ALGORITMI I STRUKTURE PODATAKA

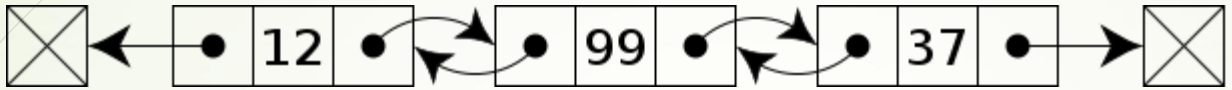
RAČUNSKE VEŽBE - TERMIN BR. 5 - 30.3.2020.

DVOSTRUKO I KRUŽNO ULANČANE LISTE

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

RAČUNARSKA TEHNIKA; INFORMATIKA MATEMATIKA; SOFTVERSKO INŽENJERSTVO

Dvostruko povezana lista



Struktura elementa

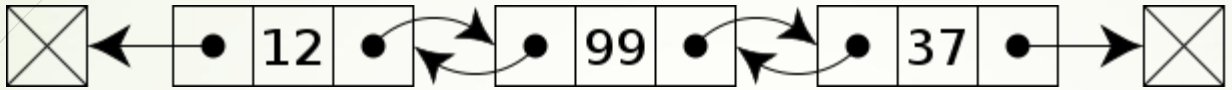
```
#include<stdio.h>
#include<stdlib.h>

struct Node  {
    int data;
    struct Node* next;
    struct Node* prev;
};

struct Node* head; //pok. na 1. elem.
```

- Kao što već znate, za kreiranje elemenata lančanih lista potrebno nam je znanje struktura i pokazivača
- Kod jednostruko ulančanih lista imali smo pokazivač samo na sledeći element
- To je neke operacije otežavalo
- Dvostruko ulančane liste rešavaju te probleme, ali zauzimaju više memorije po elementu, jer čuvaju po jedan pokazivač više
- NULL i ovde označava kraj liste

Dvostruko povezana lista



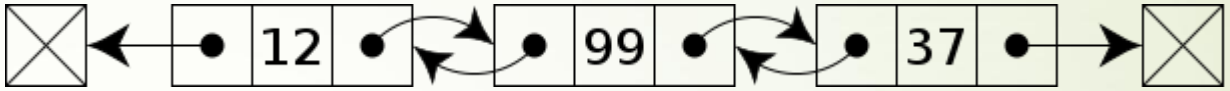
Kreiranje novog elementa

```
struct Node* GetNewNode(int x) {
    struct Node* newNode
        = (struct
Node*)malloc(sizeof(struct Node));
    newNode->data = x;
    newNode->prev = NULL;
    newNode->next = NULL;
    return newNode;
}
```

Objašnjenje:

- Ova funkcija uzima argument x, to je informacija koja se čuva u elementu lančane liste, recimo u listi na slici to su 12, 99, 37
- Da biste kreirali novi element liste morate dinamički zauzeti memoriju za njega
- Zatim u data upisati vrednost argumenta
- Na kraju kažete da su njegov prethodnik i sledbenik NULL

Dvostruko povezana lista - dodavanje elementa



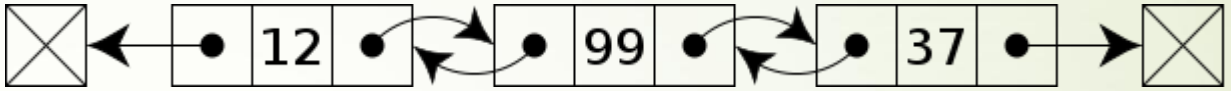
Na početak liste

```
void InsertAtHead(int x) {
      struct Node* newNode = GetNewNode(x);
      if(head == NULL) {
            head = newNode;
            return;
      }
      head->prev = newNode;
      newNode->next = head;
      head = newNode;
}
```

Objašnjenje:

- Ovo je funkcija za dodavanje elementa x na početak liste
- Prvo napravimo element x čiji su prethodnik i sledbenik null koristeći funkciju za kreiranje čvora
- Zatim pitamo da li je lista prazna. Lista je prazna ako joj je prvi element (head) nula
- Ako je prazna, onda novokreirani element proglašavamo za head
- Ako lista nije bila prazna, onda od trenutnog njenog prvog elementa head, prethodnih postaje novi element.
- Sledbenik novog elementa postaje head
- I sada je taj novi element početak liste

