KANDÓ KÁLMÁN VILLAMOSIPARI MŰSZAKI FŐISKOLA
SZÁMITÁSTECHNIKAI TANSZÉKE

SZÁMITÁSTECHNIKAI FÜZETEK 8.

Budapest 1971.

KANDÓ KÁLMÁN VILLAMOSIPARI MŰSZAKI FŐISKOLA
SZÁMITÁSTECHNIKAI TANSZÉKE

SZÁMITÁSTECHNIKAI FÜZETEK 8.

FORTRAN reprezentációk TPA kisszámitógépre

Szerkesztette:

Ivanyos Lajos

TARTALOMJEGYZÉK

oldal

1. FORTRAN reprezentációk a TPA számitógépekre

A TPA 1001 és TPAi 1001 számitógépek alapkiépitésben 4 kiloszó kapacitású belső memóriával és ASCII-7 bites kódban dolgozó távirógéppel /teletype/, mint input/output perifériával működnek.

A továbbiakban TPA MINI FORTRAN-nak nevezett FORTRAN-szerű programnyelv alapkiépitésü TPA számitógépeken használható kétmenetes /forditás, futtatás/ feldolgozásra. Input/output utasitásai és formátum kezelése eltérnek a szabványos FORTRAN nyelvtől.

A FORTRAN szabványokhoz közelebb áll a TPA MINI D-FORTRAN változat, amely az előbb emlitett MINI FORTRAN bővitése két értelemben:

a./ Feltételezi az alapkiépités 32 kiloszó kapacitásu mágneslemez tárolóval /disk/ történt bővitését;

b./ Input/output utasitásai között szerepelnek a szabványos FORTRAN utasitások is, de formátumkezelése ugyanugy eltérő. Az input/output utasitások gyors lyukszalag olvasó és lyukasztó, valamint a disk kezelésére is alkalmasak.

A harmadik változat - melyet TPA FORTRAN-nak nevezünk - utasitásait és formátumkezelését tekintve elegettesz a FORTRAN programnyelv előirásainak, alig tér el a FORTRAN-II szabályaitól.

A TPA FORTRAN nyelvben elkészitett programok formai változtatás nélkül felhasználhatók az ICT 1900 sorozatú számitógépeken is, pontosabban a TPA FORTRAN nyelv részhalmaza az ICT 1905 FORTRAN reprezentációnak.

A TPA FORTRAN 4 kiloszó kapacitású TPA-n is alkalmazható, ha gyors lyukszalagolvasóval rendelkezik a számitógép, azonban célszerü 8 kiloszós, illetve 12 kiloszós gépet használni.

A továbbiakban a TPA FORTRAN-t ismertetjük ebben a részben, mig a TPA MINI FORTRAN-t és TPA MINI D-FORTRAN-t a részletes ismertetés mellőzésével a III. rész táblázataiban hasonlitjuk össze a többi változattal.

2. A TPA FORTRAN alapfogalmai

2.1 Program irási szabványok

A FORTRAN kódlap sorai 80 egységre, un. karakterpozicióra vannak felosztva. A kódlapot négy mezőre osztjuk:

1-5. karakterpozició: cimkemező
6. karakterpozició: folytatási mező
7-72. karakterpozició: utasitásmező
73-80. karakterpozició: azonositási mező.

A TPA FORTRAN cimkéi 1-től 4095-ig terjedő, előjel nélküli egész számok.

A folytatási mezőt abban az esetben használjuk, ha egy FORTRAN utasitás nem fér el egy sorban. Azt, hogy egy adott sor az előző sor folytatásaként értelmezendő, azzal jelezzük, hogy a folytatási mezőbe betüt vagy szám jegyet irunk. Valamennyi FORTRAN utasitás legelső sorában a 6. pozicionak üresnek kell lennie.

A FORTRAN utasitásokat az utasitásmezőben helyezzük el, nem nyulhatnak túl a 72. karakterpozición. A 73-80. karakterpoziciók a TPA FORTRAN-ban nem használhatók.

Ha egy sor első karakterpozicijába C betüt irunk, a forditóprogram a sort teljes egészében ignorálja. Ezt magyarázó szövegek elhelyezésóre használhatjuk ki /"comment-sor"/.

A programot lyukszalagra kell rögziteni.

2.2 Jelkészlet

A TPA FORTRAN programok irásához az alábbi jeleket használhatjuk fel:

Betük /az angol ábécé nagybetüi/: A, B, C, D, E, F, G, H, I, J, K, L, M, N, O, P, Q, R, S, T, U, V, W, Z, X, Y.

Számjegyek: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9.

Egyéb jelek: +, -, *, /, (,), =, vessző, pont, betüköz, HT /Horizontális Tabulátor/, CR /kocsi-vissza/, LF /soremelés/.

A betüköz és CR karaktereket a forditóprogram ignorálja. Az LF karakter sorterminátor. A HT karakter sor elején ugratást jelent a hetedik karakterpozicióra.

2.3 Konstansok

A konstansok tipusát alakjuk határozza meg.

Egész konstansok: legfeljebb 7 számjegyet és előjelet tartalmazhatnak. Értékük -2^{23} és $+2^{23}-1$ közé eshet.

Valós konstansok: legfeljebb tiz értékes számjegyet tartalmaznak. /Ha ennél többet irunk, nem hiba, de a belső ábrázolásban ezek nem vesznek részt./ Minden valós számkonstans tartalmaz tizedespontot, és tartalmazhat decimális exponenst. A valós konstansok alakja

$$\pm n . m E \pm s$$

ahol n és m decimális számjegyek sorozatai, s a decimális kitevő /egy vagy két decimális számjegy/. Mind a szám, mind a kitevő előjele elhagyható, ha az pozitiv. A számok abszolut értéke kb. 10^{76} és 10^{-76} között lehet, vagy pedig egyenlő zérussal.

2.4 Azonositók

Az azonositók hossza legfeljebb 6 karakter. Az első karakternek betünek kell lennie, a többi betü vagy számjegy lehet. Az első négy karakter nem egyezhet meg a TPA FORTRAN kulcsszavak első négy karakterével, tiltottak a következő betünégyesekkel kezdődő azonositók:

CALL	FORM	READ
CONT	FUNC	RETU
COMM	GOTO	STOP
DIME	MAST	SUBR
FINI	PAUS	WRIT

továbbá a DO, IF és EN betüpárokkal kezdődő azonositók is.

Az azonositók első betüje meghatározza az azonositó tipusát.

Azok az azonositók, amelyek az I, J, K, L, M vagy N betük valamelyikével kezdődnek, egész tipusuak, amelyek pedig a fentiektől különböző betüvel kezdődnek, valós tipusuak.

Ha egy azonositó 6 karakternél hosszabb, a forditóprogram a hetediktől kezdődő karaktereket ignorálja. Két

olyan azonositó, amelyek első hat karaktere megegyezik, a gépen belül nem különböztethető meg.

2.5 Skaláris változók

A FORTRAN nyelvben skaláris és indexes változókat különböztetünk meg. A skaláris változó valós vagy egész tipusu változó mennyiség, amelyhez szimbolikus nevet /azonositót/ rendeltünk hozzá. A valós változóknak a gépen belül egy rekesznégyes felel meg, amelyben egy négyszavas lebegőpontos ábrázolásu bináris számot tárolhatunk. Az egész változónak a gépen belül egy rekeszpár felel meg, amelyben egy kétszavas fixpontos /egész/ ábrázolásu bináris számot tárolhatunk.

A változókhoz rendelt azonositók meghatározzák a változó tipusát. A változó csakis olyan értéket vehet fel, amely az azonositója által meghatározott tipusnak megfelel.

2.6 Indexes változók

Az indexes változók összetartozó változókból álló csoportnak, un. tömbnek egy-egy elemét jelölik. Ilyen változócsoport lehet vektor, vagy mátrix. A változócsoportnak közös azonositója van, a tömbazonositó. Elemei közül 1 vagy 2 index segitségével választjuk ki a megfelelőt. A tömb-

azonositó meghatározza a tömb tipusát, ez a tipus a tömbhöz tartozó valamennyi indexes változóra érvényes.

Az indexet /vagy indexeket/ un. indexkifejezések segitségével adjuk meg. Az indexkifejezések lehetséges alakjai a következők:

k I ± j k I j

I ± j I

ahol k és j előjel nélküli egész konstansok, I pedig skaláris egész változó.

Példák indexes változókra:

A(I,J) B(I+2,2 K+1) Q(14)

P(KLIM, JLIM+5) SAM(J-2) L(1,3)

A példák közül az első öt valós, a hatodik egész tipusu tömbhöz tartozó változót jelöl. Q és SAM egyméretüek, a többi tömb kétméretü.

Ahhoz, hogy egy tömbelemre történő hivatkozás értelmes legyen, szükséges, hogy az indexkifejezésekben szereplő változók értékkel birjanak.

2.7 Tömbök

A tömbök összetartozó egész- vagy valós változók rendezett csoportjai. A TPA FORTRAN-ban egy- és kétdimenziós tömbökkel dolgozhatunk; az egydimenziós tömböket vektoroknak, a kétdimenziós tömböket mátrixoknak nevezzük. Egyméretü tömbökhöz egyindexes változók, kétméretü tömbökhöz kétindexes változók tartoznak.

Az egyméretü tömbök elemei az operativ tárolóban az indexek növekvő sorrendjében helyezkednek el. A kétméretü tömbök oszlopfolytonos elrendezésüek. A mátrix elemeit olymódon rendezzük lineáris sorozatba, hogy oszlopait hézag nélkül folytatólagosan egymás után illesztjük.

2.8 Számábrázolás

Az egész mennyiségek -2^{23} és $2^{23}-1$ között változhatnak, a valós mennyiségek exponense pedig -2^{11} és $2^{11}-1$ között változhat.

A valós számok ábrázolási határai kb. a $10^{\pm 615}$ nagyságrendü számok belső tárolását teszik lehetővé. A compiler célprogram konverziós szubrutinjai azonban csak kb. $10^{\pm 76}$ nagyságrendü számok ki- és bevitelét engedik meg. Amikor valamely aritmetikai müvelet eredményében a bináris exponens abszolut értéke eléri, vagy meghaladja az 1024-et, a gép figyelmeztető jelzést ad.

