æ

Kozics Sándor: A FORTRAN programozási nyelv

1. A FORTRAN nyelv áttekintése

A FORTRAN nyelv(ek)et az IBM gépeire írt autokódok továbbfejlesztett változatainak tekinthetjük. A legelső említésre méltó változat a FORTRAN-II, amelyet az IBM cég 1956-ra dolgozott ki. Nem sokkal később a nagyobb IBM gépeket már magasszintű nyelven, a FORTRAN-IV nyelven írt programok lefordítására alkalmas fordítóprogrammal is ellátták. [3]

1962 májusában lehetővé vált, hogy az ASA (American Standards Association) munkacsoportot hozzon létre, amelynek feladatául tűzte ki, hogy a FORTRAN nyelveket szabványos programozási nyelvként definiálja. A szabványtervezet 1964 októberére készült el, és azóta is ezt tekintik a FORTRAN nyelv hivatalos definíciójának. Ez a nyelv az ASA FORTRAN. A nyelv gépi realizációi általában eltérnek ettől a definíciótól, és az ASA FORTRAN-nál bővebb nyelvet valósítanak meg. A továbbiakban FORTRAN alatt mindig ASA FORTRAN-t értünk, az időközben hagyományossá vált bővítéseket külön jelölni fogjuk.

A FORTRAN nyelven érezhető egy speciális gyakorlati körülmény hatása: az IBM előszeretettel használt adatok és programok bevitelére lyukkártyát. Ennek következtében a FORTRAN nyelvnek is sajátossága az "egy sor — egy utasítás" megfeleltetés. Az utasításokat a sor $7-72$. pozícióira írjuk, a 6. pozíciót a folytatósor jelzésére tartották fenn, míg az $1-5$. pozíciókra a címkék kerülnek. A nyelv érdekes sajátossága, hogy a szóközökkel szemben rendkívül nagyvonalú: bárhol tetszőleges számú szóköz elhelyezhető, és a stringliterálokon kívül pusztán tipográfiai szerepe van. A nyelv neve (FORmula TRANslation) jelzi a felhasználási területet: a FORTRAN-t formulák kódolására, vagyis numerikus módszerek leírására, tudományos számítások elvégzésére szánták. Numerikus típusokban gazdag, viszont típus- és utasításkonstrukciókban elég szegény. A FORTRAN program az ALGOL 60 blokkstruktúrájával szemben önállóan fordítható szegmensekből (modulokból) áll. Érdekes módon ez a később fontossá vált két fogalom egyidőben, de két különböző nyelvben jelent meg. Az 1970-es évek nyelveinél már felismerték, hogy ez nem két, lehetséges fejlődési út, hanem egymást kiegészítő elemek.

2. Lexikális alapelemek

A FORTRAN programok felírására az angol ábécé 26 nagybetűjét, a számjegyeket, valamint az "=", "+", "−", "*", "/", "(", ")", ",", ".", "$", "'", "␣" jeleket használhatjuk.

A FORTRAN azonosítói változók, tömbök, függvények és eljárások jelölésére szolgálnak. Az azonosító betűvel kezdődik, betűket és számjegyeket tartalmazhat, és legfeljebb 6 jelből áll (egyes implementációkban megengedik a tetszőlegesen hosszú azonosítókat is, azzal a megkötéssel, hogy a fordítóprogram csak az első hat karaktert veszi figyelembe).

Megjegyzésnek számítanak azok a sorok, amelyek első pozícióján `C` betű áll. Megjegyzés bárhol elhelyezhető a programban.

A nyelv egész (`INTEGER`), valós (`REAL`), kétszeres pontosságú valós (`DOUBLE PRECISION`) és komplex (`COMPLEX`) számokat ismer (a kétszeres pontosságú számok olyan valós számok, ahol a mantissza ábrázolására kb. kétszer annyi bit áll rendelkezésre, mint az egyszerű valós számoknál).

```
−3              314.D−2
3.14            .7
0.314E1         (3.5,−1.E−5)
```

A FORTRAN a skalár numerikus értékekre a szokásos relációkat vezeti be, az `.EQ.`, `.NE.`, `.GT.`, `.GE.`, `.LT.`, `.LE.` jelöléssel. A logikai értékek neve `.TRUE.` és `.FALSE.`. A stringliterálokat kétféleképpen írhatjuk:

```
'FORTRAN'
7HFORTRAN
```

3. Programszerkezet, szegmensek

A FORTRAN program szegmensekből áll, amelyek közül az egyik a főszegmens, a fennmaradók háromfélék lehetnek : függvények (FUNCTION), eljárások (SUBROUTINE), és lehet a programban egy adatszegmens (BLOCK DATA). A szegmensekben használt azonosítók (kivéve a szegmens és a COMMON adatmezők nevét, ld. 3.) lokálisak. A szegmensek a nyelv fordítási egységei. Mivel a nyelv nem tartalmaz elemet a szegmensek közötti kapcsolatok leírására, fordítás közben nincs mód a kapcsolatok helyességének ellenőrzésére (így pl., ha egy eljáráshívásban az aktuális paraméterek típusa nem felel meg a formális paraméterekének, vagy ha a paraméterek száma eltér a formális paraméterekétől, ezt a hibát a fordítóprogram nem képes felismerni).

```
      SUBROUTINE ORTOG (X,Y,U,V,FI)
      U=X*COS(FI)+Y*SIN(FI)
      V=-X*SIN(FI)+X*COS(FI)
      RETURN
      END

      REAL FUNCTION SH(X)
      REAL Z,W
      INTEGER K
      Z=X**2
      SH=X
      WORD=X
      K=3
1     W=Z/FLOAT(K*(K-1))*W
2     SH=SH+W
3     IF (W/SH.LT.1E-7) RETURN
      K=K+2
      GOTO 1
      END
```

A szegmens elején deklarációs sorok állnak, utánuk következik a végrehajtható rész. A deklarációs rész kiértékelése statikus, a változók leképezése a memóriára fordítási időben történik. Sem az eljárások, sem a függvények nem hívhatják meg önmagukat, és hívási kört sem alkothatnak. A paraméterátadás minden esetben cím szerint történik. Az alprogram végrehajtása mindig a RETURN utasítás végrehajtására fejeződik be.

A függvények aktivizálása a szokásos módon kifejezésben, az eljárásoké

```
      CALL ORTOG (1.0,2.0,A,B,ALFA)
```

formában történik.

A főprogram első sora a MASTER alapszót és a program nevét tartalmazza, ezután — ugyanúgy, mint SUBROUTINE és FUNCTION esetén — deklarációs rész,

majd utasítások következnek (a `MASTER` sor egyes gépeken elmarad, s a főprogram a deklarációs utasításokkal kezdődik). A főprogram nem tartalmazhat `RETURN` utasítást.

```
MASTER FOPROG
    ...
END
```

A programban lehet egy közös adatszegmens is. Ennek egyetlen szerepe, hogy a `COMMON` változók itt kaphatnak kezdőértéket.

```
BLOCK DATA
COMMON /XXXX/ X(10)
DATA X /10*1.0/
END
```

Alkalmazási területének megfelelően a nyelv sok elemi matematikai függvényt tartalmaz. A legfontosabbak: `ABS` (`IABS, DABS`), `SIGN` (`ISIGN, DSIGN`), `REAL` (valós rész képzése), `AIMAG` (képzetes rész leválasztása), `CMPLX` (komplex szám képzése), `EXP` (`DEXP, CEXP`), `ALOG` (`DLOG, CLOG`), `SIN` (`DSIN, CSIN`), `COS` (`DCOS, CCOS`), `SQRT` (`DSQRT, CSQRT`), `MOD` (`AMOD, DMOD`). (A `D` kezdőbetűs függvények paraméterei `DOUBLE PRECISION`, a `C` kezdőbetűseké `COMPLEX`, az `I` kezdőbetűseké és a `MOD`-é `INTEGER`, a többié `REAL` típusú.)

Gyakorlatok

1. Írjuk fel az alábbi függvényt matematikai képlettel!

```
      FUNCTION F(T,B)
      DIMENSION T(15),B(15)
      EL=0.0
      F=0.0
      DO 1 K=1,3
      Q=K
      S=0.0
      V=0.0
      DO 2 I=5*K-4,5*K
      S=S+B(I)*T(I)
2     V=V+T(I)
      H=S/V
      W=H/Q
      EL=EL+W
1     F=F+EL
      RETURN
      END
```

4. Deklarációk

A deklarációs rész a FORTRAN-ban (mivel nincsenek sem konstans-, sem típusdefiníciók, és a szegmensek nem ágyazhatók egymásba) szinte kivétel nélkül változókkal kapcsolatos sorokból áll. Sorrendben

1) a változók típusának deklarálása,
2) a változók dimenziójának deklarálása (tömbök),
3) `COMMON` mezők definíciói,
4) `EQUIVALENCE` utasítások,
5) `DATA` utasítások (kezdőértékadás),
6) `EXTERNAL` utasítások,
7) utasításfüggvények definíciói.

