Liste simplu înlănțuite

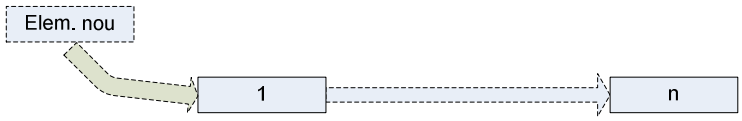
Operațiile care se pot efectua asupra listelor simplu înlănțuite sunt: adăugarea unui element la listă, ştergerea unui element din listă şi căutarea după anumite criterii. Pentru toate aceste operații se consideră că există o variabilă, *p*, care memorează începutul listei (adresa primului element), iar elementele sunt definite prin următoarea structură:

```
struct element

{int data;

struct element *next;

} *p, *q, *aux, el;
```

unde *p* va fi utilizată pentru memorarea primului element, *q* va fi utilizat pentru parcurgerea listei iar *aux* se va folosi la crearea unui nou element în listă; *el* este utilizat doar pentru a avea dimensiunea unui element, necesară pentru funcția malloc.

1. Adăugarea unui element la începutul listei

Pentru a insera un element la începutul listei trebuie alocat spațiu de memorie pentru un nou element şi creată legătura între noul element şi primul element din listă. Apoi trebuie mutat începutul listei pentru a indica noul element.

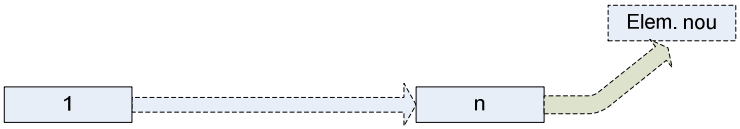


Instrucțiunile sunt:

```
aux = (element *) malloc (sizeof(el));

cin << aux->data;

aux->next = p;

p=aux;
```

2. Adăugarea unui element la sfârşitul listei

Pentru această operație trebuie parcursă lista până la ultimul element care va fi legat de un element nou creat.

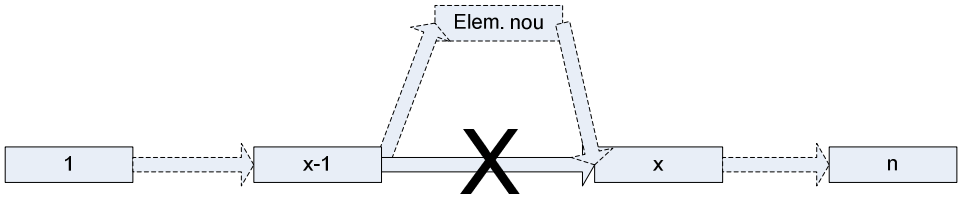


Instrucțiunile sunt:

```
q = p;
while (q->next != NULL)
      q = q->next;
aux = (element *) malloc (sizeof(el));
cin << aux->data;
aux->next = NULL;
q->next = aux;
```

3. Adăugarea unui element înaintea unui element care îndeplineşte o anumită condiție

Pentru această operație trebuie parcursă lista până la găsirea elementui căutat, astfel încât indicatorul să se oprească pe elementul anterior. Apoi trebuie creat un nou element şi se vor crea legăturile (elementul indicat este legat de elementul nou şi elementul nou este legat de următorul element din listă). Situația trebuie tratată în mod diferit pentru primul element față de celelalte, deoarece inserarea unui element înaintea primului element al listei presupune mutarea începutului listei.



