acs
applied computer science

UNIVERZITET U NOVOM SADU
FAKULTET TEHNIČKIH NAUKA
KATEDRA ZA PRIMENJENE RAČUNARSKE NAUKE

Osnovi programiranja i programskih jezika

Računarske vežbe – vežba 9

Zimski semestar 2025/2026.

Studijski program: Informacioni inženjering

Spregnuta lista
Stek i red

Spregnuta lista

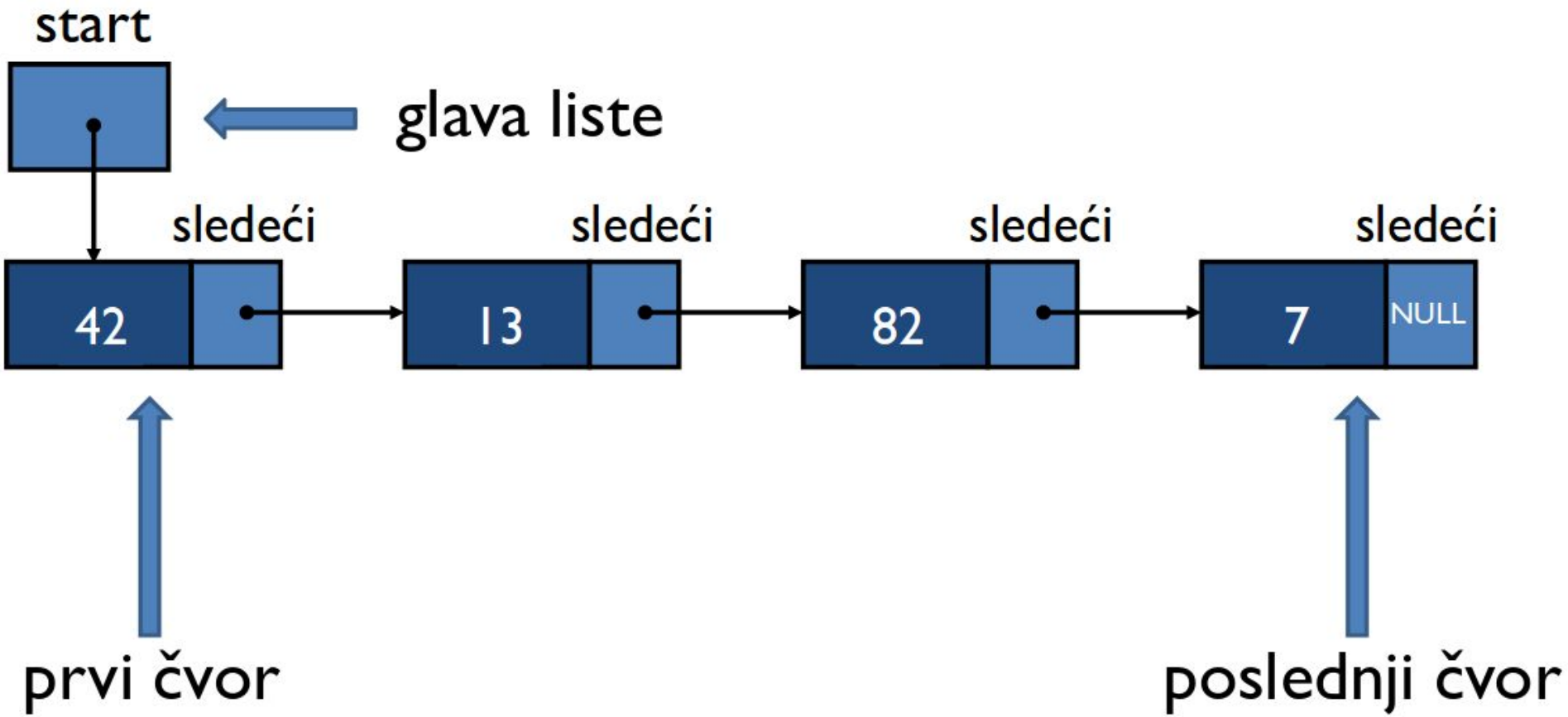
- Linearna dinamička struktura podataka
- Logički susedni elementi liste ne moraju biti i fizički (u memoriji) susedni
- Gradivni element spregnute liste naziva se čvor

