

Mindenekelőtt a korábbi közleményeinkben előfordult hibákat igazítjuk helyre.

– Az 1976. évi 6. szám 21. oldalán szó esik a programlapon az utasításmezőben elhelyezhető karakterek számáról. Erre nézve  $n \leq 66$  a helyes korlát.

– Az 1976. évi 7. szám 76. oldalán szerepel egy nyomtatási utasítás negyedik variánsa. Ez helyesen az alábbi:

```
2 | | FORMAT(///2(//46X,3F12.5))
```

*A januári számban kitűzött feladataink megoldásai*

1. Készítsünk programot négyzet alapú csonka gúlák térfogatának kiszámítására, ha az egyik négyzet területe minden számításnál egységnyi, a másiké  $1 \leq a^2 \leq 5$  között változik.  $\Delta a^1 = 0,25$  lépésközzel. Az alábbi nyolc magassággal kell a térfogatot számítani:  $m_1 = a$ ;  $m_2 = \sqrt{a}$ ;  $m_3 = a^2 - 1$ ;  $m_4 = \sqrt{a^2 + 1}$ ;  $m_5 = \sqrt{1 + \sqrt{a}}$ ;  $m_6 = \sqrt{a^2 - a - 1}$ ;  $m_7 = \sqrt[3]{a^2}$ ;  $m_8 = (a + 1)/\sqrt{a} + 1$ . Az oszlopok felett nyomtassuk ki az  $M$ -betűt és közvetlen mellette az indexeket. A térfogatokat négy tizedesjeggyel nyomtassuk. A program utasításfüggvény felhasználásával készüljön.

### Megoldás

Programunkban utasításfüggvény számítja a csonka gúla képletét VOL azonosítóval deklaráltuk, két formális paramétere a csonkagúla magassága és az egyik négyzet oldala. A ciklusban egy TERF azonosítójú mátrixot töltünk fel úgy, hogy minden oszlopban más értékadó utasítással számítjuk a térfogatot. Figyeljük meg, hogy itt a VOL aktuális paraméterei esetenként kifejezések, azokon belül standard függvények is szerepelhetnek. A ciklus befejeztével egy I azonosítójú egydimenziós tömb (vektor) betöltése következik, amely az M mellé nyomtatandó indexeket tartalmazza. Ezeket az első WRITE utasítással írjuk a mátrix „fejlécébe”. Az alábbiakban bemutatjuk a programot.

```

1 | MASTER PHAT
2 | DIMENSION TERF (17,8), I(8)
3 | VOL(H,A)=H*(A**2+A+1)/3.0
4 | AREA=1.0
5 | DO 1 K=1,17
6 |   A=SQRT(AREA)
7 |   TERF(K,1)=VOL(A,A)
8 |   TERF(K,2)=VOL(SQRT(A),A)
9 |   TERF(K,3)=VOL(A**2-1,A)
10 |  TERF(K,4)=VOL(SQRT(A**2+1),A)
11 |  TERF(K,5)=VOL(SQRT(1+SQRT(A)),A)
12 |  TERF(K,6)=VOL(SQRT(ABS A**2-A-1),A)
13 |  TERF(K,7)=VOL(A**(2/3),A)
14 |  TERF(K,8)=VOL((A+1)/(SQRT(A)+1),A)
15 | AREA=AREA+0.25
16 | DO 2 J=1,8
17 |   I(J)=J
18 |   WRITE(3,3)(I(J),J =1,8)
19 |   FORMAT(1H1,10(/),15X,8(4X,1HM,I1,4X)///)
20 |   WRITE(3,4)((TERF(K,J),J=1,8),K=1,17)
21 |   FORMAT(17(15X,8F10.4/))
22 | STOP
23 | END
```

2. Készítsünk programot a kétismeretlenes egyenletrendszer megoldására. A megoldásban alkalmazzunk másodrendű determinánsokat, s ezeket utasításfüggvényben számíttassuk ki. Az

$$a_{11}x + a_{12}y = b_1$$

$$a_{21}x - a_{22}y = b_2$$

egyenletrendszert kell megoldani, ha

$$\begin{array}{llll} 1 \leq a_{11} \leq 10 & \text{és} & \Delta a_{11} = -1; & 8,8 \leq a_{12} \leq 10,6 \quad \text{és} \quad \Delta a_{12} = -0,2; \\ -6 \leq b_1 \leq 7,5 & \text{és} & \Delta b_1 = -1,5; & 1,7 \leq a_{21} \leq 4,4 \quad \text{és} \quad \Delta a_{21} = -0,3; \\ 2,6 \leq a_{22} \leq 9,8 & \text{és} & \Delta a_{22} = 0,8; & 0 \leq b_2 \leq 4,5 \quad \text{és} \quad \Delta b_2 = 0,5. \end{array}$$

A program nyomtassa ki az egyenletrendszereket és tőlük jobbra két tizedesjegy pontossággal a megoldásokat, ha azok léteznek. Ha nincs pontosan egy megoldás, akkor a VÉGTELEN SOK MEGOLDÁS, ill. a NINCS MEGOLDÁS szöveg nyomtatandó, aszerint, hogy melyik eset áll fenn.

