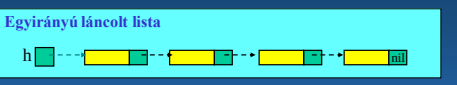
Objektum elvű alkalmazások fejlesztése

Egyirányú láncolt lista

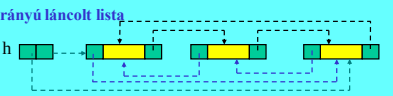
Készítette: Gregorics Tibor
Szabóné Nacsa Rozália

Láncolt adatszerkezetek

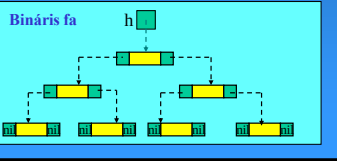
Egyirányú láncolt lista



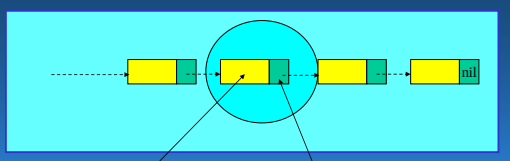
Kétirányú láncolt lista



Bináris fa



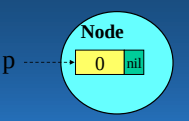
A lista eleme



Listaelem típusa

```
struct Node {
    int   value;
    Node *next;
};
```


Listaelem létrehozása és lebontása



Listaelem típusa

```
struct Node {
    int   value;
    Node *next;

    Node(int i=0, Node *q=NULL)
        :value(i), next(q){}
};
```

Listaelem létrehozása

```
Node *p = new Node();
```

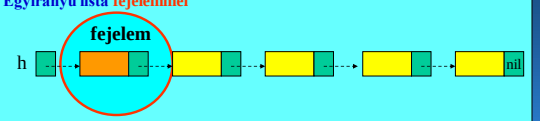
```
Node *p = new Node(3);
```

Listaelem lebontása

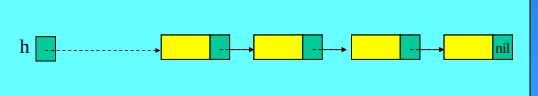
```
delete p;
```


Egyirányú láncolt listák

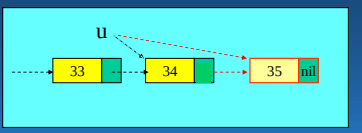
Egyirányú lista fejelemmel



Egyirányú lista fejelem nélkül



Láncolt lista felépítése



Egy újabb elem

```
Node *u;
...
u->next = new Node(35);
u = u->next ;
```

Teljes lista

```
Node *h,*u;
h = ?;
u = ?;
for(int i=1;i<=n;i++){
    u->next = new Node(i);
    u = u->next ;
}
```

Nem: `Node* h,u;`

Láncolt lista felépítése

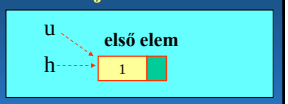
7

Fejelemmel


```
Node *h = new Node();
Node *u = h;

for(int i=1;i<=n;i++){
   u->next = new Node(i);
   u = u->next;
}
```

Fejelem nélkül

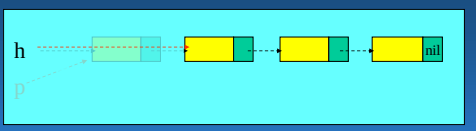



```
Node *h = new Node(1);
Node *u = h;

for(int i=2;i<=n;i++){
   u->next = new Node(i);
   u = u->next;
}
```

Láncolt lista lebontása

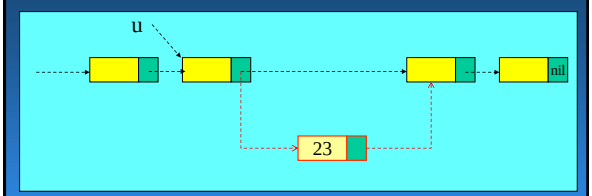
8




```
while(h!=NULL){
   Node *p = h;
   h = p->next;
   delete p;
}
```

Beszúrás adott elem után

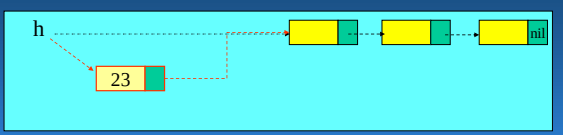
9




```
Node *u;
...
u->next = new Node(23,u->next);
```

Beszúrás fejelem nélküli láncolt lista elejére

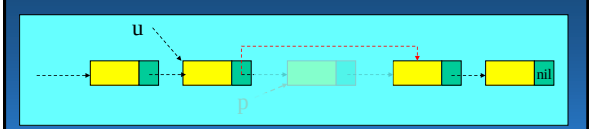
10




```
h = new Node(23,h);
```

Törlés adott elem mögül

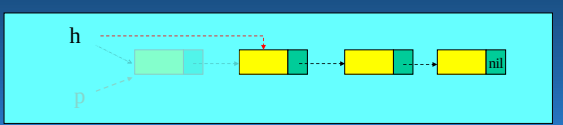
11




```
Node *u;
...
Node *p = u->next;
if(p!=NULL){
   u->next = p->next;
   delete p;
}
```

