MEMÓRIAMODELL
(STATIKUSAN LÁNCOLT TÖMB)

TARTALOM

0 A FELADAT

Valósítsuk meg a dinamikus memóriakezelést egy adott típushoz (TElem), tömbben!

0.1 Elvárások

- egy unit-ba foglaljuk a „kezelőszerveket”
- az exportált műveletek
  - helyet foglal egy adott típusú adat számára, ha nem sikerült, akkor azt a Sehova értékkel jelzi
  - felszabadítja az adott (korábban lefoglalással szerzett) helyét az elemnek, majd a címváltozót Sehova értékűre állítja
  - a „teljes” memória pillanatnyi állapotát –ellenőrzési céllal– kilistázza a képernyőre
- a lefoglalt helyet minél kényelmesebben tegye lehetővé Pascal-ban használni (azaz az adott „címen” levő elem módosítását és értékének elérését)

1 A MEGOLDÁS

1.1 Ötletek

- a *tömb*, amelyet a unit használ, legyen *exportálva*, és álljon `TElem`-ekből; így egy lefoglalt eleme szerepelhet akár értékadás bal oldalán, akár kifejezés részeként – a Pascal korlátozásainak megfelelően
- a memóriát reprezentáló tömb a *unit számára* –a „munka közben”– listába fűzött elemek tömbjeként látszik, azaz *statikusan* (*tömbben) láncolt ábrázolású*
- az eleddig még ki nem adott elemeket fűzzük listába, azaz *szabadlista*ként kezeljük
- ugyanazon tömb kétféle (exportált: `TElem`-ek tömbje; és a belső: tömbalapú láncolt listás) „értelmezésének” alapja a Turbo Pascal `absolute` fogalma: az `interface`-ben mint `TElem` bázistípusú tömb található, az `implemetation`-ben –előzővel pontosan egyező elemméretben és indexszámban– olyan tömbös szabadlista, amelyben a szabadként nyilvántartott elemek „végét” mint listamutatót használja föl; az elem „vége” természetesen szabadon felhasználható, amint memóriakezelő a Lefoglal művelettel felhasználásra átadta a hívó programnak.
- a memória elemtípusa a `TElem` vagy a unit-on belül definiálandó, vagy unit-importtal (`Uses`) építendő bele a unit-ba

A lényeg pontosabb leírását lásd a program elején levő kommentben!