2.9 Aritmetikai müveletek

A TPA FORTRAN-ban az alábbi aritmetikai müveletek vannak:

Összeadás, jele: +	Osztás, jele: /
Kivonás, jele: -	Hatványozás, jele: **
Szorzás, jele: *	

A négy alapművelet csak azonos tipusu mennyiségek között van értelmezve, és eredményük ugyanolyan tipusu, mint a komponensek. A hatványozás művelete egész alap esetén csak egész kitevőre, valós alap esetén mind egész, mind valós kitevőre értelmezve van.

Az egész osztás eredménye a hányados abszolut értékének egész részével egyenlő, az előjelszabály szerint előjelezve. Az eredmény egész tipusu.

Példák:

25/5 = 5, 29/5 = 5, -13/4 = -3, 3/5 = 0.

Nullával való osztás értelmetlen.

A hatványozás eredménye egész tipusu, ha mind az alap, mind a kitevő egész. Az egész kitevőre történő hatványozást a gép ismételt szorzásokkal és - negativ exponens esetén - reciprok képzéssel számolja. A valós-valós tipusu hatványozás az

A**B = EXP (B*ALOG(A))

azonosság alapján történik. A valós hatványozás az EXP és ALOG standard függvényeknek a programba való beépitésével jár, növeli a program helyfoglalását. Ezért gazdaságosabb pl. A 2. helyett A 2-t irni.

2.10 Standard függvények

Elemi függvények

A TPA FORTRAN szubrutinkönyvtár az alábbi elemi függvények szubrutinjait tartalmazza:

függvény:	FORTRAN azonositója:
sinus	SIN
cosinus	COS
arcus tangens	ATAN
term.logaritmus	ALOG
exponenciális	EXP
négyzetgyök	SQRT

A függvények argumentumát mindig zárójelbe kell zárni, akkor is, ha egyetlen változó vagy konstans ez az argumentum. Argumentumként tetszőleges aritmetikai kifejezés is megadható. A függvény hivatkozások legfeljebb ötszörös mélységben egymásba skatulyázhatók.

Példák:

```
SIN(X)      EXP(A-C)      COS(SQRT(1.0-X**2))
ATAN 1./ 1.-EXP T
```

Egyéb függvények

függvény	definiciója	FORTRAN azonositója
abszolut érték /valós argumentum/	(X)	ABS(X)
abszolut érték /egész argumentum/	(I)	IABS(I)
Előjelezés /valós argumentumok/	sign(X1) (X2)	SIGN(X1, X2)

Előjelezés /egész argumentumok/	$sign(I_2)\ (I_1)$	ISIGN(I2,I1)
Átalakitás valósból egészbe	X→IX	IFIX(X)
Átalakitás egészből	IX→X	FLOAT(I)

A SIGN és ISIGN függvényeknek két argumentumuk van, melyek mindegyike lehet konstans, változó vagy aritmetikai kifejezés. Az argumentumokat egymástól vesszővel választjuk el.

A FLOAT függvény argumentumának tipusa egész, a függvényérték tipusa valós. IFIX argumentuma valós, a függvényérték egész. A függvényérték számértéke megegyezik az argumentum értékével, csak ábrázolási módja változik meg.

Bármely FORTRAN-ból forditott FUNCTION szegmens standard függvénnyé tehető, ha azt a Szubrutinkönyvtár szerkesztő program segitségével besoroljuk a TPA FORTRAN Programkönyvtárba.

2.11 Aritmetikai kifejezések

Az aritmetikai kifejezések operandusoknak müveleti jelekkel összekapcsolt sorozatai.

Az operandusok lehetnek:

a/ számkonstansok;

b/ skaláris változók;

c/ indexes változók;

d/ függvényhivatkozások;

e/ aritmetikai kifejezések, kerek zárójelbe zárva.

Aritmetikai kifejezés egy magában álló operandus is. Aritmetikai kifejezésben semmiképpen nem kerülhet egymás mellé két müveleti jel.

Példák aritmetikai kifejezésekre:

A

3.141592

B + 16.8946

(A - B(I,J+5))

3*C(J) + 4.1/(Z(I+2,3*K) * SIN(V))

(Q + V(M,N) - Y** 2) / (G*H - F(K+3))

Az operandusok aktuális értéke alapján kiszámitott értéket a kifejezés aktuális értékének nevezzük.

Az aritmetikai kifejezés aktuális értéke egész, ha valamennyi operandus egész tipusu, és valós, ha az operandusok valós tipusuak. Vegyes tipusu aritmetika /azaz egész és valós tipusu operandusok keverése/ a TPA 4K--FORTRAN-ban nem megengedett.

Az aritmetikai kifejezések kiszámitásában a müveletek sorrendjét egyrészt a müveletek elsőbbségi /precedencia/ szabálya, másrészt a zárójelezés mélysége határozza meg.

Elsődleges a hatványozás müvelete: **

Másodlagos a szorzás és osztás: * , /

Harmadlagosak az összeadás és a kivonás: +,

A TPA 4K-FORTRAN-ban a kiszámitásban balról

jobbfelé haladunk, pl. A+B+C kiszámitási sorrendje (A+B)+C. Az A/B/C kifejezés a TPA FORTRAN-ban (A/B)/C értelemben használható.

A precedenciaszabály által megszabott sorrend módositására zárójelezést alkalmazhatunk. A FORTRAN nyelvben csakis kerek zárójelek vannak értelmezve.

Amennyiben a programozó az egyenrangu müveletek között is szigoruan meghatározott müveleti sorrendet kiván előirni, ezt is zárójelezés segitségével érheti el. A redundáns zárójelek alkalmazása nem hiba, és biztositékot nyujthat a helyes müveleti sorrendre.

3. Végrehajtható utasitások

A TPA FORTRAN nyelv utasitásait két csoportba soroljuk: megkülönböztetünk végrehajtható és nem végrehajtható utasitásokat. A nem végrehajtható utasitások a forditóprogramnak szólnak, annak munkáját vezérlik, vagy olyan információt közölnek, melyet vagy közvetlenül a forditóprogram, vagy a beolvasóprogram használ fel.

3.1 Az értékadó utasitás

Az értékadó utasitás aritmetikai kifejezés kiszámitását, és az eredmények egy adott változóban való tá-

rolását irja elő.

V = A

ahol A tetszőleges aritmetikai kifejezés, és V vele megegyező tipusu - skaláris vagy indexes - változó.

Az A kifejezés maga is tartalmazhatja a V változót: a kifejezés kiszámitásánál V régi értékét kell figyelembe venni, majd a kiszámitás végeztével ez az érték törlődik, és helyébe a kifejezés kiszámitott aktuális értéke lép.

A TPA 4K-FORTRAN-ban megköveteljük, hogy a baloldali változó tipusa megegyezzék a jobboldali kifejezés tipusával, tipusváltás érdekében hivni kell a FLOAT, illetve IFIX átalakitó függvényeket.

3.2 Cimkék. A GO TO utasitás

A cimkék /hivatkozási számok/ az utasitások azonositására szolgáló 1-4 jegyü egész számok, melyeket a FORTRAN kódlap cimkemezőjében helyezhetünk el. A cimkék számértéke az $1 \leq n \leq 4095$ intervallumban kell legyen. A cimkék belsejében esetleg előforduló betükhöz /space/ karaktereket, valamint a szám elején álló értéktelen zérusokat a forditóprogram ignorálja, igy pl. a 121, 0121 és 01 21 cimkék ugyanazt jelentik.

Amennyiben egy utasitásnak nincs cimkéje, a cimkemezőt üresen kell hagyni, azaz az 1.-5. karakterpoziciókban betüköz karaktereknek kell állniuk. Amennyiben az 1.

karakterpozición betü áll, a megfelelő sort a fordító-program kommentárnak /Comment sor/ tekinti és ignorálja. Ha a cimkemező üres, a betüköz karakterek helyettesithetők egyetlen HT karakterrel; a HT karakter a cimke és a hozzátartozó utasitás közti karakterpoziciók kitöltésére is felhasználható.

Példák:

```
1234  X = X + A
  01  DELTA = A**2 + B**2
1  2  GAMMA = BETA - FLOAT(I)*TAU
```

A cimkék az utasitások azonositására szolgálnak, a vezérlésátadó /ugrató/ utasitásokban. A GO TO utasitás alakja:

GO TO n

ahol n valamely végrehajtható utasitás cimkéje. Az utasitás hatására a program végrehajtása azon az utasitáson folytatódik, amely az n cimkét viseli. Példa:

```
GO TO  1234
GO TO  1
GO TO  12
```

Tekintettel arra, hogy a forditóprogram figyelmen kivül hagyja a betüköz karaktereket, a fenti utasitások irhatók GOTO 1234, GO TO1, GOT 01 2 stb alakban is.

3.3 Az aritmetikai IF utasitás

Az aritmetikai IF utasitás adott aritmetikai kifejezés aktuális értékének előjele szerinti program elágaztatást /feltételes ugrás/ tesz lehetővé. Az utasitás alakja:

IF(A) n1, n2, n3

ahol A tetszőleges aritmetikai kifejezés és n1, n2, n3 cimkék. Az utasitás hatására az IF utasitás után az n1, n2 vagy n3 cimkéjü utasitás kerül végrehajtásra aszerint, hogy A előjele negativ, zérus vagy pozitiv.

Példa:

IF (R) 12, 13, 14

A fenti példában vezérlésátadás történik a 12-es cimkére, ha R negativ, a 13-asra, ha R egyenlő zérussal, és a 14-esre, ha R pozitiv.

Felhivjuk a figyelmet, hogy mind a GO TO, mind az IF utasitásban szereplő cimkének végrehajtható /nem FORMAT/ utasitás cimkéjének kell lennie.

3.4 Ciklusképzés. A DO utasitás

A DO utasitást programciklusok /hurkok/ képzésére használhatjuk fel. Az utasitás alakja:

DO n I = m1, m2, m3

vagy

DO n I = m1, m2

ahol n valamely végrehajtható utasitás cimkéje, I egész tipusu skaláris változó, és m1, m2 és m3 egész konstansok vagy egész tipusu skaláris változók. Az n cimkéjü utasitásnak az utasitások felirási sorrendjében a DO utasitás után kell következnie.

