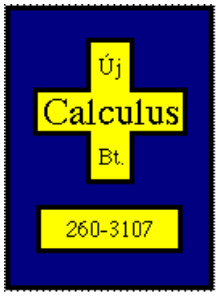
SAS PROGRAMOZÁS

Az angliai AMADEUS Software Training, UK engedélyével, annak eredeti anyagát felhasználva készítette az Új Calculus Számítógép alkalmazási Bt.

2000. március

Új Calculus Számítógép alkalmazási Bt.



Levelezési cím: 1475. Budapest Pf. 184.
e-mail: office@calculus.hu **mobil: 06-209-350-645**

TARTALOMJEGYZÉK

P1 FEJEZET

BEVEZETÉS

1.1 MI A SAS RENDSZER?

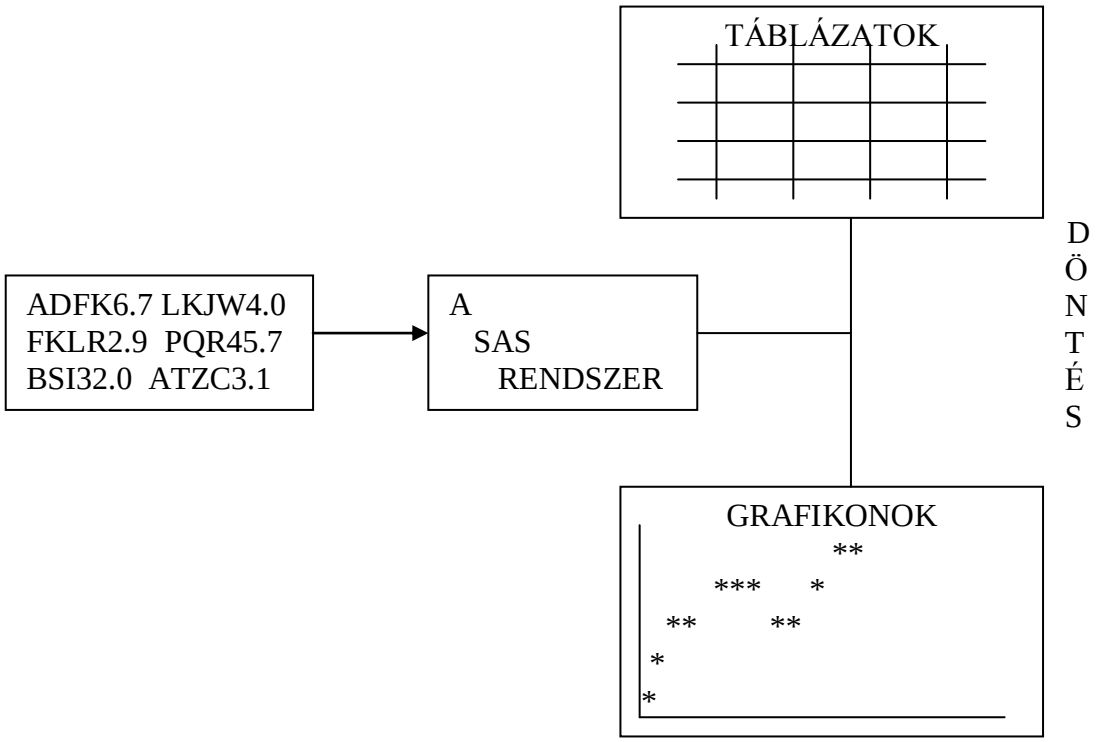
1.2 A SAS RENDSZER ELEMEI

1.3 ALAPELVEK

P1.1. MI A SAS RENDSZER?

EGY ADATELEMZŐ ESZKÖZ

A SAS rendszer segítségével külső adatokat véve végre tudjuk hajtani a kívánt elemzést. Végül az eredményt elő tudjuk állítani táblázatban vagy grafikonban.



ADAT → INFORMÁCIÓ → TUDÁS

Statistical Analysis System
(STATISZTIKAI ELEMZŐ RENDSZER)

Simple And Sophisticated

MIT IS CSINÁL A SAS RENDSZER?

ADATOKAT OLVAS
RIPORTOKAT KÉSZÍT
ADATOKAT TÁROL
REGRESSZIÓS ANALÍZIST VÉGEZ
TÁBLÁZATOKAT KÉSZÍT
KOORDINÁTARENDSZERT RAJZOL
OSZLOPGRAFIKONOKAT RAJZOL
TÉRKÉPEKET KÉSZÍT
VARIANCIAANALÍZIST VÉGEZ
ÁTLAGOT SZÁMOL
GYAKORISÁGI TÁBLÁZATOKAT ÁLLÍT ELŐ
ADATLISTÁKAT ÍR
EGYVÁLTOZÓS STATISZTIKÁKAT SZÁMOL
MATEMATIKAI EGYENLETEKET OLD MEG
POSZTERT RAJZOL
LEVELEZÉSI LISTÁT KÉSZÍT
KEZELI A DÁTUMOT ÉS AZ IDŐT
VEZÉRLI A PRINTEREKET ÉS A PLOTTEREKET
TÖBBVÁLTOZÓS ANALÍZIST VÉGEZ
LEHETŐSÉGET AD MÁSIK GÉPHEZ VALÓ KAPCSOLÓDÁSRA
OPERÁCIÓKUTATÁST VÉGEZ
JELZÁLOG-ARÁNYOKAT SZÁMOL
ADATBÁZIS-ELÉRÉST TESZ LEHETŐVÉ
MÁTRIX PROGRAMOZÁST HAJT VÉGRE
ADATKONVERZIÓT VÉGEZ
ELEMZI AZ ADATOKAT
ADATRÖGZÍTŐ RENDSZEREKET HOZ LÉTRE
MENÜ-RENDSZEREKET ÉPÍT FEL
TANFOLYAMOKAT LÉTESÍT
NAGYON HATÉKONY ADATKEZELŐ NYELVE VAN

RÉSZLETESEBBEN:

A SAS rendszer birtokában manapság a világon az egyik legnagyobb, legszélesebb körű, legrugalmasabb adatkezelő rendszert tudhatjuk magunkénak.

A SAS rendszer teljesen egységes szoftvermodulokból áll, mindegyik ugyanolyan módon működik nagyon sokféle hardver eszközön s a legnépszerűbb szoftver környezetekben.

A SAS rendszer használható akkor is, amikor kilépünk egy adatbázisból, bár a SAS-nak meg vannak a lehetőségei egy saját adatbázis kezelő rendszerhez. Ahogy az elnevezés is sugallja, az adatbázis a külső adatok egyszerű raktára, amely tartalmazhat bármit, egy szervezet alkalmazottait, kísérletek eredményeit, számítógép teljesítményadatait, vagy gépészeti specifikációkat.

A SAS rendszer nagyon sokféle formában tudja beolvasni az adatokat – mint egyszerű fájlokat, hierarchikus vagy flat fájlokat. A SAS-ban tárolt adatok egyszerűen és gyorsan elérhetők a beépített eljárások és ablakok használatával.

Az adatkezelési lehetőségek segítségével a programozók elérhetik adataikat, sorokat és oszlopokat cserélhetnek ki vagy módosíthatnak, új fájlokat hozhatnak létre és adatkönyvtárakat kezelhetnek.

Az elemzési lehetőségek használatával a felhasználó statisztikákat számolhat, táblázatokat és riportokat készíthet az adataiból, vagy grafikonokat generálhat a képernyőre vagy papírra.

További lehetőségekkel a programozók automatizált rendszereket létesíthetnek, amelyekkel a gyakorlatlan felhasználók is elérhetik a SAS rendszert, lekérdezhetnek adatokat, riportokat gyárthatnak és grafikonokat rajzolhatnak ki.

P1.2 A SAS RENDSZER ELEMEI

A TERMÉKEK CSOPORTOSÍTÁSA

ALAPVETŐ FELHASZNÁLÁS

- Base SAS szoftver
- SAS/FSP szoftver
- SAS/AF szoftver
- SAS/ASSIST szoftver
- SAS/GRAPH szoftver

STATISZTIKÁK

- SAS/STAT szoftver
- SAS/INSIGHT szoftver
- SAS/ETS szoftver
- SAS/OR szoftver
- SAS/QC szoftver
- SAS/IML szoftver

ELÉRÉS ÉS ADATTÁRHÁZAK

- SAS/ACCESS interfaces
- SAS/MDDB szoftver
- SAS/SHARE szoftver
- SAS/CONNECT szoftver
- SAS/WAREHOUSE adminisztrátor

ANALÍZIS

- SAS/CPE szoftver
- MXG szoftver
- SAS/CALC szoftver
- SAS/LAB szoftver

ALKALMAZÁSOK

- SAS/EIS szoftver
- SAS/INTRNET szoftver
- SAS/DMI szoftver

UTILITY

- SAS/IML szoftver
- SAS/TOOLKIT szoftver
- SAS/SHARE szoftver
- SAS/ENTERPRICE REPORT szoftver

TERMÉKLEÍRÁSOK

A SAS rendszer a BASE SAS szoftverből és ehhez csatlakozó egyéb, szabadon megválasztható egységekből épül fel.

BASE SAS szoftver

DATA lépés

Riportok készítése

Leíró statisztikák

A SAS rendszer szíve, nélküle egyetlen más SAS termék sem használható. Ez az alaptermék tartalmaz egy „adatkezelési nyelvet" egy magas szintű programozási nyelvet, amely külső adatállományokat olvas be és amely SAS adatállományokat tud létrehozni, valamint néhány alapstatisztikákat számoló eljárást. Hozzátartozik még egy „programgeneráló nyelv" is, - a MACRO nyelv – amelyet SAS utasítások generálására, feltételhez kötött végrehajtásra és SAS rutinok beillesztésére használjuk.

MACRO nyelv

Adatkezelés

Adatfájl beolvasása

SAS/STAT szoftver

Többváltozós regresszió

A legegyszerűbbtől a legbonyolultabb többváltozós statisztikai elemzésig használható.

Faktoranalízis

SAS/INSIGHT szoftver

Adatmegjelenítés

Az interaktív adatkezelésre szolgáló termék lehetőséget biztosít adataink háromdimenziós ábrázolására, valamint valós idejű modellezésre.

Interaktív elemzés

SAS/ETS szoftver

Előrejelzés

Különböző lineáris és nem lineáris módszerek felhasználásával ökonometriai és idősoros modellezésre és előrejelzésre szolgál.

Modellezés

SAS/FSP szoftver

Adatbeviteli rendszerek

Teljes képernyős adatbeviteli rendszerek definiálására, szövegszerkesztésre, levélcímzésre és írásra alkalmas.

Szövegszerkesztés

SAS/CALC szoftver

Adatmegtekintés

Egy teljes táblázatkezelő szoftver.

Képernyők tervezése

SAS/AF szoftver

Olyan alkalmazások fejlesztésére alkalmas termék, amely képes a végfelhasználó választásainak megfelelő programrészeket futtatni.

Alkalmazások írása

Adatbeviteli rendszerek

Menüvezérelt rendszerek

CBT

EIS

SAS/QC szoftver

Shewhart diagram

Képességelemzés

A termék statisztikai minőségellenőrzéshez tartalmaz módszereket, amelyeket széles körben alkalmaznak statisztikai folyamatvezérlési környezetben (SPC). Tartalmaz eljárásokat kísérlettervezés, statisztikai folyamatvezérlés, képességelemzés és mintavételi terv kiértékelés témaköréből is.

Kumulálás

Mozgó átlagok

SAS/GRAPH szoftver

Függvényábrázolás

Háromdimenziós képek

Térképek

A világ elsőszámú grafikus szoftvere, amely az adatábrázolás minden típusát tudja, képes katalógusba menteni és újra megjeleníteni az elkészült képet, lehetővé teszi az ábrák feliratozását a felhasználó által megadott szövegekkel és grafikus jelekkel. Lehetőséget biztosít térképek, háromdimenziós ábrák, kontúrok rajzolására adatállomány alapján. Ma már egy igazi, interaktív módon használható grafikus szoftver.

Oszlopdiagramok

Újrarajzolás

Feliratozás

SAS/OR szoftver

Lineáris programozás

Az operációkutatás technikáin alapuló eljárások sorozata, többek között lineáris programozás, kritikus utak módszere, Gantt diagram készítés, háló problémák peremfeltételekkel és anélkül.

Gráfelmélet

SAS/ACCESS szoftver

Nézetek

Adatbázis lekérdezés

Terméksorozat, amely lehetővé teszi különböző adatbázis kezelő rendszerek elérését, pl. DB2, SQL-DS, Oracle, Rdb/VMS, Ingres

SAS/IML szoftver

Mátrixalgebra

Szubrutinok írása

Mátrixok kezelését lehetővé tevő termék. A nyelv mátrixalgebrai jelöléseket használ, így a matematikai formulák könnyen átírhatók SAS/IML szintaxisba.

SAS/ASSIST szoftver

Egyszerű menürendszer

A SAS rendszer tanulása

A SAS egészét egy, a többitől teljesen eltérő módon használhatjuk a SAS/ASSIST segítségével. Ez egy felhasználóbarát kapcsolatteremtő eszköz a SAS és a végfelhasználó között, egy menüvezérelt szoftver, amely meggenerálja és lefuttatja a programot, az eredményt pedig egy másik képernyőn mutatja meg. Könnyen lehet a SAS programozás elemeit oktatni a segítségével, mivel a generált programokat mintaként használhatjuk újabb elemzések elkészítésekor.

SAS kódok generálása

Vezérlés az egérrel

Alkalmazások írása

Programgyűjtemény

SAS/MDDB server szoftver

Adattárház

Többdimenziós

Adattárház építésére és elérésére szolgáló rendszer. A SAS/Warehouse adminisztrátorral együtt oldja meg a komplett feladatot.

SAS/SHARE szoftver

Osztott könyvtárak

E termék lehetővé teszi, hogy egyszerre több felhasználó is hozzáférjen ugyanahhoz a SAS könyvtárhoz, memberhez, blokkhoz vagy megfigyeléshez.

Több felh. elérés

SAS/CPE szoftver

Teljesítménykövető

Számítógép teljesítmény kiértékelő termék.

Rendszer hangolás

SAS/CONNECT szoftver

Egyenrangú kapcsolat

Lehetővé teszi adatok és feldolgozások megosztását különböző számítógépeken futó SAS rendszerek között. E termékkel soros állományok, SAS adatállományok és katalógusok telepíthetők egyik rendszerből a másikba, valamint az egyik gépen megírt SAS program végrehajtható úgy, hogy outputunkat az eredeti lokális gépen kapjuk meg.

Fájlok átvitele

Gépek közti választás

Grafikák átvitele

SAS/CBT szoftver

Számítógépes oktatás

A SAS különböző termékeinek alapjait oktató programok készültek ezzel a termékkel.

On line oktató program

SAS/EIS szoftver

Vezetői információ

Ez a termék a vezetés különböző szintjeinek szolgáltat információt különböző részletességgel.

Kül. részletességű elem.

SAS/CALC szoftver

Táblázatkezelő

Táblázatkezelő termék (spreadsheet).

Cella és progr. műv.

SAS/LAB szoftver

Táblázatkezelő adatbevitelhez

Tudományos adatelemző és adatbevivő termék.

Elemzések

SAS/TOOLKIT szoftver

Programnyelvek

Szoftver, amely lehetővé teszi újabb input és output formátumok, függvények és eljárások írását több programnyelven is.

Szabványos függvények

SAS/INTRNET szoftver

SAS szerver

Internetes alkalmazások segítségével a SAS szolgáltatások elérése.

Kliens SAS nélkül

SAS/Enterprice Reporter szoftver

Kiadványszerkesztés

Számítógépes kiadványszerkesztés és SAS outputok integrálása a kiadványokba.

Fotó-kész dokumentumok

P1.3 ALAPELVEK

Szerencsénk van; a SAS rendszer, mint a legtöbb jó rendszer, hihetetlenül egyszerű; és akárcsak a legtöbb jó rendszer, nagyon bonyolult tud lenni. Míg a SAS eredetileg a 'Statistical Analysis System' (Statisztikai Elemző Rendszer) kezdőbetűit jelentette, most azt mondhatjuk, hogy a 'Simple And Sophisticated' (Egyszerű és bonyolult) kifejezésből alkotott betűszó.

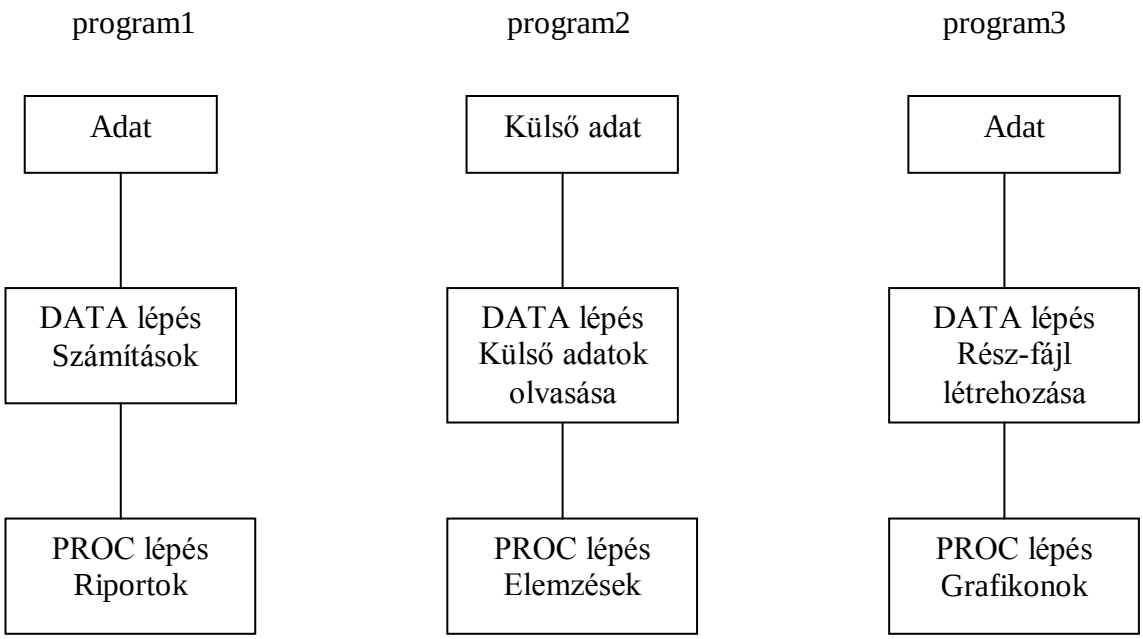
DATA STEP

A **DATA** lépés: Ez a lépés hozza létre a SAS adatállományokat, kezeli az adatokat, észleli az elágazásokat és ciklusokat. A DATA step tetszőleges formában tud külső adatállományt olvasni, és abból így egy vagy több SAS adatállományt, vagy egy, vagy több külső (nem SAS) fájlt létrehozni. A DATA step az a lépés, ahol programozhatunk, és ahol az algoritmust specifikálnunk kell.

PROC STEP

A **PROC** lépés: Az eljárások előre megírt rutinok. Néhány egyszerű utasítással készíthetünk riportokat, statisztikákat, összegzéseket, grafikákat, előrejelzéseket, hálózati munkákat és sok egyebet. Minden eljárásnál vagy PROC-nál az a lényeg, hogy SAS adat állomány az inputja.

A PROGRAM FOLYAMATA



TEHÁT AZ ESEMÉNYEK NORMÁL SORRENDJE:

- Olvassuk be a külső állományunkat egy SAS adatállományba a DATA lépésben.
- Hajtsuk végre néhány műveletet a DATA lépésben – például számítsuk ki egy új értéket, vagy válasszuk ki az input adatok egy részhalmazát.
- Hívjunk meg egy vagy több eljárást (PROC) az előbb létrehozott SAS adatállományt használva.

KIEGÉSZÍTÉS

A DATA lépés a SAS alapszoftver (BASE SAS) része, a többi SAS termék tartalmazza az eljárásokat (PROC), amelyek bármikor használhatók:

BASE SAS
DATA lépés

BASE SAS
PROC PRINT
PROC TABULATE
PROC MEANS
PROC CONTENTS
PROC FREQ
PROC COPY
PROC V5TOV6
PROC PMENU
SAS/FSP
PROC FSEDIT
PROC FSLIST
PROC FSLETTER
PROC FSVIEW
SAS/AF
PROC BUILD
SAS/GRAPH
PROC GCHART
PROC GPLOT

SAS PROGRAMOK ÍRÁSA

A SAS programozásnak van néhány szabálya

- utasítások írására
- állományok és változók elnevezésére
- a nyelv írására (szintaxis).

Néhány tanács SAS programok írásához:

- Nagyon figyelmesen írjuk a programot, a példákat pontosan másoljuk le.

- A programírásnál ízlés szerint váltogathatjuk a kis- és nagybetűket, ám az adatok kezelésénél a SAS különbséget tesz közöttük, ezért mindig a pontos értékhez hasonlítsunk.

- Kulcsszavakat, a változók neveit bátran írhatjuk nagybetűvel is, de ha pl a SZOVEG nevű változónk tartalma ***'ABCDE'*** , hasonlításnál azt soha nem fogja a SAS az ***'abcde'*** karaktersorozattal egyenlőnek találni.

- Nagyon ügyeljünk az idézőjelek ('aposztrófok' és „macskakörmök") használatára, ellenőrizzük, hogy megvan-e mindegyiknek a párja.

SAS UTASÍTÁSOK SZABÁLYAI

1. SZABÁLY

KULCSSZÓVAL KEZDJÜK

A kulcsszavak SAS utasítások nevei – és mivel a programnyelv szabad formátumú, kell egy utasítás-elválasztó jel:

2. SZABÁLY

PONTOSVESSZŐVEL ZÁRJUK

Az utasítások közötti határoló jel a pontosvessző, nem pedig egy sor szöveg jelent egy utasítást.
Egy SAS utasítás többsoros is lehet, bárhol kezdődhet és végződhet, ugyanakkor egy sorba több utasítást is írhatunk. A szavakat szóközöknek (blank) kell elválasztani.

KULCSSZAVAK:

Írjuk be a kulcsszavakat:

```
data work.nevek ;
     input nev $ kor ;
datalines ;
béla    33
edit     47
proc print ;
run ;
```

_____

_____

_____

_____

___

PONTOSVESSZŐ:

```
data work.nevek ;
      input nev $ kor ;
datalines ;
béla   33
edit    47
proc print
run ;
```

Hol végződik a **DATA** utasítás?

Végrehajtódik-e a **PROC PRINT** lépés?

SAS NEVEK SZABÁLYAI

A SAS nevek SAS állományneveket és SAS változóneveket jelentenek.

Két szabály van:

1. NÉVSZABÁLY

KEZDJÜK	**BETŰVEL**
VAGY	**ALÁHÚZÁSJELLEL**
FOLYTASSUK	**BETŰVEL**
VAGY	**ALÁHÚZÁSJELLEL**
VAGY	**SZÁMMAL**

2. NÉVSZABÁLY

6.12-ES VERZIÓIG LEGFELJEBB 8 KARAKTER HOSSZÚ LEHET,
A 7.0-TÓL MAXIMUM 32 KARAKTER

Példák SAS állománynevekre és változónevekre:

Kati	Valt4	Kek_toll	Ideiglen
Robert	Adat	Uj_auto	Arany
Teszt	Print	B_31_57	Sajatom
Program1	Minta	XXXXXXXX	Valami
Harminc	AAA	A8767648	Cikk

SAS PROGRAMOK SZABÁLYAI

Csak egy szabály van:

1. PROGRAMSZABÁLY

A SZAVAKAT **SZÓKÖZÖK**NEK KELL ELVÁLASZTANIUK

Például:

Helyes:

```
data frici ;
       set betegek ;
       if nem='N' ;
run ;
proc print ;
run ;
```

Hibás:

```
datafrici;
       setbetegek ;
       ifnem='N' ;
run ;
procprint ;
run ;
```

Figyeljünk a kulcsszavakra, pontosvesszőkre, lépésekre, nevekre és szóközökre.

```
data work.nevek ;
      input nev $ kor ;
datalines ;
béla   33
edit    47
proc print ;
run ;
```

Nincs pontosvessző az adatsorokban!

A fentebb említett kötöttségektől eltekintve a programírás meglehetősen szabad formájú. Ez lehetővé teszi, hogy kiválasszuk saját stílusunkat.

SAS ADATÁLLOMÁNYOK

Ha egy programot megírtunk, küldjük el a SAS-nak (**SUBMIT**):

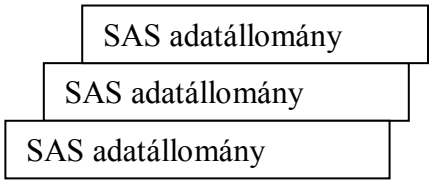
```
PROGRAM EDITOR
Command ==> submit

00001  data work.nevek ;
00002        input nev $ kor ;
00003  datalines ;
00004  béla   33
00005  edit    47
00006  proc print ;
00007  run ;
```

Az outputot az **OUTPUT** ablakban nézhetjük meg, a **LOG** ablakban pedig a program-üzenetek látszanak.

Ezennel létrehoztunk egy SAS adatállományt, amiben az adataink vannak.

A SAS adatállomány egy olyan adatfájl, amit a SAS rendszer tárol.

SAS program >>>>> **SUBMIT** >>>>>



AZ ADATÁLLOMÁNY FELÉPÍTÉSE

A SAS ADATÁLLOMÁNYOK TÁBLÁZATOK

Ezek lehetnek csak olvashatóak, írhatóak, megnézhetőek, editálhatóak, elemezhetőek, feldolgozhatók, mégpedig **SAS PROGRAMOKKAL**

Például:

Változók	**----->**	**NEV**	**KOR**
Megfigyelések			
↓	1	Béla	33
	2	Edit	47

TERMINOLÓGIA: AZ OSZLOPOKAT VÁLTOZÓKNAK (VARIABLES) HÍVJUK, A SOROKAT MEGFIGYELÉSEKNEK (OBSERVATIONS).

VÁLTOZÓK TÍPUSAI

A változóknak két típusa van:

KARAKTERES (CHARACTER)

Ezek a változók számokat, betűket vagy speciális karaktereket tartalmazhatnak.

		↓	
Változók -->		NEV	KOR
Megfigyelések			
↓	1	Béla	33
	2	Edit	47

Értéküket mindig ‘ ‘ között hivatkozzunk.

NUMERIKUS

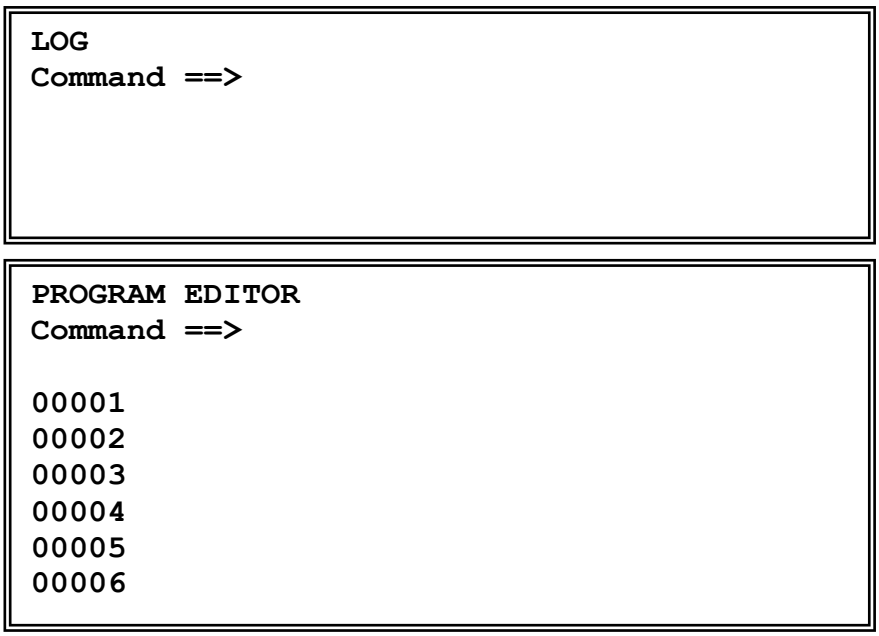
Ezek a változók csak számokat tartalmazhatnak.

			↓
Változók -->		**NEV**	**KOR**
Megfigyelések			
↓	1	Béla	33
	2	Edit	47

P2 FEJEZET
A SAS KÖRNYEZET

P2.1 DISPLAY MANAGER

Ha a SAS-t meghívjuk, akkor a 6.12-es verzióban az alábbihoz hasonló képernyőt kapunk. Két ablak látszik. Az alsó a **PROGRAM EDITOR,** ahová a SAS kódot írjuk. A felső a SAS **LOG**, amelyben nyomon követhető egy SAS kód lefuttatásának minden mozzanata lépésről lépésre.



Nem látható , de még ’élő’ ablak az **OUTPUT** ablak. Ez az a hely, ahová a nem grafikus eljárások az eredményeiket írják.

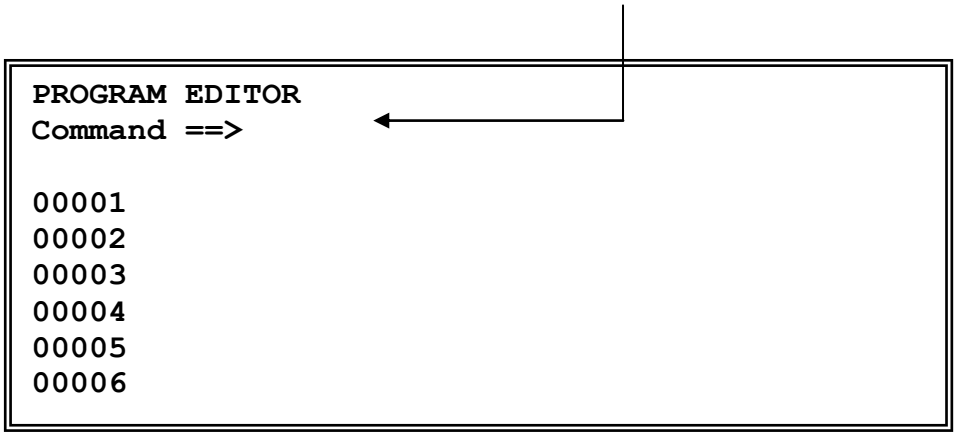
A SAS 7.0 verziójától fogva a képernyő baloldali részén látható még az **EXPLORER** ablak, melynek segítségével különböző fájlkezelési műveleteket végezhetünk.

Ugyancsak az új verziótól kezdve létezik a **RESULT** ablak, amely az OUTPUT ablakkal egyetemben mindaddig a háttérben marad, amíg egy program futtatása során a vezérlés rá nem mutat.

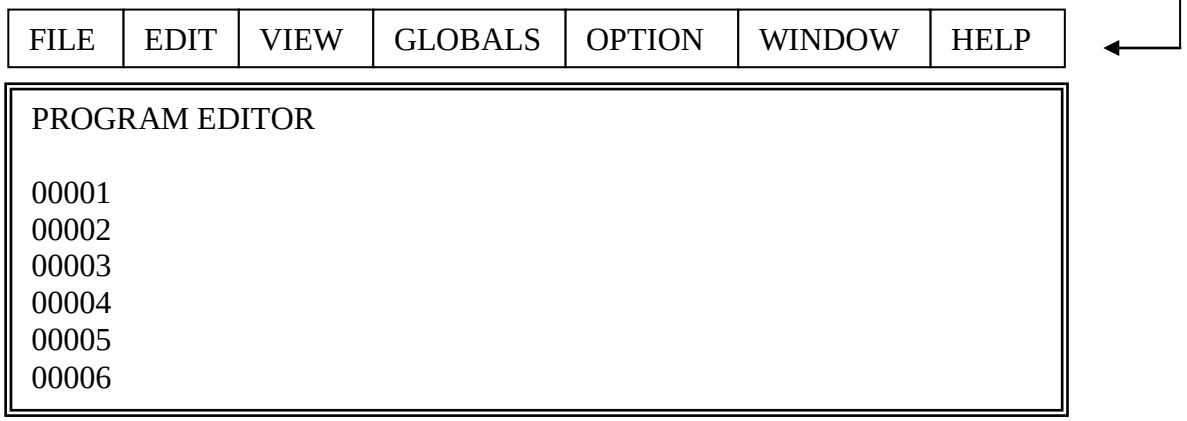
Habár a hagyományos **PROGRAM EDITOR** (továbbiakban PGM) ablak továbbra is használható, helyette belépéskor egy **EDITOR – Untitled** ablak jelenik meg, amelynek számtalan előnye mutatkozik a hagyományos PGM ablakkal szemben. Az áttekinthetőség kedvéért egyelőre maradjunk a hagyományos PGM editor ablaknál, (előhívása a PGM utasítással, vagy az F5 gomb lenyomásával lehetséges,) s majd később foglalkozunk az új lehetőségekkel.

ABLAK-STRUKTÚRA

Mindegyik ablaknak lehet parancssora (**COMMAND LINE**):



Vagy legördülő menüje (**PMENU**), amely a régebbi verzióknál pl.:



a 8.0-s verzióban pedig pl.

File	Edit	View	Tools	Run	Solutions	Window	Help

Ez egyrészt függ attól, hogy a SAS rendszer alapértelmezéseit hogyan állítottuk be, másrészt, hogy melyik ablak aktív éppen.

Váltogathatjuk a kétféle módot, ha a parancs mezőbe (PC-n van ilyenünk a képernyő bal felső csücskében) a **PMENU** kulcsszót írjuk, vagy a **TOOL/OPTIONS/PREFERENCES/VIEW** menüpontok kiválasztása után beállítjuk, vagy a UNIX-on korábbi verzióknál a menüsorból a **GLOBALS**, aztán a **GLOBAL, OPTIONS**, és végül az **COMMAND LINE** menüpontokat választjuk.

A tanfolyam alatt azt feltételezzük, hogy a parancssor aktív.

