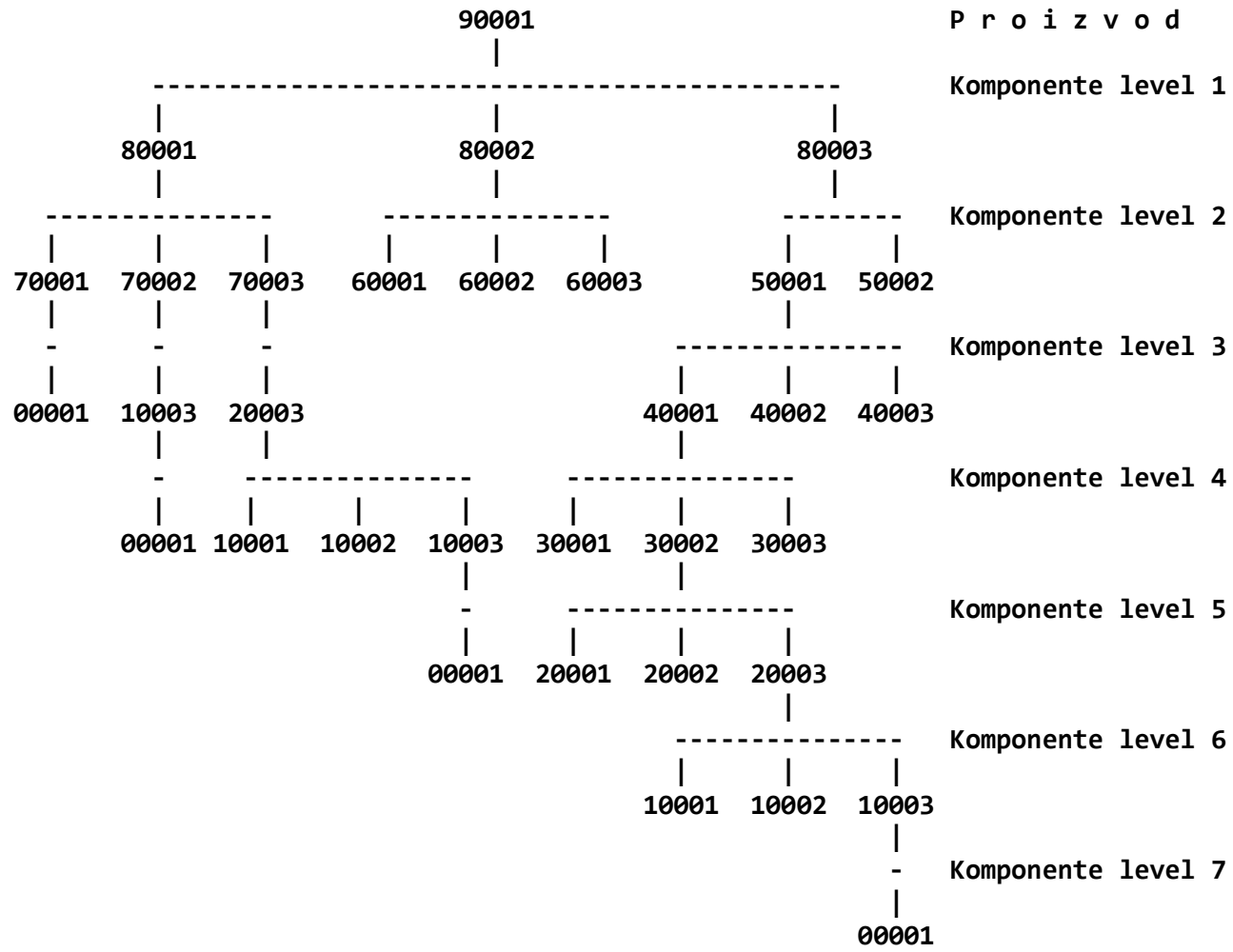
SOFTVERSKO REŠENJE CSYSTEMS™
ZA ULAZNI-IZLAZNI DIJAGRAM
BOM SASTAVNICE ZA PROIZVOD

Stanojević Slobodan, dipl.ing – Blagojević Dragan, dipl.ing, 25.12.2012

Slika 1.



