Programozás
(GKxB_INTM114)

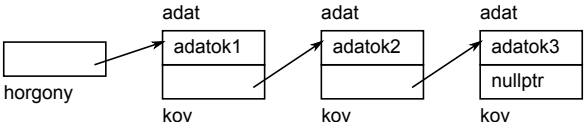
Dr. Hatwagner F. Miklós

Széchenyi István Egyetem, Győr

https://github.com/wajzy/GKxB_INTM114.git

2024. május 22.

Verem megvalósítható tömbbel, de annak mérete véges, fordítási időben adott.
Lehetséges megoldás: láncolt lista (Linked List, önhivatkozó adatszerkezet)



Egyszeresen láncolt lista (Singly Linked List)

Felhasználható struktúra

```
struct Lista1 {
    ADAT adat;
    Lista1 *kov;
};
```

veremTeszt2.cpp

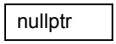
```
1  #include <iostream>
2  #include "verem2.h"
3  using namespace std;
4
5  #define N 5
6  int main() {
7    cout << N << " egesz verembe rakasa: ";
8    for(int i=0; i<N; i++) {
9      cout << i << '\t';
10     berak(i);
11   }
12   cout << "\nVisszaolvasva:\t\t";
13   while(not ures()) {
14     cout << kivesz() << '\t';
15   }
16   cout << endl;
17   return 0;
18 }
```

verem2.h

```
4  struct Lista1 {
5    int adat;
6    Lista1* kov;
7  };
```

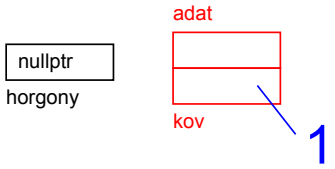
verem2.cpp

```
6  bool berak(int adat) {
7    Lista1* uj = new Lista1;
8    if(uj != nullptr) {
9      uj->adat = adat;
10     uj->kov = horgony;
11     horgony = uj;
12     return true;
13   } else {
14     return false;
15   }
16 }
```



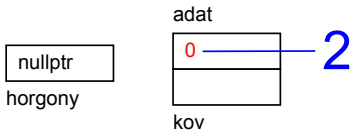
verem2.cpp

```
6  bool berak(int adat) {
7    Lista1* uj = new Lista1;
8    if(uj != nullptr) {
9      uj->adat = adat;
10     uj->kov = horgony;
11     horgony = uj;
12     return true;
13   } else {
14     return false;
15   }
16 }
```



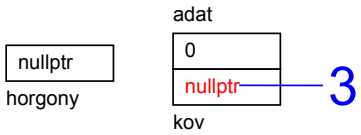
verem2.cpp

```
6  bool berak(int adat) {
7    Lista1* uj = new Lista1;
8    if(uj != nullptr) {
9      uj->adat = adat;
10     uj->kov = horgony;
11     horgony = uj;
12     return true;
13   } else {
14     return false;
15   }
16 }
```



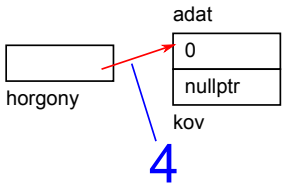
verem2.cpp

```
6  bool berak(int adat) {
7    Lista1* uj = new Lista1;
8    if(uj != nullptr) {
9      uj->adat = adat;
10     uj->kov = horgony;
11     horgony = uj;
12     return true;
13   } else {
14     return false;
15   }
16 }
```



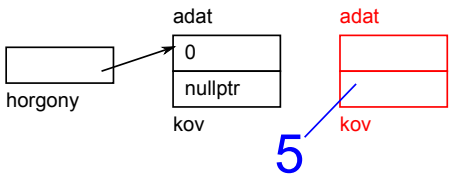
verem2.cpp

```
6  bool berak(int adat) {
7    Lista1* uj = new Lista1;
8    if(uj != nullptr) {
9      uj->adat = adat;
10     uj->kov = horgony;
11     horgony = uj;
12     return true;
13   } else {
14     return false;
15   }
16 }
```



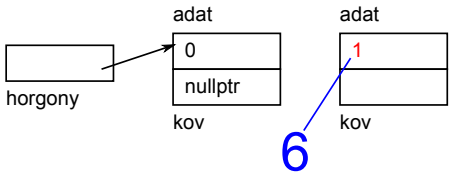
verem2.cpp

```
6  bool berak(int adat) {
7    Lista1* uj = new Lista1;
8    if(uj != nullptr) {
9      uj->adat = adat;
10     uj->kov = horgony;
11     horgony = uj;
12     return true;
13   } else {
14     return false;
15   }
16 }
```



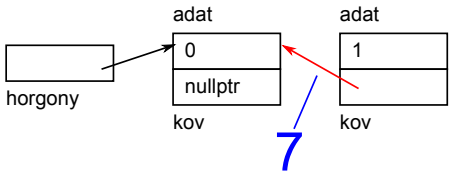
verem2.cpp

```
6  bool berak(int adat) {
7    Lista1* uj = new Lista1;
8    if(uj != nullptr) {
9      uj->adat = adat;
10     uj->kov = horgony;
11     horgony = uj;
12     return true;
13   } else {
14     return false;
15   }
16 }
```



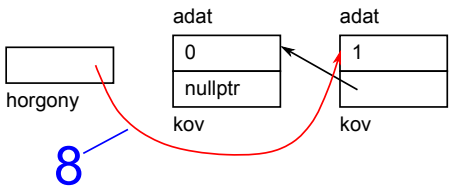
verem2.cpp

```
6  bool berak(int adat) {
7    Lista1* uj = new Lista1;
8    if(uj != nullptr) {
9      uj->adat = adat;
10     uj->kov = horgony;
11     horgony = uj;
12     return true;
13   } else {
14     return false;
15   }
16 }
```



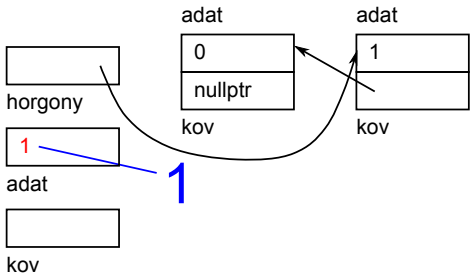
verem2.cpp

```
6  bool berak(int adat) {
7    Lista1* uj = new Lista1;
8    if(uj != nullptr) {
9      uj->adat = adat;
10     uj->kov = horgony;
11     horgony = uj;
12     return true;
13   } else {
14     return false;
15   }
16 }
```