Čvor - samoupućujuća struktura

- Čvor sadrži:
  - podatak – informativni deo čvora
  - sledeći – adresa sledećeg čvora u listi

```
typedef struct node {
    int data;
    struct node *next;
} TNODE;
```

Primer jednostruko spregnute liste



Tipične operacije nad listom

Tipične operacije nad jednostruko spregnutom listom su:

- inicijalizacija liste
- dodavanje novog elementa
- obilazak (listanje) liste
- brisanje elementa iz liste
- brisanje kompletne liste

Inicijalizicija liste

- Dovođenje liste u početno stanje postavljanjem glave liste na NULL

```
head = NULL;
```

Dodavanje novog elementa

- Formira se novi čvor
- Popunjava se polje podatka
- Uvezuje se u postojeću listu na predviđeno mesto
  - na početak liste
  - na kraj liste
  - u sredinu

Dodavanje na početak liste

```
void push(TNODE **phead, int new_data) {
    TNODE *new = (TNODE*) malloc(sizeof(TNODE));

    new->data = new_data;
    new->next = *phead;

    *phead = new;
}
```

Dodavanje na kraj liste

```
void append(TNODE **phead, int new_data) {
    TNODE *new = (TNODE*) malloc(sizeof(TNODE)); // novi čvor

    new->data = new_data;
    new->next = NULL;           // biće poslednji

    if (*phead == NULL) {       // da li je lista prazna
       *phead = new;
        return;
    }

    TNODE *last = *phead;

    while (last->next)          // pratimo pokazivače do poslednjeg čvora
        last = last->next;

    last->next = new;           // novi čvor postaje poslednji
    return;
}
```

Obilazak liste

```
// Ispis podatka svih čvorova
void printList(TNODE *first_node) {
   while (node) {
       printf(" %d ", node->data);
       node = node->next;
   }
}
```

Traženje elementa u listi

```
TNODE* find(TNODE *node, int data) {
    while (node)
        if (node->data == data)
            return node;  // nađen
        else
            node = node->next;

    return NULL;   // čvor ne postoji
}
```

Brisanje elementa iz liste

```
void removeNode(TNODE **phead, int data) {
    TNODE *tmp = *phead, *node = find(*phead, data);

    if (!node) return;          // nije nađen

    if (*phead == node) {       // nađeni čvor je prvi
        *phead = node->next;
        free(node);
        return;
    }

    while (tmp->next != node)
        tmp = tmp->next;

    tmp->next = node->next;     // prevezivanje liste
    free(node);                 // oslobađanje čvora
}
```

Brisanje liste

```
void destroy(TNODE **phead) {
    TNODE *tmp;
    // oslobađa se jedan po jedan čvor
    // počevši od prvog
    while (*phead) {
        tmp = *phead;
        *phead = tmp->next;
        free(tmp);
    }
}
```

Zadatak 1

Napisati C program kojim se pomoću menija bira jedna od sledećih akcija u radu sa spregnutom listom celobrojnih vrednosti:

- dodavanje na početak,
- dodavanje na kraj,
- brisanje čvora sa zadatom vrednošću podatka,
- prikaz liste.

Omogućiti da se svaka od akcija može izvršiti proizvoljan broj puta sve dok korisnik ne zatraži izlaz iz programa.

Zadatak 2

Napisati C program kojim se realizuju sledeće operacije sa spregnutom listom znakova:

- sortirani unos karaktera u listu,
- modifikacija karaktera (uz održavanje osobine sortiranosti),
- traženje karaktera u listi.

Interakciju korisnika sa navedenim akcijama realizovati putem menija.

Stek

Stek (eng. *stack*) je apstraktni tip podataka u koji se elementi dodaju ili iz koga se elementi brišu po LIFO (*last in, first out*) principu.