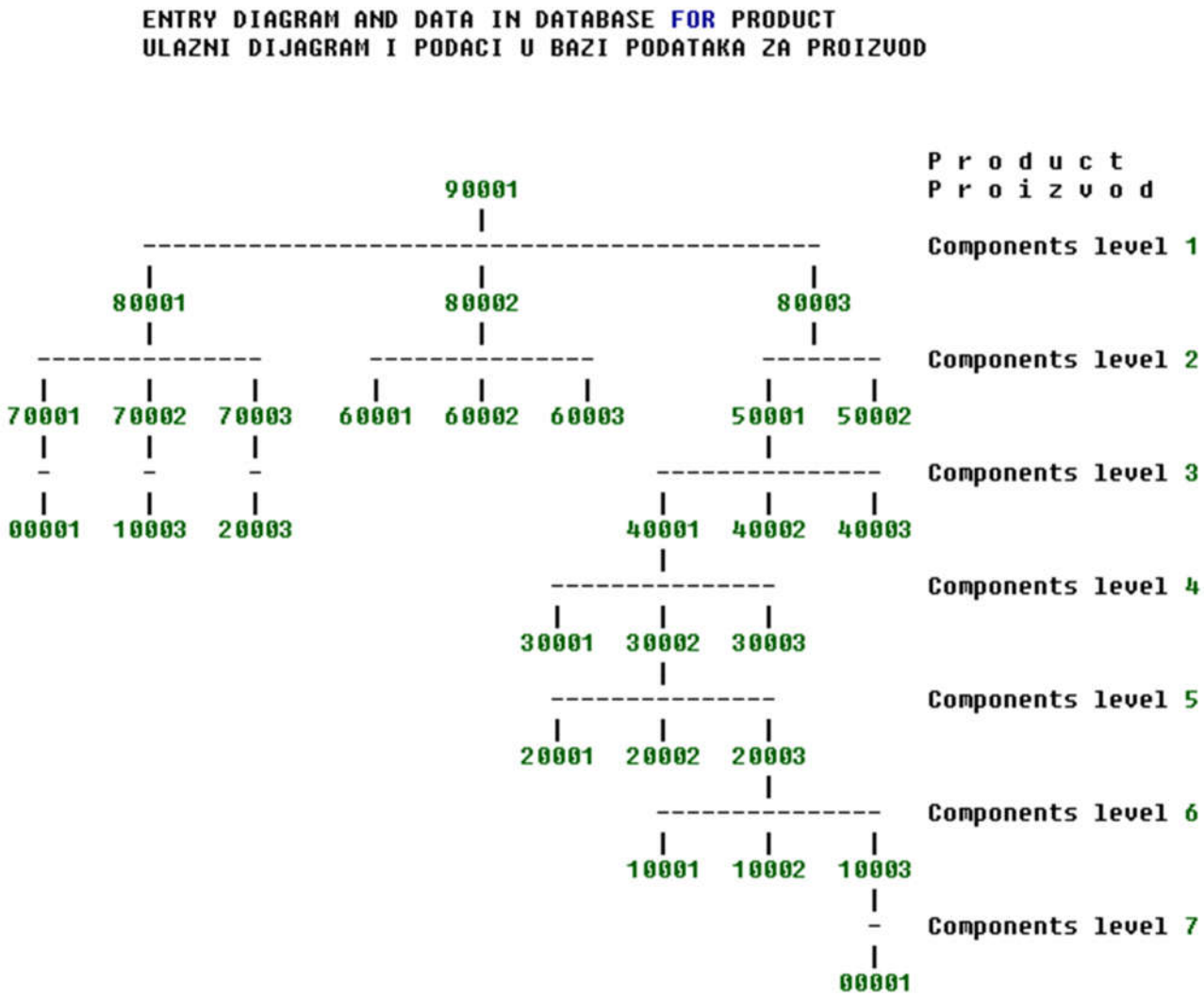
Zadatak:
Na dijagramu (slika 1) je dat proizvod 90001 i njegove komponente (poluproizvodi i delovi) od kojih je proizvod sačinjen. Brojevi 90001, 80001, 70001 su šifre proizvoda i komponenti a ujedno su i brojevi njihovih BOM sastavnica. Svi podaci sa dijagrama upisani su u bazu podataka datu na slici (slika 3) i računar ih ima u tom obliku. Izborom bilo kog broja sa ovog dijagrama mora se dobiti lista svih njegovih komponenti, koju treba da formira program odnosno aplikacija. Ova lista mora da bude organizovana po nivoima i po pripadnosti i rasporedu komponenti (slika 6), istovetno kao što je rganizovana na dijagramu (slika 1 i 4).