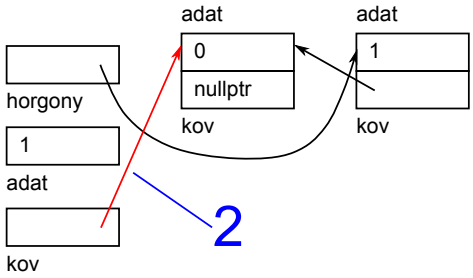
verem2.cpp

```
19 int kivesz() {
20   if(horgony == nullptr) {
21     std::cerr<<"A verem ures.\n";
22     return 0;
23   } else {
24     int adat = horgony->adat;
25     Lista1* kov = horgony->kov;
26     delete horgony;
27     horgony = kov;
28     return adat;
29   }
30 }
```



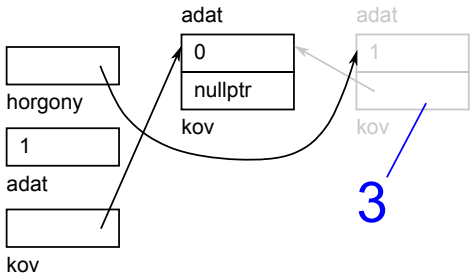
verem2.cpp

```
19 int kivesz() {
20   if(horgony == nullptr) {
21     std::cerr<<"A verem ures.\n";
22     return 0;
23   } else {
24     int adat = horgony->adat;
25     Lista1* kov = horgony->kov;
26     delete horgony;
27     horgony = kov;
28     return adat;
29   }
30 }
```



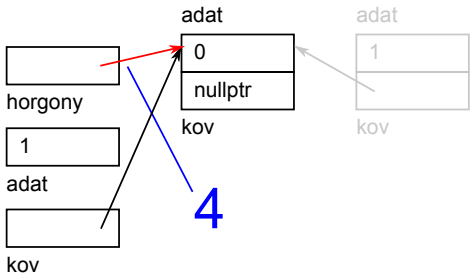
verem2.cpp

```
19  int kivesz() {
20    if(horgony == nullptr) {
21      std::cerr<<"A verem ures.\n";
22      return 0;
23    } else {
24      int adat = horgony->adat;
25      Lista1* kov = horgony->kov;
26      delete horgony;
27      horgony = kov;
28      return adat;
29    }
30  }
```



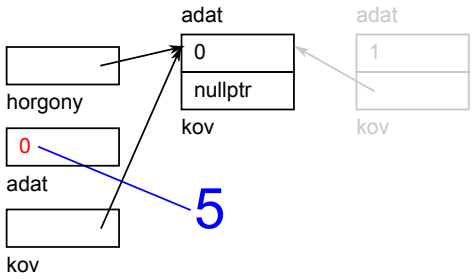
verem2.cpp

```
19  int kivesz() {
20    if(horgony == nullptr) {
21      std::cerr<<"A verem ures.\n";
22      return 0;
23    } else {
24      int adat = horgony->adat;
25      Lista1* kov = horgony->kov;
26      delete horgony;
27      horgony = kov;
28      return adat;
29    }
30  }
```



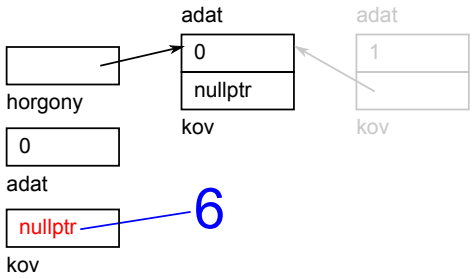
verem2.cpp

```
19 int kivesz() {
20   if(horgony == nullptr) {
21     std::cerr<<"A verem ures.\n";
22     return 0;
23   } else {
24     int adat = horgony->adat;
25     Lista1* kov = horgony->kov;
26     delete horgony;
27     horgony = kov;
28     return adat;
29   }
30 }
```



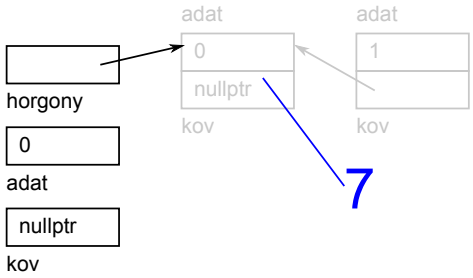
verem2.cpp

```
19 int kivesz() {
20   if(horgony == nullptr) {
21     std::cerr<<"A verem ures.\n";
22     return 0;
23   } else {
24     int adat = horgony->adat;
25     Lista1* kov = horgony->kov;
26     delete horgony;
27     horgony = kov;
28     return adat;
29   }
30 }
```



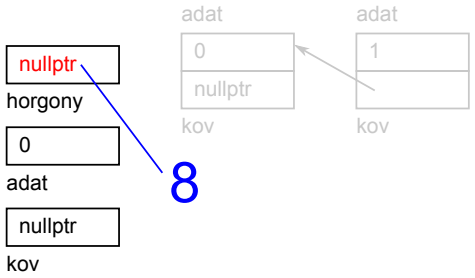
verem2.cpp

```
19 int kivesz() {
20   if(horgony == nullptr) {
21     std::cerr<<"A verem ures.\n";
22     return 0;
23   } else {
24     int adat = horgony->adat;
25     Lista1* kov = horgony->kov;
26     delete horgony;
27     horgony = kov;
28     return adat;
29   }
30 }
```



verem2.cpp

```
19 int kivesz() {
20   if(horgony == nullptr) {
21     std::cerr<<"A verem ures.\n";
22     return 0;
23   } else {
24     int adat = horgony->adat;
25     Lista1* kov = horgony->kov;
26     delete horgony;
27     horgony = kov;
28     return adat;
29   }
30 }
```



Készítsünk általánosan használható függvényeket egyszeresen láncolt lista manipulálásához!

listaTeszt1.cpp (Lista1.cpp, Lista1.h)

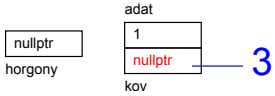
```
1  #include <iostream>
2  #include "Lista1.h"
3  using namespace std;
4
5  int main() {
6    cout << "Adjon meg egeszeket, leallas negativ szamra!\n";
7    Lista1 *horgony=nullptr, *seged=nullptr;
8    int szam;
9    while(cin>>szam, szam>=0) {
10     seged = beszur1(szam, seged);
11     if(horgony == nullptr) horgony = seged;
12   }
13   cout << "Ezeket adta meg:\n";
14   kiir1(horgony);
15   torolMindet1(horgony);
16   return 0;
17 }
```

Lista1.cpp

```
5  // 'elozo' utan beszur egy uj elemet
6  Lista1 *beszur1(int adat, Lista1 *elozo) {
7    Lista1* uj = new Lista1;
8    if(uj) { // if(uj != nullptr) { //...
9      uj->adat = adat;
10     if(elozo) {
11       uj->kov = elozo->kov;
12       elozo->kov = uj;
13     } else {
14       uj->kov = nullptr;
15     }
16   }
17   return uj;
18 }
```