A FORTRAN nyelvben nem kötelező minden változót deklarációs utasításban deklarálni. Ha a végrehajtható utasításokban olyan változónév fordul elő, amit nem deklaráltunk, akkor a fordítóprogram utólag felveszi ezt is a változók közé. Ez az ún. *implicit deklaráció*. Az implicit módon deklarált változó típusa a név első betűjéből állapítható meg: az I,J,K,L,M,N betűkkel kezdődő azonosítókat egész típusú, a többi betűvel kezdődőeket valós típusú változóként kezeli a fordítóprogram.

A nyelv öt különböző típust ismer. Az explicit módon deklarált változó típusát típusdeklarációs utasítással lehet definiálni.

```
INTEGER A,A1,AA
DOUBLE PRECISION W,Q
LOGICAL F1,F3
REAL B1
COMPLEX C
```

A tömb méretét a tömbdeklarációs utasítással lehet megadni.

```
DIMENSION A(8),B(12,3,4),C(5,5)
DIMENSION F1(3),F3(5),AA(2,4)
```

A tömbök indexeinek alsó határa mindig 1. Rövid írásmód is használható: a típusdeklarációkban megadhatók a tömbméretek, így a `DIMENSION` sor elhagyható.

```
INTEGER A(8),A1,AA(2,4)
DOUBLE PRECISION W,Q
LOGICAL F1(3),F3(5)
REAL B1,B(12,3,4)
COMPLEX C(5,5)
```

(Mivel a deklaráció kiértékelése statikus, a tömbdeklarációkban a tömbméretet számokkal kell megadni.)

A FORTRAN tömbök a memóriában mindig oszlopfolytonosan helyezkednek el. Ennek az a következménye, hogy pl. egy mátrix oszlopa átadható egy eljárásnak paraméterként, egy sora viszont nem. Pl. a

```
      SUBROUTINE SUM(T, S)
      DIMENSION T(5)
      S = 0
      DO 1 I=1,5
1     S = S + T(I)
      RETURN
      END
```

eljárás segítségével összegezhető a `C(5,5)` mátrix második oszlopa :

```
      CALL SUM(C(1,2), S)
```

Az ASA FORTRAN-ban indexelni csak ún. standard indexkifejezéssel lehet. Ennek formái a következők (`c` és `k` pozitív egész számok, `I` pedig `INTEGER` változó): `c*I+k, c*I-k, c*I, I+k, I-k, I, k`. Ezt a szabályt azonban szinte minden FORTRAN implementáció felváltja azzal, hogy indexkifejezés tetszőleges `INTEGER` kifejezés lehet.

A szegmensek közötti adatáramlás a paramétereken és a `COMMON` mezőkön keresztül történik.

```
      COMMON /MEZO/ X(10),Z(10,10)
      COMMON /WORK/ Y,A(100)
```

Egy `COMMON` mezőnek más-más szegmensekben a leírása különbözhet, de a hosszuknak meg kell egyezniük.

A közös adatmezők lehetőséget adnak adatabsztrakció kódolására. Példaként figyeljük meg az alábbi vermet :

```
      SUBROUTINE PUSH(X)
      COMMON /VEREM/ V(100), N
      N = N + 1
      V(N) = X
      RETURN
      END

      SUBROUTINE POP(X)
      COMMON /VEREM/ V(100), N
      X = V(N)
      N = N - 1
      RETURN
      END

      BLOCK DATA
      COMMON /VEREM/ V(100), N
      DATA N /0/
      END
```

Ezek a szegmensek egymástól függetlenül fordíthatók, összetartozásukat nem jelölhetjük. Durva végrehajtási hibához vezethet, ha a `VEREM` adatmező felépítése a különböző szegmensekben eltérő, pl. az egyik szegmensben felcseréljük az `N` és `V` változókat.

A FORTRAN egyik sajátos deklarációs utasítása az `EQUIVALENCE`. Az `EQUIVALENCE` utasítással előírható, hogy különböző változók ugyanazon a memóriaterületen legyenek ábrázolva. (A későbbi nyelvekben ennek alkalmazási szabályait egyre szigorították, míg a Pascal nyelvben már átalakult variáns részt tartalmazó rekorddá.)

```
      EQUIVALENCE (B1,B(1,1,1)),(F1(1),F3(3))
      EQUIVALENCE (A(1),AA(1,1))
```

Az `EQUIVALENCE` utasítást elsősorban a program gyorsítására használhatjuk. Pl.

```
      DO 1 I=1,2
      DO 1 J=1,4
1     AA(I,J)=B(1,1,1)
```

helyett ajánlható

```
      DO 1 I=1,8
1     A(I)=B1
```

Az ilyen lehetőségek kihasználásakor fontossá válik annak ismerete, hogyan fordítják a FORTRAN programot, hogyan ábrázolják a memóriában a típusokat és a változókat. A korai nyelvekben a "programozási nyelv" és a "programozási nyelv implementációja" fogalmak teljesen egybefonódtak.

A `DATA` utasítás segítségével definiálható a változók kezdőértéke (a program végrehajtása közben, a szegmensbe való első belépéskor, ez lesz a változó értéke, s ez marad addig, amíg értékadással új értéket nem kap).

```
      DATA AA, B1 /11,21,12,22,13,23,14,24,1.0/,
     *     F1(1) /.TRUE./
      DATA F3 /5*.TRUE./
```

A deklarációs rész végén lokális függvények definiálhatók.

```
      CH(X)=1.0+X**2*(0.5+X**2*(1.0/24.0+X**2/720.0))
```

Az utasításfüggvény törzse erősen korlátozott, itt mindössze egy kifejezés megengedett. Mégis, már ez is nagyon jól használható a programszöveg kifejezőbbé tételére. A példa a `CH` nevű függvényt definiálja, amelynek egy valós paramétere van, a függvény törzse pedig az egyenlőségjel jobboldalán álló kifejezés.

Gyakorlatok

2. Mi lesz az `A` mátrix értéke az alábbi programrészlet végrehajtása után?

```
      REAL A(3,4),B(6),C(6)
      EQUIVALENCE (A(2,2),C(2))
      DATA A/0.1,7*2.1,4*-0.3/,
           B/-3.05,4.27,3*-2.34,6.21/
      DO 1 I=1,6
1     C(I)=B(I)
```

3. Rövidítsük és tegyük világosabbá a következő kódrészletet!

a)
```
C     AZ OLDALAK HOSSZÁNAK SZÁMITÁSA
      AB=SQRT((X2-X1)**2+(Y2-Y1)**2)
      AC=SQRT((X3-X1)**2+(Y3-Y1)**2)
      BC=SQRT((X3-X2)**2+(Y3-Y2)**2)
C     A TERÜLET KISZÁMITÁSA
      S=(AB+BC+AC)/2.0
      TER=SQRT(S*(S-BC)*(S-AC)*(S-AB))
C     A SZÖGEK KISZÁMITÁSA
      ALFA=ATAN2((4.0*TER)/(AC**2+AB**2-BC**2))
      BETA=ATAN2((4.0*TER)/(AB**2+BC**2-AC**2))
      GAMMA=ATAN2((4.0*TER)/(AC**2+BC**2-AB**2))
```

b)
```
      SUBROUTINE LET(S,K,I,J)
      DIMENSION S(20)
      COMMON /T/ A(20),B(20),C(20),D(20)
      GOTO (100,200),I
100   GOTO (110,120),J
      RETURN
110   S(K) = A(K)
      RETURN
120   S(K) = B(K)
      RETURN
200   GOTO (210,220),J
210   S(K) = C(K)
      RETURN
220   S(K) = D(K)
      RETURN
      END
```

5. Utasítások

A végrehajtható utasítások közé soroljuk az értékadó utasításokat, az I/O utasításait, az `IF` és `DO` utasításokat, valamint a `CONTINUE, PAUSE, STOP, RETURN` utasításokat. (A "végrehajtható" szót azért használtuk, mert a FORTRAN-ban szokásos szóhasználat szerint utasításnak, pontosabban nem végrehajtható utasításnak hívják a deklarációs utasításokat, valamint a szegmens kezdetét jelző utasítást és a végén álló `END`-et is.)

5.1. Az értékadás

A FORTRAN értékadás alakja

```
változó = kifejezés
```

Beszélhetünk logikai és aritmetikai értékadásról. Ez utóbbi esetben a bal- és jobboldal típusának nem kell megegyeznie, a jobboldal kiértékelése után automatikus konverzió történik a baloldal típusára. (A következő három táblázatban a zárójelbe tett esetek az ASA FORTRAN-ban nem megengedettek, de a legtöbb FORTRAN reprezentáció megengedi használatukat.)

4. Ábra. Az értékadás konverziós szabályai

=	INTEGER	REAL	DBLE	COMPLEX
INTEGER	normál	IFIX	SNGL+IFIX	
REAL	FLOAT	normál	SNGL	
DBLE	FLOAT+DBLE	DBLE	normál	
COMPLEX				normál

`IFIX` és `SNGL` csonkítást jelentenek, az új formában nem ábrázolható bitek lemaradnak. (Pl., ha `X = 4.0` , az `I = SQRT(X)` értékadás eredményeként `I` értéke valószínűleg `1` lesz.) Az `INTEGER` és a `DOUBLE PRECISION` típusok közötti konverzió két lépésben, a `REAL` típuson keresztül történik.