### Megoldás

A determinánsok:

$$D = \begin{vmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{vmatrix} = a_{11} \cdot a_{22} - a_{21} \cdot a_{12} \text{ (ez az ún. rendszerdetermináns)}$$

$$D_x = \begin{vmatrix} b_1 & a_{12} \\ b_2 & a_{22} \end{vmatrix} = b_1 \cdot a_{22} - b_2 \cdot a_{12} \quad D_y = \begin{vmatrix} a_{11} & b_1 \\ a_{21} & b_2 \end{vmatrix} = a_{11} \cdot b_2 - a_{21} \cdot b_1$$

A megoldást az  $x = \frac{D_x}{D}$  és  $y = \frac{D_y}{D}$  összefüggések adják meg. Az egyenletrendszernek mindig van pontosan egy megoldása, ha  $D \neq 0$ . (A koordináta-rendszerben a két egyenlet két egymást egy pontban metsző egyenes képét határozza meg.)

Ha viszont  $D = a_{11} \cdot a_{22} - a_{12} \cdot a_{21} = 0$ , akkor  $a_{11} \cdot a_{22} = a_{12} \cdot a_{21}$ , azaz  $\frac{a_{11}}{a_{21}} = \frac{a_{12}}{a_{22}} = n$ .

Az egyik egyenlet ismeretlenjeinek együtthatóit egy  $n$  valós számmal való szorzás útján állíthatjuk elő a másik egyenlet megfelelő együtthatóiból, azaz  $a_{11} = n \cdot a_{21}$  és  $a_{12} = n \cdot a_{22}$ .

Az egyenletrendszer megoldása szempontjából most két eset lehetséges.

**a)**  $D = 0$  és  $b_1 = n \cdot b_2$ .

Ekkor nyilván  $\frac{a_{11}}{a_{12}} = \frac{a_{12}}{a_{22}} = \frac{b_1}{b_2}$ , amiből látjuk, hogy az egyik egyenlet a másiknak következménye, az egyenlet-rendszert végtelen sok számpár elégíti ki. (A két egyenletnek megfelelő egyenes a koordináta-rendszerben egybeesik, minden pontjuk közös.)

**b)**  $D = 0$ , de  $\frac{b_1}{b_2} \neq n$ .

Ilyenkor az egyenletrendszert ellentmondásosnak mondjuk, nincs olyan számpár, amely kielégítené. (Az egyenleteknek a koordináta-rendszerben párhuzamos egyenesek felelnek meg.) Belátható, hogy ilyenkor  $D_x \neq 0$  és  $D_y \neq 0$ .

Ha egyikük is zérus lenne, akkor  $\frac{b_1}{b_2} = n$  lenne, ami kiinduló feltételünknek ellentmond.