MOZGÁS AZ ABLAKOK KÖZÖTT

1 A WINDOWS ADTA LEHETŐSÉGEK HASZNÁLATÁVAL (window menü, vagy egér)

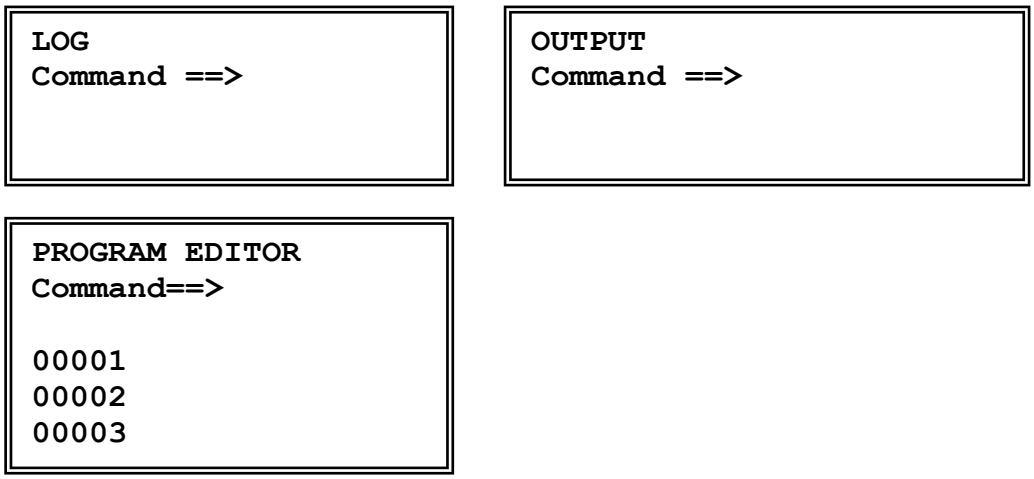
2 PARANCSOKKAL: **PGM**
LOG
OUTPUT

Próbáljuk meg az **END** parancsot az **OUTPUT** ablakban kiadni. A következő jelenik meg:

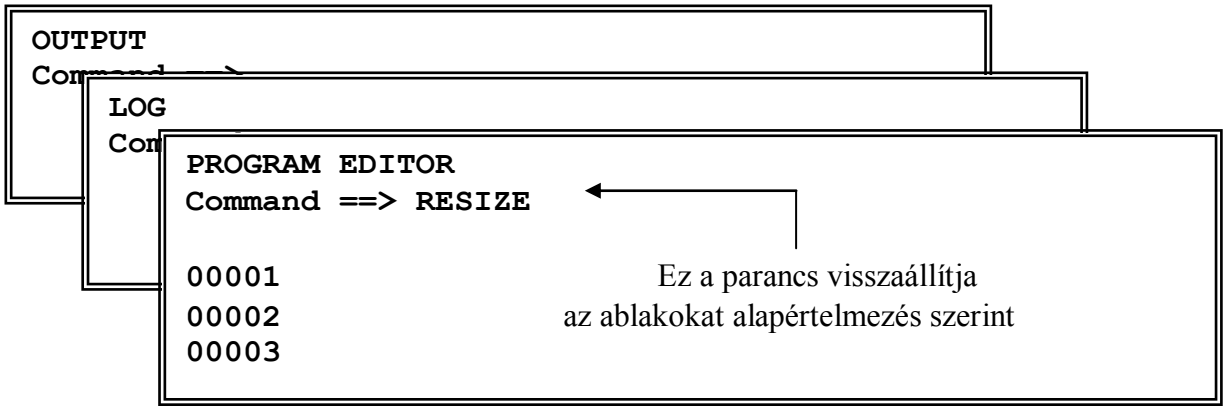
LOG
PGM

MOZAIKOS (TILE) ÉS LÉPCSŐZETES (CASCADE) ELRENDEZÉS

TILE PARANCS:



CASCADE PARANCS:



MOZGÁS EGY ABLAKON BELÜL

1 GÖRDÜLŐ SÁVOKKAL

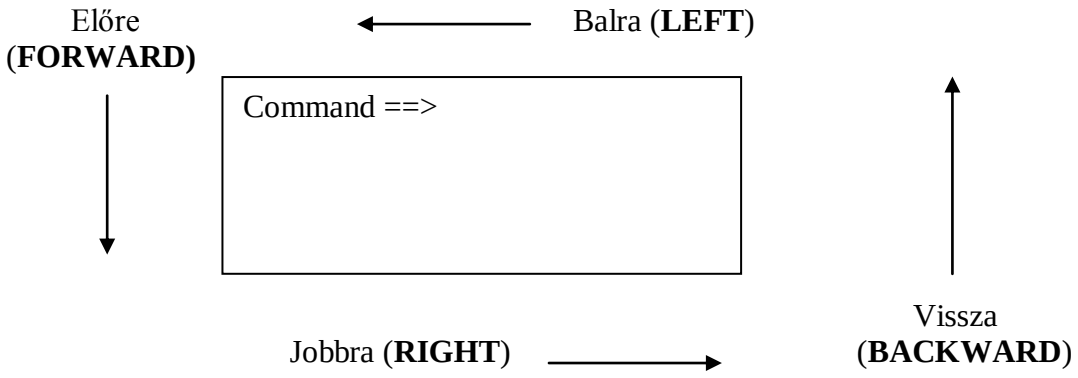
Az egér vagy a kurzor használatával mozgathatjuk a sávot fel, le, jobbra vagy balra a saját keretén belül:

2 UTASÍTÁSOKKAL

```
LOG
Command ==> FORWARD          ( - előre )
Note: At bottom                    ( - A végén vagyunk )
```

```
 PROGRAM EDITOR
 Command ==> LEFT            ( - balra )
 Note: At left side                 ( - A bal oldalon vagyunk )
 00001
 00002
 00003
```

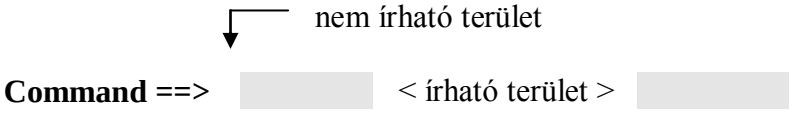
```
OUTPUT
Command ==> BACKWARD         ( - vissza )
Note: At top                       ( - Az elején vagyunk )
```



PARANCSSOROK (COMMAND LINES)

Mint már említettük, a PC-ken a SAS képernyő bal felső sarkában található egy parancsmező. Az ide beírt és elküldött parancsok hatása általános érvényű.

Ezenfelül minden SAS ablaknak van, ill. lehet parancssora, amely az ablakokon belül az illető ablak bal felső sarkában jelenik meg, és a következőképpen néz ki:



Az ebbe a sorba írt parancsok az ablak tartalmát befolyásolják – a szöveg törlése, szöveg mentése fájlba, egy bizonyos sorszámú sorra állás, a kód elküldése vagy visszahívása és még sok egyéb művelet. Sok parancsot olyan gyakran használunk, hogy ezért funkcióbillentyűhöz vannak rendelve.

KEYS ABLAK (FUNKCIÓBILLENTYŰK)

```
Command ==>   KEYS
```

```
LOG
Command ==>
```

```
PROGRAM EDITOR
Command ==>

00001
00002
00003
00004
00005
00006
```

```
KEYS
Command ==>

     Key        Definition

     F1         help
     F2         reshow
     F3         end
     F4         recall
     F5         pgm
     F6         log
     F7         output
     F8         zoom off; end
     F9         keys
     F11        command bar
     F12        clear
     ...
```

Nagyon fontos megjegyezni, hogy az **F3** gomb lenyomásának hatására (**END**) többnyire visszaléphetünk az előző ablakba, tehát pl. a KEYS ablakból is ezzel léphetünk ki.

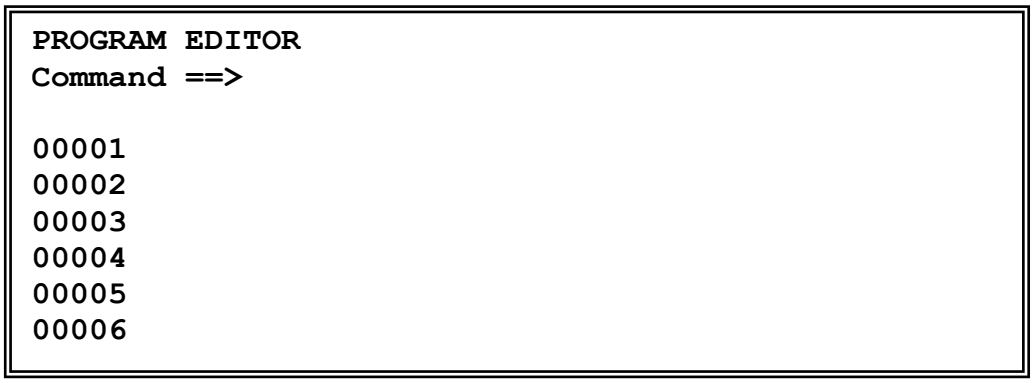
Általánosságban elmondhatjuk, hogy egy utasítás nevének elég annyi karakterét beütni, ahány egyedileg azonosítja.

Pl.	***OUTPUT***	helyett elég az	***OUT***	
	CLEAR	helyett elég a	***CLE***	
	ZOOM	helyett elég a	***Z***	(ablak megnagyítás) stb.

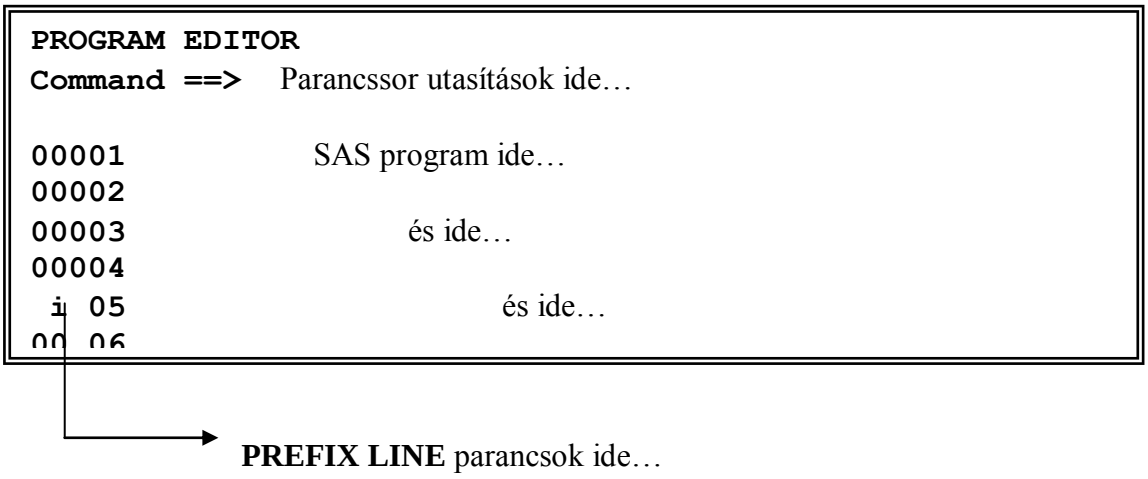
Funkcióbillentyűk használatával könnyebb mozogni az ablakokban vagy az ablakok között. Habár a legtöbb művelet a legördülő menük és az egér használatával elvégezhető, mégis, célszerű bizonyos gyakran használt utasításokat egy-egy funkció gombra tenni, hogy egy leütésre végrehajtódjanak. (pl. ablaktartalom törlése, ablakméret változtatása, ablak legtetejére, ill. legvégére ugrás stb.)

A PROGRAM EDITOR

Ez a SAS szövegszerkesztője, amellyel **PREFIX LINE** (’sor előtti’) parancsok, vagy grafikus felületen egér segítségével szerkeszthetünk szöveget. Most foglalkozzunk kicsit a sor parancsokkal.



A **PROGRAM EDITOR** ablak bal szélén lévő számozott területet hívjuk prefix területnek (Prefix area). Ide írjuk az editáló parancsok többségét. Amennyiben a prefix terület nem látható, a NUM paranccsal hívható elő és ezzel lehet eltüntetni is.



EDITOR PARANCSOK

MÁSOLÁS (COPY)

Tegyük fel, hogy egyetlen sort szeretnénk másolni. Írjunk egy ***c*** betűt a másolandó sor prefix területére, aztán mondjuk meg a szövegszerkesztőnek, hová szeretnénk a sort másolni – egy bizonyos sor mögé (***a*** =after) vagy elé (***b*** =before):

ÁTHELYEZÉS (MOVE)

Hasonlóképpen, sorokat átvihetünk (move) máshová, ha ***m*** betűt írunk a megfelelő sorba, aztán megmondjuk a SAS-nak, hogy hova vigye: egy bizonyos sor mögé (***a*** =after) vagy elé (***b*** =before):

SZÖVEG ISMÉTLÉSE

Az ismétlés jele a prefix területen: ***r*** (repeat). A mögötte lévő szám az ismétlések számát jelenti.

SZÖVEG TÖRLÉSE

Egy sor törléséhez írjunk ***d*** betűt a sor prefix területére. Két egymást követő sort az első sorhoz írt ***d2*** –vel törölhetünk.

SZÖVEG BESZÚRÁSA

Beszúrhatunk egy sort egy prefix területre írt **i**-vel, 10 sort **i10** beírásával, egy aktuális sor elé a beszúrás **ib** vagy **ib3** (insert before), stb.

Minden szövegszerkesztő alapvető részében van egy kapcsoló a beszúrási ill. átírási mód váltogatására; egy adott sorban a sor végéig törlés mindig egy-billentyűs művelet, mint az egy karakter törlése, bár ez az adott hardver és az operációs rendszer függvénye.

A fenti parancsok sorok együttesére, vagyis több egymást követő sorra is vonatkozhatnak.

Pl

```
00001
00002
RR  5  ------>  a sor-együttes első sora
00004
00005
RR006  ------>  a sor-együttes utolsó sora, amelyeket meg akarunk ötszörözni.
00007
```

PREFIX PARANCSOK

C / CC	Egy vagy több sor másolása
CL / CCL	Egy sor vagy blokk konvertálása kisbetűsre
COLS	Pozicionálást segítő sor megjelenítése
CU / CCU	Egy sor vagy blokk konvertálása nagybetűsre
D / DD	Egy vagy több sor törlése
D9999	9999 sor törlése
I / IB	Egy vagy több sor beszúrása
JC / JJC	Szöveg középre igazítása
JL / JJL	Szöveg balra igazítása
JR / JJR	Szöveg jobbra igazítása
M / MM	Egy vagy több sor másolása és törlése
A	Mögé (after), C és M (copy és move) esetén
B	Elé (before), C és M esetén
MASK	Egy új sor kezdőértékének definiálása
R / RR	Egy vagy több sor ismétlése
TC	Két sor összekapcsolása
TF	A szöveg túlcsordulása egy üres sorba
TS	A szöveg elvágása a kurzornál
) /))	Egy vagy több sor jobbra léptetése
	(adatvesztéssel járhat)
(/ ((	Egy vagy több sor balra léptetése
	(adatvesztéssel járhat)
> / >>	Egy vagy több sor jobbra léptetése
< / <<	Egy vagy több sor balra léptetése
Display manager parancsok:	
Reset	Az editor normál állapotának visszaállítása
Clear	Az ablak törlése

A fent említett sorparancsok legnagyobb része természetesen a grafikus felület adta lehetőségek következtében sorparancsok nélkül is elvégezhető az egér és a legördülő menük segítségével.

P2.2 PROGRAM VÉGREHAJTÁSA ÉS TÁROLÁSA

EGY PROGRAM ELKÜLDÉSE

Amikor megírtuk a programokat, el kell küldenünk (**SUBMIT** vagy **END**). Rendszerint van egy funkcióbillentyű, amely ezt megteszi (előző példánkban F3), és többnyire a program ablakot is visszaállítja eredeti méretére: *zoom off* ; end (példánkban F8).

A **PROGRAM EDITOR** visszaállítása normál méretűre egy nagy előnnyel bír: láthatóvá válik a **LOG** ablak, és láthatjuk az esetleges hibaüzeneteket, amint a programot elküldtük.

A **LOG** ablakban a

NOTE: The data set …….. has …. observations and ….. variables.

megjegyzés azt jelenti, hogy a most létrehozott SAS adatállományban ….. sor és …..oszlop van és hibátlanul jött létre.

Az **OUTPUT** ablakban a megfelelő eredmény látszik.

OLVASSUK EL A SAS LOG-OT!!! Minden hibaüzenetet, figyelmeztetést és megjegyzést.

A SAS **LOG** és **OUTPUT** ablakok folyamatosan íródnak egy SAS futás alatt. Ne felejtsük el, hogy a SAS a **LOG** ablak végét mutatja; **BACKWARD** funkciógombbal, vagy a **PageUp** billentyű segítségével lapozhatjuk visszafelé.

PROGRAM VISSZAHOZÁSA

Ha egyszer a programot elküldtük, a **PROGRAM EDITOR** kitörlődik. A **RECALL** (REC) parancs a program visszahozására szolgál.

Az adott SAS futás alatt elküldött összes programot visszahozhatjuk a futás sorrendjében visszafelé a **RECALL** funkciógomb (F4) többszöri megnyomásával.

PARANCS VISSZAHOZÁSA

A kérdőjel visszahívja az előző parancsot. Hosszabb parancsok begépelése esetén ennek igen nagy hasznát vehetjük. Pl. ha egy karaktert hibásan ütöttünk be, nem kell az egész sort megismételnünk. Célszerű ezt is funkcióbillentyűre tenni.

PROGRAM TÁROLÁSA

PROGRAM KIÍRÁSA EGY ADOTT KÖNYVTÁRBA:

az operációs rendszertől függő formában meg kell megadni a helyet és a nevet.

Pl.
```
PROGRAM EDITOR
Command ==> file 'c: \tanf\program\nevek.sas'
```

PROGRAM MENTÉSE AZ AKTUÁLIS KÖNYVTÁRBA (pl. UNIX esetén):

Pl.
```
PROGRAM EDITOR
Command ==> file 'nevek.sas'
```
vagy
```
file nevek
```

Ilyenkor automatikusan megkapja a **.sas** kiterjesztést

PROGRAM VISSZATÖLTÉSE

PROGRAM VISSZAHOZÁSA EGY ADOTT KÖNYVTÁRBÓL:

Pl.
```
PROGRAM EDITOR
Command ==> include 'c:\tanf\program\nevek.sas'
```
vagy
```
Command ==> inc 'c:\tanf\program\nevek.sas'
```

PROGRAM VISSZANYERÉSE AZ AKTUÁLIS KÖNYVTÁRBÓL (pl. UNIX):

Pl.
```
PROGRAM EDITOR
Command ==> include 'nevek.sas'
```
vagy
```
Command ==> inc 'nevek.sas'
```

FÁJLNÉV (FILENAME) UTASÍTÁS

Ahelyett, hogy magát a külső fájlt neveznénk meg a programban, megadhatunk egy fájl-hivatkozást (fileref), amely a külső fájlra hivatkozik, ill. mutat.

```
PROGRAM EDITOR
Command ==> submit

00001  filename prog 'c:\tanf\program\adatolv.sas' ;
00002
```

A parancs:

```
PROGRAM EDITOR
Command ==> inc prog

00001
00002
00003
```

Ennek eredményeként a program editor ablak tartalmazni fogja a 'c:\tanf\program\adatolv.sas' nevű állományt.

MEGJEGYZÉS:
A fájlneveket (**FILENAME**) használhatjuk bárhol a SAS rendszerben.

```
PROGRAM EDITOR
Command ==> submit

00001  filename adat 'c:\tanf\adatok\demo.dat' ;
00002
```

Ez a program olvassa a 'c:\tanf\adatok\demo.dat' nevű fájlt:

```
PROGRAM EDITOR
Command ==>

00001 data work.demo ;
00002   infile adat ;          * FILEREF, nem igazi név ;
00003   input ;
00004 run ;
```

P2.3 TOVÁBBI DISPLAY MANAGER LEHETŐSÉGEK

HELP KÉPERNYŐK

A fő help-képernyő további menükhöz vezet, míg végül az utasítások és eljárások szintaxisához érünk.

A 8.0-s help-képernyő eléréséhez írjunk a parancssorba a kulcsszót, vagy kattintsunk a 

gombra egerünkkel.

LEGÖRDÜLŐ MENÜK (PMENU)

Bármelyik ablak parancssorába írt utasítások helyettesíthetők a menüsor használatával
Egy lehetséges felállás a 8.0-s verzióban:

Ha a **LOG** ablak aktív:

FILE	EDIT	VIEW	TOOLS	SOLUTIONS	WINDOW	HELP

Ha a **PROGRAM EDITOR** ablak aktív:

FILE	EDIT	VIEW	TOOLS	RUN	SOLUTIONS	WINDOW	HELP

**** FILE**: katalógusokba való mentésre és visszahozásra, külső fájlba mentésre és onnan visszahozásra, az aktuális ablakból vagy a SAS-ból való kilépésre használjuk.
**** EDIT**: editálni, cserélni, elvágni és összeilleszteni lehet vele.
**** VIEW**: a képernyőn való megjelenítést lehet vele megváltoztatni.
**** TOOLS**: ilyen opció minden ablakban van, a SAS segédeszközeinek elérésére, továbbá a SAS felhívásakor érvényben lévő beállítások megváltoztatására és elmentésére való
** **RUN**: az editor ablakokban a program elküldésére, ill visszahívására használható
**** SOLUTIONS**: a SAS lehetséges eszközeinek (pl. ASSIST, Alkalmazás-fejlesztés, stb) felhívására szolgál.
**** WINDOW**: itt választhatjuk ki, hogy melyik ablakba akarunk átlépni, továbbá rendelkezhetünk az ablakok elrendeződéséről
**** HELP**: a **KEYS** és a **HELP** ablakok elérésére való.

DISPLAY MANAGER ABLAKOK

A már látott ablakokon kívül van még néhány ablak a **DISPLAY MANAGER**-ben, amit megnyithatunk.

Ezek némelyike nagyon hasznos az adatkezelésben.

ALAP ABLAKOK

** Program Editor
** Log
** Output

TOVÁBBI ABLAKOK

Libname	Az élő könyvtár-hivatkozások
Filename	Az élő fájlhivatkozások
Dir	Adatállományok és katalógusok egy adott könyvtárban
Var	Változók egy adott adatállományban
Cat	Egy adott katalógus bejegyzései
Options	Rendszer-opciók és aktuális beállításuk
Keys	Funkcióbillentyűk és aktuális jelentésük
Titles	Max. 10 cím aktuális beállítása
Footnotes	Max. 10 lábjegyzet aktuális beállítása
Notepad	Szövegszerkesztővel szabadon formázható terület különféle feljegyzéseknek
Help	**HELP** ablak megnyitása
Graph1-4	Legfeljebb 4 grafikus ablak nyitható ki, a grafikus egységtől függően, mindegyik egyenként mozgatható és méretezhető, akár a többi ablak. Van még egy ablak, amelyik kezeli ezt a 4 grafikus ablakot

A SAJÁT ABLAKOZÓ RENDSZER ÉS A SAS

Ezek a felsorolt ablakok saját ablakozó rendszerünk által kezelhetők. Tehát a Microsoft Windows alatti SAS ablakaink úgy fognak kinézni, mint a többi Microsoft Windows alatti ablak. Ha a SAS olyan rendszer alatt fut, amely ablakokat kezel, akkor a rendszer ablakozó rendszerét használja.

Ha nincs ablakozó rendszerünk (pl. UNIX alatt X-Windows nélkül), akkor a SAS saját ablakozó rendszerét használja.

Minden ablakozó módszer ugyanaz és mégis más. Az ablakozó rendszerek az ablakokat körülvevő dolgokban különböznek, mint például a mozgások, az ikon létrehozásának ellenőrzése, stb.

DISPLAY MANAGER PARANCSOK

Ablak parancsok	**Definíció**
AF	Egy alkalmazást meghív
Appointment	Meghívja az APPOINTMENT ablakot
Autoadd	Automatikusan új sort ad
Autoflow	A szöveg a következő sorban folytatódik
Autoscross	A képernyő sorait szabályozza (csak a LOG és OUT-PUT ablakokban)
Autopop	Az ablakot előhozza, ha írunk rá
Backward	Visszafelé lapoz
Bfind	Megkeresi egy érték korábbi előfordulását
Bottom	Az ablak aljára megy
Bounds	A szövegnek bal- és jobboldali határt szab
Bye	Kilép a SAS-ból
Calculator	Meghívja a kalkulátort
Cancel	Érvényteleníti a módosításokat az ablakban
Caps	A betűkészletet nagybetűsre állítja
Cascade	Az ablakokat lépcsőzetesen rendezi el
Catalog	Meghívja a katalógus-ablakot
Change	Kicseréli az ’X’ szöveget ’Y’-ra
Clear (LOG, PGM)	Törli az aablakot
Clear recall	Törli a visszahívás pufferét
Color	Egy ablak különböző részeit színezi
Command	Visszaállítja a parancssort
Cursor	A kurzort a parancssorba viszi
Cut	A kijelölt területet kivágja
Details on	A DIR ablak megjelenik a felbukkanó menüben
Dict	Létrehoz és kezel egy szótárat
Dir	Meghívja a DIRECTORY ablakot
End	Becsukja az ablakot
File	Az ablak tartalmát egy külső fájlba írja
Filename	Meghívja a FILENAME ablakot
Fill	Töltőkaraktereket ír a kurzor helyére
Find	Egy szöveget keres
Footnote	Meghívja a FOOTNOTE ablakot
Forward	Előre lapoz
Fsform	Meghívja a FSFORM ablakot
Help	Meghívja a HELP ablakot
Home	A kurzort a parancssorba viszi
Hscroll	Beállítja a vízszintes lapozás mértékét
Icon	Ikonná zsugorítja az ablakot
Include	Egy külső fájlt betölt a PGM ablakba
Indent	Baloldali margót hagy szövegíráskor
Keydef	Újradefiniálja a funkcióbillentyűket
Keys	Meghívja a KEYS ablakot
Left	Balra lapoz
Lib	Meghívja a LIB ablakot

Log	Megnyitja a LOG ablakot
Manager	Az OUTPUT MANAGER-t meghívja
Mark	Szöveget kijelöl
Mzoom	A hosszú üzenetet olvashatóvá teszi
Next	Megnyitja a következő ablakot
Notepad	Meghívja a NOTEPAD ablakot
Nums	A prefix területet megjeleníti ill. eltünteti
Options	Meghívja az OPTIONS ablakot
Output	Megnyitja az OUTPUT ablakot
Pclear	Törli a szerkesztő puffert
Program (PGM)	Megnyitja a PROGRAM EDITOR ablakot
Plist	Megjeleníti a szerkesztő puffert
Pmenu	Visszaállítja a menüsort
Prevcmd (vagy ?)	Visszahozza az előző parancsot
Prevwind	Visszahozza az előző ablakot
Print	Kinyomtatja az ablak tartalmát
Ralign	A szöveget írás közben jobbra igazítja
Reset	A prefix területet visszaállítja
Reshow	Újra felépíti az ablakot
Resize	Visszaállítja az ablakokat alapértelmezés szerint
Rfind	Megismétli az előző FIND parancsot
Right	Jobbra lapoz
Save	Az ablakot SAS katalógusba menti
Scrollbar	Gördülősávot rendel az ablakhoz
Setinit	Megnyitja a SETINIT ablakot
Siteinfo	Meghívja a SITEINFO ablakot
Smark	Kijelöl a képernyőn egy területet
Spell	Ellenőrzi a helyesírást
Status	A végrehajtási üzenetek kiírását ki-be kapcsolja
Store	A kijelölt területet eltárolja
Submit	Elküld egy programot (PGM)
Subtop	Elküldi a legfelső valahány sort
Tile	Az ablakokat mozaikosan rendezi el
Titles	Meghívja a TITLES ablakot
Top	Az ablak tetejére lapoz
Undo	Érvényteleníti a szövegbeli módosításokat
Unmark	A kijelölést megszünteti
Var	Megnyitja a VAR ablakot, változók egy adott adatállományban
Vscroll	Beállítja a függőleges lapozás mértékét
Wpopup	Meghívja a LIB, CAT, DIR, VAR és OUTPUT MANAGER menüjét
Wdef	Az ablak méretét definiálja
Wsave	Elmenti az ablak konfigurációját
X	Ideiglenesen kilép az operációs rendszerbe
Zoom	Teljes képernyőssé teszi az ablakot

Megjegyzés: Nem minden ablak-parancsot soroltunk itt fel, a határidőnaplóval, kalkulátorral, képernyőformáló és egyéb ablakokkal kapcsolatos parancsok kimaradtak.

ELVÁGÁS ÉS ÖSSZERAGASZTÁS

A SAS rendszerben mindenhol ugyanazt az editort használjuk, bárhol is van szükségünk szövegszerkesztőre, így ha egy programot szerkesztünk, szöveget írunk **FSLETTER**-ben, adatbeviteli képernyőt tervezünk vagy éppen a **NOTEPAD**-et használjuk, az editor a szoftvernek mindig ugyanaz a része.

SZÖVEG KIJELÖLÉSE

A szövegkijelölésnek 3-féle módja van:

A **MARK** parancs egyszerűen kijelöl egy szövegrészt az első jeltől az utolsóig, így ki kell adnunk a **MARK** parancsot a szövegrész elejének, majd külön a végének a rögzítéséhez; a szövegrész lehet többsoros is. A kiválasztott szöveg inverz módban látszik.
A **MARK BLOCK** (blokk kijelölése) parancsot szintén kétszer kell kiadnunk, de itt most a szövegblokk bal felső és jobb alsó sarkát kell kijelölnünk az aktuális ablakban.
A **SMARK** hasonlóan működik, mint a **MARK BLOCK**, de ezzel már a teljes képernyőn jelölhetünk ki vele területet, akár ablakok között is.

UNMARK

A kijelölést megszüntethetjük az UNMARK paranccsal.
(Az egész sokkal egyszerűbb, ha MARK, MARK BLOCK, SMARK és az UNMARK parancsok be vannak állítva funkcióbillentyűre.)

SZÖVEG ELTÁROLÁSA

A kijelölt szöveget eltárolhatjuk **STORE** utasítással.
A puffereknek (BUFFER) nevet is adhatunk. Pl.: STORE BUFFER=TEXT

ELTÁROLT SZÖVEG BEMÁSOLÁSA

Az eltárolt szöveget aztán a PROGRAM EDITOR bármely részére bemásolhatjuk (**PASTE**).
A tárolásnál a megnevezett puffert is használhatjuk: PASTE BUFFER=TEXT.

SZÖVEG KIVÁGÁSA

A szöveg kijelölt blokkját ki is vághatjuk (**CUT**).

A fenti utasítások végrehajtására szintén alkalmazhatjuk a funkció billentyűket, bár a WINDOWS-os felületnek megvan ezekre a maga eszköztára.

HULLÁMJEL

Használjuk a hullámjelet (~) a KEYS ablakban, hogy a gyakran használt szövegeket, program sorokat funkcióbillentyűvel helyettesíthessük:

```
KEYS <DMKEYS>
Command ==>
Key       Description

SHF F12   forward
CTL F1    mark block
CTL F2    ~proc print; run;
CTL F3    :ccl
CTL F4    :ccu
CTL F5
CTL F6
CTL F7    smark
CTL F8    :ib
CTL F9
CTL F10   :tc
```

KETTŐSPONT A KEYS ABLAKBAN

Használjuk a kettőspontot (:) a KEYS ablakban, hogy prefix parancsot rendeljünk egy funkcióbillentyűhöz:

```
KEYS <DMKEYS>
Command ==>
Key       Description

SHF F12   forward
CTL F1    mark block
CTL F2    ~proc print; run;
CTL F3    :ccl
CTL F4    :ccu
CTL F5    :d
CTL F6
CTL F7    smark
CTL F8    :ib
CTL F9
CTL F10   :tc
```

ÚJ LEHETŐSÉGEK A 7.0-S VERZIÓTÓL FOGVA

Néha nehéz visszaemlékeznünk, hová mentettünk el egy állományt, vagy melyik programmal (program résszel) milyen output-ot állítottunk elő. Az újabb SAS verziók megadják a lehetőséget a fent említett feladatok könnyű nyomon-követésére.

EXPLORER ABLAK

Az induló SAS képernyő baloldali részén látható a korábban már említett **EXPLORER** ablak, amely a SAS könyvtárakat mutatja. Kérhetjük a WINDOWS Intézőjéhez hasonló fa szerkezetben (View / Show Tree) is. Ilyenkor 2 részre oszlik az ablak, s a jobb oldalon az aktív könyvtár tartalma jelenik meg (definiált SAS könyvtárak, SAS fájlok)

LEHETSÉGES MEGJELENÉSI FORMÁI:

- nagy ikonok,
- kis ikonok
- lista forma,
- lista forma részletes információkkal kiegészítve (típus, elérési útvonal, stb.)

HASZNÁLATA ÁLTAL NYÚJTOTT LEHETŐSÉGEK:

- új könyvtárak és SAS állományok hozhatók létre.(File / New)

- megvizsgálhatjuk a könyvtárak tartalmát (dupla kattintás a könyvtárra)

- a jobb egérgombot használva

    - megnézhetjük jellemzőiket (Properties)
    - törölhetjük, átnevezhetjük, másolhatjuk őket

- belenézhetünk a SAS állományokba (dupla kattintás hatására feljön a VIEWTABLE ablak, amelyben editálhatunk és megnézhetünk létező táblákat, vagy újat hozhatunk létre))

    - szintén megnézhetjük általános jellemzőiket (jobb egérgomb – rendezettség, indexeltség, sűrítettség, sorok és oszlopok száma, stb.)

    - megtudhatjuk a változók és az indexek nevét

    - host információkat nyerhetünk (hely, méret, stb)

RESULT ABLAK

A SAS felhívásakor az alapértelmezés szerint a háttérben lévő RESULT ablak a programok futása során elkészült output listák azonosítására és azok felhívására szolgál, s mihelyst újabb output listánk keletkezik, aktívvá válik.

Amíg a SAS aktív, az OUTPUT ablakban gyűlnek a listáink, amelyeket a RESULT ablakban kiemelve és a jobb egérgombot használva pl. törölhetünk és átnevezhetünk.

EDITOR – UNTITLED ABLAK

Ez az ablak tulajdonképpen megfelel a hagyományos PGM ablaknak, csak lényegesen többet tud annál.

- Színek használatával képes felhívni a figyelmünket a szintaktikai hibákra, így azok még a futtatás előtt kijavíthatók (pl. a lépéshatárokat, mint a PROC és a RUN, sötétkékkel, a kulcsszavakat világoskékkel, a numerikus konstansokat tengerzölddel, az idézőjelben lévő karakter sorozatokat lilával, a megjegyzéseket fűzölddel, a számára értelmezhetetlen részeket pedig piros színnel jelzi.)

- A lépések az Intéző könyvtárstruktúrájához hasonlóan becsukhatók és kinyithatók, megkönnyítve ezzel a hosszabb programokban való eligazodást

- Lehetőséget nyújt arra, (kiemelés) hogy az ablakban jelenlévő SAS kódnak csak valamelyik részét futtassuk, mentsük, vagy töröljük az ablakból.

P2.4 GYAKORLATOK

1. FELADAT

Hívjuk meg a SAS-t és nézzük meg, melyik funkciógombok jelentik a következőket:

ZOOM OFF; SUBMIT
RECALL
LOG
PGM
OUTPUT
END
FORWARD
BACKWARD
LEFT
RIGHT
ZOOM
KEYS
HELP
PMENU
HOME vagy CURSOR

Próbáljuk ki mindegyik funkcióbillentyűt.

Írjuk le, melyik billentyű hatása:
- egy karakter törlése
- egy karakter beszúrása
- törlés a sor végéig
- a sor végére állás.

2. FELADAT

Használjuk a **TILE** és **CASCADE** parancsokat, hogy lássuk a nyitott ablakokat. A **RESIZE** paranccsal állítsuk vissza alapértelmezésre.

3. FELADAT

Írjuk be a következő programot a **PROGRAM EDITOR**-ba, ha a sima **PGM** ablakot használtuk, és küldjük el (**SUBMIT**):

```
DATA sportok;
INPUT nev $ tagsorsz jatek $ díj;
DATALINES;
KOVÁCS      111   sakk  2.50
KOVÁCS      111   dáma  1.50
JÁNOSI      121   dáma  1.75
FEHÉR       176   dáma  1.50
FEKETE      275   sakk  2.50
```

```
RUN;
PROC PRINT DATA=sportok;
RUN;
PROC SORT DATA=sportok;
BY nev;
RUN;
PROC PRINT DATA=sportok;
BY nev;
RUN;
```

Ha nem működne, nézzük meg a **LOG** ablakban az üzeneteket.
Lehetséges hibák:
- hiányzó pontosvessző,
- hiányzó dollárjel,
- pontosvessző nem lehet az adatok között.

Menjünk vissza a **PROGRAM EDITOR** ablakba, hívjuk vissza a programot (**RECALL**), javítsuk ki és küldjük el újra.

Nézzük meg az eredményt az **OUTPUT** ablakban.

4. FELADAT

Most cseréljük ki a két ***BY nev;*** utasítást ***BY jatek;***-ra. Milyen változást látunk az ***OUTPUT*** ablakban?

5. FELADAT

Tároljuk el a programot egy külső fájlban. Nézzük meg a megjegyzést (**NOTE**) az elmentett sorok számáról.
Töröljük a **PROGRAM EDITOR** ablakot és töltsük vissza a programot a külső fájlból. Nézzük meg a megjegyzést a visszatöltött sorok számáról.

6. FELADAT

Használjuk a **MARK BLOCK, CUT, STORE** és a **PASTE** parancsokat és írjunk is egy keveset, hogy a programunk a következő legyen:

```
DATA sportok;
INPUT nev $ jatek $ díj;
DATALINES;
KOVÁCS      111   sakk        2.50
KOVÁCS      111   dáma        1.50
JÁNOSI      121   dáma        1.75
FEHÉR       176   dáma        1.50
FEKETE      275   sakk        2.50
KOVÁCS      111   kártya      1.50
KOVÁCS      111   dominó      0.50
JÁNOSI      121   kártya      1.25
FEHÉR       176   dominó      0.50
```

```
FEKETE      275   kártya      1.50
RUN;
PROC PRINT DATA=sportok;
RUN;
PROC SORT DATA=sportok;
BY nev;
RUN;
PROC PRINT DATA=sportok;
BY nev;
RUN;
```

Teszteljük a programot.

7. FELADAT

Menjünk végig a különböző menükön és vessük össze a már általunk használt parancsokkal. Gyakoroljuk azokat, amik érdekelnek minket.

8. FELADAT

Az előbbi programunkat lássuk el több helyen megjegyzésekkel
- az egész DATA lépést
- egyetlen sort
- egy sor egy részét

9. FELADAT

Gyakoroljuk az alábbi sorparancsokat:

I	IB	C	A	M	R	D	O
I5	IB3	CC	B	MM	RR	DD	
<	<<	(	((	JJL	JJR	CL	CU

A leghasznosabbakat rendeljük egy-egy funkciógombhoz.