1.2 A unit

A kód (MEMUNIT.TPU) forrása (MEMUNIT.PAS):

```
1   Unit MemUnit; {Type TElem; Const MaxMem:Integer}
2   (*
3     A memóriamodell megvalósítása.
4     Lényeg:
5     * a memória egy adott típusú elemből álló tömb,
6     * a cím ennek a tömbnek egy indexe,
7     * a használó a tömböt, azaz a dinamikusan kezelt memóriát,
8       TElem bázistípusúként látja,
9     * a használó ehhez, mint hagyományos Pascal tömbhöz hozzáfér
10      értékadásban (az értéket kapóként), és
11      --a Pascal kötöttségeit betartva-- kifejezés részeként értékviselőként
12      (vissza is tud élni vele ugyanúgy, mintha tényleges címekkel dolgozna),
13    * a tömböt a unit-on belül --tehát a használó szeme elöl elrejtve--
14      megfelelö (TElem-) méretü, de láncolást is tartalmazó elemü tömbként
15      kezeljük,
16    * a láncolás célja, hogy a még fel nem használt elemek egy ún. szabadlistát
17      alkotva szolgálja a minél egyszerübb "memória" gazdálkodást (Lefoglal,
18      Felszabadít).
19    Exportált fogalmak:
20    * TCím         = a címek típusa
21    * Sehova       = speciális cím, nem használható címzésre,
22                   = memóriahivatkozásként használva futási hibát okoz
23                   = (a 'Range Checking' On állapota esetén)
24    * mem          = a memóriát jelentö tömb, így a megkapott cim címü
25                   = memóriatartományt lehet módosítani:
26                   = "mem[cim]:=..."
27                   = memóriatartomány értékét fel lehet használni:
28                   = "...mem[cim]..."
29    * Lefoglal     = eljárás a szabadlistából egy elemnyi memóriatartomány
30                   = címét (tömbindexét) adja vissza,
31                   = ha nincs már szabad hely, akkor Sehova-t
32    * Felszabadít  = visszacsatolja a szabadlistába az adott memóriatartományt
33                   = a paraméter Sehova értékü lesz
34    * MemDump      = kiírja a memóriát mint tömböt a nyomkövetés kedvéért,
35                   = pontosabban a tömbelemek láncolásra használt értékét,
36                   = hogy lehessen ellenőrizni a szabadlista állapotát;
37                   = e mellett megadja az éppen foglalt/szabad elemek számát.
38  *)
39    Interface
40      Type  TElem=String[9];
41      Const MaxMem=10;
42            Sehova=0;
43      Type  TCim=0..MaxMem;
44            TMem=Array [1..MaxMem] of TElem;
45      Var   Mem:TMem;
46      Procedure Lefoglal(Var i:TCim);
47      Procedure Felszabadit(Var i:TCim);
48      Procedure MemDump;
49
50    Implementation
51
52      Uses  Newdelay,Crt;
```

```
53      Type  TMemElem=Record
54                      mas:String[Sizeof(TElem)-Sizeof(TCim)-1];
55                      kov:TCim
56                    End;
57            TbMem=Array [1..MaxMem] of TMemElem;
58      Var   bMem:TbMem absolute Mem;{láncolt memória belsöábrázolásban}
59            i,
60            szabad:TCim;{a szabad elemek listájának fej-indexe}
61            szDb:Integer;{a szabad elemek pillanatnyi száma}
62
63      Procedure Lefoglal(Var i:TCim);
64      Begin
65        If szabad<>Sehova then
66        Begin
67          i:=szabad; szabad:=bMem[i].kov; Dec(szDb)
68        End
69          else
70        Begin
71          i:=Sehova
72        End;
73      End;
74
75      Procedure Felszabadit(Var i:TCim);
76      Begin
77        If i<>Sehova then
78        Begin
79          bMem[i].kov:=szabad; szabad:=i; i:=Sehova; Inc(szDb)
80        End;
81      End;
82
83      Procedure MemDump;
84        Var i:TCim;
85      Begin
86        Writeln('----------------------------------------'+
87                '----------------------------------------');
88        Writeln('A szabadlista feje:',szabad:4,
89                '. Szabad elemek száma:',szDb,
90                ', feltöltöttség:',100*(1-szDb/MaxMem):5:1,' % .');
91        For i:=1 to MaxMem do
92        Begin
93          HighVideo; Write(i:4,':');
94          NormVideo; Write(bMem[i].kov:4,'-');
95        End;
96        Writeln; Writeln;
97        Writeln('----------------------------------------'+
98                '----------------------------------------');
99        ReadKey;
100     End;
101
102   Begin {Memória-inicializálás:}
103     ClrScr;
104     Writeln('▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒ Memória modell (statiku'+
105             'san láncolt tömb) unit-ja ▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒');
106     For i:=1 to MaxMem-1 do bMem[i].kov:=i+1;
107     bMem[MaxMem].kov:=Sehova;
108     szabad:=1; szDb:=MaxMem;
109     MemDump;
```

```
110   Writeln('▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒ Inicializálás '+
111           'megtörtént... ▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒');
112   ReadKey
113 End.
```