A FORTRAN aritmetikai kifejezéseiben a különböző típusú mennyiségek keverhetők. A `+`, –, `*` és `/` műveletek esetén az eredmény típusát az alábbi táblázatból olvashatjuk ki:

5. Ábra. Az operátorok konverziós szabályai

+,–,*, /	INTEGER	REAL	DBLE	COMPLEX
INTEGER	INTEGER	(REAL)	(DBLE)	(COMPLEX)
REAL	(REAL)	REAL	DBLE	COMPLEX
DBLE	(DBLE)	DBLE	DBLE	
COMPLEX	(COMPLEX)	COMPLEX		COMPLEX

A `**` (hatványozás) műveletére az alábbi táblázat érvényes (a típusokra vonatkozó megkötéseken felül megszorítás még, hogy a nulla csak pozitív egész kitevőre emelhető, negatív mennyiség pedig sosem emelhető valós vagy duplapontos kitevőre):

6. Ábra. A hatványozás konverziós szabályai

**	INTEGER	REAL	DBLE	COMPLEX
INTEGER	INTEGER	(REAL)	(DBLE)	
REAL	REAL	REAL	DBLE	
DBLE	DBLE	DBLE	DBLE	
COMPLEX	COMPLEX	(COMPLEX)		(COMPLEX)

A kifejezések kiértékelését a precedencia-szabályok határozzák meg. Elsődleges művelet a hatványozás, másodlagos a szorzás és az osztás, harmadlagos az összeadás és a kivonás. Az azonos osztályba tartozó műveletek kiértékelési sorrendjét a nyelv nem írja elő, így pl. az `I*J/5*K` kifejezés értéke nem egyértelműen definiált. Az egyes gépi reprezentációk a balról-jobbra szabályt alkalmazzák. A kiértékelés sorrendjét zárójelezéssel szabályozhatjuk.

Gyakorlatok

4. Állapítsuk meg, ekvivalensek-e az alábbi utasítások!

```
a)   M=I*J/K        ill.   M=J/K*I
b)   Y=Y**(3/2)     ill.   Y=Y**(3./2.)
c)   X=X**2.4*A     ill.   X=X**(2.4*A)
```

5. Helyesek-e szintaktikusan az alábbi utasítások?

```
a)   A=1.D2+E2
b)   DO 1 I=1.2
c)   A=SQRT(SIN(2*X(5,7))*Y(3,IFIX(SIN(X**3)))
```

6. Az alábbi programrészletek közül melyiket részesítenénk előnyben? Indokoljuk a választást!

```
      F1=X1-X2*X2
      F2=1.0-X2
      FX=F1*F1+F2*F2
```

ill.

```
      FX=(X1-X2**2)**2+(1.0-X2)**2
```

5.2. Vezérlésátadás

Az ALGOL-szerű nyelvekkel ellentétben a FORTRAN-ban a `GOTO` utasításnak fontos szerepe van. Itt ugyanis nincsenek összetett utasítások (kivéve az egyetlen `DO` utasítást), így már egy elágazás megszervezéséhez is szükség van feltétlen vezérlésátadásra. A `GOTO` argumentuma egy címke, vagy egy címkét értékül kapott `INTEGER` változó. Címkeként (legfeljebb ötjegyű) természetes számok használhatók.

```
      IF (.NOT. L) GOTO 20
         ...
20    CONTINUE
```

A `GOTO` egy másik alakja, a kiszámított `GOTO`, többirányú elágazás kódolására ad lehetőséget.

```
      GOTO (10,20,30),K
10    ...
      GOTO 40
20    ...
      GOTO 40
30    ...
40    CONTINUE
```

Ez a programrész csak akkor működik helyesen, ha `1` $\leq$ `K` $\leq$ `3`.

5.3. Az `IF` utasítás

Az `IF` utasításnak két formája van. Az ún. aritmetikai `IF` egy kifejezés előjele szerinti elágazásra ad lehetőséget.

```
      IF (I-J) 9,10,11
```

A program a `9`-es címkénél folytatódik, ha `I-J<0`, a `10`-esnél, ha `I-J=0`, és a `11`-es címkénél, ha `I-J>0`.

Az `IF` másik formája az ún. logikai `IF` utasítás.

```
      IF (A.GT.B) B=A
```

Itt a zárójelben logikai kifejezésnak kell állnia. Ha ennek az értéke igaz, végrehajtódik a csukózárójel után álló utasítás. Nem írható azonban a csukózárójel

után sem IF, sem DO utasítás. Példa az IF utasítás használatára :

```
      W=2.0*A
      IF (ABS(A).LT.1E-8) GOTO 21
      D=B**2-4.0*A*C
      IF (D) 20,18,16
16    D=SQRT(D)
      X1=(-B+D)/W
      X2=(-B-D)/W
      GOTO 22
18    X1=-B/W
      X2=X1
      GOTO 22
      ...
```

A logikai kifejezések írásának szabályai az aritmetikai kifejezésekéhez hasonlóak. A műveleti jelek : .NOT., .OR., .AND. . A kifejezések kiértékelésének módja:

1) aritmetikai műveletek elvégzése
2) a relációk kiértékelése
3) a logikai műveletek kiértékelése, sorrendben: .NOT., .AND., .OR.

Gyakorlatok

7. Milyen szempontból kifogásolható az alábbi programkód? Javítsuk ki!

a)
```
      IF (X1.GE.T(I)) GOTO 2
      IF (T(I).LT.T2) IS=IS+1
2     CONTINUE
```

b)
```
      GOTO (65,70),KIIR
65    WRITE (6,105) X
70    CONTINUE
```

c)
```
      IF (T(I).GT.E) GOTO 50
      GOTO 20
50    E=T(I)
      IND=I
20    CONTINUE
```

8. Az IF (I-J) 10,20,10 utasítás az I és J változók összehasonlításakor túlcsordulás léphet fel. Megfelel-e az alábbi részlet a fenti elágazás helyett a túlcsordulás elkerülésére?

```
      IF (I) 1,2,2
1     IF (J) 3,10,10
2     IF (J) 10,3,3
3     IF (I-J) 10,20,10
```

9. Melyek helyesek szintaktikusan az alábbi IF utasítások közül?

a)
```
      IF (A*-B) 2,01,101
```

b) `IF (A.GE.B) THEN 30 ELSE 100`
c) `IF (X.OR.Y.AND..NOT.Z) 9,18,1`
d) `IF (D*A.LT.D-C) 2,K,1`
e) `IF (X+B*Y.GT.0) IF(X) 3,4,5`
f) `IF (X.AND..NOT.Y) GOTO 100`
g) `IF (A+B) A=B`
h) `IF (X.OR.Y.OR.TRUE.) Z=X.OR.Y`

10. Hasonlítsuk össze az alábbi programrészleteket! Milyen szempontból, melyiket részesítenénk előnyben?

```
        IF (X.LT.Y) GOTO 30
        IF (Y.LT.Z) GOTO 50
        A=Z
        GOTO 70
30      IF (X.LT.Z) GOTO 60
        A=Z
        GOTO 70
50      A=Y
        GOTO 70
60      A=X
70      CONTINUE
```

ill.

```
        A=X
        IF (Y.LT.A) A=Y
        IF (Z.LT.A) A=Z
```

5.4. A `DO` utasítás

A FORTRAN ciklusok dolgában igen szegény. Mindössze egy ilyen utasítást tartalmaz, az is igen korlátozottan használható.

```
        DO c I=M1, M2, M3
```

Itt `c` egy olyan címke, ami egy későbbi sor elején szerepel. Az `c`-ig tartó utasítások képezik a ciklusmagot. A címke után álló `I` egy `INTEGER` skalár változót szimbolizál, ez a ciklusváltozó. `M1, M2, M3` pozitív egész számok vagy változók (egyes gépi reprezentációkban kifejezések is lehetnek). `M1` a ciklusváltozó kezdőértéke, `M2` a végértéke, `M3` a növekmény (ez elmaradhat, ekkor a növekmény `1`). A `DO` ciklussal ekvivalens a következő program:

```
        I=M1
c1      ...
           ciklusmag
        ...
        I=I+M3
        IF (I.LE.M2) GOTO c1
c       ...
```

Egymásbaágyazott ciklusok esetén, ha a két ciklusmag vége egybeesik, a két `DO` utasításban ugyanazt a címkét használhatjuk.

```
      M=N-1
      DO 20 K=1,M
      M1=K+1
      DO 20 K1=M1,N
      B=A(K,K1)
      A(K,K1)=A(K1,K)
20    A(K1,K)=B
```

A CONTINUE utasítás a FORTRAN üres utasítása. Egyetlen szerepe, hogy címkét viselhet. A ciklusmag utolsó utasításaként gyakran használunk CONTINUE utasítást.

```
      Z=A(1)
      DO 1 I=1,N
      IF (Z.LE.A(I)) GOTO 1
      Z=A(I)
1     CONTINUE
```

Gyakorlatok

11. Állapítsuk meg, ekvivalensek-e a következő programrészletek!

