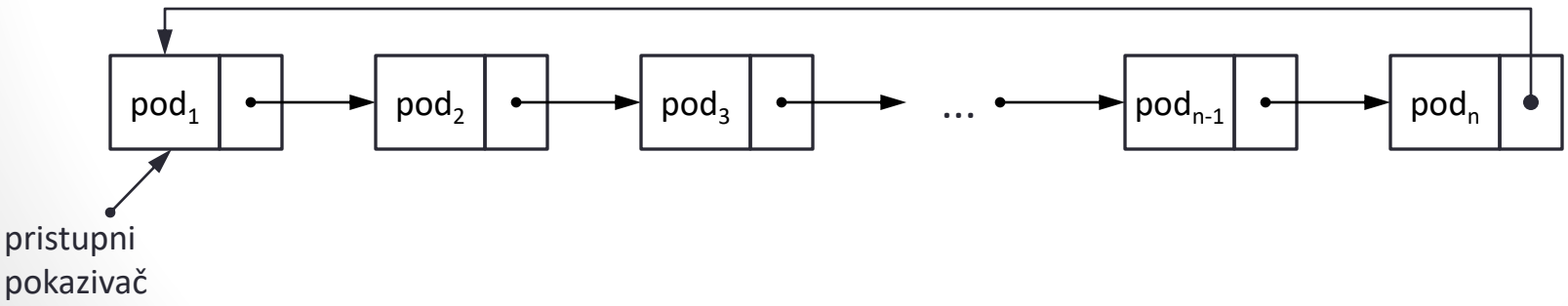
KRUŽNE LISTE

Kružne (cirkularne) liste

- Liste kod kojih poslednji element pokazuje na prvi – nemaju ni početak ni kraj
- Prolaz kroz listu se može započeti od bilo kog elementa
- Umesto termina glava liste koristi se termin pristupni pokazivač
- Koriste se za podatke koji su po prirodi kružni – podela procesorskog vremena različitim korisnicima



Primer

- Kreiranje i štampanje sadržaja kružne liste

```
# include <stdio.h>
# include <stdlib.h>

struct element
{
    int podatak;
    struct element *sledeci;
};
```

Primer

```
struct element *dodaj(struct element *p, int n)
{
    struct element *pom;

     /* ukoliko je lista prazna, novi element se dodaje kao pocetni */
    if(p==NULL)
    {
        /* kreiranje novog elementa */
        p=(struct element *)malloc(sizeof(struct element));
        if(p==NULL)
        {
            printf("Greska.\n");
            exit(0);
        }

        p->podatak = n;
        p->sledeci = p; /* element pokazuje na samog sebe posto je
                                lista kruzna */
    }
    ...
```

Primer

```
    ...
    else
    {
        pom = p;
        /* prolazak bi se dobio pokazivac na poslednji element */
        while (pom->sledeci != p)  pom = pom->sledeci;

        /* kreiranje novog elementa */
        pom->sledeci = (struct element *)malloc(sizeof(struct element));
        if(pom->sledeci == NULL)
        {
            printf("Greska.\n");
            exit(0);
        }
        pom = pom->sledeci;
        pom->podatak = n;
        pom->sledeci = p; /* poslednji element pokazuje na  pocetak*/
    }
    return (p);
}
```

Primer

```
void stampaj_listu( struct element *p )
{
    struct element *pom;

    pom = p;
    printf("Podaci u listi su:\n");
    if(p!= NULL)
    {
        do
        {
            printf(%d\t",pom->podatak);
            pom=pom->sledeci;
        } while (pom!= p);

        printf("\n");
    }
    else
        printf("Lista je prazna.\n");
}
```