10. FELADAT

Másoljuk át a 6. Feladat alatti programot az **EDITOR – UNTITLED** ablakba és nézzük meg a szintaktikai kijelzéseket, majd gyakoroljuk a lépések ki-be csukását.

11. FELADAT

Az **EXPLORER** ablakot használva definiáljuk a ***TANF\SASOK*** könyvtárat ***SASOK*** néven és nézzünk meg benne mindent, amit csak tudunk (fájlok, fájljellemzők).

P3 FEJEZET

A DATA LÉPÉS

3.1 KÜLSŐ ADATOK ÉS SAS ADATÁLLOMÁNYOK

3.2 A DATA LÉPÉS

3.3 MÓDSZEREK KÜLSŐ ADATOK OLVASÁSÁRA

3.4 A DATA STEP ÉS A PDV BELÜLRŐL

3.5 GYAKORLATOK

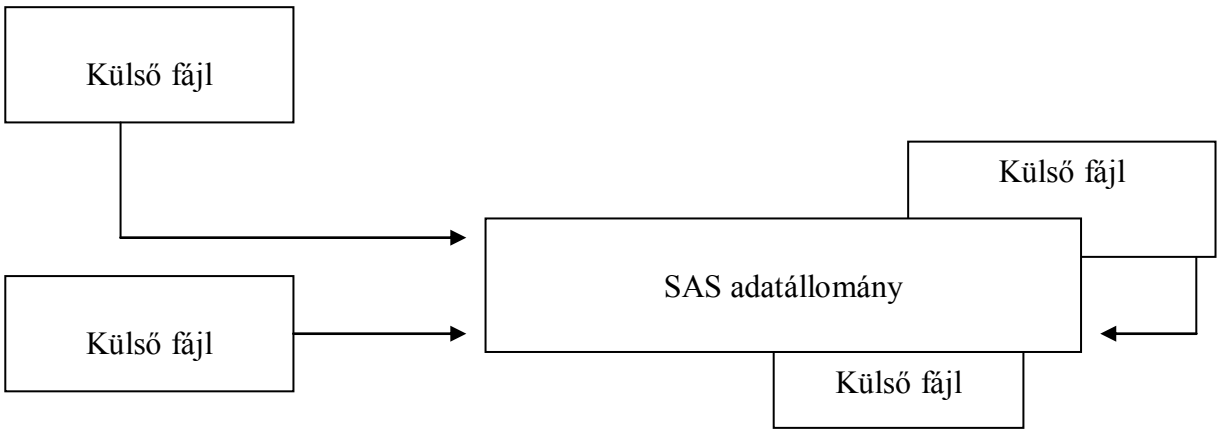
P3.1 KÜLSŐ ADATOK ÉS SAS ADATÁLLOMÁNYOK

Az adatok tárolása nagyon sokféle lehet a különböző operációs rendszerekben. A SAS rendszer az adatait SAS adatállománynak nevezett fájlokban tárolja.

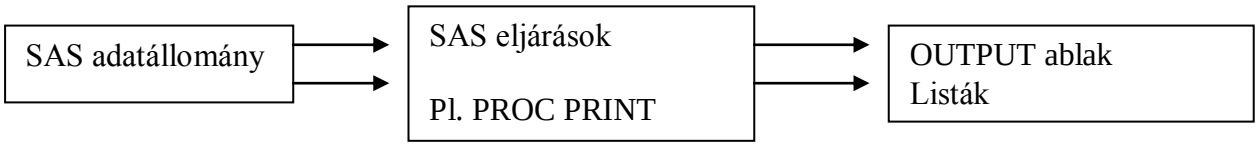
A SAS adatállomány egy lemezes fájl, amit SAS eljárások segítségével használhatunk. Két részből áll:

A SAS rendszer törődik a leíró résszel helyettünk, így amikor megváltoztatjuk vagy aktualizáljuk az adatállományt, a leíró részt automatikusan hozzáigazítja.

A külső adatok rendszerint lemezen (vagy szalagon) vannak, és a SAS rendszer bármilyen külső fájlt el tud érni. Általában a külső fájlt SAS adatállománnyá alakítjuk további felhasználás előtt.



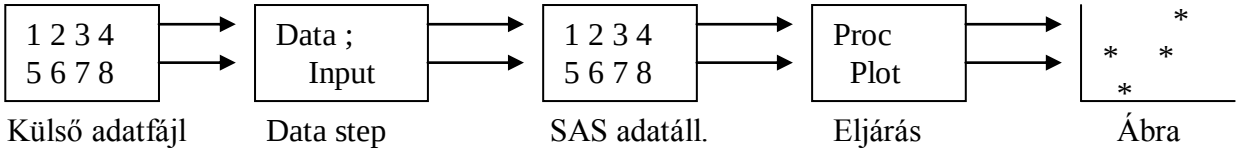
A SAS eljárások csak SAS adatállományokkal dolgoznak:



SAS ADATÁLLOMÁNYT KELL LÉTREHOZNUNK A KÜLSŐ ADATOKBÓL.

P3.2 A DATA LÉPÉS

A **DATA LÉPÉS** az a SAS program, amely a külső adatokat olvassa és SAS adatállományt állít elő:



Először is megismerkedünk a DATA lépéssel, amely olvassa a külső adatokat és SAS adatállományt készít, amely tartalmilag ugyanaz, mint a külső fájl. Később látni fogjuk, hogy a SAS adatállomány mennyivel hatékonyabb tárolási mód, és hogy használhatjuk a DATA step inputjaként.

A DATA LÉPÉS UTASÍTÁSAI

Négy fő utasítása van egy külső állományt beolvasó SAS DATA lépésnek:

```
1. DATA SAS_állomány_nevek ;
2. INFILE 'Külső_fájl' ;
3. INPUT változó1 változó2 változó3 ;
4. RUN ;
```

Sok egyéb utasítás van még, amit a DATA lépésben használhatunk az adataink ellenőrzésére, kezelésére és módosítására. Ez a fejezet áttekintést ad arról, hogy a DATA lépés programja hogyan olvassa a külső adatokat és hogyan hozza létre a SAS adatállományt.

A DATA UTASÍTÁS

A DATA utasítás a DATA lépés első utasítása. Három műveletet hajt végre:

1. Jelzi a DATA lépés kezdetét
2. Hol tárolódik az adatállomány?
3. Mi az adatállomány neve?

Megjegyzés: A DATA utasítás legfeljebb 100 adatállományt tud létrehozni egy lépésben.

Pl.
```
DATA egy kettő három ..... száz ;
... a többi data step utasítás ...
RUN ;
```

AZ INFILE UTASÍTÁS

Ez az utasítás a külső adatfájlra mutat. Ez mondja meg a DATA lépésnek, hogy hol található a külső adatállomány, amelyből a SAS adatállomány készül.

Hivatkozhatunk bármely lemezre, userid-re, bármely könyvtárra vagy fájlra és bármely kiterjesztésre (pl: .dat) az INFILE utasításban.

AZ INFILE UTASÍTÁS FORMÁJA

A külső adatállomány helymegadásának szintaktikája az operációs rendszertől függ.

MS-DOS rendszerben:

```
INFILE 'c:\userdir\nevek.dat' ;
```

UNIX-ban:

```
INFILE '/users/userdir/nevek.dat' ;
```

AZ INPUT UTASÍTÁS

Az input utasítás az az utasítás, amely ténylegesen beolvassa a külső adatok rekordjait egyenként a SAS DATA lépés programjába.

KARAKTERES ÉS NUMERIKUS VÁLTOZÓK

A DATA lépésben a változóknak két típusa van:
1. Karakteres változók
2. Numerikus változók.

KARAKTERES VÁLTOZÓK

Ezek a változók bármilyen értéket felvehetnek. Tartalmazhatnak számokat, betűket, speciális karaktereket, vagy ezek kombinációit.

A karakteres változókat a változónév után írt dollárjellel specifikáljuk.

NUMERIKUS VÁLTOZÓK

Ezek csak számokat tartalmazhatnak.
Ezen kívül néhány speciális jel megengedett, mint a tizedespont és az „E” az exponenciális együttható jeleként:

2.3E2
0.002

A RUN UTASÍTÁS

A RUN utasítás lezárja a DATA lépést.

```
DATA ……;
INFILE …..;
INPUT …..;
RUN;
```

A RUN utasításnak opcióként megadhatjuk, hogy a lefordított programot tárolja el. Egyéb opciója nincs.

AZ UTASÍTÁSOK SORRENDJE

A DATA lépésben az utasítások sorrendje fontos. Figyeljük meg, hogy az INFILE utasítás megelőzi az INPUT utasítást.

ALTERNATÍVÁK

A DATALINES VAGY CARDS UTASÍTÁS

Ahelyett, hogy a külső adatokra INFILE utasítással hivatkoznánk, beírhatjuk az adatainkat, mint a program sorait. Ez a DATALINES vagy CARDS utasítással hajtható végre, amely minden esetben a DATA lépés utolsó utasítása kell, legyen a RUN előtt.

```
DATA work.gyumolcs ;
     INPUT nev $ ar keszlet $ ;
CARDS;
alma         125.75      250,625
körte        189.10       75,230
barack       250.00       99,117
RUN ;
```

AZ EGYSZINTŰ ADATÁLLOMÁNYNÉV

Ha egyszintű állománynevet használunk, akkor munka állományokra hivatkozunk Például a ***DATA gyumolcs;*** és a ***DATA work.gyumolcs;*** DATA lépés AZONOS, mivel a LIBREF alapértelmezése (ld. 5. Fej.) WORK.

A WORK KÖNYVTÁR

A WORK könyvtár törlődik, amikor kilépünk a SAS-ból **BYE** paranccsal. Többet fogunk látni a WORK könyvtárról és egyéb könyvtárakról az 5. Fejezetben.

P3.3 KÜLSŐ ADATOK OLVASÁSÁNAK MÓDSZEREI

Az INPUT utasítást háromféle módon használhatjuk:

1. Listás input	egyszerű, de korlátozott
2. Oszlopos input	meglehetősen könnyű
3. Formázott input	bonyolultabb, de hatékony

LISTÁS INPUT

A listás input a külső adatmezőket megfelelteti az INPUT utasításban felsorolt változóneveknek.

```
DATA gyumolcs ;
INPUT nev $ ar keszlet $ ;
CARDS;
alma        125.75      250,625
körte       189.10       75,230
barack      250.00       99,117
RUN ;
```

A NEV az első mezőnek felel meg, az AR a következőnek, a KESZLET pedig az utolsónak. Mivel a KESZLET nem numerikus karaktereket is tartalmaz, csak karakteresen tudjuk beolvasni.

MEGJEGYZÉS A LISTÁS INPUTRÓL

Az INPUT utasítás végignézi a külső rekordot, amíg egy szóközt nem talál. Ezt feltételezi a mező végének. Ahogy várható, van néhány megkötés ennek a módszernek a használatára:

MEGSZORÍTÁSOK

- Dollárjelnek kell jeleznie, melyik változó karakteres.
- A mezőket szóközöknek kell elválasztania egymástól.
- A karakteres változók értékeire 2 megkötés van:
    - a hossz alapértelmezése 8 karakter
    - közbülső szóközök nem megengedettek.
- Ha az egyik mező adata hiányzik, akkor a változók párosítása elcsúszik, rossz lesz.
- Az alapértelmezés szerinti 8 karakteres hossz megváltoztatható a LENGHT utasítással, vagy formátum módosítással. Ennek taglalása másik tanfolyam témája.
- A listás input nagyon jól használható gyors tesztelésre, hogy megnézzük, a DATA lépésben van-e hiba.
- A listás input nagyon hasznos, ha a külső adatmezők ugyanabban a sorrendben, de más pozíciókon találhatók az egyes rekordokban.

OSZLOPOS INPUT

Az INPUT utasításnak ez a típusa a külső adatmezők oszlophelyeit használja.

COLS

A COLS parancsot a prefix területre írva megjelenik egy ún. vonalzó a PROGRAM EDITOR-ban, amely segítségünkre van a pozicionálásnál.

```
PROGRAM EDITOR
Command ==>

*COLS ----|----10---|----20---|----30---|----40---
00002 alma     125.75    250,625
00003 körte    189.10     75,230
00004 barack   250.00     99,111
```

MEGJEGYZÉSEK AZ OSZLOPOS INPUTRÓL

- Bármely érték az adott oszlophatárokon belül input értéknek számít. Így szóközt tartalmazó karakteres értékek is megengedettek.
- A karakteres változókat dollárjellel specifikáljuk.
- Numerikus változóknak nem kell a dollárjel.

REKORDOLVASÁS OSZLOPOS INPUT UTASÍTÁSSAL

MEZŐK ÚJRAOLVASÁSA

```
INPUT  nev $ 1-8 kezdobet $1;
INPUT  mezo1 $ 1-80 mezo2 $ 1-40 mezo3 $ 41-80;
```

MEZŐK OLVASÁSA TETSZŐLEGES SORRENDBEN

```
INPUT  valt1 $ 71-80  valt2 $ 1-15  valt3 $ 33-42;
```

A KÜLSŐ FÁJL REKORDHOSSZÁNAK KORLÁTJA

A DATALINES utasítást használva a PROGRAM EDITOR-ban az olvasáshoz a rekordhossz maximuma 80. Az INFILE utasítás esetén a maximum 32767 karakter.

FORMÁZOTT INPUT

Ez a leghatékonyabb típusú INPUT utasítás, de ennek írása egy kicsit bonyolultabb. Ezzel bármilyen típusú külső rekordot el tudunk olvasni.

SZINTAXIS

INPUT Pointer Változónév Input_formátum
Pointer Változónév Input_formátum
Pointer Változónév Input_formátum
Pointer Változónév Input_formátum ;

Pl. `INPUT @1 nev $12. @10 ar $6. @20 keszlet $7. ;`

POINTEREK

A pointerek típusa lehet:

ABSZOLÚT - mindegyikük egy bizonyos pozíciót jelöl a rekordban.

@n
@10
@145
@c
@c+5
@x+y

RELATÍV - mindegyik az előző mező végétől számol ekképpen megadott számú pozíciót.

+1
+100
+5
-10
+n
+(c+1)

A pointer minden új rekord esetén 1 kezdőértéket kap.

INPUT FORMÁTUMOK

Ezek a külső adatok tárolási képei.

Legutóbbi példánkban az ár és a készlet adatok, mint karakterváltozók kerültek beolvasásra. Ez nagyon kényelmetlen lehet, ha pl. összesent vagy átlagot szeretnénk számolni a gyümölcsök értékadataiból. Karakteres értékekkel nem végezhetünk semmilyen aritmetikai műveletet.

Az adatokat nem olvashatjuk be egyszerűen numerikusként, mert az adatmezők vesszőt tartalmaznak.
Szükségünk van valami olyan módszerre, amivel a gyümölcsök adatait számként olvashatjuk be. Hogy ezt megtegyük, specifikáljunk egy input formátumot.

```
DATA gyumolcs ;
INPUT nev $7. @10  ar  6.2  $20  keszlet  comma7. ;
DATALINES;
alma      125.75    250,625
körte     189.10     75,230
barack    250.00     99,111
RUN;
```

GYAKORI INPUT FORMÁTUMOK

NUMERIKUS VÁLTOZÓK ALAPVETŐ INPUT FORMÁTUMAI

Formátum		Input érték:
3.1	⟶	5.5 9.1 1.2
5.	⟶	100 99999 56
6.2	⟶	258.32 100.99 123.25

Megjegyezzük, hogy a külső adatban lévő tizedespont felülbírálja az input formátumot.

EGYÉB NUMERIKUS INPUT FORMÁTUMOK

Comma.	Pl. Comma6. ⟶	98,000	(ezreseket elválasztó
	Comma9. ⟶	1,284,750	vessző)
Hex.	Pl. Hex3. ⟶	4A7 (1191)	
Date.	Pl. Date7. ⟶	12Jul90	
	Date9. ⟶	12Jul1990	
	DDMMYY6. ⟶	120790	

KARAKTERES VÁLTOZÓK ALAPVETŐ INPUT FORMÁTUMAI

$6. ⟶	Albert
$16. ⟶	Toldi Miklós
$40. ⟶	Budapest, Keleti K. u. 5-7.

AZ INPUT FORMÁTUMOK RÉSZLETESEN A P10. FEJEZETBEN TALÁLHATÓK.

ADATOK ÉRVÉNYESSÉGE

NUMERIKUS ADATOK

Egy numerikus adat egyszerűen egy érvényes szám:

Adat	Érvényes szám-e?	Érvényes karakter-e?
100000	Igen	Igen
0.999999	Igen	Igen
1.3E5	Igen	Igen
-34054	Igen	Igen
- 55	Nem	Igen
1 0	Nem	Igen
5%	Nem	Igen
$12	Nem	Igen

A SAS figyelmeztető üzenetet ír a SAS LOG-ba, ha érvénytelen számot akarunk olvasni.

Belül a numerikus adatok 8 byte-os lebegőpontos számként ábrázolódnak. Ha egész adataink vannak, helyet takaríthatunk meg a numerikus változók hosszának csökkentésével (LENGHT utasítás).

KARAKTERES ADATOK

Azt mondjuk a numerikus értékeknél, hogy a szám a specifikált mezőben akárhol lehet, és mégis az INPUT utasítás helyesen olvassa be, amíg ez egy érvényes szám.
Karakteres értékekre ugyanez igaz, nem számít, hogy hol van a mezőben az érték, az INPUT utasítás kiveszi és a változóba tölti az értéket.

BALRA IGAZÍTOTT KARAKTERES ÉRTÉKEK

```
DATA betuk ;
INPUT v1 $ 1-4 v2 $ 6-9 v3 $ 11-14 v4 $ 16-19 ;
----|----10---|----20---|----30---|----40---
CARDS;
A    B    C    D
 A    B    C    D
  A    B    C    D
RUN;
```

→ program esetén

```
v1='A   '
v1=' A  '
v1='  A '
```

az értékek **NEM ÍGY** tárolódnak, hanem balra igazítva.

AZ ÖSSZEHASONLÍTÁS KÖNNYŰ

IF var1='A' THEN INPUT … ;

Hogy megtartsuk a vezető és a hátul lévő szóközöket, használjuk a **$CHAR.** input formátumot.

REKORDCSOPORTOK OLVASÁSA

Néha egy megfigyeléshez több adatrekord tartozik, pl. a háztartások felmérésénél egy háztartáshoz több adatrekord tartozhat.

A HASH (#) SOR-POINTER

A hash (#) sor-pointer a beolvasandó külső rekordra mutat:

```
PROGRAM EDITOR
Command ==>

00001   DATA kert ;
00002   INPUT nev $
00003      #2 nem $ kert $ allat $ gyerek $
00004      #3 koltseg meret $
00005      #4 vissza $ ;
00006   DATALINES ;
00007   Jánosi
00008   Férfi Igen Nem 3
00009   10.00 Kicsi
000010  Igen
000011  Varga
000012  Nő Igen Nem 1
000013  19.50 Nagy
000014  Igen
000015  Apró
000016  Férfi Igen Igen 3
000017  25.99
000018  Igen
000019  Dudás
000020  Nő Igen Igen 1
000021  5.85 Nagy
000022  Igen
000023 RUN ;
000024  PROC PRINT ; RUN ;
```

AZ OUTPUT LISTA

Megjegyzés: A legnagyobb hash-szám határozza meg a csoport méretét.

```
LOG
Command ==>

      1     DATA kert ;
      2     INPUT nev $
      3             #2 nem $ kert $ allat $ gyerek $
      4             #3 koltseg meret $
      5             #4 vissza $ ;
      6     DATALINES ;
      7     RUN ;
NOTE: The data set WORK.KERT has 4 observations and 8
Variables.
NOTE: The DATA statement used 3.00 seconds.
  36  PROC PRINT ; RUN ;
NOTE: The PROCEDURE PRINT used 2.00 seconds.
```

```
OUTPUT
Command ==>


OBS    NEV      NEM     KERT    ALLAT    GYEREK    KOLTSEG    MERET    VISSZA
1      Jánosi   Férfi   Igen     Nem        3       10.00     Kicsi    Igen
2      Varga    Nő      Igen     Nem        1       19.50     Nagy     Igen
3      Apró     Férfi   Igen     Igen       3       35.99     Nagy     Igen
4      Dudás    Nő      Igen     Nem        1        5.85     Nagy     Igen
```

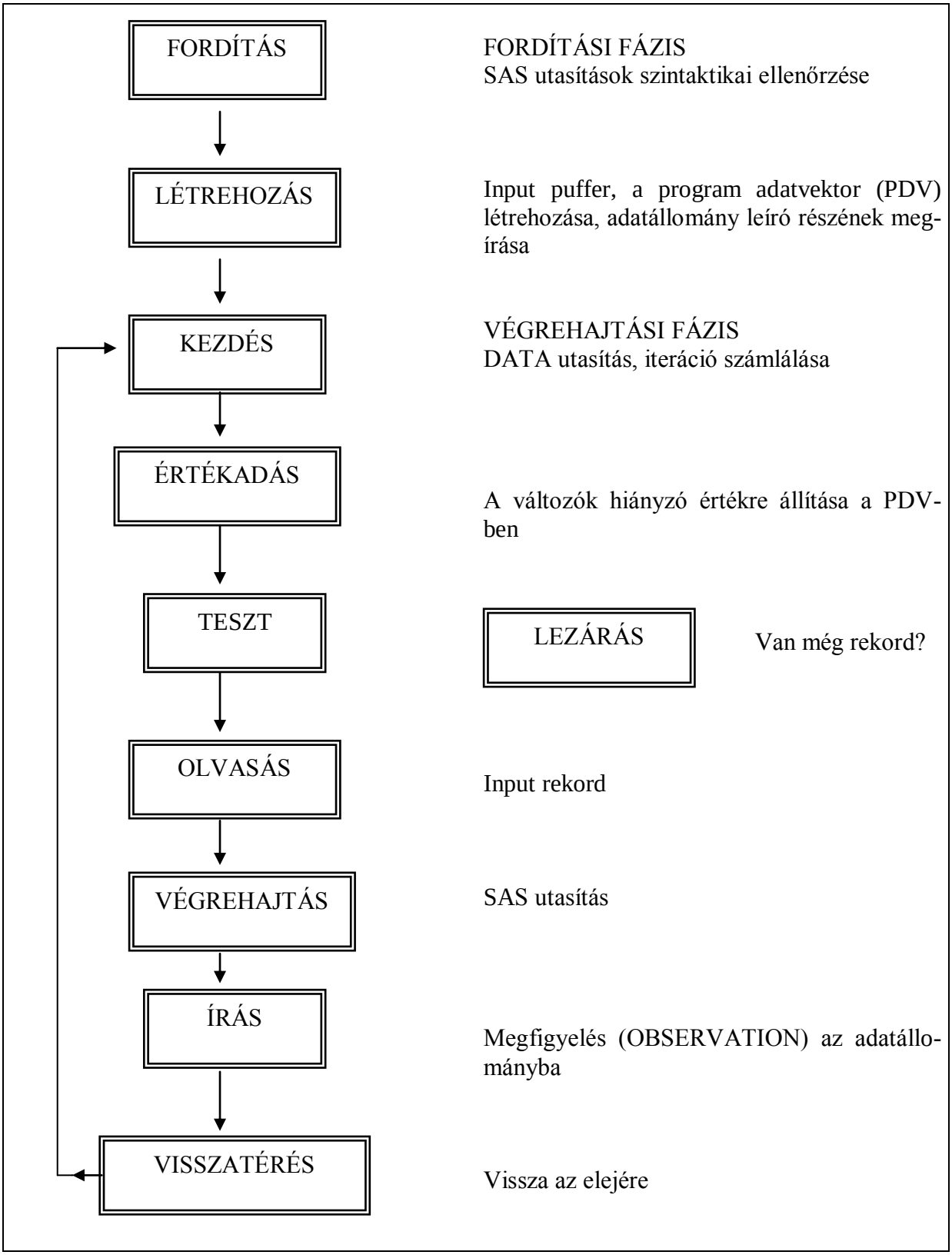
P3.4 A DATA LÉPÉS ÉS A PDV BELÜLRŐL

A DATA LÉPÉS ALAPETŐ MŰVELETEI

A **DATA** lépés SAS utasítások halmaza, amely DATA utasítással kezdődik. A SAS rendszer programozási nyelvének ez képezi az alapját, ez olvassa be a külső adatokat, és ez hozza létre a SAS adatállományt.

A DATA lépést képzelhetjük úgy, hogy külső rekordokra alkalmazott utasítások egy ciklusa. Rendszerint minden egyes külső rekord egyszer megy végig a DATA lépésen, amelynek eredményeként a SAS adatállományban egy megfigyelés (**OBSERVATION**) jön létre.

A következő ábra egy tipikus DATA lépés legfontosabb műveletit mutatja be.


FORDÍTÁS
FORDÍTÁSI FÁZIS
SAS utasítások szintaktikai ellenőrzése
LÉTREHOZÁS
Input puffer, a program adatvektor (PDV) létrehozása, adatállomány leíró részének megírása
KEZDÉS
VÉGREHAJTÁSI FÁZIS
DATA utasítás, iteráció számlálása
ÉRTÉKADÁS
A változók hiányzó értékre állítása a PDV-ben
TESZT
LEZÁRÁS
Van még rekord?
OLVASÁS
Input rekord
VÉGREHAJTÁS
SAS utasítás
ÍRÁS
Megfigyelés (OBSERVATION) az adatállományba
VISSZATÉRÉS
Vissza az elejére

A FORDÍTÁSI FÁZIS

Három dolog jön létre fordítási időben:

- AZ INPUT PUFFER:

A memória egy része, amibe a külső rekordok töltődnek be. A puffer vagy egysoros vektor, vagy többsoros vektor, amit a Hash n sor-pointer határoz meg.

- A PROGRAM ADATVEKTOR (PDV: Program Data Vector):

A memória egy része, amely az éppen létrehozandó SAS megfigyelést (OBSERVATION) tartalmazza. Ez az a hely, ahol a DATA lépésben létrehozott változókra vonatkozó műveletek eredményei tárolódnak.

- A LEÍRÓ RÉSZ:

A létrehozott SAS adatállomány fejrésze.

A VÉGREHAJTÁSI FÁZIS

A végrehajtási fázis alatt a DATA lépés utasításai hajtódnak végre, rendszerint minden egyes külső rekordra egyszer. A rekord adatait a rendszer az input pufferbe olvassa be, átviszi a PDV-be, végrehajtja a számításokat a többi változóra, és végül a SAS adatállományba kiírja a rekordot.

Ha a rekord kiíródott, a program visszatér a DATA lépés elejére és a változókat beállítja hiányzó értékre. A ciklus megismétlődik a következő külső rekordra.

P3.5 GYAKORLATOK

1. FELADAT

Olvassuk be ***SZEMELY.DAT*** fájlunkat a ***TANF\DATA*** könyvtárból ***SZEMELY*** néven **listás inputtal** és írjuk ki az OUTPUT ablakba! Milyen problémával kerülünk szembe?

2. FELADAT

Ismételjük meg a futást úgy, hogy ezúttal **oszlopos input** -tal olvassuk be adatainkat, amelyeket most közvetlenül a programban helyezzünk el. Írjuk ki az OUTPUT ablakba! (A kezdőpozíciók meghatározására használjuk a **COLS** sorparancsot! Ezt természetesen csak a PGM ablakban tehetjük meg.)

3. FELADAT

Még egyszer futtassunk, most azonban a **formázott input** -ot használva. A változóknak a következő neveket adjuk: NEV, SZULEV, NEM, CSAL, GYEREK, SZEM, MAGAS. Ha sikerrel jártunk, ezúttal mentsük is el a programot a ***TANF\PROG*** könyvtár alá ***SZEMELY.SAS*** néven!

4. FELADAT

Nézzük meg a ***HIVATAL.DAT*** nevű külső fájlt a ***TANF\DATA*** könyvtár alatt. Jegyezzük meg, hogy milyen adatokat tartalmaz és azok hogy helyezkednek el. (Használjuk a PGM ablakot!)

5. FELADAT

Olvassuk be a ***HIVATAL.DAT*** nevű fájlt a SAS-ba a legmegfelelőbb módszert használva. Használjuk a következő változóneveket: OSZTALY, NEV, FOGLALK, FIZETES, SZORZO, PREMIUM.
Mentsük el a programunkat a ***TANF\PROG*** könyvtárba ***HIVATAL.SAS*** néven.

6. FELADAT

A ***TANF\DATA*** könyvtárban lévő ***UZLET.DAT*** nevű fájl egy megfigyeléséhez több adatrekord tartozik. Az **első** sorban az üzlet neve, azonosító száma és telefonszáma található. A **második** sor a vásárló nevét és a hitel kódját tartalmazza. A **harmadik**ban találhatók a következők: az üzlet irányítószáma, a helységnév és a pontos cím. Az **utolsó**ban pedig az üzlet postafiók száma.
Próbáljuk meg ezt SAS-ba tölteni és mentsük el a programunkat a ***TANF\PROG*** könyvtárba ***UZLET.SAS*** néven.

7. FELADAT

Hogyan módosíthatnánk a programot úgy, hogy csak az üzlet nevét, irányító számát és címét olvassuk be.

P4 FEJEZET

SAS ELJÁRÁSOK

4.1 ADATOK LISTÁZÁSA

4.2 ADATOK RENDEZÉSE

4.3 AZ OUTPUT FORMÁZÁSA

4.4 INFORMÁCIÓ AZ ADATÁLLOMÁNYRÓL

4.5 GYAKORLATOK

P4.1 ADATOK LISTÁZÁSA

A SAS eljárások olyan számítógépes programok, amelyek elérik a SAS adatállományokat és valamilyen elemzést végeznek el rajtuk. Az eljárások – eljárások - (PROC), amelyek kiegészítik a SAS rendszert, a PROC PRINT egyszerű listázási lehetőségeitől a faktoranalízis és az operációkutatás összetettségéig terjednek.

Ha már egyszer megismertük néhány SAS eljárás általános használatát, nagyon egyszerű ezt a tudást alkalmazni a többi, bonyolultabb eljárásra is. Ebben a fejezetben három eljárást tárgyalunk:

1. PROC PRINT
2. PROC SORT
3. PROC CONTENTS.

SAS ADATÁLLOMÁYOK LISTÁZÁSA PROC PRINT-TEL

A PROC PRINT valószínűleg a legkönnyebb SAS eljárás, minthogy egyszerűen kilistázza egy SAS adatállomány adatait.

A legegyszerűbb formája:

```
PROC PRINT ;
RUN ;
```

Ez kiírja az éppen most (utoljára) létrehozott SAS adatállományt.

Rendszerint az adatállományra a nevével akarunk hivatkozni:

```
PROC PRINT DATA=gyumolcs ;
RUN ;
```

Mint a többi SAS eljárásnak a PROC PRINT utasításnak is adhatunk opciókat, és más utasításokat is írhatunk a RUN utasítás elé, és ezzel létrehozunk egy PROC lépést.

Az egyéb utasítások között szerepelhet például egy FORMAT utasítás is, amellyel javíthatjuk az output megjelenését, egy BY utasítás, amellyel csoportosíthatjuk az adatokat, vagy egy VAR utasítás, amellyel átrendezhetjük vagy korlátozhatjuk a listázandó változókat.

LISTÁZÓ PROGRAM ÍRÁSA

Egy DATA steppel az adatokat tartalmazó külső fájlt beolvassuk egy SAS állományba.

A PROC PRINT eljárást a SAS adatállomány értékeinek kilistázására használjuk.

Az OUTPUT lista előállítása a legtöbb esetben nagyon könnyű. Ám abban az esetben, ha az adatállományunkban lévő változók nem férnek el egy sorban az OUTPUT ablakban, az eredmény nem jól olvasható. (Pl. a P3.5 / 6. Feladatban az *UZLET.DAT* állomány adatai)

AZ ELJÁRÁSOK ÁLTALÁNOS STRUKTÚRÁJA

Mielőtt áttekintenénk, hogyan lehet a PROC PRINT outputját befolyásolni, nézzük meg előbb az összes eljárásra vonatkozó struktúrát és terminológiát.

Az eljárások végrehajtható modulok (a gép által érthető eltárolt programok), de olyan formában, hogy a felhasználó adhat át nekik paramétereket. Ezek a paraméterek azt jelentik, hogy az outputot futásról futásra megváltoztathatjuk más-más opciók és paraméterek megadásával.

Gyakorlatilag minden eljárás ad valamilyen outputot a következő utasításokkal:

```
PROC XXXX ;
RUN ;
```

ahol az XXXX a meghívandó eljárás neve.

Az eljárások igazi rugalmassága az eredeti outputnak utasításokkal és opciókkal való megváltoztatásában rejlik.

```
PROC  eljárásnév opció1 opció2 .. ;
Utasítás .. ;
Utasítás .. ;
.
.
RUN;
```

A VEZÉRLÉS HÁROM SZINTJE

1. RENDSZER-OPCIÓK

Utasítás	Használata
OPTIONS utasítás	Rendszer-opciók beállítása

2. ELJÁRÁS-UTASÍTÁSOK

Utasítás	Használata
VAR utasítás	- a változók vagy oszlopok átrendezése listázandó változók kiválasztása
BY utasítás	- az output osztályokba csoportosítása mindegyik osztály megfelel a BY-változó egy értékének
FORMAT utasítás	- az értékek kiírási módjának megváltoztatása
LABEL utasítás	- a változónevek kiírásának megváltoztatása

3. AZ ELJÁRÁSOK OPCIÓI

Opció	Használata
DATA=	- megmondjuk a PROC PRINT-nek, melyik SAS adatállományt listázza
D	- az output sorai között két üres sor legyen (dupla sorhúzás)
NOOBS	- a megfigyelések sorszáma ne kerüljön kiírásra
LABEL	- a LABEL utasításban megadott szöveg szerepeljen a változónév helyett
UNIFORM	- az output lapjai egyformák legyenek

Megjegyzés: Vannak további utasítások és opciók lásd a felhasználói kézikönyvekben.

A VAR UTASÍTÁS

A VAR utasítással mondhatjuk meg egy eljárásnak, hogy melyik változókkal foglalkozzon. A PROC PRINT esetében a VAR utasítás a változók kiírási sorrendjének megváltoztatására is használható.

Más SAS eljárások hasonlóképpen használják a VAR utasítást, kiválaszthatjuk vele az elemzésben részvevő változókat.

Formája:

```
VAR változó1 változó2 … ;
```

A többi utasítást és opciót a fejezet későbbi részében ismertetjük, de előbb tegyünk egy kis kitérőt.

P4.2 ADATOK RENDEZÉSE

PROC SORT

A PROC SORT az az eljárás, amellyel csökkenő vagy növekvő sorrendben rendezhetünk. Az eljárás nem készít listát, csak átrendezi a SAS adatállományban a megfigyeléseket. Az eljárásnak tudnia kell, hogy hogyan rendezze az adatokat pl. dátum szerint, vezetéknév szerint, keresztnév szerint, vagy esetleg valamilyen kulcs szerint. Ezt a BY utasítással adjuk meg.

Az eljárás szokásos formája:

```
PROC SORT DATA=sasdata OUT=temp ;
 BY {DESCENDING} változó1 {DESCENDING} változó2  ... ;
RUN ;
```

MEGJEGYZÉSEK:

Figyeljük meg, hogy nincs DATA lépés a programban. Ha az adatállományt már egyszer létrehoztuk, azt bármely SAS eljárás használni tudja.

A PROC SORT CSAK A SAS LOG-BA ÍR ÜZENETET!!! Az OUTPUT ablakba nem kerül semmi.

FELÜLÍRÁS

A PROC SORT felülírja a SAS adatállományunkat, hacsak nem adunk OUT= opciót a PROC utasításban. A rendezés „helyben” hajtódik végre. Lehet, hogy ezt akartuk, bár megéri megkülönböztetni melyik fájlokat tekintjük „csak olvasható”-nak, és melyeket rendezhetjük át bármikor.

Az adatállományok tárolhatók csak olvasható könyvtárban, így megakadályozhatjuk, hogy a rendezés felülírja őket.

A RENDEZÉS SORRENDJE

A SORT mindig növekvő sorrendben rendez, hacsak nem adjuk meg a DESCENDIGN (csökkenő) opciót a változónév előtt.
A `BY DESCENDING FIZETES;` utasítás a fizetések értéke szerint csökkenő sorrendben fog rendezni, vagyis a legnagyobbtól a legkisebb felé.

A rendezés sorrendjét az ASCII vagy az EBCDIC kódok sorrendje határozza meg, a hiányzó értéket a lehető legkisebbként kezelve.
Ha karakteres változóra rendezünk, akkor az ékezetes betűk a végére kerülnek.

