Programozás
(GKxB_INTM021)

Dr. Hatwagner F. Miklós

Széchenyi István Egyetem, Győr

2021. április 18.

Szavak tárolása, kiírása ABC-sorrendben

Feladat:

- Olvasson be előre nem ismert számú szót végjelig,
- tárolja el ezeket, majd
- jelenítse meg ABC-sorrendben!

Ötletek adatszerkezetre:

Dinamikus tömb

A szavakat legalább az adatbevitel végén rendezni kell majd, vagy eleve a megfelelő helyre beszúrni $\rightarrow$ sok adatmozgatás

Láncolt lista

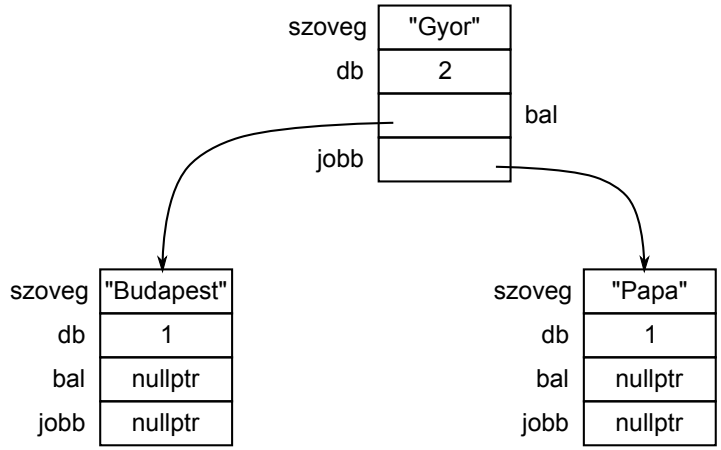
Az adatokat csak sorosan lehet elérni, emiatt a megfelelő helyre beszúrás/rendezés nagyon lassú

Olyan dinamikus adatszerkezet kellene, ahol gyors az adatok tárolása, rendezése, visszaolvasása!

Szavak tárolása, kiírása ABC-sorrendben

Javaslat:

Bináris fa



Kimenet

```
Szavak beolvasasa, tarolasa, abc-sorrendben kiirasa
Adja meg a szavakat - vegjelig!
Gyor
Budapest
Gyor
Papa
-
A megadott szavak abc sorrendben a kovetkezok:
Budapest
Gyor
Gyor
Papa
```

Szavak tárolása, kiírása ABC-sorrendben

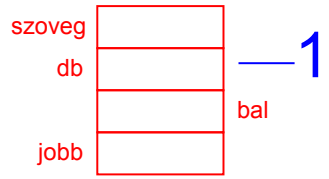
binfa1.cpp

```
46  int main() {
47    csucs* gyoker = nullptr;
48    string szo;
49    cout << "Szavak beolvasasa, tarolasa, "
50         << "abc-sorrendben kiirasa\n"
51         << "Adja meg a szavakat - vegjelig!\n";
52    while(cin>>szo, szo!="-") {
53      gyoker = beszurBF(gyoker, szo);
54    }
55    cout << "A megadott szavak abc sorrendben "
56         << "a kovetkezok:\n";
57    kiirBF(gyoker);
58    felszabaditBF(gyoker);
59    return 0;
60  }
```

Szavak tárolása, kiírása ABC-sorrendben

binfa1.cpp

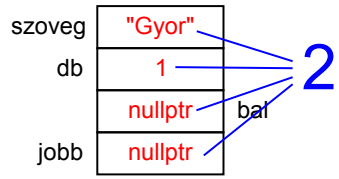
```
11  csucs* beszurBF(csucs* cs, string s) {
12    if(cs == nullptr) {
13      cs = new csucs;
14      cs->szoveg = s;
15      cs->db = 1;
16      cs->bal = cs->jobb = nullptr;
17    } else if(s == cs->szoveg) {
18      cs->db++;
19    } else if(s < cs->szoveg) {
20      cs->bal = beszurBF(cs->bal, s);
21    } else {
22      cs->jobb = beszurBF(cs->jobb, s);
23    }
24    return cs;
25  }
```