Dvostruko povezana lista - dodavanje elementa



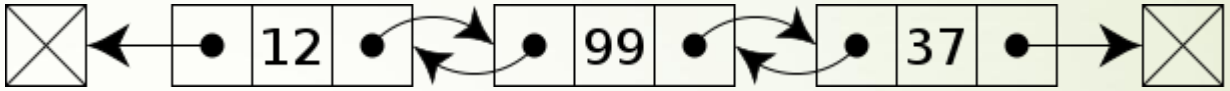
Na kraj liste

```
void InsertAtTail(int x) {
      struct Node* temp = head;
      struct Node* newNode = GetNewNode(x);
      if(head == NULL) {
            head = newNode;
            return;
      }
      while(temp->next != NULL) temp = temp->next;
// Go To last Node
      temp->next = newNode;
      newNode->prev = temp;
}
```

Objašnjenje:

- Ovom funkcijom se element x dodaje na kraj liste
- Međutim, moramo obići celu listu da bismo došli do kraja, tj. Mesta gde treba dodati element.
- Kada god imamo situaciju da se lista obilazi, tada uvodimo čvor temp kome dodeljujemo početak liste, on nam dođe kao brojač
- Zatim kao u prethodnoj funkciji kreiramo novi čvor koji treba dodati
- Opet pitamo da li je lista prazna, ako jeste, onda prvi element liste postaje taj kreirani čvor
- Ako lista nije prazna, pomoću while petlje je obilazimo, sve dok ima sledećeg elementa, prelazimo na njega dok ne dođemo do kraja, i dodamo veze

Dvostruko povezana lista - prikaz elemenata



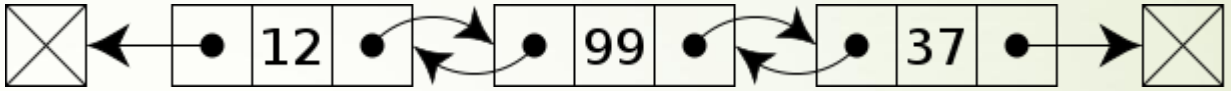
Standardno

```
void Print() {
    struct Node* temp = head;
    printf("Forward: ");
    while(temp != NULL) {
        printf("%d ",temp->data);
        temp = temp->next;}
    printf("\n");
}
```

Objašnjenje:

- Štampanje se svodi na obilazak liste
- Postavimo temp na početak liste
- I preko while petlje sve dok ne dođemo do kraja, štampamo info deo liste (data)
- Zatim prelazimo na sledeći
- Treba napraviti razliku kad se kreira novi element i kad se samo postavlja pokazivač na neki element (temp se ne kreira!)

Dvostruko povezana lista - prikaz elemenata



Inverzno

```
void ReversePrint() {
    struct Node* temp = head;
    if(temp == NULL) return;
    while(temp->next != NULL) {
        temp = temp->next; }
        printf("Reverse: ");
    while(temp != NULL) {
        printf("%d ",temp->data);
        temp = temp->prev; }
    printf("\n");}
```

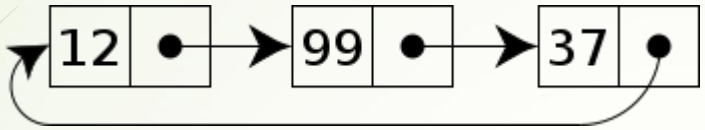
Objašnjenje:

- Lista se može štampati i inverzno
- Opet postavimo brojač temp na početak liste
- Zatim obilazimo listu dok ne dođemo do zadnjeg elementa
- Sada pet obilazimo listu, ali unazad i štampamo elemente
- Ovo nismo mogli kod jednostruko ulančane liste, jer njeni čvorovi nemaju pokazivač na prev

Dvostruko ulančane liste

- Da biste u potpunosti savladali liste, potrebno je da probate da implementirate sve funkcije koje smo radili kod jednostruko ulančanih
- Probajte razne vrste brisanja, zamenu elemenata i slično
- Da li vam funkcija radi proverićete tako što ćete je testirati kako se ponaša kada recimo brišete s početka, iz sredine i s kraja, znači mora da radi za sve granične slučajeve