RENDEZŐ MEZŐK

A rendezőmezőket egymástól space karakterrel elválasztva soroljuk fel.
Nincs korlát arra, hogy hány mező szerint rendezhetünk.

RENDEZŐ SEGÉDPROGRAMOK

A SAS rendszer a saját rendező programját használja, ha a SORTPGM=SAS opció be van állítva; ha az opció SASPGM=HOST , akkor az operációs rendszer szolgáltatta rendezés fut; a SORTPGM=BEST esetén a SAS maga választja ki a leghatékonyabb módszert. Gyakorlati szabály, hogy a 2000 megfigyelésnél nagyobb adatállományoknál nem szabad a SAS belső rendezőprogramját használni, és ez a szám néhány operációs rendszernél még sokkal kisebb lehet.

INDEX-FÁJLOK

A SAS rendszer lehetősége, hogy indexelhetjük az adatállományt. Ez a módszer index-fájlokat hoz létre, amelyek segítségével az adatállomány logikailag az index sorrendjében jelenik meg. Az adatállományoknak lehet többszörös indexelése, így többszörösen „rendezve” jelenhet meg. (Indexeket a PROC DATASETS és a PROC SQL segítségével állíthatunk elő.)

Pl.
```
PROC SQL ;
CREATE {UNIQUE} INDEX indexnév ON libref.sasdata(v1,v2, .. vn);
QUIT ;
```

ahol a UNIQUE kulcsszó jelentése az, hogy nem lehetnek dupla rendezési kulcsaink, az indexnév a rendezési kulcsmező neve (pl. v1, ha csak egy mezőre rendezünk, több kulcsmező esetén ide bármit beírhatunk).

A LEGFONTOSABB

Ha az adatállományt egyszer már rendeztük (vagy indexeltük) egy változó szerint, akkor csoportműveleteket is végrehajthatunk rajta.

CSOPORT-MŰVELETEK

ADATOK EGY CSOPORTJÁNAK ELEMZÉSE - BY UTASÍTÁS

A SAS rendszer legtöbb eljárása megengedi a **BY** utasítás használatát. Ez arra utasítja az eljárást, hogy az elemzést az adatok egy olyan csoportján hajtsa végre, amelyet a BY-változó határozz meg.
Például, ha a GYUMEXP állományunkat a következőképpen íratjuk ki:

```
PROC PRINT DATA=gyumexp ;
BY gyumolcs ;
RUN;
```

akkor a PRINT eljárás készít egy listát az almákról , utána a körtékről és így tovább.

ALMÁK LISTÁJA
KÖRTÉK LISTÁJA

:
:

EGYÉB ELJÁRÁSOK

Egyéb eljárásokkal is hajthatunk végre műveleteket adatcsoportokon, mint például **a PROC MEANS** eljárással MEAN (átlag) opciót adva. megkaphatjuk a gyümölcsönkénti átlagárat az adatállományból, ha előtte a GYUMOLCS változóra rendezünk.

```
PROC SORT DATA=gyumexp ;
BY gyumolcs;
RUN;
PROC MEANS DATA=gyumexp MEAN ;
BY gyumolcs;
VAR ar keszlet;
RUN;
```

Az output lista a következő:

```
----------------- GYUMOLCS=alma ----------------------

             Variable          Mean
             ----------------------
             AR         130.9500000
             KESZLET      62145.40
             ----------------------

----------------- GYUMOLCS=körte ---------------------

             Variable          Mean
             ----------------------
             AR         202.9666667
             KESZLET       3597.00
             ----------------------

---------------- GYUMOLCS=szilva ---------------------

             Variable          Mean
             ----------------------
             AR         125.0000000
             KESZLET      23201.67
             ----------------------

-------------- GYUMOLCS=sárgabarack -------------------

             Variable          Mean
             ----------------------
             AR         187.3333333
             KESZLET       7233.33
             ----------------------

-------------- GYUMOLCS=őszibarack --------------------

             Variable          Mean
             ----------------------
             AR         230.5000000
             KESZLET      16068.00
             ----------------------
```

P4.3 AZ OUTPUT FORMÁZÁSA

CÍMEK ÉS LÁBJEGYZETEK

Többféle módon tehetjük jobbá listáinkat, ezek egyike a címek és lábjegyzetek használata. Bármelyik output lapnak legfeljebb 10 címe és 10 lábjegyzete lehet, amelyeket a TITLE (cím) és a FOOTNOTE (lábjegyzet) ablakokban illetve utasításokkal adhatunk meg.

TITLE ÉS FOOTNOTE ablakok hívása:

Command ==> TITLES (címek listája)
Command ==> FOOTNOTE S (lábjegyzetek listája)

GLOBÁLIS UTASÍTÁSOK

A TITLE és FOOTNOTE utasítások globálisak, azaz a DATA vagy a PROC lépéseken kívül is lehetnek.

Formájuk:

```
TITLEn 'CÍMn';
```

és

```
FOOTNOTEn 'LÁBJEGYZETn';
```

ahol **n** maximum 10 lehet.

Megjegyzés: Ha egyszer definiáltuk, akkor érvényben maradnak mindaddig, amíg nem érvénytelenítjük őket, vagy ki nem lépünk a SAS-ból.

Érvénytelenítés: TITLE ;
FOOTNOTE;

OUTPUT FORMÁTUMOK A MEGJELENÍTÉS MEGVÁLTOZTATÁSA

Az output formátumok nem hoznak létre adatot, csak az írásképet változtatják meg. Ez olyan, mint amikor leírunk papírra egy telefonszámot:

Ha azt írjuk : 2024011,
a számnak semmi értelme sincs.
Ha azt írjuk: 202 40 11 vagy 2 024 011,
akkor a szám kb. úgy néz ki, mint egy telefonszám, és jelentése van.

OUTPUT FORMÁTUMOK A SAS-BAN

A formátumok megváltoztatják a számok külsejét, megjelenési formáját magának a számnak a megváltoztatása nélkül.

Például számokat szeretnénk kiírni vesszőkkel és tizedespontokkal:

`4,500.90` **`4500.9`** helyett

A ***COMMAw.d*** SAS formátummal érhetjük ezt el. (comma8.2)

A **FORMAT utasítás** az output formátum definiálására szolgál. Segítségével egy, vagy több változóhoz hozzárendelhetünk egy output formátumot.

Pl.
```
PROC PRINT  …. ;
FORMAT változó1           COMMA10.2
       változó2 változó3  DOLLAR12.2 ;
RUN ;
```

A formátumok nagyon jól használható részei a SAS rendszernek. Ezek segítségével az adatokat nagyon sokféleképpen tudjuk kiírni.

Output formátummal olvashatóbbá tehetjük az eredményeinket, választhatunk a sok standard SAS formátumból, amelyek meg tudják változtatni az adatok megjelenítését.

AZ OUTPUT FORMÁTUMOK RÉSZLETESEN A P10. FEJEZETBEN TALÁLHATÓK.

MEGFIGYELÉSEK KIVÁLASZTÁSA

A WHERE UTASÍTÁS

Ahogy a VAR utasítást használhatjuk változók kiválasztására, úgy kiválaszthatunk megfigyeléseket a listázáshoz a WHERE utasítással.

Pl. a korábbi gyümölcs export állományunknál maradva a

```
PROC SORT DATA=gyumexp ;
  BY gyumolcs ;
PROC PRINT DATA=gyumexp ;
  VAR fajta ar keszlet ;
  BY gyumolcs ;
  FORMAT keszlet comma8. ar 7.2 ;
  WHERE keszlet  > 20000 ;
RUN ;
```

program lefuttatása a következő output listát eredményezi:

```
---------------------- GYUMOLCS=alma --------------------------------

              OBS     FAJTA           AR    KESZLET

                1     Jonatán      105.75    150,625
                2     Golden       148.00     85,730
                5     Jonagold     142.00     66,334


---------------------- GYUMOLCS=szilva ------------------------------

              OBS      FAJTA           AR    KESZLET

               10     Olasz kék     135.00    65,705


---------------------- GYUMOLCS=őszibarack --------------------------

              OBS     FAJTA           AR    KESZLET

               15     Champion     250.00    29,111
```

A PROC PRINT OPCIÓI

LABEL – UTASÍTÁS ÉS OPCIÓ

A leggyakoribb opciója a PROC PRINT utasításnak a LABEL, amellyel megnevezést írhatunk ki a változók nevei helyett. Amíg a változónevek max. 8 karakter hosszúak, addig ezek a megnevezések 40 karakterig terjedhetnek. Sőt a 7.0-s verziótól ezek a számok 32, ill. 256-ra módosultak

```
PROC PRINT DATA=gyumexp LABEL;
  VAR gyumolcs fajta ar keszlet ;
  BY gyumolcs fajta ;
  FORMAT keszlet comma8. ;
  WHERE keszlet  > 20000 ;
  LABEL gyumolcs  = 'Gyümölcs megnevezése'
        fajta     = 'Gyümölcs fajta nevek'
        ar        = 'Exportálási ár'
        keszlet   = 'Exportálható mennyiség (kg)' ;
RUN ;
```

DUPLA SORHÚZÁS ÉS A MEGFIGYELÉS SORSZÁMÁNAK ELNYOMÁSA

```
PROC PRINT DATA=gyumexp LABEL D NOOBS ;
      .
      .


-------- Gyümölcs megnevezése=alma ------
                              Exportálható
  Gyümölcs     Exportálási     mennyiség
   fajta           ár             (kg)

  Jonatán         105.75          150,625

  Golden          148.00           85,730

  Jonagold        142.00           66,334


------- Gyümölcs megnevezése=szilva ------

                              Exportálható
  Gyümölcs      Exportálási     mennyiség
    fajta           ár             (kg)

  Olasz kék        135.00           65,705


----- Gyümölcs megnevezése=őszibarack ---

                              Exportálható
  Gyümölcs     Exportálási     mennyiség
   fajta           ár             (kg)

  Champion        250.00           29,111
```

P4.4 INFORMÁCIÓ AZ ADATÁLLOMÁNYRÓL

Minden SAS adatállomány két részből áll:

1. maguk az adatok
2. leíró rész.

Az adatokat kilistázhatjuk a PROC PRINT segítségével, a leíró részt pedig a PROC CONTENTS használatával. A SAS rendszer kezeli a leíró részt, aktualizálja olyanokkal, mint a változók és a megfigyelések száma, a változók neve és típusa, hossza, formátuma, stb.

Pl . korábbi példánkból kiindulva futtassuk le a következő programot:

```
PROC CONTENTS DATA=gyumolcs;
RUN;
```

Az eredmény a következő lesz:

```
                          CONTENTS PROCEDURE

Data Set Name: WORK.GYUMOLCS                    Observations:          3
Member Type:   DATA                             Variables:             3
Engine:        V612                             Indexes:               0
Created:       11:23 Sunday, April 2, 2000      Observation Length:    28
Last Modified: 11:23 Sunday, April 2, 2000      Deleted Observations:  0
Protection:                                     Compressed:            NO
Data Set Type:                                  Sorted:                NO
Label:

               -----Engine/Host Dependent Information-----

                     Data Set Page Size:       8192
                     Number of Data Set Pages: 1
                     File Format:              607
                     First Data Page:          1
                     Max Obs per Page:         290


                          CONTENTS PROCEDURE

                     Obs in First Data Page:   3

            -----Alphabetic List of Variables and Attributes-----

                    #    Variable    Type    Len    Pos
                    -----------------------------------
                    2    AR          Num       8     12
                    3    KESZLET     Num       8     20
                    1    NEV         Char     12      0
```

Csupán a változókról a VAR ablak meghívásával is szerezhetünk információkat.

P4.5 GYAKORLATOK

Javaslat: Először olvassuk el a 7. Feladatnál leírtakat!

1. FELADAT

Hozzuk vissza a ***TANF\PROG*** könyvtárból a ***SZEMELY.SAS*** programot, és futtassuk le újból, ha a ***SZEMELY*** nevű SAS állományunk nincs már meg!

2. FELADAT

Nyomtassuk ki most csupán a ***NEV, SZEM, MAGAS*** nevű változókat úgy, hogy a *NEV* változó szerint ABC sorrendben legyenek az adatok. Veszünk-e észre valami érdekeset?

3. FELADAT

Csinosítsuk ki az outputunkat; adjunk listánknak címet, a lábjegyzetbe írjuk be, hogy ki készítette a listát, az egyes változókhoz rendeljünk értelmes megnevezéseket, használjunk dupla soremelést és ne kerüljön kiírásra a megfigyelések sorszáma!

4. FELADAT

Rendezzük a ***SZEMELY*** adatállományunkat a családi állapot (CSAL) szerint, majd listázzuk ki csoportosítva úgy, hogy minden családi állapot alkosson egy csoportot.

5. FELADAT

Írjuk ki az adatállomány leíró részét és tanulmányozzuk a kapott információkat! Ellenőrizzük, hogy figyelmesen dolgoztunk-e, nincsenek-e az adatállományt leíró információkon felül felesleges soraink. Ha igen, töröljük ki az output ablakot, hívjuk vissza az előző programot, javítsuk ki és futtassuk le újból. Mentsük ki az output ablak tartalmát egy fájlba.

6. FELADAT

Listázzuk ki az egész adatállományt a ***SZEM*** változó szerint rendezett csoportokban! Mindegyik szemszínen belül rendezzük az adatokat a legnagyobb GYEREK-től (gyerekek száma) a legkisebbig! Természetesen hagyjuk el megint a megfigyelések számát és adjuk a listánknak a megfelelő címet és lábjegyzetet!

7. FELADAT

Mentsük el sorra a programjainkat ***P4_n.SAS*** néven a ***TANF\PROG*** könyvtárba (ahol is **n** a feladat sorszámát jelenti)! Ezt a mentési szisztémát alkalmazhatjuk később is, hogy utólag tudjuk rekonstruálni, milyen feladatokat is oldottunk meg a tanfolyam során!

P5 FEJEZET
SAS ADATKÖNYVTÁRAK

5.1 SAS ADATÁLLOMÁNYOK ELÉRÉSE

5.2 AZ ADATKÖNYVTÁR KEZELÉSE

5.3 ÜGYES FOGÁSOK

5.4 SAS ADATÁLLOMÁNYOK OLVASÁSA

5.5 GYAKORLATOK

P5.1 SAS ADATÁLLOMÁNYOK ELÉRÉSE

A SAS adatállományok könyvtárakban vannak tárolva. A SAS könyvtárak szerkezete operációs rendszertől függően más és más.

A SAS adatállományban jobb helykihasználással vannak tárolva az adatok, mint a külső fájlokban. Tömörebben tárolódnak és a SAS eljárásokkal közvetlenül elérhetők.

Minden operációs rendszernek megvannak a maga konvenciói és sajátosságai, amelyek befolyásolják a SAS rendszerrel való kapcsolatukat. Minden SAS felhasználónak két elvárása van általában:

1. A SAS ADATÁLLOMÁNYT LEMEZEN TÁROLNI,
2. A LEMEZEN TÁROLT SAS ADATÁLLOMÁNYT ELÉRNI.

EGY- ÉS KÉTSZINTŰ SAS ÁLLOMÁNYNEVEK

Minden SAS adatállomány, amit eddig létrehoztunk ideiglenes adatállomány volt. Ezt tudjuk is, mivel minden eddigi DATA lépésben, amik létrehozták őket, egyszintű neveket adtunk.

```
DATA gyumolcs ;
   INFILE … ;
   INPUT ;
   …
RUN ;
```

Ebben az estben a LOG-ban a következő üzenetet láthattuk:

```
LOG ————————————————————————————————————————
```

Data set WORK.GYUMOLCS has 3 observations

Tehát a fájl neve előtt a WORK szó azt jelenti, hogy ideiglenes SAS állományt hoztunk létre (mivel mi nem ütöttünk be a fájl neve elé semmit).

Az állandó SAS adatállományok SAS könyvtárakban vannak, és kétszintű nevük van.

```
DATA tanf.gyumolcs ;
   INFILE … ;
   INPUT … ;
RUN ;
```

Itt az ADATOK nevű SAS adatállomány jön létre és tárolódik a TANF nevű SAS könyvtárban.

Ebből adódik a szabály:

AZ IDEIGLENES SAS ADATÁLLOMÁNYOKNAK EGYSZINTŰ NEVÜK VAN.
AZ ÁLLANDÓ SAS ADATÁLLOMÁNYOKNAK KÉTSZINTŰ NEVÜK VAN.

Hogyan tud hivatkozni a TANF előtag a SAS könyvtárra?
Mielőtt ezt a kérdést megválaszolnánk, előbb nézzük meg, mit is jelent a SAS könyvtár

FIZIKAILAG AZ OPERÁCIÓS RENDSZERNEK

és

LOGIKAILAG A SAS-NAK.

SAS KÖNYVTÁRAK

Operációs rendszer	Minek látszik a SAS könyvtár az operációs rendszernek?	Minek látszik a SAS könyvtár a SAS-nak?
MS-DOS	Directory	A SAS által generált specifikus dolgok legmagasabb szintű szervezése, amelyben a legegyszerűbb elem az adatállomány.
UNIX	Directory	

Láthatjuk, hogy lennie kell valami kapocsnak a kétszintű név első része és a SAS könyvtár fizikai leírása között.

KÖNYVTÁRAK HIVATKOZÁSI NEVEI (LIBREFS)

A LIBREF az, aminek segítségével a SAS rendszer eléri a lemezen lévő SAS adatállományokat.
Későbbi fejezetekben látni fogjuk, hogyan lehet egy SAS adatállományt beolvasni a DATA lépésbe, de mielőtt ezt tennék, egy LIBREF-et kell előállítanunk.

A LIBNAME UTASÍTÁS

LIBNAME *xxxxxxx* '*operációs rendszernek megfelelő könyvtárhivatkozás*' ;

xxxxxxx: libref

'*operációs rendszernek megfelelő könyvtárhivatkozás*': directory a lemezen

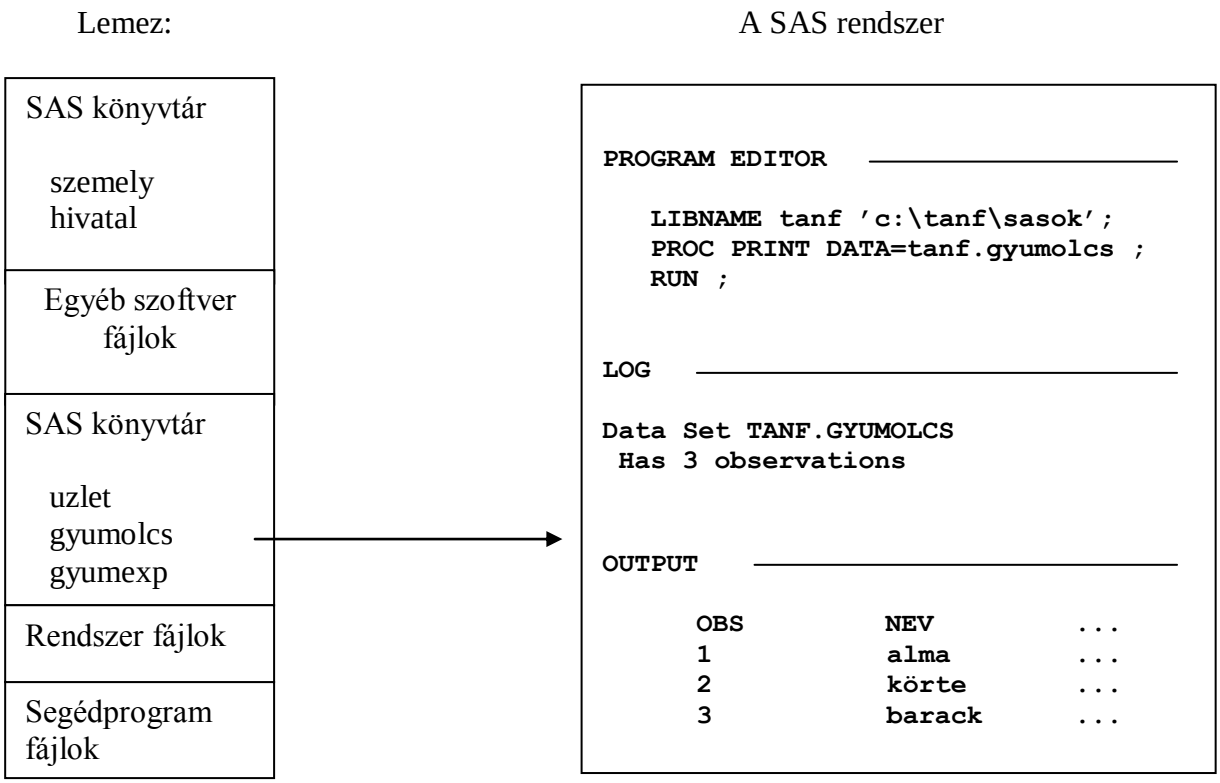
Mindegyik operációs rendszer sajátos módon hivatkozik a könyvtárra vagy a fájlra, összhangban a fájlmegnevezés szintaxisával az adott operációs rendszerben.

LIBREF ELŐÁLLÍTÁSA

```
LIBNAME tanf 'c:\tanf\sasok' ;
```
/submit/

Most már el tudjuk érni a tanf LIBREF-ben lévő adatállományokat, amint az a következő oldalon látható.

A LIBREF HASZNÁLATA



SAS ADATÁLLOMÁNYOK VÉGLEGES TÁROLÁSA

Azt is tudnunk kell, hogyan tároljunk SAS adatállományokat egy tetszőleges könyvtárban.

Először adjuk ki a LIBNAME utasítást.

```
LIBNAME tanf 'c:\tanf\sasok';
```

Aztán írjuk meg a DATA lépést.

```
DATA tanf.gyumolcs ;
   INFILE … ;
   INPUT ;
   …
RUN ;
```

AZ ÁLLANDÓ ADATÁLLOMÁNY HASZNÁLATA

Amikor legközelebb meghívjuk a SAS-t, nincs szükség az adatok újraolvasására.
Egyből hivatkozhatunk rá akár egy DATA, akár egy PROC lépésben.

```
LIBNAME tanf 'c:\tanf\sasok';
   PROC PRINT DATA=tanf.gyumolcs ;
   RUN ;
```

A LIBNAME utasítás néhány előnye:

- A LIBNAME utasítás feltételes is lehet a makró nyelv segítségével.

- A LIBNAME utasításnak nagyon sok opciója van. A 6.06-os verziótól kezdve a SAS rendszerben használható egy ún. Multi Engine Architecture (MEA). Ezáltal a SAS egy ENGINE nevű szoftver modullal is tud adatokat elérni: ezt a LIBNAME utasításban lehet megadni és elérési lehetőséget nyújt más típusú fájlokhoz, mintha azok egyszerű SAS adatállományok lennének.

P5.2 AZ ADATKÖNYVTÁR KEZELÉSE

A WORK LIBREF

A WORK LIBREF automatikusan hozzárendelődik minden egyszintű SAS adatállomány névhez.

Az ideiglenes SAS adatállományok is a lemezen tárolódnak, legalábbis a SAS futás végéig. DISPLAY MANAGER módban addig, amíg el nem hagyjuk a rendszert a BYE paranccsal, vagy a FILE menü EXIT parancsával.

A SAS rendszer automatikusan törli az összes WORK könyvtárbeli adatállományt a SAS futás végén.

A LIBNAME ABLAK

A Display Manager LIBNAME ablakában ellenőrizhetjük, hogy a LIBREF-ek hová mutatnak. (Ld. 7.0 verziótól az EXPLORER ablak)
Az ablak tartalma az operációs-rendszer függvénye.

A korábbi verziókban bármelyik mezőt egy „s”-sel kiválasztva, az újabbakban a DIR parancs hatására a DIRECTORY ablak aktivizálódik.

A DIRECTORY ABLAK

A korábbi verziókban bármelyik mezőt „s”-sel kiválasztva, az újabbakban a VAR parancs hatására megjelenik a VAR ablak.

Ugyanezeket az információkat, sőt bővebbet kaphatunk a PROC CONTENTS használatával is.

SEGÉDPROGRAMOK

PROC CONTENTS (TARTALOM)

A korábban már bemutatott **PROC CONTENTS** eljárás alkalmazható egy egész könyvtárra, és ekkor listát ad az ott található SAS adatállományokról.

PROGRAM EDITOR

```
PROC CONTENTS DATA=tanf._ALL_ NODS ;
RUN;
```

OUTPUT

```
                    The CONTENTS Procedure

                     -----Directory-----

                  Libref:         SASOK
                  Engine:         V8
                  Physical Name: c:\tanf\sasok
                  File Name:      c:\tanf\sasok


                                 File
#  Name       Memtype     Size  Last Modified
-----------------------------------------------
1  GYUMEXP    DATA        5120   02APR2000:14:50:50
2  GYUMOLCS   DATA        5120   02APR2000:14:51:00
3  UZLET      DATA       13312   02APR2000:14:51:08
```

PROC COPY (MÁSOLÁS)

Sokféleképpen másolhatunk SAS adatállományokat, a legszélesebb körben használt eszköz a PROC COPY. Ezzel a SAS eljárással egy könyvtárnak néhány vagy akár az összes adatállományát átmásolhatjuk egy másik könyvtárba.

A szintaktikája:

```
PROC COPY IN=libref OUT=libref ;
RUN ;
```

Hozzáírhatjuk még a SELECT (kiválasztást) vagy az EXCLUDE (kizár) utasításokat, amivel meghatározzuk a másolandó adatállományokat.

```
SELECT sasdata_1 .. sasdata_n ;
```

vagy

```
EXCLUDE sasdata_1 .. sasdata_n ;
```

PROC DATASETS (ADATÁLLOMÁNY-KEZELÉS)

Ezzel az eljárással egy könyvtár tartalmát változtathatjuk meg. Például átnevezhetünk, törölhetünk, indexelhetünk egy SAS adatállományt.
Átnevezésre csak ezt használhatjuk, mivel átnevezi az adatállományt az állomány leíró részében is, az operációs rendszer átnevezésre szolgáló parancsa viszont nem.

Pl.
```
PROC DATASETS LIBRARY=tanf;
CHANGE sasdata_1 sasdata_2 .. ;
DELETE sasdata_3 sasdata_4 .. ;
```
stb.

PROC COMPARE (ÖSSZEHASONLÍTÁS)

A PROC COMPARE összehasonlítja két SAS adatállomány változóinak értékét, és a különbségeket kiírja. A PROC COMPARE a következőkre használható:

- két SAS adatállomány közötti különbség megkeresésére
- a törzsállomány változásainak regisztrálására
- egy adatállományon belüli változók összehasonlítására
- változók közötti különbség százalékának kiszámítására.

```
PROC COMPARE DATA=libref.állomány1
             COMPARE=libref.állomány2
             METHOD=ABSOLUTE
             CRITERION=5 ;
      VAR valt11 valt12 ;
      WITH valt21 valt22 ;
      ID nev ;
RUN ;
```

A PROC COMPARE eljárás kiírja mindegyik adatállományra vonatkozóan a megfigyelések számát, a változók számát és összehasonlítja a VAR és a WITH utasításában felsorolt változók értékét.

Az eljárás egyenlőnek tekinti a két adatállományt a megadott tűréshatáron belül (CRITERION=5). A párosodó változók száma az output jobb oldalán olvasható.
Végül a változók neve, típusa és hossza is kiírásra kerül.

Természetesen a két állományban azonos sorrendben kell következniük a megfigyeléseknek. Jelen példánkban a rendezettségi szempont a NEV változó.

PROC APPEND (HOZZÁFŰZÉS)

Az APPEND eljárás az egyik SAS adatállomány megfigyeléseit egy másik SAS adatállomány végéhez hozzáadja. Az eljárást alapvetően két SAS adatállomány konkatenálására használjuk és nagyon hatékony abban, hogy nem kell végigolvasnia az első állományt mielőtt a másikat hozzáfűzné.

A PROC APPEND nagyon hasznos, ha:
- a változóértékekkel nem kell műveletet elvégezni
- gyakran kell nagy adatállományokat konkatenálni
- nincs más változó a DATA adatállományban mint a BASE állományban.

Egyébként használhatjuk a SET utasítást is (ld.7 fej.) a SAS adatállományok konkatenálására, különösen az alábbi esetekben:
- csoportműveletet kell végrehajtani (BY)
- adatállomány-opciókra van szükség, mint az IN= opció
- a változók értékeivel további műveleteket kell végrehajtani.

A PROC FSEDIT – a SAS/FSP szoftver része – is használható egy meglévő SAS adatállomány megfigyelésekkel való bővítésére. Ez interaktív eljárás, adatrögzítő rendszerekhez és adatellenőrzést végző alkalmazásokhoz szoktuk használni. Ezzel is lehet meglévő SAS adatállományhoz megfigyeléseket hozzáfűzni, de jobb a PROC APPEND, mert automatikus módszert kínál, amely háttérben is futtatható.

P5.3 ÜGYES FOGÁSOK

AUTOEXEC

A SAS rendszer képes egy SAS utasításokat tartalmazó fájlt AUTOEXEC fájlként használni. Ez azt jelenti, hogy a SAS rendszer meghívásakor ezek az utasítások automatikusan végrehajtódnak.

Használhatjuk ezt a technikát LIBNAME utasítások vagy egyéb opciók beállítására, amit minden SAS futáskor ugyanúgy szeretnénk használni.

A MÓDSZER:
1. Hozzunk létre egy SAS utasításokat tartalmazó lemezes fájlt.
2. Hívjuk meg a SAS rendszert egy erre a fájlra mutató AUTOEXEC opcióval.

SAS MEGHÍVÁSA AUTOEXEC OPCIÓVAL

```
SAS –AUTOEXEC sasauto.sas
```

Ahol a SASAUTO.SAS az a fájl, ahová az autoexec utasításokat elmentettük.

SASUSER

Ez egy olyan LIBREF, amely arra a SAS könyvtárra mutat, ahol a saját PROFILE állományunk van. Például megváltoztathatjuk a DISPLAY MANAGER ablak pozícióit, és a konfigurációt elmenthetjük ebbe a fájlba. A PROFILE érvényben marad az egész SAS futás alatt.

```
SAS –SASUSER  c:\userlib          <== ez a könyvtár
```

A SASUSER opció egy SAS könyvtárra mutat.

OPCIÓK

Meghívhatjuk a SAS rendszert a számunkra éppen szükséges rendszeropciókkal is.

Pl. **SAS –OPTIONS nonumber nodate pagesize=64 ;**
(Jelentése: ne számozza a lapokat, ne írjon dátumot, a lapméret 64 sor legyen.)

CONFIG

A SAS rendszer használ egy konfigurációs fájlt is, amely a SAS szoftver fájlokra mutat. Ezt lemásolva megváltoztathatunk néhány paramétert (pl. a képernyő típusát) és aztán meghívhatjuk a SAS-t a saját konfigurációs állományunkkal.

Pl. `SAS -CONFIG c:\ujconfig.sas`

A DM UTASÍTÁS

Néha szükségünk van DISPLAY MANAGER parancsokra a SAS programjainkban, vagy az AUTOEXEC fájlunkban. Ezt a DM utasítással oldhatjuk meg:

```
DM  'bármilyen DISPLAY MANAGER parancs'  ablak ;
```

Az alapértelmezés a PROGRAM EDITOR

Pl.

```
DM  'OUTPUT ; CLEAR' ;
```

Törli az OUTPUT ablakot, De a kurzort a PROGRAM EDITOR-ban hagyja.

```
DM  'OUTPUT ;  ZOOM ;  TOP'  OUTPUT ;
```

Kinagyítja az OUTPUT ablakot, a tetejére megy és ott is marad.

FILENAME UTASÍTÁS

A FILENAME utasításban definiált fájlnevet sok eljárás használhatja az adott fájl elérésére. Az itt definiált fájlnév egy lemezes fájl nevének „rövidítése”.

SZINTAXIS:

```
FILENAME xxxxxx  'Egy_lemezes_fájl ' ;
```

Pl.

```
FILENAME  uzlet  'c:\tanf\data\uzlet.dat' ;
```

Ettől kezdve ***uzlet*** néven fogunk hivatkozni a ***tanf\data*** könyvtárban található ***uzlet.dat*** nevű külső fájlra.

PRINTTO

A PROC PRINTTO segítségével átirányíthatjuk az eljárás outputját.

SZINTAXIS:

```
FILENAME xxxxx 'egy;lemezes;fájl' ;
PROC PRINTTO PRINT=xxxxx  {NEW} ;
RUN;
```

Pl.

```
FILENAME egyik 'output.lst' ;
PROC PRINTTO PRINT=egyik ;
RUN;
```

Ettől kezdve mindegyik eljárás outputja az ***output.lst*** nevű fájlba kerül. A PROC PRINTTO nem készít outputot, csak a többi eljárás eredményét irányítja át.

A LOG ablak tartalmát is irányíthatjuk fájlba a

```
PROC PRONTTO LOG=xxx ;
```

utasítással.

VISSZAÁLLÍTÁS

Az átirányítást visszaállítani az eredetire (OUTPUT ablak) a következőképpen lehet:

```
PROC PRINTTO;
RUN;
```

AZ X PARANCS

Az X parancsot arra használhatjuk, hogy ideiglenesen felfüggesszük a SAS futást és visszatérjünk az operációs rendszerhez.

```
Command ==> X
```

Az operációs rendszer utasítását követve térhetünk vissza a SAS-hoz.

AZ X UTASÍTÁS

Az operációs rendszer parancsai is végrehajthatók programutasításokként az X utasítás segítségével:

```
PROGRAM EDITOR ____________________________

00001 X 'Operációs rendszer parancsa' ;
00002
```

P5.4 SAS ADATÁLLOMÁNYOK OLVASÁSA

Szükségünk lehet az adatkönyvtárakban tárolt SAS adatállományok módosítására, vagyis:
- változók hozzáadására
- változók módosítására
- a megfigyelések egy részhalmazára
- a változók egy részhalmazára
- a megnevezések, hosszak, input és output formátumok megváltoztatására.

A SET UTASÍTÁS

Ha már egyszer létrehoztunk egy SAS adatállományt, akkor esetleg szeretnénk inputként használni egy további DATA lépésben – sok adatkezelési funkció működik ilyen módon. A SET utasításban több név megadásával több adatállományt lehet kombinálni.

Pl.
```
DATA libref.sasdata_2 ;
  SET libref.sasdata_1 ;
  ujvalt=valt1+valt2 ;
RUN;
```

A futtatáskor végbemenő folyamat négy szakaszra osztható:

1. A DATA lépés kezdete
2. A SAS adatállomány(ok) beolvasása (egyszerre egy megfigyelés)
3. A műveletek végrehajtása az adott megfigyelésen
4. A megfigyelés kiírása az output SAS adatállományba.

A vezérlés mindaddig visszakerül a DATA lépés elejére, amíg az utolsó megfigyelést is be nem olvastuk.

MEGJEGYZÉS

Ha az input egy SAS ADATÁLLOMÁNY:

INFILE SINCS
INPUT SINCS
A SET A KINCS

(NO INFILE
NO INPUT
SET DOES THE LOT)

P5.5 GYAKORLATOK

1. FELADAT

Tekintsük a ***TANF\SASOK*** könyvtárat a saját SAS könyvtárunknak és küldjük el egy LIBNAME utasítást ***TANF*** LIBREF-fel erre a könyvtárra hivatkozva!

2. FELADAT

Ellenőrizzük a LIBREF-et a LIBNAME ablakban, ill. EXPLORER ablakban! Vizsgáljuk meg, hogy milyen ’élő’, a SAS rendszer által generált LIBREF-ek vannak!

3. FELADAT

Hozzuk létre a korábban ideiglenesen létrehozott ***SZEMELY*** nevű SAS állományt az 1. Feladatban definiált könyvtárba! Menjünk ki az operációs rendszerbe és nézzük meg a létrehozott állomány teljes nevét!

4. FELADAT

Az ideiglenes állományként betöltött ***HIVATAL*** állományunkat (ha nincs meg, töltsük be újból a ***HIVATAL.SAS*** program lefuttatásával) hozzuk létre állandó SAS állományként az első feladatban definiált könyvtárba másolással.