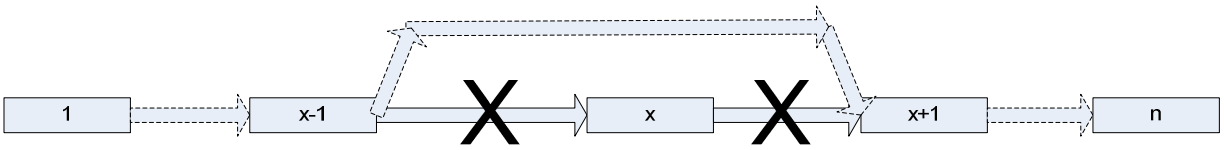
Instrucțiunile sunt:

```
cout << “Introduceti elementul din lista”;
cin >> n;
if (p->data == n)
      {aux = (element *) malloc (sizeof(el));
```

```
      cin << aux->data;
      aux->next = p;
      p=aux;}
else
      {q = p;
      while (q->next->data != n) //dacă data din elementul următor
este egală cu n
            q = q->next;
      aux = (element *) malloc (sizeof(el));
      cin << aux->data;
      aux->next = q->next;
      q->next = aux;
      }
```

4. Stergerea unui element din listă

Presupune crearea legăturii între elementul anterior elementului şters şi cel următor. Se va face distincție între primul element şi celelalte elemente din listă.



Instrucțiunile sunt:

```
cout << “Introduceti elementul pe care doriti sa il stergeti”;
cin >> n;
if (p->data == n)
      {aux = p;
      p = p->next;
      free(aux);
      }
else
```

```
{q = p;
while (q->next->data != n) //dacă data din elementul următor este egală cu n
      q = q->next;
aux = q->next;
q->next = q->next->next;
free(aux);
}
```

Liste dublu înlănțuite

Listele dublu înlănțuite prezintă avantajul legării fiecărui element atât cu elementul următor cât şi cu cel anterior, astfel încât parcurgerea listei se poate face în ambele sensuri.

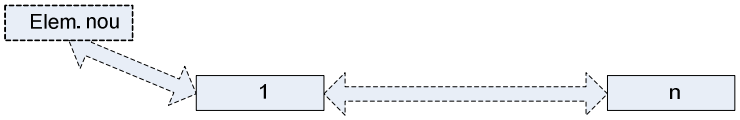
Operațiile care se pot efectua asupra listelor dublu înlănțuite sunta aceleaşi ca pentru listele simplu înlănțuite, adică: adăugarea unui element la listă, ştergerea unui element din listă şi căutarea după anumite criterii. Pentru toate aceste operații se consideră că există o variabilă, *p*, care memorează începutul listei (adresa primului element), iar elementele sunt definite prin următoarea structură:

```
struct element
{int data;
struct element *next, *prev;
} *p, *q, *aux, el;
```

unde *p* va fi utilizată pentru memorarea primului element, *q* va fi utilizat pentru parcurgerea listei iar *aux* se va folosi la crearea unui nou element în listă; *el* este utilizat doar pentru a avea dimensiunea unui element, necesară pentru funcția malloc.

1. Adăugarea unui element la începutul listei

Pentru a insera un element la începutul listei trebuie alocat spațiu de memorie pentru un nou element şi creată legătura între noul element şi primul element din listă. Apoi trebuie mutat începutul listei pentru a indica noul element.



Instrucțiunile sunt:

```
aux = (element *) malloc (sizeof(el));

cin << aux->data;

aux->next = p;

aux->prev = p->prev;

p->prev = aux;

p=aux;
```

2. Adăugarea unui element la sfârşitul listei

Pentru această operație trebuie parcursă lista până la ultimul element care va fi legat de un element nou creat. Instrucțiunile sunt:

```
q = p;

while (q->next != NULL)

      q = q->next;

aux = (element *) malloc (sizeof(el));

cin << aux->data;

aux->next = NULL;

q->next = aux;

aux->prev = q;
```

3. Adăugarea unui element după unui element care îndeplineşte o anumită condiție

Pentru această operație trebuie parcursă lista până la găsirea elementui căutat. Apoi trebuie creat un nou element şi se vor crea legăturile. Situația trebuie tratată în mod diferit pentru primul element față de celelalte, deoarece inserarea unui element înaintea primului element al listei presupune mutarea începutului listei.

