Krafft

X H E L P

Felhasználói kézikönyv

Marx Károly Közgazdaságtudományi Egyetem
Matematikai és Számitástudományi Intézet

1982.

T a r t a l o m

0 - 0

X H E L P

Általános ismertető

MKKE - MSZI

1982.

1. BEVEZETÉS

Az XHELP a Számitástechnikai Koordinációs Intézet és az MKKE Matematikai és Számitástudományi Intézet konprodukciójában kidolgozás alatt levő programfejlesztői rendszer. Célja a teljes programfejlesztési folyamat automatizálása, ill. a különböző fázisok segédeszközökkel való ellátása.

Az XHELP jelen változatban PASCAL nyelvi orientációju, azaz a tervezői nyelvben PASCAL nomenklatura szerint dolgozik, a generált program pedig PASCAL nyelvü.*

2. AZ XHELP KONCEPCIÓ

A koncepcióban központi helyet foglal el egy Fejlesztői Adatszótár (Data Dictionary), mely egy programrendszer fejlesztésekor a fejlesztés összes lényeges információját tartalmazza. Az XHELP alrendszerei a DD-vel szoros kapcsolatban állnak, adataikat oda töltik, ill. onnan veszik.

Az XHELP alrendszerek:

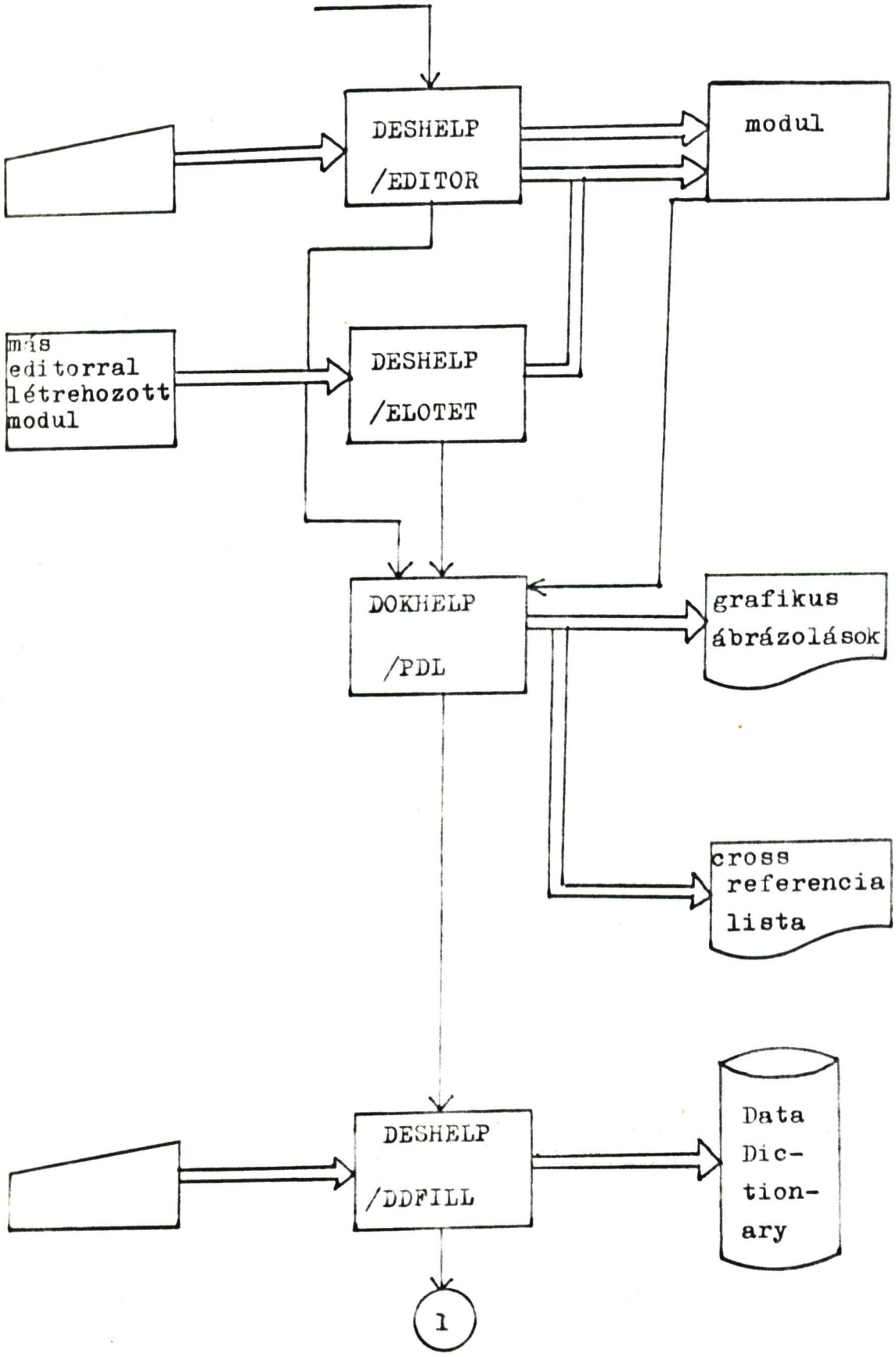
- DESHELP : tervezési alrendszer,
- PROGHELP : program generátor alrendszer,
- TESZTHELP : teszt processzor,
- SPECHELP : a Fejlesztői Adatszótárt kezelő alrendszer
- DOKHELP : dokumentáló alrendszer.

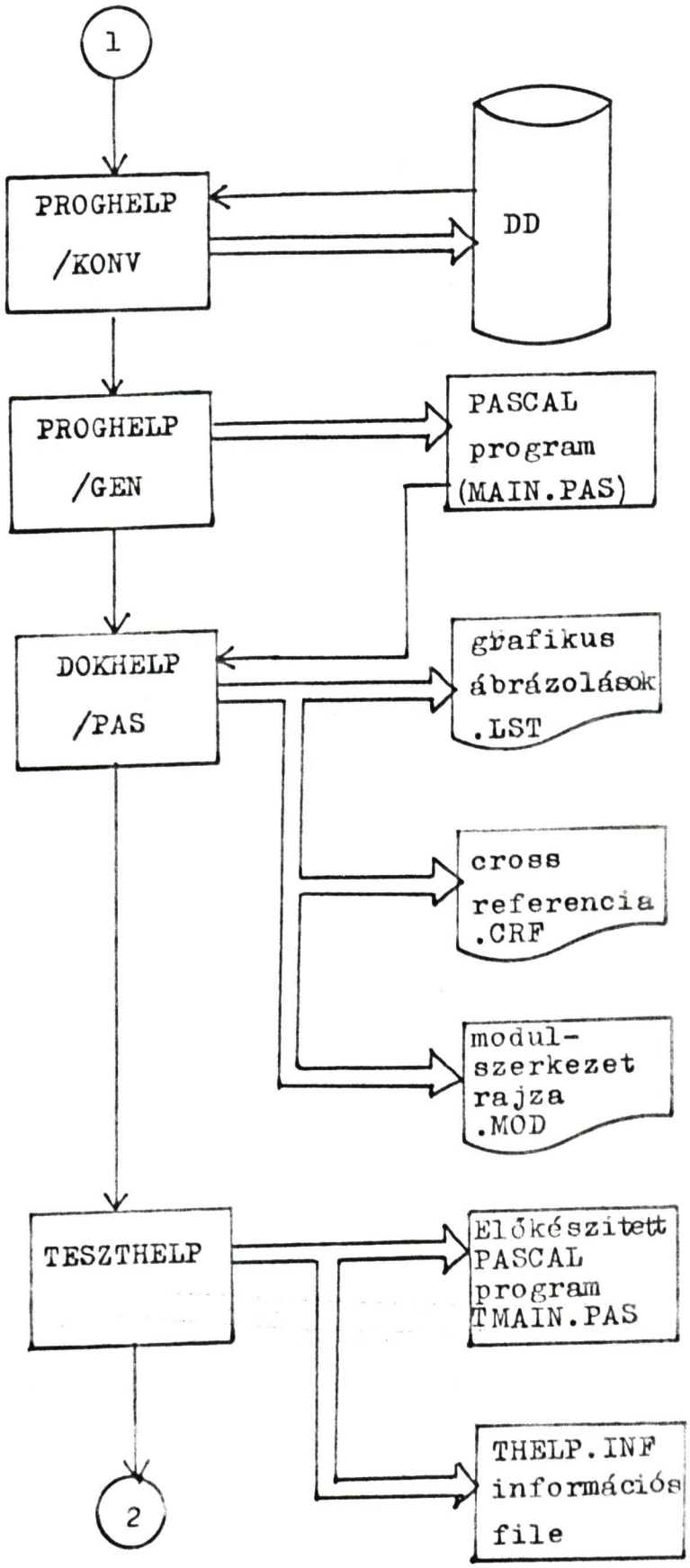
A DESHELP tovább bontható az EDITOR alrendszerre és az Adatszótárt töltő alrendszerre.

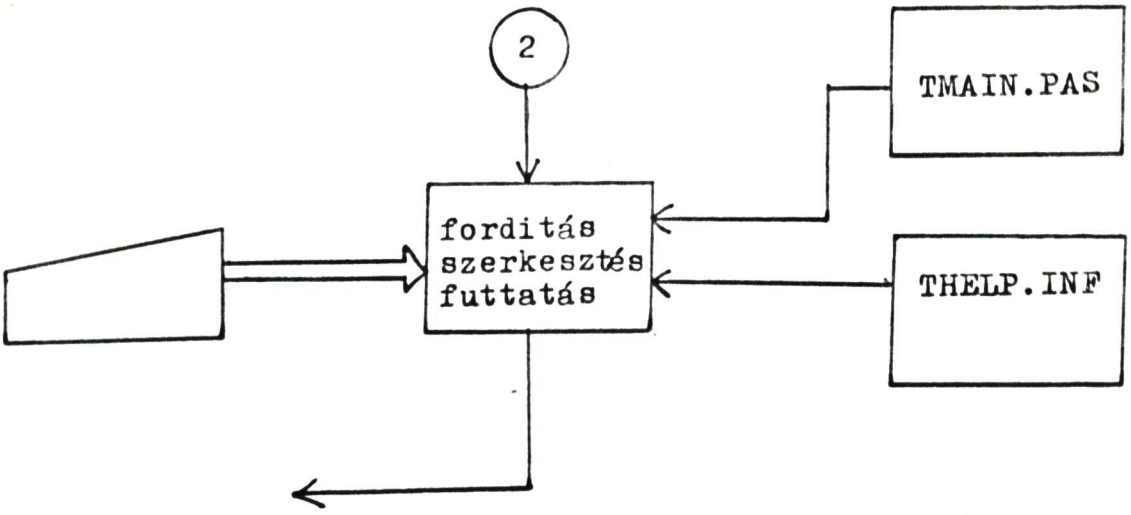
Az alrendszerek összefüggését az 1. ábra mutatja.

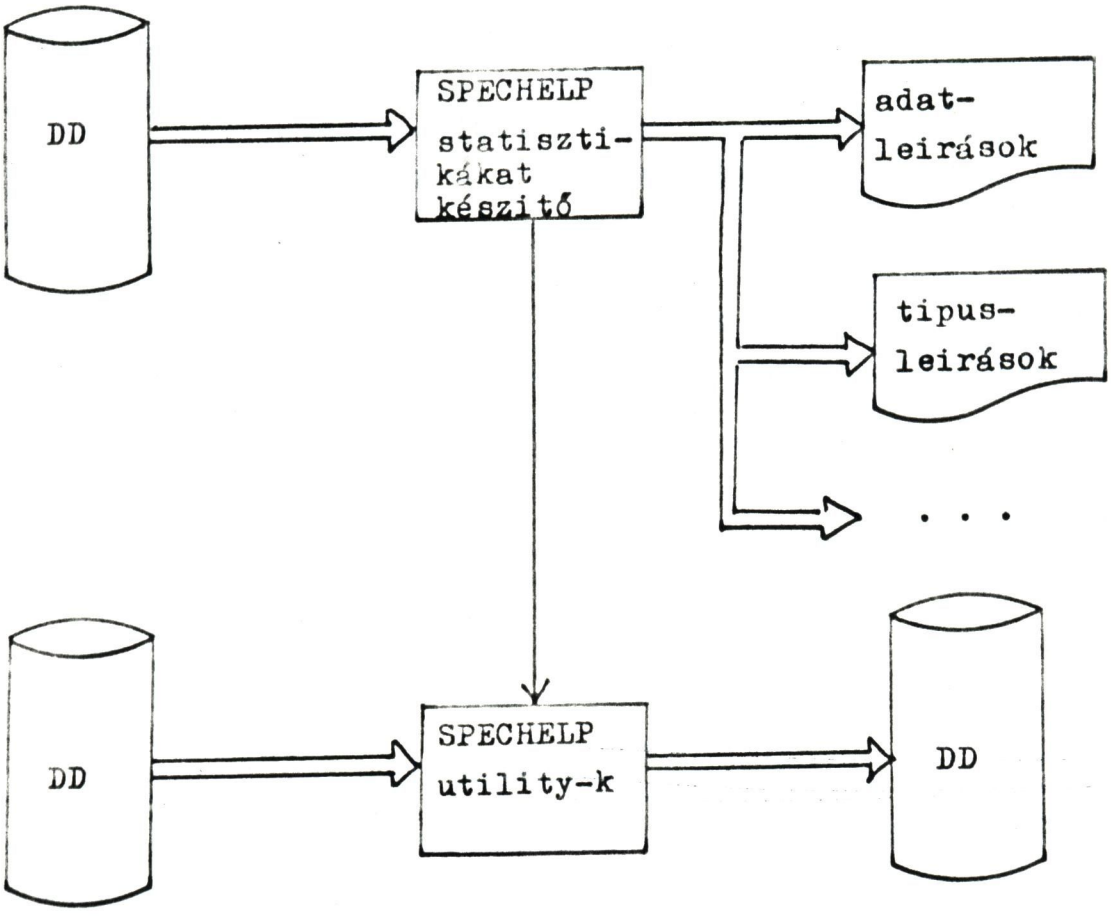
---

* Az 1. verzió PDP-11 tipusu kisszámitógépen, RSX-11M operációs rendszer alatt, a PASCAL-2 forditóprogramot használva müködik.


DESHELP
/EDITOR
modul
más
editorral
létrehozott
modul
DESHELP
/ELOTET
DOKHELP
/PDL
grafikus
ábrázolások
cross
referencia
lista
DESHELP
/DDFILL
Data
Dic-
tion-
ary
1


1
PROGHELP
/KONV
DD
PROGHELP
/GEN
PASCAL
program
(MAIN.PAS)
DOKHELP
/PAS
grafikus
ábrázolások
.LST
cross
referencia
.CRF
modul-
szerkezet
rajza
.MOD
TESZTHELP
Előkészitett
PASCAL
program
TMAIN.PAS
THELP.INF
információs
file
2





Minden egyes alrendszer display orientált, vezetett párbeszédeken keresztül kommunikál a felhasználóval.

2.1. A DESHELP : tervezői alrendszer

2.1.1. AZ EDITOR ALRENDSZER

Az EDITOR program PDL formalizmusu algoritmusleirások megadására alkalmas.

A PDL (Problem Description Language) egy széles körben elterjedt tervezési és dokumentálási eszköz. Kontroll utasitásai kötött szintaxisuak, (while...do - endwhile, if... then - else - endif, ...), azokon belül viszont lényegében teljesen kötetlen leirásokat használhatunk.

Az XHELP-ben használt PDL keret utasitásai:

program - ... - end
procedure - ... - endproc
function - ... - endfunction
if ... then - ... - else - ... - endif
while ... do - ... - endwhile
for ... do - ... - endfor
repeat - ... - until ...
case ... of - ... - endcase
(A case cimkék az or szóval kezdődnek!)
with - ... - endwith
goto ...
break ...
call ...
assert ...
read ...
write ...
... := ... (értékadás)

A call, read, write csupán ajánlás, nem kötelezően használandó kulcsszó.

Az adatokat az XHELP kényszeritő erővel definiáltatja az adatszótárt töltő fázisban. Ehhez viszont ki kell tudni választania a kötetlen formátumu leirásból a változókat, amit

ugy teszünk lehetővé, hogy a PDL leirásban minden változónévnek legalább első előfordulását egy jelzéssel látjuk el. A jelzés az " (idézőjel) karakter. A kereszt-referencia érdekében egyébként célszerü a változónevek jelzése a további előfordulásoknál is. Az idézőjel elmaradhat a zárójelben, paraméterlistaszerüen felsorolt változók esetében. Zárójelnél az XHELP automatikusan paraméterlistát kezd értelmezni. Tehát az alábbi sorok ekvivalensek:

```
beolvasás "n
beolvasás(n)
beolvasás("n)
```

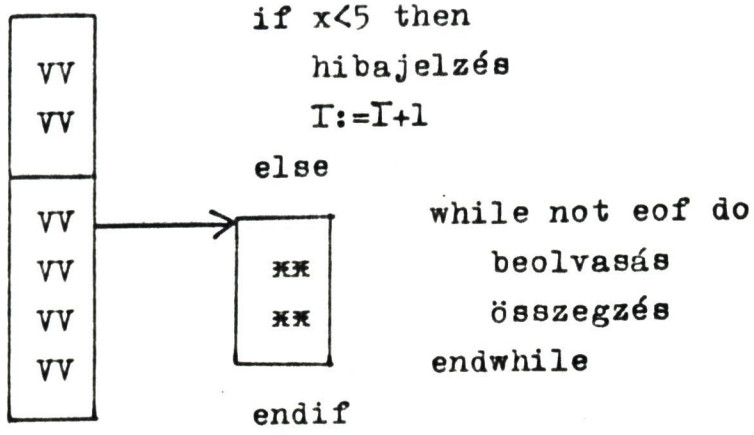
Mindegyik esetben egy beolvasás nevü funkciót és egy n nevü változót azonosit az XHELP.

A PDL sor-orientált, azaz utasitásait a CR karakterrel zárjuk. Folytatósor jelzése a - (minusz) karakter.

PDL nyelven megirt például az alábbi modul:

```
program összeg
   "n beolvasása
   "s:=0
   for "I=1 to n do
      call beolvasás(x(I))
      s:=s+x(I)
   endfor
   if spozitiv(s) then
      write s
   else
   write 'negativ osszeg'
endif
end
```

Az EDITOR egy strukturaeditor. Képszerü ábrázolása egy prettyprint forma, mely jól kiemeli a blokkszerkezetet.



Az EDITOR-ban különleges utasitások állnak rendelkezésre a strukturablokkok kezelésére. Pl. egy utasitással törölhetünk blokkot, blokk-keretet, stb.

Az EDITOR-ral létrehozott modulok a DD-ben megőrzésre kerülnek.

2.1.2. ADATSZÓTÁR TÖLTŐ

A modulban ismeretlen változónevek, meg nem definiált rutinokra való hivatkozások lehetnek. Az Adatszótárt töltő program kényszeritő párbeszédek révén információkat kér a felhasználótól ezekre vonatkozóan, igy a DD-ben mindig logikailag teljesnek tekinthető információhalmaz található.

Egy kényszeritő párbeszéd pl. az "S" adatra vonatkozó alábbi map:

UJ ADAT:
ADATNÉV:
FUNKCIÓ:
TIPUS:
ÉRVÉNYESSÉGI KÖR:

A felhasználó a display-n kiadott keretet kitölti, amennyire tudja s ezentul az S adat a DD-ben ezen információk alapján lesz kezelve.

A PDL formátumu algoritmus-leirásokban a nem standard szintaxist követő karaktersorok funkcionálisan kijelölendő rutinokként értelmeződnek. Igy lesz például az

```
if nem egyenlő then
  átlagszámitás
else
  s:=o
endif
```

Programrészletben a "nemegyenlo", és az "atlagszamitas" egy-egy procedure.

2.2. PROGRAM GENERÁTOR

A DD-ben a fejlesztés minden információja meglévén, lehetőség van arra, hogy a fejlesztés bármely szintjén programot generáltassunk az XHELP rendszerrel.

A program generátor kiválasztja a DD-ből a szükséges modulokat, adatleirásokat, a meg nem definiált modulok helyére üres rutinokat generál, a PDL szintaxist átváltja PASCAL szintaxisra, a még nem kielégitően tisztázott adatdefiniciókat default-okkal helyettesiti.

2.3. A TESZT PROCESSZOR

A dummy rutinokkal teletüzdelt program futása azonban általában semmitmondó vagy éppen értelmetlen, hiszen az üres rutinnal helyettesitett funkcióra a tesztelés során szükség lehet. Például: hiába helyes szintaktikailag egy programunk, ha egy kontrollváltozót állitó rutint üres rutinnal váltjuk ki, a rutinból való visszatérés esetén határozatlan lesz a program. A továbbmenetre két ut kinálkozik: a szimbolikus teszt vagy pedig a rutin funkcionális szimulációjának megengedése. Az XHELP egyenlőre a második lehetőséget támogatja, ugy, hogy minden dummy rutinban megáll, s bekéri a rutintól várt funkciók leirását (azaz az output ill. update tipusu

nem-lokális változók értékét). Lehetőség van ezen a megállási ponton az input adatok ellenőrzésére, változtatására, az adathalmazt "be tudjuk fagyasztani", ill. a megállási pont passzivizálására is módunk van.

A megállási pontok a dummy rutinokba automatikusan belekerülnek, ha a futtatás a teszt processzor beiktatásával történik. Kérhetők a megállási pontok nem üres rutinok elejére és végére is, sőt: a megállási pont mechanizmusával verifikációs feltételeket tudunk kezelni a program bármely pontján.

Az XHELP ugyanis megengedi a programban bárhol egy ASSERT utasitás elhelyezését. A program futás közben az ASSERT-ben megadott feltétel teljesülésétől függően aktivizál egy megállási pontot: a szintén az ASSERT utasitásban megadott változók kiiratásának és módositásának lehetőségét biztositva.

Például:

ASSERT (1<meret) and (100>meret) (meret)

Hatására a "meret" változó értéke minden esetben megvizsgálásra kerül. Ha a feltétel hamis, életbe lép a megállási pont, ahol a méret nevü változót kiirhatjuk, hamis érték esetén uj értékre átirathatjuk.

2.4. A DOKUMENTÁCIÓS ALRENDSZER

A DD lehetővé teszi, hogy a fejlesztés során bármikor előállithassuk az aktuális állapot dokumentációját. Ez részben automatikusan, részben segédeszközökkel támogatott felhasználói közremüködés révén történik. A dokumentáció áll:

- szöveges leirásokból,
- a rendszert globálisan leiró információkból: modulstruktura kimutatások, adattáblák, stb.
- a modulok leirásaiból, általában három szinten: a funkcionális leirás, modultervezet és a programszintü leirások formájában.

Krafft

A dokumentáció felé a leglényegesebb elvárások a teljesség és a világos, érthető, lehetőség szerint képszerü ábrázolás.

Az XHELP számos képszerü ábrázolást tud automatikusan előállitani. Témakör szerint:

- modul ábrázolás,
- modulszerkezet rajza

jöhet szóba és ezekre a következő formáju dokumentációk készülhetnek:

- strukturadiagram,
- LEIGHTON diagram,
- keretdiagram,
- fadiagramok.

2.4.1. Modulok képszerü ábrázolása

A.**** Strukturadiagramos forma

A strukturadiagram lényegében egy továbbfejlesztett pretty-print forma. A strukturablokkok a paragrafusozáson kivül a blokk tipusára jellemző jelekkel ki vannak emelve, a modulok külön fejlécet kapnak és lista formára lesznek tördelve.

A strukturadiagramot szemlélteti a 2. ábra. A szereplő program csak demonstrációs célt szolgál, értelme nincs.

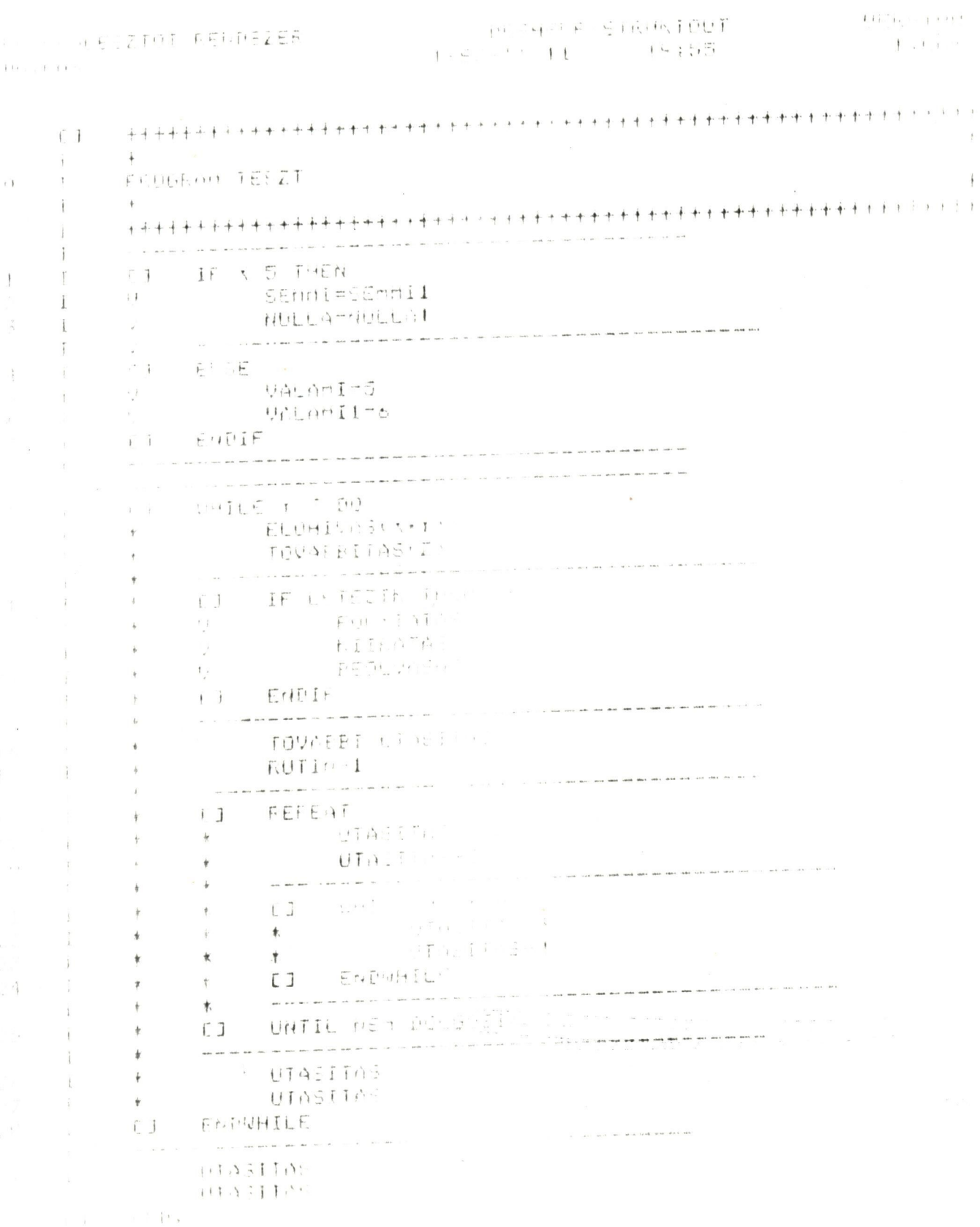
B.**** LEIGHTON diagram

Elvében a strukturadiagrammal azonos, kissé más képszerü formával. A 3. ábra egy demonstrációs LEIGHTON diagramot mutat.