==========
BOM (Bill of materials)
==========

DEMONSTRATION PROGRAM FOR THE FORMING OF COMPONENTS PRODUCT (Bill of Materials)
Algorithm and code given by technique of forming BOM components: COBA Systems 2012
The function of the database product and its components:
lista1 (array - Memory array)
forming levels of all the components that are integral parts of the product, and then formed a separate list for each component of the first level with all its accompanying components, according to the given diagram:

DEMONSTRACIJA RADA PROGRAMA ZA FORMIRANJE SASTAVNICE PROIZVODA (Bill of materials)
Algoritam i kod dati prema Tehnici formiranja BOM sastavnice: COBA Systems 2012
Ova funkcija iz baze podataka proizvoda i njegovih komponenti :
lista1 (array - memorijski niz)
formira nivoe (level) svih komponenti koje su sastavni delovi proizvoda, a zatim formira posebnu listu za svaku komponentu prvog nivoa sa svim njenim pripadajućim komponentama, a prema datom dijagramu:

slika 2



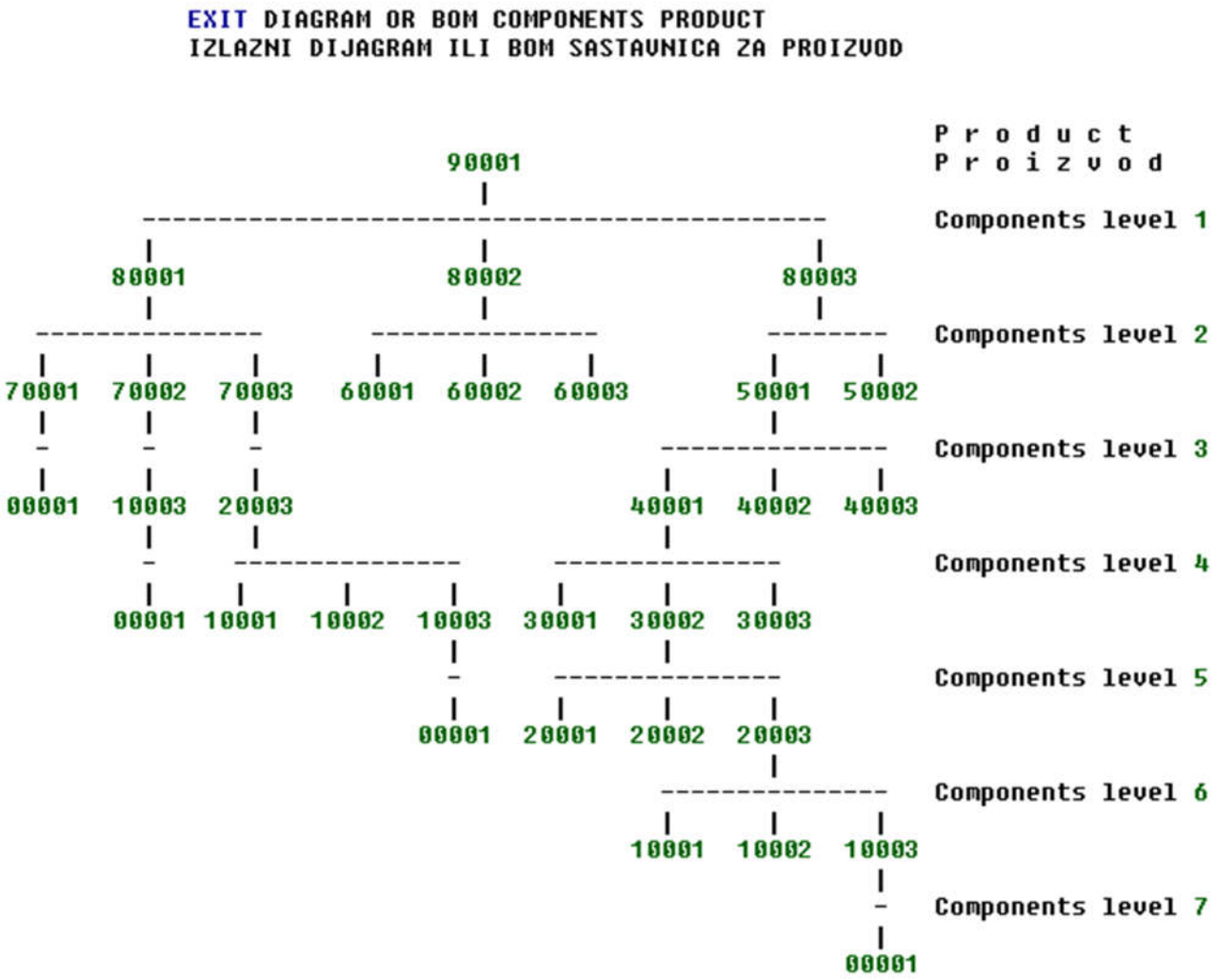
Slika 3

In this example it was used databases:

Lista1 for input diagram
Lista1 za ulazni dijagram

ID	Link	Level	
90001	90001	0	Product
80001	90001	0	component level 1
80002	90001	0	component level 1
80003	90001	0	component level 1
70001	80001	0	
70002	80001	0	
70003	80001	0	
60001	80002	0	
60002	80002	0	
60003	80002	0	
50001	80003	0	