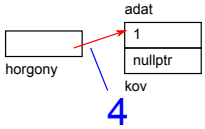
Lista1.cpp

```
5  // 'elozo' utan beszur egy uj elemet
6  Lista1 *beszur1(int adat, Lista1 *elozo) {
7    Lista1* uj = new Lista1;
8    if(uj) { // if(uj != nullptr) { //...
9      uj->adat = adat;
10     if(elozo) {
11       uj->kov = elozo->kov;
12       elozo->kov = uj;
13     } else {
14       uj->kov = nullptr;
15     }
16   }
17   return uj;
18 }
```



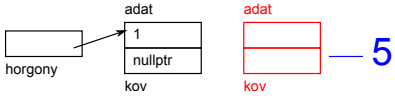
Lista1.cpp

```
5  // 'elozo' utan beszur egy uj elemet
6  Lista1 *beszur1(int adat, Lista1 *elozo) {
7    Lista1* uj = new Lista1;
8    if(uj) { // if(uj != nullptr) { //...
9      uj->adat = adat;
10     if(elozo) {
11       uj->kov = elozo->kov;
12       elozo->kov = uj;
13     } else {
14       uj->kov = nullptr;
15     }
16   }
17   return uj;
18 }
```



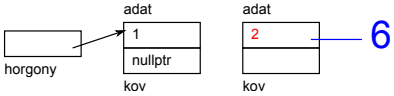
listaTeszt1.cpp

```
6     cout << "Adjon meg egeszeket, leallas negativ szamra!\n";
7     Lista1 *horgony=nullptr, *seged=nullptr;
8     int szam;
9     while(cin>>szam, szam>=0) {
10      seged = beszur1(szam, seged);
11      if(horgony == nullptr) horgony = seged;
12    }
13    cout << "Ezeket adta meg:\n";
14    kiir1(horgony);
15    torolMindet1(horgony);
16    return 0;
17  }
```



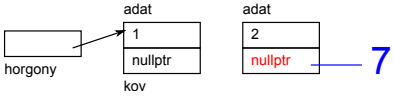
Lista1.cpp

```
5  // 'elozo' utan beszur egy uj elemet
6  Lista1 *beszur1(int adat, Lista1 *elozo) {
7    Lista1* uj = new Lista1;
8    if(uj) { // if(uj != nullptr) { //...
9      uj->adat = adat;
10     if(elozo) {
11       uj->kov = elozo->kov;
12       elozo->kov = uj;
13     } else {
14       uj->kov = nullptr;
15     }
16   }
17   return uj;
18 }
```



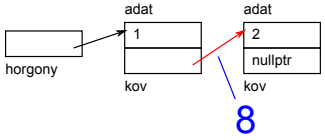
Lista1.cpp

```
5  // 'elozo' utan beszur egy uj elemet
6  Lista1 *beszur1(int adat, Lista1 *elozo) {
7    Lista1* uj = new Lista1;
8    if(uj) { // if(uj != nullptr) { //...
9      uj->adat = adat;
10     if(elozo) {
11       uj->kov = elozo->kov;
12       elozo->kov = uj;
13     } else {
14       uj->kov = nullptr;
15     }
16   }
17   return uj;
18 }
```



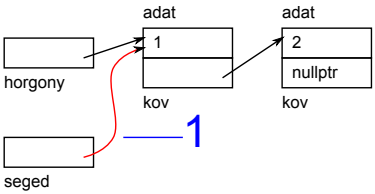
Lista1.cpp

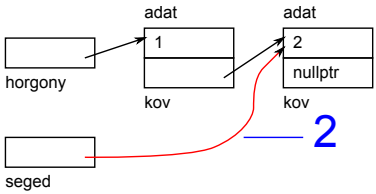
```
5  // 'elozo' utan beszur egy uj elemet
6  Lista1 *beszur1(int adat, Lista1 *elozo) {
7    Lista1* uj = new Lista1;
8    if(uj) { // if(uj != nullptr) { //...
9      uj->adat = adat;
10     if(elozo) {
11       uj->kov = elozo->kov;
12       elozo->kov = uj;
13     } else {
14       uj->kov = nullptr;
15     }
16   }
17   return uj;
18 }
```

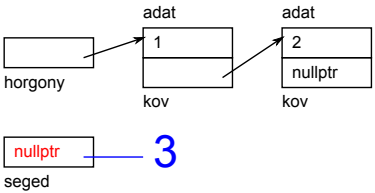


listaTeszt1.cpp

```
6     cout << "Adjon meg egeszeket, leallas negativ szamra!\n";
7     Lista1 *horgony=nullptr, *seged=nullptr;
8     int szam;
9     while(cin>>szam, szam>=0) {
10      seged = beszur1(szam, seged);
11      if(horgony == nullptr) horgony = seged;
12    }
13    cout << "Ezeket adta meg:\n";
14    kiir1(horgony);
15    torolMindet1(horgony);
16    return 0;
17  }
```

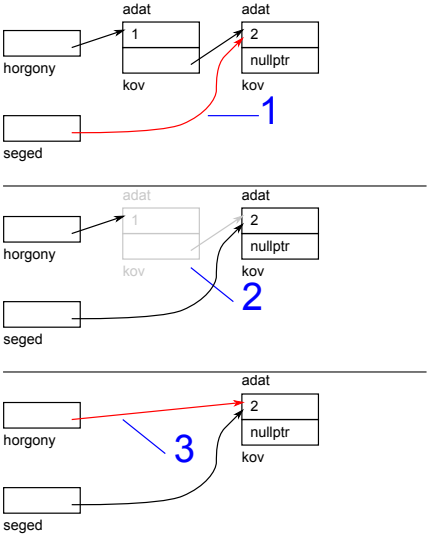






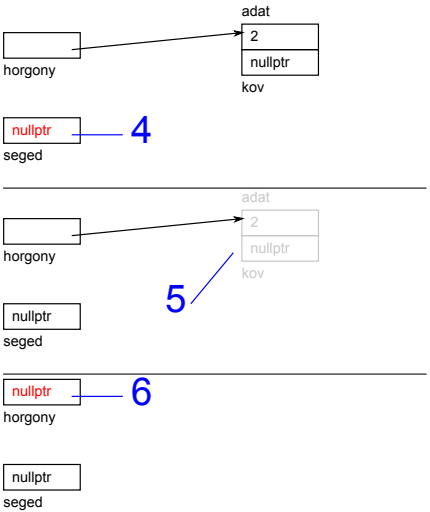
Lista1.cpp

```
// kiirja a lista osszes elemet                    20
void kiir1(Lista1 *horgony) {                      21
  Lista1 *seged;                                   22
  for(seged=horgony; seged;                        23
      seged=seged->kov) {                          24
    std::cout << seged->adat                       25
              << '\t';                             26
  }                                                27
```