Törlés fejelem nélküli lista elejéről

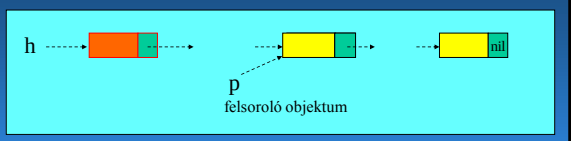
12




```
if(h!=NULL){
   Node *p = h;
   h = h->next;
   delete p;
}
```

Felsoroló objektum láncolt listára

13



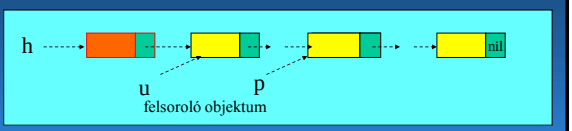

```
t:enor(E)    ~ p
t.Current()  ~ p->value
t.First()    ~ p = h->next
t.End()      ~ p == NULL
t.Next()     ~ p = p->next
```

Fejelem nélküli listában:

```
...
t.First()   ~ p = h
...
```

A felsoroló objektum az előző listaelemet is nyilvántarthatja

14




```
t:enor(E)    ~ Node *p, *u
t.Current()  ~ p->value
t.First()    ~ u = h
               p = h->next
t.End()      ~ p==NULL
t.Next()     ~ u = p
               p =p->next
```

Maximum kiválasztás

15

```
t.First();
int max = f(t.Current());
int elem = t.Current();
for(t.Next(); !t.End(); t.Next()){
   if(max < f(t.Current())){
       max = f(t.Current());
       elem = t.Current();
   }
}
```

Nem az elemet, hanem az azt tartalmazó és azt megelőző listaelem címét tároljuk

```
Node *u = h;
Node *p = h->next;
int max = f(p->value);
Node *cur, *prev;
cur = p; prev = u;
for(u=p, p=p->next; p!=NULL; u=p, p=p->next){
   if(max < f(p->value){
       max = f(p->value);
       cur = p; prev = u;
   }
}
```

Lineáris keresés

16

```
bool l = false;
int elem;
for(t.First(); !l && !t.End(); t.Next()){
   elem = t.Current();
   l = felt(elem);
}
```

Nem az elemet, hanem az azt tartalmazó és azt megelőző listaelem címét tároljuk

```
bool l = false;
Node *cur, *prev;
Node *u, *p;
for(u=h, p=h->next; !l && p!=NULL; u=p, p=p->next){
   cur = p; prev = u;
   l = felt(p->value);
}
```

Feladat

17

Rendezzük növekvő sorrendbe a szabványos bemenetről érkező egész számokat!

Használjunk beszúrásos rendezést!

Programterv

18

- Bemenő adat: *számsorozat (szabványos bemenet)*
- Kimenő adat: *rendezett számsorozat (szabványos kimenet)*
- Segéd adat: *számsorozatot tartalmazó láncolt lista*

kiválasztás programozási tétele

összegzés (másolás!) programozási tétele

1. Egyenként beolvassuk a számokat, és listaelembe ágyazva azokat a rendezettségnek megfelelő helyére beszúrjuk egy fokozatosan épülő láncolt listába.
2. Végigolvasva a láncolt listát kiírjuk annak értékeit (azaz a rendezett sorozatot) miközben a listát lebontjuk.

összegzés (másolás) programozási tétele

19

```
#include <iostream>
#include <iomanip>
using namespace std;

struct Node {
   int   value;
   Node *next;
   Node(int i=0, Node *q=NULL) :value(i), next(q){}
};
Node* keres(Node *h, int e);

int main()
{
   // Beolvasás és lista-építés
   ...
   // Kiírás és lista-lebontás
   ...
   char ch; cin >> ch;
   return 0;
}

Node* keres(Node *h, int e){ ... }
```

main.cpp

20

```
// Beolvasás és lista-építés

   cout << "Kérem a rendezendő egész számokat"
        << endl;

   Node *h = new Node();
   int i;

   while( cin >> i ){
      Node *u = keres(h,i);
      u->next = new Node(i,u->next);
   }
```

main.cpp

21

kiválasztás

```
for(t.First(); !t.End() && !felt(t.Current()); t.Next);
```

```
Node* keres(Node *h, int e)
{
   Node *u, *p;
   for(u = h, p = h->next;
       p!=NULL && p->value<e;
       u = p, p = p->next);
   return u;
}
```

lusta kiértékelést feltételezve

main.cpp

22

```
// Kiírás és lista-lebontás

   cout << "A rendezett számsorozat:"  << endl;

   Node *p = h;
   h = h->next;
   delete p;
   while(h!=NULL){
      p = h;
      h = h->next;
      cout << setw(5) << p->value;
      delete p;
   }
   cout << endl;
```

main.cpp