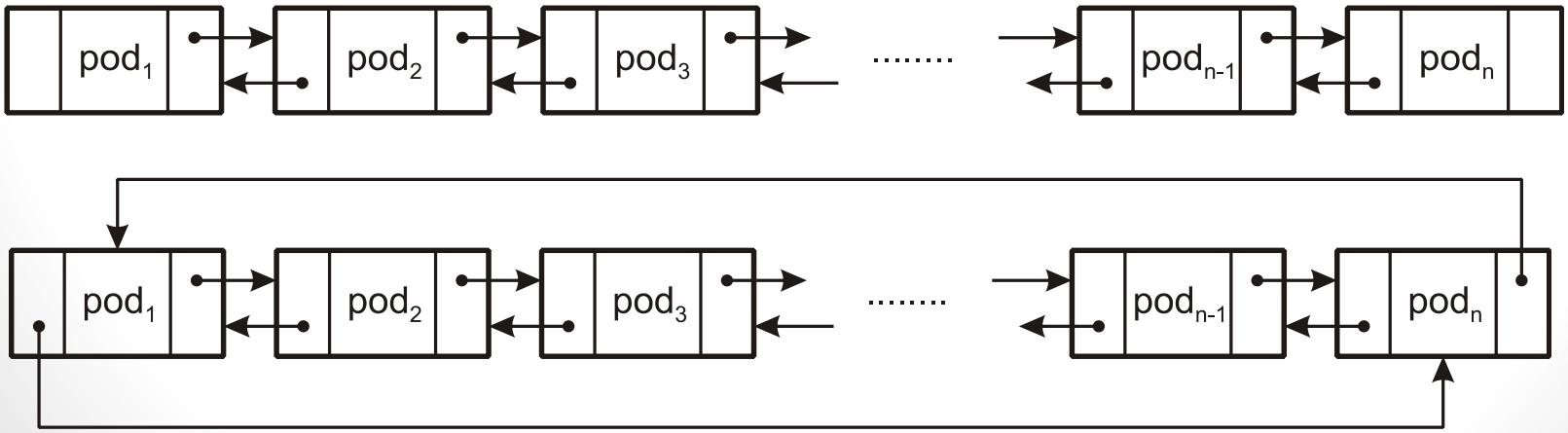
Primer

```
void main()
{
    int n,i;
    int x;
    struct element *pristupni = NULL;

    printf("Unesite broj elementa liste:\n");
    scanf("%d",&n);

    for(i=0; i<n; i++)
    {
        printf("Unesite vrednost:\n");
        scanf("%d",&x);
        pristupni = dodaj( pristupni, x );
    }

    printf("Kreirana lista je:\n");
    stampaj_listu( pristupni );
}
```

DVOSTRUKO POVEZANE LISTE

Dvostruko povezane liste

- Problemi u radu sa povezanim listama
  - Kretanje samo u jednom smeru
  - Prilikom brisanja i dodavanja stalno čuvanje informacije o prethodniku
  - Gubljenjem veze kod jednog elementa ostali elementi postaju nedostupni
- Navedeni problemi se mogu prevazići dvostruko povezanim listama
- Može biti u lancu ili kružna



Dvostruko povezane liste

- Struktura za realizaciju

```
struct delement
{
  int podatak;
  struct delement *levi,*desni;
};
```

Primer

- Kreirati i štampati dvostruko povezanu listu

```
# include <stdio.h>
# include <stdlib.h>

struct delement
{
  int podatak;
  struct delement *levi, *desni;
};
```

Primer

```
struct delement *dodaj(struct delement *p, struct delement **q, int n)
{
  struct delement *pom;

  /* ukoliko je lista prazna, novi element se dodaje kao pocetni
     element liste */
  if(p==NULL)
  {
    p=(struct delement *)malloc(sizeof(struct delement));
    if(p==NULL)
    {
      printf("Greska.\n");
      exit(0);
    }

    p->podatak = n;
    p->levi = p->desni = NULL; /* novi element nema ni levog
                                  ni desnog suseda */
    *q = p;/* kraj liste je i pocetak liste */
  }
  ...
```

Primer

```
  ...
  else
  {
    pom = (struct delement *)malloc(sizeof(struct delement));
    if(pom == NULL)
    {
      printf("Greska.\n");
      exit(0);
    }

    pom->podatak = n;
    pom->levi = (*q);  /* poslednji el. liste postaje levi novom el. */
    pom->desni = NULL; /* novi el. nema desnog suseda */
    (*q)->desni = pom; /* novi el. postaje desni poslednjem el. liste */
    (*q) = pom;         /* novi el. postaje poslednji el. liste */
  }

  return (p);
}
```