Lista1.cpp

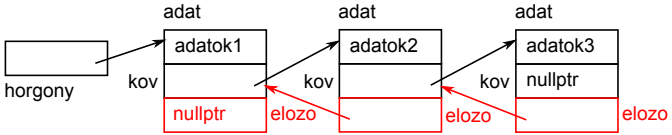
```
// torli a teljes listat                    38
void torolMindet1(Lista1 *horgony) {        39
  while(horgony) {                          40
    Lista1 *seged = horgony->kov;           41
    delete horgony;                         42
    horgony = seged;                        43
  }                                         44
}                                           45
```



Lista1.cpp

```
// torli a teljes listat                    38
void torolMindet1(Lista1 *horgony) {        39
  while(horgony) {                          40
    Lista1 *seged = horgony->kov;           41
    delete horgony;                         42
    horgony = seged;                        43
  }                                         44
}                                           45
```

Készítsük el a verem mintájára a sor megvalósítását is láncolt listával!
Probléma: a lista utolsó elemének eltávolítása nehézkes.
Megoldás: kétszeresen láncolt lista.



Kétszeresen láncolt lista (Doubly Linked List)

Felhasználható struktúra

```
struct Lista2 {
    ADAT adat;
    Lista2 *elozo, *kov;
};
```

sorTeszt2.cpp

```
1  #include <iostream>
2  #include "sor2.h"
3  using namespace std;
4
5  int main() {
6    berak(1); berak(2); berak(3); berak(4);
7    cout << kivesz() << '\n';
8    cout << kivesz() << '\n';
9    berak(6);
10   cout << kivesz() << '\n';
11   cout << kivesz() << '\n';
12   cout << kivesz() << '\n';
13   // nincs mit kivenni
14   cout << kivesz() << '\n';
15   return 0;
16 }
```

sor2.h

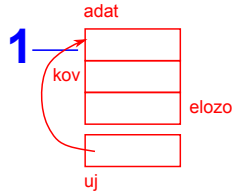
```
struct Lista2 {          4
  int adat;              5
  Lista2 *elozo, *kov;   6
};                       7
```

sor2.cpp

```
4  static Lista2* eleje = nullptr;
5  static Lista2* vege  = nullptr;
6
7  bool berak(int adat) {
8    Lista2* uj = new Lista2;
9    if(uj != nullptr) {
10     uj->adat = adat;
11     uj->elozo = nullptr;
12     uj->kov = eleje;
13     if(eleje != nullptr) {
14       eleje->elozo = uj;
15     }
16     eleje = uj;
17     if(vege == nullptr) {
18       vege = uj;
19     }
20     return true;
21   } else {
22     return false;
23   }
24 }
```

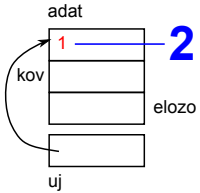
sor2.cpp

```
4  static Lista2* eleje = nullptr;
5  static Lista2* vege  = nullptr;
6
7  bool berak(int adat) {
8    Lista2* uj = new Lista2;
9    if(uj != nullptr) {
10     uj->adat = adat;
11     uj->elozo = nullptr;
12     uj->kov = eleje;
13     if(eleje != nullptr) {
14       eleje->elozo = uj;
15     }
16     eleje = uj;
17     if(vege == nullptr) {
18       vege = uj;
19     }
20     return true;
21   } else {
22     return false;
23   }
24 }
```



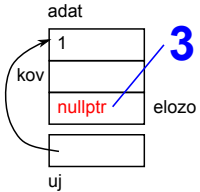
sor2.cpp

```
4  static Lista2* eleje = nullptr;
5  static Lista2* vege  = nullptr;
6
7  bool berak(int adat) {
8    Lista2* uj = new Lista2;
9    if(uj != nullptr) {
10     uj->adat = adat;
11     uj->elozo = nullptr;
12     uj->kov = eleje;
13     if(eleje != nullptr) {
14       eleje->elozo = uj;
15     }
16     eleje = uj;
17     if(vege == nullptr) {
18       vege = uj;
19     }
20     return true;
21   } else {
22     return false;
23   }
24 }
```



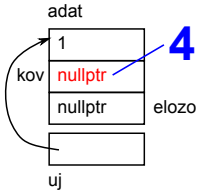
sor2.cpp

```
4  static Lista2* eleje = nullptr;
5  static Lista2* vege  = nullptr;
6
7  bool berak(int adat) {
8    Lista2* uj = new Lista2;
9    if(uj != nullptr) {
10     uj->adat = adat;
11     uj->elozo = nullptr;
12     uj->kov = eleje;
13     if(eleje != nullptr) {
14       eleje->elozo = uj;
15     }
16     eleje = uj;
17     if(vege == nullptr) {
18       vege = uj;
19     }
20     return true;
21   } else {
22     return false;
23   }
24 }
```



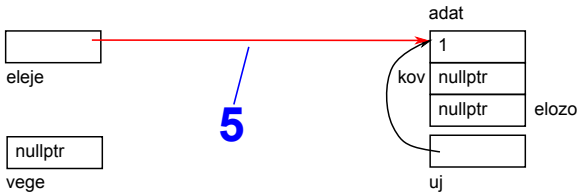
sor2.cpp

```
4  static Lista2* eleje = nullptr;
5  static Lista2* vege  = nullptr;
6
7  bool berak(int adat) {
8    Lista2* uj = new Lista2;
9    if(uj != nullptr) {
10     uj->adat = adat;
11     uj->elozo = nullptr;
12     uj->kov = eleje;
13     if(eleje != nullptr) {
14       eleje->elozo = uj;
15     }
16     eleje = uj;
17     if(vege == nullptr) {
18       vege = uj;
19     }
20     return true;
21   } else {
22     return false;
23   }
24 }
```



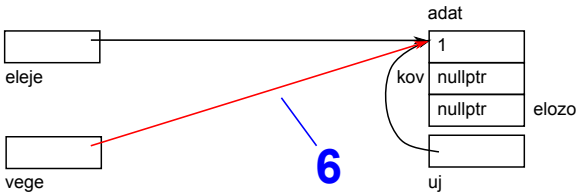
sor2.cpp

```
4  static Lista2* eleje = nullptr;
5  static Lista2* vege  = nullptr;
6
7  bool berak(int adat) {
8    Lista2* uj = new Lista2;
9    if(uj != nullptr) {
10     uj->adat = adat;
11     uj->elozo = nullptr;
12     uj->kov = eleje;
13     if(eleje != nullptr) {
14       eleje->elozo = uj;
15     }
16     eleje = uj;
17     if(vege == nullptr) {
18       vege = uj;
19     }
20     return true;
21   } else {
22     return false;
23   }
24 }
```