C.**** Keretdiagram

Több esetben (például tesztpont tervezésnél) hasznos lehet egy olyan ábra, mely a mldulnak csupán a kontroll-szerkezetét mutatja, felesleges részletek nélkül. Ez a vázdiagram a LEIGHTON ábrázolási formára épül, de az igazi LEIGHTON ábrának csupán kivonata.

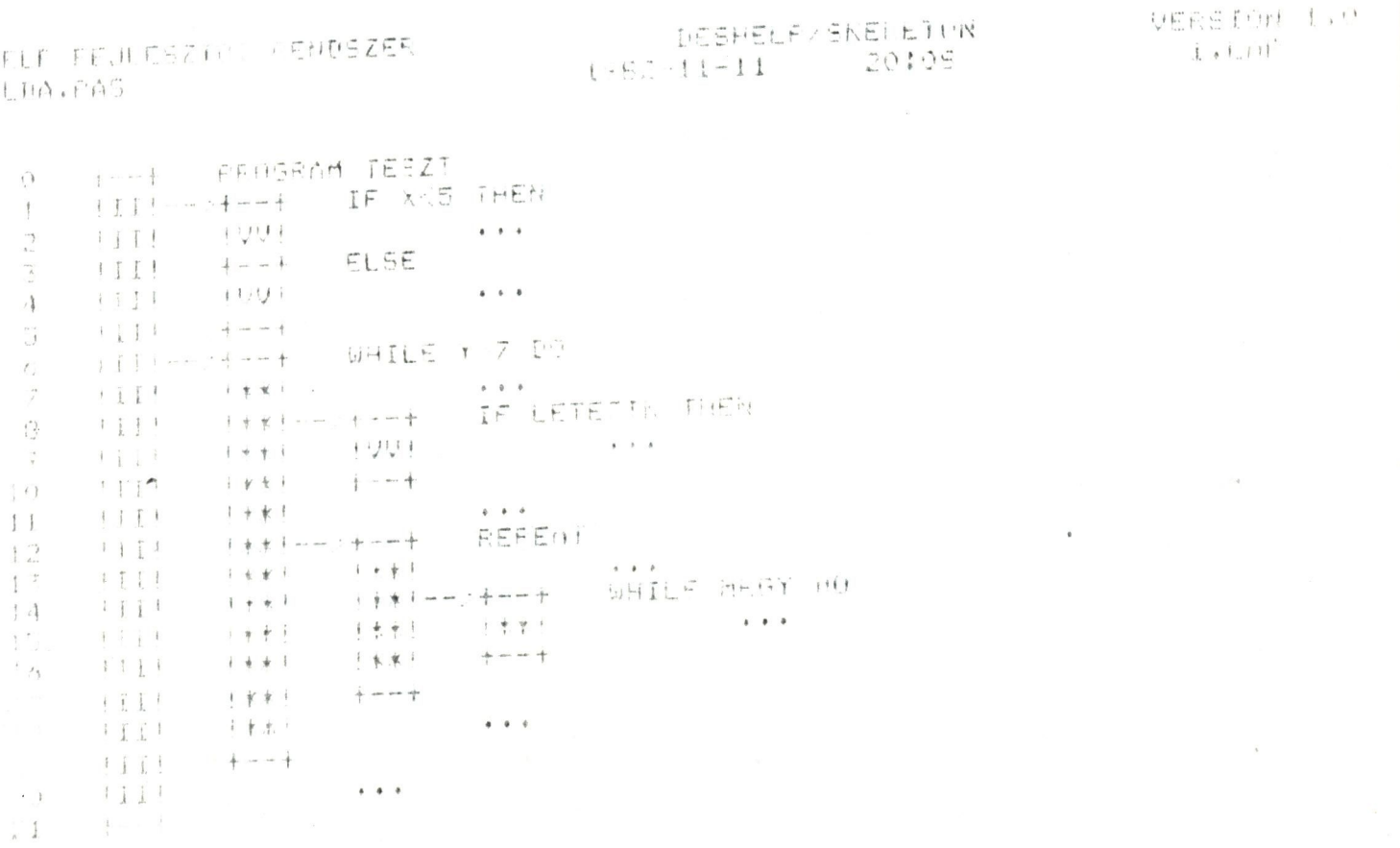
Az 5. ábra az előzőekben használt program vázdiagramját mutatja.



2. ábra

3. ábra


MODULESTRUCTURE OF
15:49:35.1
CODE
NAME
TYPE
NESTING
CALLED
CALLING
MODE
forward


DESHELP/SKELETON
1-SZ-11-11 20:05
LIN.PAS
1.LIN
PROGRAM TESZT
IF x<5 THEN
ELSE
WHILE x > 7 DO
IF LETEZIK THEN
REPEAT
WHILE MEGY HO

5. ábra

2.4.2. A rendszer ábrázolása

Elvben használható itt a modulok ábrázolásánál ismertetett bármelyik módszer, amennyiben a rendszer legfelsőbb szintü moduljait dolgozzuk fel vele. Igy azonban nem kapnánk tiszta képet a teljes modulhalmaz összefüggéseiről, a hivási strukturáról.

A dokumentációs alrendszer külön erre a célra rendelkezik egy olyan kirajzoló résszel, mely a teljes hivási fát kirajzolja. Egy egyszerü mintaprogramot láthatunk az 4. ábrán.

2.5. ADATSZÓTÁR UTILITY-K

Egyfelöl dokumentativ jellegü információkat adnak a teljes Adatszótárról (pl. a modulok funkcionális leirásának listázására képesek), másfelöl pedig az adatszótár kezeléséhez szükséges funkciókat (CREATE,DELETE,stb.) biztositják.

3. PROGRAMFEJLESZTÉS AZ XHELP RENDSZER SEGITSÉGÉVEL

3.1. Az XHELP mint egy PASCAL fejlesztői környezet

Az XHELP-et ebben a felhasználási módban több, egymástól lényegében független hasznos eszköz gyüjteményének tekintjük, s ezen eszközöket használjuk céljainknak megfelelően.

Különösen jól használható önállóan a képszerü ábrázolásokat biztositó alrendszer, az EDITOR és a modulszerkezet-kirajzoló.

3.2. Az XHELP, mint integrált programfejlesztői rendszer

Az XHELP lehetőségeinek, a DD-nek teljes kihasználása csupán akkor érhető el, ha a fejlesztést kezdettől fogva az XHELP rendszer segitségével végezzük. Igy mindig komplett dokumentáció, módszertanilag helyes vágányon tartott, jól szervezett részek állnak rendelkezésünkre. Ennek érdekében célszerü magunkra vállalni az XHELP esetenként tulzottnak ható adminisztrációját.

Bevezetés a DESHELP

alrendszerbe

MKKE
Matematikai és Számitástudományi
Intézet

1982.

1. Bevezetés

1.1. A DESHELP helye az XHELP-ben

Az XHELP interaktiv programok illetve programrendszerek kidolgozására tervezett interaktiv fejlesztői rendszer. A felhasználó az elképzeléseit viszonylag kötetlen formában, képernyőn keresztül betáplálja egy fejlesztői adatszótárba; s onnan automatizáltan programot generálhat ill. készithet elő tesztelésre, továbbá dokumentációkat kérhet le.

A DESHELP a felhasználó elképzeléseinek kialakitását támogató alrendszer. Az "elképzelés" fogalmába beleértjük a követelményrendszert, rendszertervet, programtervet, operátori utasitást stb.-t, azaz minden algoritmikus jellegü információt.

A DESHELP müködése

A DESHELP alrendszer főbb lépései sorrendben a következők:

- strukturadiagramos editálás,
- strukturadiagram kirajzolása,
- kereszt-referencia lista készitése,
- tipusdefiniciók lekérdezése,
- a hivatkozott procedurák lekérdezése,
- a hivatkozott adatok lekérdezése,
- az információk beépitése a Felhasználói Adatszótárba.

A DESHELP/EDITOR a DESHELP alrendszer első, kulcsfontosságu része, mely a strukturadiagramos editálást teszi lehetővé. Az editált szöveg meglehetősen szabad formátumu, az un. PDL /Problem Definition Language/ szabályainak felel meg.

A DESHELP második alrendszere a DDFILL, mely az információk DD-be való beépitését ill. a módositást végzi.

A DESHELP harmadik alrendszere, az ELOTET egy konverziós program. A nem DESHELP/EDITORRAL létrehozott PDL leirásokat hozza a DDFILL által már értelmezhető formára. Szükségességét az a tény indokolja, hogy számos, a felhasználók által megszokott editor program létezik; sok köztük több szolgáltatást nyujt, mint a DESHELP/EDITOR - igy értelmetlenség lenne a PDL modulok létrehozásának lehetőségét kizárólag az XHELP saját editorára korlátozni.

1.2. A PDL formátumu algoritmus-leirás

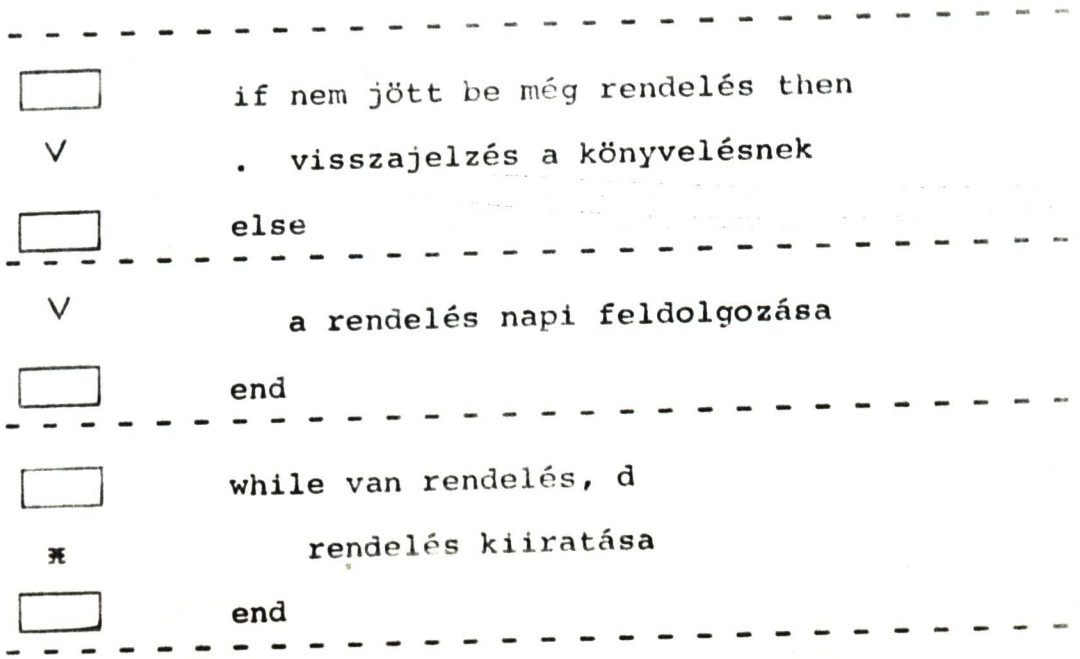
Programnyelvi szintaxisra hasonlitó formáju keretutasitásokon belül tetszés szerinti leirási mód - ez a PDL lényege. Az egyes /elemi/ lépések formai és tartalmi kötetlensége teszi a PDL-t rendkivül széleskörüen használhatóvá.

Példa PDL formalizmusra:

```
if      nem jött be még rendelés then visszajelzés
        a könyvelésnek
else    a rendelés napi feldolgozása
endif
while   van rendelés do
        rendelés kiiratása
endwhile
:
```

1.3. Strukturadiagram

A tervezéskor elképzeléseink ábraszerüen is megjelennek a display-n. A strukturadiagramos grafikus ábrázolási mód az algoritmus logikai blokkjait jeleniti meg, a következő módon:



A vizszintes vonalakkal az egyes blokkokat különitjük el. A paragrafusokkal a logikai mélységet szemléltetjük. A paragrafusban előforduló karakterek a blokk-összetételre utalnak:

V - vagylagos blokk

x - ciklusblokk

----> - megszakitás

<=====> - alprogramhivás

□ - blokk-kezdet vagy vég

I - modul-blokk.

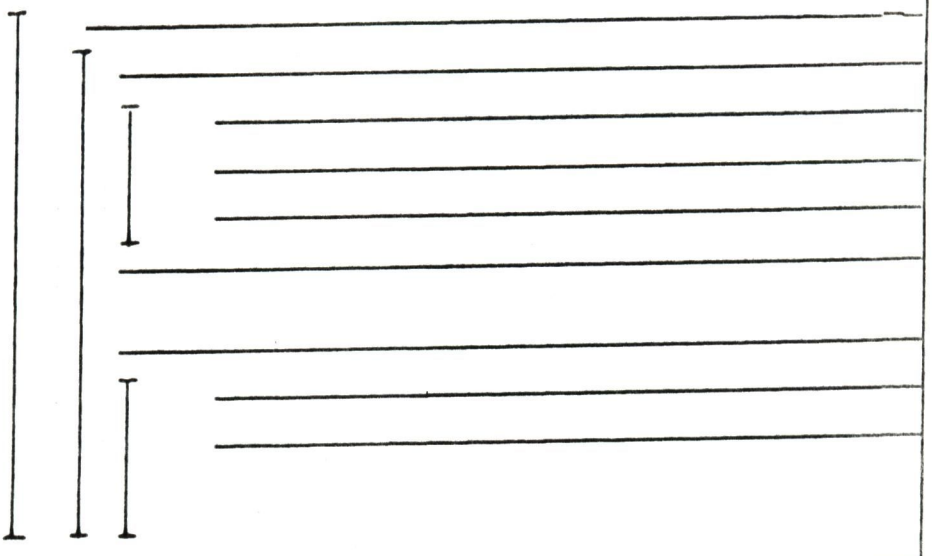
1.4. A strukturadiagramos editálás

Egy egyszerü editor program szolgáltatásait és a strukturadiagram automatikus, editálás közbeni képzését biztositja az XHELP. A felhasználó által beütött,minden egyes szövegsor azonnal strukturadiagrammal együtt jelenik meg a képernyőn. A szöveg módositásakor a diagram automatikusan átszerkesztődik.

A képernyő tehát mindig a következőképpen néz ki:

(A vízszintes vonalak helytakarékosság miatt a képernyőn nem jelennek meg).

EZ EGY PÉLDA:



Az editor nagyjából a PDP 11 editorának szolgáltatásait nyujtja, néhány utasitása:

* next <n>
* bottom
* top
* print <n>
* change /<régistring>/<ujstring>
* locate <string>
*
*
* end

2. A PDL szintaxis

Az alábbiakban röviden összefoglaljuk a PDL kötött szintaxisu keret-utasitásait. Leirásunk csupán informális és igen rövid, tekintettel arra, hogy általában közismert szerkezetekről van szó. A PDL formalizmus bővebb ismertetése az A. függelékben található.

PDL utasitások:

1. procedure
   ...
   end
   - külön logikai egység /modul/
     /a top-down fában egy levél/

2. program
   ...
   end
   - a programrendszer főmoduija
     /a top-down fa csucsmodulja/

3. if...then
   [else...] end
   - kétirányu elágazás

4. while...do
   ...
   end
   - while ciklus

5. repeat
   ...
   until...
   - until ciklus

6. for...do
...
end — számlált ciklus

7. case...of
&...:...
&...:...
...
end — többirányu elágazás

8. go... — kiugrás

9. ...:=... — értékadás

10. xxx:... — cimke. Az xxx alfabetikus karaktereket jelöl

11. return — modul; logikai vég

12. break — ciklus megszakitása

13. call — külön procedura-ként definiált modul hivása

14. read... — beolvasás

15. write... — kiirás

16. map... — előre tervezett teljes display-s párbeszéd

17. assert... — verifikációs pont.

A comment-eket ! vezeti be.

A PDL utasitásokat a sorvégi RETURN zárja; ha nem fér el egy sorban, az uj sor elején - jellel folytatósort jelzünk.

A fentiek alapján nem azonositható sorok a PDL-ben az un. aktiv comment-ek, azaz olyan kötetlen formátumu leirások, melyek a késõbbiekben nyernek majd valamilyen tartalmat. Ezen sorokat az XHELP call nélküli eljáráshivásoknak fogja fel.

A PDL formalizmus részletekbe menő leirását az A. függelék tartalmazza.

ED - 0

DESHELP/EDITOR

Felhasználói kézikönyv

MKKE - MSZI

1982.

1. AZ EDITOR FUNKCIÓJA ÉS HIVÁSA

Az EDITOR program (a továbbiakban röviden: ED) az XHELP rendszerben PDL nyelvü programmodulok létrehozására és javitására szolgál.

Az ED egy interaktiv program. A felhasználó editor parancsokkal kommunikál a programmal. Az editor parancsok a kivánt tevékenységet irják le. Egy parancsnévből és egy azt követő (sok esetben opcionális) paraméterből állnak. A parancsnév egy vagy legfeljebb két karakterre rövidithető. A parancsok elnevezése értelmes: a funkció angol neve.

Például:
a L parancs a "locate" funkciót jelenti.

Az editor parancsokat a 3. fejezetben részletesen ismertetjük.

Az ED egy struktura-editor, azaz a szövegszerkesztés bármely pillanatában ismert az addig megadott algoritmus-leirás blokkstrukturája. Igy szövegszerkesztés közben módunk van:
- a begépelt szöveg strukturadiagramos megjelenitésére a képernyőn,
- strukturautasitások révén egész strukturablokkok ill. blokk-keretek elhagyására.

A strukturadiagramos ábrázolás jelentős vizuális segitséget nyujt a program létrehozásakor.

Az ED az alábbi strukturablokkokat értelmezi:

PROGRAM - END
PROCEDURE - ENDPROC
FUNCTION - ENDFUNCTION
IF - ELSE - ENDIF
CASE - ENDCASE
FOR - ENDFOR
WHILE - ENDWHILE
REPEAT - UNTIL
WITH - ENDWITH

Az ED hivása

Az ED a többi XHELP alrendszerhez hasonlóan a MONITOR-on keresztül hivható. (Ld. a MONITOR felhasználói kézikönyvet!) A MONITOR első inditáskor lekérdezi a fejlesztés nevét és jelszó-ellenőrzést végez; a továbbiakban pedig lekérdezi a kivánt funkciót és inditja a megfelelő programrészt.

A DESHELP alrendszer EDITOR funkcióját választva az ED indul. Bejelentkezése az alábbi panellal történik:

```
XHELP FEJLESZTOI RENDSZER                                   EDITOR

A MODUL: .....
```

Válaszként meg kell adnunk a szerkeszteni kivánt modult; ami általában, PDL-ről lévén szó, a funkció verbális leirását jelenti.

Pl.:

egy rekord beolvasasa (x)

A zárójel ill. sorvége előtti alfabetikus karakterekből a rendszer egy filenevet képez, hozzáadja a .DES tipust, s ezt a nevet használja 1/o filenévként.

A válaszadás során a BACKSPACE karakter az utoljára adott karaktert, a \ (backslash) jel a teljes addigi választ törli. A minusz jellel kiugorhatunk az editorból.

A kezdőpanel után a

```
+RUN EDIT
FILE: .....
```

kiirással uj képernyő kezdődik, s indulhat a későbbiekben ismertetendő editálási folyamat.

2. ALAPFOGALMAK

2.1. PERIFÉRIAIGÉNY

Az ED konzolról beütött parancsokat ill. szöveget értelmez, a párbeszéd konzolról történik. A hibajelzések is a konzolra iródnak vissza.

A szöveg manipulálása a központi memóriában történik. Erre a célra egy max. 100 sort befogadni képes szöveg-buffert használ a program.

Input file létezése esetén az ED induláskor automatikusan betölti az input file-t, egyébként a szerkesztés üres bufferral indul. A bufferban javithatunk, bővithetünk, ameddig a bufferban van hely.

Az ED-ben a lapozási parancsok passzivizálva vannak, mert 100 sornál nagyobb PDL modul hibás programozási stilusra utal, ezért nincs megengedve.

2.2. GÉPELÉSI SZABÁLYOK

Mind a parancsokat, mind a tárolandó szöveget a megszokott sor-orientált begépelési módon adjuk meg. Azaz:

- A sort CR (carridge return) zárja.
- Javitási lehetőség a CR előtt:
  Karaktertörlés: BACKSPACE billentyüvel,
  Teljes sor törlése: a CTRL/U kombinációval, a megszokott módon.

2.3. AZ EDITÁLÁSI MÓDOK

Az editornak két működési módja van: az input mód és az editor mód. Input módban az ED minden konzulon beütött sort a szövegfile-ba tárolandó sornak veszi. Ezzel a móddal kreálhatunk file-t ill. irhatunk uj sorokat már létező file-ba. Editor módban az ED a megadott sorokat editor parancsként ér-

telmezi, és a parancs által előirt módon kezeli ill. módositja a szöveget. Az editor módot az ED egy * karakter kiirásával jelzi.

2.3.1. Input mód -

Uj file létrehozásakor

Ha a felhasználó input file-ként nem létező file-t ad meg, akkor az ED automatikusan uj file létrehozását tételezi fel és input módba megy. Közben egy üzenettel jelzi, hogy az input file nem létezik.

Például:

```
+RUN EDIT
FILE:UJFILE;DES
CREATING NEW FILE
INPUT
```

A felhasználó ezután elkezdheti begépelni a szöveget. Minden beütött karakter bekerül a szöveg file-ba.

Üres sor begépelését ugy végezzük, hogy egy sorköz karaktert ütünk be és CR-rel lezárjuk.

Az INSERT parancs

Edit módból az I (INSERT) paranccsal válthatunk át az input módra.

Kilépés az input módból

A szöveg begépelését abbahagyni s az editor módra rátérni egy csak CR-t tartalmazó, valóban üres sor megadásával lehet, a megjelenő * jel jelzi, hogy a gép editor parancsot vár.

2.3.2. Editor mód -

A sor elején megjelenő * karakter jelzi, hogy az ED editor módban van. Csak editor parancsot hajlandó elfogadni.

Létező file-ok editálása

Ha az ED inditásakor létező file-t adtunk meg, akkor ugyanezen a név alatt, eggyel növelt verziószámmal létesitődik egy output file, majd az input file-ról betöltődik az első lap a memóriába s a gép editor módba megy.

Példa:

```
+RUN EDIT
FILE:REGI;DES
     56 LINES READ IN
*
```

Sormutató

A szövegkezelés sor orientáltan történik. Ez azt jelenti, hogy a szöveget sorok egymásutánjának kezeli a ED, s az éppen feldolgozás alatt levő sort egy sormutatóval azonositja.

Az edit mód első bejelentkezésekor a sormutató az első sor előttre mutat. A felhasználó különböző editor parancsokkal az általa kivánt helyre mozgathatja a sormutatót.

A sormutató állitása történhet sorszámra történő utalás vagy a keresendő szöveg megadása révén.

Például:

```
+RUN EDIT
FILE?UJFILE.PA
CREATING NEW FILE
INPUT
EZ AZ ELSO SOR
EZ A MASODIK
HARMADIK
NEGYEDIK ÉS EGYBEN AZ UTOLSO
(CR)
*T
*L MASODIK
EZ A MASODIK
```

```
*N
HARMADIK
*/
EZ A MASODIK
*L ES
NEGYEDIK ES EGYBEN AZ UTOLSO
*END
      EXIT
 ....
```

A példában először létrehoztunk egy file-t, ami négy sorból áll. Az üres sorral kiléptünk az input módból, a sormutató az utoljára beütött szövegsornál marad, a TOP paranccsal a szöveg első sora elé pozicionálunk, majd az L(=LOCATE) parancs segitségével az első, MASODIK karektersorozatot tartalmazó sorra állitjuk a sormutatót. Az N(=NEXT) parancs a következő sorra viszi a sormutatót, a / parancs egy sorral visszalép. A LOCATE ES az első ES-t keresi meg, igy a sormutató a negyedik sorra mutat. Az END paranccsal lezárjuk a file-t, kilépünk az editorból.