A DO utasitás egy programciklust /hurkot/ határoz meg. Hatása a következő:

1. A ciklusváltozó felveszi az m1 értéket.

2. Végrehajtódnak azok az utasitások, amelyek a DO utasitás és az n cimkéjü utasitás között helyezkednek el, beleértve magát az n cimkéjü utasitást is.

3. A program megvizsgálja, hogy I értéke meghaladta-e már az m2 értéket. Ha igen, a program végrehajtása az n cimkéjü utasitás utáni FORTRAN utasitással folytatódik. Ha nem:

4. Az I ciklusváltozó értékéhez hozzáadódik m3 értéke /vagy 1, ha m3 nem szerepel a DO utasitásban/.

5. Visszatérés /visszaugratás/ következik a DO utasitást követő FORTRAN utasitásra, és a 2.-5. alattiak mindaddig ismétlődnek, amig az I változó meg nem haladja m2 értékét.

Példa:

```
      S = 0.0
      DO 12 K = 1,20
   12 S = S + X(K)
```

A példabeli program hatására összeadódik az X vektor első 20 eleme, és az összeg az S skaláris változóban képződik.

Másik példa:

```
      Y = 1.0
      Z = X**2
      T = 1.0
      DO 16 K = 2, 19, 2
      AK = FLOAT(K)
      T = T*Z/(AK*(AK - 1,0))
   16 Y = Y + T
```

A fenti program a ch(x) függvény sorfejtésének:

$$ch = 1 + \frac{x^2}{2!} + \ldots + \frac{x^{2n}}{2n!} + \ldots$$

első 10 tagját számitja ki.

A DO ciklust lezáró utasitás - amely az n cimkét viseli - végrehajtható utasitás kell legyen, és nem lehet GO TO, IF, DO, PAUSE, STOP, RETURN. Annak érdekében, hogy könnyen elkerülhessük ezeknek az utasitásoknak DO végi alkalmazását, használjuk a CONTINUE utasitást. Az utasitás alakja:

```
CONTINUE
```

A CONTINUE utasitás önmagában hatástalan; ha DO ciklus záróutasitásaként használjuk, akkor a programnak azt a helyét jelöli ki, ahonnét vissza kell térni a ciklus elejére, vagy folytatni kell a program végrehajtását.

Példa:

```
      Y = 1.0
      Z = X**2
      T = 1.0
      DO 16 K = 2, 39, 2
```

```
      AK = FLOAT(K)
      T = T*Z/(AK*(AK - 1.0))
      Y= Y + T
      IF T/Y - 4.0E - 9  20, 20, 16
   16 CONTINUE
```

A program az előző példa módositott változata, melyben a ch(x) függvény sorfejtését adott pontossági követelménynek megfelelő számu, de legfeljebb 20 tagra számitjuk ki.

A ciklustörzs utasitása között lehetnek olyanok, amelyek vezérlésátadást hajtanak végre a ciklustörzsön kivülre, viszont tiltott a vezérlésátadás kivülről a ciklustörzs belsejébe.

A ciklusváltozó értéke a ciklusból való kilépés után az az érték, mellyel a ciklus utoljára végrehajtódott.

A DO ciklusok egymásba is skatulyázhatók. A többszörösen egymásba skatulyázott ciklusok esetében a korábban elmondottakat értelemszerüen, többszörös mélységben alkalmazzuk.

A DO ciklusok egymásba skatulyázása a zárójel struktura szabályai szerint történik, azaz a legutoljára megnyitott DO ciklust kell legelsőként bezárni.

Egymásba skatulyázott DO strukturák esetében szabad GO TO vagy IF utasitással egy belsőbb DO utasitás hatásköréből egy külsőbb DO hatáskörébe /vagy a teljes DO strukturán kivülre/ ugratni; tilos azonban a vezérlésátadás valamely DO strukturán kivülről a struktura hatáskörén belülre.

A TPA 4K-FORTRAN-ban a DO ciklusok maximálisan 10-szeres mélységben skatulyázhatók egymásba.

3.5 Megállás. A STOP és PAUSE utasitások

A program futásának megállitására a TPA 4K-FORTRAN-ban két utasitás áll rendelkezésünkre: a PAUSE és a STOP. Az ASA FORTRAN nyelv különbséget tesz a két utasitás hatása között, amennyiben a PAUSE ideiglenes, a STOP végleges megállás. A TPA FORTRAN-ban a két utasitás teljesen egyenértékü. Az utasitások lehetséges alakjai:

```
STOP
STOP xx
PAUSE
PAUSE xx
```

ahol xx kétjegyü oktális szám. Az utasitások hatására a program kinyomtatja a konzolirógépen a

```
HALTED: xx
```

illetve

```
HALTED:
```

üzenetet aszerint, hogy xx szerepelt-e az utasitásban vagy nem. A program a CONTINUE gomb megnyomásával tovább inditható, ekkor a program végrehajtása a soronkövetkező FORTRAN utasitással folytatódik.

4. Ki- és bevitel. A READ, WRITE és FORMAT utasitások

A ki- és beviteli müveleteket a TPA 4K-FORTRAN-ban a WRITE és a READ utasitások segitségével programozhatjuk. Az utasitások alakja:

WRITE (n,f) L

és

READ (n,f) L

ahol n a periféria száma, f az utasitáshoz tartozó FORMAT utasitás cimkéje, L pedig az un. input-output lista. Az n perifériaszám a TPA 4K-FORTRAN-ban az 1, 2, 3 és 4 számok valamelyike lehet, az alábbiak szerint:

n = 1 gyors szalagolvasó

n = 2 gyors szalaglyukasztó

n = 3 Teletype olvasó

n = 4 Teletype kiiró/lyukasztó

A READ utasitást beolvasási, a WRITE utasitást kiirási vagy szalaglyukasztási müveletek programozására használjuk. Ennek megfelelően n a READ utasitásban 1 vagy 3, a WRITE utasitásban 2 vagy 4 lehet.

Minden READ és WRITE utasitáshoz hozzá van rendelve egy FORMAT utasitás, melynek mindig kell legyen cimkéje. Erre a cimkére hivatkozik az f szám a WRITE és READ utasitásokban. Ugyanaz a FORMAT utasitás egyszerre több READ/WRITE utasitáshoz is hozzá lehet rendelve.

4.1 Az input-output lista

Az L input-output (I/O) lista az átvitelben résztvevő változókat és tömbelemeket határozza meg. Elemei lehetnek:

1. Skaláris változók,
2. Indexes változók,
3. Tömbök,
4. Implicit DO ciklusok,

vagy pedig a lista lehet teljesen üres.

A skaláris változók és tömbök a megfelelő változónak vagy tömbelemnek kiirását, illetve beolvasását jelentik. A listában szereplő tömbök a tömb valamennyi elemének átvitelére jelenenek utasitást a tömbelemek tárolási sorrendjében. Az input-output lista elemeit vesszővel választjuk el egymástól.

Példa: Legyenek I, ALFA, K és BETA egész, ill. valós skaláris változók, TOMB pedig 3x3-as tömb. A

WRITE(4,25) I, ALFA, K, BETA, TOMB

utasitás az I, ALFA, K, BETA, TOMB(1,1), TOMB(2,1), TOMB(3,1), TOMB(1,2), TOMB(2,2), TOMB(3,2), TOMB(1,3), TOMB(2,3), TOMB(3,3) változók, illetve tömbelemek kiirását jelenti /a fenti sorrendben/, a teletype irógépen. A kiirás a 25-ös cimkéjü FORMAT utasitás vezérlete alatt történik.

Az implicit DO ciklusok több tömbelem csoportos átvitelét irják elő. Az implicit DO ciklus alakja:

$(L, I = m_1, m_2, m_3)$

vagy

$$(L, I = m_1, m_2)$$

ahol I, valamint m_1, m_2 és m_3 jelentése ugyanaz, mint a DO ciklusoknál, L pedig tetszőleges I/O lista az alpont elején adott definiciónak megfelelően.

Példák:

(A(I), B(I), C(I), I = 1, N)

(K, X(K+1), Y(3*K+2), K = N1, MAX, N2)

Az implicit DO ciklus értelmezése a következő:

1. A ciklusváltozó felveszi az m_1 értéket.
2. Átvitelre kerülnek az L I/O lista elemei.
3. A program megvizsgálja, hogy I értéke elérte, vagy meghaladta-e már az m_2 értéket. Ha igen, az implicit ciklus által meghatározott átvitel befejeződött. Ha nem:
4. Az I ciklusváltozó értékéhez hozzáadódik m_3 értéke, /vagy 1, ha m_3 nem szerepel az implicit DO ciklusban/.
5. A megnövelt I értékkel ujra megismétlődnek a 2-4. alattiak mindaddig, amig I értéke el nem éri, vagy meg nem haladja m_2 értékét.

Az első példa az A(1), B(1), C(1), A(2), B(2), C(2), ..., A(N), B(N), C(N) vektorelemek átvitelére szóló utasitást jelent.

A második példában a következő értékek kerülnek átvitelre: N1, X(N1+1), Y(3*N1+2), N1+N2, X(N1+N2+1),

Y 3 N1+N2 +2 mindaddig, amig K értéke el nem éri, vagy meg nem haladja a MAX változó aktuális értékét. Az implicit DO ciklusban szereplő I/O lista tartalmazhatja magát a ciklusváltozót is, de csak WRITE utasitásban, READ-ben nem.

Az implicit DO ciklus csupán egy eleme az I/O listának, és egy I/O listában ilyenből több is előfordulhat. Példa:

(X(I), I = 1, 21, 2), (X(I), I = 2, 20, 2)

Ez a példa egy 21 elemü vektor elemeinek átvitelét jelenti olymódon, hogy először a páratlan, utána a páros indexü elemek kerülnek átvitelre. Az implicit DO ciklusok a listán egyéb elemek között is előfordulhatnak.

Többszörösen egymásba skatulyázott implicit DO ciklusok is létrehozhatók.