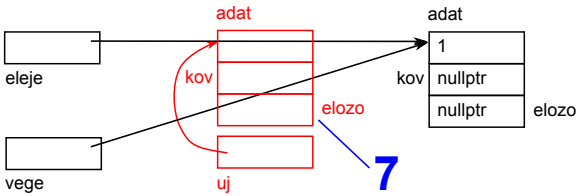
sor2.cpp

```
4  static Lista2* eleje = nullptr;
5  static Lista2* vege  = nullptr;
6
7  bool berak(int adat) {
8    Lista2* uj = new Lista2;
9    if(uj != nullptr) {
10     uj->adat = adat;
11     uj->elozo = nullptr;
12     uj->kov = eleje;
13     if(eleje != nullptr) {
14       eleje->elozo = uj;
15     }
16     eleje = uj;
17     if(vege == nullptr) {
18       vege = uj;
19     }
20     return true;
21   } else {
22     return false;
23   }
24 }
```



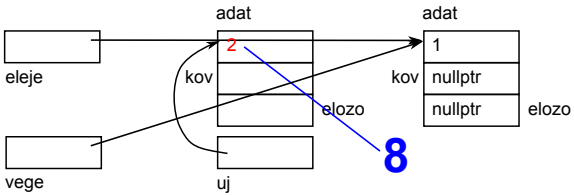
sor2.cpp

```
4  static Lista2* eleje = nullptr;
5  static Lista2* vege  = nullptr;
6
7  bool berak(int adat) {
8    Lista2* uj = new Lista2;
9    if(uj != nullptr) {
10     uj->adat = adat;
11     uj->elozo = nullptr;
12     uj->kov = eleje;
13     if(eleje != nullptr) {
14       eleje->elozo = uj;
15     }
16     eleje = uj;
17     if(vege == nullptr) {
18       vege = uj;
19     }
20     return true;
21   } else {
22     return false;
23   }
24 }
```



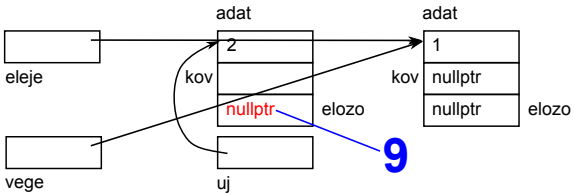
sor2.cpp

```
4  static Lista2* eleje = nullptr;
5  static Lista2* vege  = nullptr;
6
7  bool berak(int adat) {
8    Lista2* uj = new Lista2;
9    if(uj != nullptr) {
10     uj->adat = adat;
11     uj->elozo = nullptr;
12     uj->kov = eleje;
13     if(eleje != nullptr) {
14       eleje->elozo = uj;
15     }
16     eleje = uj;
17     if(vege == nullptr) {
18       vege = uj;
19     }
20     return true;
21   } else {
22     return false;
23   }
24 }
```



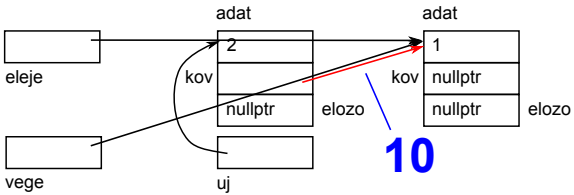
sor2.cpp

```
4  static Lista2* eleje = nullptr;
5  static Lista2* vege  = nullptr;
6
7  bool berak(int adat) {
8    Lista2* uj = new Lista2;
9    if(uj != nullptr) {
10     uj->adat = adat;
11     uj->elozo = nullptr;
12     uj->kov = eleje;
13     if(eleje != nullptr) {
14       eleje->elozo = uj;
15     }
16     eleje = uj;
17     if(vege == nullptr) {
18       vege = uj;
19     }
20     return true;
21   } else {
22     return false;
23   }
24 }
```



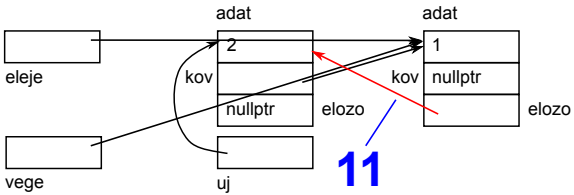
sor2.cpp

```
4  static Lista2* eleje = nullptr;
5  static Lista2* vege  = nullptr;
6
7  bool berak(int adat) {
8    Lista2* uj = new Lista2;
9    if(uj != nullptr) {
10     uj->adat = adat;
11     uj->elozo = nullptr;
12     uj->kov = eleje;
13     if(eleje != nullptr) {
14       eleje->elozo = uj;
15     }
16     eleje = uj;
17     if(vege == nullptr) {
18       vege = uj;
19     }
20     return true;
21   } else {
22     return false;
23   }
24 }
```



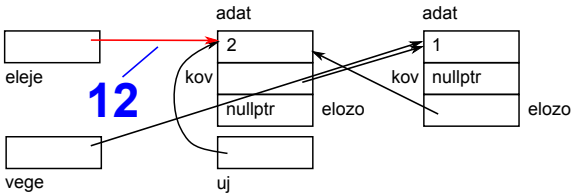
sor2.cpp

```
4  static Lista2* eleje = nullptr;
5  static Lista2* vege  = nullptr;
6
7  bool berak(int adat) {
8    Lista2* uj = new Lista2;
9    if(uj != nullptr) {
10     uj->adat = adat;
11     uj->elozo = nullptr;
12     uj->kov = eleje;
13     if(eleje != nullptr) {
14       eleje->elozo = uj;
15     }
16     eleje = uj;
17     if(vege == nullptr) {
18       vege = uj;
19     }
20     return true;
21   } else {
22     return false;
23   }
24 }
```



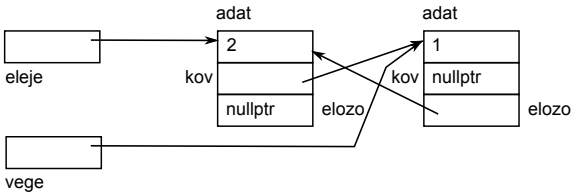
sor2.cpp

```
4  static Lista2* eleje = nullptr;
5  static Lista2* vege  = nullptr;
6
7  bool berak(int adat) {
8    Lista2* uj = new Lista2;
9    if(uj != nullptr) {
10     uj->adat = adat;
11     uj->elozo = nullptr;
12     uj->kov = eleje;
13     if(eleje != nullptr) {
14       eleje->elozo = uj;
15     }
16     eleje = uj;
17     if(vege == nullptr) {
18       vege = uj;
19     }
20     return true;
21   } else {
22     return false;
23   }
24 }
```