P6 FEJEZET

ADATKEZELÉS

6.1 A DATA LÉPÉS

6.2 VÁLTOZÓK MÓDOSÍTÁSA

6.3 ADATOK RÉSZHALMAZÁNAK KIVÁLASZTÁSA

6.4 A DATA LÉPÉS VEZÉRLÉSE

6.5 GYAKORLATOK

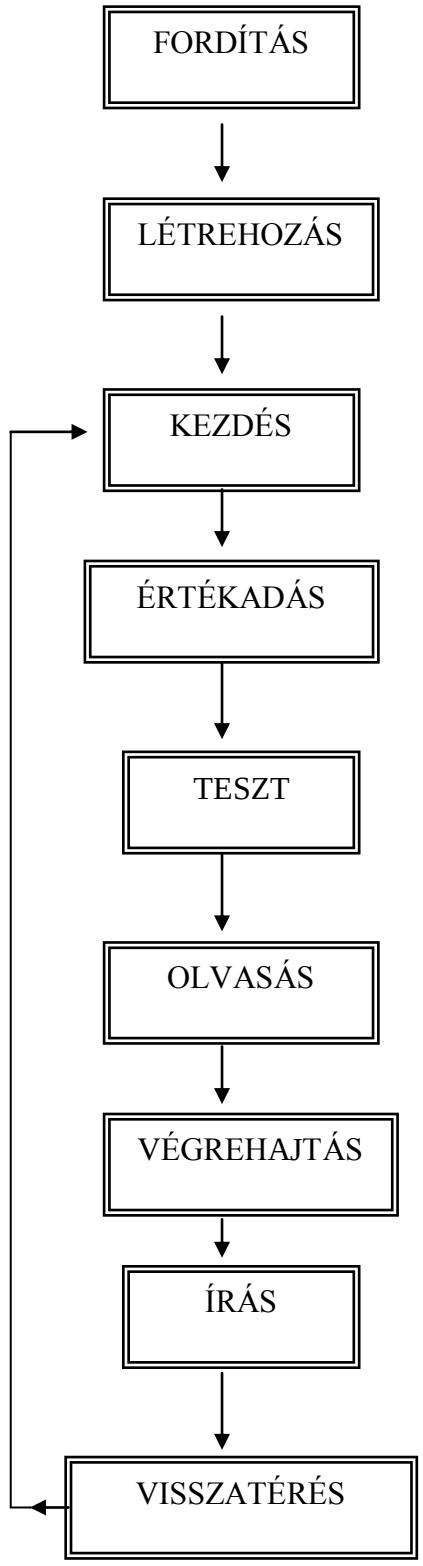
P6.1 A DATA LÉPÉS

Eddig láthattuk, hogy a DATA lépésben lehet külső adatokat olvasni és SAS adatállományt létrehozni. A DATA lépés utasításai 4. Generációs programnyelvet alkotnak. Valóban, a DATA lépés nyelve nagyon hasznos eszköz az adatkezelésben.

Ebben a fejezetben részletesebben megnézzük a DATA lépést, különös tekintettel a SAS nyelv néhány olyan részére, mint az értékadás, részhalmaz képzés és feltételes végrehajtás (és persze még a függvények, összegzések és hiányzó értékek). Jelen fejezetünk fő témája az adatkezelés.

Láthattuk, hogy a DATA lépés tartalmaz egy implicit ciklust. A 3. Fejezet ennek a műveletsorozatnak egy részét leírta, amint minden egyes rekordot elolvas a rendszer, végrehajtódnak az előre definiált utasítások és az eredmények egy-egy SAS megfigyelésként kiíródnak egy SAS adatállományba.

A 3.4 fejezetben látott diagram található a következő oldalon is:



FORDÍTÁSI FÁZIS
SAS utasítások szintaktikai ellenőrzése

Input puffer, a program adatvektor (PDV) létrehozása, adatállomány leíró részének megírása

VÉGREHAJTÁSI FÁZIS
DATA utasítás, iteráció számlálása

A változók hiányzó értékre állítása a PDV-ben

LEZÁRÁS Van még rekord?

Input rekord

SAS utasítás

Megfigyelés (OBSERVATION) az adatállományba

Vissza az elejére

A FORDÍTÁSI FÁZIS

FORDÍTÁS:

A DATA lépés első tevékenysége a SAS utasítások szintaktikai ellenőrzése. Ha bármilyen hibát talál a rendszer, hibaüzenetet ír a LOG-ba, és a DATA lépés leáll. Különben a DATA lépést lefordítja az adott gép gépi kódjára.

LÉTREHOZÁS

A 2. Művelet az input puffer és a program adatvektor (PDV) létrehozása. Ebben a fázisban még nem történik külső adat olvasása, semmilyen számítás nem hajtódik végre, és egyetlen megfigyelés sem kerül outputra a SAS adatállományba. Ekkor készül el azonban a SAS adatállomány leíró része.

A DATA STEP BELSEJE

Ha belelátnánk a rendszer belsejébe, láthatnánk az input puffert és a PDV-t, ami a DATA lépéshez keletkezik.

AZ INPUT PUFFER

----¦----¦10--¦----¦20--¦----¦30--¦--…<- - ->…--¦----¦80

A PROGRAM ADATVEKTOR (PDV)

Változó1 változók _N_ _ERROR_

A CARDS vagy DATALINES utasítás 80 karakter hosszú input puffert definiál. Az ennél hosszabb rekordokat INFILE utasítással olvashatjuk be.

A PDV a DATA lépésben definiált változókat tartalmazza, de nem csak az INPUT utasításban felsoroltakat. Az _N_, ami azt jelöli, hányszor kezdődött el a DATA lépés, és az _ERROR_, ami az előforduló adathibákat jelzi (ez egy flag), szintén bekerül a PDV-be.

A DATA lépés STOP; utasítással való megállítása az adatrekordok írása illetve olvasása előtt nagyon hasznos lehet. Ezt a technikát tesztelésre használhatjuk nagy adatállományok esetén. Hogy ellenőrizzük, hogy a megfelelő változók, típusok, hosszak keletkeznek-e, mielőtt ténylegesen beolvasnánk egyetlen adatot is.

Ebben a fejezetben látni fogunk néhány utasítást, amelyek inkább fordítási utasítások, mintsem végrehajthatók. Más szóval, ezek információval szolgálnak a SAS compiler-nek, és a DATA lépésben keletkező változókat befolyásolják. A fordítási utasítások nem „hajtódnak végre” a DATA lépéssel, így más SAS utasításokkal nem vezérelhetők.

A VÉGREHAJTÁSI FÁZIS

Amint a fordítási fázis befejeződött, kezdődik a végrehajtási fázis; ekkor a külső adatokat beolvassa a rendszer egy SAS adatállományba, rendszerint egyszerre egy rekordot, és a PDV-t a DATA lépés utasításaihoz rendeli.

KEZDÉS

A DATA utasítás az opciókkal együtt végrehajtódik. Jelzi, hogy a DATA lépés számolja a ciklus elejére való visszatérést. Ezt a **_N_** nevű automatikus változóba tölti.

Az _N_-et a SAS supervisor állítja be, hogy vezérelje a DATA lépést. 0-ról indul és eggyel növekszik mindig a DATA lépés elején. Ez nagyon hasznos dolog, mivel ugyanúgy tesztelhetjük ezt a változót, mint bármely másikat.
Pl: az

```
IF _N_=1 THEN STOP ;
```

Utasításnak ugyanaz a hatása, mint a DATA… ; utasítás után közvetlenül beírt **STOP;** utasításnak. A DATA lépés első végrehajtásakor az _N_ értéke 1 lesz, és a program végrehajtása befejeződik.

ÉRTÉKADÁS

A következő a változók hiányzó értékre állítása a PDV-ben. Ez nem vonatkozik, azokra a változókra, amelyek RETAIN („megmarad”) utasításban szerepelnek, amelyeket a SET, MERGE, vagy UPDATE utasítással olvasunk be, vagy az _N_ és az _ERROR_ nevű automatikus változókra.

PROGRAM ADATVEKTOR

Az automatikus változók megmaradnak (RETAIN) és a kezdőértékük 0.

A karakteres változók hiányzó értéke szóköz (blank), míg a numerikusoké pont (.) .

A változók hiányzó értéket kapnak valahányszor a DATA lépés újrakezdődik, ha külső adatok kerülnek beolvasásra. Ennek megakadályozását egy későbbi fejezetben tárgyaljuk.

TESZT

Itt megnézi a rendszer, hogy a fájl-vége jelet elérte-e már. Ha igen, akkor lezárja az adatállományt és a DATA lépés befejeződik. Ha nem, akkor továbbmegy az „olvasásra”.

OLVASÁS

A külső adatrekordot beolvassa a rendszer az input pufferbe és átviszi a PDV-be. A változók olyan sorrendben kerülnek be a PDV-be, amilyen sorrendben a DATA lépésben találhatók.

VÉGREHAJTÁS

A PDV-ben lévő külső adatokból származó értékekkel a DATA lépés többi utasítása is végrehajtódik.

ÍRÁS

A DATA lépés végén vagy egy OUTPUT utasítást elérve, a PDV tartalma egy megfigyelésként kiírásra kerül a SAS adatállományba. Az automatikus változók nem kerülnek outputra.

VISSZATÉRÉS

A DATA lépés végén, illetve egy RETURN utasítást elérve, a vezérlés visszakerül a DATA lépés elejére és elölről kezdődik a folyamat.

A fejezet további részében a DATA lépés vezérléséről lesz szó. Megnézzük, hogyan kap értéket, hogyan változik egy változó, és hogyan kell a DATA lépésben szereplő adatok részhalmazát kiválasztani.

P6.2 VÁLTOZÓK MÓDOSÍTÁSA

A DATA lépés alkalmazható új változók hozzáadására és meglévők módosítására is.

A SET UTASÍTÁS

Ahhoz, hogy egy új változót adjunk egy adatállományhoz, be kell olvasnunk azt egy DATA lépésben, és az új változónak értéket kell adnunk egy értékadó utasítással.

SAS adatállományok olvasására a SET utasítás használatos:

```
DATA új_sasdata ;
  SET libref.sasdata ;
RUN ;
```

A DATA lépés beolvassa a FILE1 adatállományt és létrehozza az UJFILE nevű SAS adatállományt.

Az UJFILE adatállomány pontosan ugyanaz, mint a FILE1, valójában egy másolatot készítettünk

VÁLTOZÓK HOZZÁADÁSA

Egy adatállományhoz nagyon egyszerűen adhatunk új változókat: értékadó utasítással létrehozunk egy új változót.

Számítsuk ki pl. egy VALT1 nevű változóban méterben megadott távolságot yardban is.

```
ujvalt= valt1*1,094;
```

Mi történik a PDV-ben? (Struktúrája kiegészül az UJVALT változóval, melybe belekerül a számítás eredménye.)

VÁLTOZÓK MÓDOSÍTÁSA

Létező változók értékeit ugyanúgy módosíthatjuk:

```
valt=valt*1,094 ;
```

Mi történik a PDV-ben? (Struktúrája nem változik. Tartalmilag: a már beolvasott VALT érték felülíródik a számítás eredményével.)

KIFEJEZÉSEK TÍPUSAI

Sok különböző típusú kifejezést írhatunk ilyen módon:

```
A = B + C ;
A = B – C ;
A = B * C ;
A = B / C ;
A = B ** C ;
A = - B ;
A = 0 ;
A = ’Férfi’ ;
A = A + B + C ;
A = ( B + C ) / D ;
A = SQRT ( B ) ;
```

A kiértékelés sorrendje: () ** * / + - , illetve balról jobbra a kétesélyes helyzetekben.

FÜGGVÉNYEK

A függvények, mint pl. a négyzetgyök (SQRT) egy későbbi fejezetben szerepelnek majd.

VÁLTOZÓK ÁTNEVEZÉSE

A változók nevét a **RENAME** utasítással változtathatjuk meg.

Szintaktikája: **RENAME *régi_név=új_név ;***

Ezen kívül még a **PROC DATASETS** is használható változók nevének megváltoztatására.

P6.3 ADATOK RÉSZHALMAZÁNAK KIVÁLASZTÁSA

Nagyon sokféleképpen választhatunk ki megfigyeléseket egy SAS adatállományból, a legáltalánosabb módszer a **kiválasztó IF** utasítás.

KIVÁLASZTÓ IF

```
IF  SAS_kifejezés ;
```

A **kiválasztó IF** utasítás kapuként működik, csak azokat a megfigyeléseket ereszti át, amelyek kielégítik a feltételeit. Tehát **csak azok kerülnek bele a keletkező adatállományba, amelyeknél a feltétel igaz.**

OPERÁTOROK

Az IF utasításban nagyon sokféle operátort használhatunk:

EQ	**v**agy =	Egyenlő
NE	^=	Nem egyenlő
LE	<=	Kisebb vagy egyenlő
LT	<	Kisebb
GE	>=	Nagyobb vagy egyenlő
GT	>	Nagyobb

LOGIKAI OPERÁTOROK

AND	**&**	Logikai ÉS
OR	¦	Logikai VAGY
NOT	^	Logikai NEM.

Megjegyzés: Mindkét esetben akár a jelek, akár a betűk használhatók.

A WHERE UTASÍTÁS

A WHERE (ahol) utasítás közvetlenül **a beolvasandó fájlra vonatkozik**, és nem lehet vezérelni, mint a **kiválasztó IF** utasítást, azaz mindenképpen érvényre jut.

A WHERE OPCIÓ

A WHERE adatállomány név opciót szintén használhatjuk:

Pl.
```
SET tanf.demograf(WHERE=(nem = 'F')) ;
```

- hatására csak a férfiak rekordjai kerülnek beolvasásra.

A WHERE AZ ELJÁRÁSOKKAL EGYÜTT

A WHERE utasítást használhatjuk PROC lépésben is, hogy leválasszuk az elemzéshez szükséges adatokat.

Pl.
```
PROC PRINT DATA=tanf.demograf ;
  WHERE nem = 'N' ;
```

Mi a különbség a kiválasztó IF és a WHERE között?

P6.4 A DATA LÉPÉS VEZÉRLÉSE

FELTÉTELES VÉGREHAJTÁS

Az **IF** utasítást használhatjuk a **THEN** és az **ELSE** utasításokkal együtt a SAS utasítások végrehajtásának vezérlésére a DATA lépésben. A **DO ; … END ;** utasításokat is hozzávéve akárhány SAS utasítást vezérelhetünk, és végrehajthatjuk őket többféle feltétel szerint.

HATÁKONYSÁGI SZEMPONTOK

IF… THEN… ; utasítások sorozata helyett sokkal hatékonyabb az **ELSE** utasítást használni, ahol csak lehet, különösen, ha sok megfigyelésünk van.

Általános alakban:

```
IF feltétel_1  THEN utasítás_1 ;
ELSE IF feltétel_2 THEN utasítás_2 ;
ELSE IF feltétel_2 THEN utasítás_3 ;
```

DO – END BLOKKOK

Bármilyen SAS utasítás írható a THEN után egy IF-THEN szerkezetben, de ha a feltételtől függően több utasítást akarunk végrehajtani, akkor a DO – END utasítás-zárójeleket kell használnunk.

Általános formája a következő:

```
DATA fájlnév ;
      IF    feltétel          THEN
          DO ;
            SAS_utasítás ;
            SAS_utasítás ;
            SAS_utasítás ;
            SAS_utasítás ;
            SAS_utasítás ;
            …
            END ;
         ELSE DO ;
            SAS_utasítás ;
            SAS_utasítás ;
            SAS_utasítás ;
            …
            END ;
RUN ;
```

A SELECT UTASÍTÁS

A SELECT utasítása, pontosabban SELECT-blokk logikusabb írásmód az IF-THEN-DO.END szerkezetnél. Egy programozó szemszögéből nézve világosabb, mint a régimódi IF-THEN-ELSE.

Általánosan:

```
SELECT (változó) ;
  WHEN (érték_1)  utasítás_1 ;
  WHEN (érték_2)  utasítás_2 ;
  …
  OTHERWISE  utasítás_n ;
END ;
```

MEGFIGYELÉSEK KIVÁLASZTÁSA A LÉTREHOZANDÓ (OUTPUT) ÁLLOMÁNYBA

Sokféle módja van, hogy megmondjuk, mi történjen egy megfigyeléssel a SAS DATA lépésében. Már láttuk, hogyan lehet egy kiválasztó IF utasítást használni olyan feltétel vizsgálatára, amely alapján vagy töröljük vagy megtartjuk a megfigyelést a teszt eredményétől függően.

Pl.:

```
IF ter NE 9 THEN DELETE ;
```

vagy

```
IF ter = 9 ;
```

Mindkét utasításnak ugyanaz a hatása, mindkettő csak a 9 TER értéket tartalmazó megfigyeléseket engedi kiírni az output SAS adatállományba.

GO TO UTASÍTÁS

Hatására a SAS a címkében meghatározott helyre ugrik és ott folytatja a végrehajtást. Az utána használt RETURN utasítás hatására a SAS visszatér a DATA lépés elejére. A címkének a DATA lépésen belül kell lennie.
Formája:

```
GOTO címke ;      vagy      GO TO címke ;
```

LINK UTASÍTÁS

Hatására a DATA lépés következő utasítása helyett egy címkével azonosított utasításon (utasítássoron) folytatódik a feldolgozás egészen egy RETURN utasításig. A RETURN visszaadja a vezérlést a a LINK-et követő utasításnak.
Formája:

```
LINK  címke ;
```

RETURN UTASÍTÁS

Hatására a SAS megszakítja a DATA lépés utasításainak végrehajtását és máshol folytatja a feldolgozást.
- LINK utasítás esetén a LINK utasítás utáni utasítás kapja meg a vezérlést
- A FILE utasítás HEADER= opciójához tartozó utasítások RETURN-je az utolsó végrehajtott utasítás utáni utasításnak adja át a vezérlést. (Ld. később)
- Minden más esetben a DATA lépés elejéről egy új iterációval folytatódik a feldolgozás.

Minden DATA lépés utolsó utasítása egy implicit RETURN utasítás.

OUTPUT UTASÍTÁS

Az **OUTPUT** utasítással irányítjuk a megfigyelést kiírásra a SAS adatállományba.

Szintaktikája: **OUTPUT** ***libref.sasdata*** **;**

A fejezet elején lévő ábra „ÍRÁS doboza” azt jelezte, hogy a megfigyelés kiírása a DATA step végén történik. Alapértelmezés szerint ez igaz is. Ha nem írunk OUTPUT utasítást, akkor a rendszer a DATA lépés végén feltételez egyet.

Viszont ha írunk OUTPUT utasítást, akkor a PDV tartalma akkor kerül kiírásra, amikor OUTPUT utasítás következik a folyamatban.

Amikor egy OUTPUT utasításhoz érünk:

====>> NINCS AUTOMATIKUS OUTPUT A DATA STEP VÉGÉN

====>> A VEZÉRLÉS NEM KERÜL VISSZA A DATA LÉPÉS ELEJÉRE AZ OUTPUT UTASÍTÁS UTÁN

Sokszor használjuk az OUTPUT utasítást:

- HA TÖBB MEGFIGYELÉST HOZUNK LÉTRE EGY INPUT REKORDBÓL

- HA TÖBB SAS ADATÁLLOMÁNYT GYÁRTUNK EGY DATA LÉPÉSBEN

VÁLTOZÓK KIVÁLASZTÁSA AZ OUTPUT ÁLLOMÁNYOKHOZ

A SAS adatállományba írandó változók kiválasztásának lehetősége is nagyon fontos. Ezt a **DROP** (kihagy) és a **KEEP** (megtart) utasításokkal és opciókkal tehetjük meg.

Formájuk:
```
DROP valt_1 valt_2 … valt_n ;
KEEP valt_1 valt_2 … valt_n ;
```

A DROP utasítás a SAS supervisor-nak ad át információt fordítási időben, ezért akárhol előfordulhat a DATA lépésen belül. Csak a keletkező adatállományokra van hatása, a beolvasottakra nincs.

Programozás során eldönthetjük, hogy melyik utasítást célszerű választanunk ahhoz, hogy kevesebbet kelljen írnunk. A lényeg, hogy egymásnak ellentmondó rendelkezéseket ne tegyünk!

VÁLTOZÓK KIHAGYÁSA AZ OUTPUTBÓL

Ahogyan meghatározhatjuk, hogy melyik változók maradjanak ki vagy kerüljenek bele az összes, a DATA utasításban felsorolt állományba, úgy adatállományonként is rendelkezhetünk róla.
Ezt ugyancsak a KEEP és DROP adatállomány opciókkal tehetjük meg.

Pl.
```
DATA  elso   (KEEP = V1 V2 V3)
      masod  (KEEP = V1 V4)
      harmad (KEEP = V1 V3 V4) ;
```

HATÉKONYSÁGI SZEMPONTOK

Sokkal hatékonyabb megtartani vagy kihagyni a változókat, mielőtt bekerülnének a PDV-be. **A DROP és a KEEP opciókat használhatjuk a SET utasításban is** ugyanúgy, mint a DATA utasításban, de vegyük figyelembe, hogy ebben az esetben ezek a változók nem kerülnek beolvasásra az aktuális PDV-be, így nem is használhatók más programutasításokban sem a DATA lépésben.

P6.5 GYAKORLATOK

1. FELADAT

Bővítsük ki a ***TANF\SASOK*** könyvtárban található ***SZEMELY*** nevű adatállományunkat a KOR változóval, melyet számítsunk ki a SZULEV - ből (születési év). Nyomtassuk ki a 30 évnél idősebbek adatait.

2. FELADAT

Hozzunk létre a ***SZEMELY*** állományunkban három korcsoportot, KCSOP néven .
KCSOP=1 értéket kapjanak a 30 évesek és annál fiatalabbak.
KCSOP=3 értéket kapjanak a 60 évnél idősebbek.
A többieknél KCSOP legyen 2. Ellenőrizzük, hogy jól dolgoztunk-e.

3. FELADAT

Hozzunk létre ideiglenes állományokat a személyeket családi állapotuk szerint különválasztva. Nevezzük el az állományokat a családi állapot szerint. A SZEM, MAGAS, SZULEV változókat hagyjuk ki mindegyik állományból. Egyetlen DATA lépésben hajtsuk végre a feladatot és írjuk ki mindegyik állományt úgy, hogy az oszlopok fölött ne a mezőnevek szerepeljenek. Írjuk ki mindegyik állományt úgy, hogy a listákból látszódjék, melyikről is készültek. Tudunk-e több megoldást a feladat elvégzésére?

4. FELADAT

Mi történik, ha a kiíró lépések közül elhagyjuk a RUN utasítást? Próbáljuk ki!

P7 FEJEZET

ÖSSZEGZÉSEK ÉS FÜGGVÉNYEK

7.1 FÜGGVÉNYEK

7.2 ÖSSZEADÁS ÉS ÖSSZESENKÉPZÉS

7.3 GYAKORLATOK

P7.1 FÜGGVÉNYEK

A SAS rendszerben függvények tucatjai találhatók. Ezek olyan rutinok, amelyek egy értéket szolgáltatnak 0, egy vagy több argumentum alapján. Általános formájuk:

```
függvény_név (arg_1,arg_2, … arg_n) ;
```

Néhány függvényt megnézünk itt, a többit a tanfolyam később részében.

A ROUND (KEREKÍTÉS) FÜGGVÉNY

Ez a függvény használható értékeknek a legközelebb eső kerekítési egységre történő kerekítésére.

Pl. ha a teszt1 változó értéke 194.916 akkor a

```
A=ROUND(teszt1,0.1) ;
```
utasítás eredménye a=194.9

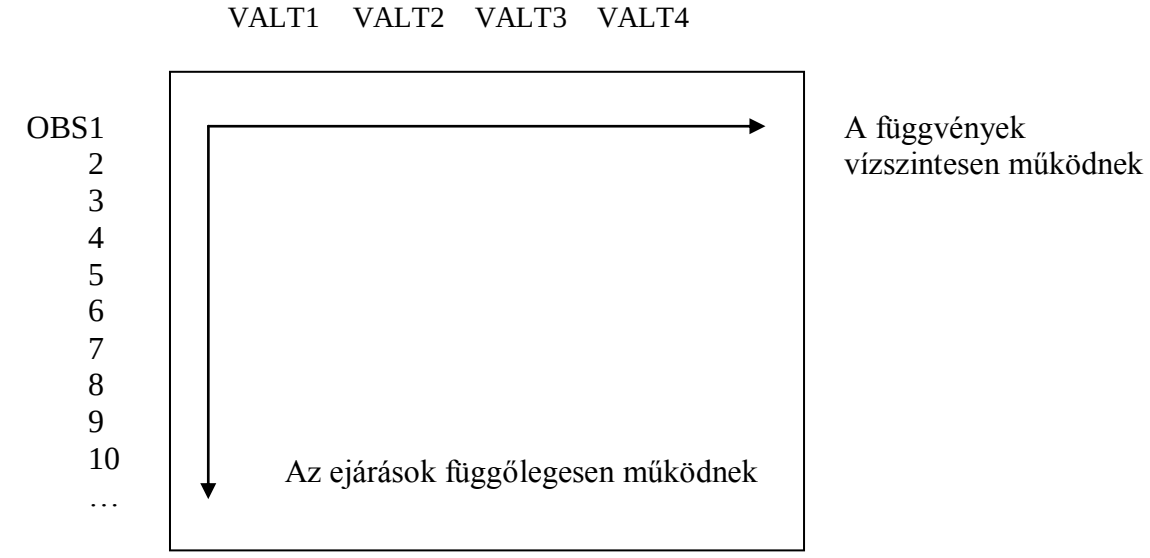
```
B=ROUND(teszt1,10) ;
```
utasítás eredménye b=190

```
C=ROUND(teszt1,1)-1 ;
```
utasítás eredménye c=194

```
D=ROUND(teszt1,10)-0.1 ;
```
utasítás eredménye d=189.9

STATISZTIKAI FÜGGVÉNYEK

A MIN, MAX és STD (szórás) függvények megfigyelésenkénti statisztikák kiszámítására valók, nem pedig változónkénti számításokra. Például a PROC MEANS egy változónak az átlagát adja az összes megfigyelésre. Más szóval a PROC MEANS az átlag költséget, az átlagos hosszt vagy az átlagos súlyt számítja ki az összes megfigyelést tekintetbe véve.
A statisztikai függvények viszont a megfigyelés összes változójára tudnak statisztikát számolni:



A SUM (ÖSSZEGEZŐ) FÜGGVÉNY

A SUM függvénnyel egy megfigyelésben lévő értékeket összegezhetünk

Pl
```
valt1=SUM(valt2,valt3,-valt4) ;
```

A SUBSTR FÜGGVÉNY

Karakteres változók részeit emelhetjük ki a nagyon hasznos SUBSTR függvénnyel, amely kétféleképpen használható.

1. értékadásra

```
nev='William Shakespeare' ;
kereszt=SUBSTR(nev,1,7) ;
vezetek=SUBSTR(nev,9,11) ;
```

2. szövegcserére

```
nev='William Shakespeare' ;
SUBSTR(nev,1,7)='  Bill ';
```

A SCAN FÜGGVÉNY

A SCAN függvény is karaktersorozatból emel ki szavakat, de akkor használjuk, ha a kezdőpozíció ismeretlen.

Tegyük fel, hogy a következő karakteres értékünk van:
KONYVCIM: A Manderley-ház asszonya

```
egyik=SCAN(konyvcim,1) ;          ===>  egyik='A'
masik=SCAN(konyvcim,4) ;          ===>  masik='asszonya'
```

A SCAN függvény első argumentuma a karakteres változó, a második pedig a kiválasztandó szó sorszáma.
Létezik egy harmadik argumentum is, az elhatároló jel, amelyet nem kötelező megadnunk.

Az **alapértelmezés szerinti elhatároló jelek** a következők:

Szóköz . < (+ ¦ & ! $ * ; ^ - / , %

Csak a kötőjelet megadva elhatároló jelnek:

```
egyik=SCAN(konyvcim,1,'-' ) ;      ==>   egyik='A Manderley'
masik=SCAN(konyvcim,2,'-') ;       ==>   masik='ház asszonya"
```

AZ INDEX FÜGGVÉNY

Az INDEX függvény egy karaktersorozat helyét keresi meg egy megadott szövegben.

KONYVCIM = 'A Manderley-ház asszonya' ;

Nézzük meg, hogy az 'and' jelsorozat szerepel-e a fenti szövegben:

```
poz=INDEX(konyvcim,'and') ;
```
===> poz=4

Az INDEX függvény a keresett string kezdőpozícióját adja vissza.
Ha a karaktersorozatot nem találja, az értéke 0 lesz.

UPCASE FÜGGVÉNY

Az UPCASE függvény a szöveget nagybetűssé konvertálja; ez nagyon hasznos az összehasonlításoknál:

KONYVCIM: A Manderley-ház asszonya

```
nagy=UPCASE(konyvcim) ;
```
==> nagy='A MANDERLEY-HÁZ ASSZONYA'

Ezt akkor tudjuk jól kihasználni, ha az input lehet kis és nagybetűs, és nekünk úgy kel vizsgálnunk az input értékét, hogy a kis- és nagybetűs szöveget nem különböztetjük meg.

A SAS RENDSZER FÜGGVÉNYEINEK LISTÁJA

Az alábbiak az alap SAS szoftver függvényei, továbbiak találhatók még a SAS rendszer többi termékében.

Aritmetikai függvények		Karakterkezelő függvények	
ABS	Abszolút érték	BYTE	
DIM	Tömb mérete	COLLATE	
HBOUND	A tömb felső határa	COMPRESS	Karakterek elvétele
LBOUND	A tömb alsó határa	INDEX	Szövegrész keresése
MAX	Legnagyobb érték	INDEXC	Első előfordulás
MIN	Legkisebb érték	LEFT	Balra igazítás
MOD	Maradék	LENGHT	Szöveg hossza
SIGN	Előjel	RANK	Karakterek pozíciója
SQRT	Négyzetgyök	REPEAT	Karakterek ismétlése
		REVERSE	Inverzió
Csonkító függvények		RIGHT	Jobbra igazítás
TRUNC	Csonkítás	SCAN	Szavak keresése
CEIL	Nagyobb egész	SUBSTR	Egy rész kivétele
INT	Egész rész	TRANSLATE	Karakterek cseréje
FLOOR	Kisebb egész	TRIM	Záró blankek törlése
FUZZ		UPCASE	Nagybetűs konverzió
ROUND	Kerekítés	VERIFY	Megerősítés, igazolás
Matematikai függvények		Dátum és idő függvények	
DIGAMMA, ERF, ERFC, EXP, GAMMA		DATE, DATEJUL, DATEPART,	
LGAMMA, LOG, LOG2, LOG10,		DATETIME, DAY, DHMS, HMS, HOUR,	
ORDINAL, TRIGAMMA		INTCK, INTNX, JULDATE, MDY,	
		MINUTE, QTR, SECOND, MONTH, TIME,	
Trigonometriai függvények		TIMEPART, TODAY, WEEKDAY, YEAR,	
ARCOS, ARSON, ATAN, COS, COSH,		YYQ	
SIN, SINH, TAN, TANH			
		Speciális függvények	
Valószínűségi függvények		DIF, INPUT, LAG, PUT, SASVER,	
POISSON,		SYMGET, SOUND, ERF, ERFC	
PROBBETA, PROBBNML, PROBCHI,			
PROBF, PROBGAM, PROGHYPR,		Kvantilis függvények	
PROBNEGB, PROBNORM,		BETAINV, CINV, FINV, GAMINV, TINV	
PROBT		PROBIT,	
		Pénzügyi függvények	
Statisztikai függvények		DEPDB, DEPDBSL, DEPSL, DEPSYD,	
CSS, CV, KURTOSIS, MAX, MEAN, MIN, N,		DEPTAB, COMPOUND, DACCTAB, INTRR,	
NMISS, RANGE, SKEWNESS, STD,		IRR, DACCDC, DACCDBSL, DACCSL,	
STDERR, SUM, USS, VAR		DACCSYD, MORT, NETPV, NPV, SAVING	
Véletlenszám függvények			
NORMAL, RANBIN, RANCAU, RANEXP,			
RANGAM, RANNOR, RANPOI, RANTBL,			
RANTRI, RANUNI, UNIFORM			

P7.2 ÖSSZADÁS ÉS ÖSSZESENKÉPZÉS

Összesenek felgyűjtése az adatállományból egy speciális utasításformát követel. Az összeadás (ellentétben a SUM függvénnyel) használható erre a feladatra a **RETAIN** utasítással együtt, és nagyon széleskörűen alkalmazható.

AZ EREDMÉNY HIÁNYZÓ ÉRTÉK

Példa: számoljuk meg a beolvasott rekordok számát.

```
darabsz= darabsz+1 ;
```

1. probléma:

A DARABSZ a DATA lépés elején minden iterációnál hiányzó értéket kap kezdőértékként, így az eredmény mindig hiányzó érték lesz

(**hiányzó érték + valami = hiányzó érték**).

Valamilyen módon meg kell mondanunk a DATA lépésnek, hogy szeretnénk megtartani a változó értékét egyik megfigyeléstől a másikig. Ezt a **RETAIN** utasítással tehetjük meg:

```
RETAIN darabsz 0 ;
```

Ez azt is jelenti, hogy 0 kezdőértéket szeretnénk adni a változónak.

2. probléma

A hiányzó értékek befolyásolják az aritmetikai kifejezéseket.

Tehát `A = A + B;` esetén ha pl. **B** valamelyik megfigyelésben hiányzó érték, **A** értéke is az lesz.

Hogy kikerüljük a problémát, használjuk az összegző utasítást:
Ki fogjuk cserélni a következő utasításokat:

```
darabsz=darabsz+1 ;          ===>  darabsz+1
A = A +B ;                   ===>  A + B ;
```

Ugyanaz a hatásuk, csak a hiányzó érték nem befolyásolja az eredményt. Az összegző utasítás az 1. problémát is megoldja.
A kivonás az `A + (-B) ;` formában oldható meg.

Megjegyzések:

Bevált programozói gyakorlat az összegző utasítás **(a+b) ;** használata a szokásos **a=a+b ;** forma helyett, mivel így a hiányzó érték nem befolyásolja az aritmetikai kifejezéseket. Az összegző utasítás mechanizmusa egy beépített RETAIN-t is tartalmaz, így a RETAIN utasítást kitörölhetjük a programból. Az összegző utasítás 0 kezdőértéket is generál.

P7.3 GYAKORLATOK

1. FELADAT

A ***TANF\SASOK*** könyvtárban lévő ***HIVATAL*** állomány NEV mezőjét alapul véve bővítsük ki az állományt egy VNEV (vezetéknév) és egy KNEV (keresztnév) változóval.

2. FELADAT

Az előbbi feladatot folytatva hozzunk létre egy 4 karakter hosszú AZON nevű mezőt a ***HIVATAL*** állományunkban, mely a hivatali osztály azonosító kódját és a dolgozó nevének kezdőbetűit tartalmazza. Ellenőrzésképpen írjuk ki a OSZTALY, NEV, VNEV, KNEV, AZON nevű változókat. (Segítségül: a konkatenálás jele a ||)

3. FELADAT

Rendeljünk a ***HIVATAL*** állományunkban az egyes dolgozókhoz az AZON mező szerinti sorrendben egy sorszámot. Csak az AZON, NEV és SORSZ mezőket listázzuk ki.

4. FELADAT

A ***HIVATAL*** állományból számítsuk ki a dolgozók várhatóan emelt fizetését és bővítsük ki vele állományunkat. Az emelés mértékét %-ban a SZORZO mező tartalmazza. A számításnál kerekítsünk egész számra. Írassuk ki az AZON, FIZETES, SZORZO, UJFIZ változókat.

P8 FEJEZET

PROBLÉMAMEGOLDÁS

8.1 HIÁNYZÓ ÉRTÉKEK

8.2 ÉRTÉKADÁS ÉS ÖSSZEHASONLÍTÁS

8.3 SZINTAKTIKAI ÉS ADATHIBÁK

8.4 PROBLÉMAMEGOLDÁSOK

8.5 GYAKORLATOK

P8.1 HIÁNYZÓ ÉRTÉKEK

MIK A HIÁNYZÓ ÉRTÉKEK?

Az adatgyűjtések kérdőívei ritkán olyan jól kitöltöttek, mint ahogyan szeretnénk. Például némelyik adatfelvételben lehetnek „lyukak”, ahol a megkérdezettek nem válaszoltak minden kérdésre. Adatstruktúrákat tervezhetünk úgy, hogy bizonyos mezők lehetnek „nem kitöltöttek”. Ezeket az adatokban lévő lyukakat nevezzük hiányzó értékeknek (missing values). A SAS rendszer különösen erős és következetes a hiányzó értékek kezelésében.

A karakteres hiányzó értékek szóközökként tárolódnak, a numerikus hiányzó értékek pontokként jelennek meg.

HIÁNYZÓ ÉRTÉKEK AZ ARITMETIKAI KIFEJEZÉSEKBEN

A HIÁNYZÓ ÉRTÉKEK BEFOLYÁSOLJÁK AZ ARTIMETIKAI KIFEJEZÉSEKET!!!