Szavak tárolása, kiírása ABC-sorrendben

binfa1.cpp

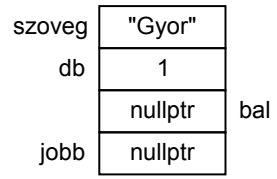
```
11  csucs* beszurBF(csucs* cs, string s) {
12    if(cs == nullptr) {
13      cs = new csucs;
14      cs->szoveg = s;
15      cs->db = 1;
16      cs->bal = cs->jobb = nullptr;
17    } else if(s == cs->szoveg) {
18      cs->db++;
19    } else if(s < cs->szoveg) {
20      cs->bal = beszurBF(cs->bal, s);
21    } else {
22      cs->jobb = beszurBF(cs->jobb, s);
23    }
24    return cs;
25  }
```

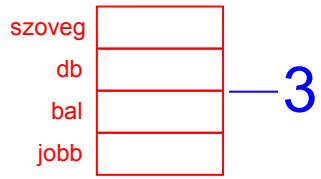


Szavak tárolása, kiírása ABC-sorrendben

binfa1.cpp

```
11  csucs* beszurBF(csucs* cs, string s) {
12    if(cs == nullptr) {
13      cs = new csucs;
14      cs->szoveg = s;
15      cs->db = 1;
16      cs->bal = cs->jobb = nullptr;
17    } else if(s == cs->szoveg) {
18      cs->db++;
19    } else if(s < cs->szoveg) {
20      cs->bal = beszurBF(cs->bal, s);
21    } else {
22      cs->jobb = beszurBF(cs->jobb, s);
23    }
24    return cs;
25  }
```

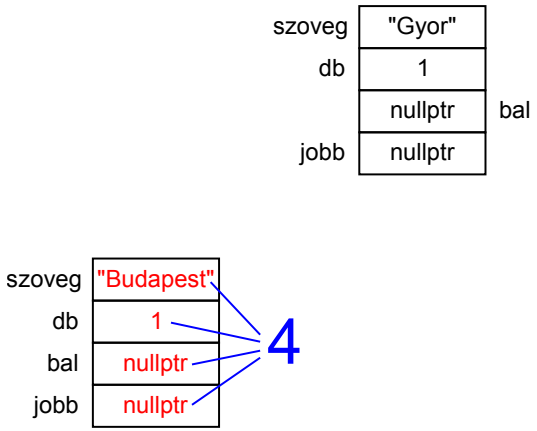




Szavak tárolása, kiírása ABC-sorrendben

binfa1.cpp

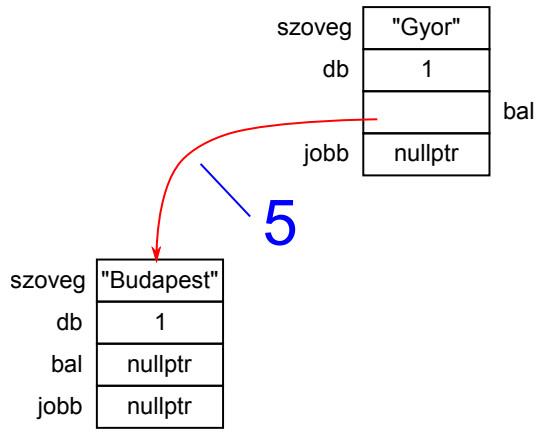
```
11  csucs* beszurBF(csucs* cs, string s) {
12    if(cs == nullptr) {
13      cs = new csucs;
14      cs->szoveg = s;
15      cs->db = 1;
16      cs->bal = cs->jobb = nullptr;
17    } else if(s == cs->szoveg) {
18      cs->db++;
19    } else if(s < cs->szoveg) {
20      cs->bal = beszurBF(cs->bal, s);
21    } else {
22      cs->jobb = beszurBF(cs->jobb, s);
23    }
24    return cs;
25  }
```



