Álló Géza: ***Az Autokód programnyelv***

Az **autokód** gyűjtőnév az 1950-es évtizedben kialakult „egyszerűsített programozási rendszerek” családját jelenti, amelyből később kifejlődtek a magasabb szintű programozási nyelvek. Minden autokód *emberközeli gépi kód* nyelv, emiatt erősen gépfüggő, vagyis autokódban a programot gépi utasításonként kell megírni, olyan utasításokra lebontva, amelyeket egyszerű gépi kód utasításokra lehet lefordítani.

Bármely magasabb szintű nyelven írt programot az adott gépen végrehajtható gépi utasításokra egy *fordító*program alakít át, amelynek jól kell ismernie mind a használt a programnyelv szintaxisát, mind az adott gép gépi kódját. A futáskész lefordított programot pedig automatikusan kell elhelyeznie a főtárban.

Az alábbiakban vázlatosan ismertetjük – az Elliott 803 számítógépen implementált – Mark 3 Autokód-nyelv szintaxisát.[1]

Deklarációk

A programot deklarációkkal kell kezdeni.
Mindenekelőtt a fix-és a lebegőpontos *változókat* kell deklarálni, az angol ábécé egy-egy nagybetűjével; például

SETS I, J, K, L(4) 3 fixpontos (egész számot tartalmazó) változó és egy 4-elemű Fixpontos tömb, illetve

SETV X, Y, Z 3 lebegőpontos (valós számot tartalmazó) változó deklarálása.

A programban pozitív és negatív egész (fixpontos) és valós (lebegőpontos) konstans számokat is lehet használni, ezeket nem kell deklarálni.

Az egészszámoknak nem lehet törtrészük, a lebegőpontos számok törtrészét pedig tizedesponttal választjuk el!

Például: 12 fixpontos egészszám; 12.0 lebegőpontos egészszám.

Az Elliott803 gépen az egészszámok értékkészlete $\{-2^{38} \div 2^{38} = 274\ 877\ 906\ 944\}$; a lebegőpontosaké $\{-2{,}94*10^{-39} \div 1{,}70*10^{38}\}$ volt. Az értékhatárok túllépése túlcsordulás (overflow) hibajelzést váltott ki, amit a vezérlőpulton egy lámpa jelzett.
Fixpontos túlcsordulás (fixpoint overflow) esetén a program tovább futott, de az eredmény természetesen hibás lett. A hibajelzés lekérdezhető volt, így esetleg meg lehetett keresni a hibát okozó utasítást.
A felső lebegőpontos határ túllépése (floating point overflow) esetén a program leállt; egy segédprogram („post mortem rutin”) segítségével meg lehetett keresni a hibát okozó utasítást. Az alsó lebegőpontos határnál kisebb eredményt a gép automatikusan nullázta.

Az indexelés mindig 0-val kezdődik, és az index nem lehet nagyobb a tömbméret-1-nél! Sajnos, az indexhatár túllépését a fordító nem ellenőrzi, így könnyen tönkre lehet tenni a futó programot!

A egyszerű (egytagú) indexet – akár változónév akár konstans formában –közvetlenül a tömbnév mellé lehet írni, az összetett indexet zárójelbe kell tenni; például: L0 ≡ L, L3, LI ($0 \leq I \leq 3$), L(I+K)

Hozzá kell rendelni továbbá, a programhoz a futáshoz szükséges *külső eszközöket*; például:

READER <azn> az adatbeviteli eszköz (lyukszalag olvasó),

PUNCH <azn> a kiíró eszköz (lyukszalag lyukasztó, sornyomtató, konzol írógép),

FILM <azn> a használni kívánt mágnesszalagos egység azonosító kódszáma (kötelezően természetes szám).

Csak a konfigurációban létező eszköz kódszámát szabad megadni!

Fel kell még sorolni a programban használni kívánt programkönyvtári függvényeket (lásd alább), például:

SETF EXP, LOG, TRIG (→ SIN, COS, TAN, ARCTAN)

és meg kell adni a használni kívánt legnagyobb címke-, illetve szubrutinszámot (lásd alább), a

SETR n kulcsszóval.