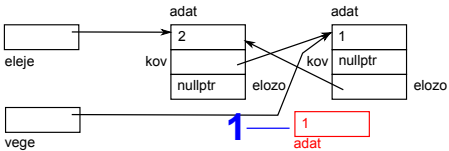
sor2.cpp

```
4  static Lista2* eleje = nullptr;
5  static Lista2* vege  = nullptr;
6
7  bool berak(int adat) {
8    Lista2* uj = new Lista2;
9    if(uj != nullptr) {
10     uj->adat = adat;
11     uj->elozo = nullptr;
12     uj->kov = eleje;
13     if(eleje != nullptr) {
14       eleje->elozo = uj;
15     }
16     eleje = uj;
17     if(vege == nullptr) {
18       vege = uj;
19     }
20     return true;
21   } else {
22     return false;
23   }
24 }
```



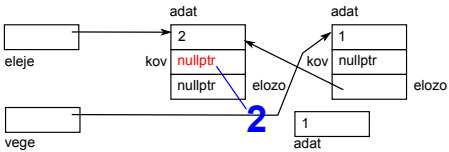
sor2.cpp

```
26  int kivesz() {
27    if(vege == nullptr) {
28      std::cerr << "A sor ures.\n";
29      return 0;
30    } else {
31      int adat = vege->adat;
32      if(vege->elozo != nullptr) {
33        vege->elozo->kov = nullptr;
34      }
35      Lista2* ujVege = vege->elozo;
36      delete vege;
37      vege = ujVege;
38      if(vege == nullptr) eleje = nullptr;
39      return adat;
40    }
41  }
```



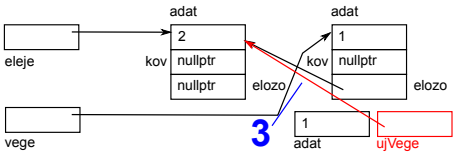
sor2.cpp

```
26  int kivesz() {
27    if(vege == nullptr) {
28      std::cerr << "A sor ures.\n";
29      return 0;
30    } else {
31      int adat = vege->adat;
32      if(vege->elozo != nullptr) {
33        vege->elozo->kov = nullptr;
34      }
35      Lista2* ujVege = vege->elozo;
36      delete vege;
37      vege = ujVege;
38      if(vege == nullptr) eleje = nullptr;
39      return adat;
40    }
41  }
```



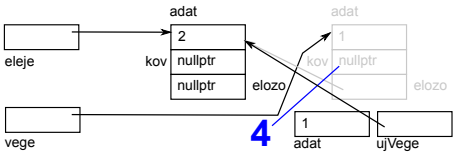
sor2.cpp

```
26  int kivesz() {
27    if(vege == nullptr) {
28      std::cerr << "A sor ures.\n";
29      return 0;
30    } else {
31      int adat = vege->adat;
32      if(vege->elozo != nullptr) {
33        vege->elozo->kov = nullptr;
34      }
35      Lista2* ujVege = vege->elozo;
36      delete vege;
37      vege = ujVege;
38      if(vege == nullptr) eleje = nullptr;
39      return adat;
40    }
41  }
```



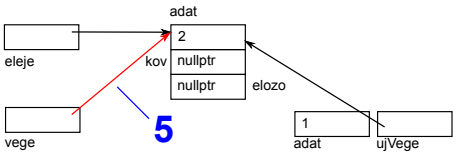
sor2.cpp

```
26  int kivesz() {
27    if(vege == nullptr) {
28      std::cerr << "A sor ures.\n";
29      return 0;
30    } else {
31      int adat = vege->adat;
32      if(vege->elozo != nullptr) {
33        vege->elozo->kov = nullptr;
34      }
35      Lista2* ujVege = vege->elozo;
36      delete vege;
37      vege = ujVege;
38      if(vege == nullptr) eleje = nullptr;
39      return adat;
40    }
41  }
```



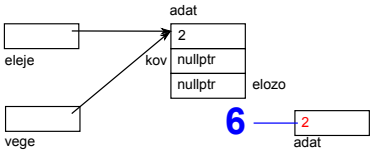
sor2.cpp

```
26  int kivesz() {
27    if(vege == nullptr) {
28      std::cerr << "A sor ures.\n";
29      return 0;
30    } else {
31      int adat = vege->adat;
32      if(vege->elozo != nullptr) {
33        vege->elozo->kov = nullptr;
34      }
35      Lista2* ujVege = vege->elozo;
36      delete vege;
37      vege = ujVege;
38      if(vege == nullptr) eleje = nullptr;
39      return adat;
40    }
41  }
```



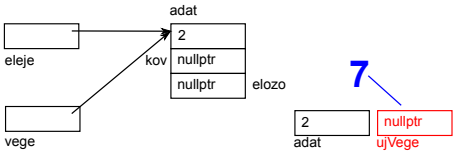
sor2.cpp

```
26  int kivesz() {
27    if(vege == nullptr) {
28      std::cerr << "A sor ures.\n";
29      return 0;
30    } else {
31      int adat = vege->adat;
32      if(vege->elozo != nullptr) {
33        vege->elozo->kov = nullptr;
34      }
35      Lista2* ujVege = vege->elozo;
36      delete vege;
37      vege = ujVege;
38      if(vege == nullptr) eleje = nullptr;
39      return adat;
40    }
41  }
```



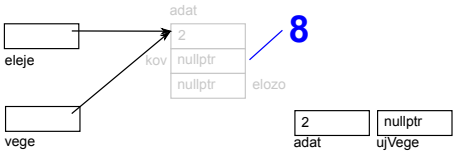
sor2.cpp

```
26  int kivesz() {
27    if(vege == nullptr) {
28      std::cerr << "A sor ures.\n";
29      return 0;
30    } else {
31      int adat = vege->adat;
32      if(vege->elozo != nullptr) {
33        vege->elozo->kov = nullptr;
34      }
35      Lista2* ujVege = vege->elozo;
36      delete vege;
37      vege = ujVege;
38      if(vege == nullptr) eleje = nullptr;
39      return adat;
40    }
41  }
```