a)
```
      DO 1 I=1,5
      DO 1 J=1.5
      A(I,J)=(I/J)*(J/I)
1     CONTINUE
```
ill.
```
      DO 1 I=1,5
      DO 1 J=1,5
      IF (I.EQ.J) A(I,J)=1
      IF (I.NE.J) A(I,J)=0
1     CONTINUE
```
ill.
```
      DO 1 I=1,5
      DO 1 J=1,5
      A(I,J) = 1
      IF (I.NE.J) A(I,J)=0
1     CONTINUE
```

b)
```
      DO 1 I=1,N,2
      J=(I+1)/2
1     B(J)=FLOAT(L(I)+L(I+1)/FLOAT(10))
```
ill.
```
      DO 1 I=1,N,2
      J=(I+1)/2
1     B(J)=L(I)+L(I+1)/10.
```

12. Az alábbi programrészletek közül milyen szempontból, melyiket részesítenénk előnyben? Indokoljuk!

```
      F=A(1)
      DO 25 I=2,10
```

```
              IF (A(I).GT.F) GOTO 30
              GOTO 25
    30        F=A(I)
    25        CONTINUE
```

ill.

```
              F=A(1)
              DO 25 I=2,10
                IF (A(I).GT.F) F=A(I)
    25        CONTINUE
```

ill.

```
              F = A(1)
              DO 25 i=2,10
                IF(A(I).LE.F) GOTO 25
                F = A(I)
    25        CONTINUE
```

13. Állapítsuk meg, megfelelő-e az alábbi programrészlet egy mátrix elemeinek /a)/ ill. egy mátrix pozitív értékkel kezdődő sorai elemeinek /b)/ összeadására!

a)
```
              S=0.0
              DO 1 I=1,5
              DO 1 J=1,5
                S=S+A(I,J)
    1         CONTINUE
```

b)
```
              S=0.0
              DO 1 I=1,5
                IF (A(I,1)) 1,1,2
    2           DO 1 J=1,5
                  S=S+A(I,J)
    1         CONTINUE
```

14. Mi lesz a `K` változó értéke a programrészlet végrehajtása után?

```
              K=0
              DO 10 I = 2 , 7 , 2
              DO 10 J = 1 , I - 2
    10        K = K + 1
```

15. Írjuk gazdaságosabban!

a)
```
              DO 1 I=1,100
              Q(I)=0.0
              DO 1 J=1,100
    1         Q(I)=Q(I)+A*B*X(I,J)
```

b)
```
              F=X**5-64*X**4+3*X**2-17
```

c)
```
              DIMENSION X(10,20,8)
              DO 1 I=1,10
              DO 1 J=1,20
              DO 1 K=1,8
    1         1 X(I,J,K)=0.0
```

16. Írjunk függvényt az *Arctg x* kiszámítására az alábbi végtelen sorból! (A sorból annyi tagot kell figyelembe venni, hogy az első elhagyott tag kisebb legyen az addig kapott összeg `1.E-6` -szorosánál.)

$$Arctg\ x = x - \frac{x^3}{3} + \frac{x^7}{5} - \frac{x^7}{7} + ...$$

17. Írjunk az alábbi ALGOL 60 utasítássorozattal azonos hatású FORTRAN programrészletet!

a)
```
s := 0.0;
for x := -2.0, -1.2, 1.7, 2.4, 4.9 do
  s := s + (x ↑ 3 - a ↑ 3) / 3.0;
```

b)
```
integer procedure F(n); integer n; value n;
begin
  if n = 1 then F := 1 else F := n * F(n-1)
end F;
```

5.5. PAUSE, STOP

A program megállítására kétféle utasítás is használható. A

`PAUSE` vagy `PAUSE n`

utasítás hatására a program megáll, az `n` számot kijelzi, és ezután továbbindítható. A

`STOP` vagy `STOP n`

végleges megállító utasítások, végrehajtásuk eredményeképpen a program törlődik a memóriából.

6. I/O utasítások

Ugyanúgy, ahogy a nyelv is sororientált, a FORTRAN I/O is az. Egy olvasás mindig új sorból kezdi az olvasást, egy kiírás mindig új sort hoz létre. (Itt a nyomtatónál találunk kivételt.) A FORTRAN I/O viszonylag terjedelmes, mi csak a kártyaolvasó (standard input) és a sornyomtató kezelésével kívánunk foglalkozni. Két átviteli utasítást tárgyalunk:

```
READ (u,f) L
WRITE (u,f) L
```

`u` a file logikai száma, `f` egy formátumleírás (`FORMAT`) címkéje, vagy egy `FORMAT` utasítást (zárójeltől zárójelig) tartalmazó tömb neve (!), `L` pedig az ún. I/O-lista. Az I/O-lista lehet egyszerűen változók felsorolása, de valamely eleme lehet ún. implicit ciklus is:

```
A,B,Q(4,3), R(I,J)
(A(I), I=1,N)
(A,B(I),C(I),I=1,N)
((A(I,J),I=1,M),J=1,N)
```

Az I/O-listában egy tömb neve indexek nélkül is állhat, ilyenkor az a teljes tömb átvitelét jelenti. (A nyelvben létezik a `READ/WRITE`-nak `FORMAT` nélküli, bináris változata is, de ezzel nem kívánunk foglalkozni.)

A `FORMAT` utasítás írja le, hogy az átvitelhez kijelölt változók értékei az adathordozón milyen formában találhatók. Így tehát az I/O-listán megadott minden skalár értékhez a `FORMAT` utasításban tartozik egy konverziós formátumleíró elem.

```
f     FORMAT ( ...  )
```

A `FORMAT`-listán elhatárolójelekkel elválasztva, kétféle formátumleíró elem (formátumspecifikáció) állhat: konverziós specifikáció és szerkesztési specifikáció.

6.1. Az `I` specifikáció

Az `INTEGER` értékek átviteléhez az `I` konverziós specifikációt alkalmazzuk:

`I`w

ahol w a mezőszélességet jelöli. Legyen `K = 12397, L =` $-$`3540, M = 52888`, ekkor a

```
      WRITE (6,15) K,L,M
15    FORMAT (I6,I6,I6)
```

kiírás eredménye (nyomatáskor az első jel a nyomtató vezérlésére szolgál; ld. az 5.9. pontot).

```
12397␣−3540␣52888
```

Az előbbi formátumot egyszerűsíthetjük ismétlési tényező alkalmazásával:

```
15    FORMAT (3I6)
```

Olvasáskor a kijelölt mezőben a szám jobboldalán álló szóközök nulla-értékűek(!).

Gyakorlatok

18. Állapítsuk meg, ekvivalensek-e a következő programrészletek!

```
      DIMENSION A(50,50)
        ...
      READ (5,1) A
```

ill.

```
      DIMENSION A(50,50)
        ...
      READ(5,1)((A(I,J),J=1,50),I=1,50)
```

19. Mi kifogásolható a következő kódrészletben? Indokoljuk!

```
      DIMENSION A(10,10)
      N=5
      READ(5,1) ((A(I,J),I=1,N),J=1,N)
      WRITE(6,1) A
```

20. Mit nyomtat ki a következő programrészlet?

```
      INTEGER A(2,4), B(4)
      EQUIVALENCE (A(1,1),B(1))
      READ(5,10) A
10    FORMAT(1X, 8I1)
      DO 1 I = 1,4
1     B(I) = A(1,I)
      WRITE(6,10) A
```

Az adatok : `210987654`

6.2. Az `F` specifikáció

Az `F` specifikációt fixpontos alakú számok beolvasásához, ill. kiírásához használjuk.

`F` *w.d*

Itt *w* ismét a mezőszélesség, *d* pedig ebből a törtrész hossza. A szám a mezőben

most is jobbra tömörítve helyezkedik el. Legyen `X = −0.3638440557, Y = 3892.184, Z = −20.099633`. Ekkor a

```
      WRITE (6,5) X, Y, Z
5     FORMAT (F11.8, F8.2, F8.4)
```

kiírás eredménye

```
−0.36384406␣3892.18−20.0996
```

Olvasáskor, ha a mezőben van tizedespont, ez felülbírálja a d értéket, ha nincs, akkor a jobboldali d db jegy számít törtrésznek. A mezőben szabályos számnak kell állnia, különben a program végrehajtási hibával leáll. Legyen a rekord tartalma

```
␣␣−12.098␣␣␣−3.25E−31248␣␣␣␣␣␣␣␣
```

Ekkor `X = −12.098, Y = −0.00325` és `Z = 1248000.0` lesz a változók értéke a

```
      READ (5,3) X, Y, Z
3     FORMAT (F12.4, F8.2, F10.3)
```

beolvasás végrehajtása után.

Gyakorlatok

21. Mely sorok tartalmaznak szintaktikus hibát a következő programban, és mondjuk meg azt is, mi a hiba!

```
      DIMENSION S(5)
      A(K)=B+X**K
      READ(5,1) B,X
1     FORMAT (F5.2)
      DO 2 I=2,4
      C(I)=A(I−2)
      S(I)=F(I−1)
2     CONTINUE
      END
```

6.3. Az `E` és a `D` specifikáció

Az `E` konverziós specifikáció valós (lebegőpontos alakú) számok leírására szolgál.