3. AZ EDITOR PARANCSOK áttekintése

Az ED az alábbi parancsokat értelmezi:

APPEND - hozzáfüzés sorhoz
BOTTOM - pozicionálás az utolsó sorra
CHANGE - karakterstring változtatás a soron belül
DELETE - sorok elhagyása
DF - strukturakeret elhagyása (delete frame)
DS - strukturablokk elhagyása (delete structure)
END - kilépés az editorból
INSERT - uj sorok begépelése
KILL - editor abortálása
LOCATE - string keresés bufferen belül
NEXT - sorpozicionálás relativ sorszám alapján
PRINT - sorok kiiratása
SI - strukturadiagramos mód bekapcsolása
SO - strukturadiagramos mód kikapcsolása
TOP - sorpozicionálás a buffer elejére
(CR) - sorlépés előre
(/) - sorlépés hátra
(.) - abszolut sorszám kiirása
(n) - sor kiirása abszolut sorszám szerint.

A B. függelékben minden parancsnak megadjuk a formátumát, a funkcióját, kiemelt jelentősége miatt a sormutatóra gyakorolt hatását és egy példát.

DESHELP / DDFILL

Felhasználói kézikönyv

MKKE - MSZI

1982.

T a r t a l o m

1. Bevezetés

A DDFILL alrendszer az XHELP Fejlesztői Adatszótárát (DD) tölti információkkal a fejlesztésben használt adatokról, modulokról és tipusokról, ill. biztositja ezen információk módositásának lehetőségét.

Az XHELP rendszerbe illeszkedés tisztázása előtt érdemes felidézni néhány, az XHELP alapjául szolgáló módszertani alapelvet.

- Az XHELP segitségével fejlesztett rendszerek funkcionális modulokból épülnek ki. A fejlesztés során egyszerre mindig csupán egy kiválasztott modullal foglalkozunk. Például: az EDITOR alrendszerrel egyetlen procedurát helyezhetünk el egy file-ban. Az XHELP ezen alapelvének megfelelően a DDFILL szintén egy modult tekint feldolgozási egységnek. Különböző modulok a DDFILL ismételt hivásával dolgozhatók fel.
- A fejlesztői rendszer egyik célja, hogy a felhasználót a tervezési tevékenységben segitse. Ennek eszköze a felhasználó "kikérdezése", melynek során a rendszer a Fejlesztői Adatszótárra támaszkodik. A párbeszédtől elvárjuk, hogy világos, érthető kérdései legyenek, könnyen kezelhető legyen s meg kell teremteni a válaszokhoz esetleg szükséges információk lekérdezésének lehetőségét.
- A Fejlesztői Adatszótárba információkat betölteni ill. adatait módositani az integritás védelmének érdekében csak különleges esetekben lehet közvetlenül. Egyébként a DDFILL az egyetlen olyan eszköz, mellyel a felhasználó a DD-ben módosithat. Ebből viszont az következik, hogy a módositási igényeket is a DDFILL-nek kell kiszolgálnia.

Az XHELP alrendszerek logikai láncában a DDFILL az EDITOR után helyezkedik el. Inditásakor feltételezhető, hogy a felhasználó az XHELP/EDITOR-ral vagy más, szokványos PDP EDITOR programmal létrehozott egy PDL nyelvü modulleirást. (XHELP-en kivüli EDITOR-t használva a szövegfile-t egy ELOTET elnevezésü programon keresztülfuttatva kapható XHELP által feldolgozható

formátum.) Ez a PDL modul egy .DES tipusu file-ban a felhasználó saját könyvtárában vagy a Fejlesztői Adatszótárak könyvtárában helyezkedik el. A DDFILL hivásával az a cél, hogy a .DES modulról, a hivatkozott adatokról, procedurákról a DD-ben információkat helyezzen el, illetve már esetleg meglevő információkat módositson.

A DDFILL befejeződésekor a DD-be befüzve ott szerepel a feldolgozott modul, s annak minden adatáról, a használt tipusokról, ill. a hivatkozott eljárásokról létezik bejegyzés. Lényeges látni, hogy a bejegyzések nem szükségszerüen teljesen kitöltöttek. Nem biztos, hogy a fejlesztés során minden adatot, minden tipust stb. azonnal végleges formában meg tud a felhasználó adni. Ez a top down fejlesztés nyilvánvaló velejárója. Ilyenkor nincs hibajelzés, nem létezik semmi, a feldolgozást blokkoló szankció. A rendszer tudomásul veszi, hogy az információ majd valamikor pontositásra kerül, addig csak a csonka tartalomra támaszkodhat.

2. Az algoritmus

A DDFILL feldolgozás technikai okokból két fázisu. A DDFILL1 feladata a .DES file által hivatkozott adat- és proceduranevek kigyüjtése. Egy opcionális Cross Referencia Lista kérhető.

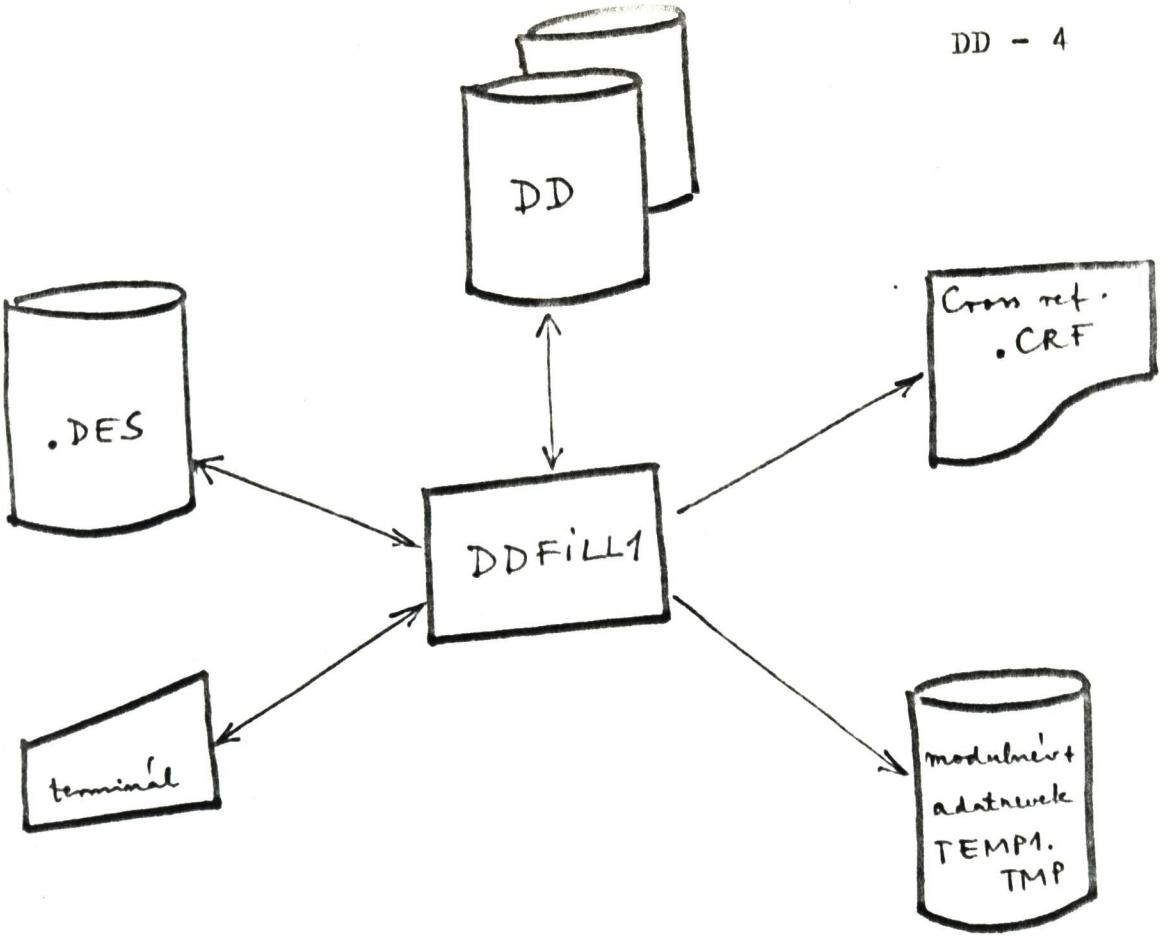
A DDFILL2 az előző fázisban kigyüjtött adatneveket sorra veszi s párbeszédeivel kikényszeriti azok valamilyen szintü definicióját.

A DDFILL1 környezetét az 1. ábra mutatja.

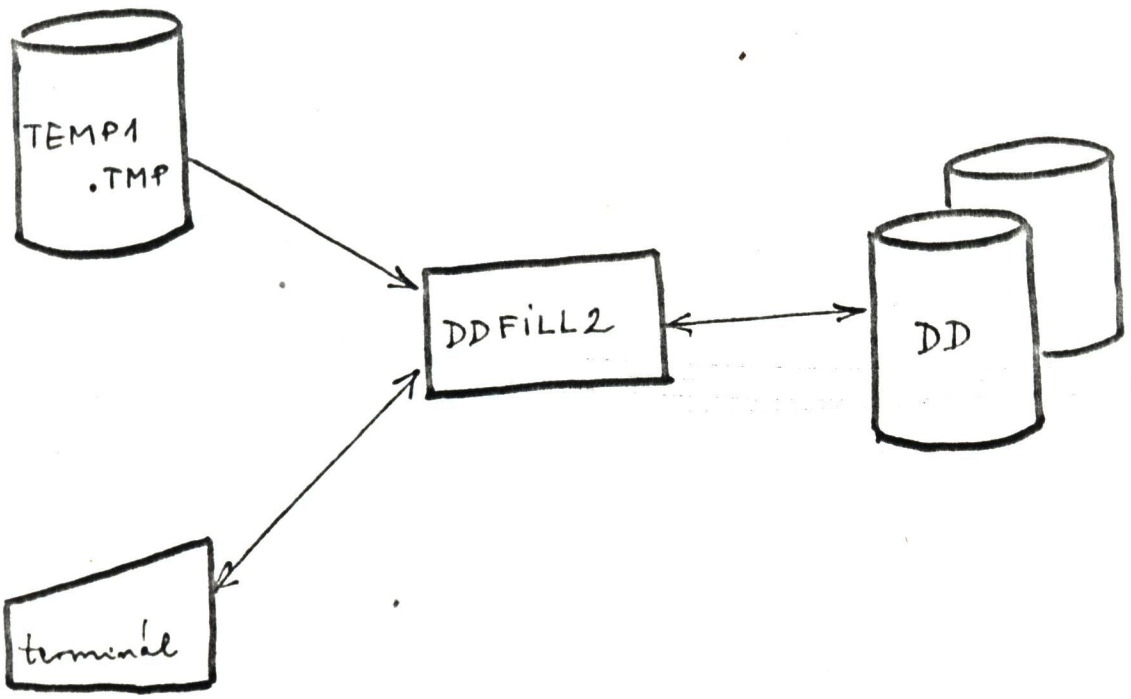
A .DES file a feldolgozandó PDL modult tartalmazza. Ha a felhasználó könyvtárában nincs, akkor a DDFILL1 a DD könyvtárhoz fordul s onnan átmásolja.

A DD-vel való kapcsolat:

- A program megvizsgálja, hogy a modul létezik-e. Ha nem, akkor kreál egy entry-t!
  Ha létezik, akkor kiolvassa a modulhoz füzött adatneveket és azok érvényességi körét.
  A .DES modulban szereplő adatnevek és a DD-ből kiolvasott adatnevek egyesitése kerül majd feldolgozásra a DDFILL2-ben.


DD
.DES
Cross ref.
.CRF
DDFILL1
terminál
modulnévtér
adatnevek
TEMP1.
TMP

1. ábra


TEMP1
.TMP
DDFILL2
DD
terminál

2. ábra

- Lekérdeződik a modul informális leirása (= funkciója) s bekerül a DD-be.
- Aktualizálódik a modul hivási lánca. Az esetleg létező régi lánc megszünik és kialakitásra kerül az uj lánc. A nem létező hivott modulok dummy-ként bejegyzésre kerülnek.
- A "dummy" kulcsszót tartalmazó modul üres modulként kerül bejegyzésre, de a DDFILL2 végrehajtódik. Igy lehet a "stub" procedurák I/O környezetét definiálni!

A DD file-ok a DDFILL1 indulásakor bemásolódnak a felhasználó saját könyvtárába. A feldolgozás ezen a privát kópián történik, s csak a hibamentes feldolgozás végén kerül vissza a DD file-ok uj verziója a DD gyüjtő-könyvtárba.

A .CRF cross referencia lista opcionális. Megadja, hogy az egyes változókra és modulokra melyik sorokban történt hivatkozás.

A változókat a PDL szabályainak megfelelően az őket bevezető idézőjelről ismeri fel, illetve a zárójelen belül paraméterlistát értelmez. Ide már nem kell az idézőjel, minden alfabetikus jellel kezdődő, alfanumerikus string adatnévnek minősül.

A proceduranév a sor végéig, a nyitó zárójelig, vagy a sorközi comment-et bevezető felkiáltójelig az összes alfabetikus karakterből képződik.

A változó- és proceduranevek maximális hossza 30 karakter.

A TEMP1 . TMP ideiglenes file tartalma:

- modulnév
- változónév és mutató a létezési módra
  (a mutató lehetséges értékei:

  vglob - régebben is és most is szereplő adatnév
  vpar -
  vlok -

  ujadat - uj adatnév

  voltglob régebben létező, de most nem hivatkozott adatnév.)
  voltpar

... a modul minden változójára.

A változónevek létezési módra és azon belül alfabetikusan csökkenően rendezettek.

A DDFILL2 környezetét a 2. ábra mutatja. A TEMP1 . TMP file-on alapuló sorrendben minden adatnév és tipus definiálható ill. módositható.

Ha a DDFILL2 normálisan fejeződött be, akkor a módositott DD file-ok visszakerülnek a DD gyüjtő könyvtárba.

3. Kezelés

3.1. Konvenciók:

A DDFILL egy párbeszédekkel vezetett rendszer. A képernyők egymásutánját és magukat a paneleket az alábbiakban ismertetjük. A képernyő-kezelés általános konvenciói:

- A cursor minden esetben annál a kérdésnél áll, amelyre válaszolni kell.
- Minden input-ot CR zár.
- Valaminek a jóváhagyása mindig egy üres CR.
- Hibajelzés esetén CR-rel lehet előhivni a következő képernyőt.
- Tévesztéskor javitani lehet: RUBOUT-tal karaktert, backslash jellel teljes sort, minusz jellel teljes képernyőt törölhetünk.
- Minden panel az

  XHELP FEJLESZTOI RENDSZER ... DESHELP/DDFILL

  standard fejléccel jelenik meg.

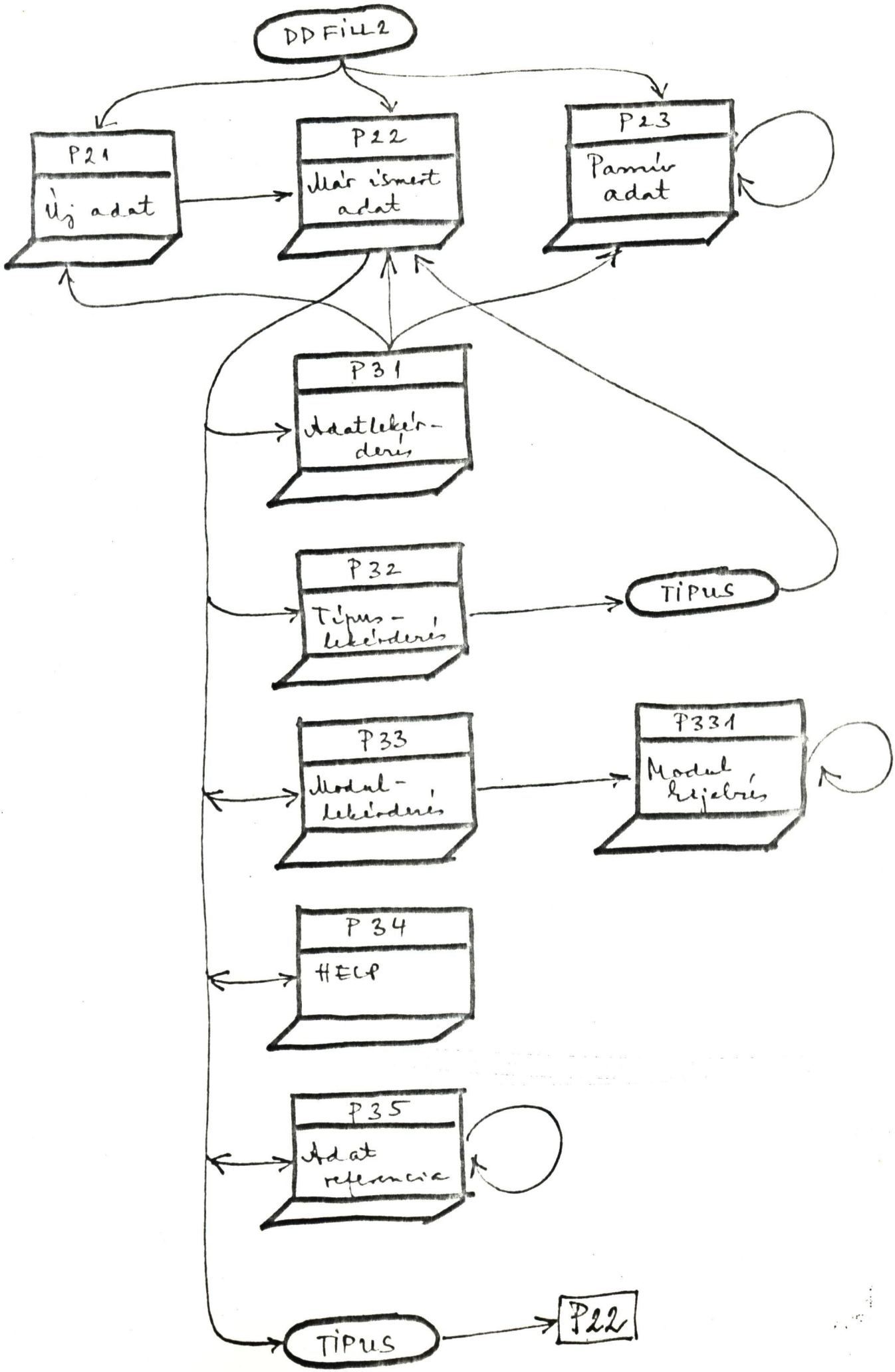
3.2. A panel-háló

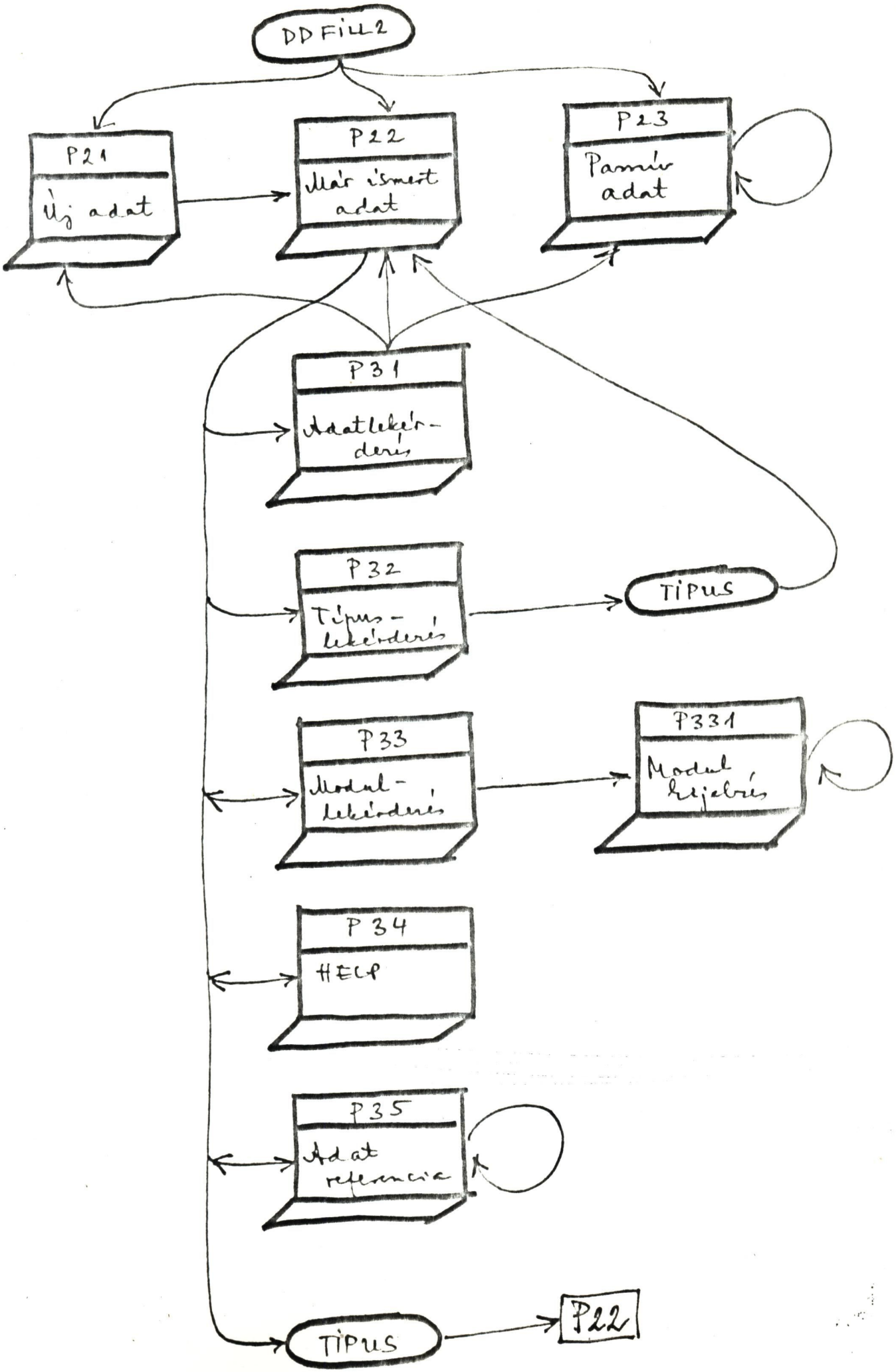
A DDFILL az alábbi panelekkel dolgozik:

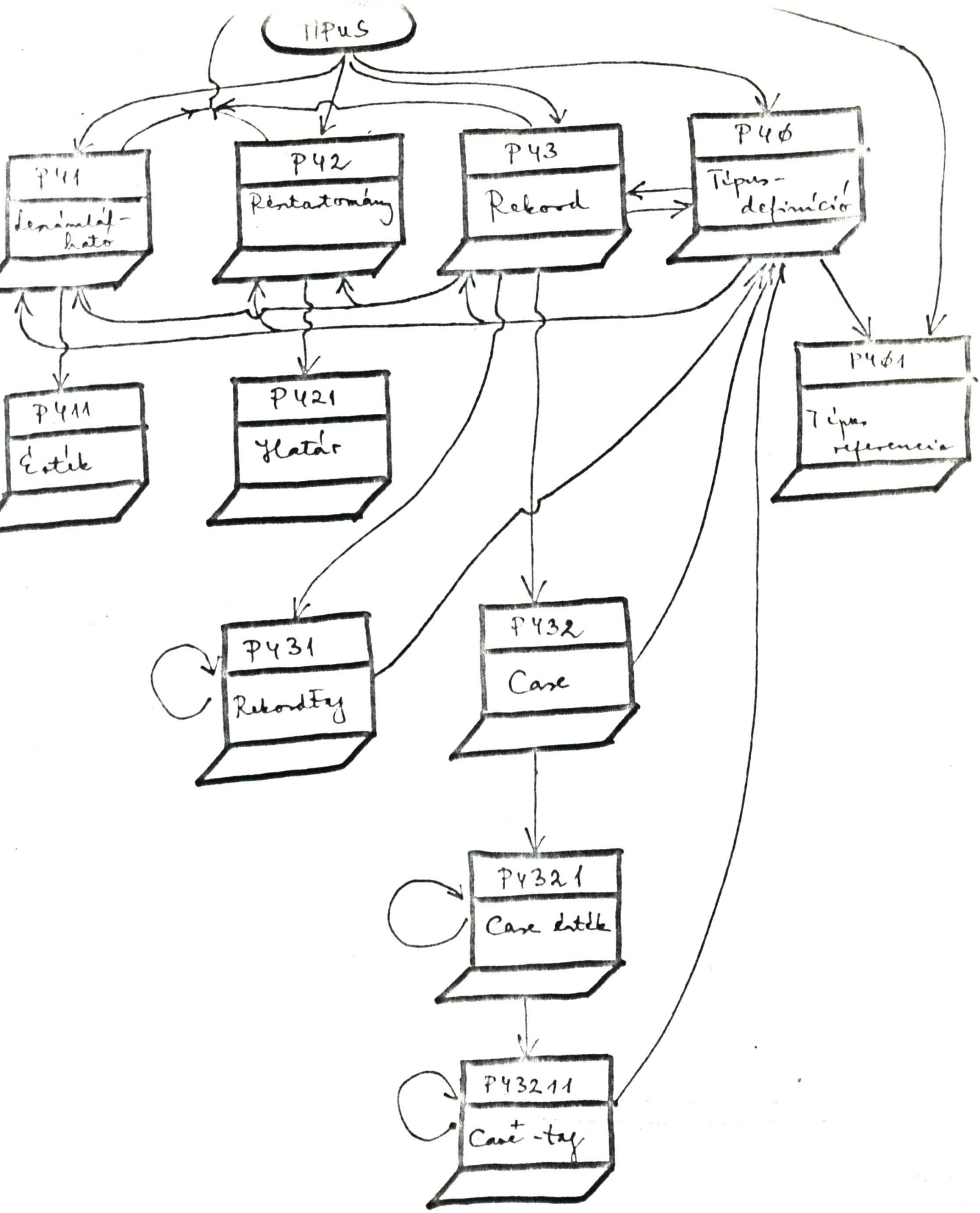
P1 : DDFILL1 kezdő panel,
P11 : DDFILL1 futás közbeni panel,
P12 : Modul definiciós panel,

P13 : DDFILL1 végét jelző panel,
P21 : Uj adatot kezelő panel,
P22 : Már ismert adatot kezelő panel,
P23 : Használaton kivüli adatot kezelő panel,
P31 : Adatlekérdező panel,
P32 : Tipuslekérdező panel,
P33 : Modul-lekérdező panel,
P331 : Modul információs panel,
P34 : HELP panel,
P35 : Adat referenciákat kijelző panel,
P4Ø : Tipusdefinició panel,
P4Ø1 : Tipus referencia panel,
P41 : A leszámlálható tipust kezelő panel,
P411 : A leszámlálható tipus egyes értékeit kezelő panel,
P42 : Résztartomány tipust kezelő panel,
P421 : Résztartomány határait kezelő panel,
P43 : Rekord definicios panel,
P431 : Egy rekordtag leirását kezelő panel,
P432 : Case definiciós panel,
P4321 : Case értékeket kezelő panel,
P43211: Case ágban levő tagokat kezelő panel.