Ezek után majd az utasítások címrészében a deklarált alfanumerikus változó- vagy tömbnevek állhatnak. A deklarációkat a fordító nyilvántartja, és lefoglalja számukra a megfelelő hosszúságú tárhelyet.

Értékadó utasítások

Ezeket a matematikában szokásos algebrai műveleti jelekkel kell megadni. Példák:

I = I+J Számítsd ki az I+J összeget és tárold I-ben

A program először mindig a jobboldali kifejezést értékeli ki, miközben az előforduló változók értéke nem változik; az eredménnyel pedig felülírja a baloldali változó értékét, ennek eredeti tartalma elvész!

K=-8*I Tárold K-ban az I -8-szorosát

K=3*K 3-szorozd meg K értékét

J=I:K Oszd el I-t K-val (egész osztás) majd tárold a hányadost J-ben

(Például: 13:5 = 2)

Z=Z/Y Oszd el Z-t Y-nal (lebegőpontos osztás!), majd tárold a hányadost Z-ben

(Például: 13.0/5 = 2.6)

Z=I*X Számítsd ki X I-szeresét (az eredmény lebegőpontos!) és tárold Z-ben

Az utasításokban *függvényhívások* is szerepelhetnek; néhány példa a leggyakoribb könyvtári (a számítógéppel együtt szállított) függvények közül:

Y=EXP(X)	10^X hatvány
Y=LOG(X)	X értékének 10 alapú logaritmusa
Y=SIN(X)	X (radiánban értendő értékének) szinusza
Y=COS(X)	X (radiánban értendő értékének) koszinusza
Y=TAN(X)	X (radiánban értendő értékének) tangense
X=ARCTAN(Y)	Y tangensértéknek megfelelő szög, radiánban ($\pm \pi/2$)
Y=SQRT(X)	X négyzetgyöke
I=INT(X)	X értékének egész része (kerekítés nélkül)
Y=FRAC(X)	X értékének tört része; X = INT(X) + FRAC(X)
Y=MOD(X)	X abszolút értéke
X=STAND(I)	I fixpontos értéke lebegőpontossá alakítva

Vezérlő utasítások

A számítógép a programutasításokat megadásuk sorendjében, *szekvenciálisan* hajtja végre. Ettől eltérő végrehajtási sorrendet vezérlő utasításokkal lehet megadni; ilyenkor a végrehajtandó utasítás helyét *címkéje* határozza meg. A címke a sor elején álló egyedi természetes szám kettősponttal lezárva; a továbbiakban ezt n: jelöli.

Feltétel nélküli ugrás

JUMP @n

A program az n: címkéjű utasítás végrehajtásával, utána szekvenciálisan folytatódik

Feltételes ugrás

JUMP IF (feltétel)@n

Ha a feltétel *teljesül*, a program az n: címkéjű, ha nem teljesül a soron következő utasítás végrehajtásával, utána szekvenciálisan folytatódik

JUMP UNLESS (feltétel)@n

Ha a feltétel *nem teljesül*, program az n: címkéjű, ha teljesül a soron következő utasítás végrehajtásával, utána szekvenciálisan folytatódik

A *feltétel* a $ → kisebb, = → egyenlő, % → nagyobb relációjelekkel megadott reláció; ennek bal és jobboldalán mindazon kifejezések állhatnak, amelyek értékadó utasításokban szerepelhetnek; például:

IF (X$I)	értsd: „ha X kisebb mint I”,
IF (XI+3%2*K)	értsd: „ ha XI+3 nagyobb K kétszeresénél”,
IF (I*X$Y)	értsd: „ha I*X kisebb Y-nál” stb.

Ciklusképzés

CYCLE I=J:K:L	*Léptetős ciklus*; az I, J, K, L változót előzőleg deklarálni kell
<ciklustörzs>	J a kezdőérték, K a lépésérték, L a végérték
REPEAT I	A <ciklustörzs>-et alkotó utasítások ismételten, rendre I = J, J+K, J+2*K, J+3*K , … , J+L értékkel hajtódnak végre

Ha I értéke nem „talál bele” J+L-be, végtelen ciklus keletkezik!