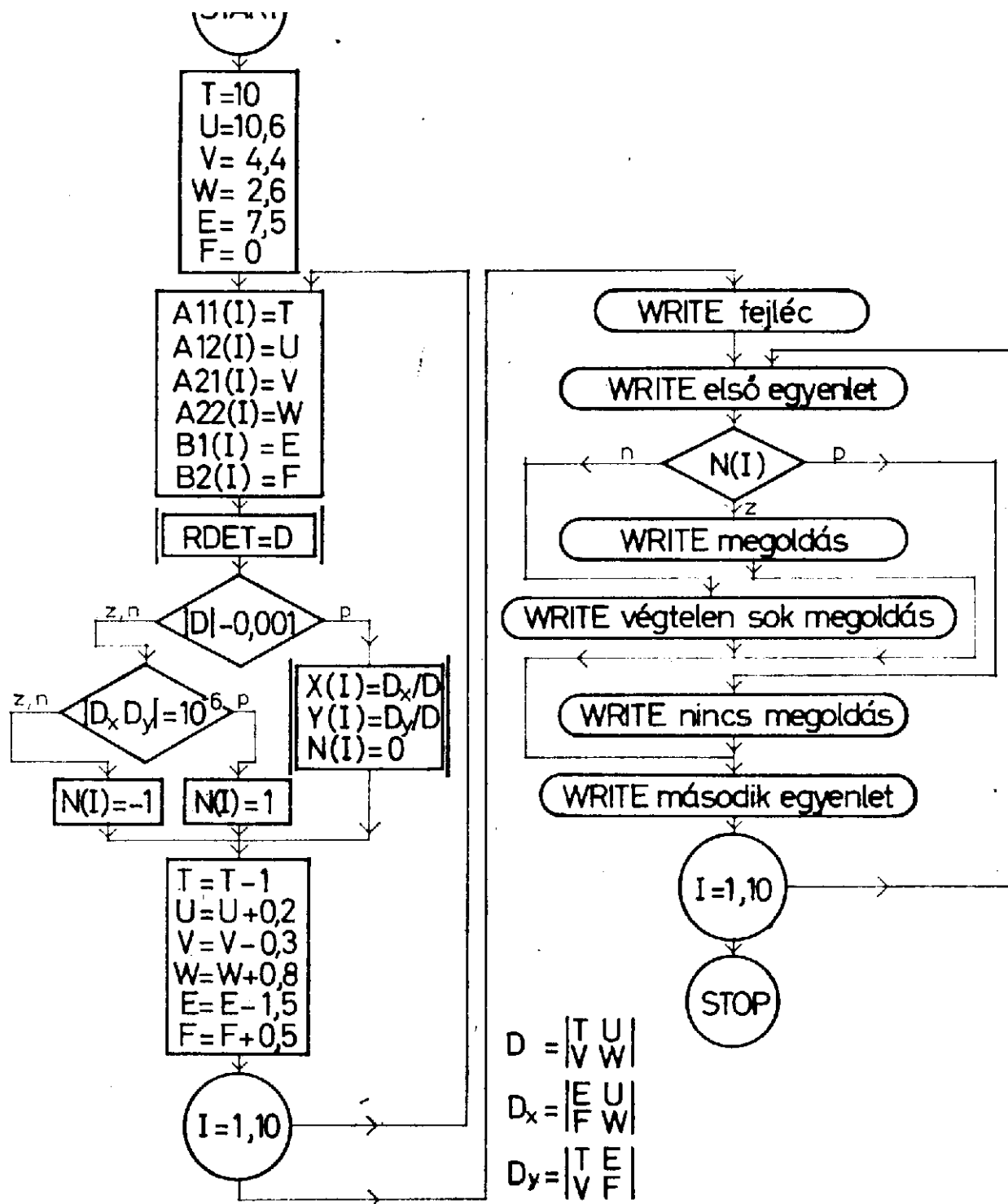
Programunkban tehát először megvizsgáljuk  $D$ -t, ha ez nem zérus, úgy képezzük a megoldást, ha zérus, akkor újabb vizsgálatot tartunk. Ha  $D_x$  és  $D_y$  közül csak egyik is zérus, akkor végtelen sok megoldás van, ha viszont egyik sem zérus, akkor egy megoldás sincsen.

A programban a determinánsokat egy DET azonosítójú utasításfüggvény számítja, amelynek négy formális paramétere rendre a determináns első és második sorának elemei.

Az egyenletrendszer hat paraméterének mindenkor (aktuális) értékeit a **T, U, V, W, E, F** azonosítók tartalmazzák. A zérusra történő vizsgálat tulajdonképp azt nézi, hogy a szám beleesik-e egy zérus körüli zárt intervallumba. Erre azért kényszerülünk, mert a valós típusú számok – a gépen belüli kerekítési, folyamatok miatt – akkor sem lesznek pontosan zérus értékűek, amikor a számítás szerint azoknak kellene lenniük. Ezért bizonyos korlátnál és az alatt minden számított pozitív valós számot zérusnak tekintünk. Feladatunkban a rendszerdetermináns abszolút értékét zérusnak tekintjük, ha az 0,001-nél nem nagyobb. Ugyanis az ismeretlenek együtthatói egy tizedesjegy pontossággal vannak megadva, a rendszerdeterminánsok legfeljebb két nem zérus értékű tizedesjegyet tartalmazhatnak, a harmadik tizedesjegy zérus értékű kell, hogy legyen.

Ezután következik a másik két determináns,  $D_x$  és  $D_y$ , melyek közül elegendő lenne egyiket vizsgálni, ha számolhatnánk annak pontos zérus értékével. Mivel nem így van, mindkettőt vizsgáljuk, éspedig a szorzatukat. Ha  $|D_x \cdot D_y| = 10^{-6}$ , akkor a determinánsokat zérusnak tekintjük.