`E` $w.d$

Kiíráskor a következő jelek jelennek meg: ± `0.yy ... y E` ± `yy` ahol w a mezőszélesség, d a kiírt törtjegyek száma, és mint leolvasható $d + 7 \leq w$. Input esetén az `E` specifikáció teljesen megegyezik az `F`-fel. A kétszeres pontosságú

mennyiségek átvitelekor a D specifikációt használjuk. Ennek formája teljesen megegyezik az E specifikációval.

Gyakorlatok

22. Lyukkártyán adott a következő jelsorozat.

```
472372E+086.2371517␣␣␣11734␣␣␣␣␣␣
```

Mi lesz a

```
READ(5,3) N, X, G, M, Y
```

olvasás eredménye, ha a FORMAT utasítás a következő:

a)
```
FORMAT (I5, E5.2, F5.2, I4, F8.3)
```
b)
```
FORMAT (I4, E6.0, 3X, F4.3, I3, F7.5)
```
c)
```
FORMAT (I6, E5.3, F8.6, I3, F7.3)
```

6.4. Az L specifikáció

Az L konverziós specifikáció a logikai mennyiségek átvitelére szolgál.

```
L w
```

Kiíráskor *w–1* darab szóköz és egy T vagy F betű jelenik meg, aszerint, hogy az I/O-lista megfelelő elemének értéke TRUE vagy FALSE. Olvasáskor, ha a bevezető szóközök után a rekordban T betű következik, akkor a beolvasott érték TRUE, ha F, akkor FALSE lesz.

6.5. Az A specifikáció

Bár a nyelv nem tartalmaz sem karakter, sem string típust, mód van szövegek beolvasására és kiírására úgy, hogy közben a szöveget valamilyen numerikus változóban tároljuk. Az A konverziós specifikációt használhatjuk stringek átvitelekor.

```
A w
```

A konverziós specifikációk között ez a leginkább gépfüggő, szabályai viszonylag bonyolultak, ezért itt csak azzal az esettel foglalkozunk, amikor egy skalár (vagy tömbelem) egy karaktert tartalmaz (*w=1*). Ha az input rekord elején a HOVANSCSINA szöveg áll, ezt a következőképpen olvashatjuk be és írhatjuk ki:

```
DIMENSION IT(11)
  ...
READ (5,1) IT
```

```
1     FORMAT (11A1)
      WRITE (6,2) IT
2     FORMAT (1X,11A1)
```

6.6. A H specifikáció

Szövegek átvitelére szolgál még az előzőeken kívül a H specifikáció is. (Ez nem konverziós specifikáció, vagyis nem tartozik hozzá elem az I/O-listában!)

wH$x_1x_2x_3...x_w$

Kiíráskor a H betű után álló w darab jel változtatás nélkül kerül az outputra. A H specifikációt fejlécek, feliratok kinyomtatására használjuk.

```
      DO 1 I=1,10
1     WRITE (6,8) (VAL(I,J),J=1,3)
8     FORMAT (3H $,F9.2,5H SFR,F9.2,4H DM,F9.2)
```

Beolvasáskor, ha a FORMAT utasításban H specifikáció szerepel, ez beolvasást jelent magába a FORMAT utasításba, átírva az illető szövegliterált (!).

Gyakorlatok

23. Egy kártyán a következő jelsorozat van :

HETFO␣␣␣␣␣␣␣7␣␣␣FOLYOSO␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣

Mi jelenik meg a sornyomtatón a következő utasítások nyomán?

```
      READ (5,1) I
      WRITE (6,1) I
1     FORMAT (5HALMAK,5X,I2,3X,2HJA)
```

6.7. Az X specifikáció

Helyközök kialakíthatók az outputon az A vagy a H specifikáció alkalmazásával is, ez azonban kissé körülményes lenne. A

w X

szerkesztési specifikáció az outputon w darab helyközt képez, az input rekordban pedig a következő w darab pozíciót figyelmen kívül hagyja.

```
      WRITE (6,4)
      WRITE (6,5)
      DO 2 I=1,1000
      X=0.001*FLOAT(I-1)
      F=10.*X/(1.-X**2)
```

```
2     WRITE (6,3) X,F
3     FORMAT (48X,F6.3,5X,E13.6)
4     FORMAT (51X,18HFUGGVENYTABELLAZAS)
5     FORMAT (51X,1HX,12X,4HF(X))
      STOP
      END
```

6.8. A vessző és a törtvonal, mint elhatároló jelek

Formátumleírásban az elemek elválasztására eddig mindig a vesszőt használtuk. Használható még a törtvonal (/) is, ennek azonban még más funkciója is van: áttér a következő rekord első pozíciójára.

```
      WRITE (6,4)
4     FORMAT (51X,18HFUGGVENYTABELLAZAS////
     *         51X,1HX,12X,4HF(X)//)
```

6.9. Vezérlő karakterek a sornyomtatón

Sornyomtatóra való kiíráskor a sor első karaktere nem kerül nyomtatásra, ez a nyomtatás vezérlésére szolgál. A vezérlő karakterek jelentése a következő :

'␣' kiírás a következő sor első pozíciójától,
'+' kiírás soremelés nélkül a sor első pozíciójától (ráír az előző sorra),
'0' kiírás egy sor kihagyásával,
'1' kiírás a következő lap első sorába.

Az előző pontban használt példát most tehát már így is írhatjuk:

```
      WRITE (6,4)
4     FORMAT (1H1///51X,18HFUGGVENYTABELLAZAS////
     *         51X,1HX,12X,4HF(X)//)
```

Gyakorlatok

24. Határozzuk meg, mi jelenik meg a sornyomtatón a

```
WRITE (6,1) X, Y, Z
```

kiírás hatására, ha `X = 4.38, Y = −6.47, Z = 9.07` és a `FORMAT` a következő:

```
a)    FORMAT (/5X,F5.2/5X,F5.2/5X,F5.2)
b)    FORMAT (3F7.2)
c)    FORMAT (3F5.2)
d)    FORMAT (3X,3E14.6)
e)    FORMAT (F4.1,1X,F7.2/2X,E11.2)
f)    FORMAT (F6.3,F5.2,E9.3)
g)    FORMAT (F6.2,F5.2,F7.3)
```

25. Helyes-e szemantikusan a következő nyomtatás?

a)
```
      A=-67.32
      WRITE (6,1) A
1     FORMAT (E8.3)
```

b)
```
      DO 1 I=10,20
1     WRITE (6,50) I,A(I)
50    FORMAT (I2,4X,F10.5)
```

c)
```
      WRITE (6,50) I
50    FORMAT (/I2)
```

6.10. A FORMAT- és az I/O-lista kölcsönhatása

Ebben a pontban arról lesz szó, hogyan történik a FORMAT-lista és az I/O-lista elemeinek egymáshoz rendelése.

Az I/O-lista és a FORMAT-lista értelmezése balról-jobbra halad, és az I/O-lista minden eleméhez párosul a FORMAT-lista egy konverziós eleme. A FORMAT-lista eközben átlépett szerkesztési specifikációi (H, X, /) természetesen végrehajtásra kerülnek.

Nemcsak az I/O-listának lehetnek "többszörös" elemei (az index nélkül álló tömbnevek), hanem a FORMAT-listának is: az ismétlési tényezővel ellátott elemek. Nemcsak egy specifikációt láthatunk el ismétlési tényezővel, hanem specifikációk zárójelbe tett csoportját is.

```
FORMAT (1H1,5X,8HTABLAZAT///1X,5(I10,F8.3))
```

Mi történik abban az esetben, ha a két lista párhuzamos feldolgozása közben valamelyik előbb véget ér, mint a másik? A szabályokat három pontban foglaljuk össze:

1) Ha az I/O-lista üres, a FORMAT-lista feldolgozásra kerül az első konverziós specifikációig.
2) Ha az I/O-lista nem üres és előbb véget ér, akkor a FORMAT-lista fel nem dolgozott elemei figyelmen kívül maradnak.
3) Ha a FORMAT-lista ér előbb véget, akkor új rekord kezdődik az átvitelben, és

3a) ha van zárójelezett rész (ismétlési csoport) a FORMAT-listán, akkor — a FORMAT-ot lezáró zárójelet nem számítva — az utolsó csukózárójelhez tartozó nyitózárójel ismétlési tényezőjétől folytatjuk a FORMAT-listát,

3b) ha nincs ismétlési csoport, akkor a FORMAT-listát az elejétől folytatjuk.

A FORMAT-lista statikussága sok nehézséget okoz a programozónak. Próbáljuk pl. megoldani a következőt: az X(10) vektor első N elemét nyomtassuk egy sorba egymás mellé, majd a számok után közvetlenül tegyünk három pontot!