1.3 A unit-ot kipróbáló program

A kód (MEMPROB.EXE) forrása (MEMPROB.PAS):

```
1   Program MemUnitProba;
2     Uses Newdelay,Crt,MemUnit;
3     {$i AltRutin.inc}
4     Var  i,j:0..MaxMem+1;
5          cim:Array [1..MaxMem+1] of TCim;
6          e:TElem;
7   Begin
8     UjLap('Lefoglalás, végkimerülésig',0);
9     i:=0;
10    Repeat
11      Inc(i); Lefoglal(cim[i]);
12      Writeln('A(z) ',i:2,'. foglaláskor kapott elem "memóriacíme":',
13              cim[i]);
14      If cim[i]<>Sehova then
15      Begin
16        Str(i,e);
17        mem[cim[i]]:=Copy(e+'._elem___',1,SizeOf(TElem));
18        Writeln('A memóriafoglalás sikeres volt.'+
19                ' A(z) ',i,'. elem értéke:',mem[cim[i]]);
20      End
21        else
22      Begin
23        Writeln('A memóriafoglalás nem sikerült.');
24      End;
25      MemDump;
26    Until cim[i]=Sehova;
27    UjLap('Az összes felszabadítása, söt...',-1);
28    For j:=1 to i do
29    Begin
30      Writeln('A(z) ',j:2,'.-ként lefoglalt elem felszabadítása,'+
31              ' melynek "memóriacíme":',cim[j]);
32      Felszabadit(cim[j]);
33      Writeln('... a címe a felszabadítás után:',cim[j]);
34      MemDump;
35    End;
36  End.
```

A fentiek együtt: 0.zip.

2. ALKALMAZÁSOK

2.1 Hiányos mátrix

A hiányosan kitöltött mátrixok kezeléséhez írt, „hagyományos”, láncolt ábrázolás mechanikus átalakításával.

A átalakítás lépései mechanikus cserékkel elvégezhetők. Pl.

- minden cím-típus (`^TValami`) helyettesítendő az univerzális `TCím` típussal
- `mut^`, `emut^`, vagy `Nil` helyett `mem[mut]`, `mem[emut]` ill. `Sehova` írandó
- `New` és `Dispose` helyett `Lefoglal` és `Felszabadít` írandó
- a `TElem` gyanánt a hiányos mátrix (láncolást tartalmazó) elemének típusa kerül, amelyet a MemUnit-ba át kell emelni (ennek rekurzivitása némi többlet figyelmet igényel)

Ezek után az igazított modul, a MemUnit1 így alakul `interface`-rész:

```
  Interface
    Const MaxMem=100;
          Sehova=0;
    Type  TCim=0..MaxMem;
          TElem=Record
            ert:Integer; i,j:Byte;
            sKov{sorban következö},oKov{oszlopban következö}:TCim
          End;
          TMem=Array [1..MaxMem] of TElem;
    Var   Mem:TMem;
    Procedure Lefoglal(Var i:TCim);
    Procedure Felszabadit(Var i:TCim);
    Procedure MemDump;
```

A kód: MEMUNIT1.PAS, és a hiányos mátrix modulja: HIANYMA1.PAS[1], valamint próbaprogramja: HIMXPRO1.PAS (és futó kódja: HIMXPRO1.EXE)

```
1   Unit HianyMa1;
2   (*
3     Hiányosan kitöltött négyzetes mátrixok statikusan láncolt ábrázolás
4     melletti megvalósítása. (A sima láncolt mechanikus átírásával kapva.)
5     Asszociált műveletek:
6     * Letrehoz {konstans mátrixot}
7     * Ki
8     * Be
9     * Osszead
10    * Kivon
11    * Szoroz
12    * SkalSzor
13    * HibasE
14    * ElemErtek
15    * ElemModosit
16    * Tomorit
17    Import:
18    * MemUnit
19  *)
20    Interface
21      Uses MemUni1;
22      Const
23        MaxN = 50;
```