Szavak tárolása, kiírása ABC-sorrendben

binfa1.cpp

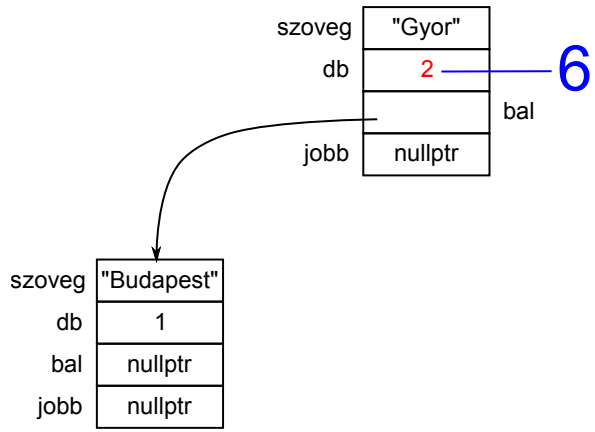
```
11  csucs* beszurBF(csucs* cs, string s) {
12    if(cs == nullptr) {
13      cs = new csucs;
14      cs->szoveg = s;
15      cs->db = 1;
16      cs->bal = cs->jobb = nullptr;
17    } else if(s == cs->szoveg) {
18      cs->db++;
19    } else if(s < cs->szoveg) {
20      cs->bal = beszurBF(cs->bal, s);
21    } else {
22      cs->jobb = beszurBF(cs->jobb, s);
23    }
24    return cs;
25  }
```



Szavak tárolása, kiírása ABC-sorrendben

binfa1.cpp

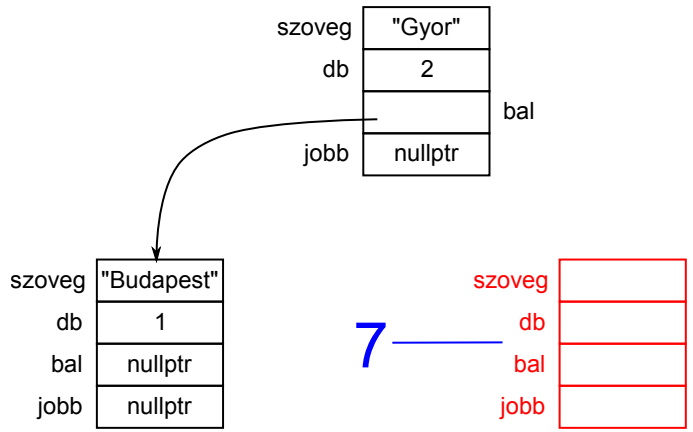
```
11  csucs* beszurBF(csucs* cs, string s) {
12    if(cs == nullptr) {
13      cs = new csucs;
14      cs->szoveg = s;
15      cs->db = 1;
16      cs->bal = cs->jobb = nullptr;
17    } else if(s == cs->szoveg) {
18      cs->db++;
19    } else if(s < cs->szoveg) {
20      cs->bal = beszurBF(cs->bal, s);
21    } else {
22      cs->jobb = beszurBF(cs->jobb, s);
23    }
24    return cs;
25  }
```



Szavak tárolása, kiírása ABC-sorrendben

binfa1.cpp

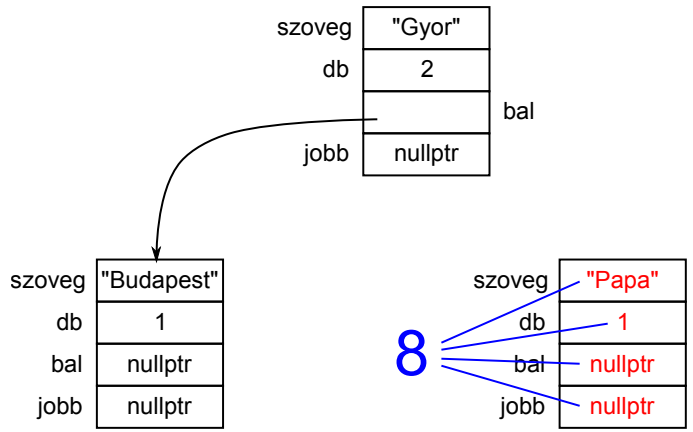
```
11  csucs* beszurBF(csucs* cs, string s) {
12    if(cs == nullptr) {
13      cs = new csucs;
14      cs->szoveg = s;
15      cs->db = 1;
16      cs->bal = cs->jobb = nullptr;
17    } else if(s == cs->szoveg) {
18      cs->db++;
19    } else if(s < cs->szoveg) {
20      cs->bal = beszurBF(cs->bal, s);
21    } else {
22      cs->jobb = beszurBF(cs->jobb, s);
23    }
24    return cs;
25  }
```