50002	80003	0	
40001	50001	0	
40002	50001	0	
40003	50001	0	
30001	40001	0	
30002	40001	0	
30003	40001	0	
20001	30002	0	
20002	30002	0	
20003	30002	0	
10001	20003	0	
10002	20003	0	
10003	20003	0	
00001	10003	0	
00001	70001	0	
10003	70002	0	
20003	70003	0	

Slika 4


EXIT DIAGRAM OR BOM COMPONENTS PRODUCT
IZLAZNI DIJAGRAM ILI BOM SASTAVNICA ZA PROIZVOD
Product
Proizvod
90001
Components level 1
80001
80002
80003
Components level 2
70001
70002
70003
60001
60002
60003
50001
50002
Components level 3
00001
10003
20003
40001
40002
40003
Components level 4
00001
10001
10002
10003
30001
30002
30003
Components level 5
00001
20001
20002
20003
Components level 6
10001
10002
10003
Components level 7
00001

Slika 5

Results obtained from the procedure for the formation levels:

Lista1 for exit diagram
Lista1 za izlazni dijagram

No	ID	Link	level
1	80001	90001	1
2	80002	90001	1
3	80003	90001	1
4	70001	80001	2
5	70002	80001	2
6	70003	80001	2
7	60001	80002	2
8	60002	80002	2
9	60003	80002	2
10	50001	80003	2
11	50002	80003	2
12	00001	70001	3
13	10003	70002	3
14	20003	70003	3
15	40001	50001	3
16	40002	50001	3
17	40003	50001	3
18	00001	10003	4
19	10001	20003	4
20	10002	20003	4
21	10003	20003	4
22	30001	40001	4
23	30002	40001	4
24	30003	40001	4
25	00001	10003	5
26	20001	30002	5
27	20002	30002	5
28	20003	30002	5
29	10001	20003	6
30	10002	20003	6
31	10003	20003	6
32	00001	10003	7