A panelek hálóját a 3/1, 3/2, 3/3. ábrák szemléltetik.


DDFILL2
P21
Új adat
P22
Már ismert adat
P23
Paraméradat
P31
Adatlekérdezés
P32
Típus-lekérdezés
TIPUS
P33
Modul-lekérdezés
P331
Modul kijelzés
P34
HELP
P35
Adat referencia
TIPUS
P22


DDFILL2
P21
Új adat
P22
Már ismert adat
P23
Pannív adat
P31
Adatlekér-dezés
P32
Típus-lekérdezés
TÍPUS
P33
Modul-lekérdezés
P331
Modul kijelzés
P34
HELP
P35
Adat referencia
TÍPUS
P22


TÍPUS
P41
Sorrendi-
hato
P42
Résztartomány
P43
Rekord
P46
Típus-
definíció
P411
Érték
P421
Határ
P461
Típus
referencia
P431
Rekordfej
P432
Case
P4321
Case érték
P43211
Case-tag

3/3. ábra.

3.3. DDFILL1 panelek

P1 panel

DDFILL1 kezdő panel. Beolvassa a feldolgozandó modul nevét s a CREF listára vonatkozó opciókat.

Formátum:

Standard fejléc
A forrásprogram neve: .....
Kiván CREF listát?: N
Lapméret?: 60

Válaszok:

a.) Modulnév: a feldolgozandó modul neve. A tipus kötelezően .DES.

b.) Kér CREF listát: Igen vagy Nem.
A válasz első karaktere számit.

c.) Lapszélesség: A CREF lista egy lapra kerülő sorainak száma.

Hibaüzenetek:

<< nincs ilyen modul

<< nem értelmezett válasz

<< a lapszélesség nem numerikus

P11 panel

DDFILL1 futás közbeni kijelző panel. Passziv, állapotjelző szerepü.

Formátum:

```
Standard fejléc

A feldolgozás folyik
A forrásprogram : xxxxxx
{CREF lista készül
{CREF lista nem készül
!!!!  Várj  !!!!
```

P12 panel

A modul informális leirását kéri. Csak uj modul esetén! Már meglevő modul informális leirása a P33 panelen lehetséges.

Formátum:

```
Standard fejléc

A modul neve: xxxxx
Funkciója: ....
```

Válaszok:

Megadhatjuk a modul informális leirását, max. 70 karakterben.

P13 panel

A DDFILL1 végét jelzi. Passziv panel.

Formátum:

	Standard fejléc
	DDFILL END A forrásprogram: xxxxxx
A.	A CREF lista található: xxxxxx

Az A. sor csak akkor iródik ki, ha kértünk CREF listát.

3.4. Adatpanelek

P21 panel

A DD-ben még nem szereplő változókat kérdezi le.

Formátum:

	Standard fejléc
	Uj adat
	Adatnév: xxxxxx
A.	Funkció: .....
B.	Tipus: undefined
C.	Érv. kör: L
D.	Használat: U

A válaszok:

A. Funkció: az adat jelentésének informális leirása, max. 70 karakterben.

B. Tipusa: ld. a P22 panelnél!

C. Érvényességi köre:

G - globális,
L - lokális,
P - paraméter.

A globalitás csak főprogram szintü lehet. (Ez XHELP megkötés!) A paraméterek sorrendje az előfordulási sorrend.

D. Használata:

I - csak inputra használjuk,
O - csak output,
U - update, azaz I/O változó.

Csak globális vagy paraméter adatoknál iródik ki.

P22 panel

Már ismert adatot jelez ki, ill. lehetőséget ad a módosításra.

Formátum:

	Standard fejléc
	Adatnév: xxxxxx
A.	Funkció:
B.	Tipus:
C.	Érv. kör:
D.	Használat:
E.	OK/show adat (A)/show tipus (T)/show modul (M)/help (H)/ref (R)/-

Válaszok:

A : Funkció: az adat jelentésének rövid leirása.
Max. 70 kar.

B : Tipus:
rekordleirás, leszámlálható és részhalmaz leirás kivételével tetszőleges, 65 karakterben elférő tipusleirás.
A rekordleirást a RECORD szó jelzi, a leszámlálható tipust egy nyitó zárójel, a részhalmaz tipust pedig két pont. Ezek a nem kifejthető tipusok a SHOW TIPUS paranccsal kérhetők le részletesen.
A tipusmegadásban használt ismeretlen tipusokra a program azonnal rákérdez (a P4Ø panel indul).
A szándékosan definiálatlan tipusok jelzésére az undefined tipust használjuk. Ezt a DDFILL alaptipusként értelmezi.

C : Érvényességi köre:

G - globális,
L - lokális,
P - paraméter.

D : Használat:

I - input érték,
O - output érték,
U - update, azaz I/O érték.

A "használat" kérdés csupán használt globális és paraméter változók esetén jelenik meg, lokálisnál nincs értelme.

E : Egy menü, ahol is választani lehet a továbbmenet alábbi lehetőségei között:

a.) OK: egy üres CR. Jelzi, hogy elfogadjuk a kijelzett adatot, változtatni nem akarunk, kiváncsiak sem vagyunk semmire, jöhet a következő adat.

b.) Show adat: A.
Egy adatról kérünk információkat, esetleg módositani is fogjuk. Indul a P31 panel.

c.) Show tipus: T
Tipust kérdezünk le, esetleg módositani is fogjuk. Indul a P32 panel.

d.) Show modul: M
Egy modulról kérünk információkat, esetleg a funkcióját módositani fogjuk. Indul a P33 panel.

e.) HELP : H
Utbaigazitó szöveg iródik ki. Az 1. verzióban csak bejelentkezés. (P34 panel)

f.) Ref : R
Kérjük kijelezni, hogy a feldolgozás alatt levő globális adat milyen modulokban szerepelt eddig. Indul a P35 panel.

g.) Nem OK : -

Nem értünk egyet a kijelzett adatleirással, változtatni kivánjuk. A cursor az A kérdésre áll, lehet módositani. A régi információk törlődnek. Ha egy lokális változót globálisra vagy paraméterre cseréltünk, akkor megjelenik a "használat" kérdés is.

A menü-ben hivott összes panelről előbb-utóbb visszatér a feldolgozás a jelenlegi panelre. Addig a jelenleg feldolgozás alatt levő változó felfüggesztett, nem hozzáférhető állapotban van.

P23 panel

Azokat a globális változókat, melyek régebben szerepeltek a modulban, de az ujabb, a feldolgozás alatt levő .DES már nem hivatkozza őket, ez a panel jelzi ki. Ha valóban feleslegesek, akkor kitörölhetjük őket.

Formátum:

Standard fejléc
Adatnév: xxxxxx Funkció: ..... Tipus: Érv. kör: Használat:

A. OK / -:

Válaszok:

A. : Ha üres CR-rel OK-t adunk, akkor az adat továbbra is fel lesz füzve a modul adatai közé, minusz jellel törölhetjük. Maga az adat-entry nem törlődik, csupán hivatkozási láncából tünik el a modul. Hivatkozás nélküli globális változók viszont fizikailag is törlődnek.

P31 panel

Soronkivüli adat feldolgozását inditja.

Formátum:

	Standard fejléc
	a P22 panel, a menü hiján változatlanul.
	Show adat
A.	Adatnév:

Válasz:

A. : Adatnév: a lekérdezni (vagy módositani) kivánt adat neve. Csak a modulhoz kötődő vagy globális adat lehet.

Az adat milyenségétől függően a P21, P22 panelek egyike indul.

P32 panel

Tipus feldolgozását inditja.

Formátum:

	Standard fejléc
	a P22 panel, a menü hiján változatlanul.
	Show tipus:
A.	Tipusnév:

Válasz:

A. Tipusnév: a lekérdezni (vagy módositani) kivánt tipus neve.
A P4Ø, P41, P42, P43 panelek egyike indul.
Ha üres CR-t adunk válaszként, akkor a rendszer az adathoz kötődő tipus kezelését kezdi a P4Ø panellel. Ilyenkor nincs tipusnév, helyette az adatnév szerepel.

P33 panel

Modulok lekérdezését inditja.

Formátum:

	Standard fejléc
	A P22 panel, a menü hiján változatlanul.
	Show modul
A.	A modul neve:

Válasz:

A. A modulnév: annak a modulnak a neve, melynek a funkcionális leirására, azaz funkciójára, 1/O környezetére kíváncsiak vagyunk.
Indul a P331 panel.

P331 panel

Egy modul funkcionális leirását kijelző panel.

Formátum:

Standard fejléc
Modulnév: xxxxxx Funkciója: xxxxxx 1/O környezete: Változónév tipus: A. OK / -:

Válaszok:

A: A menü.

a.) Ha üres CR-rel OK.-t jelzünk, akkor a modul kijelzése megszakad, visszatérünk a hivó panelhoz.

b.) Minusz jelet ütve az 1/O környezet kijelzése folytatódik. Ha már nincs több környezeti változó, akkor visszatér a hivó panelhoz.

P34 panel

Az adat definiálásához ad hasznos utmutatást. Az 1. verzióban azonban egyenlőre üres.

Formátum:

Standard fejléc
HELP panel

CR-t ütve a feldolgozás a hivó panelba tér vissza.

P35 panel

Kijelzi, hogy globális adatunk mely modulban szerepel.

Formátum:

	Standard fejléc
	Adatnév: Használók: Proc: . . . .
A.	OK/Következő(K):

Válasz:

A. OK esetén a feldolgozás a hivó panelba tér vissza.
K esetén a modulok kijelzése folytatódik. Ha nincs több modul, akkor visszatér a hivó panelba.

3.5. Tipuspanelek

P4Ø panel

Tipus definicióját kijelző ill. definiáló panel.

Formátum:

	Standard fejléc
	Tipusnév: xxxxxx
A.	Funkció:
B.	Tipus:
C.	OK/REF (R)/ -:

Válaszok:

A: Funkció: a tipus informális leirása, max. 70 karakterben.

B: Tipus:

Rekordleirás, leszámlálható és részhalmaz leirás kivételével tetszőleges, 65 karakterben elférő tipusleirás.

A rekordleirást a RECORD szó jelzi, a leszámlálható tipust egy nyitó zárójel, a részhalmaz tipust pedig két pont. Ezek az egyből nem kifejthető tipusok részletes leirása a P41, P42, P43 paneleken keresztül kezelhető.

A tipusmegadásban használt ismeretlen tipusokra a program azonnal rákérdez (ujra indul a P4Ø panel, most már az uj tipusra). A félbehagyott tipusdefinició csak időlegesen függesztődik fel.

C: Menü:

a.) OK : üres CR. Nem kivánunk változtatni, jóváhagyjuk a képernyőt. Nem kifejthető tipusnál a P41, P42, P43 panelek közül a megfelelő indul, egyébként visszatérés a hivó panelbe.

b.) Ref : R
Le kivánjuk kérdezni, hogy a tipust mely változók és mely más tipusok használják. Ez hasznos információ tipusmódositás előtt. Indul a P4Ø1 panel.

c.) nem OK: -
Nem értünk egyet a tipus képernyőn kijelzett leirásával, változtatni akarunk.
A cursor a funkcióra áll, s lehet módositani.

P4Ø1 panel

A tipus hivatkozási helyeit mutatja.

Formátum:

	Standard fejléc
	Tipusnév: xxxxxx Használók Adat: xxxxxx Tipus
A.	OK/következő (K):

Válasz:

A. OK esetén visszaugrás az előző panelra.
K esetén további használó előhivása.

P41 panel

Leszámlálható tipus definiálására

Formátuma:

	Standard fejléc
	Tipusnév: xxxx
A	Funkció: xxxx
B	Tipus: (
C	OK Show érték (E) /Ref (R)/-:

Válaszok:

C : Menü:

a.) Show érték: E
Az elemek leirására vagyunk kiváncsiak, esetleg módositani is kivánjuk őket. Indul a P411 panel.

b.) Referencia : R
Indul a P4Ø1 panel.

c.) Nem OK: -
A funkció és tipus változtatható (a cursor a funkción áll)

d.) OK: üres CR
Jóváhagyjuk a tipusleirást.

P411 panel

Funkciója: A leszámlálható tipus elemeinek kijelzése és azok esetleges módositásának lekérdezése.

Formátum:

	Standard fejléc
	Tipusnév Funkció Tipus: } az előző panelből
A.	elem: .....
B.	funkció: .....
C.	OK/KOV.ERTEK (K)/TÖRÖL (D)/ -:

Válaszok:

A. Elem:
A leszámlálható halmaz egy eleme.

B. Funkció:
Egy, az elemre vonatkozó megjegyzés.

C. Menü:

a.) OK: elfogadjuk a tipust.

b.) Show következő érték: K.
A köv. elem kijelzését kérjük. Ha nincs, akkor a VÉGE kiirás jelenik meg, s a menü leszükül az OK/- lehetőségekre.

c.) Törlés : D.
Törli a kijelzett elemet.

d.) Nem OK : -
Nem fogadjuk el a kijelzett elemet, vagy a kijelzett VÉGE kiirást. A cursor az A kérdésre áll, s lehet módositani.

P42 panel

A résztartomány tipust kezeli.

Formátum:

Standard fejléc	
Tipusnév Funkció Tipus: ..	az előző panelből örökölt tartalommal
OK/Show határok (H)/Ref (R)/-:	

Válaszok:

H : indul a P421 panel.
nem OK: a cursor a funkcióhoz áll. Módositható a funkció és a tipus.
OK : a tipusleirás rendben, visszatérünk az előző panelra.

(P421 panel)

Résztartomány határait definiáló panel

Formátuma:

	Standard fejléc	
	Tipusnév Funkció Tipus: ..	az előző panelből
A.	alsó határ : ..	
B.	felső határ: ..	
C.	OK/-	

Válaszok:

A. A résztartomány alsó határa,
B. A résztartomány felső határa,
C. Menü:
   a.) OK: a kijelzett tipust jónak itéljük,
   b.) Nem OK:-
   Javitani akarunk. A cursor az A kérdésre áll.

P43 panel

A rekord tipus leirását vezérli

Formátuma:

Standard fejléc	
Tipusnév: } Funkció: }	az előző panelből
Tipus: RECORD	
OK/SHOW TAG:T / SHOW CASE:C / REF(R) / - :	

Válaszok:

Menü:
C (show case) : előhivja a Case definiciós ágat (P432)
T (show tag) : előhivja a rekordleirás köv. tagját (P431)
- : a cursor visszaáll a funkcióra és lehet módositani a funkciót és a tipust.
OK : rendben; visszatérés az előző panelre.
R : referencia
Indul a P4Ø1 panel.

P431 panel

Egy rekordtag leirását kezelő panel

Formátum:

	Standard fejléc
	Tipusnév } örökölt! Funkció } örökölt! Tipus: RECORD
A.	tagnév:
B.	funkció:
C.	tipus:
D.	OK/Köv.tag (K)/TÖRÖL (D)/ -:

Válaszok:

D : A menü:

D : Tag törlése, jön a következő

K : Show köv. tag

- : javitás

OK(CR): rendben, visszatérés.

Uj elem beirása a végén lehetséges csak, amikor elfogytak a régi elemek. Ilyenkor a tagnév helyén a vége kiirás jelenik meg, s a Menü-ben OK és — között választhatunk. Ha OK, akkor visszatérés a P43-ba, ha nem, akkor egy üres tag-menü panel jelenik meg, ahol beüthetjük az uj tagot, funkciót és tipust.

P432 panel

Case definiciós panel

Formátum:

	Standard fejléc
	Tipusnév Funkció } örökölt Tipus
A.	Casenev:
B.	Funkciója:
C.	Tipus:
D.	OK / SHOW cimke (C) / TOROL : D / -

Válaszok:

C : Show cimke (átmenet a cimkeleiró P4321 panelre!)

\- : változtatás a case-változó leirásán

OK(CR): helyben hagyás

Torles: törli a case-t.

Ha nincs case, akkor a casenév helyén a NINCS CASE kiirás jelenik meg. A menü-ben OK-val elfogadhatjuk ezt az állapotot vagy - jellel kérhetjük egy case ág definiálását.

P4321 panel

Case cimkét definiál

Formátum:

	Standard fejléc
	Tipusnév Funkció Tipus } örökölt
A.	Case-cimke: xx
B.	Funkció: xx
C.	OK / KOV.CIMKE (K) / TOROL (D) / SHOW TAG (T) / - :

Válaszok:

C: A menü:

D : az érték törlése (az egész ág elmarad)

T : átmenet a tag leirására (P43211)

K : átmenet a következő cimkére.

- : változtatás

OK: jóváhagyás.

Ha már nincs több cimke, akkor a VÉGE kiirás jelenik meg a case cimke helyén, s a menüben OK és - között választhatunk. - ra uj cimke megadására van mód.

P43211 panel

Case ágban levő tag leirása

Formátum:

	Standard fejléc
	Tipusnév Funkció Tipus Case - cimke } örökölt
A.	tagnév:
B.	funkció:
C.	tipus:
D.	OK / köv.tag (K) / töröl: (D) / - :

Válaszok:

D. A menü:
D: tag törlése,
K: köv.tagra való átmenet,
-: változtatás,
OK: jóváhagyás

Uj bevitel a végén, üres tagnév ill. tipus esetén lehetséges.

Ha nincs több tag, akkor a VEGE sor iródik ki a a menüben
OK és - jel között választhatunk. Ha -, akkor uj elemet illeszthet
be.

DESHELP / ELOTET

Felhasználói kézikönyv

MKKE - MSZI

1982.

1. Általános leirás:

Az XHELP alrendszerek által használt PDL modul-file-ok nem közönséges szöveg-file-ok. Az őket létrehozó DESHELP/EDITOR egy strukturaeditor, igy ismeri a modul box-szerkezetét, s a box-szerkezet jellemző információit minden sor szövege elé két karakterben ki is irja, mikor output file-t készit. Ezt a struktura-információval bővitett szöveget nevezzük DESHELP formátumnak.

Normál, XHELP-en kivüli editorok nem DESHELP formátumot hoznak létre. Használatuk ugyanakkor indokolt lehet, hiszen vitathatatlan, hogy a forgalomban levő context, full-screen stb. editorok kezelési komfortja tulszárnyalhatja a DESHELP/EDITOR-ét.

Célszerü tehát megteremteni annak a lehetőségét, hogy XHELP-en kivüli editorok által létrehozott file-ok az XHELP-ben feldolgozhatók legyenek.

Ezt biztositja a DESHELP/ELOTET alrendszer. A megadott szövegfile-t DESHELP formára alakitja, igy további feldolgozásra alkalmassá teszi.

Az XHELP feldolgozási egysége a PDL modul, ami egy funkcionális egység, és egy procedura is egyben. Ebből az következik, hogy idegen editorral sem lehet egynél több procedurát egy file-ba tenni.

A file névre az XHELP-ben kötelezően betartandó konvenció, hogy a file-ban tárolt procedura nevéből kell képződnie, az alábbiak szerint:

Az XHELP-ben a procedura neve és a file neve között szoros összefüggés van. Az általános szabály a következő:
a proceduranév a funkció leirásának alfabetikus jeleiből képződik, a file-név a proceduranév első 9 karaktere, ellátva egy alrendszertől függően változó file-tipussal.

Például:

Egy procedura PDL hivatkozása:

AZ "X VALTOZO BEOLVASASA (INSTREAM)

A proceduranév:

AZXVALTOZOBEOLVASASA

A modult tartalmazó file-név:

AZXVALTOZ.tipus

Ezen képzési szabály betartását az ELOTET program automatikusan biztositja, ugy, hogy:

az első "procedure" kulcsszó után álló procedura-névből képzi az output file nevét, a fenti konvenció szerint.

Tehát ha pl. az X.PDL file-ban editáltuk a

procedure osszeg (S,Y)

modult, s erre engedjük rá az ELOTET-et, akkor az output file neve a következő lesz:

OSSZEG.DES

2. Kezelés:

Az ELOTET program a MONITOR-ból a "DESHELP" és "ELOTET" válaszokkal hivható.

Induláskor megjelenik az alábbi panel:

```
XHELP FEJLESZTOI RENDSZER                    DESHELP/ELOTET

A NEM DESHELP EDITORRAL LETREHOZOTT MODUL ATVALTASA DESHELP
FORMARA

A MODUL NEVE: .....
```

Válaszként adjuk meg az átalakitandó file nevét, extensionnal együtt.

A válasz adásakor a szokásos konvenciók élnek, azaz

DELETE karakter egy beütött karaktert töröl,
\ jel a teljes választ törli,
- jel hatására a program visszatér a MONITOR-ba.

Ha nem létező file-nevet adtunk meg; akkor a

<<< NEM LETEZO FILE

hibajelzés iródik ki. Egy CR karakterrel nyugtázva a panel ujra inditható.

Hibás file-név (pl. üres file-név, hibás karakterek) esetén a hibajelzés szövege:

<<< HIBAS FILENEV

Helyes file-név esetén a konvertálás indul.

Ezt a képernyőn a

RUN

kiirás jelzi.

A program futásának végén kiirásra kerül:

```
ELOTET END
AZ INPUTFILE : XXX ...
AZ OUTPUTFILE: XXX ...
```

Az outputként létrehozott DESHELP formátumu file neve az első procedura nevéből képződik. Ha a PDL-leirás nem a "procedure" fejléc-sorral kezdődik, akkor az output file alapneve azonos az input file alapnevével, tipusa pedig .DES.

Az output file bekerül a DD könyvtárba, de benne marad a felhasználó saját könyvtárában is.

PR - 0

PROGHELP

Felhasználói kézikönyv

MKKE - MSZI

1982.

1. Általános leirás:

A PROGHELP az XHELP rendszer programgeneráló alrendszere. Feladata a tervező alrendszerek (EDITOR, DDFILL) által létrehozott PDL modulokból PASCAL eljárások generálása ill. komplett PASCAL programok kialakitása. Ennek megfelelően két funkciója van:

a) PDL nyelvü modul-leirás átkonvertálása PASCAL nyelvre s egyidejüleg az adatdeklarációs rész illesztése.
b) A Fejlesztői Adatszótárban (DD-ben) szereplő modulokból egy komplett program összeállitása.

Az a) funkciót a PROGHELP konvertáló része: a továbbiakban PROGHELP/KONV (vagy röviden KONV) végzi, a b) funkció a PROGHELP/GEN (röviden: GEN)feladata.

1.1. A PROGHELP/KONV funkciója

A Fejlesztői Adatszótárba bejegyzett modulok három csoportba tartoznak:

a) A PDL szinten kifejtett, leirt modulok. Környezetük, procedurális meghatározásuk, hivási strukturájuk már adott.
b) Definiált dummy modulok: Csak a környezetük adott. Procedurális részük definiálatlan, hivott moduljuk nincs.
(A környezet megirása a "dummy" kulcsszóval történik - ld. PDL ismertetőt!)
c) Meghatározatlan modulok: Már hivatkozott, de PDL szinten még semilyen formában nem definiált modulok. Környezet nélküli fekete dobozok.

Az a) és b) kategóriákba tartozó modulok leirása PDL formalizmusu. Az XHELP-ben használt PDL nyelv PASCAL irányultságu, de néhány lényeges eltérés van a két leirási mód között. A legfontosabbak:

- A PDL sororientált utasitásszerkezetü, nem a pontosvessző a terminátor.
- A strukturált utasitásokat és a modulokat mindig explicit END... zárja.
- A PDL leirás adat- és tipusdeklarációkat nem tartalmaz; azokat egy külön, gép által vezérelt dialógusban a DDFILL alrendszerrel adja meg a felhasználó.
- A PDL leirások külön-külön modulokat jelentenek. Ezek nagyobb egységekbe kombinálását az XHELP rendszer végzi, egy PDL leirás mindig egyetlen "procedure" ill. ahogy az XHELP dokumentációkban nevezzük : modul.

A PDL ⟷PASCAL eltérései miatt a programgenerálás első lépése egy konvertálás kell legyen. Ez általában szorosan kapcsolódik a modul létrehozásának folyamatához, vagyis többnyire az EDITOR - DDFILL - PROGHELP/KONV a feldolgozás sorrendje.

A PROGHELP/KONV kiteszi a pontosvesszőket, törli a speciálisan PDL-re jellemző karaktereket, beilleszti a szükséges begin-end párokat; és a modul elejére fejlécet ill. adatdeklarációkat ir.

A létrejövő PASCAL modul a DD-ben megőrzésre kerül; a PDL változattal azonos névvel, de .PAS tipusu file-ként.

1.2. A PROGHELP/GEN funkciója

Egy programrendszer fejlesztésénél a készülő részeket már készitése közben verifikálni lehet. Ehhez viszont szükséges, hogy

- a rendszer elemeit rugalmasan variálni lehessen,
- ill., hogy a még nem irt részek valamilyen módon - legalább a hibátlan szintaxist lehetővé téve - pótlásra kerüljenek.