Szavak tárolása, kiírása ABC-sorrendben

binfa1.cpp

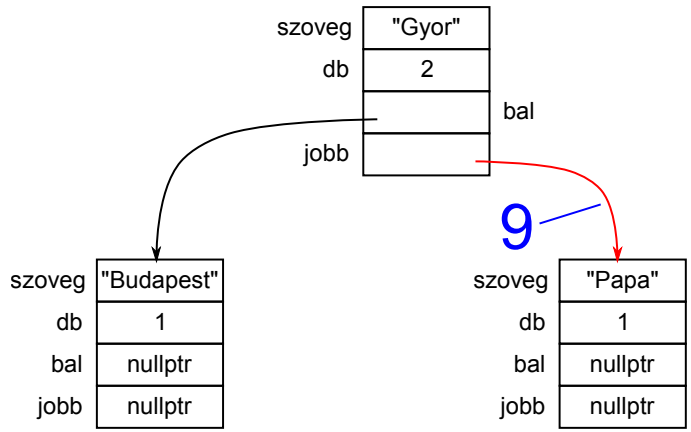
```
11  csucs* beszurBF(csucs* cs, string s) {
12    if(cs == nullptr) {
13      cs = new csucs;
14      cs->szoveg = s;
15      cs->db = 1;
16      cs->bal = cs->jobb = nullptr;
17    } else if(s == cs->szoveg) {
18      cs->db++;
19    } else if(s < cs->szoveg) {
20      cs->bal = beszurBF(cs->bal, s);
21    } else {
22      cs->jobb = beszurBF(cs->jobb, s);
23    }
24    return cs;
25  }
```



Szavak tárolása, kiírása ABC-sorrendben

binfa1.cpp

```
11  csucs* beszurBF(csucs* cs, string s) {
12    if(cs == nullptr) {
13      cs = new csucs;
14      cs->szoveg = s;
15      cs->db = 1;
16      cs->bal = cs->jobb = nullptr;
17    } else if(s == cs->szoveg) {
18      cs->db++;
19    } else if(s < cs->szoveg) {
20      cs->bal = beszurBF(cs->bal, s);
21    } else {
22      cs->jobb = beszurBF(cs->jobb, s);
23    }
24    return cs;
25  }
```



Szavak tárolása, kiírása ABC-sorrendben

binfa1.cpp – Kiírás, felszabadítás

```
27  void kiirBF(csucs* cs) { // inorder bejaras
28    long i;
29    if(cs != nullptr) {
30      kiirBF(cs->bal);
31      for(i=0; i<cs->db; i++) {
32        cout << cs->szoveg << endl;
33      }
34      kiirBF(cs->jobb);
35    }
36  }
37
38  void felszabaditBF(csucs* cs) { // postorder bejaras
39    if(cs != nullptr) {
40      felszabaditBF(cs->bal);
41      felszabaditBF(cs->jobb);
42      delete cs;
43    }
44  }
```

Kiírás fordított ABC-sorrendben

binfa2.cpp

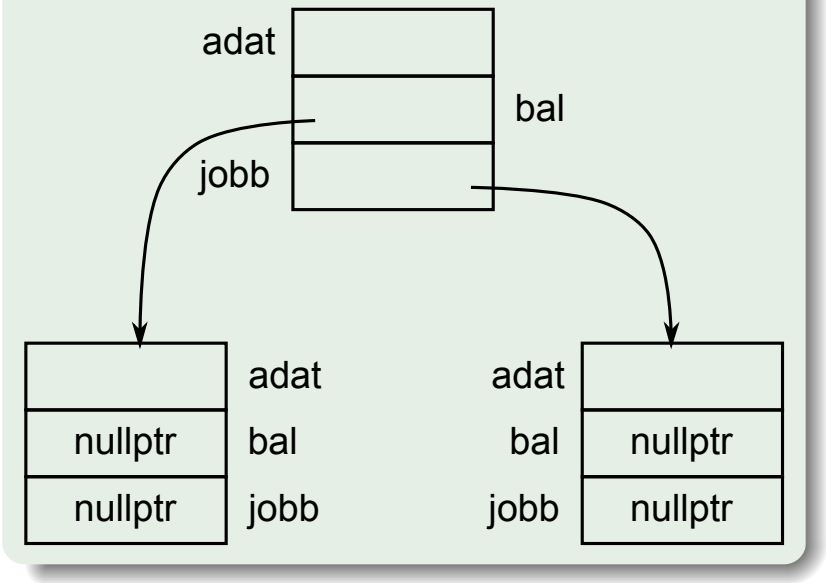
```
27  void kiirBF(csucs* cs) { // forditott abc sorrend
28    if(cs == nullptr) return;
29    kiirBF(cs->jobb);
30    for(long i=0; i<cs->db; i++)
31      cout << cs->szoveg << endl;
32    kiirBF(cs->bal);
33  }
```