Primer

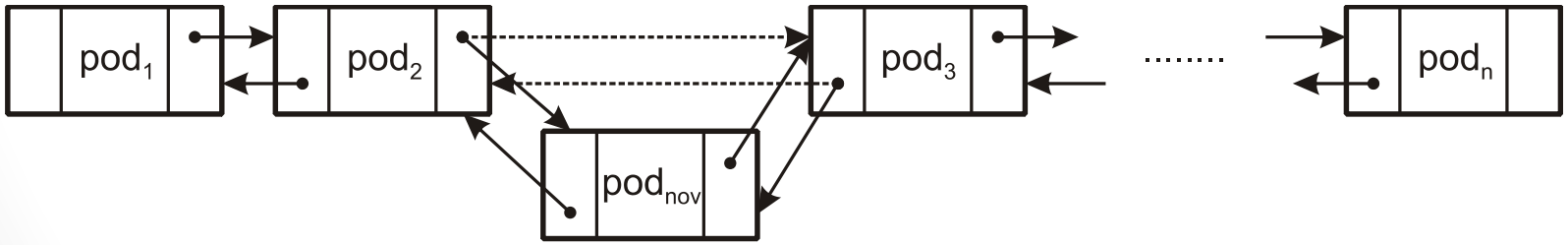
```
void stampaj_desno( struct delement *p )
{
  printf("Podaci u listi stampani u desno su:\n");
  while (p!= NULL)
  {
    printf("%d\t",p->podatak);
    p = p->desni;
  }
}

void stampaj_levo( struct delement *p )
{
  printf("Podaci u listi stampani u levo su:\n");
  while (p!= NULL)
  {
    printf("%d\t",p->podatak);
    p = p->levi;
  }
}
```

Primer

```
void main()
{
  int n,i;
  int x;
  struct delement *pocetak = NULL ;
  struct delement *kraj = NULL;

  printf("Unesite broj elementa liste:\n");
  scanf("%d",&n);

  for(i=0; i<n; i++)
  {
    printf( "Unesi vrednost:\n");
    scanf("%d",&x);
    pocetak = dodaj( pocetak, &kraj, x );
  }

  printf("Kreirana lista je:\n");
  stampaj_desno( pocetak );
  stampaj_levo( kraj );
}
```

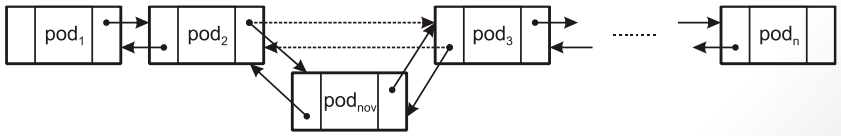
Umetanje elementa iza određene poziciju

1. Prolaskom kroz listu određuje se element iza koga se umeće novi element
2. Kreira se novi elementa, njegovom levom i desnom linku se dodeljuju pokazivači na odgovarajuće elemente
3. Menjaju se pokazivači elemenata između kojih se umeće novi element



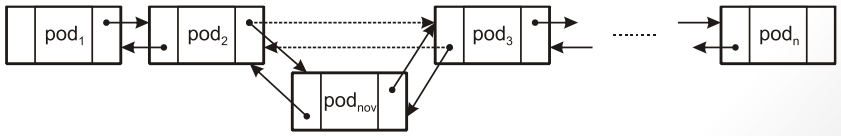
Umetanje elementa iza određene poziciju

```
struct delement * umetni(struct delement *p, struct delement **q,
                         int redni_broj, int vrednost )
{
  struct delement *pom, *pom1;
  int i;

  if ( redni_broj < 0 || redni_broj > duzina(p) )
  {
    printf("Greska! Zadati element ne postoji.\n");
    exit(0);
  }
  ...
```



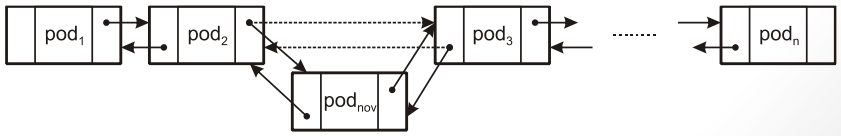
Umetanje elementa iza određene poziciju

```
...
if ( redni_broj == 0)
{
  pom = ( struct delement * )malloc( sizeof( struct delement ));
  if ( pom == NULL )
  {
    printf( "Greska pri alociranju memorije.\n");
    exit (0);
  }
  pom->podatak = vrednost;
  pom->desni = p;
  pom->levi = NULL;
  p->levi = pom;
  p = pom;
}
...
```



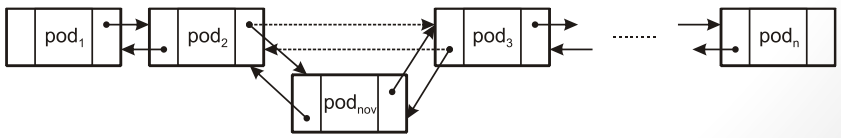
Umetanje elementa iza određene poziciju

```
...
else
{
  pom = p ;
  i = 1;
  while ( i < redni_broj )
  {
    i = i+1;
    pom = pom->desni;
  }

  pom1 = ( struct delement * )malloc( sizeof(struct delement));
  if ( pom == NULL )
  {
    printf( "Greska pri alociranju memorije.\n");
    exit(0);
  }

  ...
```