Bármilyen aritmetikai kifejezés, amelyik hiányzó értékeket tartalmaz, hiányzó értéket ad eredményül.

Ilyenkor ***NOTE: Missing values were generated as a result of performing an operation on missing values.*** üzenet jelenik meg a LOG ablakban.

HIÁNYZÓ ÉRTÉKEK A FÜGGVÉNYEKBEN

Nagyon sok hiányzó értékekből eredő problémát megoldhatunk függvényekkel. A SAS rendszer függvényei úgy kezelik a hiányzó értékeket, mintha ott sem lennének.

A SUM FÜGGVÉNY SZINTAKTIKÁJA

```
Változó = SUM(változó1, változó2, stb…) ;
```

SAS ELJÁRÁSOK ÉS HIÁNYZÓ ÉRTÉKEK

Az egyes SAS eljárások különbözőképpen kezelik a hiányzó értékeket. Többnyire kihagyják a hiányzó értékeket az elemzésből. Néha viszont szükségünk lehet arra, hogy ismerjük a hiányzó értéket tartalmazó megfigyelések számát; például a PROC UNIVARIATE ezt megmondja.

Más, bonyolultabb statisztikai eljárások figyelembe veszik a számításoknál, hogy ha egy érték hiányzó érték. Az egyes eljárások leírásai tartalmaznak egy-egy fejezetet a hiányzó értékekről. Ezeket kell megnéznünk, ha az adataink között előfordulhat hiányzó érték is.

P8.2 ÉRTÉKADÁS ÉS ÖSSZEHASONLÍTÁS

ÖSSZEHASONLÍTÁS HIÁNYZÓ ÉRTÉKEKKEL

A hiányzó értékek a legkisebb lehetséges értékek egy összehasonlításnál.

HIÁNYZÓ ÉRTÉK BEÁLLÍTÁSA

Teljesen helyénvaló egy SAS változónak hiányzó értéket adni értékül, és hasonlíthatunk is változóértéket hiányzó értékkel.

Pl.: `IF reggeli=.  THEN regtip='Nem kért' ;`

HIÁNYZÓ ÉRÉTKEKHEZ KAPCSOLÓDÓ OPCIÓK

Használhatjuk a MISSING opciót a numerikus hiányzó érték nyomtatási karakterének megváltoztatására:

Az `OPTIONS MISSING=X ;`

utasítás hatására a pont helyett X kerül nyomtatásra. Ez persze nem változtatja meg a hiányzó érték tárolási módját vagy a speciális hiányzó értékek nyomtatását.

Megjegyzés: Ha számot akarunk megjeleníteni, azt aposztrófok közé kell tenni.

```
OPTIONS MISSING='0' ;
```

P8.3 SZINTAKTIKAI ÉS ADATHIBÁK

HIBÁK AZ ADATOKBAN

Kétféle hiba fordul elő egy SAS programban, szintaktikai hiba és adathiba. Mindegyik kijavítható, ha tudjuk mit keressünk a SAS LOG-ban.

SZINTAKTIKAI HIBÁK

Ezek a hibák akkor fordulnak elő, ha hibásan írjuk le a kulcsszavakat, vagy kifelejtjük azokat, vagy az a híres-nevezetes pontosvessző lemarad.

A szintaktikai hibákat az 1. „dobozban” (ld. Korábbi ábra) találja meg a rendszer, a DATA lépés elején, így nagyon kevés gépi erőforrást fecsérel el egy ilyen hibás programra a rendszer. Amikor a SAS rendszer egy szintaktikai hibát talál, különféle műveleteket hajt végre:

- HIBAÜZENETEKET ÍR A SAS LOG-BA
- A HIBÁS SZÓT ALÁHÚZZA
- A HIBÁT LEÍRÓ ÜZENETET ÍR.

ADATHIBÁK

Ezek a hibák az adatokban vannak, olyanok, mint amikor egy nem numerikus karaktert akarunk egy numerikus változóba beolvasni, vagy nullával akarunk osztani. A SAS rendszer különböző dolgokat tesz adathiba esetén:

- EGY MEGJEGYZÉST ÍR A LOG-BA
- KIÍRJA AZ INPUT PUFFERT
- KIÍRJA A PDV-T
- KIÍRJA AZ AUTOMATIKUS VÁLTOZÓK ÉRTÉKÉT
- FOLYTATJA A PROGRAM VÉGREHAJTÁSÁT.

P8.4 PROBLÉMAMEGOLDÁSOK

ERROR

Az _ERROR_ változó értéke 1 lesz, ha adathiba fordul elő, egyébként 0. Ez a flag hasznos lehet akkor, ha adathiba esetén meg akarjuk állítani a DATA lépést. A következő DATA step egyetlen megfigyelést sem állít elő, ha pl. az

```
IF _ERROR_ = 1 THEN STOP ;
```

utasítást használjuk.

ÍRÁS A SAS LOG-BA

A **PUT utasítással** írhatunk szöveget vagy értékeket a SAS LOG-ba. Ez nagyon jól használható hibakeresésnél.

Pl.:

```
IF ujkeres=. THEN
PUT 'A kereset hiányzik – hibás sorsz.:' _N_ ;
```

ALL

Az **_ALL_ speciális változónév**, amely az összes változóra hivatkozik, beleértve az _N_ és az _ERROR_ változókat is.

A **_NUMERIC_ vagy a _CHARACTER_** is használható, ha csak a numerikus vagy csak a kerekteres változókra akarunk hivatkozni.

Pl.:

```
IF ujkeres=. THEN PUT _ALL_ ;
```

KÜLSŐ ADATOK LISTÁZÁSA

Az INPUT ; és a **LIST** ; utasításokkal beolvashatunk egy külső adatsort és kilistázhatjuk a SAS LOG-ba.

Pl.

```
DATA libref.sasdata ;
INPUT ;
LIST ;
DATALINES ;
.
.
RUN ;
```

Megjegyzés: A LIST utasítás csak a DATA lépés iterációjánál hajtódik végre, ezért ciklusban nem célszerű használni.

PÁRATLAN IDÉZŐJELEK

A páratlan idézőjelek furcsa hibaüzeneteket eredményeznek a LOG-ba!

> ***WARNING: The current word or quoted string has become more than 200 characters long. You may have unbalanced quotation marks.***

Egy pótlólagos idézőjelet is be kell írnunk, mivel a program ilyenkor még fut.
(A **`R – running`** – jel látható az editor ablak tetején)

A HIÁNYZÓ IDÉZŐJEL ELKÜLDÉSE

```
' ;
RUN ;
```

Erre az R jel eltűnik, azaz sikerült leállítani a futó programot.
A DATA lépés befejeződik, de hibásan. Újból el kell küldenünk, immár kijavítva.

PROC OPTIONS

Az aktuális opciókat megnézhetjük a LOG-ban a

```
PROC OPTIONS ;
RUN ;
```

program lefuttatásával,

vagy az OPTIONS ablak kinyitásával:

A LEGFONTOSABB RENDSZEROPCIÓK A KÖVETKEZŐK:

Options	Value	
CENTER	ON	- SAS output középre igazítva
DATE	ON	- kiírja-e a dátumot a címben
FIRSTOBS	1	- a feldolgozása az első megfigyeléstől kezdődik
FORMCHAR	¦ -- ¦ + ¦ --+=	- sor és oszlopelválasztó karakterek
LABEL		- használjon-e címkét az eljárásokban
LINESIZE	78	- a nyomtatott outputon az egy sorba írható kar.
MISSING	.	- a hiányzó érték helyett nyomtatandó karakter
NOTES	ON	- kiírja-e a megjegyzéseket a SAS LOG-ba
NUMBER	ON	- lapszámot írjon-e a SAS LOG-ba
OBS	MAX	- utolsóként feldolgozandó megfigyelés sorszáma
PAGENO	1	- egytől kezdje a lapszámozást az outputon
PAGSIZE	24	- az egy lapra írható sorok száma
XWAIT		- az X parancs nem vár

.
Stb.

Az OPTIONS ablak operációs rendszertől függően különböző lehet

HIBAKERESÉS A SAS PROGRAMBAN

A SAS programok ellenőrzésére szolgáló útmutatóval elkerülhetjük a leggyakoribb hibákat:

1. Szintaktikailag ellenőrizzük:
   - a hiányzó pontosvesszőket
   - kulcsszavak elírását
   - kezdő és záró idézőjeleket – párosaknak KELL lenniük
   - az utasítások kulcsszóval kezdődnek-e
   - a DO és a SELECT utasítást követi-e END utasítás
   - a páratlan megjegyzés határoló jeleket, a /* -ot lezárja-e */

2. Ellenőrizzük az utasítások sorrendjét, pl. az INPUT előtt van-e INFILE , stb.

3. Legalább az utolsó lépést zárjuk le RUN; utasítással, hogy az utolsó outputot is lássuk!

4. Ellenőrizzük az input adatokat, az-e amit várunk.
   (Használjuk a LIST; utasítást a külső adatok megnézésére.)

5. A PUT utasítással írjuk ki az értékeket a SAS LOG-ba, hasonlítsuk össze a várt értékekkel.

6. Egyszerűsítsük le a problémát – minden különleges sort vegyünk ki, amíg meg nem találjuk a probléma forrását.

7. Másképpen, kezdjük az alapoknál, egy kicsi DATA lépést teszteljünk csak a rosszul működő utasításokkal.

8. Próbáljuk ki a HELP-et, a hivatkozási kézikönyvet (ez az utolsó mentsvár!), és a SUGI vagy SEUGI kézikönyveket.

9. Nézzük meg a felhasználói megjegyzéseket (USAGE NOTES).

10. Forduljunk a SAS adminisztrátorhoz.

11. Kérjünk technikai támogatást a SAS Intézettől.

12. Próbáljuk kézzel végrehajtani a programutasításokat!

P8.5 GYAKORLATOK

1. FELADAT

A ***HIVATAL*** állományunkból számítsuk ki a dolgozók összjövedelmét (JOV néven) a FIZETES és PREMIUM változók összeadásával, A jelenlegi (FIZETES) és várható (UJFIZ) fizetés különbözeteként állapítsuk meg a várható emelés mértékét Ft-ban. Természetesen tartsuk meg az új változót is az állományunkban. Irassuk ki az AZON, FIZETES, UJFIZ, EMELES, PREMIUM, JOV változókat. A hiányzó értékek helyett nyomtassunk * karaktert.
Első pillantásra kiderül a listánkról, hogy jól dolgoztunk-e.

2. FELADAT

Keressük meg a következő program hibáit:

```
data=majus/1 ;
  input ev-ho esomm napsut homers felho ;
  if esomm < '60' then idojaras='Szép ;
  where hom not=. :
datelines
9705  78  1  15  8
9805  65  A 16  7
run;
9905  55  3  19 5
data majus.1 ; run ;
proc prink data majus.2 ;
eso nap felho comma9 ;
```

Próbáljunk meg egy futtatható programot eszkábálni belőle.

P9 FEJEZET

SAS ADATÁLLOMÁNYOK

P9.1 A SET UTASÍTÁS

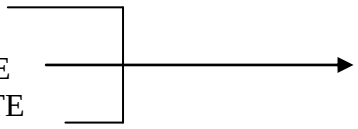
A SAS ADATÁLLOMÁNYOK SPECIÁLISAK

Tudjuk, hogy az összes eljárás SAS adatállományt használ inputként. A SAS adatállományok használhatók adatkezelésre szolgáló DATA lépés inputjaként is. Ez a SAS adatállományok speciális jellemzői miatt van; főleg azért, mert a SAS adatállományok önleírók. Valójában a SAS adatállományok két részből állnak: egy leíró (header) és egy adatrészből. A leíró rész tartalmazza az összes információt az adatállomány változóiról, beleértve a jellemzőiket is. Az adatrész a SAS adatállomány összes adatát tartalmazza.

A leíró részt a CONTENTS eljárással listázhatjuk ki. Az adatrész kiírására több eljárás is használható, a legegyszerűbb a PRINT eljárás.

A DATA lépés szempontjából az önleíró sajátosság nagyon fontos. A DATA lépésben 3-féle utasítás van, amelyekkel SAS adatállományokat olvashatunk be:

1. SET
2. MERGE
3. UPDATE



SAS ADATÁLLOMÁNYOK OLVASÁSA

SAS ADATÁLLOMÁNYOK OLVASÁSA

A SET utasítással olvashatunk például csak egyetlen SAS adatállományt:

ELSO SAS adatállomány:

A	B	C
1	5	3
2	6	4
3	7	5
4	8	6

Program:

```
DATA uj ;
 SET elso ;
RUN ;
PROC PRINT ;
RUN ;
```

UJ SAS adatállomány:

A	B	C
1	5	3
2	6	4
3	7	5
4	8	6

A DATA lépésben képezhetünk részhalmazokat, vagy használhatunk tetszőleges programutasításokat.

SAS ADATÁLLOMÁNYOK EGYESÍTÉSE

1. MÓDSZER: KONKATENÁLÁS

A SET utasításban legfeljebb 100 SAS adatállomány nevét lehet megadni. Ezzel az összeállítással egyesíthetünk több SAS adatállományt.

ELSO SAS adatállomány

A	B	C
1	5	3
2	6	4
3	7	5
4	8	6

MASODIK SAS adatállomány

A	B	C
4	8	6
3	7	5
2	6	4
1	5	3

A program:

```
DATA harmadik ;
SET elso masodik ;
RUN ;
PROC PRINT DATA=harmadik ;
RUN ;
```

HARMADIK SAS adatállomány

A	B	C
1	5	3
2	6	4
3	7	5
4	8	6
4	8	6
3	7	5
2	6	4
1	5	3

Itt az elsőnek megnevezett adatállomány megfigyeléseinek beolvasása után következnek a második adatállomány megfigyelései, tehát lényegében egymás után kerültek kiírásra, konkatenálás történt.

Megjegyzés: Nagy adatállományoknál ugyanezt a **PROC APPEND** eljárással hatékonyabban elvégezhetjük, a két adatállomány megfigyeléseinek beolvasása nélkül. Viszont ha más műveletet is végre akarunk hajtani az adatokon, akkor a DATA lépést használhatjuk.

KÜLÖNBÖZŐ VÁLTOZÓK

Ha az adatállományokban különböző változók vannak, akkor az hiányzó értékeket eredményez az output állományban:

ELSO SAS adatállomány

A	B	C
1	5	3
2	6	4
3	7	5
4	8	6

MASODIK SAS adatállomány

A	E	F
4	8	6
3	7	5
2	6	4
1	5	3

```
DATA harmadik ;
SET elso masodik ;
RUN ;
PROC PRINT DATA=harmadik ;
RUN ;
```

HARMADIK SAS adatállomány

A	B	C	E	F
1	5	3	.	.
2	6	4	.	.
3	7	5	.	.
4	8	6	.	.
4	.	.	8	6
3	.	.	7	5
2	.	.	6	4
1	.	.	5	3

A közös változók jellemzőit az elsőnek megnevezett adatállomány leíró részéből olvassa a SAS rendszer.

2. MÓDSZER: ÖSSZEFÉSÜLÉS

Ez a „rendezett összefésülés” :

```
PROC SORT DATA=elso ; BY a ;
RUN ;
```

```
PROC SORT DATA=masodik; BY a;
RUN;
```

ELSO SAS adatállomány

A	B	C
1	5	3
2	6	4
3	8	6

MASODIK SAS adatállomány

A	B	C
1	9	5
2	3	2
3	5	8

```
DATA harmadik ;
SET elso masodik ;
  BY a ;
RUN ;
PROC PRINT DATA=harmadik ;
RUN ;
```

HARMADIK SAS adatállomány

A	B	C
1	5	3
1	9	5
2	6	4
2	3	2
3	8	6
3	5	8

Ebben az esetben még egy utasítást használtunk – a **BY** utasítást - , aminek hatására automatikusan létrejönnek a ***FIRST.A*** (első A) és a ***LAST.A*** (utolsó A) változók.

P9.2 FIRST. ÉS LAST.

Az egy vagy több változóra rendezett SAS adatállományokon lehet a BY utasítással képzett csoportokra vonatkozó műveleteket végrehajtani. Ha a BY utasítást a SET utasítással együtt használjuk, a PDV-ben automatikusan létrejön két új verzió:

FIRST.BY-csoportváltozó (FIRST pont …-nak hívjuk)
LAST.BY-csoportváltozó (LAST pont …-nak hívjuk)

Ha a PDV-ben lévő aktuális megfigyelés első abban a bizonyos BY-csoportban, például a BY-csoportválozónak éppen megváltozott az értéke az előző megfigyeléshez képest, akkor a **FIRST.** változó értéke 1 lesz, különben 0. Ugyanígy a **LAST.** változó értéke akkor 1, ha az éppen a **PDV**-ben lévő megfigyelés a **BY**-változó aktuális értékéből az utolsót tartalmazza.

Ezeket használhatjuk pl. - duplikátum ellenőrzésre
- csoport elemek számolására

A **FIRST.** és **LAST.** változókat meg kell feleltetnünk más, DATA lépésbeli változóknak, mivel ezek automatikusan kimaradnak az output állományból.

EGY CSOPORT ELEMEINEK SZÁMOLÁSA

ÁLTALÁNOSAN:

1. Rendezzünk kulcs szerint.
2. DATA lépés, olvasás kulcs szerint.
3. Ha ez első kulcs, inicializáljunk.
4. Gyűjtsük fel az összesent.
5. Ha ez utolsó kulcs, akkor írjuk ki a megfigyelést.

Például: kíváncsiak vagyunk a gyumexp állományunkban hány fajtánk van gyümölcsönként.

```
PROC SORT DATA=tanf.gyumexp ;
BY gyumolcs ;
DATA gyumszam (KEEP=gyumolcs fszam);
  SET tanf.gyumexp ;
  BY gyumolcs ;
  IF FIRST.gyumolcs  THEN  fszam=0 ;
  fszam+1 ;
  IF LAST.gyumolcs ;
RUN;
PROC PRINT ;
RUN ;
```

A fenti program eredménye:

Obs	gyumolcs	fszam
1	alma	5
2	körte	3
3	szilva	3
4	sárgabarack	3
5	őszibarack	2

P9.3 ÖSSZEMÁSOLÁS PÁROSÍTÁSSAL

SAS adatállományok összemásolása (MERGE) hasonlóképpen történik, mint a SET utasításnál, de egészen más eredményt ad.

ELSO SAS adatállomány

A	B	C
1	5	3
2	6	4
3	7	5
4	8	6

MASODIK SAS adatállomány

D	E	F
4	8	6
3	7	5
2	6	4
1	5	3

`MERGE elso masodik ;` utasítás hatására:

HARMADIK SAS adatállomány

A	B	C	D	E	F
1	5	3	4	8	6
2	6	4	3	7	5
3	7	5	2	6	4
4	8	6	1	5	3

A MERGE a megfigyelések sorszáma szerint kapcsolja egymáshoz a megfigyelések tartalmát. „Összeragasztja” a SAS adatállományokat „oldalról”.

A MERGE egy másik VÉGREHAJTHATÓ OLVASÁS, amelyet csak SAS adatállományokra használhatunk. Feladata az adatállományok összekapcsolása. Ez kétféleképpen is használható, egyszerű MERGE utasítás BY utasítás nélkül, és párosított összemásolás BY utasítással.

AZ EGY AZ EGYHEZ MÁSOLÁS

PÉLDA:

ALLATOK állomány

ALLOMAS	ALLAT
01	nyúl
02	teknős
03	róka

TERULET állomány

ALLOMAS	TERULET
01	mező
02	mocsár
03	erdő

A program:

```
DATA vadon ;
  MERGE allatok terulet ;
PROC PRINT ;
RUN ;
```

Az összemásolás eredménye a (vadon):

ALLOMAS	ALLAT	TERULET
01	nyúl	mező
02	teknős	mocsár
03	róka	erdő

MEGJEGYZÉS AZ EGY AZ EGYHEZ MÁSOLÁSHOZ :

- Akkor alkalmazható, ha pontosan ugyanannyi megfigyelést tartalmaz mindkét adatállomány.
- Ez az egyszerű összemásolás (merge) a megfigyelés sorszám alapján párosít, azaz az első adatállomány első megfigyelése párosodik a második adatállomány első megfigyelésével.
- Az összemásolás nem csak két adatállománnyal végezhető el, ilyen módszerrel maximum 100 másolható össze.
- A mindkét állományban egyaránt meglévő változók csak egyszer szerepelnek az outputban, a második állománybeli értékkel.

Az adatok azonban soha nem ilyen egységesek és jól rendezettek! Legtöbbször már a megfigyelések száma is különböző.

PÉLDA:

ALLOMAS	ALLAT
01	nyúl
03	róka
04	menyét
02	teknős

ALLOMAS	TERULET
01	mező
02	mocsár
03	erdő
04	rét

Ha a program megegyezik az előbbivel, akkor az eredmény :

ALLOMAS	ALLAT	TERULET
01	nyúl	mező
02	róka	mocsár
03	menyét	erdő
04	teknős	rét

Az eredményünk rossz lett, mert a hiányzó érték miatt a további megfigyelések rossz területekkel párosodtak. Hogy ezt kikerüljük, ilyen esetekben a párosított összemásolást kell alkalmazni.

PÁROSÍTOTT ÖSSZEMÁSOLÁS

A párosított összemásolás a **BY** utasítással meghatározott csoportosítást használja.

Az előbbi példa helyesen:

```
DATA vadon ;
  MERGE allatok terulet ;
  BY allomas ;
RUN ;
```

Most az eredmény:

ALLOMAS	ALLAT	TERULET
01	nyúl	mező
02	teknős	mocsár
03	róka	erdő
04	menyét	rét

MEGJEGYZÉSEK A PÁROSÍTOTT ÖSSZEMÁSOLÁSHOZ

- A közös változó értéke a második adatállományból jön.

- A hiányzó érték ellenére sem következik be elcsúszás, mert a párosítás a BY-változó értéke szerint történik.

A BY-változó szerinti csoportosításhoz az adatállományoknak a BY-változó szerint kell elsődlegesen rendezettnek lennie, tehát ha közös változót tartalmazó állományokat kell összemásolnunk, használjuk előbb a PROC SORT-ot, vagy meg kell indexelni az állományt.

A közös változónak egyforma nevűnek és típusúnak kell lennie. Ha a hosszuk különböző, akkor az első adatállományban lévő hossz lesz az érvényes.

ISMÉTLŐDŐ ÉRTÉKEK

Az előbbi módszer akkor is beválik, ha az egyik, vagy akár mindkét adatállomány ismétlődő értékeket tartalmaz.

P9.4 A PÁROSÍTOTT ÖSSZEMÁSOLÁS MŰKÖDÉSE

PÉLDA:

ALLATOK

ALLOMAS	TÍPUS
01	**NYÚL**
01	borz
04	menyét
05	hal
06	őz
06	borz
06	róka

TERULET

ALLOMAS	TERULET
01	**MEZŐ**
02	mocsár
03	puszta
04	rét
05	tó
06	erdő

A program:

```
DATA vadon ;
  MERGE allatok terulet ;
  BY allomas ;
RUN ;
```

Az output lista a következő lesz:

ALLOMAS	ALLAT	TERULET
01	nyúl	mező
01	borz	mező
02		mocsár
03		puszta
04	menyét	rét
05	hal	tó
06	őz	erdő
06	borz	erdő
06	róka	erdő

A PÁROSÍTOTT ÖSSZEMÁSOLÁS LÉPÉSEI

1. A SAS rendszer elolvassa az első adatállomány leíró részét és felveszi a PDV-be a változóit:

   ALLOMAS TIPUS

2. A SAS rendszer elolvassa a második adatállomány leíró részét is és felveszi a PDV-be az előzővel nem közös változóit.

   ALLOMAS TIPUS **TERULET**

Tehát a fenti példában az ALLOMAS és a TIPUS az ALLATOK nevű adatállományból, a TERULET nevű változó a TERULET nevű adatállományból került a PDV-be. Az állomás numerikus, a típus és a terület karakteres.

3. Mindegyik adatállomány első megfigyelése külön input pufferben tárolódik és a rendszer a közös változó értékeit összehasonlítja.

4. Ha egyenlők, mint itt, akkor először az első adatállomány értékeit olvassa be a rendszer a PDV-be, aztán a másodikét. A közös változó értéke az utolsó adatállományból származó értékkel íródik felül. A megfigyelés a PDV-ből íródik ki az output SAS adatállományba.
5. Ezután a SAS rendszer beolvassa a következő megfigyeléseket az input pufferekbe, és összehasonlítja a közös csoportváltozók (BY-változók) értékét az előző értékükkel.

6. A SAS berakja a meglévővel egyező értékű közös változót tartalmazó megfigyelést a PDV-be és kiírja a megfigyelést az output SAS adatállományba.

7. Most csak egy SAS adatállományt kell beolvasni és input pufferbe. Az ALLATOK nevű állomány újabb megfigyelését beolvassa a SAS rendszer. A közös változók értékeit most 04 és 02, tehát mindegyik különbözik a PDV-ben meglévő értéktől. Amikor ez történik, akkor a PDV törlődik és a ciklus megismétlődik.

8. Ha a közös változók input pufferekben lévő értékei egymástól is és a PDV-ben lévő értéktől is különböznek, akkor az input pufferekben lévő értékeket összehasonlítja a SAS rendszer, és a kisebbet viszi át a PDV-be és írja ki az output adatállományba.

AZ *IN = VÁLTOZÓ* ADATÁLLOMÁNY OPCIÓ

Az **IN=változó** opció hatására létrejön egy új változó, amelyben a SAS azt jelzi, hogy egy SET, MERGE, vagy UPDATE utasítás során feldolgozásra kerülő megfigyelés ahhoz az állományhoz tartozik-e, amelyhez az opciót megadtuk. Értéke 1,ha a SAS a megfigyelést állományunkból vette, egyébként 0 .
Az így megadott változó a DATA lépés során elérhető, de a létrehozandó állománynak nem lesz része.
Ezt a lehetőséget igen jól alkalmazhatjuk párosított összemásolás esetén az outputra kerülő megfigyelések vezérlésére.

Pl. ha az alábbiak szerint határozzuk meg a párosított összemásolást:

```
DATA cfile ;
  MERGE     afile(IN=a)
            bfile(IN=b) ;
  BY  kulcs ;
```

akkor a következők a lehetőségeink:

1. **IF a ;** → Csak azok a megfigyelések kerüljenek outputra, amelyek kulcsa az ***afile***-ban mindenképpen szerepel

2. **IF a and b ;** → Csak azok a megfigyelések kerüljenek outputra, amelyek kulcsa **mindkét fájlban** szerepel.

3. **IF a and not b ;** Csak azok a megfigyelések kerüljenek outputra, amelyek kulcsa **csak az *afile*-ban** szerepel

4. Ha nem rendelkezünk sehogy, az **összes kulcs**hoz tartozó megfigyelések változói outputra kerülnek, függetlenül attól, hogy melyik állományban szerepelnek.

P9.5 GYAKORLATOK

1. FELADAT

A ***TANF\SASOK*** nevű könyvtár ***SZEMELY*** nevű állományából készítsünk egy FERFI és egy NO ideiglenes állományt egy lépésben úgy, hogy csak a szükséges változók kerüljenek beolvasásra. A ***FERFI*** állomány tartalmazzon egy FELESEG nevű változót is, amelybe, ha házasságban él, 1 kell, hogy kerüljön. A ***NO*** állomány pedig egy FERJ mezőt, amelynek értéke 1, ha házasságban él. Másoljuk egymás mögé ezt a két állományt és írassuk ki. Mit tapasztalunk?

2. FELADAT

Hozzuk létre a ***SZEMELY*** állományunkból a ***SZEM2*** ideiglenes állományt a NEV, KOR változók megtartásával. A ***HIVATAL*** állomány NEV, OSZTALY változóiból pedig a ***HIV2*** ideiglenes állományt. Írjuk ki mindkettőt az output ablakba, majd egyesítsük a két állományt egy az egyhez másolással. Az eredményt az előbbi két listához hasonlítva mit tapasztalunk?

3. FELADAT

Hajtsuk végre az előző feladatot úgy, hogy a ***SZEM2*** és a ***HIV2*** állományokat most már a NEV mezőn párosítva egyesítsük.
A párosítást 4 féleképpen oldjuk meg.
1./ Csak a mindkét állományban megtalálható személyek kerüljenek outputra.
2./ Csak a HIV2 állományban lévő személyek kerüljenek outputra.
3./ Csak a HIV2 állományban nem lévők kerüljenek outputra;
4./ Mindenki kerüljön outputra.
Segítség: az állományoknak adjunk IN=SZ és IN=HIV data lépés opciót. Ezeket használjuk fel a kiválasztó feltételünkben.
Mindegyik esetben nyomtassuk ki és tanulmányozzuk az eredményt.

P10 FEJEZET
FORMÁTUMOK

10.1 INPUT ÉS OUTPUT FORMÁTUMOK

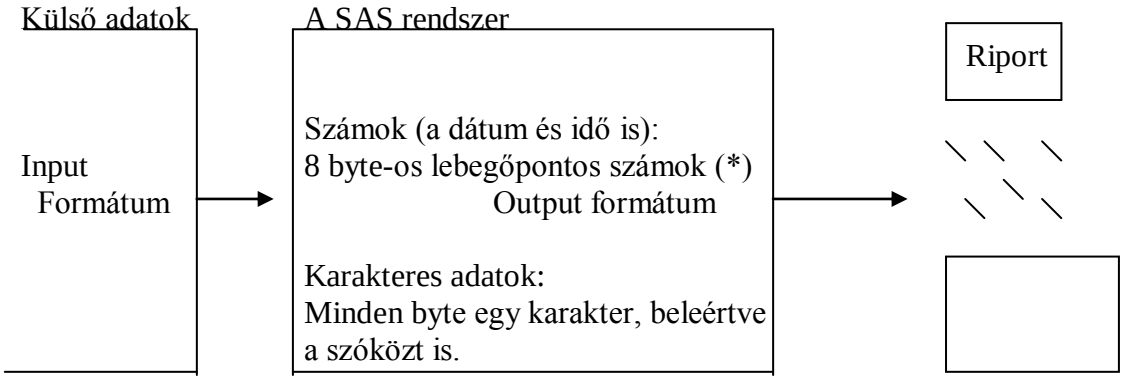
10.2 PROC FORMAT

10.3 DÁTUM ÉS IDŐ

10.4 KÜLSŐ FILE LÉTREHOZÁSA SAS ÁLLOMÁNYBÓL

10.5 GYAKORLATOK

P10.1 INPUT ÉS OUTPUT FORMÁTUMOK



(*) Különböző gépeken eltérő a megvalósításuk

AZ INPUT FORMÁTUMOK A KÜLSŐ ADATOK ELRENDEZÉSÉT ÍRJÁK LE.

AZ OUTPUT FORMÁTUMOK AZ ADATOK SAS-BELI ÁBRÁZOLÁSA HELYETT AZ ADOTT ELRENDEZÉS SZERINT JELENÍTIK MEG AZ ADATOKAT.

AZ INPUT FORMÁTUMOK LISTÁJA

Formátum	Leírás	Mérethatár	Alapért.
NUMERIKUS			
w.	Standart numerikus	1-32	
w.d	Standart numerikus	1-32	
BINARYw.d	Pozitív bináris egész	1-64	8
BITSw.d	Bitek kifejtése	1-64	1
BZw.d	A nullák szóközök	1-32	1
COMMAw.d	Vesszők a számban	1-32	1
COMMAXw.d	. és , a számban felcserélve	1-32	1
Ew.d	Exponenciális ablak	7-32	12
HEXw.	Num. hexadecimális	1-16	8
IBw.d	Bináris egész	1-8	4
OCTALw.d	Oktális egész	1-24	3
MRBw.d	Microsoft bináris lebegőpontos	2-8	4
PDw.d	Pakolt decimális	1-16	1
PERCENTw.	Százalék numerikus értékhez	1-32	6
PKw.d	Előjel nélküli pakolt	1-16	1
PIBw.d	Pozitív bináris egész	1-8	1
RBw.d	Bináris lebegőpontos	2-8	4
DATEw.	NNHHHÉÉ formájú dátum	7-32	7
DATETIMEw.d	Dátum, idő	13-40	18
DDMMYYw.	NNHHÉÉ formájú dátum	6-32	8
JULIANw.	Julianus-féle dátum (ÉÉNNN vagy ÉÉÉÉNNN)	5-32	5
MMDDYYw.	HHNNÉÉ formájú dátum	6-32	6
MONYYw.	Hónap, év	5-32	5
NENGOw.	Japán dátum (R.ÉÉHHNN)	7-32	10
PDTIMEw.	Pakolt dec. SMF/RMF rekord	4	4
RMFDURw.	RMF intervallumok időtartama	4	4
RMFSTAMPw.	RMF rekordok dátum és idő mezői	8	8
SMFSTAMPw.	SMF rekordok dátum és idő mezői	8	8
TODSTMPw.	Idő óó:pp:mm.mm formában	5-32	8
TUw..	A timer egységeket olvassa	4	4
TIMEw.d	Idő	5-32	8
YYMMDDw.	ÉÉHHNN formájú dátum	6-32	8
YYQw.	Év és negyedév	4-32	4
S370FIBw.d	IBM 370 bináris egész	1-8	4
S370FPDw.d	IBM 370 pakolt dec.	1-16	1
S370FPIBw.d	IBM 370 pozitív bináris egész	1-8	4
S370FRBw.d	IBM 370 valós bináris lebegőpontos	2-8	6
ZDw.d	Zónázott decimális	1-32	1
ZDBw.	Zónázott dec. szóközökkel	1-32	1

KARAKTERES			
$w.	Standard karakteres	1-200	1
$ASCIIw.	ASCII kar.-ből az aktuálisra	1-200	1
$BINARYw.	Binárisból karakteres	1-200	8
$CHARw.	Karakteres, vezető szóközzel	1-200	1
$CHARZBw.	Karakteres, szóköz helyett bináris nulla	1-200	1
$HEXw.	Karakteres hexadec.	1-16	2
$VARYINGw.	Változó hosszú kar.	1-200	8
$OCTALw.	Karakteres oktális	1-200	3
$PHEXw.	Karakteres pakolt hexadec.	1-200	2
$EBCDICw.	EBCDIC karaktereket ASCII-ra konvertál	1-200	1

AZ OUTPUT FORMÁTUMOK LISTÁJA

Output form.	Leírás	Méret-határok	Alapért.	Igazítás
w.	Standard numerikus	1-32		Jobbra
w.d	Standard numerikus	1-32		Jobbra
BESTw.	SAS szerint legjobb	1-32	12	Jobbra
BINARYw.	bináris	1-64	8	Balra
COMMAw.d	Vesszők a számban	2-32	6	Jobbra
DOLLARw.d	Dollárjel, vesszők	2-32	6	Jobbra
DOLLARXw.d	Dollárjel, '**.**' és '**,**' felcserélve	2-32	6	Jobbra
Ew.	Exponenciális alak	7-32	12	Jobbra
FRACTw.	Törtrész	4-32	10	Jobbra
HEXw.	Numerikus hexadecimális	1-16	8	Balra
IBw.d	Bináris egész	1-8	4	Balra
MRBw.d	Microsoft bináris valós (lebegőpontos)	2-8	4	Balra
NEGPARENw.d	A negatív zárójelben	1-32	6	Jobbra
OCTALw.	Numerikus oktális	1-24	3	Balra
PDw.d	Pakolt decimális	1-16	1	Balra
PERCENTw.d	A számok százalékok	3-32	6	Jobbra
PKw.d	Előjel nélküli pakolt	1-16	1	Balra
PIBw.dw.	Pozitív Bináris egész	1-8	1	Balra
RBw.d	Bináris lebegőpontos	2-8	4	Balra
ROMANw.	Római számok	2-32	6	Balra
Zw.d	Vezető nullák	1-32	1	Jobbra
WORDFw.		5-200	10	Balra
WORDSw.	Számról angol szóra	5-200	10	Balra
ZDw.d	Zónázott decimális	1-32	1	Balra
$w.	Standard karakteres	1-200	1 v. hossz	Balra
$ASCIIw.	Aktuálisról ASCII-ra	1-200	1	Balra
$BINARYw.	Karakteres bináris	1-200	8	Balra
$EBCDICw.	Aktuálisról EBCDID-re	1-200	1	Balra
$CHARw.	Karakteres, vezető szóközzel	1-200	1 v. hossz	Balra
$HEXw.	Karakteres hexadec.	1-200	4	Balra
$VARYINGw.	Változó hosszú karakteres	1-200	8 v. hossz	Balra
$OCTALw.	Karakteres oktális	1-200	1	Balra
DATEw.	NNHHHÉÉ formájú Dátum	5-9	7	Jobbra
DATETIMEw.d	Dátum, idő	7-40	16	Jobbra
DAYw.	A hónap napja	2-32	2	Jobbra
DDMMYYw..	NNHHÉÉ formájú Dátum	2-8	8	Jobbra
DOWNAMEw.	A hét napjának neve	1-32	9	Jobbra
HHMMw.d	Óra, perc	2-20	5	Jobbra
HOURw.d	Óra	2-20	2	Jobbra
JULDAYw.	Az év napja (Julianus)	3-32	3	Jobbra
JULIANw.	Julianus-dátum	5-7	5	Balra
MMDDYYw.	HHNNÉÉ formájú Dátum	2-8	8	Jobbra
MMSSw.d	Perc és másodperc	2-20	5	Jobbra
MMYYxw.	Hó és év elhatárolva	5-32	7	Jobbra

MONNAMEw.	A hónap neve	1-32	9	Jobbra
MONTHw-	Az év hónapja	2-32	2	Jobbra
MONYYw.	Hónap, év	5-7	5	Jobbra
NENGOw.	Japán dátum	2-10	10	Balra
QTRw.	negyedév	1-32	1	Jobbra
QTRRw.	n. év római számmal	1-32	3	Jobbra
TIMEw.d	Idő	2-20	8	Jobbra
TODw.	Napi idő	2-20	8	Jobbra
WEEKDATEw.	Dátum	3-37	29	Jobbra
WEEKDATXw.	Dátum	3-37	29	Jobbra
WEEKDAYw.	A hét napja	1-32	1	Jobbra
WORDDATEw.	Dátum	3-32	18	Jobbra
WORDDATXw.	Dátum	3-32	18	Jobbra
YEARw.	Az érték év része	2-32	2	Jobbra
YYMMxw.	Év és hó elhatárolva	5-32	7	Jobbra
YYMMDDw.	ÉÉHHNN formájú Dátum	2-8	8	Jobbra
YYMONw.	Év és hónap rövidítve	5-32	7	Jobbra
YYQRxw.	És + negyedév római számmal, elhat.	6-32	8	Jobbra
YYQw.	Év és negyedév	4-6	4	Jobbra
S370FIBw.d	IBM 370 bin. Egész	1-8	4	Balra
S370FPDw.d	IBM 370 pakolt decimális	1-16	1	Balra
S370FPIBw.d	IBM 370 pozitív Bin. Egész	1-8	4	Balra
S370FRBw.d	IBM 370 valós bináris	2-8	6	Balra
$EBCDICw.	ASCII ---> EBCDIC	1-200	1	Balra

AZ OUTPUT FORMÁTUMOK HASZNÁLTA

A formátumok adják az outputok MEGJELENÍTÉSÉBEN a különbséget, de nincs hatásuk az adatok belső tárolására. Tekintsük a következő példát. Az adatok egyszerű karaktersorozatok, egy szám és egy dátum. A PROC PRINT eljárásban ezeket használjuk először formátum nélkül, aztán két különböző output formátummal:

FORMÁTUM NÉLKÜL:

```
DATA … ;
  INPUT szoveg $ szam @20 datum DATE7. ;
DATALINES ;
ezilyen 1234567890 01mar90
PROC PRINT ;
RUN ;
```

Az eredmény:

```
OBS         SZOVEG            SZAM        DATUM

1           ezilyen           1234567890  11017
```

OUTPUT FORMÁTUMMAL:

```
PROC PRINT ;
  FORMAT szoveg $HEX.
         szam   COMMA14.
         datum  WORDDATE. ;
RUN ;
```

Az eredmény:

```
OBS   SZOVEG                SZAM                    DATUM
1     657A696C79656E20      1,234,567,890           March 1, 1990
```

P10.2 PROC FORMAT

Az eddig látott összes formátum a SAS termék által szolgáltatott standard formátum volt. Ezzel szemben mi is előállíthatunk saját, felhasználó által definiált formátumot a PROC FORMAT segítségével.