Az implicit DO utasitások egymásba skatulyázásának mélységére - az explicit ciklusokkal ellentétben - a TPA 4K-FORTRAN nem tesz korlátozást.

4.2 A FORMAT utasitás

Minden WRITE és READ utasitáshoz tartozik egy FORMAT utasitás, amely az átvitelek formáját határozza meg. Az utasitás alakja:

n FORMAT (S)

ahol n a FORMAT utasitás cimkéje, S pedig az un. FORMAT lista. A FORMAT utasitások a programban a végrehajtható

utasitások között bárhol előfordulhatnak.

1. A perifériális átvitelekben szereplő információt rekordokra bontjuk. Egy rekord a teletype irógépen megfelel egy sornak, a lyukszalagon pedig két LF /line feed/ karakter közti szakasznak. Egy rekordban egy vagy több számadat, szövegek és szóközök helyezkedhetnek el, vagy pedig a rekord lehet teljesen üres. Teletypeon történő kiíráskor egy rekord legfeljebb 72 karakterből állhat. Szalaglyukasztáskor a rekord hossza tetszőleges.

2. A rekordot mezőkre osztjuk. A mező lehet egy számadat /numerikus mező/, vagy egy szövegrész /Hollerith mező/. A mező karakterekből áll. A mező karaktereinek számát mezőszélességnek nevezzük, és a továbbiakban w-vel jelöljük.

3. Csak számadatok /numerikus mezők/ esetében értelme van beszélni a tizedesrész számjegyeinek számáról /ez a tizedespont utáni jegyek száma/. Ezt a továbbiakban d-vel jelöljük.

4. Az S FORMAT lista formátum specifikációk sorozata, amelyeket vesszők, és/vagy rekord vége jelek választanak el egymástól. A specifikációk lehetnek konverziós specifikációk, melyeket numerikus mezők átvitelénél használunk, vagy pedig szerkesztési specifikációk, melyek az adatok elrendezését szabályozzák. A TPA 4K-FORTRAN-ban lehetnek I, E, F konverziós specifikációk. A szerkesztési specifikációk közé tartozik a H és az X specifikáció, valamint a rekord vége jel.

5. Az S FORMAT lista bármely részlistáját kerek zárójelbe zárhatjuk, és a zárójel elé egy előjel nélküli egész számot irhatunk. Ezt a számot ismétlési tényezőnek nevezzük, és k-val jelöljük. A zárójelbe zárt részlista neve ismétlési csoport. Ha az ismétlési tényező értéke 1, akkor el is hagyható.

Példák: FORMAT utasitásokra

123 FORMAT(I6, 3F7.2, 3HSOK/2(E8.1,I2))

Itt I6, F7.2, E8.1, I2 konverziós specifikációk, 3HSOK szerkesztési specifikáció, a ferde törtvonal (/) a rekord vége jel; az F7.2 specifikáció előtt a 3 és az (E8.1, I2) részlista előtt álló 2 ismétlési tényezők. 123 a FORMAT utasitás cimkéje.

Másik példa:

555 FORMAT (/(4I5, 4F7.4))

Itt 555 a FORMAT utasitás cimkéje, a törtvonal ismét rekord vége jel, a (4I5, 4F7.4) részlista zárójelben áll, hiányzik előle az ismétlési tényező, ami egyértelmü azzal, hogy értéke 1.

A FORMAT lista és az I/O lista szoros kölcsönhatásban vannak: az I/O lista minden egyes eleméhez tartozik egy konverziós specifikáció, amely a megfelelő elem átvitelét vezérli. Az átvitel végrehajtása ugy történik, hogy először is uj rekordot nyitunk, majd balról jobbra tartó sorrendben sorravesszük az I/O és a megfelelő FORMAT listaelemeket, és mindegyik I/O listaelemét a hozzárendelt FORMAT elem rendelkezései szerint visszük át. A FORMAT lista-

elem tipusának összhangban kell lennie a hozzátartozó I/O listaelem tipusával; egész tipusu változó átvitelének vezérlésére I, valós elem átvitelének vezérlésére pedig E vagy F specifikációt alkalmazhatunk. A szerkesztési specifikációkhoz nem tartozik I/O listaelem.

4.3 Egészek átvitele. Az I specifikáció

Az I konverziós specifikáció az I/O lista egész tipusu elemeihez lehet hozzárendelve. A specifikáció alakja:

kIw

ahol k az ismétlési tényező /elhagyható, ha értéke 1/, és w a mezőszélesség. A specifikáció hatására az I/O listának soronkövetkező eleme - melynek egész tipusu elemnek kell lennie - w karakterből álló egész számként konvertálódik.

Input esetén az input rekord soronkövetkező w számu karaktere kétszavas ábrázolásu bináris egésszé konvertálódik, és elhelyeződik az I/O listaelem által meghatározott változóban. Az input rekord szóban forgó karakterei között csak számjegy, előjel és betüköz karakter fordulhat elő. Betüköz csak az első számjegy, illetve az előjel előtt állhat. Ha a számnak van előjele, akkor ennek közvetlenül az első számjegy előtt kell állnia.

Példa: Legyen a beolvasási utasitás

READ(1,20) I, J, K

a hozzá tartozó FORMAT utasitás pedig

20 FORMAT(3I5)

vagyis az ismétlési tényező 3. Ez az utána következő konverziós specifikáció háromszori ismétlését jelenti, azaz egyenértékü a következővel:

20 FORMAT (I5, I5, I5)

Output esetén az I konverziós specifikáció az I/O listaelemnek w jegyü egész számként történő kinyomtatását /lyukasztását/ irja elő. Pozitiv számok kiirásakor a "+" előjel nem jelenik meg, a "-" előjel pedig közvetlenül az első számjegy elé kerül. A szám előtt annyi betüköz nyomtatódik, amennyi a karakterek számát w-vé egésziti ki.

Példa: legyenek az I, J és K változók értékei rendre 197, -6 és 0, legyen a kiirási utasitás

WRITE(4,20) I, J, K

a 20-as cimkéjü FORMAT utasitás pedig mint fent. Az utasitás hatására az output rekord 15 karaktere

xx197xxx-6xxxx0

lesz.

Általános szabály, hogy amit egy adott FORMAT utasitás szerint kinyomtattunk, azt ugyanezen FORMAT szerint vissza is olvastathatjuk, vagyis a beolvasás a kiirás megforditottja.

A mezőszélesség megválasztásánál ügyelni kell arra, hogy a w számu karakter elegendő legyen a kiirandó szám aktuális értékének elhelyezésére, ellenkező esetben kiirási tulcsordulás jön létre.

4.4 Valós mennyiségek átvitele. Az F specifikáció.

Az F konverziós specifikáció az I/O lista valós tipusu elemeihez lehet hozzárendelve. A specifikáció alakja:

kFw.d

ahol k az ismétlési tényező /elhagyható, ha értéke 1/, w a mezőszélesség és d a tizedespont utáni jegyek száma.

Input esetén az input rekord soron következő w számu karaktere - amelyek egy decimális tizedesszám karakterei - négyszavas lebegőpontos ábrázolásu bináris számmá konvertálódik, és elhelyeződik az I/O listaelem által meghatározott változóban. Az input rekord szóbanforgó karakterei között csak számjegy, előjel, tizedespont, az E betü és betüköz karakter fordulhat elő.

Input adatok decimális exponensében nem szükségképpen kell előfordulnia az E betünek; decimális exponensként kerül értelmezésre bármely, közvetlenül a törtrész után álló előjeles egész szám is /pl. 3.2+1=3.2E1/.

Input adatként előfordulhatnak tizedespontot nem tartalmazó /vagyis formailag egész/ számok is.

A d számnak /tizedesrész jegyeinek száma/ input esetén csak akkor van jelentősége, ha az input adat nem tartalmaz tizedespontot, azaz formailag egész szám. Ebben az esetben d a hallgatólagos tizedespont helyét adja meg: az input adatból d számu tizedesjegyet kell "levágni". Ha az input adat tartalmaz tizedespontot, akkor a specifikációban szereplő d értéke teljesen közönbös.

Példák:

Specifikáció	Input adat	A beolvasás eredménye
F6.2	3.18	+3.18
	318	+3.18
	31.8	+31.8
	0.318	+0.318
	-0.318	-0.318
F7.3	3.2E2	320.0
	+3.2E-1	+0.32
	+3.2+4	+32.0
F5.0	-2.16	-2.16
	-216	-216.0

Output esetén az F konverzió fixpontos decimális kiírási formátumot határoz meg; a kiirt szám összesen w számu karakterből áll, melyből d számu jegy következik a tizedespont után. Pozitiv számok kinyomtatásakor a "+" előjel nem jelenik meg, a "-" előjel közvetlenül az első számjegy elé kerül. Ügyelni kell arra, hogy az output listaelem aktuális értéke olyan legyen, hogy a kiiráshoz szükséges karakterek száma w-t ne haladja meg, mert ellenkező esetben kiirási tulcsordulás lép fel.

Példák:

Specifikáció	Az I/O listaelem aktuális értéke	Kinyomtatott szám
F6.2	+3.18	3.18
	+318.	318.00
	+0.318	0.31
F5.0	-2.16	-2.
	-216.	-216.

F specifikáció esetében is az output mező balszélén betüköz karakterek jelennek meg, ha a szám nagyságrendje nem teszi szükségessé valamennyi karakterpozició kihasználását. Érvényes az a szabály is, amely szerint az F specifikációval kiiratott számanyag azonos FORMAT lista szerint vissza is olvasható.

4.5 Valós mennyiségek átvitele. Az E specifikáció.

Az E konverziós specifikáció az I/O lista valós elemeihez lehet hozzárendelve. A specifikáció alakja:

kEw.d

ahol k az ismétlési tényező /elhagyható, ha értéke 1/, w a mezőszélesség, és d a decimális törtrész jegyeinek száma. A specifikáció hatására az I/O lista soronkövetkező eleme

összesen w karakterből álló, decimális számként konvertálódik.

Input esetén a specifikáció teljesen egyenértékü az F specifikációval.