[1] A program nem teljes, ui. a témához tartozó keretprogram kerül itt átírásra.

```
24      Type
25        THianyMat = Record
26                      n:Byte;
27                      ism:Integer;
28                      sFej,oFej:Array [1..MaxN] of TCim;{lehetne takarékosabban is}
29                      siker:Boolean;
30                    End;
31
32      Procedure Tomorit(Var m:THianyMat);
33      {Ef: -
34       Uf: m'^.ism=m leggyakoribb elemének értéke ES
35           m' elemei ELEME m-nek ES elemei<>m'.ism}
36
37      Procedure Letrehoz(Const n:Byte; ism:Integer; Var m:THianyMat);
38      {Ef: -
39       Uf: m.n=n ES m.ism=ism ES m.sFej=Nil ES m.oFej=Nil ES m.siker}
40
41      Procedure Ki(Const cim:String; Const m:THianyMat);
42      {Ef: -
43       Uf: ...}
44
45      Procedure Be(Const cim:String; Var m:THianyMat);
46      {Ef: -
47       Uf: ...}
48
49      Procedure Osszead(Const m1,m2:THianyMat;
50                        Var osszeg:THianyMat);
51      {Ef: -
52       Uf: ...}
53
54      Procedure Kivon(Const m1,m2:THianyMat;
55                      Var kulonbseg:THianyMat);
56      {Ef: -
57       Uf: ...}
58
59      Procedure Szoroz(Const m1,m2:THianyMat;
60                       Var szorzat:THianyMat);
61      {Ef: -
62       Uf: ...}
63
64      Procedure SkalSzor(Const m:THianyMat; Const skal:Integer;
65                         Var szorzat:THianyMat);
66      {Ef: -
67       Uf: ...}
68
69      Function HibasE(Const m:THianyMat):Boolean;
70      {Ef: -
71       Uf: ...}
72
73      Function ElemErtek(Const i,j:Byte; Const m:THianyMat):Integer;
74      {Ef: -
75       Uf: m-nek nem tárolt eleme az (i,j) => ElemErtek(i,j,m)=m.ism ES
76           m-nek tárolt eleme az (i,j) => ElemErtek(i,j,m)=tárolt érték}
77
```

```
78      Procedure ElemModosit(Const i,j:Byte; Const e:Integer;
79                            Var m:THianyMat);
80      {Ef: -
81       Uf: BARMELY ii<>i,jj<>j ELEME [1..m.n]: m' elemei = m elemei ES
82           m' (i,j) eleme = e}
83
84    Implementation
85      Uses
86        Newdelay,Crt;
87        {$i AltRutin.inc}
88
89      {Elem-keresö függvény: ----------------------------------------}
90
91      Function HolVan(Const i,j:Byte; Const m:THianyMat):TCim;
92      {Ef: (i,j)-szerint növekedöen rendezett
93       Uf: LETEZIK (i,j) ELEME m.(i,j) => Holvan^.(i,j)=(i,j)
94           NEM LETEZIK (i,j) ELEME m.(i,j) => Holvan=Sehova}
95        Var
96          mut:TCim;
97      Begin
98        mut:=m.sFej[i];
99        While (mut<>Sehova) and (mem[mut].j<j) do
100       Begin
101         mut:=mem[mut].sKov;
102       End;
103       If (mut=Sehova) or (mem[mut].j>j) then HolVan:=Sehova
104                                        else HolVan:=mut;
105     End; {HolVan}
106
107     Function SorbanElozo(Const i,j:Byte; Const m:THianyMat):TCim;
108     {Ef: (i,j)-szerint növekedöen rendezett
109      Uf: ...}
110       Var
111         mut,emut:TCim;
112     Begin
113       mut:=m.sFej[i];
114       If mut=Sehova then {nincs az i. sorban még elem}
115       Begin
116         SorbanElozo:=Sehova
117       End
118         Else
119       Begin
120         emut:=Sehova;
121         While (mut<>Sehova) and (mem[mut].j<j) do
122         Begin
123           emut:=mut; mut:=mem[mut].skov;
124         End;
125         SorbanElozo:=emut
126       End;
127     End; {SorbanElozo}
128