Osnovne operacije za rad sa stekom su:

- push – stavljanje elementa na stek
- pop – skidanje elementa sa steka

Stek

Stek se može realizovati:

- sekvencijalno (niz)
- spregnuto (spregnuta lista)

Ostale operacija nad stekom:

- `peek` – provera vrednosti na vrhu steka
- `isEmpty` – da li je stek prazan
- `isFull` – da li je stek pun
- `size` - vraća veličinu steka

Red

Red (eng. *queue*) je apstraktni tip podataka u koji se elementi dodaju ili iz koga se elementi brišu po FIFO (*first in, first out*) principu.

Osnovne operacije za rad sa redom su:

- enqueue – dodavanje elementa u red (na kraj)
- dequeue – uklanjanje elementa iz reda (sa početka)

Red

Red se može realizovati:

- sekvencijalno (niz)
- spregnuto (spregnuta lista)

Ostale operacija nad redom:

- `peek/front` – provera vrednosti na početku reda
- `rear` – provera vrednosti na kraju reda
- `isEmpty` – da li je red prazan
- `isFull` – da li je red pun
- `size` - vraća veličinu red

Zadatak za domaći 1

Napisati C program kojim se vrši evaluacija celobrojnog izraza datog u postfiksnoj notaciji.

Primeri izraza:

- 2 3 + 5 * 7 9 - / (2+3)*5/(7-9)
- 3 6 * 4 – 8 / ((3*6)-4)/8
- 9 4 – 4 12 + * 11 2 / - (9-4)*(4+12)-11/2