Gyakorlatok

26. Mi kerül a nyomtatóra a következő kiírás eredményeként?

a)
```
      WRITE (6,8) (X1(I),Y1(I),X2(I),Y2(I),I=1,20)
8     FORMAT (1H1//51X,19H A VEKTOROK ELEMEI:///
     * 36X,2HX1,13X,2HY1,13X,2HX2,13X,2HY2///
     * 1(33X,2(F7.4,5X,E13.6,5X)))
```

b)
```
      WRITE (6,8) ((B(I,J),J=1,30),I=1,30)
8     FORMAT (6(5(10X,E15.8,5E17.8,/),3(/)),1H1)
```

c)
```
      WRITE (6,8) (B(I),I=1,60)
8     FORMAT (10(16X,E13.6,5E15.6/))
```

27. Hasonlítsuk össze az alábbi két programrészletet! Melyiket részesítenénk előnyben?

```
10    READ (5,11) X
11    FORMAT (F10.2)
      IF (X.GE.0.0) GOTO 20
          WRITE (6,13) X
13        FORMAT (' SQRT(',E12.5,') NEM LETEZIK')
          GOTO 10
20    IF (X.GT.0.0) GOTO 30
          B=0.0
          GOTO 50
30    B=1.0
40    A=B
          B=(X/A+A)/2.0
          IF (ABS((X/B)/B-1.0).GE.1.0E-5) GOTO 40
50    WRITE (6,51) X,B
51    FORMAT (' SQRT(',E12.5,') =',E12.5)
```

ill.

```
100   READ (5,110) X
110   FORMAT (F10.2)
C
      IF (X.LT.0.0) WRITE (6,120) X
120   FORMAT (1X,'SQRT(',E12.4,') NEM LETEZIK')
C
      IF (X.EQ.0.0) WRITE (6,130) X,X
130   FORMAT (1X,'SQRT(',E12.4,') = ',E12.4)
C
      IF (X.LE.0.0) GOTO 100
      B=X/2.0
200   IF (ABS(X/B-B).LT.1.0E-5*B) GOTO 300
      B=(X/B+B)/2.0
      GOTO 200
300   WRITE (6,130) X,B
```

28. Keressük meg a programozási hibákat a következő függvényben!

```
      DOUBLE PRECISION FUNCTION SINUS(X)
```

```
      DOUBLE PRECISION TAG,SUM
      REAL X
      TAG=X
      DO 20 I=3,10,2
          TAG=TAG*X**2/(I*(I-1))
          IF (TAG.LT.1.0D-8) GOTO 30
20        SUM=SUM+(-1**(I/2))*TAG
30    SINUS=SUM
      END
```

29. Alkossunk véleményt, kifogástalan-e az alábbi programkód!

```
      FUNCTION RITKA(I,J)
      COMMON /RIT/ N,NSOR(500),NOSZL(500),ERTEK(500)
      DO 10 K=1,N
          IF (NSOR(K).NE.I.OR.NOSZL(K).NE.J) GOTO 10
          RITKA=ERTEK(K)
          GOTO 99
10    CONTINUE
      RITKA=0.0
99    RETURN
      END
```

30. Alkossunk véleményt, kifogástalan-e az alábbi függvény a tömb legkisebb elemének megkeresésére!

```
      FUNCTION P(A,N)
      DIMENSION A(N)
      P=A(1)
      DO 1 K=2,N
          IF (A(K)-P) 2,1,1
2     P=A(K)
1     CONTINUE
      RETURN
      END
```

31. Az alábbi S szubrutin feladata a mátrixelemek összegének előállítása. Helyesen használja-e a főprogram S-et?

```
      DIMENSION A(100,10)
          ...
      N=10
      READ (5,1) ((A(I,J),J=1,10),I=1,N)
1     FORMAT (8F10.2)
      CALL S(A,N,10,X)
          ...
      END
      SUBROUTINE S(A,M,N,X)
      DIMENSION A(M,N)
      X=0.0
      DO 1 I=1,M
      DO 1 I=1,N
1     X=X+A(I,J)
      RETURN
```

```
END
```

32. Mi lesz a

```
CALL A(1,2,3)
X=1.0
X=(X+1.0)*2+3
```

programrészlet végrehajtása után `X` értéke, ha az `A` szubrutin a következő:

```
SUBROUTINE A(I,J,K)
I=I+1
J=K+2
RETURN
END
```

33. Írjuk meg FORTRAN nyelven az alábbi ALGOL 60 programrészletnek megfelelő utasítássorozatot!

a)
```
if A>B then S:=1;
else if A=B ∧ C>D then S:=2;
else if A=B then S:=3;
else if C>D then S:=4;
else if C=D then S:=5;
else S:=6;
```

b)
```
if K−(K÷2)*2 = 0 then begin
    SUM := SUM+X;
    PAROS := PAROS+1;
end
```

c)
```
integer array N[1:1000];
integer L, M;
L:=1000; M:=sqrt(L);
for I:=2 step 1 until L do
begin
    if N[I]=0 then goto UGRAS;
    outinteger(I,5);
    if I≤M then
        for K:=1 step 1 until L÷ I do N[K*I]:=0;
    UGRAS:;
end;
```

d)
```
if X≥Y then
    begin
        if Y≥Z then A:=Z else A:=Y
    end else
if X≥Z then A:=Z else A:=X;
```

e)
```
if A>B then begin
    X:=A; goto V1;
end;
X:=B;
V1:  if X>C then goto V2;
X:=C;
V2:
```

f)
```
        if A=0 then goto T1;
        goto V1;
        T1:  if B=0 then goto T2;
        goto V1;
        T2:  if C=0 then goto V2;
        V1:  A:=A+1;
        V2:  B:=B+1;
```

34. Írjuk meg ALGOL 60 nyelven az alábbi programrészletnek megfelelő utasítássorozatot!

a)
```
        ISUM=0
        DO 3 I=1,5
        IF (HELYES(I).EQ.FELEL(I)) GOTO 4
        IELL(I)=0
        GOTO 3
4       IELL(I)=1
        ISUM=ISUM+1
3       CONTINUE
```

b)
```
        IF (KA-KB) 5,5,4
4       KR=KA
        KA=KB
        KB=KR
5       IF (KA) 6,7,6
6       KR=KB-KB/(KA*KA)
        KB=KA
        KA=KR
        GOTO 5
7       CONTINUE
```

c)
```
        SUBROUTINE RENDEZ(V,M)
        DIMENSION V(M)
        IF (M.LT.2) GOTO 251
        DO 250 J=1,M-1
            T=V(J+1)
            DO 235 K=1,J
                I=J+1-K
                IF (T.GE.V(I)) GOTO 245
                V(I+1)=V(I)
235         CONTINUE
            I=0
245         V(I+1)=T
250     CONTINUE
251     CONTINUE
        RETURN
        END
```

d)
```
        DIMENSION A(100)
        J=1
1       IF (J.EQ.N) GOTO 4
        IF (A(J).LE.A(J+1)) GOTO 3
        X=A(J)
        A(J)=A(J+1)
        A(J+1)=X
        K=J
```

```
2       K=K-1
        IF (K.EQ.0.OR.A(K).LE.A(K+1)) GOTO 3
        X=A(K)
        A(K)=A(K+1)
        A(K+1)=X
        GOTO 2
3       J=J+1
        GOTO 1
4       CONTINUE
```

7. Példaprogramok

1) Pont és poligon helyzete

Legyenek x(i), y(i) (i=1,...,n) egy konvex sokszög csúcspontjainak koordinátái (ciklikusan, sorrendben megadva, és legyen x(1) = x(n), y(1) = y(n)). Legyen x0, y0 egy olyan pont, ami nem fekszik a sokszög határán. A következő FORTRAN függvény ekkor meghatározza, hogy a pont a sokszög belsejébe esik-e. [2]

```
      LOGICAL FUNCTION POINTPOL(N, X, Y, X0, Y0)
C     PONT ES KONVEX SOKSZOG HELYZETENEK MEGHATAROZASA.
C     A FUGGVENY ERTEKE AKKOR IGAZ, HA A PONT A SOKSZOG
C     BELSEJEBE ESIK.
      INTEGER N
      REAL X0, Y0, X(N), Y(N)
      INTEGER I
      LOGICAL B

      B = .TRUE.
      DO 10 I = 1, N-1
      IF (.NOT. ((Y0 .GT. Y(I) .AND. Y0 .LE. Y(I+1)) .OR.
     *     (Y0 .LE. Y(I) .AND. Y0 .GT. Y(I+1))) GOTO 10
      IF (X0-X(I) .LT.
     *     (Y0-Y(I)) * (X(I+1)-X(I)) / (Y(I+1)-Y(I)))
     *     B = .NOT.B
10    CONTINUE
      POINTPOL = .NOT. B
      RETURN
      END
```