Output esetén a szám decimális normálalakra konvertálódik, 0.1 és 1.0 közé eső törtrésszel, és megfelelő decimális kitevővel iródik ki. A specifikáció hatására mindig zérus egészrész nyomtatódik. A decimális exponenst az E betü vezeti be, utána következik a decimális kitevő.

Példák:

Specifikáció	Az I/O listaelem aktuális értéke	Kinyomtatott szám
E10.4	+4.239	0.4239E 1
	+0.4239	0.4239E 0
	+0.004239	0.4239E -2
E9.2	-5.64	-0.56E 1
	-0.564	-0.56E 0
	+0.000000000002	0.20E -11
	$+2.3\ 10^{36}$	0.23E 37
	+3.	0.30E 1

Ha a szám nagyságrendje nem teszi szükségessé valamennyi karakterpozició kihasználását, az output mező balszélén betüköz karakterek jelennek meg.

4.6 Karakterinformáció átvitele. A H specifikáció.

A H specifikáció az un. szerkesztési specifikációk csoportjába tartozik, nincs hozzárendelve I/O listaelem. A specifikáció alakja:

$$wHh_1h_2\ldots h_w$$

ahol a w a mezőszélesség, és h_1, ..., h_w tetszőleges karakterek, amelyeknek van 6 bites belső kódjuk.

A specifikáció hatására output esetén az output rekord soron következő w karaktere a h_1 ... h_w karaktersorozat lesz. Input esetén az input rekord soron következő w karaktere beolvasódik a leforditott programnak azon a rekeszeibe, ahol eredetileg a h_1 ... h_w karaktersorozat tárolódott, s azt felülirja /az eredeti h_1 ... h_w sorozat elvész/. A H specifikációt szövegek beolvastatására és kiiratására használhatjuk.

Példa. Az input rekord soron következő 7 karaktere a következő karaktersorozat:

MASODIK

Tekintsük a következő FORTRAN utasitás sorozatot:

```
      WRITE(4,11)
      READ(1,11)
      WRITE(4,11)
11    FORMAT 7HELSOxxx
```

A fenti FORTRAN utasitások hatására az output rekordon a következő nyomtatási kép jelenik meg:

```
ELSOxxx
MASODIK
```

Az első kiirásnál még az eredeti szöveg iródik ki, majd ezt felülirja a lyukszalagról beolvasott ujabb szöveg, és másodszorra már ez utóbbi kerül kiirásra. Uj READ/WRITE utasitás uj rekordot jelent, igy a két kiirás különböző sorokba kerül.

4.7 Betüközök kiirása és input mezők ignorálása. Az X specifikáció.

Az X specifikáció a szerkesztési specifikációk csoportjába tartozik, nincs hozzárendelve I/O listaelem. A specifikáció alakja:

wX

ahol w a mezőszélesség.

A specifikáció hatására output esetén w számu betüköz karakter kerül kiirásra az output rekordra. Input esetén a gép az input rekord soron következő w számu karakterét ignorálja.

4.8 Rekord végének jelzése. A "/" specifikáció.

Minden READ és WRITE utasitás végrehajtása uj rekord megnyitásával kezdődik. Input esetén ez annyit jelent, hogy amennyiben az előző rekordnak nem került volna beolvasásra valamennyi karaktere, a fennmaradók figyelmen kivül maradnak, és az olvasó a következő rekord kezdetére áll.

Output esetén uj rekord nyitása egy CR és egy LF karakter kiirását /lyukasztását/ jelenti.

A FORTRAN nyelvben a FORMAT listán belül uj rekord nyitására külön specifikáció áll rendelkezésre. Ennek jelölésére a ferde törtvonalat: "/" használjuk, és mint specifikációt, "rekord vége" jelnek nevezzük. Ennek a specifikációnak az I/O listán nem felel meg elem. A ferde törtvonalnak elhatároló jel szerepe is van, amennyiben pótolja a specifikációkat elválasztó vesszőt, sem a rekord vége jel előtt, sem utána nem kötelező /de szabad/ a vesszőt kiirni. Több rekord vége jel is állhat egymás mellett.

Output esetén a rekord vége jel hatására az output rekordra egy CR-LF iródik /lyukasztódik/, vagyis a nyomtatás a következő sor elején folytatódik. Input esetén az olvasó ignorálja a kurrens rekord esetleg hátralévő karaktereit, és előre mozgatja a szalagot a legközelebbi LF karaktert követő első karakter elé. Több egymás melletti rekord vége jel hatására megfelelő számu üres rekord /üres sor/ iródik, illetve megfelelő számu input rekord beolvasás nélkül áthalad az olvasón.

4.9 Az I/O és a FORMAT listák kölcsönhatása

1. Minden egyes I/O művelet megkezdése uj rekord megnyitásával jár együtt.

2. Az átvitel az I/O és a FORMAT listák párhuzamos

átvizsgálása alapján történik. Minden I/O listaelemnek megfeleltetünk egy FORMAT listaelemet, melynek tipusa I lehet, ha az átviendő I/O listaelem egész tipusu, és E vagy F, ha az I/O listaelem valós tipusu. Az I/O lista minden egyes eleme a hozzá tartozó FORMAT specifikáció szerint kerül átvitelre. A konverziós specifikációk között szerkesztési specifikációk foglalhatnak helyet, melyekhez nem tartozik I/O listaelem.

3. Amennyiben az I/O lista elemeinek száma nem egyezik meg a FORMAT lista elemeinek számával, az átvitt elemek számát az I/O lista elemeinek száma szabja meg. Ha ez kevesebb, mint a FORMAT lista elemeinek száma, az átvitel véget ér az utolsó I/O lista elemnek megfelelő konverziós specifikáció értelmezése után. Ha az I/O listaelemek száma meghaladja a FORMAT lista elemeinek számát, akkor a FORMAT lista egyszeri teljes feldolgozása után uj rekord megnyitására kerül sor, majd a FORMAT lista értelmezése részben, vagy egészben megismétlődik, az alábbiak szerint:

4. Ha a FORMAT lista nem tartalmaz ismétlési csoportot, akkor a FORMAT lista értelmezését <u>elölről</u> kell ujrakezdeni. Ha ellenben a FORMAT lista tartalmaz ismétlési csoportot, akkor a FORMAT lista értelmezése annak az ismétlési csoportnak a kezdetétől kezdődik ujra, amelynek végzárójele a FORMAT lista végéhez a lehető legközelebb esik /ennek ismétlési tényezőjét is figyelembe véve/. A FORMAT lista végének elérése minden esetben uj rekord nyitását eredményezi.

5. Mind az I/O, mind a FORMAT listának lehetnek

"többszörös" elemei. Az I/O lista többszörös elemei a tömbazonositók, valamint az implicit DO ciklusok. A FORMAT lista többszörös elemeit az ismétlési csoportok, valamint azok a konverziós specifikációk jelentik, amelyek ismétlési tényezője 1-től különböző.

6. Az ismétlési csoportok maximálisan négyszeres mélységben egymásba skatulyázhatók.

Példa:

Legyen az L egész tipusu tömb 4x3-as méretü, melynek elemei:

```
 4  -62   7
 0    3   5
11    6 -24
-6    2   9
```

Irassuk ki ezt a

```
WRITE(4,7) L
```

WRITE utasitással.

A hozzárendelt FORMAT legyen:

```
7 FORMAT(8HEREDMÉNY/4 I4)
```

Ebben az esetben a nyomtatási kép a következő lesz:

```
EREDMÉNY
 4   0  11  -6
-62   3   6   2
 7   5 -24   9
```

5. Szegmentált programok és deklarativ utasitások

A TPA 4K-FORTRAN programok szegmentált szerkezetüek, a program egy vagy több, lényegileg független szegmensből áll, ezek a szegemensek külön-külön fordithatók, majd egy erre a célra kidolgozott program, az ÁBL segitségével állithatók össze egyetlen programmá. A TPA 4K-FORTRAN forditóprogram az egyes szegmensekről á/b formátumú forditást készit, ami lehetővé teszi ezeknek a szegmenseknek a programkönyvtárba való egyszerü beillesztését is.

5.1 Szegmensnyitó és záró utasitások: SUBROUTINE, FUNCTION MASTER, END

A TPA 4K-FORTRAN szegmensek a MASTER, a SUBROUTINE vagy a FUNCTION, un. szegmensnyitó utasitások valamelyikével kezdődhetnek. Ezen utasitások alakja:

```
SUBROUTINE SNEV(P1, P2, ..., PN)
SUBROUTINE SNEV
FUNCTION FNEV(P1, P2, ..., PN)
MASTER MNEV
```

ahol SNEV, FNEV és MNEV a 2.4 pontban elmondottaknak megfelelő azonositók, és P1, P2, ..., PN un. formális paraméterek, melyek szintén tetszőleges azonositók lehetnek. A SUBROUTINE utasitás előfordulhat akár formális paraméter listával, akár anélkül; a FUNCTION csak formális paraméter

listával, a MASTER csak paraméterlista nélkül. A TPA 4K -FORTRAN-ban a formális paraméterek száma nem haladhatja meg a hetet.

A FORTRAN szegmensek végét az END utasitás jelzi. Ez az utasitás nem törhető el több sorba. Alakja:

```
END
```

Egy TPA 4K-FORTRAN program egy vagy több szegmensből állhat. Minden végrehajtható FORTRAN program kell, hogy tartalmazzon egy és csak egy MASTER szegmenst, tartalmazhat tetszőleges számu SUBROUTINE és FUNCTION szegmenst. Az egy szegmenses végrehajtható programok mindig egy MASTER szegmensből állnak.

Példa egy három szegmensből álló FORTRAN program szerkezetére:

```
MASTER PROG
...
END

SUBROUTINE RUT1(X, Y, Z)
...
END

FUNCTION F(A, B)
...
END
```

A MASTER SZEGMENS NEVE PROG, a RUT1 nevü SUBROUTINE szegmensnek három formális paramétere van: X, Y és Z. Az F ne-

vü FUNCTION szegmensnek két formális paramétere van: A és B.

A formális paraméterek skaláris változók vagy tömbök lehetnek.