Hagyományos módszerekkel a fejlesztés közbeni verifikálás igen komoly adminisztrációt igénylő, fáradságos munka. A GEN rendszer automatizált megoldást biztosit. A felhasználó csupán kijelöli azt a modult, mely köré programot óhajt. Ez lehet a fejlesztendő rendszer tényleges "main" moduljo, de moduláris teszt is elképzelhető olyan fiktiv "főprogram"-modulok révén, melyek mindössze bizonyos adatelőkészitést, a tesztelendő modul hivását és esetleg tesztkiiratást tartalmaznak.

A GEN program a DD-ben tárolt információk alapján kinlakitja a hivási fát, s az abban szereplő modulokat egy program-keretben egyesiti. A keret a tipusok és a globális változók deklarációját és egy egyszerü, indító funkcióju procedurális részt tartalmaz.

A modulok közül a kifejtett modulok változatlan formában átmásolódnak a keretbe. A definiált dummy modulok szintaktikailag helyes keretként kerülnek be, mig a passziv, black-box modulokat a rendszer belső rutinként fogja fel, s igy semmivel nem helyettesiti.

Néha szükséges, hogy egy már kifejtett modult nem kifejtett formában kezeljen a GEN. Ilyen eset lehet például az, amikor már letesztelt programrészeket nem akarunk tovább szerepeltetni a feldolgozásban. Ezt a GEN ugy teszi lehetővé, hogy opcionálisan kijelzi a hivási fában szereplő kifejtett modulok nevét, s mindegyiknél lehetőséget ad a felhasználónak a "dummy"-vá történő visszaminősitésre.

A GEN a programtesztelés hatásos segédeszköze. Tisztán kell azonban látni, hogy szolgáltatása csupán egy szintaktikailag helyes modulgyüjtemény létrehozása. A funkcionális teszt tényleges kivitelezéséhez elengedhetetlen teszt driver ill. assertion figyelő az XHELP rendszer egy másik szolgáltatása, a TESZTHELP.

2. Kezelés:

2.1. A PROGHELP inditása

Az XHELP/MONITOR menüjében a PROGHELP-et választva a PROGHELP vezérpanel indul.

Formátuma:

```
XHELP FEJLESZTŐI RENDSZER                          PROGHELP

A PROGHELP FUNKCIOI:

K - KONVERTALAS PDL-BOL PASCALBA

G - GENERAL EGY PROGRAMOT

MELYIK FUNKCIOT KERI?: _
```

Az elfogadott válaszok: K vagy G

K esetében a PROGHELP/KONV,
G esetében a PROGHELP/GEN indul.

Rossz válasz esetén a

<<<< ROSSZ VALASZ

hibaüzenet iródik ki a panel aljára. Az üzenetet RETURN billentyüvel nyugtázni kell, majd ujra kiiródik a vezérpanel.

2.2. A PROGHELP/KONV futtatása

A KONV az alábbi panel-lal jelentkezik be:

```
XHELP FEJLESZTŐI RENDSZER                     PROGHELP/KONV

EGY MODULT KONVERTAL PDL-BOL PASCAL-BA

A MODUL NEVE:
```

Meg kell adni a feldolgozandó PDL modul nevét. Ez mindig egy .DES tipusu file, melyet a KONV elsődlegesen a felhasználó saját könyvtárában keres, s csak ha ott nem találja,

akkor megy a DD könyvtárba.

Hibaüzenetet kapunk, ha

- nem .DES tipusu a file
- nincs ilyen modul sem a saját, sem a DD könyvtárban.
- a DD-ben nincs az adott néven bejegyzett modul.

A hibaüzenet az első és második esetben:

<<<< NEM JO INPUT FILE

A hibaüzenetet RETURN-nel nyugtázva ujra a KONV bejelentkező panel indul.

A harmadik esetben a hibaüzenet:

<<<< NINCS A DD-BEN ILYEN MODUL

A hiba fatális, a KONV véget ér.

Az eredmény-file az input file nevével azonos alapnevü, de .PAS tipusu. Átmásolódik a DD könyvtárba, de megmarad a felhasználó könyvtárában is.

2.3. A PROGHELP/GEN futtatása

A GEN bejelentkező panelja:

	XHELP FEJLESZTŐI RENDSZER PROGHELP/GEN
A	MELYIK MODUL LESZ A FOPROGRAM ?: _
B	KERI A HIVOTT MODULOK FELSOROLASAT ?: _

Válaszok:

A: Meg kell adni a főprogram-modul nevét. Ez nem egy file-név, tipusa nincs - lényegében a modult leiró file-ok alapneve. A főprogramnak kijelölhetjük a tényleges "main" modult, de használhatunk más, teszt-célokra készitett modulokat is. A főprogramként használt modulnak nyilván nincs paramétere.

B: A lehetséges válaszok: I/N (igen/nem).
"Igen" esetén a program a hivási fa összeállitása során kiirja a definiált modulok nevét (egyszerre egyet), s egy, a képernyő alján megjelenő menüvel választani lehet, hogy rendben van-e, vagy pedig dummy-val történő helyettesitését kérjük.

MODULNEV: XXXXXXXX.....

OK/DUMMY (D):

CR-t ütve OK, azaz jóváhagyást, D-t ütve pedig dummy-val történő helyettesitést értelmez a gép.

Hibajelzések:

<<<< ROSSZ VALASZ (B-re)
Ha a B. válasz nem I/N

<<<< ROSSZ VALASZ (a menü-re)
Ha nem CR vagy D a válasz.

A hibajelzéseket egy CR-rel kell nyugtázni, majd ismétlődik a kérdés.

Amennyiben nem létező modult jelöltünk ki főprogramnak, a rendszer nem jelez hibát, hanem egy dummy procedurát tartalmazó keretet generál.

A hivott modulok keretbe történő bemásolása során előfordulhat, hogy nem definiált modul kerül elő. Olyan tehát, amire hivatkozás történt, de PDL-ben nincs definiálva. Ezeket a GEN standard procedurának értelmezi, s nem próbálja dummy-val helyettesiteni.

Az összeállitott program MAIN.PAS néven a felhasználó könyvtárában található.

TE - 0

TESZTHELP

Felhasználói kézikönyv

MKKE - MSZI

1982.

1. Általános leirás:

A tesztelés és hibakeresés leghatékonyabb eszköze vitathatatlanul egy szimbolikus debugger. Segitségével a program breakpoint-nak nevezett helyeken megállitható, verifikációs feltételek ellenőrizhetők, szükség esetén értékeket (adatokat!) módosithatunk s tovább futtathatjuk a feldolgozást.

A debugger alapvető hátránya, hogy nem intelligens. A felhasználó a debugger révén csaknem mindent meg tud csinálni, ami a tesztelésnél szükséges, de a beavatkozást lehetővé tevő eszközön tulmenően semmi segitséget nem kap. A felhasználónak kell eldöntenie, hogy:

- hová helyez breakpoint-okat,
- az egyes verifikációs pontokban melyek a szignifikáns változók.

Nem kap továbbá támogatást a tesztadatok generálásánál ill. archiválásánál sem.

Az XHELP képes a fenti hiányosságok egy részének megszüntetésére, hiszen rendelkezésére áll a Fejlesztői Adatszótár, ami számos olyan információt tartalmaz, mely a felhasználónak tesztelés és hibakeresés közben hasznos lehet.

A TESZTHELP alrendszer

- potenciális breakpoint-okat épit be a program bizonyos helyeire. Ezek a pontok: az alprogramok belépési ill. nem dummy proceduránál az alprogramok kilépési pontjai, valamint az assert kulcsszóval jelölt verifikációs pontok.

  Az egyes "potenciális breakpoint"-nál a program megáll s lehetőséget ad a szimbolikus debuggerbe való áttérésre. A potenciális breakpoint-ok passzivizálhatók ill. ujra élesztheték.

- A TESZTHELP által instrumentált program debug-ba való áttérés esetén a debug-ból lekérdezhetően biztositja a megállási pont jellemző információinak elérését. Tehát ha pl. egy procedura elején levő potenciális breakpoint-ot a debugger-be ugrásra használjuk fel, akkor a debuggerrel le tudjuk kérdezni, hogy milyen input és output változók szerepelnek az adott helyen, vagyis hogy melyeket érdemes vizsgálni ill. esetleg változtatni.

A TESZTHELP tehát az intelligens debug lehetőségét biztositja.

A TESZTHELP inputja a PROGHELP/GEN által létrehozott MAIN.PAS program. Outputjai:

- a TMAIN.PAS program, ami a MAIN.PAS instrumentált változata,
- egy TESZT.DAT elnevezésü direkt file, amiben a potenciális breakpoint-ra vonatkozó információk találhatók,
- egy statisztika a beépitett breakpoint-okról ill. az informativ file-ról.

Az output file-ok a felhasználó saját könyvtárában találhatók. Az input file-t is a saját könyvtárban keresi. A feldolgozáshoz természetesen szükségesek a DD file-ok is, amit a TESZTHELP csupán olvasásra használ.

Az előállitott TMAIN.PAS közönséges PASCAL programként fordítható. Szerkesztéskor néhány run-time rutin hozzászerkesztendő, a futtatásnál pedig ismerni kell az instrumentált program mozgatásának és az információk lekérdezésének módját. Ezekről bővebbet a kezelést leiró rész tartalmaz.

SP - 0

SPECHELP

Felhasználói kézikönyv

MKKE - MSZI

1982.

T a r t a l o m

1. BEVEZETÉS

A Fejlesztői Adatszótár (a Data Dictionary) az XHELP-programjaival kezelt fejlesztések

modulstrukturáját,
adatstrukturáit és a
különböző szintü modulleirásokat

hivatott tárolni, karbantartani és lekérdezések esetén megjeleniteni. Ezek a főfunkciói elsősorban az XHELP programjai (DDFILL, PROGHELP, TESZTHELP) számára közvetlenül elérhetőek, egy run-time system formájában. A DD integritását és védelmét biztositó feltételek kezelését segédprogramok biztositják. Hasonlóan segédprogramok adnak információt, listákat az arra illetékes felhasználónak.

A következőkben a Fejlesztői Adatszótár file-rendszerével, ezek elérési módjaival, a speciális DD-file-ok szerkezetével, az ezekben történő adatmozgatás lehetőségeivel, továbbá a DD-kreálás, védelem és információ-visszanyerés segédprogramjaival foglalkozunk.

2. A FEJLESZTŐI ADATSZÓTÁR FELÉPITÉSE

2.1 A DD-könyvtár

A DD-könyvtár tartalmaz egy DD gyökér - file-t (PGMTABLA.TAB), ebben helyezkednek el a fejlesztések nevei egy szekvenciális táblában. Minden fejlesztéshez tartozik egy jelző-flag, mely jelzi, hogy az adott fejlesztés éppen változtatás alatt van. Ha ez a flag folyó változtatást jelez, akkor ezen fejlesztés adathalmaza csak olvasásra fórhető hozzá.

A DD-könyvtár minden fejlesztéshez felállit három direkt DD-file-t. Nevük: DD1, DD2 és DD3, az extension-juk Dii, ahol ii a fejlesztés PGMTABLA táblabeli sorszáma.

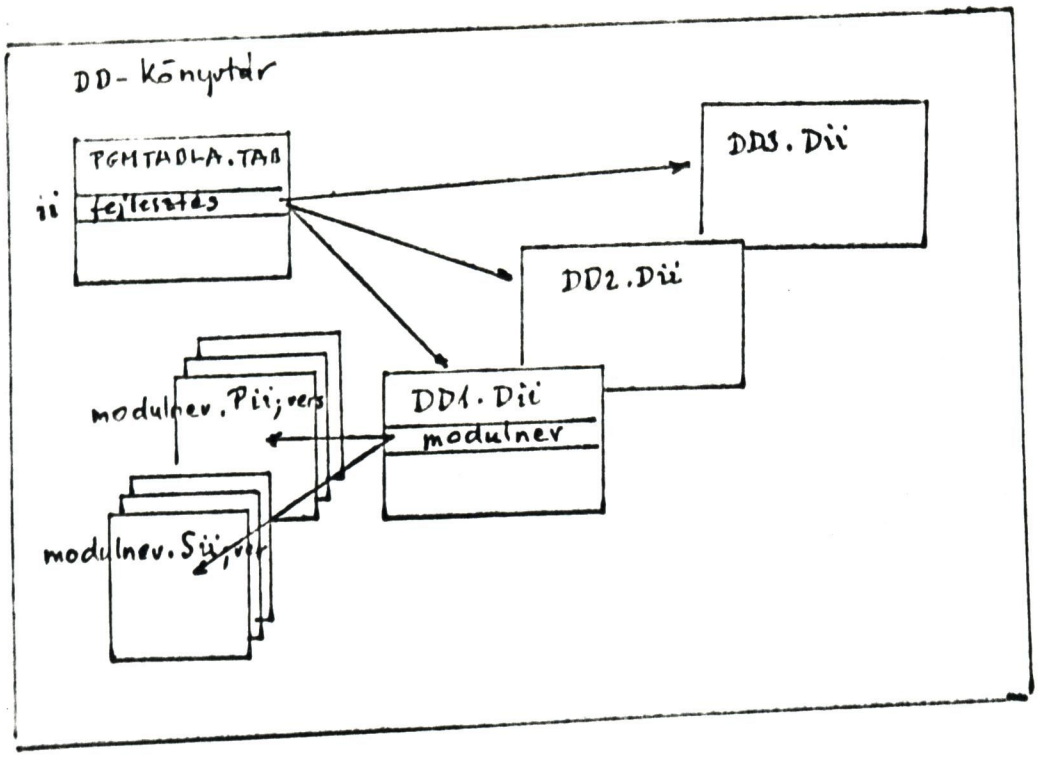
A DD1.Dii tartalmazza a fejlesztésben előforduló adat-, tipus-, konstans- és eljárásneveket. A DD2.Dii-ben a tipusokat specifikáló bejegyzéseket találhatjuk. A DD3.Dii a kereszthivatkozási láncok tárolására szolgál.

A DD-könyvtár tárolja továbbá fejlesztésenként a DDFILL modul-felvivő program által már a DD direkt-file-okba bejegyzett modulok file-jait, éspedig modulnév.Sii file-néven. A PROGHELP programgeneráló program eredményeként keletkező, PASCAL-nyelvü modulverziók is a DD-könyvtárban helyezkednek el, modulnév.Pii file-nevek alatt. A DD-könyvtár felépitését a 2.1 ábra szemlélteti.

2.2 A DD könyvtár file-jai

2.2.1 A gyökér-file (PGMTABLA.TAB) -

A fejlesztések sorszámát, nevét, a védelmi kulcsszót és a felhasználást jelző flag-et tartalmazó bejegyzésekből álló direkt file. A file fix-hosszuságu rekordjaiból alkotott táblázat a 2.2.1 alatt látható.



2.1 ábra

II	PGMNAME	PASSWORD	USERFLAG

MAXPGM

2.2.1 ábra

A tábla egy-egy sorának mezőspecifikációja:

PGMTABLA.II :integer -
a tábla sorának indexét tartalmazza, ha a fejlesztés él, egyébként 0.

.PGMNAME: array [1..30] of ascii -
a fejlesztés neve, alfabetikusan kezdődő, max. 30 karakteres string.

.PASSWORD:array [1..10] of ascii -
tetszőleges, max. 10 karakter hosszu. Ascii-string, a védelmi kulcsszó megadására.
(A PASSWORD ellenőrzését a másoló program végzi, értékét sem felhasználói, sem XHELP programokból nem kérdezhetjük le.)

.USERFLAG:boolean-
A DD1.Dii, DD2.Dii és a DD3.Dii file-ok kimásolt voltát jelző flag. Ilyenkor értéke TRUE. Az olvasás ez esetben is megengedett.

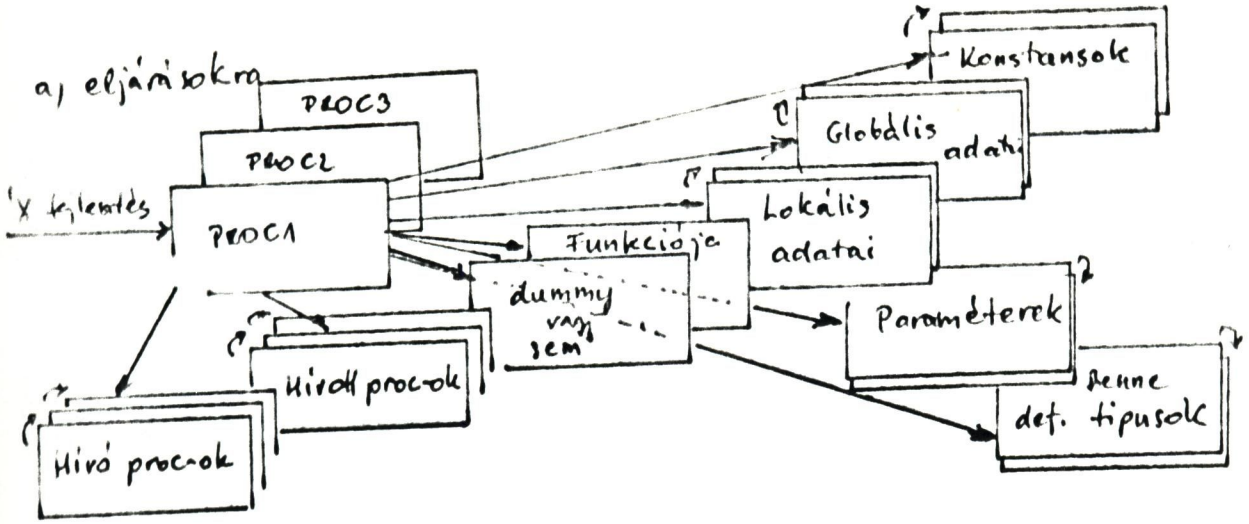
A MAXPGM, a tábla sorainak száma, a DD globális definiciói között megadott konstans.

2.2.2 Az egy fejlesztéshez tartozó DD-file-ok - (DD1.Dii, DD2.Dii, DD3.Dii)

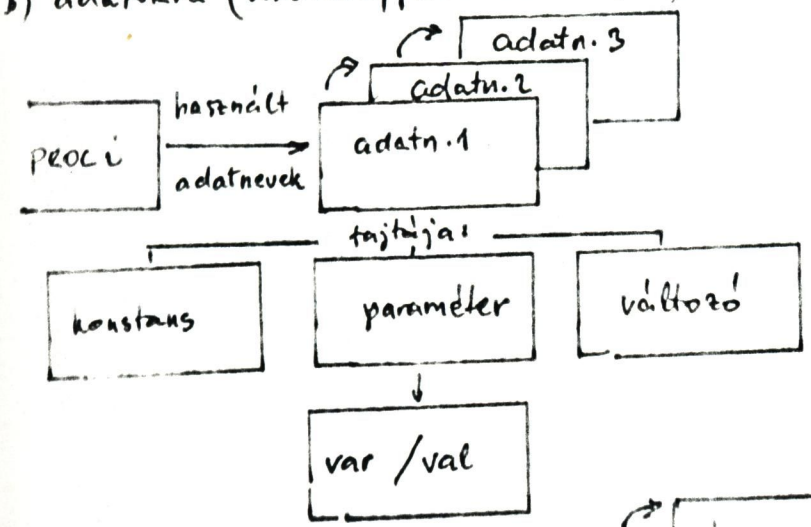
Egy-egy fejlesztés adat-, tipus- és eljárás-információit tároljuk ebben a három file-ban. A fejlesztés DD-jében gyüjtött információk elkülönülése a fentieknek megfelelően látható a 2.2.2 ábrán.

A kapcsolatok logikai vázlata is mutatja, hogy az információk többszörös kereszthivatkozásokkal rendelkeznek. Mivel a tipusok általában szerkezeti információt is hordoznak, a DD elkülöniti az azonositókat és azok tulajdonságait leiró adat

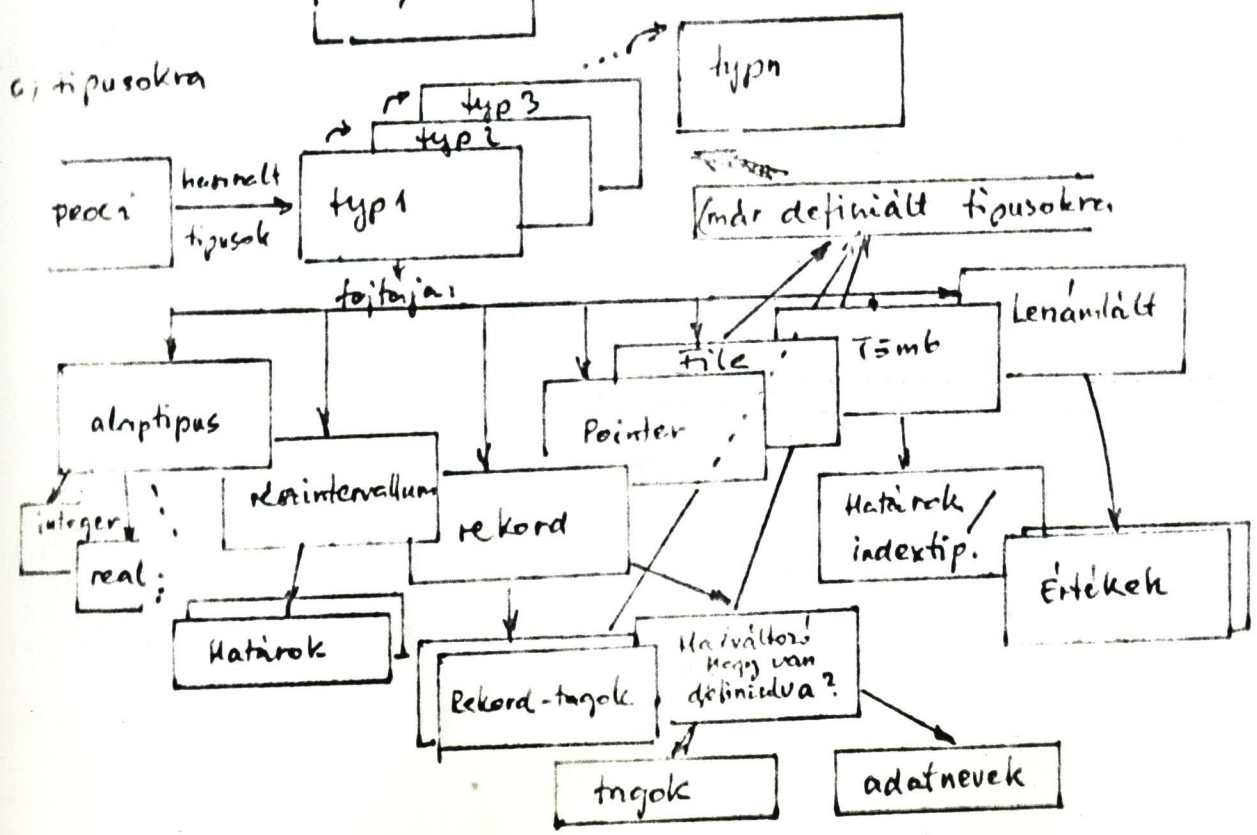
a) eljárásokra



b) adatokra (változók, paraméterek, ..)



c) típusokra



a, 2, 2 ábra

strukturáktól. Az előbbiek a DD2.Dii, az utóbbiak a DD1.Dii file-okban nyertek helyet. A kereszthivatkozások tárolását egy önálló detail-file-ban, a DD3.Dii-ben oldottuk meg.

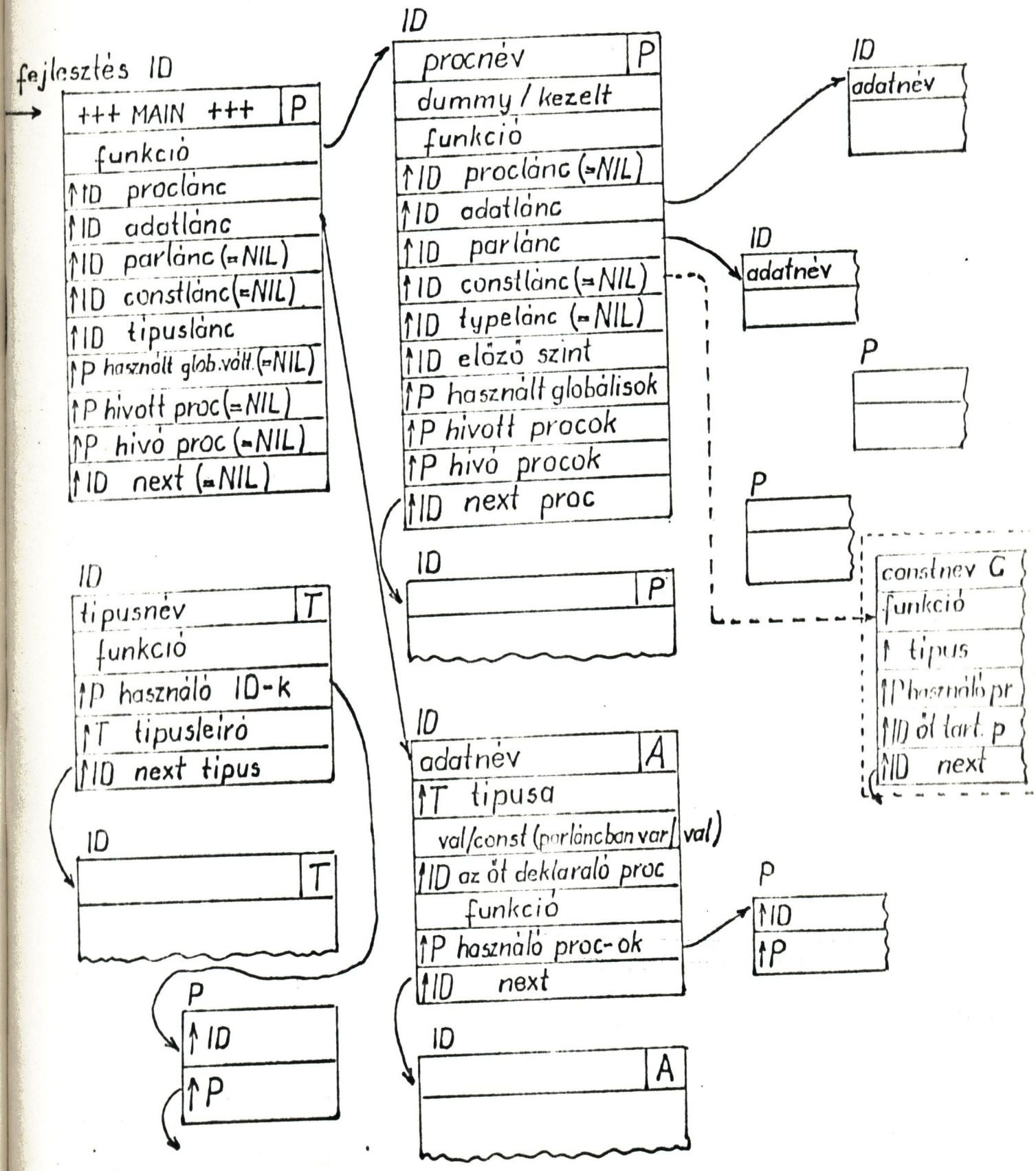
2.2.2.1 Azonositók file-ja (a DD1.Dii) -

A file ID-rekordokat tartalmaz. Az első rekordnak külön jelentése van: ez adja meg az első szabad ID-rekord sorszámát. Minden szabad rekord a következő szabad sorszámát tartalmazza, kivéve az utolsót, melyben itt végjel (0) áll. A második rekordtól kezdődően az ID-bejegyzések a fejlesztésel kapcsolatos nevek leirásait tartalmazzák, az alábbi 2.2.2.1/a ábra vázlata szerint. A file rekordjainak száma DD-konstans : MAXID.