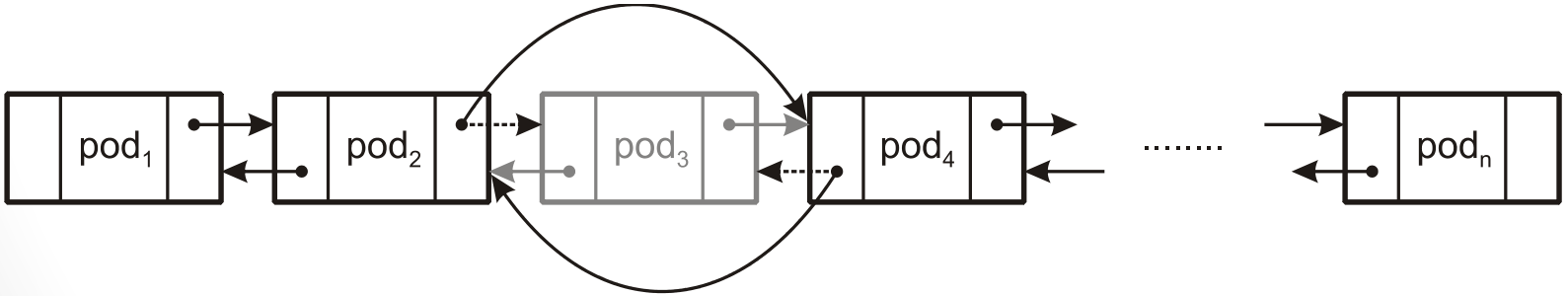
Umetanje elementa iza određene poziciju

```
    ...
    pom1->podatak = vrednost;
    pom1->desni = pom->desni;
    pom1->levi = pom;
    if (pom1->desni != NULL)
      pom1->desni->levi = pom1;
    else
      *q = pom1;
    pom1->levi->desni = pom1;
  }

  return (p);
}
```



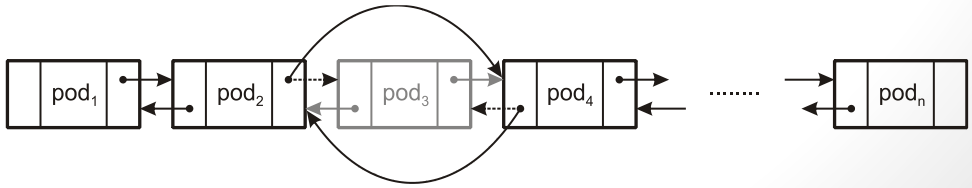
Brisanje elementa

1. Prolaskom kroz listu određuje se element koji se briše
2. Menjaju se pokazivači okolnih elemenata
3. Element se uklanja iz memorije



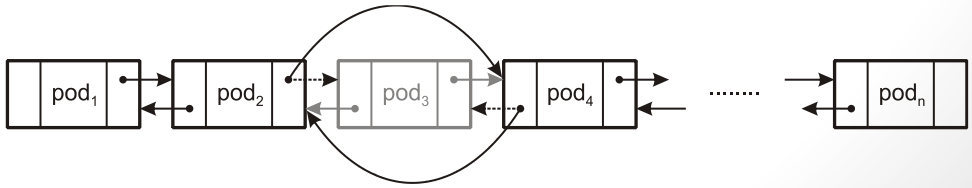
Brisanje elementa

```
struct delement * obrisi( struct delement *p, struct delement **q,
                          int redni_broj)
{
  struct delement *pom;
  int i;

  if (redni_broj <= 0 || redni_broj > duzina(p))
  {
    printf("Greska! Zadati element ne postoji.\n");
    exit(0);
  }

  pom = p ;
  i = 1;
  while ( i < redni_broj )
  {
    i = i+1;
    pom = pom->desni;
  }
  ...
```



Brisanje elementa

```
  ...
  if(pom->levi != NULL)
    pom->levi->desni = pom->desni;
  else
    p = pom->desni;

  if(pom->desni != NULL)
    pom->desni->levi = pom->levi;
  else
    (*q) = pom->levi;

  free(pom);

  return (p);
}
```



K R A J !

Do kraja maja konsultacije u terminu predavanja, nakon toga po dogovoru