A program szegmentálás alapvető tulajdonsága, hogy minden egyes szegmens önálló, és a többi szegmenstől független jelölésrendszerrel bir. Ez azt jelenti, hogy az összes azonositó érvénye egyetlen szegmensre terjed, ha ugyanaz az azonositó több szegmensben is előfordul, akkor a hozzátartozó skaláris változók, illetve tömbök szegmensenként különbözők. Tehát ugyanazt az azonositót különböző szegmensekben különböző célokra használhatjuk. Ez alól kivételek a szegmensnevek, amelyek nem használhatók egyetlen szegmensben sem változók vagy tömbök azonositására.

Minden egyes FORTRAN szegmens forditása az összes többi szegmens forditásától függetlenül megy végbe. Az egyes szegmensforditások elvégzésekor a forditóprogram mindig csak azt az információt használja fel, amely az adott szegmensben rendelkezésre áll. A különálló á/b szegmenseknek teljes programmá való összeszerkesztését az ÁBL program végzi.

5.2 A FORTRAN program végrehajtásának menete. A CALL és RETURN utasitások.

Minden végrehajtható FORTRAN program a MASTER szegmens első végrehajtható FORTRAN utasitásával indul, és

addig folytatódik, amig a végrehajtás során el nem érkezik egy STOP vagy PAUSE utasitásra, vagy amig valamely okból hibajelzés nem keletkezik, amely leállitja a program futását.

A vezérlés a MASTER szegmensnél marad, amig egy ebben előforduló CALL utasitással vagy függvényhivatkozással a vezérlést egy másik szegmensnek át nem adjuk. A CALL utasitás alakja:

CALL SNEV(A1, A2, ..., AN)

ahol SNEV az aktivizálni kivánt SUBROUTINE szegmens neve, és A1, A2, ..., AN az un. aktuális paraméterlista. Elemeinek száma és tipusa meg kell, hogy egyezzék a SNEV nevü szegmens formális paramétereinek tipusával és számával. Az aktuális paraméterek lehetnek:

a/ skaláris vagy indexes változók, konstansok, vagy a megfelelő formális paraméterrel egyező tipusu aritmetikai kifejezések, amennyiben a formális paraméter skaláris változó azonositója:

b/ tömbazonositók, amennyiben a megfelelő formális paraméter tömbazonositó.

A FUNCTION szegmensek a standard függvényekhez hasonlóan, olymódon aktivizálódnak, hogy nevüket és aktuális paramétereiket beirjuk egy aritmetikai kifejezésbe.

A CALL utasitás hatására a vezérlés átadódik a SNEV szegmens első végrehajtható utasitására. Ettől kezdve a program rendre végrehajtja az SNEV szegmens utasitásait, mindaddig, mig egy STOP vagy PAUSE utasitásra, vagy egy

RETURN utasitásra nem kerül sor. A RETURN hatására a vezérlés visszatér abba a szegmensbe, amely az adott SNEV szegmenst aktivizálta, és most már ismét ennek utasitásai hajtódnak végre, a CALL-t követő első végrehajtható utasitástól kezdve. A RETURN utasitás alakja:

RETURN

A szegmens, amelyet egy CALL utasitás segitségével aktivizáltunk, maga is tartalmazhat ujabb CALL utasitásokat, vagy függvény hivatkozásokat. Ilyen esetben a vezérlés további szegmensekbe is átléphet, tetszőleges mélységben. A RETURN mindig az "utolsó előtti" szegmensbe jelent visszatérést, vagyis abba a szegmensbe, amelyből a vezérlés az adott szegmensbe átlépett.

Ugyanazt a SUBROUTINE /vagy FUNCTION/ szegmenst ugyanannak az algoritmusnak különböző változókon és tömbökön történő végrehajtására is felhasználhatjuk.

A FUNCTION szegmensek abban különböznek a SUBROUTINE szegmensektől, hogy ezek egy függvényértéket állitanak elő, vagyis azonositójukhoz egy - az aktuális paraméterektől függő - számérték tartozik. A FUNCTION szegmenseket függvényhivatkozások segitségével aktivizáljuk, ami annyit jelent, hogy a függvény azonositóját az aktuális paraméterekkel együtt beirjuk egy aritmetikai kifejezésbe.

A FUNCTION szegmensek olymódon határozzák meg a hozzájuk tartozó függvényértéket, hogy a FUNCTION szegmens azonositója a szegmensen belül egy vagy több különböző helyen mint skaláris változó fordul elő, és ezen előfordulások között olyan is van, amely értéket ad ennek a változó-

nak. Ezen értékadások közül az utolsó eredménye határozza meg a függvényértéket. A függvényérték tipusát a függvény azonositója határozza meg, ha a függvény azonositója az I, J, K, L, M vagy N betük valamelyikével kezdődik, akkor a függvényérték egész tipusu, egyébként valós.

Példa. Az alábbi FUNCTION szegmens a 3.4 pontban tárgyalt algoritmus szerint a ch x függvény sorfejtését számitja ki.

```
      FUNCTION CH X
      CH = 1.0
      Z = X**2
      T = 1.0
      DO 16 K = 2,39,2
      AK = FLOAT(K)
      T = T*Z/(AK*(AK - 1.0))
      CH = CH + T
      IF (CH/Y - 4.0E -9) 20, 20, 16
   16 CONTINUE
      WRITE (4,15)
      PAUSE 88
   20 RETURN
   15 FORMAT(5X, 8HCH ERROR)
      END
```

A CH azonositó - a függvény neve - ugyanugy vesz részt a program működésében, mint a 3.4 pont változója, és az utolsó reá vonatkozó értékadás határozza meg a függvény adott X melletti értékét.

Példa a fenti szegmens egy aktivizálására:

CHA = CHA + CH(A**2 + B**2)

A RETURN utasitás hatása, hogy egy SUBROUTINE szegmens RETURN-jének elérése mindig az aktivizáló CALL utasitás utáni első végrehajtható utasitásra jelent visszatérést. A FUNCTION szegmensek RETURN-je ezzel szemben egy félbehagyott aritmetikai kifejezés kiértékelésének folytatását jelenti. Igy pl. az iménti értékadó utasitást csak akkor lehet végrehajtani, ha a CH(A**2 + B**2) érték már rendelkezésre áll.

5.3 Tömbök méreteinek deklarálása. A DIMENSION utasitás.

A FORTAN szegmensben előforduló valamennyi tömböt deklarálni kell. A tömbdeklaráció rögziti a tömb méretét, valamint az indexhatárokat. Indexes változók használatakor ügyelni kell arra, hogy az indexek aktuális értéke a deklarált maximumokat ne lépje tul.

A tömbdeklaráló utasitás alakja:

DIMENSION V_1, V_2, ..., V_n

ahol a V_i-k un. tömbdeklarátorok; formailag olyanok, mint az az indexes változó, amelynek indexei a maximális megengedett értéket veszik fel, vagyis a tömb azonositójából, és utána zárójelben a maximális indexből vagy indexpárból állnak. Például:

DIMENSION A(10,10), X(20), B(2,3)

Ez a deklaráció egy 10x10-es A mátrixot, egy 20 elemi X vektort és egy 2x3-mas B mátrixot deklarál.

Egy szegmensben akárhány DIMENSION utasitás lehet, ezek azonban csak a szegmensnyitó MASTER, SUBROUTINE illetve FUNCTION, és az első COMMON utasitás között, egy csoportban fordulhatnak elő. /Ha COMMON utasitás nincs, akkor helyette az első végrehajtható utasitás értendő./

```
A       SUBROUTINE RUT(A, B)
        DIMENSION A(10, 10)
        DIMENSION B(10, 10)
```

és a

```
        SUBROUTINE RUT(A, B)
        DIMENSION A(10,10), B(10,10)
```

változatok egyenértékűek.

Mint a fenti példa is mutatja, egy SUBROUTINE vagy FUNCTION szegmens formális paraméterei lehetnek tömbök, ezeket is dimenzionálni kell. Fontos kompatibilitási szabály, hogy a formális paraméter tömb elemeinek száma legfeljebb akkora lehet, mint az aktuális paraméter tömb elemeié. A formális paraméter tömbnek "el kell férnie" az aktuális paraméter tömbben. Nem követeljük meg, hogy az aktuális és formális paraméter tömb dimenziószáma megegyezzék, lehet az aktuális paraméter vektor, mig a formális paraméter mátrix, vagy megforditva.

Tömb paraméter "átvétele" az aktuális- és a formális tömb kezdőcimének egyeztetését jelenti.

Példa. Legyen a MASTER szegmensben az A tömb deklarációja:

```
DIMENSION A(56)
```

és szerepeljen a RUT nevü SUBROUTINE szegmensben az alábbi deklaráció:

```
SUBROUTINE RUT(Q)
DIMENSION Q(7,7)
```

Ezen deklarációk mellett a MASTER szegmensben elhelyezett

```
CALL RUT(A)
```

szubrutinhivó utasitás korrekt, mert a formális paraméter tömbnek 49, az aktuális paraméter tömbnek 56 eleme van.

Dimenzionálni nem csak a formális paraméter tömböket, hanem a szegmensben előforduló bármely tömböt kell.

5.4 Közös adatmező. A COMMON utasitás.

A FORTRAN nyelvben lehetőség van közös adatmező deklarálására, vagyis olyan memória terület kijelölésére, amelyhez a program valamennyi szegmensének hozzáférése van.

A TPA 4K-FORTRAN-ban csak egyetlen közös adatmező lehet.

A COMMON utasitás alakja:

```
COMMON A1, A2, ..., AN
```

ahol A1, A2, ..., AN a közös adatmezőhöz tartozó azonositók listája. Ezek lehetnek skaláris változók és tömbök azonositói. Amennyiben tömbazonositók is vannak köztük, ezeket DIMENSION utasitásban kell deklarálni.

A közös adatmező memória területéhez bármely szegmens hozzáférhet, amelyben előfordul a COMMON utasitás. A közös adatmező méretét az egyes COMMON utasitások által meghatározott méretek maximuma adja meg.