A DD-rekordok definiciójának megfelelő globális neveket és a tömbök méreteit a 2.2.2.1/b ábrán találhatjuk. A direkt file-ban történő megvalósitások miatt a pointerek integer tipusuak.

A PIDDUMMY jelentése: TRUE, ha az eljárás még kezeletlen, és FALSE, ha már definiált.

Az AIDVAR TRUE, ha az adat -ID formális paraméter és input tipusu, FALSE, ha var tipusu.


fejlesztés ID
+++ MAIN +++
P
funkció
↑ID proclánc
↑ID adatlánc
↑ID parlánc (=NIL)
↑ID constlánc (=NIL)
↑ID tipuslánc
↑P használt glob.vált. (=NIL)
↑P hivott proc (=NIL)
↑P hivó proc (=NIL)
↑ID next (=NIL)
ID
procnév
P
dummy / kezelt
funkció
↑ID proclánc (=NIL)
↑ID adatlánc
↑ID parlánc
↑ID constlánc (=NIL)
↑ID typelánc (=NIL)
↑ID előző szint
↑P használt globálisok
↑P hivott procok
↑P hivó procok
↑ID next proc
ID
adatnév
ID
adatnév
P
P
ID
P
constnev C
funkció
↑ tipus
↑P használó pr
↑ID őt tart. p
↑ID next
ID
tipusnév
T
funkció
↑P használó ID-k
↑T tipusleíró
↑ID next tipus
ID
T
P
↑ID
↑P
ID
adatnév
A
↑T tipusa
val/const (parláncban var/val)
↑ID az őt deklaráló proc
funkció
↑P használó proc-ok
↑ID next
P
↑ID
↑P
ID
A

2.2.2.1/a ábra

IDNEV : array[1..AL] of ASCII
IDFUNKCIO : array[1..max[kt]] of ASCII
IDLINK : idpoi
IDOTTARTALMAZOPROC : idpoi / idfoglalt : boolean;

IDFAJTA (pid	cid	tid	aid	tagfid)
PIDDUMMY:boolean	CIDOTHASZNALOPROC: ppoi	TIDOTHASZNALO: pid	AIDOTHASZNALOPROCOK: pid	TAGFIDTIPUS: ~~tid~~ tpoi
PIDPIDP : idpoi	CIDTIPUS : tpoi	TIDTIPUSLEIRO: tpoi	AIDTIPUS : idpoi	
PIDCIDP : idpoi			AIDVAR : boolean	
PIDTIDP : idpoi				
PIDAIDP : idpoi				
PIDPARIDP : idpoi				
PIDHASZNALTGLOB : ppoi				
PIDHIVOTTPROC : ppoi				
PIDHIVOPROC : ppoi				

~~tagfiddummy:ppoi~~

ID-rekord

2.2.2.1 / b ábra

2.2.2.2 Tipus-bejegyzések file-ja (a DD2.Dii) -

A file-ban tárolt T-rekordok 11 variációja fordul elő. Az első rekord itt is a szabad helyek láncának kezdetét adja meg, a rekord sorszámának formájában. A szabadhely lánc előremutató, végjeles lánc, végjelként tartalmazva az utolsó rekordban egy 0-T. A file méretét a MAXT programkonstans állitja be.

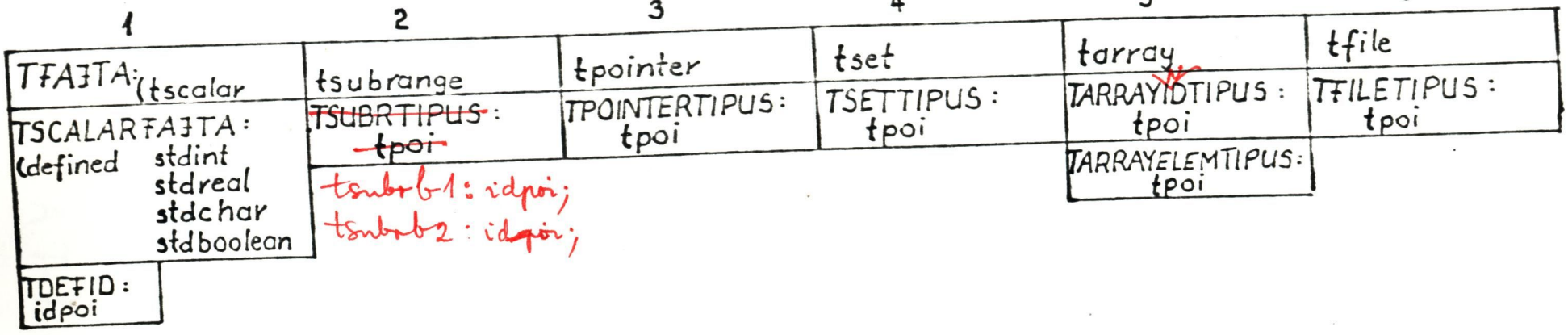
A 2.2.2.2/b ábrán mutatjuk be lehetséges T-variációkat. Tipus-variációt csak ID-rekord felöl érhetünk el önállóan, a belső láncok az adatstruktura leirását biztositják.

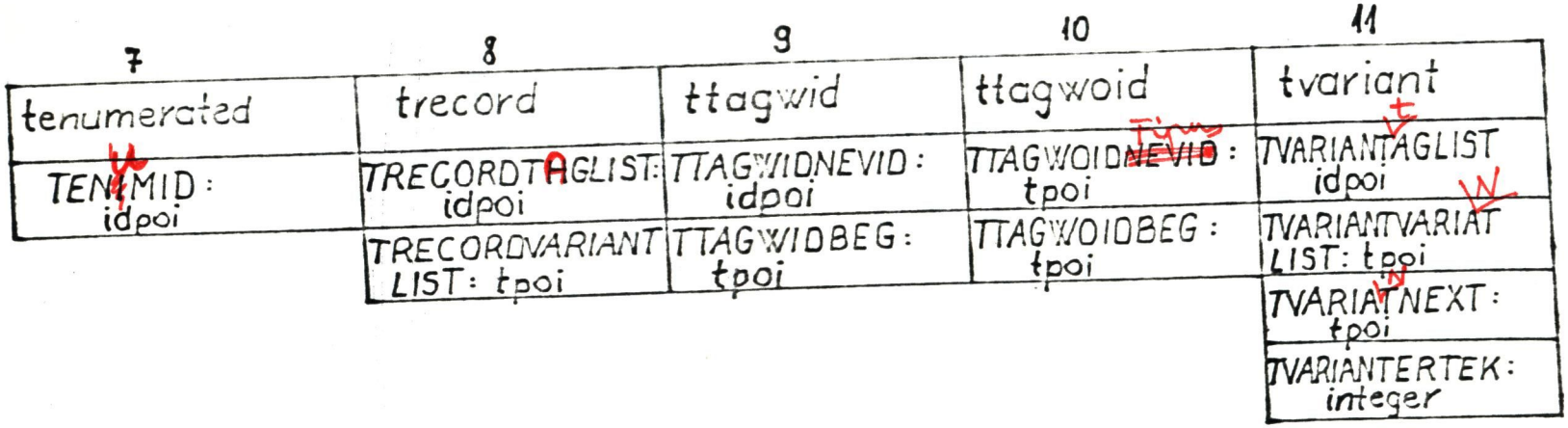
2.2.2.3 Pointerek file-ja (a DD3.Dii) -

Az adatszerkezetek külső megvalósitása miatt a hivatkozásokat leiró pointerek egy része, melyek feladata csak az információ-összekapcsolás, önálló file-ba kerülit. A szabadhely lánc megvalósitása azonos a DD2.Dii file-nál bemutatottakkal.

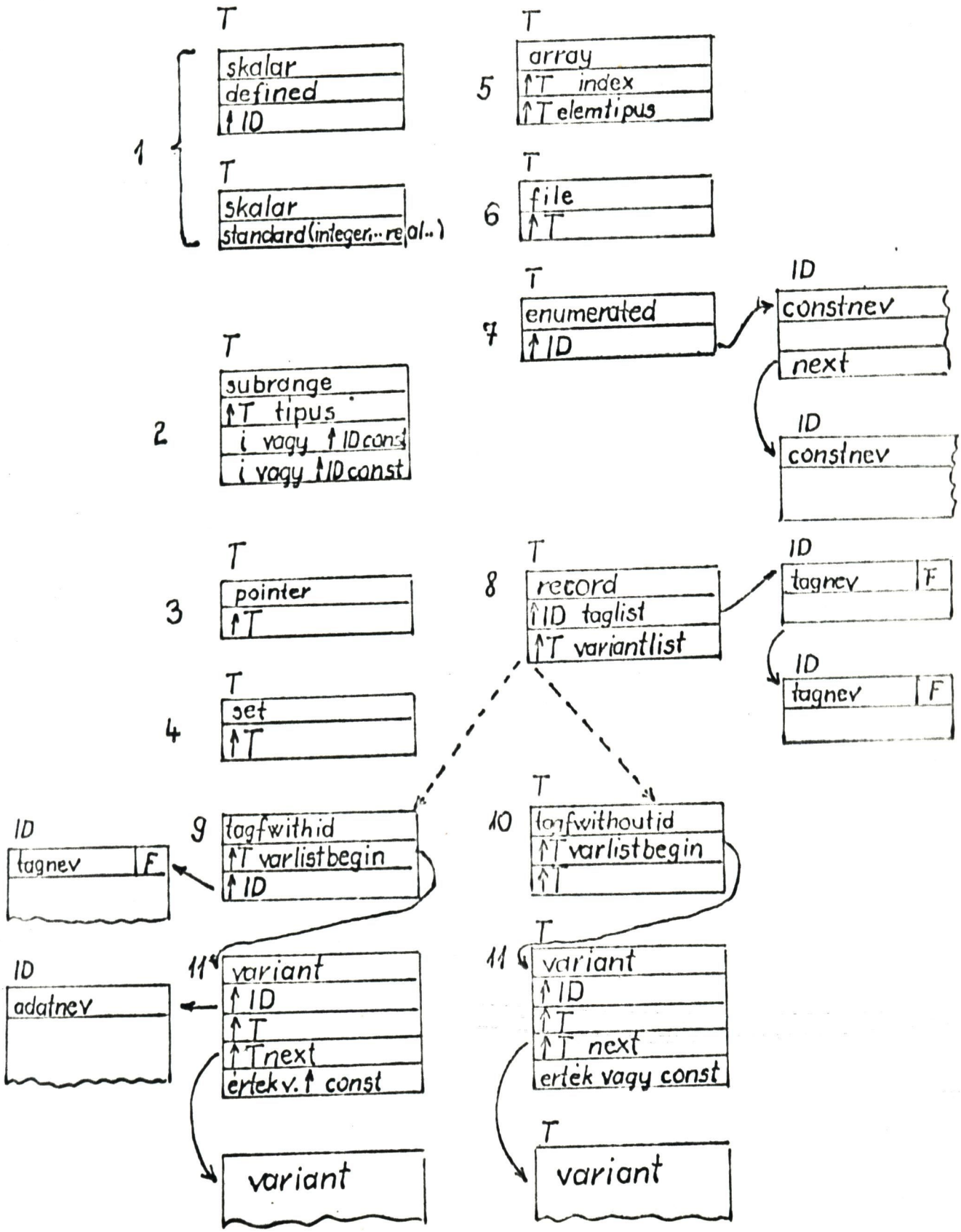
A rekordok azonos hosszuságuak, felépitésüket a 2.2.2.3 ábra mutatja.

A file méretét a MAXP DD-konstans határozza meg.


1
TFAJTA: (tscalar
TSCALARFAJTA: (defined stdint stdreal stdchar stdboolean
TDEFID: idpoi
2
tsubrange
TSUBRTIPUS: tpoi
tsubrb1: idpoi;
tsubrb2: idpoi;
3
tpointer
TPOINTERTIPUS: tpoi
4
tset
TSETTIPUS: tpoi
5
tarray
TARRAYIDTIPUS: tpoi
TARRAYELEMTIPUS: tpoi
6
tfile
TFILETIPUS: tpoi

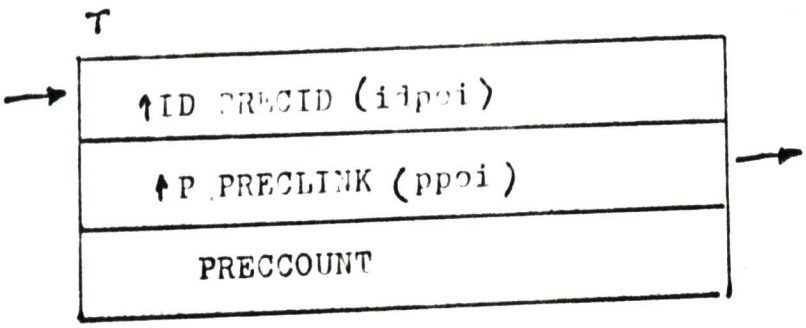

7
tenumerated
TENUMID: idpoi
8
trecord
TRECORDTAGLIST: idpoi
TRECORDVARIANTLIST: tpoi
9
ttagwid
TTAGWIDNEVID: idpoi
TTAGWIDBEG: tpoi
10
ttagwoid
TTAGWOIDNEVID: tpoi
TTAGWOIDBEG: tpoi
11
tvariant
TVARIANTAGLIST idpoi
TVARIANTVARIATLIST: tpoi
TVARIATNEXT: tpoi
TVARIANTERTEK: integer

T- rekord



A T-file bejegyzései

2.2.2.2./ a ábra



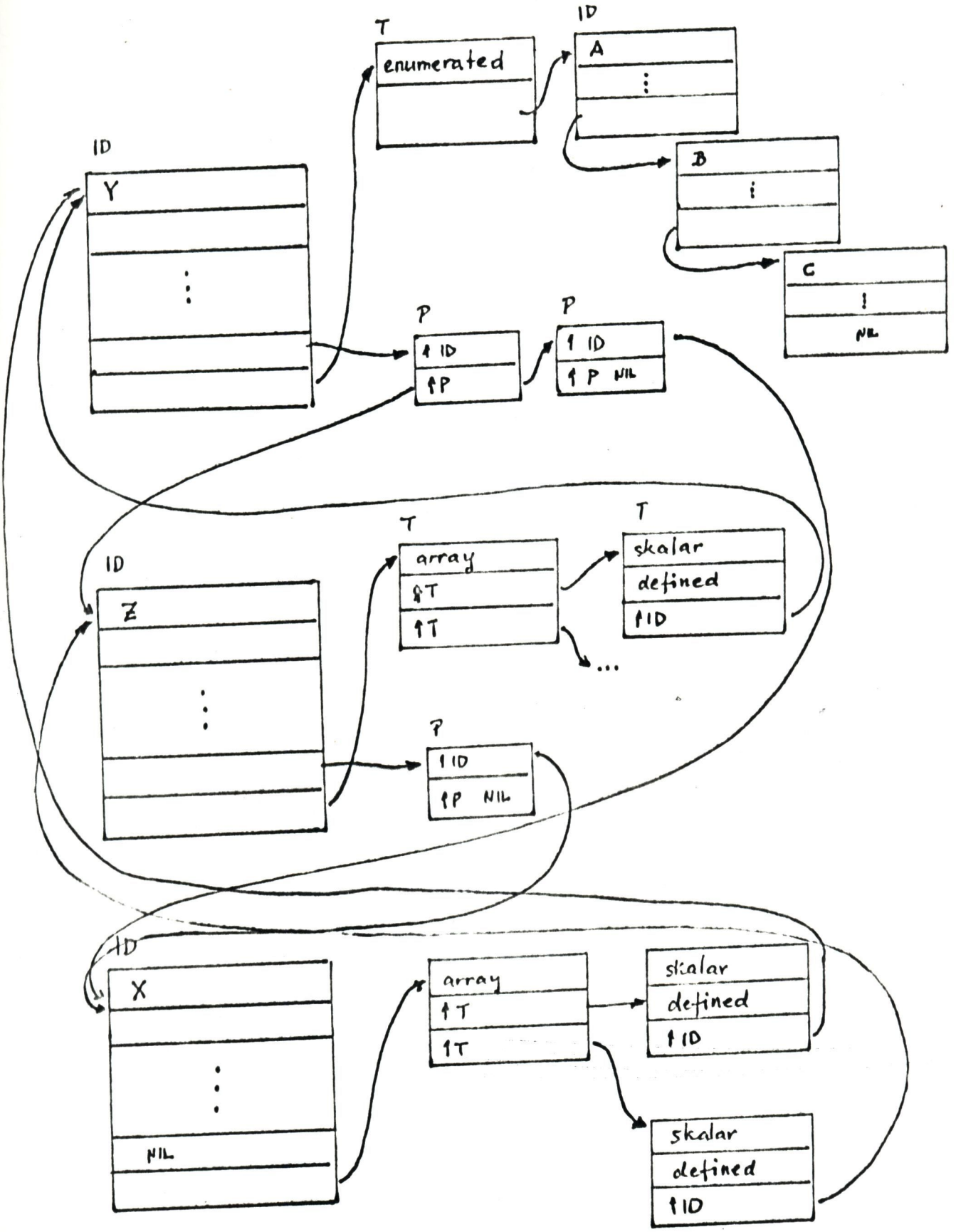
2.2.2.3 ábra

2.2.3 Szemléltető példa -

A fenti tipus-leirást mutatjuk be egy mintapéldán: A definició legyen a következő:

```
type    y = (a,b,c);
        z = array [y] of char;
        x = array [y] of z;
```

Ez a definició a DD adatszerkezetében az alábbi séma szerint nyer tárolást:


T
enumerated
ID
A
B
i
C
NIL
ID
Y
P
↑ ID
↑P
P
↑ ID
↑ P NIL
T
array
↑T
↑T
T
skalar
defined
↑ID
...
ID
Z
P
↑ ID
↑P NIL
ID
X
NIL
array
↑T
↑T
skalar
defined
↑ID
skalar
defined
↑ID

2.2.3 ábra

3. DD-T KEZELŐ PROGRAMOK IRÁSA

Minden a DD-vel kapcsolatos tevékenység nem a DD-könyvtárban, hanem a felhasználó saját UIC-je alá másolt DD-file-okban zajlik le. Ezért a tevékenység megkezdése előtt a fejlesztés direkt-file-jait és az esetleg szükséges .Sii ill. .Pii file-okat egy segédprogram révén a saját könyvtárunkba másoljuk át. A másolási időre a PGMTABLA a többi felhasználó elől blokkolt. Update-jellegü igény esetén a fejlesztés jelző-flag-e beállitódik s igy marad a visszamásolás befejezéséig. Ha az update tevékenység abort-tal fejeződik be, a visszamásolás elmarad, de a flag felszabadul. A DD1.Dii, DD2.Dii és DD3.Dii file-ok a felhasználó könyvtárában DD1.DOO, DD2.DOO ill. DD3.DOO file-nevekkel szerepelnek.

3.1 A könyvtár és a felhasználó kapcsolata

Egy adott fejlesztés hozzáférhetőségét

- védelmi kulcsszó
- használatot jelző flag

szabályozza.

A DDFILL feliró program induláskor automatikusan megtörténik a fejlesztés DD-jének átmásolása a felhasználói könyvtárba, de a kulcsszó helyes megadása kötelezően megelőzi ezt a funkciót.

Az átmásolás nemcsak hibás password esetén, de az esetben is elmarad, ha a fejlesztés DD-je éppen update-ra nyitott. A felhasználói programok nem várakoznak a visszairásig (vagy az esetleges program-megszakadásig).

A DD alá file-okat visszamásolni csak az XHELP programokon keresztül lehet.

A felhasználó UIC-jébe vitt DD-direkt file-ok a run-time system eljárásai segitségével mozgathatók meg. A run-time system az adatkezelést elemi szinten (azaz rekordonként) teszi lehetővé, a programozóra hárul az összetett funkciók (beszuráss, törlés, rekord-tartalom megváltoztatása, stb.) programozása.

3.2 A run-time system

A DD tartalomhoz fizikai szinten csak a run time system (RTS) eljárásai biztositanak hozzáférést. Csak a legegyszerübb funkciók megvalósitására törekedtünk, igy az RTS-eljárások száma nem nagy. Ezek:

DDINIT	- DOO file-ok megnyitása
DDTERMINATE	- első rekordok visszairása
ERROR	- hibajelzés
DDGETID	- rekord-beolvasások
DDGETT	- rekord-beolvasások
DDGETP	- rekord-beolvasások
DDPUTID	- rekord-visszairások
DDPUTT	- rekord-visszairások
DDPUTP	- rekord-visszairások
DDGETIDP	- szabadhelyek update-ja
DDGETTP	- szabadhelyek update-ja
DDGETPP	- szabadhelyek update-ja
DDPROCLOCATE	- névvel adott rekordok cime
DDADATLOCATE	- névvel adott rekordok cime
DDPARLOCATE	- névvel adott rekordok cime
DDPROCINSERT	- adott cimre rekord-beirás
DDADATINSERT	- adott cimre rekord-beirás
DDPARINSERT	- adott cimre rekord-beirás
DDINSOTHASZNALO	- ID-t tulajdonoshoz kapcsol
DDELOTHASZNALO	- az őt használók láncából kiveszi a hivatkozást

A RTS rutinok paraméterlistáit és leirásukat a DDUTIL fejlesztői dokumentációja tartalmazza. Csak különleges jogkörrel felruházott felhasználó használhatja őket.

4. A DD-VEL KAPCSOLATOS SEGÉDPROGRAMOK

A PGMTABLA.TAB file-al kapcsolatos tevékenységet két utility végzi. Az egyik létrehoz egy üres PGMTABLA-t. A másikkal uj fejlesztési bejegyzést helyezhetünk el a táblában. Az utóbbival egyidejüleg kreálódnak a DD1, DD2, DD3 file-ok. A file-másolással kapcsolatos tevékenységeket megfelelő engedéllyel felruházott másolóprogramok végzik. Ezek feladata a flag-állitás is.

A segédprogramok egy csoportja a DD-file-oket csak olvasásra megnyitva, listákat készit azok tartalmáról. Részletezésük a 4.2 pont alatt található.

A DD1.DOO, DD2.DOO, DD3.DOO átmásolt file-okban történő keresések, olvasások, csak a run time rendszeren keresztül valósithatók meg.

A 'UTILITY' XHELP/MONITOR command segitségével hivjuk be a DDUTIL utility-kezelő programot, mely az 'OPTION? kérdésre beütött segédprogram működését megszervezi. A lehetőségeket a következő pontokban soroljuk fel.

4.1 TECHNIKAI JELLEGÜ UTILITY-K

Egyedül az adatszótár-kreálás u.n. DD-független program. Minden más funkció csak a kreáláskor megadott (vagy az utolsó érvényes ismeretében egy segédprogrammal módositott) password begépelését követően válik lehetővé.

4.1.1 Adatszótár-kreálás (DDCREATE) -

Feladata:

Létrehozza az adatszótár programnyilvántartási táblájá és file-ját.

Müködése:

A CRE(ATE) parancs hatására létrejön a PGMTABLA.TAB tábla. A funkció csak a DD-könyvtár UIC-je alól indítható.

4.1.2 Jelszó-függő segédprogramok (DDUTIL) -

A DDUTIL program a képernyőre irja az

OPTION?

kérdést. A lehetséges válaszok (elég az első három karaktert beütni):

NEW	uj fejlesztés
PAS(SWORD)	uj jelszó
LIS(TPGM)	PGMTABLA-lista
PRO(C)	a PGm eljárásai
DAT(A)	adott eljárás adatai, tipusai
TYP(E)	eljárás tipusai
PRI(NTOF)	strukturált listák
SET	flag-et állit
CLE(AR)	flag-et töröl
DUM(P)	DD-file-ok dumpja

Minden kérést csak a PASSWORD lekérdezése és ellenőrzése után teljesit a DDUTIL.