Slika 6

```
Results obtained from the procedure for
the formation of tree diagram:

List: ListaTree is an ordered list 1

Organized Lista1 contains special lists
for each component of the first level,
with a all its components-parts

Lista: ListaTree je Uređena Lista 1

Uređena Lista1 sadrži posebne liste
za svaku komponentu prvog nivoa, sa
svim njenim komponentama-delovima:


-----------------------------------------
1  80001 90001 1    Component level 1
4    70001 80001 2
12    00001 70001 3
5    70002 80001 2
13    10003 70002 3
18     00001 10003 4
6    70003 80001 2
14    20003 70003 3
19     10001 20003 4
20     10002 20003 4
21     10003 20003 4
25      00001 10003 5
-----------------------------------------
2  80002 90001 1    Component level 1
7    60001 80002 2
8    60002 80002 2
9    60003 80002 2
-----------------------------------------
3  80003 90001 1    Component level 1
10   50001 80003 2
15    40001 50001 3
22     30001 40001 4
23     30002 40001 4
26      20001 30002 5
27      20002 30002 5
28      20003 30002 5
29       10001 20003 6
30       10002 20003 6
31       10003 20003 6
32        00001 10003 7
24     30003 40001 4
16    40002 50001 3
17    40003 50001 3
11   50002 80003 2
-----------------------------------------
© COBA Systems Software www.cobasystems.com
Stanojević Slobodan + Blagojević Dragan, 2012
Source code 32 bit Alaska Xbase++ (100,00 EUR)
```

```
FUNCTION COBA__BOM_CODE1()
    build_tree()
RETURN nil
```

```
FUNCTION build_database()

// generisanje liste proizvoda i njegovih komponenti:
LOCAL lista1 := {}
// Proizvod
     AADD(lista1,{"90001","90001",0})
// Lista komponenti proizvoda
// NIVO 1
     AADD(lista1,{"80001","90001",0})
     AADD(lista1,{"80002","90001",0})
     AADD(lista1,{"80003","90001",0})
// NIVO 2
     AADD(lista1,{"70001","80001",0})
     AADD(lista1,{"70002","80001",0})
     AADD(lista1,{"70003","80001",0})

     AADD(lista1,{"60001","80002",0})
     AADD(lista1,{"60002","80002",0})
     AADD(lista1,{"60003","80002",0})

     AADD(lista1,{"50001","80003",0})
     AADD(lista1,{"50002","80003",0})
// NIVO 3
     AADD(lista1,{"40001","50001",0})
     AADD(lista1,{"40002","50001",0})
     AADD(lista1,{"40003","50001",0})
// NIVO 4
     AADD(lista1,{"30001","40001",0})
     AADD(lista1,{"30002","40001",0})
     AADD(lista1,{"30003","40001",0})
// NIVO 5
     AADD(lista1,{"20001","30002",0})
     AADD(lista1,{"20002","30002",0})
     AADD(lista1,{"20003","30002",0})
// NIVO 6
     AADD(lista1,{"10001","20003",0})
     AADD(lista1,{"10002","20003",0})
     AADD(lista1,{"10003","20003",0})
// NIVO 7
     AADD(lista1,{"00001","10003",0})

// naknadno dodato...
     AADD(lista1,{"00001","70001",0})
     AADD(lista1,{"10003","70002",0})
     AADD(lista1,{"20003","70003",0})

RETURN lista1
```

```
FUNCTION build_tree()

LOCAL j
PRIVATE Lista1 := {}
// napravi izvornu bazu podataka brojeva sastavnica
// ( slika ULAZNI DIJAGRAM )
// gde svaki proizvod i komponenta ima jednu svoju sastavnicu
*****************************
 Lista1 := build_database()
*****************************
// Iz izvorne baze podataka napravi listu svih komponenti proizvoda
// ( slika IZLAZNI DIJAGRAM )
// odredi hijerarhijske nivoe komponenti u listi i upiši ih u listu
*******************************
 lista1 := build_level(lista1)
*******************************

   prvi      := lista1[1,1]  // PRVI ČLAN JE PROIZVOD == "90001"
   control   := {}           // kontrolna lista
   listatree := {}
   finale    := {}

// izbaci proizvod iz liste jer se lista na dalje
// mora koristiti bez proizvoda:

  adel(lista1,1)
  asize(lista1,len(lista1)-1)
  * Lista1 = "sada ne sadrži proizvod"

 // izdvoji samo prvi nivo komponenti u posebnu listu: Finale
 for j=1 to len(lista1)
   if lista1[j,2]==prvi
      AADD(finale,{lista1[j,1], lista1[j,2], lista1[j,3]})
