Structuri de date și algoritmi

Universitatea Politehnica Timișoara
P-ța Victoriei nr. 2
RO 300006 - Timișoara
Tel: +4 0256 403000
Fax: +4 0256 403021
rector@rectorat.upt.ro
www.upt.ro

Domeniul de studii: Informatică/ Specializarea: Informatică

SDA – Cursul 7

Ș.l. dr.ing. Adriana ALBU
adriana.albu@upt.ro
http://www.aut.upt.ro/~adrianaa

Universitatea
Politehnica
Timişoara

P-ţa Victoriei nr. 2
RO 300006 - Timişoara
Tel: +4 0256 403000
Fax: +4 0256 403021
rector@rectorat.upt.ro
www.upt.ro

5. Structura de date listă
(partea întâia)

5.1 TDA listă

- Structură de date dinamică, flexibilă
- MM
  - o secvență de 0 sau mai multe elemente aparținând unui tip numit tip de bază
  - $a_1, a_2, \ldots, a_n$; a_i nodurile listei
  - n – lungimea listei; n>=0
  - dacă n>=1
    - a_1 – primul nod
    - a_n – ultimul nod
  - ordonată liniar funcție de poziția nodurilor
    - a_i precede pe a_{i+1}
    - a_i succede pe a_{i-1}

- Notații:
  - TipLista l;
  - TipPozitie p;
  - TipNod x;
- Operatori:
  - inserează nod în listă
  - șterge nod din listă
  - caută nod în listă
  - următorul nod în listă
  - primul nod în listă
  - ListaVida(l);
  - InsertieInceput(x, l);
  - InsertieDupa(x, l, p);

5.2 Tehnici de implementare / a. Tablouri

```
#define LungMax …
typedef … TipNod;//în funcție de ce se stochează
typedef int TipIndice;
typedef struct{
    TipNod noduri[Lungmax];
    TipIndice ultim;
} TipLista;
TipLista Lista;
```

5.2 Tehnici de implementare / b. Cursori

```
#define LungMax …
typedef … TipNod;
typedef int TipCursor;
typedef TipCursor TipLista;
typedef struct{
    TipNod nod_lista;
    TipCursor urm;
}TipCelula;
TipCelula zona[Lungmax];
TipLista L, M, Disponibil;
/*L, M, Disponibil sunt
intrarile in diferite liste*/
```

- E specifică limbajelor care nu dispun de pointeri
- Într-un tablou se pot grupa mai multe liste care conțin același tip de elemente
- Operații :
  - Inserare:
    - se suprimă prima locație din Disponibil
    - se înlănțuie în listă pe poziția dorită
  - Ștergere:
    - se suprimă din listă și se inserează în Disponibil

5.2 Tehnici de implementare / c. Pointeri

```
typedef struct nod{
    int cheie;
    struct nod *urm;
    … info;
}TipNod;
typedef struct nod *TipPointerNod;
typedef TipPointerNod TipLista;
TipLista prim;
```

- Operații cu liste înlănțuite:
  - Inserare
    - la început, la sfârșit, după nodul curent, la poziția nodului curent etc.
  - Ștergere
    - nodul următor nodului curent, nodul curent etc.
  - Traversare liste

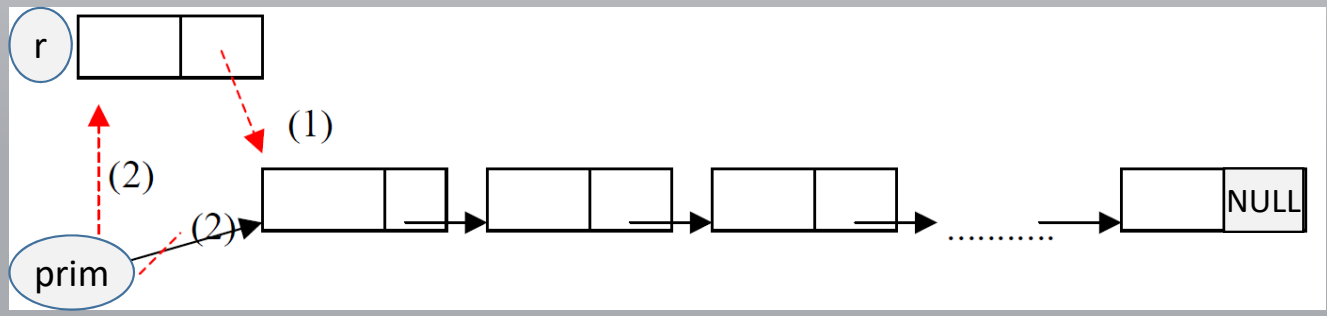
5.3 Tehnici utilizând structura de date listă simplu înlănțuită

a) Inserarea unui nod la început (capul listei)
b) Inserarea unui nod la sfârșit (coada listei)
c) Inserarea unui nod în interiorul listei:
   - după un anumit nod (se cunoaște predecesorul)
   - înaintea unui anumit nod (se cunoaște succesorul)
d) Crearea unei liste:
   - prin inserări repetate la început
   - prin inserări repetate la sfârșit
   - prin inserări într-un anumit punct, astfel încât să se păstreze lista ordonată
e) Ștergerea unui nod:
   - când se cunoaște predecesorul
   - ștergerea nodului curent
f) Traversarea listei
g) Tehnica celor doi pointeri