4.1.3 Uj fejlesztés nyilvántartásba vétele (PGM) -

Feladata:

Uj fejlesztendő program nevét adjuk meg a DD-ban. Létrehozza az eljárás-, tipus- és adat-file-oket. Müködése a

NEW PROJECTNAME?

Kérdésre az Y vagy N karakterek egyikét kell beütni. A password a DDUTIL meghivásakor adott jelszó lesz.

A megadott jelszó elvesztése a további segédprogramok hivását lehetetlenné teszi. A RETURN megnyomása előtt ellenőrizzük a gépelést, mert a program rögtön letörli a beirt jelszót.

4.1.4 Password megváltoztatása (PASSWORD) -

Feladata:

A régi védelmi kulcsszó ismeretében egy uj jelszót adnatunk meg.

Működése:

Képernyőn megjelenik:

NEW PASSWORD?

üzenet. Ennek megadását és elküldését a képernyő azonnali letörlése követi, a RETURN beütése előtt ellenőrizzük és jegyezzük meg az uj jelszót. Az érvényes password lekérdezésére a DD-utility-k nem adnak lehetőséget.

4.2 INFORMÁCIÓVISSZANYERŐ UTILITY-K

A DD-tartalmát listázó programok outputjaikat a DDLIST.LST file-ba teszik, a felhasználó UIC-je alatt.

4.2.1 Tájékoztatás a DD-tartalmáról. -

A DDUTIL hivását követően a képernyőn megjelenik az

OPTION?

kérdés. Az adott választól függően a parancsok:

4.2.1.1 Listázza a program-directory-t -

Formátuma:

LIS(TPGM)

Hatása:

A PGMTABLA-file-ba bejegyzett fejlesztések neveit adja meg. A lista formátumát az A. függelék mutatja.

4.2.1.2 Eljárás-nevek listája -

Formátuma:

PRO(C)

Hatása:

A megadott programnév alá tartozó eljárásneveket irja ki, a definiciós struktura bemutatásával. Egy példa a B. függelékben látható.

4.2.1.3 Adatneveknek listája (tipusaikkal) - egy - eljáráshoz

Formátuma:

DAT(A)

Hatása:

A lista előbb a paramétereket, majd a lokális, s végül a globális változókat sorolja fel, tipusaikkal együtt. Egy példa található a C. függelékben.

4.2.2 Adott eljárás alatt definiált (explicit) tipusok -

Formátuma:

TYP(E)

Hatása:

A megadott fejlesztés tipus-definiciói kerülnek a listába. Egy szemléltető példát a D. függelék tartalmaz.

4.2.3 A DD tartalom speciális listái -

A

DUM(P)

parancs hatására a felhasználói DD-file-ok tartalma kerül a DD1.LST, DD2.LST ill. DD3.LST file-okba. Az E. függelékpontban mutatunk be szemléltetésül belőlük egy-egy részletet.

5. HIBAÜZENETEK

--- no more space in file: pgmtabla
- a PGMTABLA.TAB betelt

--- ddfiles were not locked
- CLEARFLAG - warning

--- ddfiles are locked, you must wait
- SETFLAG - warning

F U G G E L É K

XHELP FEJLESZTOI RENDSZER
LISTPGM

DDUTILITIES
1983-02-03

```
                                                                  0
    1 UJFEJL                                                      0
    2 MEGUJABB                                                    0
    3 LEGUJABB                                                     0
    0                                                              0
    0                                                              0
    0                                                              0
    0                                                              0
    0                                                              0
    0                                                              0
    0                                                              0
    0                                                              0
    0                                                              0
    0                                                              0
    0                                                              0
    0                                                              0
    0                                                              0
    0                                                              0
    0                                                              0
    0                                                              0
    0                                                              0
    0
```

MODULNEV: MAIN

LOKALISOK

ADATNEV :X
FUNKCIO :AZ OSSZEGZENDO TOMB
TIPUS :ARRAY [MERET] OF REAL

ADATNEV :TTY
FUNKCIO :A KONZOL
TIPUS :FILE OF CHAR

ADATNEV :S
FUNKCIO :AZ OSSZEG
TIPUS :REAL

ADATNEV :LENGTH
FUNKCIO :A TOMB HOSSZA
TIPUS :INTEGER

ADATNEV :KIIRATAS
FUNKCIO :IGAZ, HA KI KELL IRATNI
TIPUS :BOOLEAN

ADATNEV :I
FUNKCIO :CIKLUSVALTOZO
TIPUS :INTEGER

```
SUM
        WRITELN
        OSSZEGZESSBEN
        MERETBEOLVASAS
        AZXTOMBBEOLVASASA
```

XHELP FEJLESZTOI RENDSZER
FEJLESZTES :LEGUJABB

ODUTILITIES
1983-02-04

TIPUSOK

++++++++++++++++++++++++++

TIPUSID : MERET

FUNKCIO: A TOMB MERETE

ALSO HATAR :1
FELSO HATAR:100

TIPUS:INTERVALLUM

LP FEJLESZTOI RENNDSZER
LESZTES :LEGUJABB

1983-02-04

MODULNEV: MERETBEOLVASAS

PARAMETEREK

ADATNEV :LENGTH

FUNKCIO: AZ ADATOK AKTUALIS SZAMA

TIPUS :INTEGER

```
ID = ENTRY-K
-----------------------------

  1.     NEV:
FUNKCIO:
LINK:            18 ID
OTTARTALMAZO:     0 ID
FOGLALTFLAG: TRUE
***  PID  ***
DUMMY              N
PROCLANC:          0 ID
CONSTLANC:         0 ID
TIPUSLANC:         0 ID
ADATLANC:          0 ID
PARLANC:           0 ID
HASZN.GLOB.:       0 P
HIVOTT PROC.:      0 P
HIVO PROCOK:       0 P
==========================================================

  2.     NEV:
FUNKCIO:
LINK:              0 ID
OTTARTALMAZO:     0 ID
FOGLALTFLAG: FALSE
***  PID  ***
DUMMY              N
PROCLANC:          7 ID
CONSTLANC:         0 ID
TIPUSLANC:         9 ID
ADATLANC:          8 ID
PARLANC:           0 ID
HASZN.GLOB.:       0 P
HIVOTT PROC.:      0 P
HIVO PROCOK:       0 P
----------------------------------------------------------

  3.     NEV: SUM
FUNKCIO:      OSSZEGZO PROGRAM
LINK:              0 ID
OTTARTALMAZO:     2 ID
FOGLALTFLAG: TRUE
***  PID  ***
DUMMY              N
PROCLANC:          0 ID
CONSTLANC:         0 ID
TIPUSLANC:         0 ID
ADATLANC:          0 ID
PARLANC:           0 ID
HASZN.GLOB.:      22 P
HIVOTT PROC.:      8 P
HIVO PROCOK:       0 P
----------------------------------------------------------

  4.     NEV: AZXTOMBBEOLVASASA
FUNKCIO:
LINK:              3 ID
OTTARTALMAZO:     2 ID
FOGLALTFLAG: TRUE
***  PID  ***
DUMMY              Y
PROCLANC:          0 ID
CONSTLANC:         0 ID
TIPUSLANC:         0 ID
ADATLANC:          0 ID
PARLANC:           0 ID
HASZN.GLOB.:      32 P
HIVOTT PROC.:      0 P
```

HIVO PROCOK: 3 P

```
TYP    ENTRY-K
-------------------------------

    1.
***  POINTER  ***
TIPUSLEIRO:     13 T
================================================

    2.
***  ARRAY  ***
INDEXTIPUS:      4 T
ELEMTIPUS:       5 T
================================================

    3.
***  SUBRANGE  ***
BOUND1:         10 ID
BOUND2:         11 ID
================================================

    4.
***  SCALAR  ***
USERTIPUSNEV:       9 ID
================================================

    5.
***  SCALAR  ***
STANDARD:        2
================================================

    6.
***  FILE  ***
TIPUSLEIRO:      7 T
================================================

    7.
***  SCALAR  ***
STANDARD:        3
================================================

    8.
***  SCALAR  ***
STANDARD:        2
================================================

    9.
***  SCALAR  ***
STANDARD:        1
================================================

   10.
***  SCALAR  ***
STANDARD:        4
================================================

   11.
***  SCALAR  ***
STANDARD:        1
================================================

   12.
***  SCALAR  ***
STANDARD:        1
================================================
```

POINTER - FILE

1.	REF:	0 ID	LINK:	33 P	COUNT:	0
2.	REF:	4 ID	LINK:	0 P	COUNT:	0
3.	REF:	3 ID	LINK:	0 P	COUNT:	0
4.	REF:	5 ID	LINK:	2 P	COUNT:	0
5.	REF:	3 ID	LINK:	0 P	COUNT:	0
6.	REF:	6 ID	LINK:	4 P	COUNT:	0
7.	REF:	3 ID	LINK:	0 P	COUNT:	0
8.	REF:	7 ID	LINK:	6 P	COUNT:	0
9.	REF:	3 ID	LINK:	0 P	COUNT:	0
10.	REF:	3 ID	LINK:	25 P	COUNT:	1
11.	REF:	8 ID	LINK:	0 P	COUNT:	1
12.	REF:	8 ID	LINK:	0 P	COUNT:	85
13.	REF:	3 ID	LINK:	28 P	COUNT:	1
14.	REF:	12 ID	LINK:	12 P	COUNT:	8277
15.	REF:	3 ID	LINK:	0 P	COUNT:	1
16.	REF:	13 ID	LINK:	11 P	COUNT:	8277
17.	REF:	3 ID	LINK:	31 P	COUNT:	1
18.	REF:	14 ID	LINK:	16 P	COUNT:	8277
19.	REF:	3 ID	LINK:	0 P	COUNT:	1
20.	REF:	15 ID	LINK:	18 P	COUNT:	8277
21.	REF:	3 ID	LINK:	0 P	COUNT:	1
22.	REF:	16 ID	LINK:	20 P	COUNT:	8277
23.	REF:	5 ID	LINK:	0 P	COUNT:	1
24.	REF:	4 ID	LINK:	0 P	COUNT:	1
25.	REF:	4 ID	LINK:	0 P	COUNT:	1
26.	REF:	8 ID	LINK:	0 P	COUNT:	85
27.	REF:	4 ID	LINK:	0 P	COUNT:	1
28.	REF:	4 ID	LINK:	0 P	COUNT:	1

DO - 0

DOKHELP

Felhasználói kézikönyv

MKKE - MSZI

1982.

1. Általános leirás

A DOKHELP alrendszer a PDL modulok ill. a generált PASCAL programok dokumentációit állitja elő.

A DOKHELP/PDL a PDL modulok strukturadiagramos listáját és cross referencia listáját tudja létrehozni.

A DOKHELP/PAS szolgáltatásai:

- strukturadiagram lista,
- cross referencia lista,
- a modulstruktura rajza.

1.1. A DOKHELP/PDL funkciója:

1.1.1. Strukturadiagram kirajzolása

A DESHELP/EDITOR ismertetésénél már tárgyalt strukturadiagram a formátumozott programlistáknak egy továbbfejlesztett változata. Képszerüsége révén komoly dokumentációs értéke van. Egy PDL modulra alkalmazva lényegében az EDITOR által a képernyőn megjelenitett formát kapjuk meg listán.

1.1.2. Keresztreferencia lista készitése

A modulban előforduló változónevek ill. funkciók (procedurahivások) listája dokumentációs célokon tulmenően a DDFILL párbeszédekor is a fejlesztő hasznára lehet.

1.2. A DOKHELP/PAS funkciója

1.2.1. Strukturadiagram kirajzolása

A PASCAL forrásprogramról strukturadiagramot készitő alrendszer két lényeges dologban különbözik PDL-re adaptált rokonától:

- A PASCAL program egy procedura-halmaz, igy a lista célszerüen kiegészitendő egy tartalomjegyzékkel, mely megadja, hogy az egyes procedurák a listának melyik sorában kezdődnek.

- A PDL kirajzoló az EDITOR vagy az ELOTET által létrehozott DESHELP formából indul ki, igy lényegében szintaktikai elemzést nem végez. A PASCAL verziónak már elemeznie kell a programsorokat. A szintaktikai vizsgálat azonban elég egyszerü szinten marad, mert minden PASCAL boxot egy un. szintaxis-jelző comment kell hogy zárja. Ezek a comment-ek megfelelnek a PDL box-záró utasitásoknak, de comment-ként felirva. Tehát:

  (* endproc *)
  (* endif *)
  (* endwhile *)
  (* endcase *)
  (* endwith *)

  A programot záró end. után nem kell (* endproc *) sor.

Ezeket a szintaxis-jelző comment-eket a PROGHELP begenerálja a forrásprogramba. Amennyiben viszont XHELP-től idegen PASCAL forrásprogramokat akarunk a DOKHELP/PAS alrendszerrel dokumentálni, akkor gondoskodnunk kell arról, hogy a box-ok mindegyike a megfelelő comment sorral záruljon. Figyelni kell továbbá arra is, hogy a DOKHELP/PAS mindig a MAIN.PAS nevü file-t dolgozza fel, lévén ebben a fix nevü file-ban a PROGHELP mindenkori utolsó összeállitott programja.

1.2.2. Keresztreferencia lista készitése

A szokásos, egyszerü keresztreferencia lista minden nem kulcsszó-szimbólum előfordulási helyeiről ad egy táblázatot.

1.2.3. Hivási fa kirajzolása

A kirajzolás egy hierarchikus fa formájában történik, ahol a fa horizontális beosztása az egyes modulok legnagyobb hivási mélységét jelöli, a vertikális tagozódás pedig a modulok hivási sorrendjét reprezentálja. Az egyes modulokat keretbe foglalt információk jelzik, amelyek tartalmazzák a modul

nevét, tipusát (program, procedure, function), a modul kódját (amely a modulok deklarálási sorrendjére utal) és esetlegesen az alapvetőtől eltérő hivási módot (externális, forward, rekurziv). Az önmagát hivó rekurziv modulokat a keret alján egy a keretbe visszatérő nyil jelöli, a forwardként deklarált modul(ok) hivását pedig a keret alján a forward modul kódja(i)-ra mutató nyil jelzi.

Példa a kirajzolás formájára az XHELP általános ismertető 5. ábrája.

A hétnél nagyobb hivási mélységü modulokat a rendszer uj lapon rajzolja ki, maximálisan 15-ös hivási mélységig.

Az opcióként kérhető modultáblázat az output-file elejére, külön lapra kerül. Ez a modulok névre rendezett táblája, amely tartalmazza a modulok kódját, legnagyobb hivási mélységét (NESTING), tipusát (TYPE), a modulra való hivatkozások számát (CALLED), a modul által hivott modulok számát (CALLING) és a modul hivásának módját (MODE).

A rendszer müködéséhez szükséges az alábbi konvenciók betartása:

1. A forrásprogram maximálisan 99 modulból állhat. Ez a központi memóriában rendelkezésre álló helytől függően változtatható.
2. Az eljárásokat lezáró "END;" utasitás után az "(* END-PROC *) commentnek kell állnia. Ezt a PROGHELP biztositja a programgenerálás során.
3. A programban nem lehet skatulyázással létrehozott rekurzivitás. Ezt a PROGHELP szintén biztositja.

2. Kezelés

2.1. DOKHELP/PDL

Hivása: Az XHELP/MONITOR menüjére a DOKHELP válasz után a PDL-t kell választani.

2.1.1. Strukturadiagram kirajzoló

Hivása: A MONITOR menüben a STRUKTOUT választással.

Induláskor a következő panel jelenik meg:

```
XHELP FEJLESZTŐI RENDSZER                    DOKHELP/STRUKTPDL

A MODUL NEVE: .....
MELYIK ABRAZOLAST KERI?:  S
        (S  -  STRUKTOUT,
         L  -  LEIGHTON,
         V  -  VAZDIAGRAM)

LAPMERET:  6Ø
```

A cursor az első válasz elején áll. Értelemszerüen válaszoljunk a kérdésekre. Amennyiben egy előre megadott választ helyben hagyunk, akkor elég csak egy üres return-t adni. A hibás válaszokat a szokásos módon javithatjuk:

DELETE	- törli az utolsó beütött karaktert,
\ (backslash)	- törli az utolsó választ,
- (minusz)	- visszalép a monitorba. (abort funkció)

A "modul neve" kérdésre a feldolgozandó PDL modul nevét adjuk meg; tipus nélkül. Mivel a modul (file) nevét a DESHELP alrendszerei esetenként a funkció-leirásból automatikusan képezik, itt is megadhatjuk a funkció leirását, amit aztán a DOKHELP tömörit, levág 9 karakterre s file névként értelmez. A .DES tipust a DOKHELP illeszti.

A file-t a program először a saját könyvtárban keresi. Ha nincs, akkor a DD könyvtárban próbálkozik.

Az ábrázolási módok között a strukturadiagramon kivül a Leighton és a vázdiagram is adott lehetőség. A Leighton más ábrajelekkel dolgozik, a vázdiagram pedig csupán a blokkok keret utasitásait irja ki, azaz a modul "vázát" dokumentálja.

Hibák esetén a panel ujra indul, az addigi helyes válaszok érvényben maradnak.

Hibajelzések:

<<< hibás file név
<<< nem létező file
<<< értelmetlen válasz.

Minden hibajelzést egy üres sorral "nyugtázni" kell.

A program futását a RUN szócska jelzi.

Futás után az alábbi képernyő jelenik meg:

XHELP FEJLESZTŐI RENDSZER DOKHELP/STRUKTPDL

STRUKTPDL END
FORRÁSPROGRAM: ...
A LISTA TALALHATO: ...
{NO ERRORS
{ERROR'S FOUND

Ha kirajzoláskor hibát észlelt a program, akkor az ERROR'S FOUND szöveg iródik ki. A hibák a listában a felfedezés helyén hibaüzenetet eredményeznek. Minden hibaüzenet <<<< jelekkel indul.

Az output lista a felhasználó könyvtárában, az inputtal azonos alapnévvel, .LST tipussal képződik.

2.1.2. Keresztreferencia listázó

Hivása: A CREF választással a MONITOR menüben.

Egy árnyalatnyit módositott DDFILL1 program készíti a kereszt-referenciát.

A kezelési sorrend azonos az ott leirtakkal, de a panelek egy keveset változnak ill. a P12 panel elmarad.

A változások:
- A P1 panelban nincs "kiván CREF listát?:" kérdés.
- A P11 panelban mindig a CREF lista készül szöveg jelenik meg.
- A P13 panel teljesen kiiródik.

2.2. DOKHELP/PAS

Hivása: Az XHELP/MONITOR menüire a DOKHELP és a PAS válaszokat kell megadni.

2.2.1. A strukturadiagram kirajzoló:

Hivása: A MONITOR menüre a STRUKTOUT választ adjuk meg.

Induláskor a következő panel jelenik meg:

```
XHELP FEJLESZTŐI RENDSZER                    DOKHELP/STRUKTPAS

A FORRASPROGRAM NEVE:  MAIN.PAS
MELYIK ABRAZOLAST KERI?: .
        (S - STRUKTOUT,
         L - LEIGHTON,
         V - VAZDIAGRAM)

LAPMERET: 6Ø
```

A cursor az ábrázolási módra való rákérdezésnél áll. A modul neve mindig MAIN.PAS!

A Leighton ábrázolási mód a strukturadiagramtól az ábra-jelekkel tér el, a vázdiagram pedig a blokkoknak csupán a keret-utasitásait irja ki.

Hibák esetén a panel ujra indul, kivéve, ha nem létezik a file. Ilyenkor a monitor indul ujra.

Hibaüzenetek:

<<< nem létező file
<<< értelmetlen válasz.

A hibajelzést egy üres sorral nyugtázni kell.

Helyes válaszok esetén a RUN kiirás jelenik meg a képernyő alján.

Futás után az alábbi képernyő jelenik meg:

```
XHELP FEJLESZTŐI RENDSZER                          DOKHELP/STRUKTPAS

STRUKTPAS  END
FORRASPROGRAM:  MAIN.PAS
A LISTA TALALHATO:  MAIN.LST
{NO ERRORS
{ERROR'S FOUND
```

Ha elemzéskor vagy kirajzoláskor hibát észlelt a program, akkor az ERROR'S FOUND szöveg iródik ki. A hibák a felfedezés helyén, a listában, hibaüzenetet eredményeznek. A hibaüzenetek <<<< jelekkel kezdődnek.

Az output lista a felhasználó könyvtárában, a MAIN.LST file-ban található.

2.2.2. Keresztreferencia listázó

Hivása: A MONITOR menüben a CREF választással.

Induláskor a következő panel jelenik meg:

```
XHELP FEJLESZTŐI RENDSZER                          DOKHELP/CREFPAS

A MAIN.PAS PROGRAMROL CREF TABLAT KESZIT
A LAPMERET: ...
```

Adjuk meg a lapméretet!

Helytelen megadás esetén a

<<< HIBAS VALASZ

hibaüzenet jelenik meg. Egy üres sorral nyugtázhatjuk, ilyenkor a panel ujra indul.

Helyes válasz után a feldolgozás megindul. A képernyő alján megjelenő RUN szócska jelzi.

Futás után a záró képernyő:

```
XHELP FEJLESZTŐI RENDSZER                    DOKHELP/CREFPAS

CREFPAS  END
FORRASPROGRAM:  MAIN.PAS
A LISTA TALALHATO:  MAIN.CRF
```

Ezt egy CR-rel nyugtázva ujra indul a MONITOR.

2.2.3. A hivási fa kirajzolása

A DOKHELP/MODUL alrendszer az XHELP monitorának segitségével futtatható a megfelelő funkció kiválasztásával (MODUL).

Az alrendszer inditása után a képernyőn a következő panel jelenik meg:

```
XHELP FEJLESZTOI RENDSZER                    DOKHELP/MODUL

A GENERALT PASCAL PROGRAM MODULSZERKEZETE.
MODULTABLAT KER?  (I/N)  I
LAPSZELESSEG?            60
```

A cursor az első opció I (igen) default helyére áll. Ha a felhasználó nem kéri a modultáblát, akkor az N karaktert és a

returnt kell beütnie, ha kéri, akkor csak a return billentyüt kell leütnie. A második opció (default 60) az egy lapra irható sorok számát határozza meg.

Helytelen válasz (első kérdésre 1,N helyett egyéb karakter, második kérdésre nem numerikus válasz) esetén a képernyőn hibaüzenet ("NEM ERTELMEZETT VALASZ" ill. "NEM NUMERIKUS VALASZ") jelenik meg és return leütése után az egész panel ujra megjelenik.

Jó válaszok megadása esetén egy müködést jelző kiirás ("MODULSZERKEZET KIRAJZOLASA FOLYIK") jelenik meg és elkezdődik a feldolgozás.

A DOKHELP/MODUL lefutása után megjelenik a következő panel:

XHELP FEJLESZTŐI RENDSZER DOKHELP/MODUL

PASCAL FORRASPROGRAM: MAIN.PAS
MODULSZERKEZET RAJZA: MAIN.MOD
NYOMJA LE A RETURN BILLENTYUT!

Ezután egy return-nel visszatérhetünk az XHELP monitorához.

MO - 0

XHELP/MONITOR

Felhasználói kézikönyv

MKKE - MSZI

1982.

1. Általános leirás:

A Monitor az XHELP rendszer vezérlőprogramja. Feladata, hogy a rendszerhez forduló felhasználót azonositsa, lekérdezze az érintett fejlesztés nevét, majd menüsorozatok révén hozzáférést biztositson az XHELP alrendszereihez. Fő funkciói az alábbiak:

1.1. A fejlesztés azonositása

A Fejlesztői Adatszótárban (DD) egy táblázat, a PGMTABLA.TAB file tartalmazza a bejegyzett (megkezdett) fejlesztések legfontosabb információit. Ezek:

- a fejlesztés tábla-sorszáma
- a fejlesztés neve
- egyedi védelmi kulcsszó
- felhasználást jelző flag

A felhasználó csak olyan fejlesztéshez férhet hozzá, melyet ebbe a táblába már felvettünk. Az uj fejlesztések bejegyzése a rendszer-felelős feladata és joga.

1.2. Password-kezelés

A Fejlesztői Adatszótár, s ezen belül az egyes fejlesztések file-jainak védelmét védelmi kulcsszavak biztositják, az alábbi konvenció szerint:

- A rendszer-felelős password-je valamennyi DD-file-hoz és XHELP-funkcióhoz hozzáférni jogosit. Csak e kulcsszó mellett indithatók bizonyos segédprogramok (pl. uj fejlesztés bejegyzése, valamelyik fejlesztés törlése, stb.). Az egyedi kulcsszavakat is csak ő változtathatja.
- Minden fejlesztésnek saját password-je van. Ennek ismeretében, erre a fejlesztésre kérhetők az XHELP-funkciók, valamint a felhasználói segédprogramok (listázások a DD-file-okról).

1.3. XHELP-eljárások kérése

Ha a felhasználó azonositotta a fejlesztést és igazolta jogosultságát, kérheti a

- DESHELP
- SPECHELP
- PROGHELP
- TESZTHELP
- MAPHELP
- DOKHELP

eljárások valamelyikét.

Ezek a főfunkciók minden esetben további részfunkciók specifikálását kérik, ezt a kezelésnél ismertetjük.

A kérés-specifikálás eredményeként utasitásokat generálunk egy kommand file-ba, melyek a kért funkció task-jánál a DD-ből a felhasználó UIC-ja alá másolását, a task inditását, majd a file-ok törlését tartalmazzák.

A jelenlegi verzió, IAS alatt, nem teszi lehetővé a funkciók ciklikus kérését, mivel az egyes funkciók önálló task-okat alkotnak. A végleges változatban az XHELP valamennyi eljárása egyetlen programba kerül, melyet az RSX/Task Bilder overlay-lehetőségeit kihasználva mozgatunk a kérés-specifikáció függvényében.