Az elv a következő:
- definiáljuk a saját formátumunkat
- normál módon használjuk a FORMAT utasítás segítségével

Kétféle formátumot definiálhatunk:

VALUE FORMÁTUM

A value (érték) formátum egyaránt használható karakteres és numerikus adatokhoz. Ennek segítségével értékeket vagy érték-intervallumokat alakíthatunk át karaktersorozattá.

PICTURE FORMÁTUM

A **picture** (kép) formátum csak numerikus adatokhoz jó. A szám megjelenítéséhez szolgáltat mintát. Opcióival az utasításokban szereplő nyomtatási képek formáját módosíthatjuk. Megadásuk zárójelben történik. Pl.:

PREFIX=*'karakter(ek)'* --> nyomtatási kép elejéhez írandó karakterek megadása
MULT=*n* --> szorzó szám, amellyel a nyomtatandó számot szorozzuk kiírás előtt (pl. mértéegység átváltások esetén)
FILL='*karakter*' → az értékes számjegyek előtti szóközök kitöltése ezzel történik

Pl.:

```
PROC FORMAT ;
  VALUE $nem          'F'='Férfi'
                      'N'='Nő' ;
  VALUE  mag             LOW-<160='Alacsony'
                    161-<180='Középtermetű'
                    181-<HIGH='Magas' ;
  PICTURE  jov          0-HIGH='00,000,009,99' (PREFIX='Ft');
RUN ;
PROC PRINT DATA = tanf.demograf  NOOBS;
VAR nev nem magas joved ;
  FORMAT nem      $nem.
            magas       mag.
            joved       jov.  ;
RUN ;
```

A z OUTPUT listánk a következő lesz:

nev		nem	magas	joved
Petrik	Péter	Férfi	Magas	Ft46,500.00
Tóth	Judit	Nő	Középtermetű	Ft43,000.00
Lengyel	Géza	Férfi	Alacsony	Ft55,000.00
Földes	Tamás	Férfi	Középtermetű	Ft123,000.00
Tornyos	Kinga	Nő	Alacsony	Ft76,500.00
Rózsa	Sándor	Férfi	Középtermetű	Ft75,000.00
Károly	Éva	Nő	Középtermetű	Ft80,000.00
Tóth	Gábor	Férfi	Magas	.
Szabó	Edit	Nő	Középtermetű	Ft130,000.00
Nagy	Judit	Nő	Középtermetű	Ft111,000.00

MEGJEGYZÉSEK A FORMÁTUMOKHOZ:

- Mint a standard input és output formátumoknál, itt is pontot írunk a formátumnév végére használatkor, viszont a definiálásnál nem.

- A formátumneveknek meg kell felelniük a SAS névkonvencióknak. A karakteres értékek formátumainak dollárjellel kell kezdődniük.

- A LOW, HIGH és az OTHER kulcsszavak jelentése a ’legkisebb érték’, a ’legnagyobb érték’ és az ’egyéb’.

- A jobboldali megnevezésnek mindig idézőjelben kell lennie és a korábbi verziókban legfeljebb 40 karakter hosszú lehet. A 8.0-s verzióban ez a határ már 256 karakter.

- Az intervallumok nem fedhetik át egymást:

```
0 - < 5 = 'megnevezés_1'
5 - < 9 = 'megnevezés_2'
```

azt jelenti, hogy pl. a megnevezés2-t az 5 értékhez hozzárendeli a rendszer, viszont a 9-hez már nem.

- Az értékek kombinálhatók kulcsszavakkal is:

```
VALUE elojel
      LOW - < 0   = 'negatív'
      0           = 'nulla'
      1 < - HIGH  = 'pozitív'
```

ÁLLANDÓ FORMÁTUMKÖNYVTÁRAK

Az összes eddig létrehozott formátum ideiglenes volt, a WORK.FORMATS nevű katalógusban tárolva. Előállíthatunk olyan formátumokat a programjaink számára, amelyek bármely SAS futás alatt elérhetők, ha állandó katalógusban tároljuk azokat a LIBRARY könyvtárnév alatt.
(LIBNAME LIBRARY fájlnév)
Használjuk a LIBRARY= opciót a formátum előállításakor.

A későbbi használatkor elég a **LIBNAME LIBRARY *'könyvtár_hivatkozás'*** ; utasítás, hogy a SAS rendszer megtalálja a formátumokat.

```
LIBNAME LIBRARY 'SAS_adat_könyvtár' ;
PROC FORMAT LIBRARY=LIBRARY ;
      VALUE kodok ..
            ……..
            ……. ;
      VALUE karakt ..
      ……..
                  ……. ;
      PICTURES szam ..
            ……
            ……;
RUN ;
```

Ezeket a megszokott módon használhatjuk. A SAS rendszer először a WORK.FORMATS katalógusban keres, aztán a LIBRARY.FORMATS nevűben. Ne felejtsük el a LIBNAME utasítást előtte kiadni!

Az FMTLIB opció dokumentálja a formátumkönyvtár tartalmát.

P10.3 DÁTUM ÉS IDŐ

DEFINÍCIÓ

Az emberek olyan rendszer szerint használják a dátumot és az időt, amely bármely racionális gondolkodású számítógép számára teljesen illogikusnak látszik.

A most használatos naptárunk inkább történelmi és konvencionális, semmint logikus. Már a görög Erathosztenész felfedezte, hogy az év 365,25 napból áll, ezzel tökéletesítette az akkori naptárunkat, amelyben mind a 12 hónap 30 napos volt!

A mi dátumírási konvenciónkkal elég nagy munka lenne megválaszolni azt a kérdést, hogy ’hány hét van még Karácsonyig?’ vagy ’a hét melyik napján születtem?’.

Hogy ezzel a feladattal ésszerűen tudjunk megbirkózni, a SAS elsajátította azt a stratégiát, hogy a dátumot és időt egyszerű számoknak tekintse.

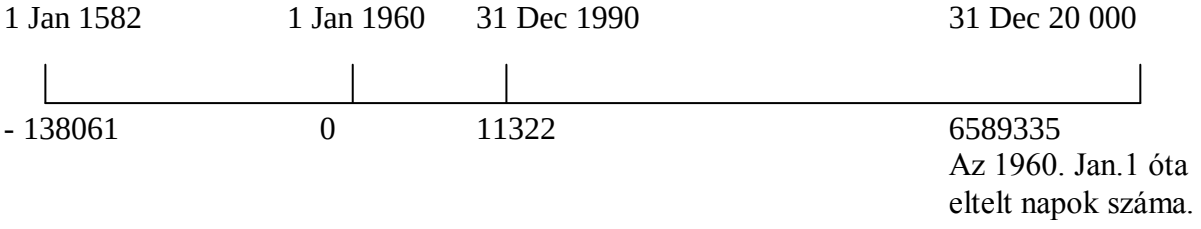
SAS DÁTUMOK

Egy SAS adatállományban:
- a dátum (DATE) az 1960 január 1 óta eltelt napok száma,
- az idő (TIME) az éjfél óta eltelt másodperceket tartalmazó szám.

A SAS-ban lehetőség van a kettő kombinációjának használatára (DATATIME). Ez az 1960. Január 1. éjfél óta eltelt másodpercek száma.

ÉRVÉNYES DÁTUMOK:

A SAS dátumok 1582-től 20 000-ig érvényesek, ebben az intervallumban a szökőévek, évszázadok és a negyedik évszázadok kezelése megfelelő.



Tehát a SAS dátumok és idők egyszerű egész számokként tárolódnak egy közös hivatkozási ponthoz viszonyítva. Mi azonban nem ismerjük fel a dátumot 8220 vagy –456 alakban, számunkra a szokásos konvenciók használatával lesz értelme.

INPUT ÉS OUTPUT FORMÁTUMOK

Hogy kiküszöböljük a konvenciók okozta problémát, a dátumot általában speciális dátum/idő input formátumok használatával olvassuk be a SAS-ba.

Hogy ezeket a számokat érthető formában láthassuk újra, egy megfelelő formátummal kell megjelenítenünk őket.

A leggyakoribb input és output formátumok táblázatát találjuk az alábbiakban. Egyéb input formátumok is léteznek, elsősorban a rendszer fájlok és óra-adatok olvasásához. További részleteket a User's Guide c. könyvben találhatunk.

SAS DÁTUM ÉS IDŐ INPUT FORMÁTUMAI

Input formátum	Leírás	Méret határok	Alapértelmezés
DATEw.	NNHHHÉÉ formájú dátum	7-32	7
DATETIMEw.d	Dátum, idő	13-40	18
DDMMYYw.	NNHHÉÉ formájú dátum	6-32	8
MMDDYYw.	HHNNÉÉ formájú dátum	6-32	6
MONYYw.	Hónap, év	5-32	5
TIMEw.d	Idő	5-32	8
YYMMDDw.	ÉÉHHNN formájú dátum	6-32	8
YYQw.	Év és negyedév	4-32	4

A SAS DÁTUM ÉS IDŐ OUTPUT FORMÁTUMAI

Output formátum	Leírás	Méret határok	Alapértelmezés
DATEw.	NNHHHÉÉ form. Dátum	5-9	7
DATETIMEw.d	Dátum, idő	7-40	16
DDMMYYw..	NNHHÉÉ form. Dátum	2-8	8
HHMMw.d	Óra, perc	2-20	5
HOURw.d	Óra	2-20	2
MMDDYYw.	HHNNÉÉ form. Dátum	2-8	8
MMSSw.d	Perc és másodperc	2-20	5
MMYYxw.	Hó és év elhatárolva	5-32	7
TIMEw.d	Idő	2-20	8
TODw.	Napi idő	2-20	8
WEEKDATEw.	Dátum	3-37	29
WEEKDATXw.	Dátum	3-37	29
WORDDATEw.	Dátum	3-32	18
WORDDATXw.	Dátum	3-32	18
YYMMDDw.	ÉÉHHNN form. Dátum	2-8	8
YYQw.	Év és negyedév	4-6	4

LITERÁLOK

Néha szükségünk lehet speciális rögzített dátumokra vagy időpontokra a programjainkban. Idézzük az értéket az alább látható formátumok szerint és írjunk utána D (Date=dátum) vagy T (Time=idő) vagy DT (Datetime= dátum és idő) betűket értelemszerűen:

'01MAR90'D
'1MAR1990'D
'12:01'T
'12:01:40'T
'01MAR90:12:01:40'DT

Ilyen literálok az összehasonlításnál is használhatók:

```
DATA uj ;
 IF TODAY () > '31dec89'D  THEN…..;
```

A DÁTUM ÉS AZ IDŐ FÜGGVÉNYEI

A SAS rendszerben található több, mint 20 dátum- és időfüggvény legtöbbje a dátumot és az időt argumentumának tekinti és egy bizonyos részét kiveszi.

A **DATE ()** és a **TODAY ()** a rendszer órájából adja vissza az aktuális dátumot. A TIME () ugyanezt teszt a rendszeridővel. A **DATETIME ()** az aktuális dátumot és időt adja vissza. Mindhárom függvény argumentum nélküli.

INTERVALLUMOK

A különbség-függvények különösen hasznosak. Kettő ilyen van:

Az INTCK (intervallum,kezdő,záró) a kezdő és a záró dátum vagy idő közötti időintervallumok számát adja meg.

Az intervallumokat a rendszer egy fix időponttól számolja, a részintervallumok nem számítanak.

Az INTNX (intervallum,kezdet,szám) a kezdeti időponttól a megadott számú intervallum elteltének megfelelő időt (dátumot) adja.

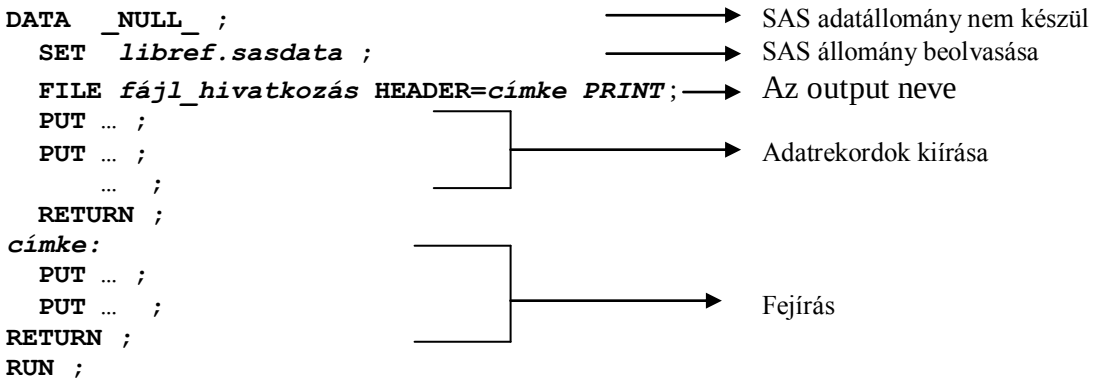
JULIANUS NAPTÁR SZERINTI DÁTUMOK

Akik a Julianus naptárt akarják használni, azoknak a JULDATE (SAS-dátum) függvény egy adott SAS –dátumból Julianus szerinti dátumot ad vissza. A DATEJUL (Julianus-dátum) függvény ennek a fordítottja, egy adott Julianus-dátumból SAS dátumot állít elő.

P10.4 KÜLSŐ FILE ELŐÁLLÍTÁSA SAS ADATÁLLOMÁNYBÓL

Sok esetben, pl. ha egyetlen, meghatározott formájú, gyakran használt riportra van szükségünk, akkor érdemes időt fordítani rá, hogy egy bonyolultabb, speciálisabb programot írjunk annak elkészítésére.

Ehhez pl. a következő DATA lépésbeli szerkezet szükséges:



A **DATA _NULL_ ;** utasítás eredményeképpen nem készül SAS állomány.
A **HEADER=** opcióban megadott címke egy érvényes SAS név, definiálásakor kettőspont áll mögötte. Itt definiáljuk a riport fejrészét. A **PRINT** opció hatására a PUT utasítással meghatározott rekordok a standard SAS nyomtató outputra íródnak. De lehet a PRINT fájlhivatkozás is.

A **RETURN** utasítások hatása:

- DATA lépésen belüli: → a lépés kezdetétől folytatódik a végrehajtás
- A HEADER-ben megadott: → a végrehajtás az után az utasítás után folytatódik, amelynél az új lap feltétel fellépett.

A DATA lépésben külső adatállományt, vagy nyomtatható szövegfájlt éppoly rugalmasan és hatékonyan írhatunk, mint ahogy olvasni már megtanultunk. Ezt a **FILE** és **PUT** utasításokkal tehetjük, amelyek a korábban már megismert INFILE és INPUT utasítások megfelelői. A legtöbb opciójuk megegyezik egymással. Úgyszintén háromféle kiírás lehetséges:

- oszlop szerinti
- listás
- formátumos

PUT UTASÍTÁS

A PUT UTASÍTÁS ÁLTALÁNOS FORMÁJA:

```
PUT  [ specifikáció ]  ;
```

Ahol a **specifikáció** a következők valamelyike:

változó	- a kiírandó változó neve
'karaktersorozat'	- a kiírandó szöveg
'pointervezérlés'	- a pointer a megfelelő oszlopra áll A / jel a soremelést vezérli.
INFILE	- az inputról utoljára beolvasott rekordot írja ki. (Külső fájlból külső fájlba írás)
ALL	- valamennyi változót kiírja a változó nevét is tartalmazó kiírási formában (pl. nev=Kati ev=25)

Kiíráskor a numerikus értékek jobbra, a karakteres változók értékei balra igazítva jelennek meg.
Vektorok kiírása történhet elemenként és teljes egészében is. Pl. **PUT tab(*) ;**

FILE UTASÍTÁS

Mindig a PUT előtt álló utolsó FILE utasítás határozza meg az aktuális output fájlt, ahová a SAS írni fog.

Szintaktikája: **FILE** ***fájl_specifikáció*** **[** ***opciók*** **]** ***;***

A fájlspecifikáció formái:

fájlhivatkozás	- a fájl rövid neve (kell FILENAME)
'külső fájl'	- az output fizikai neve (ilyenkor nem kell FILENAME)
LOG	- a SAS LOG fájl
PRINT	- a standard SAS print fájl (nyomtató esetén pl LPT1)

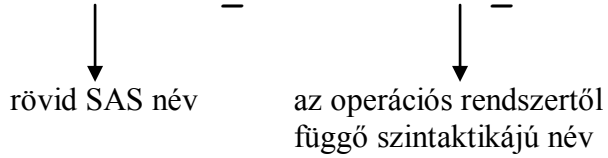
Több opció adható meg. A legfontosabbak:

Pl.	HEADER=címke	- fejléc kiíratásához (új lap esetén hajtja végre)
	N=érték	- hány sor kerüljön egy lapra
	LRECL=	- logikai rekordhossz (hosszú rekordok esetén)

FILENAME UTASÍTÁS

Általában a FILE utasításban nem a külső állomány fizikai nevét, hanem a rövidebb, hivatkozási nevét használjuk. Ennek megadása a FILENAME utasítással történik.

```
FILENAME  hivatkozási_név  'fizikai_név'  opciók ;
```



PÉLDÁK:

- SAS állomány tartalmának nyomtatóra írása (operációs rendszer függő)

```
FILENAME lista 'LPT1' ;
LIBNAME  tanf 'c:\tanf\sasok' ;
DATA  _NULL_ ;
 SET  tanf.gyumexp ;
 BY gyumolcs ;
 FILE  lista  HEADER=fejlec n=64 PRINT ;
 IF FIRST.gyumolcs THEN LINK fejlec ;
 PUT  @5     gyumolcs      $12.
      @20    fajta         $18.
      @40    ar            7.2
      @55    keszlet       comma9.  ;
 RETURN ;
Fejlec:
 PUT  //// @20 'Gyümölcsök exportja'  ///
 @5 'Gyümölcs' @20 'Fajta' @40 'Ár Ft' @55 Exportálható kg' ;
RETURN ;
RUN ;
```

- Külső állomány írása

```
FILENAME lista 'c:\tanf\lista\putlis.lst' ;
LIBNAME  tanf 'c:\tanf\sasok' ;
DATA  _NULL_ ;
 SET  tanf.gyumexp ;
 BY gyumolcs ;
 FILE  lista  HEADER=fejlec n=64 PRINT ;
 IF FIRST.gyumolcs THEN LINK fejlec ;
 PUT  @5     gyumolcs      $12.
      @20    fajta         $18.
      @40    ar            7.2
      @55    keszlet       comma9.  ;
 RETURN ;
Fejlec:
 PUT  //// @20 'Gyümölcsök exportja'  ///
 @5 'Gyümölcs' @20 'Fajta' @40 'Ár Ft' @55 Exportálható kg' ;
RETURN ;
RUN ;
```

A futtatás eredménye:

GYÜMÖLCSÖK EXPORTJA

Gyümölcs	Fajta	Ár Ft	Exportálható kg
alma	Jonatán	105.75	150,625
alma	Golden	148.00	85,730
alma	Starking	139.00	6,725
alma	Húsvéti rozmaring	120.00	1,313
alma	Jonagold	142.00	66,334

GYÜMÖLCSÖK EXPORTJA

Gyümölcs	Fajta	Ár Ft	Exportálható kg
körte	Alexander	229.10	5,230
körte	Kálmán körte	178.80	1,560
körte	Vilmos körte	201.00	4,001

GYÜMÖLCSÖK EXPORTJA

Gyümölcs	Fajta	Ár Ft	Exportálható kg
szilva	Besztercei	98.00	2,900
szilva	Olasz kék	135.00	65,705
szilva	Althan-ringló	142.00	1,000

GYÜMÖLCSÖK EXPORTJA

Gyümölcs	Fajta	Ár Ft	Exportálható kg
sárgabarack	Magyar kajszi	197.00	11,123
sárgabarack	Rózsabarack	186.00	9,342
sárgabarack	Ananász barack	179.00	1,235

GYÜMÖLCSÖK EXPORTJA

Gyümölcs	Fajta	Ár Ft	Exportálható kg
őszibarack	Champion	250.00	29,111
őszibarack	Mariska	211.00	3,025

P10.5 GYAKORLATOK

1. FELADAT

A ***SZEMELY*** adatállományt használva írjunk ki egy riportot, amely tartalmazza az egyes korcsoportokat és a hozzájuk tartozó személyek számát. A korcsoportok a következőképpen jelenjenek meg:

KCSOP=1 ==> Fiatal
KCSOP=2 ==> Középkorú
KCSOP=3 ==> Idõs

2. FELADAT

A következő értékek súlyokat font-ban és magasságokat inch-ben reprezentálnak. Írjuk ki az outputon megformázva kilogrammban és centiméterben.

199 75
173 67
150 59

Segítség: 1 font = 45.36 dekagramm , 1 inch = 2.54 cm .

3. FELADAT

A következőknek esedékes befizetéseik vannak.
Írjuk ki a neveket és az összegeket úgy, ahogy a csekken is megfelelő lehet.
Pl. *****Ft25.50 formában

Kiss 125.60
Nagy 22.75
Tóth 9.35

4. FELADAT

Milyen formátummal olvasná be a következő dátumokat és időket egy SAS adatállományba:

01JAN99
230199
021898
21/11/96

7:00
19:00
10:21:02.00

15JUN99:12:15:00.00

5. FELADAT

Állítsunk elő egy-egy külső fájlt a ***HIVATAL*** állományunkból úgy, hogy a következő mezők kerüljenek outputra: NEV FOGLALK FIZETES PREMIUM.

A SAS programból hozzunk létre egy LISTA nevű könyvtárat a TANF directory alatt.

Az outputok a következők szerint íródjanak:

- LISTA1 → A mezőket 1 db space válassza el egymástól,
- LISTA2 → Ugyan az a változó mindig ugyan azon a helyen legyen
- LISTA3 → A változókat formátumokkal írassuk ki.
- LISTA4 → Minden változót a megnevezésével együtt írassunk ki.

P11 FEJEZET

TÁBLÁZATOK

11.1 A TABULATE ELJÁRÁS

11.2 TÁBLÁZATOK CSINOSÍTÁSA

11.3 GYAKORLATOK

P11.1 A TABULATE ELJÁRÁS

A TABULATE eljárás segítségével a leíró jellegű statisztikai elemzések eredményeit táblázat formájában jeleníthetjük meg.
Készíthetünk: egy-, kettő-, háromdimenziós táblákat,
gyakoriságokat,
értékösszegeket,
%-os megoszlásokat,
elemi statisztikákat,
és ezek kombinációját tartalmazó táblázatokat.

A TÁBLAKÉSZÍTÉS LÉPÉSEI

1. AZ ADATÁLLOMÁNY KIVÁLASZTÁSA

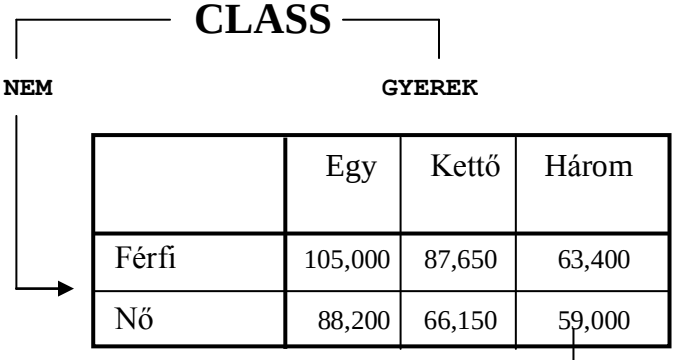
```
PROC TABULATE DATA=libref.sasdata  {egyéb_opciók} ;
```

Egyéb opciók: MISSING = a hiányzó értékeket tartalmazó megfigyelések is részt vesznek a táblázásban
FORMAT = formátumnév, amely a tábla minden egyes cellájához hozzárendeli a formátumot.

2. OSZLOP ÉS SOR VÁLTOZÓK DEFINIÁLÁSA (csoportképző, osztályozó változók)

```
CLASS változók ;
```

Pl.



	Egy	Kettő	Három
Férfi	105,000	87,650	63,400
Nő	88,200	66,150	59,000

3. A TÁBLA TARTALMÁNAK DEFINIÁLÁSA
(olyan tartalmi változók, amelyekre valamilyen statisztikát kérünk)

VAR
KERESET

```
VAR változók ;
```

4. A TÁBLA MEGJELENÉSÉNEK DEFINIÁLÁSA

```
TABLE  {kifejezés},{kifejezés},{kifejezés] {/opciók} ;
```

A TABLE utasítás

A TABLE utasítás több sorból áll és több mindent meghatároz:

- a táblázandó változókat,
- a tábla fizikai megjelenését,
- hova kerüljenek a változók a táblában,
- csinosítási opciókat, pl. formátumok, címek, szövegek,
- különböző szintű összeseneket.

A TABLE UTASÍTÁSNÁL ALKALMAZHATÓ MŰVELETI JELEK

A táblák megjelenítési formáját definiálják

CSILLAG	*	beágyazás
VESSZŐ	,	új dimenzió
SZÓKÖZ		konkatenálás
EGYENLŐSÉGJEL	=	változóhoz címkerendelés
KEREK ZÁRÓJELEK	()	csoportosítás, sorrend
CSÚCSOS ZÁRÓJELEK	<>	százalékjelző

A TABLE utasítás szintaktikája általánosan

```
TABLE              VÁLTOZÓ           = 'változó címke'
                   * STATISZTIKA     = 'statisztika címke'
                   * FORMÁTUM  = formátum
                   opcionális ÖSSZESEN specifikáció

                   MŰVELETI JEL

                   VÁLTOZÓ           = 'változó címke'
                   * STATISZTIKA     = 'statisztika címke'
                   *FORMÁTUM   = formátum
                   opcionális ÖSSZESEN specifikáció

                   MŰVELETI JEL

                   VÁLTOZÓ           = 'változó címke'
                   * STATISZTIKA     = 'statisztika címke'
                   *FORMÁTUM   = formátum
                   opcionális ÖSSZESEN specifikáció

                         .
                         .
                   / táblázási opciók
                   ;
```

Például:

1. `TABLE NEM;`
2. `TABLE NEM, CSAL * JOVED;`
3. `TABLE KOR='A megfigyeltek kora' , NEM='Nem';`
4. `TABLE NEM ALL='összesen' , JOVED*f=comma6.2;`

A PROC TABULATE ELJÁRÁS MŰKÖDÉSÉT NÉHÁNY PÉLDÁN KERESZTÜL PRÓBÁLJUK MEG ÉRZÉKELTETNI

CSAK EGY VÁLTOZÓNK VAN

```
PROC TABULATE DATA=tanf.demograf ;
 CLASS nem ;
 TABLE nem ;
Run;
```

CLASS utasítást használunk
Mivel a VAR utasítást kihagyjuk -- a cellákba a megfigyelésszám kerül
TABLE utasításban szerepel egy változó
nem szerepel operátor
összesen
formátum
címke

NEM	
F	N
N	N
5.00	5.00

ALAPÉRTELMEZÉSEK:
változónevek
értékek formátuma
statisztikák
oszlopok szélessége

TÖBB VÁLTOZÓNK VAN

Több változó esetén el kell döntenünk, hogyan kell kapcsolódniuk egymáshoz, melyikeket melyik dimenzióban kell szerepeltetnünk, egymás mellett, vagy egymásba ágyazva akarjuk e látni őket. Meg kell tehát fontolnunk, hogy a korábban említett műveleti jelek közül hol, és melyikeket alkalmazzuk.

SZÓKÖZ - konkatenálást hozunk létre

```
PROC TABULATE DATA=tanf.demograf ;
 CLASS nem csal ;
 TABLE nem csal ;
RUN;
```

CLASS utasítást használunk
VAR utasítást kihagyjuk – a cellákba megfigyelésszám kerül
TABLE utasításban szerepel két változó
egy operátor – a szóköz
nem szerepel összesen
formátum
címke

NEM		CSAL			
F	N	egy	elv	haz	ozv
N	N	N	N	N	N
5.00	5.00	2.00	1.00,	5.00,	2.00

CSILLAG - beágyazást hozunk létre vele

```
PROC TABULATE DATA=tanf.demograf ;
 CLASS nem csal ;
 TABLE nem*csal ;
RUN;
```

CLASS utasítást használunk
VAR utasítást kihagyjuk – a cellákba megfigyelésszám kerül
TABLE utasításban szerepel két változó
egy operátor – a csillag
nem szerepel összesen
formátum
címke

NEM				
F		N		
CSAL		CSAL		
egy	elv	haz	haz	ozv
N	N	N	N	N
2.00	1.00	2.00	3.00	2.00

VESSZŐ - új dimenziót hozunk létre vele

```
PROC TABULATE DATA=tanf.demograf ;
 CLASS nem csal ;
 TABLE nem,csal ;
RUN;
```

CLASS utasítást használunk
VAR utasítást kihagyjuk – a cellákba megfigyelésszám kerül
TABLE utasításban szerepel két változó
egy operátor – a vessző
nem szerepel összesen
formátum
címke

	CSAL			
	egy	elv	haz	ozv
	N	N	N	N
NEM				
F	2.00	1.00	2.00	.
N	.	.	3.00	2.00

3 DIMENZIÓS TÁBLÁK

További vessző alkalmazása a táblában a 3. dimenzió meghatározását jelenti.

Pl. **TABLE nem , csal , gyerek ;**

A harmadik dimenziót az először felsorolt változó, azaz a NEM határozza meg. Az elemzés a változó értékeinek megfelelően külön-külön készül el, vagyis külön jelenik meg a férfiakra, majd külön a nőkre vonatkozó tábla, amely a családi állapot és a gyerekek száma szerinti gyakoriságokat tartalmazza.

A VAR utasítás

A VAR utasításban adjuk meg azt a változót, amelyet a tábla celláiban akarunk szerepeltetni. Ezt a változót **analízis változó**nak nevezzük, és ***numerikusnak kell lennie***. Az analízis változót nem elég azonban csak a VAR utasításban szerepeltetni, hanem be kell ágyaznunk a TABLE utasításba, ezzel jelezvén, hogy a tábla adott helyén erről a változóról kérünk valamilyen statisztikát.

```
PROC TABULATE DATA=tanf.demograf ;
 CLASS nem ;
 VAR joved ;
 TABLE nem*joved ;
RUN;
```

CLASS utasítást használunk
VAR utasítást használunk – a cellákba a JOVED összegek kerülnek
TABLE utasításban szerepel két változó
egy operátor – a csillag
nem szerepel összesen
formátum
címke

NEM	
F	N
JOVED	JOVED
SUM	SUM
299500.00	440500.00

Mivel az alapértelmezés szerinti statisztika a **SUM** , az egyes cellákba most a megfigyelések száma helyett a JOVED változó összegét találjuk.

Összesenek - az ALL opció

Ahogy a TABLE utasításban specifikálhatjuk a tábla fizikai megjelenését és a táblában szereplő változókat, megadhatjuk azt is, hogy hol kívánunk összeseneket. Erre használjuk az ALL opciót. Bizonyos szempontból az ALL opció tekinthető a TABLE utasításban szereplő speciális „változónak” is. Ez az opció lehetővé teszi, hogy a tábla különböző helyein különböző mélységben definiáljunk összeseneket, és abban viselkedik hasonlóan a többi változóhoz, hogy elhelyezkedésére hatással vannak a szóköz, vessző, csillag és zárójel műveleti jelek.

Mivel a JOVED változó összesenjére vagyunk kíváncsiak, ezért az ALL opciót a JOVED alá kell beágyazni.

Az ALL opcióval sor, oszlop és mindösszesenek egyaránt létrehozhatók.

```
PROC TABULATE DATA=tanf.demograf ;
 CLASS nem ;
 VAR joved ;
 TABLE nem*joved all;
RUN;
```

CLASS utasítást használunk
VAR utasítást használunk – a cellákba a JOVED összegek kerülnek
TABLE utasításban szerepel két változó
két operátor -- a csillag és a szóköz
összesen -- ALL
nem szerepel formátum
címke

NEM		
F	N	
JOVED	JOVED	ALL
SUM	SUM	N
299500.00	440500.00	10.00

Láthatóan az összesen oszlopunk nem olyan, mint vártuk. A JOVED összesen helyett a személyek számát tartalmazza.