2) Polinomok osztása

Legyenek $A(x) = a_1x^m + a_2x^{m-1} + ... + a_{m+1}$ és $B(x) = b_1x^n + b_2x^{n-1} + ... + b_{n+1}$ m- ill. n-edfokú polinomok ($m \geq n$). Az $A(x)$ polinomot elosztva a $B(x)$ polinommal hányadosul a $C(x)$, maradékul az $R(x)$ polinomot kapjuk, ahol $C(x)$ fokszáma legfeljebb $m - n$, $R(x)$-é legfeljebb $n - 1$. Ekkor az együtthatókra érvényesek az alábbi egyenletek (r_1,...,r_{n-1} helyett c_{m-n+2},...,c_{m+1}-et írunk):

$$c_1 = a_1,$$

$$c_i = (a_i - \sum_{j=1}^{i-1} c_j b_{i-j+1})/b_1 \quad (2 \leq i \leq m - n + 1 - re),$$

$$c_{m-n+1} = (a_{m-n+1} - \sum_{j=1}^{m-n} c_j b_{m-n-2-j})/b_1$$

$$c_{m-n+2} = a_{m-n+2} - \sum_{j=1}^{m-n+1} c_j b_{m-n+3-j},$$

$$c_i = a_i - \sum_{j=1}^{m-n+1} c_j b_{i-j+1} \qquad (m-n+2 \le i \le m+1-re),$$

$$c_{m+1} = a_{m+1} - \sum_{j=1}^{m-n+1} c_j b_{m+2-j}$$

ahol $b_k = 0$ minden olyan k-ra, amely teljesíti az $n+2 \le k \le m+1$ feltételt. [4]

```
      DIMENSION A(10), B(10), C(19)
      INTEGER I, M, N
C     A FOKSZAMOK BEOLVASASA
      READ(5, 50) M, N
50    FORMAT(I2, I2)
C     AZ EGYUTTHATOK BEOLVASASA
      READ(5, 51) (A(I), I=1,M+1)
      READ(5, 51) (B(I), I=1,N+1)
51    FORMAT(10F5.1)
C     A POLYD SZUBRUTIN AZ A(X) / B(X) HANYADOST
C     A C(1),...C(M-N+1) ELEMEKBEN, A MARADEKOT A
C     C(M-N+2),...,C(M+1) ELEMEKBEN KEPEZI
      CALL POLYD(A, B, C, M, N)
C     AZ EREDMENYEK NYOMTATASA
      WRITE(6, 61) (A(I), I=1,M+1)
61    FORMAT(9H OSZTANDO, 10F5.1)
      WRITE(6, 62) (B(I), I=1,N+1)
62    FORMAT(9H    OSZTO, 10F5.1)
      WRITE(6, 63) (C(I), I=1,M-N+1)
63    FORMAT(9H HANYADOS, 10F5.1)
      WRITE(6, 64) (C(I), I=1,M-N+2,M+1)
64    FORMAT(9H  MARADEK, 10F5.1)
      STOP
      END

      SUBROUTINE POLYD(A, B, C, M, N)
      DIMENSION A(1), B(1), C(1)
      INTEGER M, N, I, J
C     POLINOMOK MARADEKOS OSZTASA.
C     FELTETELEZZUK, HOGY AZ EGYUTTHATOK SORRENDBEN
C     VANNAK, A(1) (ILL. B(1)) A LEGMAGASABB FOKU TAG
C     EGYUTTHATOJA. C(1..M-N+1 AZ A(1..M+1) / B(1..N+1)
C     OSZTAS HANYADOS-, C(M-N+2..M+1) A MARADEKPOLINOMANAK
C     EGYUTTHATOI.
      DO 10 I = N+2, M+1
10    B(I) = 0
C     A HANYADOS SZAMITASA
      C(1) = A(1) / B(1)
      DO 100 I = 2, M-N+1
      C(I) = A(I)
      DO 190 J = 1, I-1
190   C(I) = C(I) - C(J) * B(I-J+1)
100   C(I) = C(I) / B(1)
```

```
C       A MARADEK SZAMITASA
        DO 200 I = M-N+2, M+1
        C(I) = A(I)
        DO 290 I=1,M-N+1
190     C(I) = C(I) - C(J) * B(I-J+1)
200     CONTINUE
        RETURN
        END
```

3) Számítógépszimulátor

Ebben a példában definiálunk egy gépi kódú nyelvet, s elkészítünk ehhez egy interpretert. Számítógépünk memóriája álljon 99 szóból, amelyek mindegyikében elfér egy 8 decimális jelből álló szám. Az utasítások értelmezéséhez a számjegyeket felosztjuk négy kettes csoportba, s jelöljük e csoportokat a D0, D1, D2 és D3 szimbólumokkal. D0 (a legmagasabb helyiértékű két számjegy) jelenti az utasítás kódját, a másik három helyen az operandusok címei helyezkednek el. Az alábbi táblázat sorolja fel a nyelv utasításait (pl. az ADD jelentése: a D1 és D2 címen levő értékek összegét helyezd el a D3 címen).

az ut. neve	kód	jelentés
ADD	01 D1 D2 D3	D1 + D2 $\Rightarrow$ D3
SUB	02 D1 D2 D3	D2 – D1 $\Rightarrow$ D3
MUL	03 D1 D2 D3	D1 * D2 $\Rightarrow$ D3
DIV	04 D1 D2 D3	D1 / D2 $\Rightarrow$ D3
JMP	05 00 00 D3	D3 $\Rightarrow$ PC
	05 01 D2 D3	if D2 $<$ 0 then D3 $\Rightarrow$ PC
	05 02 D2 D3	if D2 $>$ 0 then D3 $\Rightarrow$ PC
	05 03 D2 D3	if D2 $=$ 0 then D3 $\Rightarrow$ PC
HALT	06 00 00 00	
READ	07 D1 00 00	adat olvasása a D1 címre
WRITE	08 D1 00 00	a D1 címen levő érték kiírása
SQRT	09 D1 00 D3	SQRT(D1) $\Rightarrow$ D3
ABS	10 D1 00 D3	ABS(D1) $\Rightarrow$ D3

PC a gép utasítás-számlálóját jelenti. A vezérlésátadást a fenti táblázatban az utasítás-számlálónak történő értékadással jelöltük. Az interpreter számára egy programot úgy kell megadni, hogy egy sorban két számnak kell szerepelnie: milyen címre milyen nyolcjegyű számot kell betölteni (a cím helyén megadott 00 jelenti a program végét). Egy program végrehajtása mindig a 01 -es című utasítással kezdődik. Ha végrehajtás közben hiba következik be, a memória állapotát ki kell nyomtatni, s a program végrehajtását be kell fejezni. [4]

```
        INTEGER I, I2, I3, K, PC, D0, D1, D2, D3
        DIMENSION MEM(99)

C       A GEP ALAPALLAPOTANAK BEALLITASA
```

```
      WRITE(6, 60)
60    FORMAT(1H1)
      DO 10 I = 1, 99
10    MEM(I) = 0

C     A PROGRAM BETOLTESE
11    READ(5, 50) I, K
51    FORMAT (I2, 1X, I8)
      IF (I .EQ. 0) GOTO 19
      MEM(I) = K
      GOTO 11
19    CONTINUE

      PC = 0
C     AZ UTASITASOK VEGREHAJTASAT ELVEGZO CIKLUS KEZDETE
899   PC = PC + 1
900   CONTINUE
C     EGY UTASITAS DEKODOLASA
      K = MEM(PC)
      IF (K .LT. 0) GOTO 999
      D3 = MOD(K, 100)
      K = K / 100
      D2 = MOD(K, 100)
      K = K / 100
      D1 = MOD(K, 100)
      D0 = K / 100
      IF (D0 .LT. 1 .OR. D0 .GT. 10) GOTO 999
      GOTO (901, 902, 903, 904, 905, 906, 907,
     *  908, 909, 910), D0
C     ADD
901   MEM(D3) = MEM(D1) + MEM(D2)
      GOTO 899
C     SUB
902   MEM(D3) = MEM(D2) - MEM(D1)
      GOTO 899
C     MUL
903   MEM(D3) = MEM(D1) * MEM(D2)
      GOTO 899
C     DIV
904   MEM(D3) = MEM(D1) / MEM(D2)
      GOTO 899
C     JMP
905   IF (D1 .GT. 3) GOTO 999
      GOTO (9050, 9051, 9052, 9053), D1 + 1
9050  PC = D3
      GOTO 900
9051  IF (D2) 899, 899, 9050
9052  IF (D2) 9050, 899, 899
9053  IF (D2) 899, 9050, 899
C     HALT
906   STOP
C     READ
907   READ(5, 51) MEM(D1)
51    FORMAT(I8)
      GOTO 899
C     WRITE
```

```
908     WRITE(6, 63) MEM(D1)
63      FORMAT(1X, I8)
        GOTO 899
C       SQRT
909     MEM(D3) = ISQRT(MEM(D1))
        GOTO 899
C       ABS
910     MEM(D3) = IABS(MEM(D1))
        GOTO 899
C       ERROR
999     DO 13 I = 1, 33
        I2 = I + 33
        I3 = I + 66
        WRITE(6, 65) I, MEM(I), I2, MEM(I2), I3, MEM(I3)
65      FORMAT(3(10X, I2, 1X, I8))
13      CONTINUE
        WRITE(6, 66) PC
66      FORMAT(//1X, 18HERROR IN STATEMENT, I3)
        STOP
        END
```

9. A gyakorlatok megoldásai

1.

$$\sum_{k=1}^{3}(4-k) * \frac{\sum_{i=5*(k-1)+1}^{5*k} b_i * t_i}{k * \sum_{i=5*(k-1)+1}^{5*k} t_i}$$

2.

$$A = \begin{pmatrix} 0.1 & -3.05 & -2.34 & -0.3 \\ 2.1 & 4.27 & -2.34 & -0.3 \\ 2.1 & -2.34 & 6.21 & -0.3 \end{pmatrix}$$

3.

a)
```
C     UTASITÁSFÜGGVÉNYEK :

      HOSSZ(X1,X2,Y1,Y2) = SQRT((X2-X1)**2 + (Y2-Y1)**2)
      SZOG(A,B,C,T) = ATAN2((4.0*T) / (A**2+B**2-C**2))
      ...
C     AZ OLDALAK HOSSZÁNAK SZÁMITÁSA
      AB = HOSSZ(X1,X2,Y1,Y2)
      AC = HOSSZ(X1,X3,Y1,Y3)
      BC = HOSSZ(X2,X3,Y2,Y3)

C     A TERÜLET SZÁMITÁSA
      S = (AB+BC+AC) / 2.0
      TER = SQRT(S*(S-BC)*(S-AC)*(S-AB))

C     A SZÖGEK SZÁMITÁSA
      ALFA = SZOG(AC,AB,BC,TER)
      BETA = SZOG(AB,BC,AC,TER)
      GAMMA = SZOG(AC,BC,AB,TER)
```

b)
```
      SUBROUTINE LET(S,K,I,J)

      DIMENSION S(20),Z(20,2,2)
      COMMON /T/ A(20),B(20),C(20),D(20)
      EQUIVALENCE (Z(1,1,1),A(1))
      S(K) = Z(I,J,K)
      RETURN
      END
```

4. Egyik sem.

5. a) és b) szabályos értékadás, c) szintaktikusan hibás.