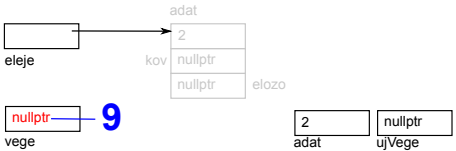
sor2.cpp

```
26  int kivesz() {
27    if(vege == nullptr) {
28      std::cerr << "A sor ures.\n";
29      return 0;
30    } else {
31      int adat = vege->adat;
32      if(vege->elozo != nullptr) {
33        vege->elozo->kov = nullptr;
34      }
35      Lista2* ujVege = vege->elozo;
36      delete vege;
37      vege = ujVege;
38      if(vege == nullptr) eleje = nullptr;
39      return adat;
40    }
41  }
```



sor2.cpp

```
26  int kivesz() {
27    if(vege == nullptr) {
28      std::cerr << "A sor ures.\n";
29      return 0;
30    } else {
31      int adat = vege->adat;
32      if(vege->elozo != nullptr) {
33        vege->elozo->kov = nullptr;
34      }
35      Lista2* ujVege = vege->elozo;
36      delete vege;
37      vege = ujVege;
38      if(vege == nullptr) eleje = nullptr;
39      return adat;
40    }
41  }
```



sor2.cpp

```
26  int kivesz() {
27    if(vege == nullptr) {
28      std::cerr << "A sor ures.\n";
29      return 0;
30    } else {
31      int adat = vege->adat;
32      if(vege->elozo != nullptr) {
33        vege->elozo->kov = nullptr;
34      }
35      Lista2* ujVege = vege->elozo;
36      delete vege;
37      vege = ujVege;
38      if(vege == nullptr) eleje = nullptr;
39      return adat;
40    }
41  }
```

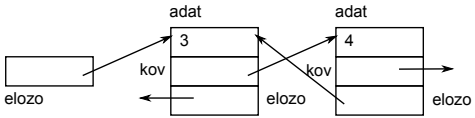
Készítsünk ismét általános célú függvényeket!

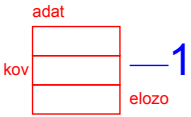
listaTeszt2.cpp (Lista2.cpp, Lista2.h)

```
5  int main() {
6    Lista2 *horgony=nullptr, *seged=nullptr, *kozepe;
7    for(int i=0; i<7; i++) {
8      seged = beszur2(i, seged);
9      if(horgony == nullptr) {
10       horgony = seged;
11     }
12     if(i == 3) {
13       kozepe = seged;
14     }
15   }
16   kiir2(horgony);
17   kozepe = beszur2(666, kozepe);
```

Lista2.cpp

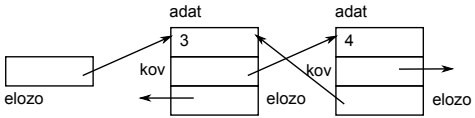
```
4  // 'elozo' utan beszur egy uj elemet
5  Lista2 *beszur2(int adat,
6                  Lista2 *elozo) {
7    Lista2 *uj = new Lista2;
8    if(uj) {
9      uj->adat = adat;
10     if(elozo) {
11       uj->elozo = elozo;
12       uj->kov = elozo->kov;
13       elozo->kov = uj;
14       if(uj->kov) uj->kov->elozo = uj;
15     } else {
16       uj->elozo = uj->kov = nullptr;
17     }
18   }
19   return uj;
20 }
```

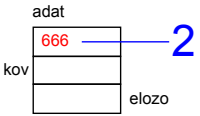




Lista2.cpp

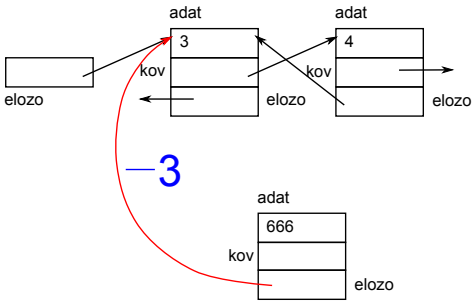
```
4  // 'elozo' utan beszur egy uj elemet
5  Lista2 *beszur2(int adat,
6                  Lista2 *elozo) {
7    Lista2 *uj = new Lista2;
8    if(uj) {
9      uj->adat = adat;
10     if(elozo) {
11       uj->elozo = elozo;
12       uj->kov = elozo->kov;
13       elozo->kov = uj;
14       if(uj->kov) uj->kov->elozo = uj;
15     } else {
16       uj->elozo = uj->kov = nullptr;
17     }
18   }
19   return uj;
20 }
```





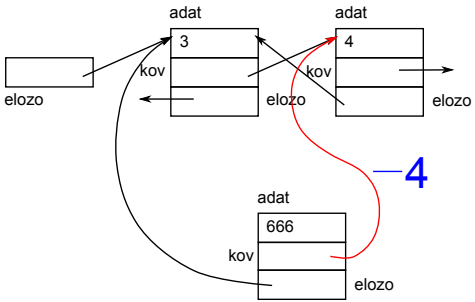
Lista2.cpp

```
4  // 'elozo' utan beszur egy uj elemet
5  Lista2 *beszur2(int adat,
6                  Lista2 *elozo) {
7    Lista2 *uj = new Lista2;
8    if(uj) {
9      uj->adat = adat;
10     if(elozo) {
11       uj->elozo = elozo;
12       uj->kov = elozo->kov;
13       elozo->kov = uj;
14       if(uj->kov) uj->kov->elozo = uj;
15     } else {
16       uj->elozo = uj->kov = nullptr;
17     }
18   }
19   return uj;
20 }
```



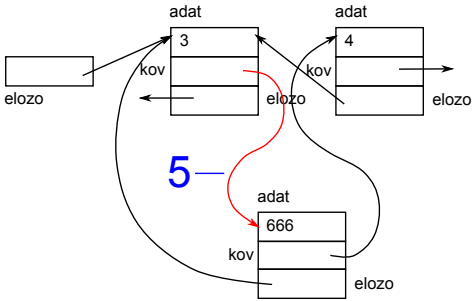
Lista2.cpp

```
4  // 'elozo' utan beszur egy uj elemet
5  Lista2 *beszur2(int adat,
6                  Lista2 *elozo) {
7    Lista2 *uj = new Lista2;
8    if(uj) {
9      uj->adat = adat;
10     if(elozo) {
11       uj->elozo = elozo;
12       uj->kov = elozo->kov;
13       elozo->kov = uj;
14       if(uj->kov) uj->kov->elozo = uj;
15     } else {
16       uj->elozo = uj->kov = nullptr;
17     }
18   }
19   return uj;
20 }
```