A ciklus 10 menetében 10 egyenletrendszer paramétereit az **A11, A12** stb. tízelemű egydimenziós tömbökbe helyezzük, az eredményeket az ugyanilyen **X** és **Y** tömbökbe. Az **N** tömb jelzőszámokat tartalmaz. Egy adott egyenletrendszer esetében **N** = 0, ha az egy megoldás, **N** = -1, ha végtelen sok megoldása van, **N** = +1, ha az egyenletrendszernek nincs megoldása.



A papír bal felére az egyenletrendszereket, tőlük jobbra, a két egyenletrendszer közötti sorral egy vonalban, a megoldásokat nyomtatjuk. A két oszlop fölé feliratot íratunk ki. Az ábra a program blokkdiagramját mutatja. Ebben a kettős vonalú blokkok szubrutinhasználatra utalnak. (A rövid írásmód céljából a blokkban használtuk a  $D$ ,  $D_x$  és  $D_y$  jeleket.)

A program egy lehetséges formája az alábbi:

```

MASTER PHAT
DIMENSION A11(10), A12(10), A21(10), A22(10), B1(10), B2(10), X(10)
DIMENSION Y(10), N(10)
DET (P,Q,R,S)=P*S-R*Q
T=10.
U=10.6
V=4.4
W=2.6
E=7.5
F=0
DO 1 I=1,10
A11(I)=T
A12(I)=U
A21(I)=V
A22(I)=W
B1(I)=E
B2(I)=F
RDET=DET(T,U,V,W)
IF(ABS(RDET-0.001)) 2,2,0
X(I)=DET(E,U,F,W)/RDET
Y(I)=DET(T,E,V,F)/RDET
N(I)=0
GO TO 3
2 IF(ABS(DET (E,U,F,W)*DET(T,E,V,F))-10.**(-6))0,0,14
N(I)=-1
GO TO 3
14 N(I)=1
3 T=T-1.
U=U-0.2
V=V-0.3
W=W+0.8
E=E-1.5
1 F=F+0.5
WRITE(3,4)
4 FORMAT(1H1,23X,11HEGYENLETEK:,36X,12HMEGOLDAASOK:/)
DO 5 I=1,10
WRITE(3,6)A11(I),A12(I),B1(I)
6 FORMAT(20X,F4.1,4HX = ,F4.1,4HY = ,F4.1)
IF(N(I))8,0,7
WRITE(3,9)X(I),Y(I)
9 FORMAT(60X,3HX =,F9.4,5X,3HY=,F9.4)
GO TO 10
7 WRITE(3,11)
11 FORMAT(60X,15HNINCS MEGOLDAAS)
GO TO 10
8 WRITE(3,12)
12 FORMAT(60X,27HVEEGTELEN SOK MEGOLDAAS VAN)
10 WRITE(3,6)A21(I),A22(I),B2(I)
5 WRITE(3,13)
13 FORMAT(/)
STOP
END

```

#### 4.2 A FUNCTION szubrutin

Van olyan feladat, melyben a szubrutin nem írható meg egy utasítással. Ilyenkor célszerű lehet a **FUNCTION** használata. Ennek a szubrutinnak is van mindig azonosítója, mely után zárójelben állnak a formális, illetőleg aktuális paraméterek. Ezekből legalább egy mindig kell, hogy szerepeljen és lehet közöttük tömbazonosító is. A **FUNCTION** végrehajtása során, annak azonosítója a szubrutinon belül legalább egyszer értéket kell, hogy kapjon és ezt az egy számot adja vissza az őt hívó programnak.

A **FUNCTION**-on belül használt címkék, azonosítók, tömbök függetlenek a hívó programrészbeliéktől. A szubrutinból kiugrani **GOTO** vagy **IF** utasítások segítségével nem lehet. A **RETURN** utasítás visz vissza a hívó programba. Ebből legalább egy van egy szubrutinban. Ha a szubrutinban a program elágazik és a végrehajtás több ágon lehetséges, akkor egy-

egy ág végére helyezhetjük a **RETURN** utasítást. Emiatt ezt az utasítást a szubrutin **logikai végének** is nevezzük. A **FUNCTION** szubrutint egy **END** utasítás zárja le, amit a szubrutin **fizikai végének** nevezünk.