129     Function OszlopbanElozo(Const i,j:Byte; Const m:THianyMat):TCim;
130     {Ef: (i,j)-szerint növekedöen rendezett
131      Uf: ...}
132       Var
133         mut,emut:TCim;
```

```
134     Begin
135       mut:=m.oFej[j];
136       If mut=Sehova then {nincs a j. oszlopban még elem}
137       Begin
138         OszlopbanElozo:=Sehova
139       End
140         Else
141       Begin
142         emut:=Sehova;
143         While (mut<>Sehova) and (mem[mut].i<i) do
144         Begin
145           emut:=mut; mut:=mem[mut].skov;
146         End;
147         OszlopbanElozo:=emut
148       End;
149     End; {OszlopbanElozo}
150
151     {Egyebek: ----------------------------------------------------------}
152
153     Function ElemSzam(Const m:THianyMat):Integer;
154     {Ef: -
155      Uf: ElemSzam(m)=a láncolt elemek száma}
156       Var
157         i,j:Byte;
158         db:Integer;
159     Begin
160       db:=0;
161       For i:=1 to m.n do
162       Begin
163         For j:=1 to m.n do
164         Begin
165           If HolVan(i,j,m)<>Sehova then Inc(db);
166         End;
167       End;
168       ElemSzam:=db
169     End; {ElemSzam}
170
171     Procedure Tomorit(Var m:THianyMat);
172     {Ef: -
173      Uf: m'^.ism=m leggyakoribb elemének értéke ES
174          m' elemei ELEME m-nek ES elemei<>m'.ism}
175       Var
176         i,j,k,db,max:Byte;
177         mut,emut:TCim;
178         maxe,e:Integer;
179         gyak:Array [1..MaxN*MaxN] of
180               Record ert:Integer; db:Integer End;
181     Begin
182       {teljessé tétel -- m.ism értéküekkel kiegészítés:}
183       For i:=1 to m.n do
184       Begin
185         For j:=1 to m.n do
186         Begin
187           mut:=HolVan(i,j,m);
```

```
188             If mut=Sehova then {m.ism értékü elem}
189             Begin
190               Lefoglal(mut);
191               mem[mut].ert:=m.ism; mem[mut].i:=i; mem[mut].j:=j;
192               mem[mut].sKov:=Sehova; mem[mut].oKov:=Sehova;
193               emut:=SorbanElozo(i,j,m);
194               If emut=Sehova then {fej-hez láncolandó}
195               Begin
196                 mem[mut].sKov:=m.sFej[i]; m.sFej[i]:=mut;
197               End
198                 else {elemhez láncolandó}
199               Begin
200                 mem[mut].sKov:=mem[emut].sKov; mem[emut].sKov:=mut;
201               End; {emut}
202               emut:=OszlopbanElozo(i,j,m);
203               If emut=Sehova then {fej-hez láncolandó}
204               Begin
205                 mem[mut].oKov:=m.oFej[j]; m.oFej[j]:=mut;
206               End
207                 else {elemhez láncolandó}
208               Begin
209                 mem[mut].oKov:=mem[emut].oKov; mem[emut].oKov:=mut;
210               End; {emut}
211             End; {mut}
212           End; {j}
213         End; {i}
214   Ki('ellenörzés -- teljes',m);
215         {gyakoriság-számlálás:}
216         db:=0;
217         For i:=1 to m.n do
218         Begin
219           For j:=1 to m.n do
220           Begin
221             e:=mem[HolVan(i,j,m)].ert;
222             k:=1;
223             While (k<=db) and (e<>gyak[k].ert) do Inc(k);
224             If k>db then
225             Begin
226               Inc(db); gyak[db].ert:=e; gyak[db].db:=1;
227             End
228               else
229             Begin
230               Inc(gyak[k].db)
231             End;
232           End; {j}
233         End; {i}