Kimenet

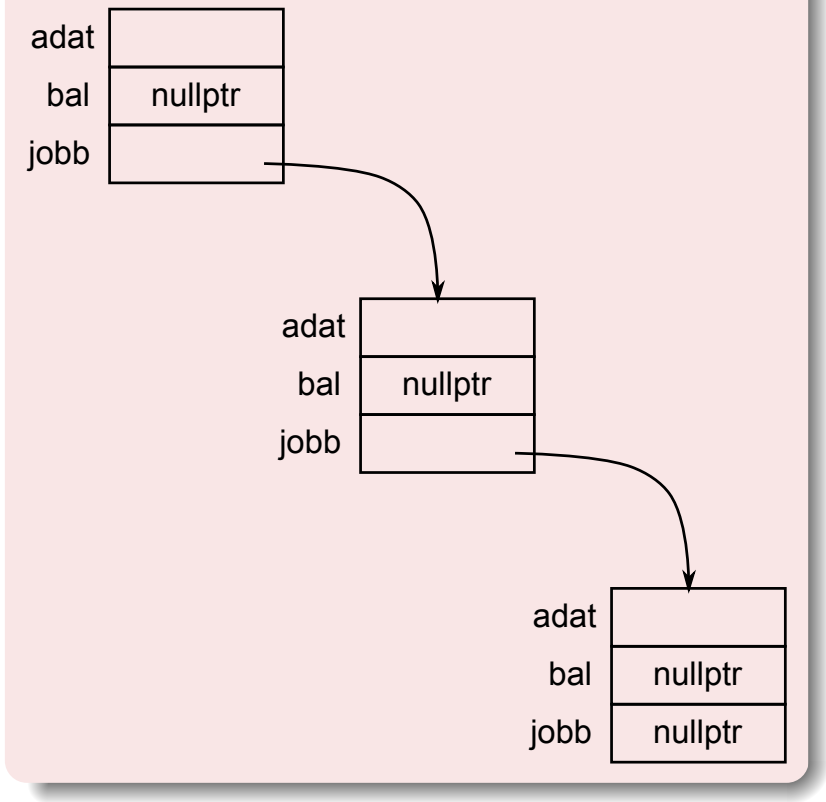
```
Szavak beolvasasa, tarolasa, forditott abc-sorrendben kiirasa
Adja meg a szavakat - vegjelig!
Gyor
Budapest
Gyor
Papa
-
A megadott szavak forditott abc sorrendben a kovetkezok:
Papa
Gyor
Gyor
Budapest
```

Bináris fák és láncolt listák kapcsolata

Legjobb eset



Legrosszabb eset



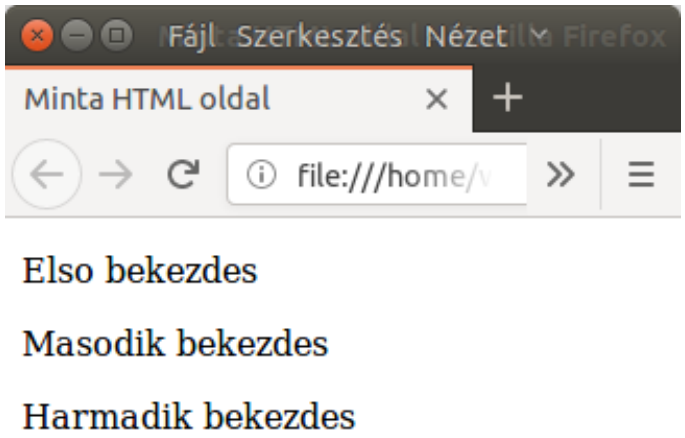
További kérdések:

- Hogyan törölhető egy csúcs?
- Hogyan biztosítható a fa kiegyensúlyozottsága?