   endif
 next j

 // Svakoj komponenti iz prvog nivoa komponenti
 // pridruži sve njene pripadajuće komponente i
 // dobijenu listu uredi po hijerahiji nivoa,
 // pa je upiši u finalnu listu: LlistaTree

 FOR j=1 TO len(finale)
                           // za funkciju: control_tree()
     control         := {}  // izbegavanje dupliranja komponenti
     jedan_do_drugog := "*" // izbegavanje dupliranja komponenti


    AADD(listatree, {finale[j,1], finale[j,2], finale[j,3]} )

    ***************************
    build_tree_part(finale[j,1])
    ***************************

 NEXT j

stampaj(listatree)

RETURN listatree
```

```
// R E K U R Z I J A
FUNCTION build_tree_part(prvi1)

LOCAL j := 0
FOR j=1 TO len(lista1)

    IF lista1[j,2] == prvi1

        * msgbox( prvi1 +chr(13)+lista1[j,1]+" "+lista1[j,2]+" "+;
        * padr(lista1[j,3],2), padr(j,2) ) // TEST

            vrati := lista1[j,1]

            // Brojevi se ne smeju duplirati u listatree osim kada je nivo
            // sledećeg nađenog broja za jedan veći od nivoa identičnog a
            // prethodno upotrebljenog broja i ako pri tome prethodni i sledeći
            // broj nisu jedan do drugog - ovo se kontroliše iz control_tree()

            IF control_tree( lista1[j,1],lista1[j,2],lista1[j,3] ) = .T.
                 // proveren je i ne sme se duplirati - upisati u listatree
            ELSE
                 // nađen je drugi broj - upiši ga u listatree
                    AADD(listatree,{lista1[j,1], lista1[j,2], lista1[j,3]})

                      // upiši i u kontrolnu listu
                    AADD(control, {lista1[j,1], lista1[j,2], lista1[j,3]} )

                     //---------------------------------------------------------------
                     // brojevi se smeju duplirati ali samo ako nisu jedan do drugog
                     // zato zapamti ovaj pa ako je sledeći identičan ne uzimaj ga:
                     jedan_do_drugog := lista1[j,1]+lista1[j,2]

                     // svaki upotrebljen broj ukloni sa radne liste: lista1
                     // kako ga sledeće pretraživanje liste ne bi ponovo našlo:
                     lista1[j,1] := "00000" ; lista1[j,2] := "00000"
                     //---------------------------------------------------------------
            ENDIF

        ****************************
         build_tree_part(vrati)
        ****************************

    ENDIF //  IF lista1[j,2] == prvi1
NEXT j
RETURN nil
```

```
FUNCTION control_tree(sas1,sas2,sas3)
/*
 Zaštita od dupliranja komponenti u listatree
 Ako je nađena dupla komponenta
 vraća .T. nađen je broj koji je upisan u listatree - ne upisuj ga - ne dupliraj
 vraća .F. nađen je broj koji nije upisan u listatree - upiši ga

 IF control_lista(lista1[i,1],lista1[i,2]) = .T.
      // nađen je broj koji je upisan u listatree - ne upisuj ga - ne dupliraj
 ELSE
      // nađen je broj koji nije upisan u listatree - upiši ga
 ENDIF

 Objašnjenje:
 Brojevi se ne smeju duplirati u listatree osim kada je nivo
 sledećeg nađenog broja za jedan veći od nivoa identičnog a
 prethodno upotrebljenog broja i ako pri tome prethodni i sledeći
 broj nisu jedan do drugog - ovo se kontroliše iz control_tree()
*/

* Funkcija zahteva array listu control := {} i varijablu: jedan_do_drugoga

LOCAL i
FOR i=1 TO len(control)

        // ako je nađen broj B isti kao broj A koji je već prethodno upisan u listatree
        IF control[i,1] == sas1 .and. ; // ako je isti broj sastavnice
           control[i,2] == sas2         // ako je isti broj veze

           * msgbox("Prethodni broj: "+;
           * jedan_do_drugog+chr(13)+"sadašnji broj: "+sas1+sas2,procname())

           // 1. ako je nađeni broj B isti kao već upisani broj A
           //    a nivo nađenog broja B je veći za 1 od nivoa upisanog broja A ...
