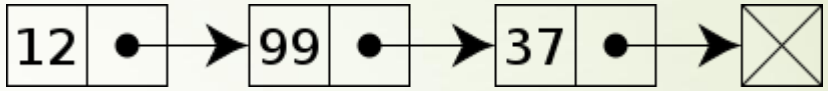
ALGORITMI I STRUKTURE PODATAKA

RAČUNSKE VEŽBE - TERMIN BR. 4 - LANČANE LISTE - 23.3.2020.

ALDINA AVDIĆ, DIPL. INŽ. apljaskovic@np.ac.rs

RAČUNARSKA TEHNIKA; SOFTVERSKO INŽENJERSTVO;

Jednostruke povezane liste




```
typedef struct node
{
    int data;
    struct node *next;
}NODE;
NODE *first=NULL,*last=NULL,*temp=NULL;


int isEmpty(){
    if (first==NULL) return 1;
    else return 0;
}
```

Kreiranje elementa

```
void create(int elem)
{
    temp=(struct node*)malloc(sizeof(struct node));
    temp->data=elem;
    temp->next=NULL;
    if(first==NULL)
    {
        first=temp;
        last=temp;
    }
    else
    {
        last->next=temp;
        last=temp;
    }
}
```

Dodavanje elementa posle zadatog elementa

```
void insert_after(NODE *pom, int elem)
{

    temp=(struct node*)malloc(sizeof(struct node));
    temp->data=elem;
    temp->next=pom->next;
    pom->next=temp;
    if (pom==last) last=temp;

}
```

Dodavanje elementa pre zadatog elementa

```
void insert_before(NODE *pom, int elem)
{

    temp=(struct node*)malloc(sizeof(struct node));
     temp->next=pom->next;
     temp->data=pom->data;
     pom->data=elem;
     pom->next=temp;
     if (pom==last){
     temp=last;
     }
}
```

Brisanje nakon zadatog elementa

```
void delete_after(NODE *pom)
{
//if (isEmpty()) printf("List is empty");
temp=pom->next;
pom->next=temp->next;
temp->next=NULL;
free(temp);
}
```

Brisanje zadatog čvora

```
void delete_actual(NODE *pom){
    if ((pom==first)&(first==last)){
        first=last=NULL;
        free(pom);
    }
    temp=pom->next;
    pom->next=temp->next;
    pom->data=temp->data;
    free(temp);

}
```

Brisanje elementa na zadatoj poziciji / vrednosti

```
void delete_on_position(int pos)
{
int count=1;
NODE *p, *q;
p=first;
q=NULL;
if (pos==1){
    first=p->next;
    free(p);
    return;
}
while(pos!=count)
{
 q=p;
 p=p->next;
 count++;
}
delete_after(q);
}
```

```
void delete_by_value(int value){
NODE *p,*q;
    p=first;
    q=NULL;
    while(p!=NULL)
    {
        if(value==p->data)
        break;
        else
        {
        q=p;
        p=p->next;
            }
    }
    if((p==first)&(first==last))
    {
        first=last=NULL;
        free(p);
        printf("Lista je sada prazna!");
    }
    else if(p==last)
    {
        q->next=NULL;
        free(p);
        last=q;
    }
    else
    {
        q->next=p->next;
        free(p);
    }
}
```

Prikaz elementa

```
void display()
{
    temp=first;
    printf("First->");
    while(temp!=NULL)
    {
    //printf("|%d|%d| --> ",temp->data,temp->next);
    printf("|%d| --> ",temp->data);
    temp=temp->next;
    }
      printf("NULL");
}
```

Invertovanje liste i konkatenacija dve liste

```
NODE *invertuj(NODE *first){
NODE *p,*q,*r;
p=first;
q=NULL;
while (p!=NULL) {
r=q;
q=p;
p=p->next;
q->next=r;
}
first=q;
return first;
}
```

```
NODE *concat(NODE *list1, NODE *list2){
    NODE *p;
    if (list1==NULL) return list2;
    else if (list2==NULL) return list1;
    p=list1;
    while(p->next!=NULL){
        p=p->next;
    }
    p->next=list2;
    return list1;
}
```

Jednostruko povezane liste - zadaci

- **Zadatak 1.** Napisati potprogram koji na osnovu zadate vrednosti elementa:
  - (a)vraća 1 ako se elementa nalazi u lančanoj listi, u suprotnom vraća 0
  - (b) vraća element ako je nađen u listi
  - (c) vraća presek dve lančane liste
- **Zadatak 2.** Napisati program kojim se svako pojavljivanje elementa **el1** u listi zamenjuje elementom **el2.**
- **Zadatak 3.** Napisati potprogram kojim se dodaje element na početak lančane liste.
- **Zadatak 4.** Napisati program kojim se briše element s početka lančane liste.

Jednostruko povezane liste - zadaci

- **Zadatak 5.** Napisati potprogram kojim se proverava da li su dve lančane liste **L1** i **L2** jednake.
- **Zadatak 6.** Napisati rekurzivni potprogram koji nalazi:
  - (a) maksimalni element lančane liste,
  - (b) sumu svih elemenata lančane liste
  - (c) ispituje da li se zadati element nalazi u listi
  - (d) zamenjuje svako pojavljivanje elementa el1 elementom el2

Hvala na pažnji!