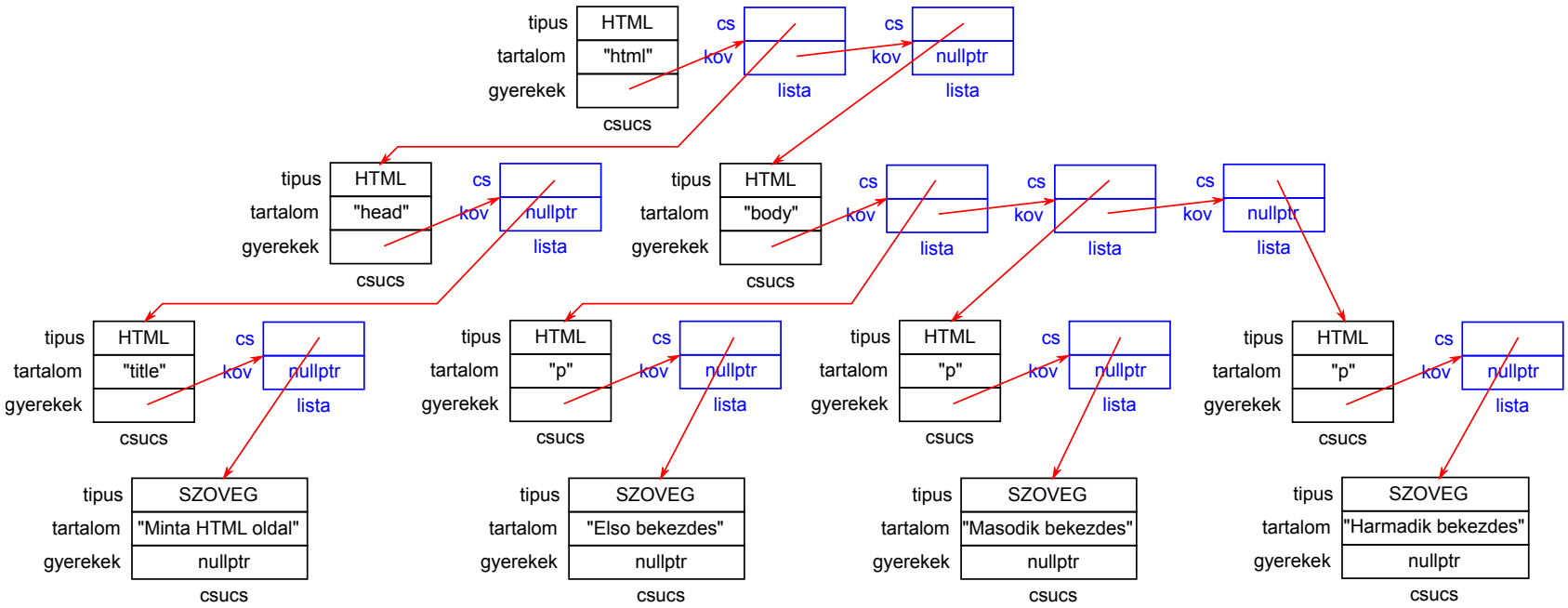
Néha általános fa struktúrára is szükség van, kettőnél több gyerek csúccsal, pl. HTML
HTML (HyperText Markup Language) → DOM (Document Object Model)

minta.html

```
<!DOCTYPE html>
<html>
  <head>
    <title>Minta HTML oldal</title>
  </head>
  <body>
    <p>Elso bekezdes</p>
    <p>Masodik bekezdes</p>
    <p>Harmadik bekezdes</p>
  </body>
</html>
```