Példa. Legyen a MASTER szegmensben szereplő deklarációs rész:

```
DIMENSION A(20), B(50)
COMMON A, B
```

az általa behivott RUT nevü szegmensben pedig

```
DIMENSION A(4,5), B(8,8)
COMMON A, B
```

A MASTER-ben a közös adatmező mérete 70, RUT-ban 84 egység /=280, ill. 336 rekesz/. A tényleges méret 84 egység /=336 rekesz/ lesz.

A COMMON utasitás azt mondja ki, hogy azok a változók, amelyek a különböző szegmensek COMMON listáin szerepelnek, a memóriában fizikailag ugyanarra a területre kerülnek.

Fenti példánkban a MASTER szegmens 20 elemü A vektora ugyanarra a helyre kerül, mint a RUT szegmens 4x5-ös A mátrixa, a MASTER 50 elemü B vektora pedig fizikailag azonos a RUT-beli B mátrix első 50 elemével. A RUT-beli B mátrix utolsó 14 elemének a MASTER szegmensben nincs megfelelője.

A különböző szegmensek a közös adatmezőt eltérő módon is feloszthatják.

A közös adatmezőben szerepelhetnek egész tipusu

változók is; ebben az esetben figyelembe kell venni, hogy egy egész változó nem 4, hanem 2 rekeszt foglal el. A közös adatmező felosztásának meghatározásánál célszerű ügyelni, hogy a fentiekben leirt megfelelkezés ne hozzon létre equivalenciát valós és egész változók között, mert hibákat okozhat.

A COMMON utasitás a szegmensben a DIMENSION utasitás után, és az első végrehajtható utasitás előtt helyezkedhet el /ha DIMENSION nincs a szegmensben, akkor közvetlenül a szegmensnyitó utasitás után/. Egy szegmensben több COMMON utasitás is lehet, de ezeknek egy csoportban kell lenniük. Több COMMON utasitás együttes hatása olyan, mintha listáikat az előfordulás sorrendjében egymás mellé állitottuk volna.

Például a

```
COMMON A, B
COMMON C, D
```

utasitáspár egyenértékü a következővel:

```
COMMON A, B, C, D
```

A COMMON utasitás listáján nem fordulhatnak elő formális paraméterek.

6. A TPA 4K-FORTRAN gépi reprezentáció

6.1 A TPA 4K-FORTRAN rendszer használata

A TPA 4K-FORTRAN felhasználása az alábbi lépések-

ben történik:

1. A program megirása a TPA 4K-FORTRAN szabályainak megfelelően;
2. A program lelyukasztása 8 csatornás lyukszalagra /ASCII 7 bites, páros paritású kódban/;
3. A forrásprogram szintaktikus helyességének ellenőrzése, a hibák kijavitása;
4. A forrásprogram leforditása á/b formátumra;
5. Memóriatérkép készités és betöltés;
6. Futtatás.

A TPA 4K-FORTRAN forditóprogram input szalagja 8 csatornás, ASCII kódban lyukasztott páros paritásu lyukszalag. Az ASR 33 teletype-ok némelyike olyan rendszerü, hogy a 8-ik /paritás/ csatornára mindig 1-est lyukaszt. Ebben az esetben páros paritásu szalagot a PUNCH nevü géptermi programmal lyukaszthatunk, ill. Telex, Optima 527, Consul 253 lyukasztógépeken készitett programszalagok esetén a PKONV nevü konvertáló programmal állithatjuk elő a páros paritásu ASCII szalagot.

A forrásnyelvi programok javitására a Forrásnyelvi Javitó Program /FJP/ alkalmazható. Ennek segitségével a forrásprogramból tetszőleges számu sort elhagyhatunk és beszurhatunk.

A lelyukasztott vagy kijavitott program szintaktikus helyességét a TPA 4K-FORTRAN Szintaktikus Analizátor segitségével ellenőrizzük. Ez a program nem készit forditást, hanem csak kijelzi a szintaktikus /és egyes szemantikai/ hibákat. A hibajelzések megjelölik a sort és a karakter

pozíciót, amelyben a hiba előfordult. Kívánság szerint a forrásprogramról teljes, vagy részleges lista készíthető.

A forrásprogram tényleges fordítását a TPA 4K-FORTRAN fordítóprogram végzi. A fordítás során a FORTRAN forrásnyelvü szegmensekről á/b formátumu szalag készül.

Az elkészült á/b formátumu célprogram szalagokról az Áthelyezhető Bináris Loader /ÁBL/ segitségével memóriatérképet kell készíteni. Ehhez egyrészt a FORTRAN forrásnyelvü á/b szalagok, másrészt a programba beépithető egyéb szegmensek á/b szalagjai szükségesek, beleértve a TPA Programkönyvtárból beépitendő szegmensek szalagját is. - Az áthelyezhető bináris szalagok bevitele után még a nemáthelyezhető szolgálati rutinokat is be kell tölteni a memóriába a FORTRAN programok futtatásához. A betöltés után következhet a futtatás, amely FORTRAN programok esetén egységes cimről /1600_8-ról/ indul.

Amennyiben a futtatás során olyan helyzet áll elő, amelyben a számitás nem folytatható, a programok hibajelzéseket adnak.

A programoknak kapcsoló szerinti elágaztatására, valamint a program kipróbálások megkönnyitése céljából a TPA programkönyvtárba négy speciális szubrutin /TRACE, ITRACE, KPR, SSWITCH/ van beépítve.

6.2 Kapcsoló szerinti elágaztatás és nyomkövetés a célprogramban. A TRACE, ITRACE, SSWITCH és KPR szubrutinok.

A TRACE függvénynek két argumentuma van, az első valós, a második egész, függvényértéke pedig valós tipusu. Az ITRACE függvény mindkét argumentuma egész, és függvényértéke is egész. A függvényérték mindkét függvénynél megegyezik az első argumentummal, de a rutin végrehajtása során ez az érték, valamint a második paraméter aktuális értéke még ki is nyomtatódik, ha a 11-es kapcsoló be van kapcsolva.

Példa:

```
A = X + TRACE (U*V, 5)
```

a program szempontjából egyenértékü az

```
A = X + U*V
```

FORTRAN utasitással, de ha végrehajtásakor a 11-es kapcsoló be van kapcsolva, az U*V érték, valamint az 5 szám még ki is nyomtatódik. A nyomtatási kép a következő:

```
-0.77908235E 12  TP:       5
```

A TRACE rutin második paramétere az ellenőrző kinyomtatások megkülönböztetését segiti elő; egész konstanssal, változóval, vagy aritmetikai kifejezéssel adható meg.

Az ITRACE függvény hasonlóan müködik, de mindkét argumentuma egész. Egész tipusu mennyiségek értékének nyomonkövetésére használhatjuk. Például, a

```
K = ITRACE(J - K, L)
```

utasitás egyenértékü a

K = J - K

utasitással, de J - K és L aktuális értéke még ki is nyomtatódik.

-120345 TP: 112

A nyomkövető függvények hivása - az egyébként megengedett ötszörös mélység határain belül - egymásba is skatulyázható.

Például:

A = TRACE (Q**2 - TRACE(SIN(X),2),3)

amelynek hatására először SIN X értéke nyomtatódik ki 2-es trace sorszámmal, majd Q 2 - SIN X értéke 3-as trace sorszámmal.

Indexkifejezésekbe az ITRACE-t nem szabad beépiteni.

A kapcsolóregiszter tartalmának érzékelésére két könyvtári szubrutin áll rendelkezésünkre: ezek a KPR és az SSWITCH rutinok. A KPR függvényeljárás hatására a vezérlőpultról beolvasódik a kapcsolóregiszter tartalma, és az argumentumtól függetlenül ez lesz KPR függvényértéke. Ez az érték mint kétszavas fixpontos egész, a programban felhasználható. Értéke mindig pozitiv, és 4095-nél nem nagyobb.

Példa:

K = KPR(K)

utasitás hatására a vezérlőpultról beolvasódik a kapcsolóregiszter tartalma, és egész számként elhelyeződik K-ban. Ennek a függvénynek az argumentuma /aktuális paramétere/ teljesen közömbös, híváskor értékkel sem kell birnia, mert egyetlen szerepe az, hogy formálisan kielégitse azt a sza-

bályt, amely szerint függvény hivatkozásnak mindig kell, hogy legyen aktuális paramétere.

A másik standard könyvtári eljárás, amely rendelkezésünkre áll, az SSWITCH szubrutin. Ennek segitségével a kapcsolóregiszter egy adott bitjének állapotát érzékelhetjük. A rutin hivása:

CALL SSWITCH(I,J)

ahol I egész változó vagy egész kifejezés, J pedig egész változó. A szubrutinhivás hatására J értéke 1-et vagy 2-t vesz fel értékként, attól függően, hogy az I-edik kapcsoló /I = 0, 1, ..., 11/ be van-e kapcsolva vagy sem.

A SSWITCH rutin bármelyik kapcsolóra hivatkozhat, azonban a felhasználói programok elágaztatására a 0-s és 11-es kapcsolókat nem használhatjuk. Az utóbbi a TRACE információ kinyomtatását szabályozza, az előbbi pedig a rendszer későbbi továbbfejlesztése céljából van lefoglalva.

A TRACE, ITRACE, KPR és SSWITCH ugyanazon szegmens különböző belépési pontjai, vagyis ha a program akár csak egyet is hiv ezek közül, a célprogramba valamennyi beépül. A szegmens terjedelme 1 lap, vagyis ennyivel csökken a program helyfoglalása, ha a nyomkövetést elhagyjuk, feltéve, hogy a program nem hivja a kapcsoló érzékelő rutinokat.

7. Hibajelzések

7.1 A hibajelzések alakja analizisnél:

ERROR xxx STATEMENT LABEL yyyy + zz LINE uu CHAR vv

ahol:

xxx hibakód;

yyyy a hibás utasitás előtt utoljára észlelt cimke

zz az utolsó cimkézett utasitás óta analizált utasitások száma

uu a hibás sor sorszáma az adott utasitáson belül /a számozás 0-val kezdődik/;

vv az a karakterpozició, amelyen az analizátor a hibát észlelte.