5.3 a) Inserarea unui nod la început (capul listei)

```
typedef … t_date;
struct nod{
    t_date date;
    struct nod *urm;
}*prim,*q,*r,*ultim;

r=(struct nod *)malloc(sizeof(struct nod));
r->urm=prim;//(1)
prim=r;//(2)
```



5.3 b) Inserarea unui nod la sfârșit (coada listei)

- Varianta listei accesată prin `prim`
  - necesită parcurgerea listei până la sfârșit

```
for(q=prim;q->urm!=NULL;q=q->urm);
r=(struct nod *)malloc(sizeof(struct nod));
r->urm=NULL;
q->urm=r;/*nu functioneaza in cazul listei vide*/
```

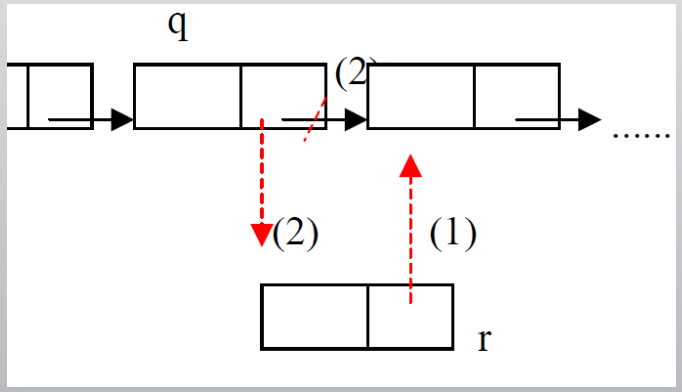
- Varianta listei accesată prin `prim` și `ultim`:

```
r=(struct nod *)malloc(sizeof(struct nod));
r->urm=NULL;
if(ultim==NULL)
    prim=ultim=r;
else{
    ultim->urm=r;
    ultim=r;
}
```

5.3 c) Inserarea unui nod în interiorul listei

➢după un anumit nod (*q) (se cunoaște predecesorul)

```
r=(struct nod *)malloc(sizeof(struct nod));
r->urm=q->urm; //(1)
q->urm=r; //(2)
```



➢înaintea unui anumit nod (*q) (se cunoaște succesorul)

```
r=(struct nod *)malloc(sizeof(struct nod));
*r=*q; /*inclusiv campul de inlantuire*/
q->urm=r;/*se va completa nodul q cu informatia
dorita*/
```

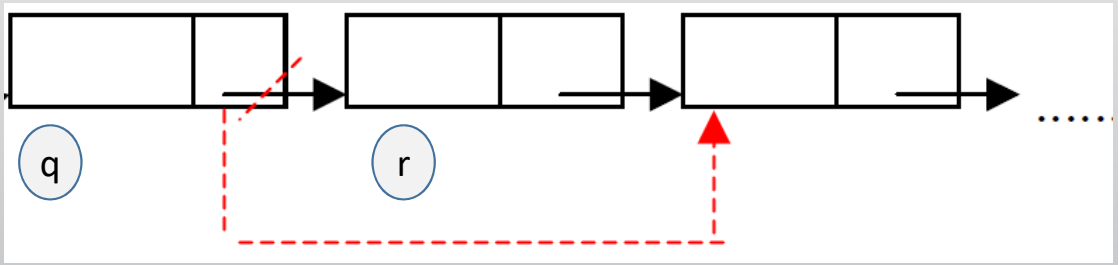
5.3 d) Crearea unei liste simplu înlănțuite

- prin inserări repetate la sfârșit
  - rezultă o listă în ordina inserării nodurilor
- prin inserări repetate la început
  - rezultă o listă în ordine inversă inserării nodurilor
- prin inserări într-un anumit punct
  - se păstrează lista ordonată

5.3 e) Ștergerea unui nod

- când se cunoaște predecesorul nodului (*q)

```
r=q->urm;
q->urm=r->urm;
free(r);
```



- ștergerea nodului curent (*q)

```
r=q->urm;
*q=*r; /*inclusiv campul de inlantuire*/
free(r);
```

- Obs. q->urm !=NULL

5.3 f) Traversarea listei

```
void procesare(struct nod *…)
{
…
}
…
for(q=prim;q!=NULL;q=q->urm)
    procesare(q);
```

5.3 g) Tehnica celor doi pointeri

- Crearea unei liste ordonate prin tehnica celor doi pointeri
  - `q2` îl precede pe `q1`

- Cei doi pointeri avansează simultan până când cheia lui `q1` devine mai mare sau egală cu `x` – nodul care se inserează
  - nodul `x` se inserează după `q2`

Universitatea
Politehnica
Timişoara

P-ţa Victoriei nr. 2
RO 300006 - Timişoara
Tel: +4 0256 403000
Fax: +4 0256 403021
rector@rectorat.upt.ro
www.upt.ro

Vă mulțumesc!