Fák

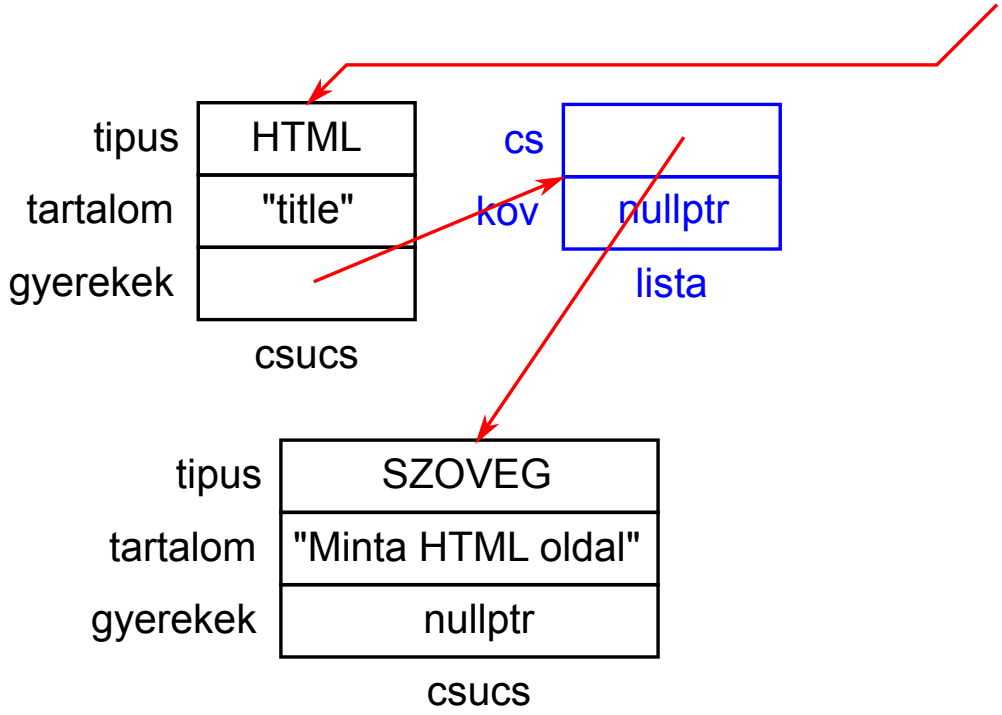


A `minta.html` „DOM”-ja

Fák

html.cpp

```
5  struct lista;
6
7  #define HTML 0
8  #define SZOVEG 1
9  struct csucs {
10   int tipus;
11   string tartalom;
12   lista* gyerekek;
13 };
14
15 struct lista {
16   csucs* cs;
17   lista* kov;
18 };
```



html.cpp

```
20  csucs* uj(int tipus, string tartalom) {
21    csucs* cs = new csucs;
22    cs->tipus = tipus;
23    cs->tartalom = tartalom;
24    cs->gyerekek = nullptr;
25    return cs;
26  }
```

html.cpp

```
28  void ujGyerek(csucs* szulo, csucs* gyerek) {
29    lista* l = new lista;
30    l->cs = gyerek;
31    l->kov = nullptr;
32    if(szulo->gyerekek == nullptr) {
33      szulo->gyerekek = l;
34    } else {
35      lista* utolso = szulo->gyerekek;
36      while(utolso->kov != nullptr) {
37        utolso = utolso->kov;
38      }
39      utolso->kov = l;
40    }
41  }
```

html.cpp

```
43  void kiir(csucs* cs, int behuz) {
44    for(int i=0; i<behuz; i++) {
45      cout << "  ";
46    }
47    if(cs->tipus == HTML) {
48      cout << '<' << cs->tartalom << ">\n";
49      lista* kov = cs->gyerekek;
50      while(kov != nullptr) {
51        kiir(kov->cs, behuz+1);
52        kov = kov->kov;
53      }
54      for(int i=0; i<behuz; i++) {
55        cout << "  ";
56      }
57      cout << "</" << cs->tartalom << ">\n";
58    } else {
59      cout << cs->tartalom << endl;
60    }
61  }
```

Kimenet

```
<html>
  <head>
    <title>
      Minta HTML oldal
    </title>
  </head>
  <body>
    <p>
      Elso bekezdes
    </p>
    <p>
      Masodik bekezdes
    </p>
    <p>
      Harmadik bekezdes
    </p>
  </body>
</html>
```

html.cpp

```
63 void torol(csucs* cs) {
64   lista* kov = cs->gyerekek;
65   while(kov != nullptr) {
66     torol(kov->cs);
67     lista* l = kov;
68     kov = kov->kov;
69     delete l;
70   }
71   delete cs;
72 }
```

html.cpp

```
74  int main() {
75    csucs* html = uj(HTML, "html");
76    csucs* head = uj(HTML, "head");
77    csucs* body = uj(HTML, "body");
78    ujGyerek(html, head);
79    ujGyerek(html, body);
80
81    csucs* title = uj(HTML, "title");
82    ujGyerek(title, uj(SZOVEG, "Minta HTML oldal"));
83    ujGyerek(head, title);
```

html.cpp

```
85      csucs* p1 = uj(HTML, "p");
86      ujGyerek(p1, uj(SZOVEG, "Elso bekezdes"));
87      ujGyerek(body, p1);
88      csucs* p2 = uj(HTML, "p");
89      ujGyerek(p2, uj(SZOVEG, "Masodik bekezdes"));
90      ujGyerek(body, p2);
91      csucs* p3 = uj(HTML, "p");
92      ujGyerek(p3, uj(SZOVEG, "Harmadik bekezdes"));
93      ujGyerek(body, p3);
94
95      kiir(html, 0);
96      torol(html);
97      return 0;
98  }
```

enum – Felsorolt típus

Időnként több konstans definiálására van szükség, pl.