Instrucțiunile sunt:

```
cout << “Introduceti elementul din lista”;
```

```
cin >> n;
if (p->data == n)
      {aux = (element *) malloc (sizeof(el));
      cin << aux->data;
      aux->next = p;
      p->prev = aux;
      p=aux;}
else
      {q = p;
      while (q->data != n)
            q = q->next;
      aux = (element *) malloc (sizeof(el));
      cin << aux->data;
      aux->next = q->next;
      q->next->prev = aux;
      q->next = aux;
      aux->prev = q;
      }
```

4. Stergerea unui element din listă

Presupune crearea legăturii între elementul anterior elementului şters şi cel următor. Se va face distincție între primul element şi celelalte elemente din listă.

```
cout << “Introduceti elementul pe care doriti sa il stergeti”;
cin >> n;
if (p->data == n)
      {aux = p;
      p = p->next;
      free(aux);
      }
```

```
      else
            {q = p;
            while (q->next->data != n) //dacă data din elementul următor
      este egală cu n
                  q = q->next;
            aux = q->next;
            q->next = q->next->next;
            q->next->next->prev = q;
            free(aux);
}
```

Liste circulare simplu înlănțuite

Sunt similare listelor simplu înlănțuite, dar ultimul element este legat de primul. Operațiile sunt similare, iar diferențele apar doar la prelucrarea ultimului element (adăugarea unui element la sfârşitul listei, ştergerea acestuia), precum şi la condiția de oprire la parcurgerea listei, deoarece adresa memorată de ultimul element nu mai este NULL, ci este adresa începutului listei.

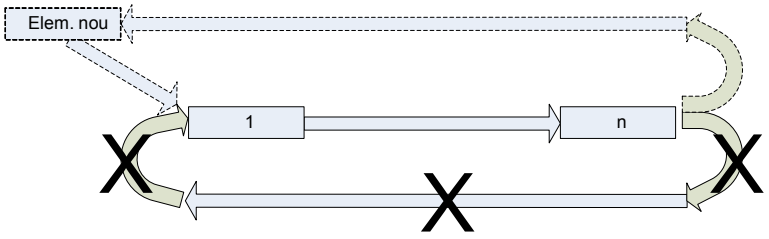
Elementele sunt definite prin următoarea structură:

```
struct element
{int data;
struct element *next;
} *p, *q, *aux, el;
```

unde *p* va fi utilizată pentru memorarea primului element, *q* va fi utilizat pentru parcurgerea listei iar *aux* se va folosi la crearea unui nou element în listă; *el* este utilizat doar pentru a avea dimensiunea unui element, necesară pentru funcția malloc.

1. Adăugarea unui element la începutul listei

Pentru a insera un element la începutul listei trebuie alocat spațiu de memorie pentru un nou element şi creată legătura între noul element şi primul element din listă, precum şi legătura între ultimul element din listă şi noul element creat. Apoi trebuie mutat începutul listei pentru a indica noul element.



Instrucțiunile sunt:

```
q = p;
while (q->next != p)
      q = q->next;          // q va indica ultimul element din lista
aux = (element *) malloc (sizeof(el));
cin << aux->data;
aux->next = p;
q->next = aux;
p=aux;
```

2. Adăugarea unui element la sfârşitul listei

Pentru această operație trebuie parcursă lista până la ultimul element care va fi legat de un element nou creat. Instrucțiunile sunt:

```
q = p;
while (q->next != p)
      q = q->next;
aux = (element *) malloc (sizeof(el));
cin << aux->data;
aux->next = p;
q->next = aux;
```

Liste circulare dublu înlănțuite

Sunt similare listelor dublu înlănțuite, dar primul şi ultimul element sunt legate între ele. Operațiile sunt similare, iar diferențele apar la prelucrarea primului şi ultimului element (adăugarea unui element la început/sfârşit, ştergerea acestuia), precum şi la condiția de oprire la parcurgerea listei, deoarece adresa memorată de ultimul element nu mai este NULL, ci este adresa începutului listei.