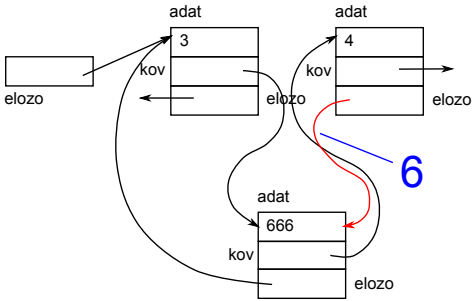
Lista2.cpp

```
4  // 'elozo' utan beszur egy uj elemet
5  Lista2 *beszur2(int adat,
6                  Lista2 *elozo) {
7    Lista2 *uj = new Lista2;
8    if(uj) {
9      uj->adat = adat;
10     if(elozo) {
11       uj->elozo = elozo;
12       uj->kov = elozo->kov;
13       elozo->kov = uj;
14       if(uj->kov) uj->kov->elozo = uj;
15     } else {
16       uj->elozo = uj->kov = nullptr;
17     }
18   }
19   return uj;
20 }
```



Lista2.cpp

```
4  // 'elozo' utan beszur egy uj elemet
5  Lista2 *beszur2(int adat,
6                  Lista2 *elozo) {
7    Lista2 *uj = new Lista2;
8    if(uj) {
9      uj->adat = adat;
10     if(elozo) {
11       uj->elozo = elozo;
12       uj->kov = elozo->kov;
13       elozo->kov = uj;
14       if(uj->kov) uj->kov->elozo = uj;
15     } else {
16       uj->elozo = uj->kov = nullptr;
17     }
18   }
19   return uj;
20 }
```

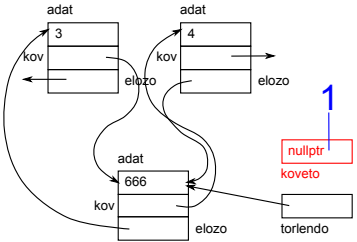


listaTeszt2.cpp (Lista2.cpp, Lista2.h)

```
18    kiir2(horgony);
19    torol2(kozepe);
20    horgony = torol2(horgony);
21    kiir2(horgony);
22    torolMindet2(horgony);
23    return 0;
24 }
```

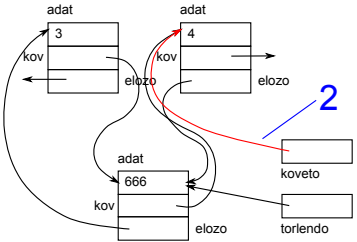
Kimenet

```
0       1       2       3       4       5       6
0       1       2       3       666     4       5       6
1       2       3       4       5       6
```



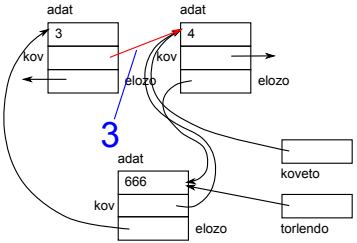
Lista2.cpp

```
30 // torli 'torlendo'-t, vissza: kov. elem
31 Lista2 *torol2(Lista2 *torlendo) {
32   Lista2 *koveto = nullptr;
33   if(torlendo) {
34     koveto = torlendo->kov;
35     if(torlendo->elozo) torlendo->elozo->kov = torlendo->kov;
36     if(torlendo->kov) torlendo->kov->elozo = torlendo->elozo;
37     delete torlendo;
38   }
39   return koveto;
40 }
```



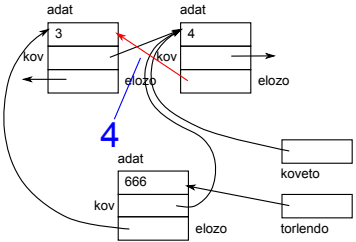
Lista2.cpp

```
30 // torli 'torlendo'-t, vissza: kov. elem
31 Lista2 *torol2(Lista2 *torlendo) {
32   Lista2 *koveto = nullptr;
33   if(torlendo) {
34     koveto = torlendo->kov;
35     if(torlendo->elozo) torlendo->elozo->kov = torlendo->kov;
36     if(torlendo->kov) torlendo->kov->elozo = torlendo->elozo;
37     delete torlendo;
38   }
39   return koveto;
40 }
```



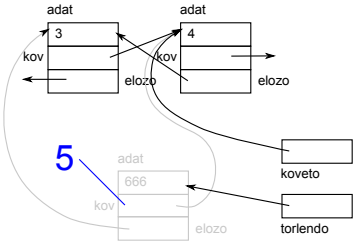
Lista2.cpp

```
30 // torli 'torlendo'-t, vissza: kov. elem
31 Lista2 *torol2(Lista2 *torlendo) {
32   Lista2 *koveto = nullptr;
33   if(torlendo) {
34     koveto = torlendo->kov;
35     if(torlendo->elozo) torlendo->elozo->kov = torlendo->kov;
36     if(torlendo->kov) torlendo->kov->elozo = torlendo->elozo;
37     delete torlendo;
38   }
39   return koveto;
40 }
```



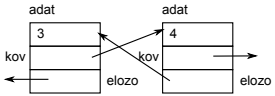
Lista2.cpp

```
30 // torli 'torlendo'-t, vissza: kov. elem
31 Lista2 *torol2(Lista2 *torlendo) {
32   Lista2 *koveto = nullptr;
33   if(torlendo) {
34     koveto = torlendo->kov;
35     if(torlendo->elozo) torlendo->elozo->kov = torlendo->kov;
36     if(torlendo->kov) torlendo->kov->elozo = torlendo->elozo;
37     delete torlendo;
38   }
39   return koveto;
40 }
```



Lista2.cpp

```
30 // torli 'torlendo'-t, vissza: kov. elem
31 Lista2 *torol2(Lista2 *torlendo) {
32   Lista2 *koveto = nullptr;
33   if(torlendo) {
34     koveto = torlendo->kov;
35     if(torlendo->elozo) torlendo->elozo->kov = torlendo->kov;
36     if(torlendo->kov) torlendo->kov->elozo = torlendo->elozo;
37     delete torlendo;
38   }
39   return koveto;
40 }
```



Lista2.cpp

```
30 // torli 'torlendo'-t, vissza: kov. elem
31 Lista2 *torol2(Lista2 *torlendo) {
32   Lista2 *koveto = nullptr;
33   if(torlendo) {
34     koveto = torlendo->kov;
35     if(torlendo->elozo) torlendo->elozo->kov = torlendo->kov;
36     if(torlendo->kov) torlendo->kov->elozo = torlendo->elozo;
37     delete torlendo;
38   }
39   return koveto;
40 }
```

Lista2.cpp

```
42 // torli a teljes listat
43 void torolMindet2(Lista2 *horgony) {
44   Lista2 *seged = horgony;
45   while(seged) {
46     seged = torol2(seged);
47   }
48 }

22 // kiirja a lista osszes elemet
23 void kiir2(Lista2 *horgony) {
24   for(Lista2* seged=horgony; seged; seged=seged->kov) {
25     std::cout << seged->adat << '\t';
26   }
27   std::cout << std::endl;
28 }
```