Hibakód táblázat

Kód	Magyarázat
1	Az utasitás nem teljes.
2	Két müveleti jel áll egymás mellett.
3	Bizonyos jelek meg nem engedett sorrendben követik egymást /konstans után kezdőzárójel, végzárójel után kezdőzárójel stb/.
4	Egyenlőségjel követ olyan jelet, amely után nem állhat /pl. konstans, vagy müveleti jel után/.
5	Vessző követ olyan jelet, amely után nem állhat /pl. müveleti jel, egyenlőségjel stb után/.

Kód	Magyarázat
6	Végzárójel követ olyan jelet, amely után nem állhat /pl. müveleti jel után/.
7	Unáris szorzás, osztás vagy hatványozás.
8	Értékadó utasitás nem megfelelően kezdődik /pl. konstans van a baloldalán/.
9	Az értékadó utasitás hiányos, pl. hiányzik az egyenlőségjel.
10	Tömbazonositó fordul elő olyan helyen, ahol ez illegális /pl. aritmetikai művelet operandusaként/.
11	Vegyes tipusu aritmetika.
12	A függvényhivatkozások ötnél nagyobb mélységben vannak egymásba skatulyázva.
13	Zárójelszint hiba aritmetikai kifejezésben, vagy I/O listán, vagy vessző fordul elő nem megengedett zárójelszinten.
14	Egyenlőségjel meg nem engedett módon történő felhasználása /egy utasitásban több egyenlőségjel; egyenlőségjel nem zérus zárójelszinten, vagy olyan helyen, ahol nem állhat. pl: A+B=C/.
15	Az indexkifejezésben tömbnév vagy nem egész tipusu mennyiség fordul elő.
16	Az indexkifejezés alakja nem szabványos /pl. I + J/.
17	Az indexkifejezés alakja nem szabványos /pl. 3+I J/.
18-19	Jelenleg kihasználatlan.
20	Az I/O lista nem megengedett jellel kezdődik, vagy folytatódik.

Kód	Magyarázat
21	Az implicit DO ciklust nem jobbzárójel zárja le.
22	Explicit vagy implicit DO utasitásban a ciklusparaméterek valamelyike nem egész tipusu, dimenzionált vagy pedig a ciklusváltozó nem változó /pl. konstans/.
23	Jelenleg kihasználatlan.
24	Függvényhivatkozás vagy CALL utasitás hibás alkalmazása /ugyanazt a külső azonositót különböző alkalmakkor eltérő számu paraméterrel hivjuk, vagy az aktuális paraméterek száma meghaladja a hetet/.
25	Az explicit DO ciklusok egymásba skatulyázásának mélysége meghaladja a tizet.
26	Explicit DO utasitás ciklusváltozója után nem egyenlőségjel áll, vagy az utolsó ciklusparaméter után az utasitás nem ér véget.
27	A ciklusparaméterek nem vesszővel vannak elválasztva vagy számuk meghaladja a hármat.
28	A DO ciklusok hatásköre átlapolódik.
29-30	Jelenleg kihasználatlan.
31	Az analizátor müködését valami megzavarta. Ennek a hibajelzésnek nem szabad előfordulnia.
32	Formális paraméter listán, DIMENSION vagy COMMON utasitásban azonositó helyett valami más áll.
33	Zérus vagy értelmetlen tömbméret deklarálása /pl. betü áll a tömbméret helyén/.

Kód	Magyarázat
48	Hiányzik a szegmenskezdő utasitás.
49	Az analizátor müködését valami megzavarta. Ennek a hibajelzésnek nem szabad előfordulnia.
50	Az adott szegmens tulságosan hosszu vagy bonyolult az analizátor és a fordító számára /táblázat tulcsordulás/.
51	Deklarativ utasitáson cimke van.
52	Deklarativ utasitás nem megengedett helyen.
53	Ugyanaz a formális paraméter a listán többször fordul elő.
54	Illegális karakter.
55	Az utasitás tulságosan hosszu, ezért a fordító és az analizátor nem tudja egy utasitásként feldolgozni.
56	Az IF utasitás cimkerésze hibás /nincs megadva három cimke, 0-s cimke fordul elő, vessző helyett más elválasztójel stb/.
57	Szintaktikus hiba valós konstans egész- vagy törtrészében.
58	Szintaktikus hiba valós konstans kitevőrészében.
59	Szintaktikus hiba FORMAT utasitásban.
60	FORMAT utasitásban tul mély zárójelezés /ötszörösnél nagyobb mélység/.
61	A FORMAT listán szereplő valamelyik szám a megengedettnél nagyobb vagy kisebb.
62	Hibás COMMENT-sor, vagy a HT vagy a 6 db betüköz karakter hiányzik az utasitás elejéről.

Kód	Magyarázat
63	Folytatósor cimkemezeje nem üres.
64	Tulságosan nagy cimke, vagy dimenzió érték /≥ 4096/.
65	Tulságosan sok betüköz az utasitás előtt.
66	HT karakter fordul elő nem megengedett mélyen.
67-89	Jelenleg kihasználatlan.
90	Ismeretlen cimke /a szegmens végén kerül kijelzésre/.
91	Lezáratlan DO ciklus /a szegmens végén kerül kijelzésre/.
92	A FORMAT utasitásnak nincs cimkéje.
93	Jelenleg kihasználatlan.
94	A DO utasitásban szereplő cimke megelőzi a DO utasitást.
95	A DO utasitás zárócimkéje olyan utasitáson van, amely lezáró utasitásként nem fordulhat elő /másik DO, RETURN, IF, GO TO, STOP, PAUSE, FORMAT/.
96	A cimke használata nem konzisztens /pl. mind GO TO mind I/O utasitásban előfordul/.
97	A cimke többszörösen fordul elő.
98	Vezérlésátadás DO ciklus hatáskörébe.
99	Az analizátor müködését valami megzavarta. Ennek a hibajelzésnek nem szabad előfordulnia.
128	Az azonositó első négy karaktere megegyezik valamelyik kulcsszó első négy karakterével.
192	Az utasitás tulnyulik a 72. karakterpozición.

Megjegyzések:

1. Az 55-ös hibakóddal kapcsolatban: a FORTRAN utasitások általában maximálisan 63 "elemből" állhatnak, ahol egy elemet alkot egy azonositó, egy konstans, vagy egy müveleti, illetve egyéb jel. A hibakód akkor képződik, ha az ilyen elemek száma meghaladja a 63-at.
2. A 61-es hibakóddal kapcsolatban: a konverziós specifikációkban a maximális megengedett mezőszélesség w=31, a törtrész jegyeinek maximális száma d=15, és az ismétlési tényező, valamint a H és X specifikációk mezőszélességének maximális megengedett értéke 512.
3. A 65-ös hibakóddal kapcsolatban: az analizátor csak listázásos üzemmódban adja ezt a hibajelzést akkor, ha a FORTRAN utasitás első 36 karaktere között nem talált értékes /nem betüköz/ karaktert.
4. A 128-as és 192-es hibakódokkal kapcsolatban: ezek a hibajelzések figyelmeztető jellegüek; az adott FORTRAN utasitás elemzése tovább folytatódik, de a forditóprogram munkáját ezek a hibák megzavarhatják.

7.2 Hibajelzések forditásnál

Szintaktikus hiba észlelése esetén a forditóprogram megáll. Az accumulátor tartalma egy hibakód:

Accumulátor:	A hiba oka lehet:
0001	Az utasitás tul bonyolult, vagy meg nem engedett unáris müvelet szerepel benne /pl. szorzásjellel kezdődő aritmetikai kifejezés/.
0003	Illegális vagy felismerhetetlen utasitás, vagy a szimbólumok sorrendje nem megengedett /pl. két egymás melletti egyenlőségjel/.
0004	Táblázat hiba: rendszerint egyéb hibák következménye.
0005	Indexes változó kettőnél több indexszel.
0007	Valami megzavarta a compiler müködését /rendszerint egyéb hibák következménye/.
0022	Ismeretlen cimke, vagy az END eléréséig lezáratlan DO utasitás.
3574	PAUSE, CONTINUE vagy RETURN utasitás után folytatósor.
4511 vagy	
1272	Hiba a szegmensnyitó utasitásban.
7300	FORMAT utasitás cimke nélkül.
7301	Az utasitás tul hosszu.

7.3 Hibajelzések futtatás közben

A futtatás közben jelentkező hibajelzések alakja:

HALTED: XX

Futtatás közben jelentkező hibakódok:

Kód	Magyarázat
A1	Indextulcsordulás: valamelyik indexkifejezés értéke negativ, zérus vagy meghaladja a deklarátlt indexhatárokat.
A2	Egész tulcsordulás: valamelyik egész mennyiség tullépte a megengedett maximális 2^{23}-1 értéket.
A3	Lebegőpontos tulcsordulás veszélye áll fenn. Valamelyik mennyiség bináris exponense 2048-nál nagyobb. Az igy kiszámitott mennyiségek már nem kinyomtathatók.
A4	A célprogram speciális munkamezeje /stackje/ betelt. A FORTRAN program tul bonyolult, jelenlegi formájában nem futtatható.
S1	Az IFIX standard könyvtári függvény a megengedettnél nagyobb egész értéket állit elő.
S2	Valós, negativ alap hatványozása valós kitevőre.
F1	A FORMAT listán nincs konverziós specifikáció, holott az I/O lista nem üres.
F2	A specifikáció nem felel meg az output listaelem tipusának /pl. egész mennyiség kiiratása E specifikációval/.
F3	Az adott mezőszélesség mellett az output listaelem nem iratható ki.
F4	A beolvasott adat nem felel meg a specifikációnak, vagy a specifikáció nem felel meg az input listaelem tipusának.

Kód	Magyarázat
SQ	Négyzetgyökvonás negativ argumentummal.
EX	Az EXP standard függvény argumentuma olyan nagy, hogy az eredmény tulcsordult mennyiség volna.
LN	Az ALOG standard könyvtári függvény hivása negativ argumentummal.
IM	Illegális makroutasitás /csak vegyes nyelvü programokban fordulhat elő/.
SW	Az SSWITCH standard könyvtári rutin hivása negativ, vagy 11-nél nagyobb argumentummal.

TÁRGYMUTATÓ

Készült a K.K.V.M.F. Sokszorosító Üzem. 1971.