html.cpp

```
7  #define HTML 0
8  #define SZOVEG 1
9  struct csucs {
10   int tipus;
11   string tartalom;
12   lista* gyerekek;
13 };
```

Hét napjai, pl.

```
#define HETFO 0
#define KEDD 1
#define SZERDA 2
#define CSUTORTOK 3
#define PENTEK 4
#define SZOMBAT 5
#define VASARNAP 6
```

Egyszerű megoldás sok, különböző értékű konstans megadására:

`enum`

`enum` – Felsorolt típus

egész értékek sorozatának mnemonikussá tétele
enum <enum-címke><{enumerátorlista}><azonosítólista>;
enumerátorlista tulajdonságai:

- elemei: enumerátorok, enum konstansok
- normál változók névterületét használják
- belső ábrázolás típusa `int`
- értéküket elhagyható konstans kifejezések definiálják
- értékek 0-tól vagy az utoljára definiált értéktől implicit módon egyesével nőnek

enum – Felsorolt típus

enum.cpp

```
4  int main() {
5    enum nemet {opel, vw=10, audi} deutscheKraft;
6    enum nemet szomszedKocsija;
7    nemet haverKocsija;
8    // Kesobb nem lehet tobb ilyen tipusu
9    // valtozot letrehozni
10   enum {toyota, honda} rizsRaketa;
11   enum {renault, peugeot};
12   // int vw; error: 'int vw' redeclared as
13   //                different kind of symbol
14   { // OK: ebben a hatokorben nincs ilyen
15     // azonosito
16     int vw; }
17   int nemet; // OK
18   deutscheKraft = opel;
```

enum – Felsorolt típus

enum.cpp

```
19    // error: cannot convert 'main()::<anonymous enum>'
20    //          to 'main()::nemet' in assignment
21    // deutscheKraft = renault;
22    // error: invalid conversion from 'int'
23    //          to 'main()::nemet'
24    // szomszedKocsija = 2;
25    // error: expected ';' before numeric constant
26    // szomszedKocsija = (nemet)2;
27    szomszedKocsija = (enum nemet)2;
28    int kocsi = honda;
29    kocsi = peugeot;
30    int it[3];
31    it[toyota] = vw;
32    // error: no 'operator++(int)' declared for
33    //          postfix '++'
34    // toyota++;
35    return 0;
36  }
```

enum – Felsorolt típus

datum3.h

```
10  enum hetnapja { HETFO, KEDD, SZERDA, CSUTORTOK,
11                  PENTEK, SZOMBAT, VASARNAP };

22  hetnapja hetNapjaEnum(const datum* d);
```

datum3.cpp

```
44  hetnapja hetNapjaEnum(const datum* d) { // het napjanak
45    int hn;                                // szamitasa
46    char seged[12] = { 6, 2, 2, 5, 0, 3, 5, 1, 4, 6, 2, 4 };
47    hn = d->ev*1.25 + d->nap;
48    hn += seged[d->ho-1];
49    if ((d->ev%4==0) and d->ho<3) hn--;
50    while (hn > 7) hn -= 7;
51    return hetnapja(hn==1 ? 6 : hn-2);
52  }
```

enum – Felsorolt típus

datum3teszt.cpp

```
5  int main() {
6    int ev, ho, nap;
7    cout << "Ev: "; cin >> ev;
8    cout << "Ho: "; cin >> ho;
9    cout << "Nap:"; cin >> nap;
10   datum d = { ev, ho, nap };
11   if(hetNapjaEnum(&d) == SZOMBAT) {
12     cout << "Night fever, night fever\n";
13   } else {
14     cout << "Another day in paradise\n";
15   }
16   return 0;
17 }
```