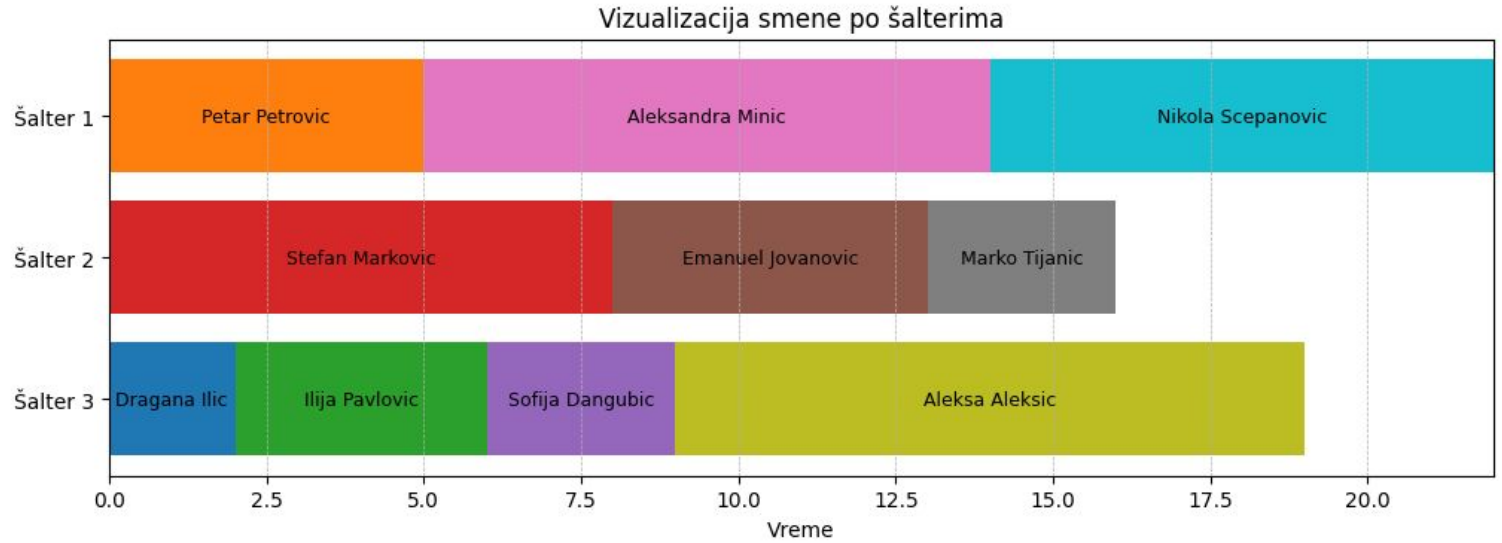
Zadatak 3

Napisati C program kojim se simulira čekanje ljudi u pošti. Pošta ima zadat broj šaltera. Svi klijenti čekaju u jednom redu i idu na prvi šalter koji je slobodan. Spisak klijenata dat je u ulaznom fajlu red.txt tako što su za svakog klijenta dati ime, prezime i broj minuta koje će provesti na šalteru. Napraviti onoliko izlaznih datoteka koliko ima šaltera sa sledećim načinom nazivanja: `"salter1.txt"`, `"salter2.txt"`, …, `"salterN.txt"`, gde N predstavlja zadati broj šaltera. U te datoteke upisati sve klijente koji su opsluženi na tom šalteru, podvući crtu i ispod nje ukupno vreme rada konkretnog šaltera. Realizovati red korišćenjem spregnute linearne liste.

Primer za zadatak 3

Vizuelizacija primera za rad sa 3 šaltera i sledećom ulaznom datotekom:

```
Petar Petrovic 5
Stefan Markovic 8
Dragana Ilic 2
Ilija Pavlovic 4
Aleksandra Minic 9
Sofija Dangubic 3
Emanuel Jovanovic 5
Aleksa Aleksic 10
Marko Tijanic 3
Nikola Scepanovic 8
```



Zadatak 4

Implementirati listu sa tzv. logičkim brisanjem. Svaki čvor liste osim pokazivača treba da sadrži i podatak u vidu strukture. Struktura sadrži sam **podatak** (jedan broj tipa *double*) i **indikator** (*int*) koji je postavljen na ***1*** ako je čvor važeći, odnosno ***0*** ako je čvor obrisan. Operaciju dodavanje treba realizovati da se izvršava na kraju liste, osim u slučaju ako postoji čvor kojem je ***indikator=0*** kada treba novi broj upisati u polje **podatak** i **indikator** postaviti na *jedinicu*. Brisanje čvorova treba realizovati tako da se za element koji treba biti obrisan postavi polje **indikator** na broj 0. Ispis liste ispisuje samo čvorove sa ***indikator=1***.

Zadatak 4

- [1, 3, 5, 6.1, 6.1, 7]
  - Brisanje čvorova koji čuvaju 6.1
- [1, 3, 5, 6.1, 6.1, 7] (sivi ⇔ logički obrisani)
  - Dodavanje broja 4.67
- [1, 3, 5, 4.67, 6.1, 7]
  - Dodavanje broja 0.3
- [1, 3, 5, 4.67, 0.3, 7]
  - Dodavanje broja -100.1001
- [1, 3, 5, 4.67, 0.3, 7, -100.1001]

Zadatak za domaći 2

U Službi smeštaja SCNS se revidiraju prijave za dom. Sve se odvija brzo tako da Mićo Jović kada proveri prijavnicu, stavi je na vrh gomile, a Jovana Mićić sa iste gomile uzima prvu prijavu sa vrha gomile. Napraviti C program koji modeluje njihovo ponašanje. Treba da postoji meni koji omogućuje dodavanje prijavnice, obradu prijavnice i završetak rada. Treba odabrati odgovarajuću strukturu koja modeluje ponašanje gomile prijavnica za dom, a podatak koji se čuva je struktura sa poljima: **ime, godina studija** i **zbir ocena**. Svaka godina studija ima po 3 ispita. Prilikom dodavanje prijave, upisuju se ime (do 20 karaktera), godina studija i ocene za svaku godinu. Prilikom uklanjanja prijave, proverava se da li je (zbir_ocena)/(godina_studija*3) > 8. Ako jeste, na standardnom izlazu treba ispisati "<ime> je dobio dom.", u suprotnom ne treba da se desi ispis. Kada se završava rad, treba ispisati koliko ostaje *nerevidiranih* prijavnica za naredni dan.