           //    sme se ili ne sme se - ponovo upisati u listatree samo ako...
           IF control[i,3] + 1 == sas3

              IF jedan_do_drugog == sas1+sas2
              // 2. ako je nađeni broj B prvi sledeći broj posle upisanog broja A
              //    ne sme se upisati u listatree
                 RETURN .T. // ne uzimaj komponentu u listatree
              ELSE
              // 2. ako nađeni broj B nije prvi sledeći broj posle upisanog broja A
              //    sme se upisati u listatree
                 RETURN .F. // uzmi komponentu u listatree
              ENDIF

           ENDIF

          // ako je ista komponenta po drugi put nađena nemoj da dupliraš !!!

           RETURN .T. // ne uzimaj komponentu u listatree
        ENDIF

NEXT i
RETURN .F. // uzmi komponentu u listatree
```

```
FUNCTION build_level(lista)

     PRIVATE ;
     control   := {}     ,;
     lista_deo := lista  ,; // radna lista
     lista_sum := {}     ,; // finalna uređena lista
     lista_tmp := {}        // pomoćna temporary lista


  // PROIZVOD
  // dodaj proizvod u finalnu listu: lista_sum (nivo=0)
     level := 0
     AADD(lista_sum,{lista_deo[1,1],lista_deo[1,2],level})

  // izbaci proizvod iz radne liste: lista
     ADEL(lista_deo,1)
     ASIZE(lista_deo,len(lista_deo)-1)

  // veza za izdvajanje prvog nivoa komponenti proizvoda:
     veza := lista_sum[1,1]

  // KOMPONENTE LEVEL 1
  // upiši u liste: lista_sum i lista_tmp prvi nivo komponenti
     level := level + 1

     build_level1(veza)

  // KOMPONENTE LEVEL 2,3,4...

     DO WHILE .T.

     level := level + 1

     control := {}  // izbegavanje dupliranja komponenti

       if len(lista_tmp) > 0

          build_level2() // puni lista_sum

       else
          exit      // nema više komponenti
       endif

     ENDDO

RETURN lista_sum
```

```
STATIC FUNCTION build_level1(veza)

LOCAL j

FOR J=1 TO len(lista_deo)

   IF alltrim(lista_deo[j,2]) == alltrim(veza)

        IF control_level( lista_deo[j,1],lista_deo[j,2] ) = .T.  // izbegavanje dupliranja
komponenti
             // nađena ista sastavnica - ne dupliraj u listatree
        ELSE

           // upiši komponentu u :
           AADD( lista_sum, {lista_deo[j,1], lista_deo[j,2], level})
           // upiši komponentu u :
           AADD( lista_tmp, {lista_deo[j,1], lista_deo[j,2], level})
           // upiši u :
           AADD( control, {lista_deo[j,1], lista_deo[j,2]})

        ENDIF

   ENDIF

NEXT j

RETURN nil
```

```
STATIC FUNCTION build_level2()

LOCAL j
LOCAL lista_nul := lista_tmp
LOCAL veza

lista_tmp  := {}

FOR j=1 TO len(lista_nul)

  veza := lista_nul[j,1]

  build_level1(veza)

NEXT j

RETURN nil
```

```
FUNCTION control_level(sas1,sas2)

/*
Zaštita od dupliranja komponenti
 na glavnom spisku komponenti
 Ako je nađena dupla komponenta
 vraća .T. (NAĐENO/DUPLIRANO) ili
 vraća .F. (NIJE NAĐENO - NIJE DUPLIRANO)

 IF control_lista(lista1[i,1],lista1[i,2]) = .T.
      // nađena ista sastavnica - ne dupliraj u listatree
 ELSE
      // nađena druga sastavnica - upiši u listatree
 ENDIF
*/
LOCAL i
FOR i=1 TO len(control)

        IF control[i,1] == sas1 .and. ; // ako je isti broj sastavnice
           control[i,2] == sas2         // ako je isti broj veze

          // ako je ista komponenta po drugi put nađena nemoj da dupliraš !!!

           RETURN .T. // nađeno
        ENDIF

NEXT i
RETURN .F. // nije nađeno
```

```
FUNCTION stampaj(lista)

LOCAL i, txt, q
txt := strtran(appname(.t.),appname())+"sastavnica.txt"
DELETE FILE (txt)
SET PRINTER TO (txt)
SET PRINT ON
SET CONSOLE OFF
SET MARGIN TO 2

           for i = 1 to len(lista)

              ? lista[i,1]+" "+lista[i,2]+" "+PADR(lista[i,3],2)

            next i

SET PRINTER TO
SET PRINT OFF
SET CONSOLE ON
SET MARGIN TO
q := RunShell(txt,"notepad.exe")
RETURN NIL
```

COBA Systems

www.cobasystems.com