234         {a leggyakoribb kiválasztás:}
235         max:=gyak[1].db; maxe:=gyak[1].ert;
236         For i:=2 to m.n do
237         Begin
238           If gyak[i].db>max then
239           Begin
240             max:=gyak[i].db; maxe:=gyak[i].ert;
241           End;
242         End;
243         {a leggyakoribb adminisztrálása, kifüzése:}
244         m.ism:=maxe;
```

```
245         For i:=1 to m.n do
246         Begin
247           For j:=1 to m.n do
248           Begin
249             mut:=HolVan(i,j,m); e:=mem[mut].ert;
250             If e=maxe then {kifüzendö}
251             Begin
252               emut:=SorbanElozo(i,j,m);
253               If emut=Sehova then {sor elejéröl}
254               Begin
255                 m.sFej[i]:=mem[mut].sKov; {még nem dobjuk el}
256               End
257                 else {sor belsejéböl}
258               Begin
259                 mem[emut].sKov:=mem[mut].sKov; {még nem dobjuk el}
260               End;
261               emut:=OszlopbanElozo(i,j,m);
262               If emut=Sehova then {oszlop elejéröl}
263               Begin
264                 m.oFej[j]:=mem[mut].oKov; Felszabadit(mut);
265               End
266                 else {oszlop belsejéböl}
267               Begin
268                 mem[emut].oKov:=mem[mut].oKov; Felszabadit(mut);
269               End; {emut}
270             End; {e}
271           End; {j}
272         End; {i}
273     Ki('ellenörzés -- tömörített',m);
274       End; {Tomorit}
275
276       {Operációk: ----------------------------------------------------------}
277
278       Procedure Letrehoz(Const n:Byte; ism:Integer; Var m:THianyMat);
279       {Ef: -
280        Uf: m.n=n ES m.ism=ism ES m.sFej=Nil ES m.oFej=Nil ES m.siker}
281         Var
282           i,j:Byte;
283       Begin
284         m.n:=n;
285         m.ism:=ism;
286         For i:=1 to MaxN do
287         Begin
288           m.sFej[i]:=Sehova; m.oFej[i]:=Sehova
289         End;
290         m.siker:=True;
291       End; {Letrehoz}
292
293       Procedure Ki(Const cim:String; Const m:THianyMat);
294       {Ef: -
295        Uf: ...}
296         Var
297           i,j:Byte;
298       Begin
299         UjLap(cim,0);
```

```
300         For i:=1 to m.n do
301         Begin
302           Writeln(i:3,'. sor:');
303           For j:=1 to m.n do
304           Begin
305             Write(ElemErtek(i,j,m):4)
306           End; {j}
307           Writeln;
308         End; {i}
309         Writeln('Tárolt elemek száma:',ElemSzam(m));
310         BillreVar;
311       End; {Ki}
312
313       Procedure Be(Const cim:String; Var m:THianyMat);
314       {Ef: -
315        Uf: ...}
316       Begin
317         {Ötlet:
318          * beolvasni a "teljes" mátrixot
319          * és tömöríteni...}
320       End; {}
321
322       Procedure Osszead(Const m1,m2:THianyMat;
323                         Var osszeg:THianyMat);
324       {Ef: -
325        Uf: ...}
326       Begin
327         {...}
328       End; {}
329
330       Procedure Kivon(Const m1,m2:THianyMat;
331                       Var kulonbseg:THianyMat);
332       {Ef: -
333        Uf: ...}
334       Begin
335         {...}
336       End; {}
337
338       Procedure Szoroz(Const m1,m2:THianyMat;
339                        Var szorzat:THianyMat);
340       {Ef: -
341        Uf: ...}
342       Begin
343         {...}
344       End; {}
345