6. Az első változat rövidebb idő alatt hajtódik végre, míg a második világosabb, jobban olvasható.

7. Mindegyik kód rendezetlen, indokolatlanul bonyolult.

a)
```
      IF(X1.LT.T(I).AND.T(I).LT.T2) IS = IS+1
```

b)
```
      IF(KIIR.EQ.1) WRITE(6,105) X
```

c)
```
      IF(T(I).LE.E) GOTO 20

          E = T(I)
```

```
              IND = I
20        CONTINUE
```

8. Igen.

9. Csak d), f) és h) helyes.

10. A második változat jobban olvasható, kisebb helyet foglal el, viszont valamivel több időt igényel.

11.
a) Ha az első program második sora

```
      DO 1 J=1,5
```

lenne, akkor azonos hatásúak lennének.
b) Az első program harmadik sora szemantikus hibát tartalmaz.

12. Az első program kódja rendezetlen, ez gyengébb megoldás a másik kettőnél. A második és harmadik megoldás egyenértékű.

13.
a) Helyes.

b) Az `IF(A(I,1)) 1,1,2` utasítás a két ciklus közös végére adja át a vezérlést, ami tilos.

14. `K = 7`.

15.

a)
```
      AB = A*B
      DO 10 I=1,100
          S = 0.0
          DO 11 J=1,100
11        S = S + X(I,J)
10    Q(I) = S * AB
```

b)
```
      F = ((X-64.0)*X*X+3.0)*X*X-17.0
```

c)
```
      DIMENSION X(10,20,8),XXX(1600)

      EQUIVALENCE (XXX(1),X(1,1,1))
      DO 1 I=1,1600
1     XXX(I) = 0.0
```

16.
```
      FUNCTION ARCTG(X)
      A = X
      DO 1 I=3,20,2
          T = (-1)**(I/2)*X**I/I
          IF(T.LT.1.E-6*A) GOTO 9
1         A = A+T
9     ARCTG = A
      RETURN
      END
```

17.

a)
```
      DIMENSION X(5)

      DATA X/-2.0,-1.2,1.7,2.4,4.9/
      S = 0.0
      DO 1 I=1,5
1     S = S + (X(I)**3 - A**3) /3.0
```

b)
```
      INTEGER FUNCTION F(N)
      INTEGER I,N
      F = 1
      DO 1 I=1,N
1     F = I*F
      RETURN
      END
```

18. Nem.

19. Veszélyes, hibára ad lehetőséget, hogy a `READ` és a `WRITE` ugyanazt a `FORMAT` -ot használja.

20. 19757654

21. Egyedül a `C(I) = A(I−2)` sor hibás, mert nincsen `C` vektor definiálva, függvény pedig nem kaphat értéket.

22.
a) `N=47237, X=2.0E8, G=6.237, M=1517,Y=11.734;`
b) `N=4723, X=72.0E8,G=3.715, M=170, Y=0.11734;`
c) `N=472372,X=0.0, G=0.2371517,M=0, Y=1173.4.`

23.

```
ETFO␣␣␣␣␣␣␣7␣␣␣ FO
```

24.

a)
```
␣␣␣␣␣␣␣␣␣␣␣
␣␣␣␣␣4.38
␣␣␣␣␣−6.47
␣␣␣␣␣9.07
```

b) `␣␣4.38␣␣−6.47␣␣␣9.07`

c) `4.38−6.47␣9.07`

d) `␣␣␣␣0.438000E+01␣−0.674␣␣␣ E+01␣␣0.907000E+01`

e) `4.4␣␣␣−6.47`

f) `4.380−6.470.907E+01`

g) `␣4.38−6.47␣␣9.070`

25. Mindegyik kifogásolható: a `FORMAT` nem határoz meg vezérlő karaktereket.

26.
a) Új lap tetejére összesen 28 sort nyomtat.
b) Hat számot írva egy sorba, kiírja a mátrix sorait úgy, hogy a mátrix két sora között mindig három üres sort hagy a nyomtatón.
c) Egy sorba hat számot írva, összesen tíz sorba kinyomtatja a mátrixot.

27. A második könnyebben olvasható.

28. A `SUM` változó nem kap kezdőértéket; a nyolcadik sorban helyesen `(−1)**(I/2)` kellene.

29. Nem. `N=0` esetén (ami nem lehetetlen, hiszen azt jelenti, hogy a mátrixnak nincs egyetlen nullától különböző eleme sem) hibás eredményt adhat.

30. Nem. `N=1` esetén hibás eredményt adhat.

31. Nem. A legegyszerűbben úgy lehet kijavítani, ha a szubrutin második sorát így írjuk : `DIMENSION A(100,10)`

32. 13.0

33.

a)
```
      IF(.NOT.(A.GT.B)) GOTO 11
      S = 1
      GOTO 19
11    IF(.NOT.(A.EQ.B.AND.C.GT.D)) GOTO 12
      S = 2
      GOTO 19
12    IF(.NOT.(A.EQ.B)) GOTO 13
      S = 3
      GOTO 19
13    IF(.NOT.(C.GT.D)) GOTO 14
      S = 4
      GOTO 19
14    IF(.NOT.(C.EQ.D)) GOTO 15
      S = 5
```

```
          GOTO 19
     15   S = 6
     19   CONTINUE
```

b)
```
          IF(MOD(K,2).NE.0) GOTO 9
          SUM = SUM+X
          PAROS = PAROS+1
      9   CONTINUE
```

c)
```
          INTEGER N(1000),L,M
          L = 1000
          M = ISQRT(L)
          DO 1 I=2,L
              IF(N(I).EQ.0) GOTO 8
                  WRITE(6,50) I
     50           FORMAT(1X,I5)
                  IF(I.GT.M) GOTO 7
                      DO 2 K=1,L/I
                          N(K*I) = 0
      2               CONTINUE
      7           CONTINUE
      8       CONTINUE
      1   CONTINUE
```

d)
```
          IF(X.GE.Y.AND.Y.GE.Z) A = Z

          IF(X.GE.Y.AND.Y.LT.Z) A = Y
          IF(X.LT.Y.AND.X.GE.Z) A = Z
          IF(X.LT.Y.AND.X.LT.Z) A = X
```

e)
```
          IF(A.LT.B) GOTO 10
          X = A
          GOTO 1
     10   X = B
      1   IF(X.GT.C) GOTO 2
          X = C
      2   CONTINUE
```

f)
```
          IF(A.NE.0) GOTO 21

          IF(B.NE.0) GOTO 21
          IF(C.EQ.0) GOTO 22
     21   A = A+1
     22   B = B+1
```

34.

a)
```
          isum := 0;
          for i:=1 step 1 until 5 do

              if helyes[i] = felel[i]
              then

                  begin iell[i]:=1; isum:=isum+1 end
              else

                  iell[i]:=0;
```

b)
```
          if ka>kb then begin kr:=ka; ka:=kb; kb:=kr end;
          for ka:=ka,kr while ka ≠ 0 do

              begin kr:=kb−kb/ka↑2; kb:=ka end;
```

c)
```
          procedure Rendez(v,m); integer m; array v[1:m];
          begin
              integer i,j,k; real t;
              for j:=1 step 1 until m−1 do begin

                  t := v[j+1];
                  for k:=1 step 1 until j do begin
                      i := j+1−k;

                      if t ≥ v[i] then goto 245;
```

```
                v[i+1] := v[i];
            end;
            i := 0;
245:    v[i+1] := t;
    end;
end;
```

d)

```
begin

    array a[1:100]; integer j,n,k; real x;
    for j:=1,j+1 while j ≠ n do begin

        if a[j] > a[j+1] then begin

            x := a[j]; a[j] := a[j+1]; a[j+1] := x;
            for k:=j−1,k−1

                while k ≠ 0 ∧ a[k] > a[k+1] do
            begin

                x := a[k]; a[k] := a[k+1]; a[k+1] := x
            end;
        end;
    end;
end;
```

æ