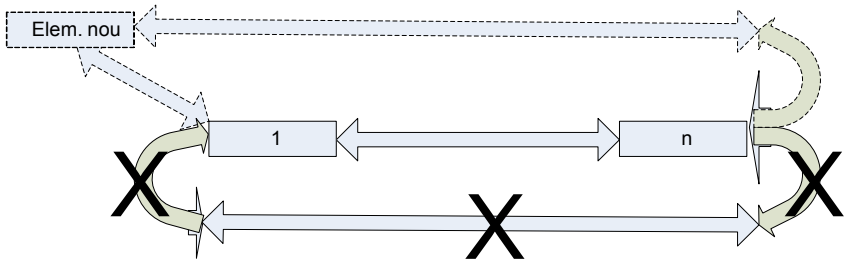
Elementele sunt definite prin următoarea structură:

```
struct element

{int data;

struct element *next, *prev;

} *p, *q, *aux, el;
```

unde *p* va fi utilizată pentru memorarea primului element, *q* va fi utilizat pentru parcurgerea listei iar *aux* se va folosi la crearea unui nou element în listă; *el* este utilizat doar pentru a avea dimensiunea unui element, necesară pentru funcția malloc.

1. Adăugarea unui element la începutul listei

Pentru a insera un element la începutul listei trebuie alocat spațiu de memorie pentru un nou element şi creată legătura între noul element şi primul element din listă, precum şi legătura între ultimul element din listă şi noul element creat. Apoi trebuie mutat începutul listei pentru a indica noul element.



Instrucțiunile sunt:

```
aux = (element *) malloc (sizeof(el));

cin << aux->data;

aux->next = p;

p->prev->next = aux;

aux->prev = p->prev;

p->prev = aux;

p=aux;
```

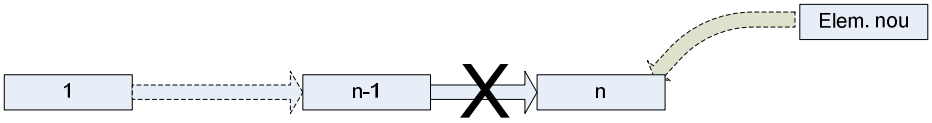
2. Adăugarea unui element la sfârşitul listei

Pentru această operație trebuie parcursă lista până la ultimul element care va fi legat de un element nou creat. Instrucțiunile sunt:

```
q = p;
while (q->next != NULL)
      q = q->next;
aux = (element *) malloc (sizeof(el));
cin << aux->data;
q->next = aux;
aux->prev = q;
aux->next = p;
p->prev = aux;
```

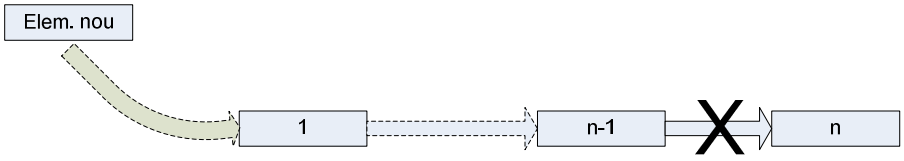
Stive

Stivele reprezintă un caz particular de liste simplu înlănțuite, în care operațiile de adăugare şi de ştergere pot fi efectuate doar asupra ultimului element.



Cozi

Cozile reprezintă un caz particular de liste simplu înlănțuite, în care operațiile de adăugare se realizează la un capăt iar operațiile de ştergere se realizează la celălalt capăt.



TEMĂ

Să se creeze o listă circulară dublu înlănțuită în care se introduc 10 numere citite de la tatatură. Să se efectueze următoarele operații:

a) Adăugarea un nou element la începutul listei
b) Ştergerea primului element par întâlnit
c) Stergerea ultimului număr prim din listă