346       Procedure SkalSzor(Const m:THianyMat; Const skal:Integer;
347                          Var szorzat:THianyMat);
348       {Ef: -
349        Uf: ...}
350       Begin
351         {...}
352       End; {}
353
```

```
354     Function HibasE(Const m:THianyMat):Boolean;
355     {Ef: -
356      Uf: ...}
357     Begin
358       {...}
359     End; {}
360
361     Function ElemErtek(Const i,j:Byte; Const m:THianyMat):Integer;
362     {Ef: -
363      Uf: m-nek nem tárolt eleme az (i,j) => ElemErtek(i,j,m)=m.ism ES
364          m-nek tárolt eleme az (i,j) => ElemErtek(i,j,m)=tárolt érték}
365       Var
366         mut:TCim;
367     Begin
368       mut:=HolVan(i,j,m);
369       If mut=Sehova then ElemErtek:=m.ism
370                     else ElemErtek:=mem[mut].ert
371     End; {ElemErtek}
372
373     Procedure ElemModosit(Const i,j:Byte; Const e:Integer;
374                           Var m:THianyMat);
375     {Ef: -
376      Uf: BARMELY ii<>i,jj<>j ELEME [1..m.n]: m' elemei = m elemei ES
377          m' (i,j) eleme = e}
378       Var
379         emut,mut:TCim;
380     Begin
381       If m.ism<>e then {tárolandó elem esete}
382       Begin
383         mut:=HolVan(i,j,m);
384         If mut<>Sehova then {van ilyen elem tárolva}
385         Begin
386           mem[mut].ert:=e
387         End
388           else {nincs tárolva ez az elem}
389         Begin
390           {MatElem létrehozás:}
391           Lefoglal(mut); mem[mut].ert:=e; mem[mut].i:=i; mem[mut].j:=j;
392           {láncolások:}
393           emut:=SorbanElozo(i,j,m);
394           If
395             m.sFej[i]=Sehova then {üres még a sor}
396              Begin
397                m.sFej[i]:=mut; mem[mut].sKov:=Sehova;
398              End else if
399             emut=Sehova then {sorelsőként illesztendő be}
400              Begin
401                mem[mut].sKov:=m.sFej[i]; m.sFej[i]:=mut;
402              End
403             else
404              Begin
405                mem[mut].sKov:=mem[emut].sKov; mem[emut].sKov:=mut;
406              End;
407           {EndIf}
408           emut:=OszlopbanElozo(i,j,m);
```

```
409             If
410               m.oFej[j]=Sehova then {üres még az oszlop}
411                Begin
412                  m.oFej[j]:=mut; mem[mut].oKov:=Sehova;
413                End else if
414               emut=Sehova then {oszlopelsőként illesztendő be}
415                Begin
416                  mem[mut].oKov:=m.oFej[j]; m.oFej[j]:=mut;
417                End
418               else
419                Begin
420                  mem[mut].oKov:=mem[emut].oKov; mem[emut].oKov:=mut;
421                End;
422             {EndIf}
423           End; {mut}
424         End
425           else {nem tárolandó elem esete}
426         Begin
427           mut:=HolVan(i,j,m);
428           If mut<>Sehova then {most már fölöslegesen tárolt elem: törlendö}
429           Begin
430             emut:=SorbanElozo(i,j,m);
431             If emut=Sehova then m.sFej[i]:=mem[mut].sKov
432                           else mem[emut].sKov:=mem[mut].sKov;
433             emut:=OszlopbanElozo(i,j,m);
434             If emut=Sehova then m.oFej[j]:=mem[mut].oKov
435                           else mem[emut].oKov:=mem[mut].oKov;
436             Felszabadit(mut);
437           End;
438         End; {m.ism}
439       End; {ElemModosit}
440
441   Begin
442   End.
```