Példaként tekintsük a következő függvényt:

$$y = \begin{cases} 0, & \text{ha } x \leq -10 \\ 0,5x + 5, & \text{ha } -10 < x \leq -2 \\ 0,5(x+2)^2 + 4, & \text{ha } -2 \leq x < 2 \\ -(x-2)^2 + 13, & \text{ha } 2 \leq x \leq \sqrt{13} + 2 = 5,6055 \\ 0, & \text{ha } 5,6005 < x \end{cases}$$

A szubrutin programja így írható:

```

1 | FUNCTION VEGYES(X)
  | IF(X+10.)0,0,1
  | VEGYES=0.
  | RETURN
2 | IF(X+2.)0,0,2
  | VEGYES=0.5*X+5.
  | RETURN
3 | IF(X-2.)0,0,3
  | VEGYES=0.5*(X+2.)*2+4.
  | RETURN
4 | F(X-5.6055)0,0,4
  | VEGYES=-(X-2.)*2+13.
  | RETURN
  | VEGYES=0.
  | RETURN
  | END

```

Mint látjuk, ezt a fajta szubrutint speciális utasítással nyitjuk meg, hasonlóan a **MASTER** szóval kezdődő programrészhez. Szabály, hogy a szubrutin első utasításában álló **FUNCTION** után szóköz, majd az általunk adott azonosító áll és ezt követik zárójelek közt a formális paraméterek. Ennek a szubrutinnak nem kell a hívó programban deklaráció. Hogy példánk teljes legyen, bemutatjuk azt a programot, amely ezt a szubrutint hívja. A program  $x$  értékeit, mint egy sorozat elemeit képezi. Első elem  $a_0 = -11$  és a további elemek képzési szabálya az alábbi összefüggés  $a_n = 0,8 \cdot a_{n-1} + 2$ . A sorozat 6-nál nagyobb elemeit ne képezzük. Egymás melletti oszlopokba nyomtatandók a sorozat elemei és a velük képzett helyettesítési értékek.

A program egy lehetséges formája a következő:

```

2 | MASTER PNYO
  | A=-11.
  | B=VEGYES(A)
  | WRITE(3,1)A,B
1 | FORMAT(/5X,6HELEM: ,F12.8,5X,9HEERTEEK: ,F12.8)
  | A=A*0.8+2.
  | IF(A-6.)2,2,0
  | STOP
  | END

```

A programlapra sorrendben első helyre mindig a **MASTER**-rel kezdődő programrész kerül. Ennek **END** utasítása után tetszőleges sorrendben közvetlenül következnek azok a szubrutinok, amelyek külön programrészként írandók. A hívó program példánkban tartalmazza a **VEGYES (A)** szubrutin behívását a harmadik sorban. Itt a zárójelben most aktuális paraméter áll, mivel a programban előtte az **A** már kapott értéket. A **MASTER**-rel kezdődő programrész után következő **FUNCTION VEGYES (X)** utasításban az **X** formális paraméter. A szubrutinon belül a **VEGYES** azonosító többször is értéket kap, de minden egyes értékadást követően egy **RETURN** utasítás hatására tér vissza a végrehajtás a hívó programrészbe. A szubrutin által számított számértéket **PNYO** nevű programunk harmadik sorában a **B** azonosító veszi fel.

Lényeges tudnivaló, hogy a **FUNCTION** szubrutin tömbazonosítót is átvehet. Ezt látjuk az alábbi vázlatos programban.

```

MASTER VEREEB
DIMENSION A(4),B(5,5)
.
.
.
Q=DETERM(A,B)
.
.
.
STOP
END
FUNCTION DETERM(X,Y)
DIMENSION X(4),Y (5,5)
.
.
.
RETURN
.
.
.
END

```

A MASTER-ban deklarált egydimenziós A tömböt és kétdimenziós B tömböt veszi át a DETERM nevű szubrutin. A tömbök elemei a kipontozott részben kapnak értéket. Vegyük észre, hogy az átadott tömböket a szubrutinon belül újra kell deklarálni.

### Feladatok

**1.** Készítsünk FUNCTION szubrutint, amely tetszőlegesen megadott koordinátájú két pont közötti távolság értékét számítja ki.

**2.** Készítsünk egy olyan FUNCTION szubrutint, amely legfeljebb 30 számadat átlagát számítja ki. (Az adatokat egy 30 elemű tömb, az adatok számát egy egész típusú azonosító tartalmazza.)

**3.** Készítsünk programot, melyben adott öt különböző pont koordinátaival. Számítsuk ki a fenti két szubrutin segítségével az összes lehetséges távolságot és ezek átlagát.

Nyomtassuk ki feliratozva az adott öt pont koordinátáit, ugyancsak feliratozva a pontok közötti lehetséges távolságokat, végül újabb felirat mellett a távolságok átlagát.