Összesenek beágyazása

Mivel a JOVED változó összesenjére vagyunk kíváncsiak, ezért az ALL opciót a JOVED alá kell beágyazni.

```
PROC TABULATE DATA=tanf.demograf ;
 CLASS nem ;
 VAR joved ;
 TABLE nem*joved all*joved;
RUN;
```

CLASS utasítást használunk
VAR utasítást használunk – a cellákba a JOVED összegek kerülnek
TABLE utasításban szerepel két változó
két operátor -- a csillag és a szóköz
összesen -- a JOVED változóra
nem szerepel formátum
címke

NEM		
F	N	ALL
JOVED	JOVED	JOVED
SUM	SUM	SUM
299500.00	440500.00	740000.00

Sor és oszlop összesenek

Az ALL opcióval sor, oszlop és mindösszesenek egyaránt létrehozhatók.

```
PROC TABULATE DATA=tanf.demograf ;
 CLASS nem csal ;
 TABLE csal all, nem all ;
RUN;
```

CLASS utasítást használunk
VAR utasítást nem használunk – a cellákba az N kerül
TABLE utasításban szerepel két változó
két operátor -- a vessző és a szóköz
összesen -- a CSAL és NEM változóra
nem szerepel formátum
címke

	NEM		
	F	N	ALL
	N	N	N
CSAL			
egy	2.00	.	2.00
elv	1.00	.	1.00
haz	2.00	3.00	5.00
ozv	.	2.00	2.00
ALL	5.00	5.00	10.00

STATISZTIKÁK A TÁBLÁKBAN

A TABULATE eljárás számos statisztika elkészítésére és kinyomtatására alkalmas.
A legfontosabb statisztikák a következők:

*** N**	**megfigyelések száma**
*** NMISS**	**hiányzó értékek száma**
*** MIN**	**minimum**
*** MAX**	**maximum**
*** SUM**	**összeg**
*** MEAN**	**átlag**
*** STD**	**szórás**
*** PCTN**	**százalék (gyakoriság)**
*** PCTSUM**	**százalék (összeg)**

Ha a tábla specifikációnál nem adunk meg statisztikát, alapértelmezésként kapjuk a SUM – ot. A többi statisztikát ugyanígy kell megadnunk, például a MEAN kulcsszóval kérhetünk átlagot.

```
PROC TABULATE DATA=tanf.demograf ;
 CLASS nem csal ;
 VAR joved ;
 TABLE csal all, nem*joved*mean all*joved*mean ;
RUN;
```

	NEM		
	F	N	ALL
	JOVED	JOVED	JOVED
	MEAN	MEAN	MEAN
CSAL			
egy	46500.00	.	46500.00
elv	123000.00	.	123000.00
haz	65000.00	78000.00	72800.00
ozv	.	103250.00	103250.00
ALL	74875.00	88100.00	82222.22

A tábla különböző részeiben lehetőségünk van különböző statisztikák kérésére.

Pl. `TABLE csal all, nem*joved*mean all*joved*sum ;`

SZÁZALÉKOS TÁBLÁK

PCTN vagy PCTSUM-mal kérhetünk

- sarokszázalékot
- oszlopszázalékot
- sorszázalékot.

ÖSSZEG SZÁZALÉK

A sarok összesent 100-nak tekintve kapjuk a cellák százalék értékeit.

```
PROC TABULATE DATA=tanf.demograf ;
 CLASS nem csal ;
 VAR joved ;
 TABLE csal all,
       nem*joved*pctsum all*joved*pctsum ;
RUN;
```

	NEM		
	F	N	ALL
	JOVED	JOVED	JOVED
	PCTSUM	PCTSUM	PCTSUM
CSAL			
egy	6.28	.	6.28
elv	16.62	.	16.62
haz	17.57	31.62	49.19
ozv	.	27.91	27.91
ALL	40.47	59.53	100.00

A SZÁZALÉKJELZŐ HASZNÁLATA OSZLOPSZÁZALÉK ESETÉN

```
PROC TABULATE DATA=tanf.demograf ;
 CLASS nem csal ;
 VAR joved ;
 TABLE csal all,
       nem*joved*pctsum<csal all> all*joved*pctsum<csal all> ;
RUN;
```

	NEM		
	F	N	ALL
	JOVED	JOVED	JOVED
	PCTSUM	PCTSUM	PCTSUM
CSAL			
egy	15.53	.	6.28
elv	41.07	.	16.62
haz	43.41	53.12	49.19
ozv	.	46.88	27.91
ALL	100.00	100.00	100.00

A SZÁZALÉKJELZŐ HASZNÁLATA SORSZÁZALÉK ESETÉN

```
PROC TABULATE DATA=tanf.demograf ;
 CLASS nem csal ;
 VAR joved ;
 TABLE csal all,
       nem*joved*pctsum<nem all> all*joved*pctsum<nem all> ;
RUN;
```

	NEM		
	F	N	ALL
	JOVED	JOVED	JOVED
	PCTSUM	PCTSUM	PCTSUM
CSAL			
egy	100.00	.	100.00
elv	100.00	.	100.00
haz	35.71	64.29	100.00
ozv	.	100.00	100.00
ALL	40.47	59.53	100.00

A százalékjelző használatával tetszőleges logikai összeseneket (részösszeseneket) választhatunk a százalékszámítás alapjául.

P11.2 TÁBLÁZATOK CSINOSÍTÁSA

CSINOSÍTÁSOK

Induljunk ki a következő táblából:

	NEM		
	F	N	ALL
	JOVED	JOVED	JOVED
	MEAN	MEAN	MEAN
CSAL			
egy	46500.00	.	46500.00
elv	123000.00	.	123000.00
haz	65000.00	78000.00	72800.00
ozv	.	103250.00	103250.00
ALL	74875.00	88100.00	82222.22

OSZLOPSZÉLESSÉG

A TABULATE eljárás FORMAT opcióval adható meg az oszlopok szélessége. Az itt megadott formátum minden oszlopra érvényes.

MISSTEXT OPCIÓ - Használatával a táblában tetszőleges szöveget írhatunk hiányzó értékként a **.** (pont) helyett. Ez az opció a TABLE utasítás végén / jel után adható meg. Ügyeljünk az oszlopszélességre, mert ha a beírt karakterek hossza túlmutat a definiált formátumon, rövidül a MISSTEXT-be beírt szöveg.

OSZLOPONKÉNT KÜLÖNBÖZŐ FORMÁTUMOK

Ez az opció lehetővé teszi, hogy különböző szélességben írjuk ki az oszlopokat. A TABLE utasításban a változó név után kell megadni. Az opció az adott oszlopra felülírja a TABULATE utasításban megadott formátumot.

OSZLOPONKÉNT EGYEDI CÍMKÉK

Bármelyik változóhoz, statisztikához egyedi címkéket rendelhetünk a TABLE utasításban.
Szintaktikája: ***változó='címke';***

FEJSZÖVEGEK ELTÁVOLÍTÁSA

Táblánkban a MEAN ill. ALL fejszövegek nem igazán szükségesek. Eltávolíthatjuk őket a ***MEAN=' '*** ill. az ***ALL=' '*** megadásával.

STATISZTIKÁK CÍMKÉZÉSE

A táblákban szereplő statisztikákhoz tartozó megnevezést (Pl. MEAN) egyszerre eltávolíthatjuk a TABLE utasítás után a KEYLABEL utasítás használatával.

AZ OLDALSZÖVEG SZÉLESSÉGE ÉS A FORMAT UTASÍTÁS

Az **RTS** opciót használhatjuk arra, hogy az oldalrovat szélességét megváltoztassuk. Emellett átkódolhatjuk a NEM és a CSAL változók értékeit is.

OLDALFEJ CSINOSÍTÁS

Az oldal felett megjelenő szöveget áttehetjük az oldat feletti üres boksz-ba a ***BOX=*** opció használatával.

Lássuk tehát a programunkat és táblánkat új köntösben:

```
LIBNAME tanf 'c:\tanf\sasok';
PROC FORMAT ;
  PICTURE forint
                LOW-HIGH='000,000,009.99' (prefix='Ft') ;
  VALUE  $nem  'F'='Férfi'
               'N'='Nő' ;
  VALUE  $csal  'haz'='Házas'
                'elv'='Elvált'
                'egy'='Egyedülálló'
                'ozv'='Özvegy' ;
PROC TABULATE DATA=tanf.demograf format=8. ;
  CLASS nem csal ;
  VAR joved ;
  TABLE csal=' '
          all='Átlagos jövedelem nemenként',
        nem*joved='Átlagos jövedelem'
        *mean
        *f=forint12.
        all=' '
        *joved='Átlagos jövedelem családi állapotoként'
        *mean
        *f=forint14.
          /RTS=15
           MISSTEXT='Nincs adat'
           BOX='Családi állapot' ;
  KEYLABEL MEAN=' ';
  FORMAT csal $csal.
         nem  $nem. ;
RUN;
```

Családi állapot	NEM		
	Férfi	Nő	Átlagos jövedelem
	Átlagos jövedelem	Átlagos jövedelem	családi állapotonként
Egyedülálló	Ft46,500.00	Nincs adat	Ft46,500.00
Elvált	Ft123,000.00	Nincs adat	Ft123,000.00
Házas	Ft65,000.00	Ft78,000.00	Ft72,800.00
Özvegy	Nincs adat	Ft103,250.00	Ft103,250.00
Átlagos jövedelem nemenként	Ft74,875.00	Ft88,100.00	Ft82,222.22

P11.3 GYAKORLATOK

1. FELADAT

Készítsünk táblát a ***SZEMELY*** adatállományból, amely azt mutatja, hogyan különbözik a gyerekek száma nemek szerint. Ha nem fér el a táblánk, csökkentsük az oszlopok szélességét!

2. FELADAT

Számítsuk ki az átlagos magasságot a ***SZEMELY*** adatállományunkból. Ennek alapján hozzuk létre egy olyan esetszámos táblát, amelynek fejrészében a nemek vannak, oldalában pedig az 'Átlag alatt' és 'Átlag felett' sorok

3. FELADAT

Hozzunk létre egy táblát a ***HIVATAL*** és a ***SZEMELY*** állományunkból, amely foglalkozásonként mutatja az egyes osztályokon nemekre lebontva a ***SZEMELY adatállományban is megtalálható dolgozók prémium összegét***. Képezzünk sor és oszlop összeseneket is. A táblánk külleme legyen elfogadható.

4. FELADAT

Az előző táblát hozzuk létre úgy, hogy a százalékos megoszlásokat mutassa. Állítsuk elő mind a három féle százalékot (sor, oszlop, sarok).

5. FELADAT

A 4. Feladatot ismételjük meg úgy, hogy megcseréljük a dimenziókat.

P12 FEJEZET

STATISZTIKÁK

P12.1 EGYVÁLTOZÓS STATISZTIKÁK

A MEANS eljárás

Ez az eljárás SAS adatállományokat elemez és változatos statisztikákat számol az állomány numerikus változóira.

SZINTAKTIKÁJA:

```
PROC MEANS DATA= libref.sasdata  OPCIÓK ;
      VAR változók ;
```

Az ***OPCIÓK***-ban pl. a következő statisztikák közül választhatunk:

MEAN SUM MIN MAX STD N

A változók kiválasztása a VAR utasítással történhet. Nélküle az összes numerikus változóra elkészülnek a kért statisztikák.
A kiírt tizedes jegyek számának korlátozása a ***MAXDEC***= opcióval lehetséges.
Csoportok szerinti elemzésre nyújt lehetőséget a ***BY*** utasítás használata.

```
TITLE 'MEANS eljárás – csoportok' ;
PROC SORT DATA=tanf.demograf OUT=rend MAXDEC=2 ;
      BY nem ;
PROC MEANS DATA=rend n min max mean ;
      BY nem ;
      VAR joved kor gyerek ;
RUN ;
```

Ez azonban mindig csak a BY-változóra rendezett állományra lehetséges.

A UNIVARIATE eljárás

A UNIVARIATE eljárással részletes leíró statisztikákat nyerhetünk numerikus változóinkról. Sok felhasználó szerint ez az egyik leghasznosabb eljárás a változók eloszlásának elemzésére és a kiterjedés, forma valamint egyéb leíró mértékek megállapítására.

Szintaktikája megegyezik a PROC MEANS eljárásával.
A BY utasítással szintén lehetséges a kategóriánkénti elemzés.

LEHETŐSÉGEI:

PERCENTILISEK
MEDIÁN
SZÉLSŐ ÉRTÉKEK
KAVANTILISEK
ILLISZTRATÍV ÁBRÁK
SÚLYPONT VIZSGÁLAT
NORMALITÁS VIZSGÁLAT

AZ UNIVARIATE OUTPUT ELEMEINEK JELENTÉSE

N	A megfigyelések száma
Mean	Átlag
Std Dev	Szórás
Skewness	Ferdeség
USS	Korrigálatlan szórásnégyzet
CV	Relatív szórás
T:Mean=0	T próba értéke
Num ^=0	Nem nulla értékek száma
Sum Wgts	Súlyok összege
Sum	Összeg
Variance	Szórásnégyzet
Kurtosis	Ferdeség
CSS	Korrigált szórásnégyzet
Std Mean	Standard hiba
Prob>¦ T ¦	A T próba szignifikancia szintje
100% Max	Maximumérték
75% Q3	Felső kvartilis
50% Med	Medián
25% Q1	Alsó kvartilis
0% Min	Minimumérték
Range	Terjedelem
Q3-Q1	A felső és az alsó kvartilis különbsége
Mode	Módusz
Extremes	Az öt legkisebb és az öt legnagyobb érték a megfigyelések sorszámával

P12.2 GYAKORISÁGI TÁBLÁK

A **FREQ eljárás** egy, kettő vagy több dimenziós táblát hoz létre numerikus és karakteres változóink gyakoriságáról.

Szintaktikája:

```
PROC FREQ DATA= libref.sasdata ;
 TABLE [vagy TABLES] változók / opciók ;
BY változók ;
RUN ;
```

EGYDIMENZIÓS TÁBLA esetén az adott változó(k)értékeihez tartozó előfordulásszámok jelennek meg az outputban.

```
TITLE 'Egydimenziós gyakorisági táblák' ;
PROC FREQ DATA=tanf.demograf ;
TABLE gyerek csal nem ;
RUN ;
```

Ha nem használunk TABLE utasítást, akkor minden változóról egydimenziós tábla készül.

KERESZTTÁBLÁK esetén az első változó értékei adják a sorokat, a második az oszlopokat.

```
TITLE 'Kétdimenziós gyakorisági táblák' ;
PROC FREQ DATA=tanf.demograf ;
TABLE gyerek*csal ;
RUN ;
```

Az output elemeinek jelentése

Egydimenziós táblák esetén:

Frequency	gyakoriság
Percent	relatív (százalékos) gyakoriság
Cummulative Frequency	kumulált gyakoriság
Cummulative Percent	kumulált relatív (százalékos) gyakoriság

Kereszttáblák esetén:

Frequency	gyakoriság
Percent	relatív (százalékos) gyakoriság
Row Pct	relatív (százalékos) gyakoriság soronként
Col Pct	relatív (százalékos) gyakoriság oszloponként

TABLE opciók

```
TITLE 'Kétdimenziós gyakorisági táblák' ;
PROC FREQ DATA=tanf.demograf ;
TABLE gyerek*csal  / norow nocol ;
RUN ;
```

Oszlopösszesenhez viszonyított %-os megoszlás elhagyása

Sorösszesenhez viszonyított %-os megoszlás elhagyása

Az eredmények megőrzése

nincs nyomtatott output

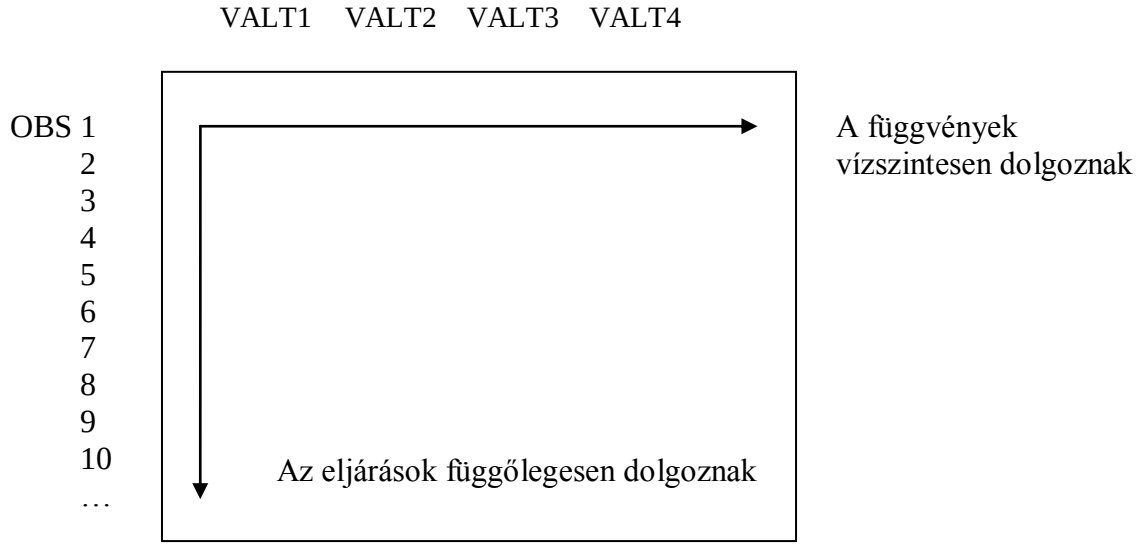
```
PROC FREQ DATA=tanf.demograf ;
TABLE  nem*gyerek*csal  / NOPRINT OUT=gyakor ;
RUN ;
TITLE 'PROC FREQ output állomány' ;
PROC PRINT DATA=gyakor ;
RUN ;
```

a gyakor nevű SAS állományba kerül a statisztika

P12.3 STATISZTIKAI FÜGGVÉNYEK

A statisztikai függvényeket általában arra használjuk, hogy egy megfigyelés különböző változóira végezzünk statisztikai számításokat, szemben pl a MEANS eljárással, amelyikkel egy változó átlaga számítható ki az összes megfigyelést tekintve.

SEMATIKUSAN ÁBRÁZOLVA:



Általános formájuk egyaránt:

```
eredményváltozó = függvénynév (argumentum_1, .. argumentum_n);
```

vagy változó sorozatok ill. tömbelemek esetén:

```
eredményváltozó = függvénynév (OF változó_1,változó2, ... változó_n);
eredményváltozó = függvénynév (OF változó_1 - változó_n);
eredményváltozó = függvénynév (OF tomb(*)) ;
```

Listájuk a **P7.1** pont alatt található

Most nézzünk egy példát, amelyben az adataink 5 nap éjfélkor, 6,12 és 18 órakor mért hőmérsékleti értékét tükrözik:

```
TITLE 'Napi hőmérsékleti statisztika';
DATA napihom;
  INPUT datum $ hom00 hom06 hom12 hom18 ;
  atlhom=MEAN(hom00,hom06,hom12,hom18);
  minhom=MIN(hom00,hom06,hom12,hom18);
  maxhom=MAX(hom00,hom06,hom12,hom18);
CARDS;
01JUN96 15.2 17.1 25.3 21.7
02JUN96 16.0 18.0 27.0 22.0
03JUN96 16.5 18.5 27.5 22.2
04JUN96 17.2 21.8 29.9 23.1
05JUN96 16.9 20.3 28.6 20.4
RUN;
PROC PRINT NOOBS;
RUN;
```

Az eredmény:

Napi hőmérsékleti statisztika

datum	hom00	hom06	hom12	hom18	atlhom	minhom	maxhom
01JUN96	15.2	17.1	25.3	21.7	19.825	15.2	25.3
02JUN96	16.0	18.0	27.0	22.0	20.750	16.0	27.0
03JUN96	16.5	18.5	27.5	22.2	21.175	16.5	27.5
04JUN96	17.2	21.8	29.9	23.1	23.000	17.2	29.9
05JUN96	16.9	20.3	28.6	20.4	21.550	16.9	28.6

P12.4 GYAKORLATOK

Futtassuk le a megismert statisztikákat a ***tanf*** könyvtárban létező SAS adatállományokra és értelmezzük az eredményeket!

P13. FEJEZET

KIEGÉSZÍTÉSEK A SAS PROGRAMOZÁSHOZ

P13.1 VÁLTOZÓK MEGADÁSA

VÁLTOZÓK HOSSZÁNAK ÉS TÍPUSÁNAK MEGADÁSA

Egy változónak megváltoztathatjuk az alapértelmezés szerinti hosszát a LENGTH utasítással:

LENGTH ***változónév*** **{$}*hossz*;**

Karakteres változónál kötelező a dollár megadása. Nélkülözhetetlen a LENGTH utasítás akkor, ha külső fájlból listás olvasással olvasunk be nyolcnál hosszabb karakteres adatokat, amikor is a hossz alapértelmezése nyolc.
Sokszor célszerű helytakarékossági okokból a numerikus változók hosszát is megváltoztatni, aminek alapértelmezése nyolc, a legkisebb megadható érték Windows alatt és Unix-ban három. Ezzel az ábrázolható szám nagyságát nem csökkentjük, viszont a pontosságát jelentősen.

VÁLOZÓK LISTÁJA

A változók felsorolásának léteznek egyszerűbb módjai is:

SORSZÁMOZOTTAK A VÁLTOZÓ NEVEINK:

Amennyiben a változóink v1, v2 … vn, ezt megadhatjuk **a1-an** formában is, ha a változók sorszámozása folyamatos és növekvő.

INTERVALLUM MEGADÁSA

Felsorolás helyett a változónevek intervallumát is megadhatjuk, ami a PDV-ben lévő helyük szerinti intervallumot jelent.

valami – akarmi a megadás formája, ami azt jelenti, hogy a programban a ***valami*** nevű és az ***akarmi*** nevű változók definíciója között definiált változókat akarjuk felsorolni. Pl.

```
DATA …;
INPUT elso masodik harmadik negyedik;
…
KEEP elso – harmadik;
```

Ebben a programrészben a KEEP utasítás hatására az ***elso***, a ***masodik*** és a ***harmadik*** nevű változók kerülnek be az output adatállományba.

MASZK HASZNÁLATA

Megengedett az azonos karakterekkel kezdődő változó nevekre való hivatkozás is.

Pl. ***TA:*** az összes TA-val kezdődő változónevet jelenti.

ADATÉRTÉKEK MEGVÁLTOZTATÁSA

Nagyon gyakran van szükségünk arra, hogy magukat az adatértékeket változtassuk meg, kódoljuk át. Ezt a konverziót a PUT és az INPUT függvénnyel tehetjük meg.

PUT függvény

```
eredményváltozó = PUT (változó,formátum) ;
```

A PUT függvényt a változó értékét az alkalmazott formátummal írja ki az outputba. Leggyakrabban numerikus értékek karakteressé konvertálására használjuk.

- a változónak és a formátumnak azonos típusúnak kell lennie)mindkettő karakteres, vagy numerikus).
- az eredményváltozó mindig egy karaktersorozat
- ha a változó és a formátum numerikusak, akkor az eredmény jobbra igazított lesz, ha mindkettő karakteres, akkor pedig balra
- az eredményváltozó hossza a formátumban meghatározott lesz

PÉLDA KARAKTERES VÁLTOZÓVAL:

A

```
DATA;
  chardata ='BE';
  hexdata=PUT(chardata,$hex4.);
  PUT hexdata;
RUN;
```

program a HEXDATA nevű változó értékét, amely **4245** , a LOG-ba írja.

PÉLDA NUMERIKUS VÁLTOZÓVAL:

A

```
DATA;
  numdata =1999;
  r=PUT(numdata,roman8.);
  b=PUT(numdata,binary14.);
  e=PUT(numdata,e.);
  d=PUT(numdata,date7.);
  PUT _ALL_;
RUN;
```

program a következőt írja a LOG-ba:

```
numdata=1999 r=MCMXCIX b=00011111001111 e=1.99900E+03 d=22JUN65
_ERROR_=0 _N_=1
```

Az automatikus konverzió, amely numerikusból karakteressé konvertál, a BEST12. Formátumot használja, eredménye jobbra igazított karaktersorozat lesz. A LEFT függvény segítségével alakíthatjuk át az eredményt a kívánt formátumúvá.

INPUT függvény

```
eredményváltozó = INPUT (változó,input_formátum) ;
```

Az INPUT függvény az értékhez rendelt formátumnak megfelelően olvassa be a változót. Leggyakrabban a karakteres értékek numerikussá konvertálásához használjuk.

A
```
DATA konv;
  karv='123,456.78';
  numv=INPUT(karv,COMMA10.);
  PUT karv numv;
RUN;
```

program a 123,456.78 123456.78 értékeket írja a LOG-ba,
s ha megnézzük a KONV állomány CONTENTS-ét, a KARV változó típusa karakteres, a NUMV-é pedig numerikus.

A két függvény természetesen kombinálhatjuk is:

A
```
DATA kombi;
  ert=11282;
  dat=INPUT(PUT(ert,Z6.),DDMMYY6.);
  PUT dat WORDDATE.;
RUN;
```

program a LOG-ba a következő dátumot írja: December 1, 1982

P13.2 CIKLUSOK

A DO ciklus

A DO CIKLUS LEGGYAKORIBB FORMÁJA:

```
DO index=kezdet TO vég BY növekmény;
      további_SAS_utasítások
END;
```

TUDNIVALÓK:
- az indexváltozó belekerül az output állományba, hacsak nem adjuk meg egy DROP utasításban.
- a kezdet, vég illetve a növekmény értéke:
  - = a ciklus elején beállítódik
  - = a ciklus közben nem változtatható meg,
  - = lehet konstans, változó ill. aritmetikai kifejezés.
- a BY kulcsszó és a növekmény opcionális, a növekményre az alapértelmezés 1.
- az index változó értéke a ciklus végén növekszik a növekmény értékével.
- az indexváltozó értékének összehasonlítása a vég értékkel a ciklus elején történik.

ENNEK A TÍPUSNAK EGY MÁSIK MEGOLDÁSI MÓDJA:

```
DO változó=konstans_1, konstans_2, konstans_3, … ;
  SAS_utasítások
END;
```

Ezek a ciklusok meghatározott számú iterációt definiálnak.

FELTÉTELES ITERÁCIÓK

```
DO WHILE (logika_kifejezés);
  SAS_utasítások
END;
```

TUDNIVALÓK:
- a ciklus addig hajtódik végre, amíg a logikai kifejezés igaz
- a logikai kifejezés a ciklus elején értékelődik ki, így lehet, hogy a ciklus egyáltalán nem hajtódik végre

```
DO UNTIL (logikai_kifejezés);
  SAS_utasítások
END;
```

TUDNIVALÓK:
- a ciklus addig hajtódik végre, amíg a logikai kifejezés igazzá nem válik
- a logikai kifejezés a ciklus végén értékelődik ki, így a ciklus legalább egyszer végrehajtódik.

P13.3 TÖMBÖK

A tömbök összefüggő változók csoportjának megadására szolgálnak. Kétféle tömb létezik a SAS-ban: implicit ill. explicit definiált tömbök.

ÁLTALÁNOS FORMÁJA:

```
ARRAY tömbnév {n} $ hossz tömbelemek {(kezdőértékek)} ;
```

ahol:

tömbnév	- a tömb SAS-névszabálynak megfelelő neve
n	- a tömb elemszáma
$	- karakteres változókat tartalmazó tömb esetén kell csak megadni
hossz	- opcionális, a változók (közös) hosszát lehet vele megadni, ha korábban ezt LENGTH utasítással nem tettük meg
tömbelemek	- a tömb változói, ha egyedi változókat akarunk tömbként kezelni, de használható a _TEMPORARY_ kulcsszó, amellyel munkaváltozókat helyettesítünk tömbbel (ez a tömb nem is kerül outputra, mert nincs tényleges változó mögötte).
kezdőértékek	- az egyes tömbelemeknek adhatunk így kezdőértéket. Ezeket az értékeket szóközzel, vagy vesszővel kell elválasztani. Ha több elem van, mint kezdőérték, akkor a maradék változók hiányzó értéket kapnak. A kezdőérték adás minden iterációs lépésnél megismétlődik.

TUDNIVALÓK:

- egy tömb nem tartalmazhat karakteres és numerikus változókat egyszerre
- a tömbre nem hivatkozhatunk a definíciója előtt
- a tömb a korábban nem definiált változókat felveszi a PDV-be
- tömb definíció nem terjed túl a DATA lépés határain. Ha egy változó csoportot egy DATA lépésben tömbnek definiáltunk, akkor egy következő lépésben újra kell definiálnunk, ha megint szükségünk van rá.
- az ARRAY utasítás nem végrehajtható
- a tömbök lehetnek több dimenziósak is
- tömbnevek nem szerepelhetnek LABEL, FORMAT, DROP, ill. KEEP utasításokban
- a tömb elemeire hivatkozhatunk akár változónévvel, akár a tömb nevét indexelve.

EXPLICIT tömbnek nevezzük azt a tömböt, ahol konstanssal megadjuk a tömb méretét, **IMPLICIT** tömbnek pedig azt, ahol az elemszámot kihagyjuk (számolja meg a SAS a tömb-elemeket), vagy változóban adjuk meg.

Implicit tömb definiálása esetén egy speciális **iterációs DO utasítást** is használhatunk és a tömb elemeit egyként kezelhetjük.

```
DO OVER tömbnév ;
SAS_utasítások
END ;
```

A ciklus annyiszor hajtódik végre, ahány elemű a tömb. Ez a forma kiválóan alkalmas arra, hogy egy tömb elemeire egyforma műveletet hajtsunk végre.

Pl.:
```
ARRAY tomb a1-a8 ;
DO OVER tomb ;
      tomb=0 ;
END ;
```

Ezzel az egész tömböt kinulláztuk anélkül, hogy a tömb méretét vagy a változók neveit megemlítettük volna.

P13.4 MAKRÓK

A **MAKRÓ NYELV** működését a következő lényeges csoportokba sorolhatjuk:

- adatokat nyer ki a rendszerből, mint pl. ha az aktuális dátumot szeretnénk megjeleníteni a TITLE utasításban
- a DATA vagy PROC step feltételtől függő végrehajtását vezérelheti
- az utasítások egy sorozatát – melyet gyakran használunk – egy „makró csoport”-ba foglalhatjuk, elhelyezhetjük a programunk megfelelő részébe, és a makró csoport nevére hivatkozva azt tetszőleges számban végrehajthatjuk
- a DATA és a PROC lépéstől függő adatokat generálhatunk
- a SAS lépések között adatokat adhatunk át
- ismétlődő SAS kódokat (utasításokat) generálhatunk.

MAKRÓ VÁLTOZÓK

A rendszerben léteznek ún. **AUTOMATIKUS MAKRÓ VÁLTOZÓK**, amelyek közül bármelyiket használhatjuk. A hivatkozás ezekre a következőképpen történik:

```
&automatikus_makró_változó_neve
```

Ha idézőjelek között használjuk a makró változókat – pl. a TITLE utasításban a futtatás időpontját tartalmazó SYSDATE automatikus makró változót, - dupla idézőjelet kell használnunk.

Pl `TITLE „A lista készítés dátuma: &sysdate „ ;` utasítás hatására az output listán a következő cím jelenik meg: **A lista készítés dátuma: 15DEC99**

Mi magunk is hozhatunk létre **MAKRÓ VÁLTOZÓ**-t a következőképpen:

```
%LET makró_változó = érték ;
```

Az érték megadása nem kötelező, ha nem adjuk meg, a makró változó akkor is létrejön, értéke 0 lesz. A vezető és a befejező szóközöket a rendszer figyelmen kívül hagyja. Az összes többi karaktert tartalmazhatja a makró változó, beleértve tetszőleges számú felsővesszőt is.

Pl.
```
%LET vars=fajta ar ;
PROC PRINT DATA=tanf.gyumexp ;
VAR &vars ;   .....
```

program hatására csak a gyümölcsfajták és a hozzájuk tartozó árértékek kerülnek outputra.

MAKRÓ CSOPORT

A makró csoport - kezdete: **%MACRO *makrónév* [(*paraméter_1*, … *paraméter_n*)] ;**
- vége: **%MEND makrónév ;**

A makró hívása a nevének, és ha van, az átadandó paraméternek a megadásával történik. Az utasítás végét nem kell pontosvesszővel lezárni.

A **%MACRO** definíción belül a **ciklusszervezés**re a **%DO** a **%END** a **%TO** makró utasításokat kell használnunk.

Pl.
```
%MACRO lista (valt);
  %DO  i=1  %TO  5 ;
      PROC TABULATE DATA = lib.adat&i ;
        BY … ;
        VAR &valt ;
        TABLE  … ;
      RUN ;
  %END ;
%MEND lista ;
%lista (v1,v2, ..)
```

program hatására a TABULATE eljárás ötször hajtódik végre, úgy, hogy mindegyik alkalommal más-más adatállományt használ.

LÉPÉSEK KÖZÖTTI ÉRTÉKÁTADÁS

Nézzünk erre egy példát. A gyumexp állomány tételes listájának tetején szerepeltetni szeretnénk valamilyen állományszintű értéket, pl. az átlagárat.

```
/* AZ ÁTLAGÁR KISZÁMÍTÁSA */
PROC MEANS DATA=tanf.gyumexp noprint mean ;
VAR ar ;
OUTPUT OUT=atl MEAN=atlagar ;
RUN ;
/* az ATLAGAR változó tartalmát elhelyezzük az ATLAG nevű változóba */
DATA  _null_ ;
  SET atl ;
  CALL SYMPUT ('atlag',LEFT(PUT(atlagar,87.2))) ;
RUN ;
/* AZ ATLAG  MAKRÓ VÁLTOZÓ ÉRTÉKÉNEK FELTÜNTETÉSE A LISTA CÍMÉBEN */
PROC PRINT DATA=tanf.gyumexp ;
  TITLE  „Az átlagár  &atlag  Ft” ;
RUN ;
```

A makró nyelv még igen bőséges lehetőségekkel rendelkezik, ezek elsajátíttatása azonban jelen tanfolyamunknak nem feladata.

P13.5 AZ INFILE UTASÍTÁS NÉHÁNY OPCIÓJA

HOSSZÚ REKORDOK

Egyszerű INFILE utasítással csak max. 256 hosszú külső fájlt tudunk beolvasni. Többnyire viszont ennél hosszabb rekorddal van dolgunk. Ilyenkor meg kell adnunk az INFILE utasításban a rekord méretét LRECL kulcsszó után:

```
INFILE 'fájlnév' LRECL=n;
```

FÁJLVÉGE

Gyakran van arra szükség, hogy egy fájl olvasásának végén hajtsunk végre bizonyos műveleteket. Lehetőség van arra, hogy figyeljük a fájl végét akár külső fájlt olvasunk, akár SAS adatállományt. Két opció használható ennek kezelésére az **INFILE** ill. a **SET** utasításban:

```
END=változó        vagy        EOF=címke.
```

Ha a SAS fájlvégét érzékel, akkor az END opció esetén az ott megadott változó értéke egy lesz, míg egyébként nulla, EOF opció esetén a végrehajtás az EOF után megadott címkén folytatódik.
Érvényes SAS címke: egy érvényes SAS név kettősponttal lezárva.
Tudnivalók: az END után megadott változó nem kerül outputra.

VIGYÁZAT! Nem használhatjuk az END opciót in-stream adatok esetén (amikor egy CARDS, vagy DATALINES után jönnek az adataink), ill. ha a külső fájlból egyszerre több rekordot olvasunk (#n használatával).

P13.6 UNIX-SZAL KAPCSOLATOS KIEGÉSZÍTÉSEK

TAGSORT OPCIÓ - A RENDEZÉS OPTIMALIZÁLÁSA

Sok mezőből álló nagy adatállományok rendezésekor hasznos lehet a TAGSORT opció használata. Ennek hatására a SORT eljárás nem a teljes megfigyelést rendezi, külön tárolja a kulcsokat és a megfigyelések helyét az állományban, és ezt rendezi.
Szintaktikája:

```
PROC SORT DATA=input_sasdata {OUT=output_sasdata} TAGSORT;
```

COMPRESS - SAS ADATÁLLOMÁNYOK SŰRÍTETT TÁROLÁSA

Ha szűkében vagyunk a lemezhelynek, lehetőségünk van arra, hogy az adatokat sűrítve tároljuk. Ez azonban a CPU idő megnövekedésével fog járni, mivel minden egyes megfigyelés olvasásakor azt futás közben a memóriában fogja a SAS kicsomagolni.

Sűrítést kérhetünk a SAS futásban létrejövő minden állományra az

```
OPTIONS COMPRESS=YES;
```
utasítással,

vagy adatállomány opcióként egyetlen adatállományra,

pl. `DATA afile (COMPRESS=YES);`

A UNIX ALATT HASZNÁLHATÓ SPECIÁLIS BILLENTYŰKOMBINÁCIÓK

CTRL C - az előtérben (Display Manager módban) elküldött program megszakítása

CTRL D - az aktuális karakter törlése

CTRL E - a kurzortól a sor végéig törlés

CTRL T - vissza TAB

CTRL X - insert mód (kapcsoló, újra megnyomva kikapcsoljuk.)

CTRL Ű - Bye – kilépés a SAS-ból.