E példának mindene, együtt: 1.zip.

2.2 Bináris fa

A bináris fa kezeléséhez írt, „hagyományos”, láncolt ábrázolás mechanikus átalakításával.

A mechanikus lépések cserékkel elvégezhetők. Pl.

- az eredeti, BinFa pontjainak megfelelő típust (`TElem`) kicseréljük egy konkrét (pl. `Integer`) típussal (mind a BinFa unit-ban, mind a főprogramban)
- a memória egység típusául szolgáló `BinFaElem` típus definíciója `TElem` néven átemelendő a MemUnit-ba,
- és benne a `^TBinFaElem` kicserélendő az analóg MemUnit2-beli `TCím` fogalomra
- minden valós címhivatkozás, azaz *`valami`*`^` kicserélendő `mem[`*`valami`*`]`-re, így teszünk a `mut^`-tal, `emut^`-tal, azaz helyettesítjük a `mem[mut]`-tal, `mem[emut]`-tal
- a címet szimbolizáló `Nil` konstans helyett a `Sehova` írandó
- `New` és `Dispose` helyett `Lefoglal` és `Felszabadít` írandó

A módosult `interface`-rész:

```
Interface
  Const MaxMem=100;
        Sehova=0;
  Type  TCim=0..MaxMem;
        TElem=Record
                ert: Integer;
                bal,jobb: TCim;
                hDb: Byte; {elemre hivatkozás-számláló}
              End;
        TMem=Array [1..MaxMem] of TElem;
  Var   Mem:TMem;
  Procedure Lefoglal(Var i:TCim);
  Procedure Felszabadit(Var i:TCim);
  Procedure MemDump;
```

A kód: MEMUNIT2.PAS, és a BinFa modulja: BINFAUN2.PAS (BinFa2.exp – exportmodul, BinFa2.rim – megvalósítási modul), valamint próbaprogramja: KERFA2P2.PAS (és futó kódja: KERFA2P2.EXE)

```
1  (*
2    A memóriamodell példájához igazítva.
3    A Binfa.rim-ből kiindulva.
4  *)
5  (* BINFA2.RIM -- Reprezentacios-Implementacios Modul *)
6
7  {
8    Bináris fa rekurzív modulja(paramétere: Integer):
9
10   Modul TBinFa2(Típus Integer):
11 }
12    Procedure InchDb(Const bf:TBinFa); {a bf minden elemének hivatkozás-
13                                        számlálóját inkrementálja}
14    Begin
15      If not UresEBF(bf) then
16      Begin
17        Inc(mem[bf].hDb);
18        InchDb(BalBf(bf)); InchDb(JobbBf(bf));
19      End
20    End;
21
22    {Const}Function UresBF: TBinFa;
23    Begin
24      UresBF:=Sehova;
25    End;
26
27    Function  UresEBF(Const bf:TBinFa): Boolean;
28    Begin
29      UresEBF:=bf=Sehova;
30    End;
31
```

```
32      Function  Gyoker (Const bf:TBinFa): Integer;
33      Begin
34        Gyoker:=mem[bf].ert
35      End;
36
37      Function  BalBF  (Const bf:TBinFa): TBinFa;
38      Begin
39        BalBF:=mem[bf].bal
40      End;
41
42      Function  JobbBF (Const bf:TBinFa): TBinFa;
43      Begin
44        JobbBF:=mem[bf].jobb
45      End;
46
47      Function  EgyElem(Const e:Integer): TBinfa;
48        Var hova:TBinfa;
49      Begin
50        Lefoglal(hova);
51        With mem[hova] do
52        Begin
53          ert:=e; bal:=UresBF; jobb:=UresBF; hDb:=0;
54        End;
55        EgyElem:=hova
56      End;
57
58      Procedure BalraIllesztBF(Var bf:TBinFa; Const bf2:TBinFa);
59      Begin
60        If mem[bf].bal<>bf2 then {InchDb(bf2)} Inc(mem[bf2].hDb);
61        mem[bf].bal:=bf2;
62      End;
63
64      Procedure JobbraIllesztBF(Var bf:TBinFa; Const bf2:TBinFa);
65      Begin
66        If mem[bf].jobb<>bf2 then {InchDb(bf2)} Inc(mem[bf2].hDb);
67        mem[bf].jobb:=bf2;
68      End;
69
70      Procedure GyokerModosit(Var bf:TBinFa; Const e:Integer);
71      Begin
72        mem[bf].ert:=e
73      End;
74
75      Function GyokerhDb(Const bf:TBinFa):Byte;
76      Begin
77        GyokerhDb:=mem[bf].hDb
78      End;
79
80  {
81    Modul vége.
82  }
```

E példának mindene együtt: 2.zip. A teljes anyag letöltése egyben: MemModell.zip.