Kružno povezana lista



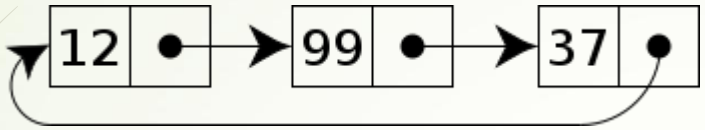
Struktura elementa

```
#include<stdio.h>
#include<stdlib.h>
typedef struct Node
{    int info;
    struct Node *next;
} node;
node
*front=NULL,*rear=NULL,*t
emp;
```

Objašnjenje:

- Kod kružno povezane jednostruke liste, kraj liste ne pokazuje na null, već na početak
- Kada je lista prazna i glava (front) i rep (rear) su NULL
- Sve funkcije ići će slično kao kod jednostruke liste, samo nećete pitati da li ste stigli do null, nego do zadnjeg elementa rear

Kružno povezana lista



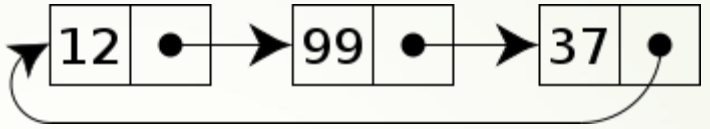
Objašnjenje:

- Slično kao kod kreiranja elementa jednostruko ulančane liste
- Ako je lista prazna, i front i rear postaju novi čvor
- Inače novi čvor dodajete na kraj, i on postaje rep, a njegov sledbenik postaje front

Kreiranje elementa

```
void create()
{  node *newnode;
    newnode=(node*)malloc(sizeof(node));
    printf("\nEnter the node value : ");
    scanf("%d",&newnode->info); newnode->next=NULL;
    if(rear==NULL) front=rear=newnode;
    else  { rear->next=newnode; rear=newnode;
    }
    rear->next=front;
}
```

Kružno povezana lista - brisanje elementa

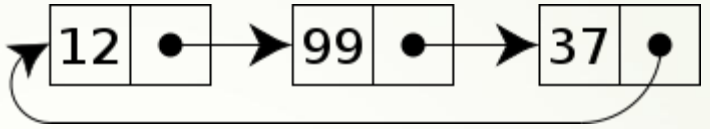



```
void del()
{   temp=front;
    if(front==NULL)
        printf("\nUnderflow :");
    else  {
        if(front==rear)
        {
            printf("\n%d",front-
>info);
            front=rear=NULL; }
```

```
else
        {
            printf("\n%d",front-
>info);
            front=front->next;
            rear->next=front;
        }

    temp->next=NULL;
    free(temp);
    }
}
```

Kružno povezana lista - brisanje elementa




```
void display()
{  temp=front;
    if(front==NULL)
        printf("\nEmpty");
    else
    {  printf("\n");
        for(;temp!=rear;temp=temp->next)
            printf("\n%d address=%u next=%u\t",temp->info,temp,temp->next);
            printf("\n%d address=%u next=%u\t",temp->info,temp,temp-
>next);}}
```

Zadaci za vežbu

- **Zadatak 1.** Brisanje elementa u dvostruko ulančanoj listi:
  a) S početka liste
  b) S kraja liste
  c) Pre zadatog elementa
  d) Nakon zadatog elementa itd. (kao za jednostruko povezanu).
- **Zadatak 2.** Pretvaranje jednostruko ulančane liste u kružno ulančanu i obrnuto.

Uputstva

- Na Moodle-u imate prateće fajlove koje treba da pokrenete u C*u za rad sa lančanim listama
- Pored ovih zadataka, sve zadatke koje smo radili za jednostruku listu, probajte da uradite za obe liste
- Uz ove vežbe ide i prateći test, koji treba da pošaljete do 6.4. u 14h na apljaskovic@np.ac.rs po uputstvu datom na sajtu univerziteta, a na osnovu kog će se voditi evidencija o tome da li ste gradivo savladali ili ne
- Na isti mejl se obratite za sve što vam nije jasno u vezi ove i prethodnih nastavnih jedinica

Korisni linkovi

- https://www.geeksforgeeks.org/data-structures/linked-list/singly-linked-list/
- https://www.geeksforgeeks.org/doubly-linked-list/
- https://www.geeksforgeeks.org/circular-linked-list/
- https://visualgo.net/en/list?slide=1

Kraj tematske celine