2. Kezelés:

A Monitor az előbbi panel-lel jelentkezik be:

```
XHELP FEJLESZTOI RENDSZER                                    MONITOR

UDVOZOLJUK AZ XHELP KISERLETI UZEMBEN
                    * * . .          . . * *
KERJUK, HOGY MEGJEGYZESEIT AZ UZENET.TXT FILEBAN HAGYJA!!

MELYIK FEJLESZTESEN KIVAN DOLGOZNI? *****
```

A fejlesztés-névre egy maximum 30 karakter hosszu, ASCII string adandó meg.

Ha a válaszul adott név nincs a bejegyzett fejlesztések között, a reakció:

```
--- NINCS ILYEN FEJLESZTES
    HA UJ, HIVJA A DDUTIL-NEW KOMMANDOT
```

Bármi volt az előző kérdés eredménye, megjelenik a

```
PASSWORD? (A ZAROKARAKTER : $ !)
```

kérdés.

A password 10 karakteres, ugyancsak ASCII string. Elküldését nem a RETURN, hanem egy $ karakterrel végezzük. A begépelt jelsorozat nem jelenik meg a képernyőn.

Ha kulcsszavunk érvénytelen, a PASSWORD? kérdés további két esetben ismétlődik.

Jogosultság esetén a menüpanel indul.

```
XHELP FEJLESZTOI RENDSZER                                    MONITOR

MELYIK ALRENDSZERREL KIVAN DOLGOZNI?

  DES - DESHELP                          SPE - SPECHELP
  PRO - PROGHELP                         TES - TESZTHELP
  MAP - MAPHELP                          DOK - DOKHELP

                        END - END

USSE BE A KIVANT ALRENDSZER NEVET .....
```

A válaszok a jelölt háromkarakteres röviditések lehetnek. Ha a kapott válasz helytelen, az

```
<<<  ERVENYTELEN KARAKTEREK
```

hibajelzéssel ujra indul a panel.

Ha a választott alrendszer a

1. DESHELP - a bejelentkező panelja:

```
XHELP FEJLESZTOI RENDSZER                                    MONITOR

DESHELP ALRENDSZER

MELYIK ALFUNKCIOT KERI? ...

                EDI - EDITOR
                ELO - ELOTET
                DDF - DDFILL
```

Amennyiben a begépelt jelsorozat a megengedettek egyikével sem azonos, az

```
<<<  ERVENYTELEN KARAKTEREK
```

válasszal ujra bejön a DESHELP-panel.

2. SPECHELP - a bejelentkező panelja:

XHELP FEJLESZTOI RENDSZER DDUTIL

OPTION? ...
(VALASZ: NEW, PAS, LIS, PRO, DAT, TYP, CRF, DUM, COP)

(A röviditések magyarázatát lásd a Fejlesztői Adatszótár leirásának Segédprogramok fejezetében.)

A panel hibás karakterek esetén nem ismétlődik.

3. PROGHELP - megjelenik a

XHELP FEJLESZTOI RENDSZER MONITOR

PROGHELP ALRENDSZER

MELYIK FUNKCIOT KERI? ...

KON - KONVERZIO
GEN - GENERATOR

panel. Ha a begépeléskor a kezdőkarakterek nem K vagy G, a

<<< HIBAS KARAKTEREK

üzenettel ujraindul a panel.

4. DOKHELP - bejelentkezéskor a

XHELP FEJLESZTOI RENDSZER MONITOR

DOKHELP ALRENDSZER

MELYIK NYELVEN DOKUMENTAL? (PAS/PDL)...

MILYEN ALFUNKCIOT KER? ...

STR - STRUKTOUT
CRE - CREF
MOD - MODUL

panelt irja fel.

Hibás nyelv-válasz ill. alfunkció-begépelés esetén a

<<< HIBAS KARAKTEREK

üzenettel megismételjük a panelt.

A MAPHELP és TESZTHELP a jelen verzióban nem áll rendelkezésre.

A Monitor futtatását a

@XHELP

kommand-file biztositja, melynek tartalma a

RUN MONITOR
@W

parancssorozat.

Jelenleg a Monitor hivása előtt az XHELP-et alkotó valamennyi task-ot át kell másolni a DD-könyvtárból a felhasználói UIC alá. Erre a célra a CXHELP.CMD parancsfile használható. A kért funkció hivását megelőző másolások lassitanák és, gyakori ismétlések esetén, gazdaságtalanná tennék a futást.

A. függelék: A PDL konvenciók

PDL (Program Design Language)

1. Bevezetés

A követelmény-leirás és a programtervezés során általában két közelitésmódot használhatunk. Egyik esetben grafikus eszközökkel (HIPO ábrák, buborék ábrák, blokkdiagramok stb.) dolgozunk, másik esetben azonnal a programkódot használjuk. Természetesen ez utóbbinál a használt nyelv keretében biztositanunk kell a programstruktura tervezésének lehetőségét, hiszen végső kódot gyakorlatilag soha nem vagyunk képesek azonnal produkálni.

A nyelvi formalizmus kereteiben történő tervezésre jól beváltak a PDL nyelv különböző verziói. A PDL az általában használatos programnyelvek (ALGOL, PL/1 stb.) mintáját követi, de mig a szokványos programnyelvek explicit adatstrukturákon végeznek jól definiált müveleteket, a tervezési nyelv olyan általános utasitásokat tartalmazhat, mint pl.: "mátrix invertálása" vagy "a maximális fizetésü dolgozó kódszámának kikeresése".

A PDL nyelv két komponensü. A "külső szintaxis"-ba tartozó néhány utasitás kötött szerkezetü. Általában a program kontrollszerkezetét irják le, kulcsszavaik és felépitésük valamelyik nyelvből (PASCAL, PL/1,...) kölcsönözöttek.

A másik komponens lényegében definiálatlan szintaxisu kijelentésekből áll. Tetszés szerinti mondatok használhatók a végrehajtandó funkciók leirására. Például:

```
procedure GYÜJTŐ
GYÜJTŐ nullázása
read (INFILE, RECORD)
while not eof (INFILE) do
```

```
        a rekord kiirása OUTFILE-ra
        REKORDTAG kiválasztása
        GYÜJTŐ: = GYÜJTŐ + REKORDTAG
        read ujabb rekord (RECORD)
    endwhile
    endproc
```

A kötött szintaxisu programkeret a procedure, while, endwhile, endproc utasitásokból áll. Ezen a kereten belül használhatunk egyéb, kulcsszavas utasitásokat: read, értékadás, stb., vagy szabad formátumu szerkezeteket: "GYÜJTŐ nullázása", stb.

A PDL leirásban szereplő összes változót "deklarálni" kell. Ez azonban nem a PDL kontrollszerkezetével együtt, hanem egy külön, gép által vezérelt dialógusban, a DESHELP/DDFILL alrendszer révén történik.

Az adat-absztrakciót az XHELP 1. verziója formalizmussal nem támogatja.

A továbbiakban részletesen ismertetjük a PDL kötött szintaxisu keret-utasitásait. A leiráshoz használt jelölések:

< > jelpár közé zárjuk a nem elemi fogalmakat.

{ } jelpár közé zárt rész opcionális.

Általános szintaktikai szabály, hogy az utasitások sor-orientáltak, azaz hasonlóan a FORTRAN-hoz, a SORVÉGE zárja őket. Ha az utasitás nem fér el egy sorban, ugy a sor végén és a következő sor elején kitett elválasztó jel (-) használható folytatósor definiálására.

2. A PDL utasitások

Öt csoportot különböztetünk meg. Ezeket az alábbiakban ismertetjük. A feltétel-leirás és adatdefiniálás bővebben a 3. és 4. pontban kerül tárgyalásra.

2.1. Vagylagos utasitások

2.1.1. Kétirányu elágaztatás

```
if  <feltétel>  then
      <az igaz ág utasitásai>
{ else
      <a nem ág utasitásai>}
endif
```

A feltétel igazságtartalmától függően hajtódik végre a két ág valamelyike. Az else ág elmaradhat.

Pl.:

```
if  X>5  then
        cseréljük fel a sorrendet
else
        töröljünk minden második elemet
endif
```

vagy

```
if  not eof (inputfile) then
        rekordfeldolgozás
endif
```

2.1.2. Többirányu elágazás

```
case <kifejezés> of
&<prefix - 1>:
   <a prefix-1-hez tartozó utasitások>
&<prefix - 2>:
   <a prefix-2-höz tartozó utasitások>
   .
   .
   .
```

```
&⟨prefix-k⟩:
    ⟨a prefix-k-hoz tartozó utasitások⟩

& otherwise
    ⟨az egyébként végrehajtandó utasitások⟩

endcase
```

A case utasitás kiértékeli a kifejezést, majd végrehajtja azt az utasitáscsoportot, melynek konstans prefix-e a kifejezés értékével egyenlő. Ha a kifejezés egyik prefix-szel sem azonos, akkor az otherwise ág hajtódik végre.

Például:

```
    case  CH  of

  &'A':
    elemrész tárgykódgenerálás
  &'X':
    az input sor törlése
    listázás a hibalistán
  & otherwise
    továbbkeresés

    endcase
```

2.2. Ciklusutasitások

2.2.1. Ciklus, előzetes feltételvizsgálattal:

```
while  ⟨feltétel⟩  do
    ⟨a ciklustörzs utasitásai⟩
endwhile
```

2.2.2. Ciklus, késleltetett feltételvizsgálattal:

```
repeat
⟨a ciklustörzs utasitásai⟩
until  ⟨feltétel⟩
```

2.2.3. Számlált ciklus:

```
for ⟨változó⟩ := ⟨kifejezés-1⟩ to
    ⟨kifejezés-2⟩ by ⟨kifejezés-3⟩
  ⟨a ciklustörzs utasitásai⟩
endfor
```

A kifejezés-1, kifejezés-2 és a kifejezés-3 a ciklus kezdésekor értékelődnek ki, a ciklusparaméter kezdőértékét, végértékét és növekményét adják meg. Ha nincs növekmény, akkor ez +1-nek értelmeződik.

2.3. Procedura definició és hivás

2.3.1. Procedura definició

A PDL-ben általában egy funkcionális egységet egy procedura jelent. A procedura formája:

```
procedure ⟨proceduranev⟩ ( ⟨paraméterlista⟩ )
    ⟨a funkciót leiró utasitások⟩
endproc
```

A proceduranév és a procedura informális leirása (ld. "aktiv comment"-ek, a 2.3.2-ben!) között szoros összefüggés kell legyen a név a leirás alfabetikus karaktereiből képződik, sor végéig vagy zárójelig. A PDP implementáció során a string a későbbiekben csupán 9 szignifikáns karakterrel értelmeződik (pl. filenév kreálásnál!).

Az XHELP-ben minden procedura önálló egység. Nincs tehát értelme a skatulyázott proceduráknak. Ezt a PASCAL-hoz képest jelentős változtatást módszertanilag a skatulyázásból eredő implicit adatcsere kizárási igénye indokolja.

Az XHELP-ben nincsen főprogram. A legfelső szintü funkcionális modul is procedura; majd a programgenerálás során kap keretet.

A procedurák lehetnek funkcionálisan kifejtettek vagy black-box-ok, vagyis olyanok, melyekről csak a környezeti elvárások ismertek, konkrét müködésük még tisztázatlan. A black-box rutinokat a procedura-kereten belül a

dummy (< I/O környezet leirása >)

procedura-törzs definiálja.

Pl.:

```
procedure nemtudommi (x, y)
          dummy (z, k)
endproc
```

2.3.2. Procedura hivás

Egy PDL leirás sok szabad formátumu utasitást tartalmaz. Közülük aktiv comment-nek nevezzük azon utasitásokat, melyek valamilyen tényleges funkciót jelölnek, informális leirással. Ezek később procedurahivássá vagy beépitett utasitásokká alakulnak át:

- az aktiv comment hossza tetszőleges lehet, de vigyázat az előző pontban leirt névmeghatározási algoritmusra!
- az aktiv comment az adathivatkozásait " jel (idézőjel) karakterrel kezdi, vagy zárójelben, paraméterlista formájában szerepelteti. (ld. 3. pont!)

Az aktiv comment egy szabad formátumu procedura-hivás. A gép proceduranyilvántartásaiban egy röviditett névvel, 9 karakterrel szerepel.

A passziv comment-ek olyan magyarázó szövegek, melyeket a gép figyelmen kivül hagy. Mindig felkiáltójellel kezdődnek s sor végéig tartanak.

Az XHELP speciális procedura hivásai:

a) A MAPHELP révén önállóan definiált képernyőkezelő programrészletek hivása:

map <mapnev> (< paraméterlista >)

A mapnev-vel jelzett map-procedura hivódik a paraméterlistá-ban megadott aktuális paraméterekkel. A paraméterek a képernyőn kiiródó ill. beolvasott változók.

b) Verifikálási pont kijelölése:

assert if <feltétel> then
<assertnev> (<paraméterlista>)

Az assert kulcsszóval jelölt helyen a feltétel bekövetkezése esetén a programfutás megszakad s a paraméterlistában megadott környezeti változók verifikálhatók. Erről bővebbet ld. a TESZTHELP alrendszer funkciójának leírásánál.

2.4. Értékadás

Formája a szokásos, a := jelet használva.
A : jel elmaradása hibát nem okoz.

3. Adatdefiniciók:

A PDL teljesen kötetlen belső formalizmusa miatt az XHELP számára külön jelölni kell a változóneveket. Erre eddig a leírásban a nagybetüt használtuk. Ez nem mindig lehetséges, igy az alábbi konvenciót vezetjük be:

A PDL leirás adatneveit mindig " (idézőjel) karakterrel kell kezdeni, hacsak nem paraméterlistában adjuk meg. Zárójelen belüli előfordulásnál az idézőjel nem kell.

Pl.:

```
. . .
"x := " összeg tömb utolsó eleme
olvassunk be egy rekordot (név, fizetés)
```

```
if "fizetés < 3000 then
      "név letárolása az "y tömbben
. . .
```

Az XHELP adatszótárába csak a kijelölt változók kerülnek be.

4. Feltétel leirás:

Az XHELP-ben function tipusu modul nincs. Igy a vezérlő utasitásokban nem szerepelhet kötetlen formátumu leirás.

Helyesek pl.:

```
if  "x < 5   and   "y > 6   then
                ...
vanutolsoelem  (van)
while "van  do
        ...
endwhile
endif
```

Látható, hogy egy procedurával és egy boolean változóval a kötetlen formátumu feltétel-leirás lényegében megvalósitható.

5. Adatstruktura

Az XHELP-ben használt PDL a PASCAL nyelv felé orientált. Ennek megfelelően a tervezéskor a PASCAL nyelv adatstruktúráiból indulhatunk ki, a tipusok megadásánál is PASCAL konvenciók érvényesek (ld. a DDFILL leirást!)

Lényeges, hogy a skatulyázás megszüntetésével lényegében megszüntek a nem abszolut globális adatok. Az XHELP procedurák közti adatcsere tehát vagy abszolut globális szinten, vagy paraméterként lehetséges.

Tipust csak globálisan, proceduráktól független érvénnyel definiálhatunk.

B. FUGGELEK

1.0 AZ EDITOR PARANCSOK

EBBEN A FEJEZETBEN AZ ED parancsait irjuk le, alfabetikus sorrendben. Az ismertetett parancsok:

- APPEND - hozzafuzes sorhoz
- BOTTOM - pozicionalas az utolso sorra
- CHANGE - karakterstring valtoztatas a soron belul
- DELETE - sorok elhagyasa
- DF - strukturakeret elhagyasa (delete frame)
- DS - strukturablokk elhagyasa (delete structure)
- END - kilepes az editorbol
- INSERT - uj sorok begepelese
- KILL - editor abortalasa
- LOCATE - string kereses bufferen belul
- NEXT - sorpozicionalas relativ sorszam alapjan
- PRINT - sorok kiiratasa
- SI - strukturadiagramos mod bekapcsolasa
- SO - strukturadiagramos mod kikapcsolasa
- TOP - sorpozicionalas a buffer elejere
- (CR) - sorlepes elore
- (/) - sorlepes hatra
- (.) - abszolut sorszam kiirasa
- (n) - sor kiirasa abszolut sorszam szerint.

Minden parancsnal megadjuk a formatumat, a funkciojat, kiemelt jelentosege miatt a sormutatora gyakorolt hatasat es egy peldat. A peldak alapja az alabbi program, amit elozoleg mar beeditaltunk a gepbe a show.xx filenev alatt.

```
C N SZAM OSSZEGENEK KISZAMITASA
      READ(5,10)N
10    FORMAT(I2)
      S=0.
      DO 20 I=1,N
      READ(5,25) X
25    FORMAT(F8.2)
      S=S+X
20    CONTINUE
      WRITE(6,30)S
30    FORMAT(' A SZAMOK OSSZEGE:',F10.2)
      END
```

1.1 AZ APPEND parancs

Formatuma:

*APPEND <szoveg>

A parancs egy vagy tobb karakterre roviditheto.

Funkcioja:

A <szoveg>-ben szereplo karaktereket az aktualis sor vegere irja.

A sormutato:

A sormutato valtozatlan marad.

Pelda:

Az aktualis sor:

C N SZAM OSSZEGENEK KISZAMITASA

A parancs:

*APPEND S-BEN

Az aktualis sor uj tartalma:

C N SZAM OSSZEGENEK KISZAMITASA S-BEN

1.2 A BOTTOM parancs

Formatum:

*BOTTOM

A parancs neve egy vagy tobb karakterre roviditheto

Funkcioja:

A sormutatot a buffer utolso feltoltott sorara allitja.

A sormutato:

Az utolso sor sorszamat veszi fel.

Pelda:

A parancs:

```
        *B
```

Kiirasra kerul:

```
??????? EOB ???????
```

Az aktualis sor az utolso lesz.

1.3 A CHANGE parancs

Formatuma:

```
            *CHANGE/<szoveg1>/<szoveg2>
```

A parancs neve egy vagy tobb karakterre roviditheto

Funkcioja:

Az aktualis sorban szoveg1-et kicsereli szoveg2-re.

A sormutato:

Valtozatlan marad.

Pelda:

Az aktualis sor:

```
                READ(5,10)N
```

A parancs:

```
                *C/10/1,END=99
```

Az uj aktualis sor:

```
                READ(5,1,END=99)N
```

1.4 A DELETE parancs

Formatuma:

```
        *DELETE (n)
```

ahol n egy pozitiv egesz szam.
A parancs neve egy vagy tobb karakterre roviditheto.

Funkcioja:

A parancssal szovegsorokat hagyhatunk el, az alabbi modon:
1.Ha n pozitiv,akkor az aktualis sor es meg n-1 sor hagyodik el.
   Ha elobb van a buffer vege, akkor addig elhagyodik valahany sor.
2. Ha nem adjuk meg az elhagyando sorok szamat, a gep az aktualis
   sort hagyja el.

A sormutato:

Ha az elhagyas a buffer vegeig terjedt,akkor a sormutato az utolso
megmaradt sorra all.

Egyebkent az aktualis sor az utolso elhagyott sor utani sor lesz.

Pelda:

Az aktualis sor es meg 3 sor elhagyasara az alabbi parancs hasznalha

```
     *DELETE 4
```

1.5 A DF (Delete Frame) parancs

++

Formatuma:

```
       *DF
```

a parancs formaja kotott.

Funkcioja:

Az aktualis sort tartalmazo legbelso struktura keretet,
azaz a blokk-kezdo es zaro utasitasokat torli.

A sormutato:

Ha az aktualis sor elmarad, akkor az elozo sorra mutat,
egyebkent marad az aktualis soron.

1.6 A DS (Delete Structure) parancs

Formatuma:

```
*DS
```

A parancs formaja kotott.

Funkcioja:

Az aktualis sort tartalmazo legbelsobb blokkot torli.

A sormutato:

A torolt blokk elotti utolso sorra mutat.

1.7 Az END parancs

Formatuma:

```
*END
```

A parancs neve NEM roviditheto le!

Funkcioja:

A buffer tartalmat kiuriti az output file-ra, majd az input file meg fel nem dolgozott lapjait is atirja az output file-ra. Az input file valtozatlan marad.
Az ED az >>>> EXIT >>>> szoveg kiirasa utan befejezodik.
Ujra rendszerparancsot var a gep.

A sormutato:

Az editornak vege, tehat a sormutatonak nincs tovabb szerepe.

Pelda:

```
*END
>>>> EXIT >>>>
```

1.8 AZ INSERT parancs

Formatuma:

```
        *INSERT  (CR)
```

A parancs neve egy vagy tobb karakterre roviditheto.

Funkcioja:

Az EDITOR atter input modba, ket (CR) leutesig minden beutott karakter bekerul a szovegfile-ba.

A sormutato:

A beutott sorok szamaval novekszik.

Pelda:

```
        *I                          (CR)
        EZ A BEUTOTT UJ SOR  (CR)
        EZ A MASODIK         (CR)
                            (CR)
        * ...
```

Itt mar ujra editor modban vagyunk.

1.9 A KILL parancs

Formatuma:

```
        *KILL
```

A parancs neve NEM roviditheto le!

Funkcioja:

Az editor abortalasa.
Az ED a >>>> EXIT >>>> szoveg kiirasa utan befejezodik, ujra rendszerparancsot var a gep.
Output file nincs.

A sormutato:

Az editornak vege, tehat a sormutatonak nincs tovabb szerepe.

Pelda:

```
*KILL
>>>> EXIT >>>>
```

1.10 A LOCATE parancs

++++++++++++++++++++

Formatuma:

```
*LOCATE <szoveg>
```

A parancs neve egy vagy tobb karakterre roviditheto.

Funkcioja:

Megkeresi az adott szoveg elso elofordulasi helyet a bufferen belul, es kiirja a szoveget tartalmazo sort.
Ha nincs a bufferen belul ilyen szoveg, akkor a lap aljara all.

A sormutato:

A szoveget tartalmazo sorra mutat, ha a szoveg letezik es a buffer utolso sorara, ha nem letezik.

Pelda:

Az aktualis sor a masodik

```
*L OSSZEG
 30     FORMAT(' A SZAMOK OSSZEGE:',F10.2)
*...
```

1.11 A NEXT PARANCS

++++++++++++++++++

Formatuma:

```
*NEXT (n)
```

Ahol n egy pozitiv vagy negativ egesz szam.
A parancs egy vagy tobb karakterre roviditheto.

Funkcioja:

Ha n-t elhagyjuk, akkor a parancs eredmenyekent a sormutato egy sor-

ral elore allitodik.

Ha n letezik, a sormutato n sorral elore mozdul, ha n pozitiv, es n sorral hatra, ha n negativ.

Az uj aktualis sor kiirodik.

Ha n akkora, hogy kifutunk a bufferbol, akkor a BOTTOM vagy a TOP parancsokkal ekvivalens hatasu a parancs.

Pelda:

A masodik sornal allva a

```
*N 2
```

parancs a negyedik sorra pozicional es kiirja annak tartalmat.

1.12 A PRINT parancs

++++++++++++++++++++

Formatuma:

```
*PRINT  (n)
```

ahol n egy pozitiv szam.

Funkcioja:

Kiirja a kovetkezo n sort.

A sormutato:

Valtozatlan marad.

Pelda:

A sormutato a lap tetejen all;

```
*P 3
C N SZAM OSSZEGENEK KISZAMITASA
      READ(5,10)N
*...
```

1.13

Az SI (Structured mode In) parancs
++

Formatuma:
----

*Si

A parancs kotott formatumu.

Funkcioja:

A kepernyon a tovabbiakban minden sor a struktura-
diagramos abrazolassal egyutt jelenik meg.

A sormutato:

Nem valtozik.

1.14 AZ SO (Structured mode Out) parancs

++++++++++++++++++++++++

Formatuma:
----

*SO

A parancs kotott formatumu.

Funkcioja:

Visszakapcsol strukturadiagram nelkuli modba.

1.15 A TOP PARANCS

++++++++++++++++++++

Formatuma:
----

*TOP

A parancs neve egy vagy tobb karakterre roviditheto.

Funkcioja:

A sormutatot a buffer elejere, az elso sor ele viszi. A

++++ TOB ++++

kiiras jelzi a funkcio megtortentet.

Pelda:

```
        *T
        ++++ TOB ++++
```

1.16 SORLEPES ELORE

++++++++++++++++++++

Formatuma:

```
        *       (CR)
```

azaz elso pozicion megadott carridge return (=ures sor).

Funkcioja:
----

a sormutatot eggyel elorebb viszi. Az uj sor kiirodik. Nem mozdul a mutato, ha a buffer vegen vagyunk.

Pelda:
------

az aktualis sor a masodik.
Az ures sor hatasara a harmadik sorra allitodik a mutato es ez ki is irodik.

1.17 SORLEPES HATRA

++++++++++++++++++++

Formatuma:

```
        */
```

azaz egy slash karakterbol allo sor.

Funkcioja:

Egy sorral hatrabb allitja az aktualis sort. Az uj sor ki is irodik. Nem allitodik a sormutato,ha a buffer elejen vagyunk. Ilyenkor a

```
        ++++ TOB ++++
```

kiiras jelenik meg.

Pelda:

Az aktualis sor a negyedik.A / sor hatasara az aktualis sor a harmadi
lesz, ez ki is irodik.

1.18 ABSZOLUT SORSZAM KIIRASA

Formatuma:

```
        *.   (CR)
```

azaz egy pont karakterbol allo sor.

Funkcioja:

Kiirja az aktualis sor bufferbeli abszolut sorszamat.

Pelda:

A sormutato a harmadik soron all.
A .(cr) parancs hatasara kiirodik a

```
        3
```

A sormutato valtozatlan marad.

1.19 SOR KIIRASA ABSZOLUT SORSZAM ALAPJAN

Formatuma:

```
        (n)
```

azaz egy pozitiv szam.

Funkcioja:

A sormutato felveszi a bufferben a megadott abszolut sorszammal jelzet
sor sorszamat. Ha n tulmutat a bufferen, akkor a bottom funkcio hajlo-
dik vegre.

Pelda:

A

```
        *4
```

parancs a sormutatot a negyedik sorra allitja es kiiratja a sort.