CYCLE X=Y,Z,…	*Felsorolós ciklus;* az X, Y, Z, … változókat előzőleg deklarálni kell
<ciklustörzs>	A <ciklustörzs>-et alkotó utasításokban X értéke rendre Y, Z, … lesz
REPEAT X	

VARY I=J:K:L	*Számlálós ciklus*; az I, J, K, L változót előzőleg deklarálni kell
<ciklustörzs>	J a kezdőérték, K a lépésérték, L (kötelezően >0) az ismétlések száma
REPEAT I	A <ciklustörzs>-et alkotó utasításokban I értéke rendre J, J+K, J+2*K, J+3*K, … , J+(L-1)*K lesz

A ciklustörzsben tetszőleges értékadó utasítások, elágazások, szubrutin-hívások és újabb ciklusok állhatnak; DE

Ciklusok legfeljebb 5-szörös mélységben ágyazhatók egymásba!

Szubrutin írása

Ha összetettebb függvényre gyakrabban van szükség, vagy egy utasítássorozatot többször kell végrehajtani a program különböző pontjain, érdemes szubrutint írni a kiszámítására. A szubrutin a főprogramtól leválasztott (szekvenciálisan nem elérhető) programrészlet, amelyet egy címke (n:) azonosít; meghívása a SUBR n utasítással történik. Lefutása után, az EXIT utasítás hatására a főprogram a hívó sor után folytatódik.

Például egy *maradékképző* szubrutin (maradékképzés csak egészszámokra értelmezett):

n: J=I:K	A szubrutin azonosító címkéje, tetszőleges természetes szám; ügyelni
K = J*K	kell, hogy ne legyen nagyobb a deklarált maximumnál
I = I-K	(Például: I=13, J=5 esetén K=2, J*K=10, I=3)
EXIT	

Szubrutinok legfeljebb 6-szooros mélységben ágyazhatók egymásba!

Külső eszközök kezelése

Mind a beolvasás, mind a kiírás a deklarációban, vagy a be/kimeneti utasítás előtt kijelölt (aktuális) eszközön történik; a megadott adatnak illeszkednie kell a bementi változó típusához.

Beolvasás

READ I — Olvasd be a következő egészszámot és tárold I-ben, fixpontosan

READ X — Olvasd be a következő valós számot és tárold X-ben, lebegőpontosan

INPUT I — Olvasd be a soron következő karaktert, és tárold I-ben

Az Elliott 803 gépen egészszámokat legfeljebb 12, valós számokat legfeljebb 9 értékes jegyre lehetett megadni.

Kiírás

PRINT I,J — Nyomtasd ki az I változó tartalmát, J jegyre (J>0)
Ha az értékes jegyek száma kevesebb, kezdő szóközökkel egészítsd ki

PRINT X,J — Nyomtasd ki az X változó tartalmát, J jegyre (J>0)
Táblázatban a számok egymás alatt lesznek, de a tizedespontok nem

PRINT X,I:J — Nyomtasd ki az X változó tartalmát, I egész és J tört jegyre (I, J >0)

PRINT X,J/ — Nyomtasd ki az X változó tartalmát J jegyre (J>0), exponenciális formában, azaz 1 egész és J-1 törtjegyre, szorozva 10 megadott kitevőjű hatványával. Például a nyomtatási kép:

X=-π és J=6 esetén: -3.14159@00__
X=2018. és J =6 esetén: _2.01800@03__

A + előjelet a gép nem írja ki, de helyét kihagyja; minden számkiírást két szóköz követ, és nincs soremelés.
Ha a kinyomtatandó érték nem fér el a megadott tartomány(ok)ban, a nyomtatás automatikusan új sorban ? jel után történik, az adott gépen lehető legtöbb értékes jegyre.

Az Elliott 803 gépen decimális formában egy egészszám legfeljebb 12, egy lebegőpontos legfeljebb 9 értékes jegyet tartalmazhatott, ennek megfelelően a kényszernyomtatás formátuma ?I,12, illetve ?X,9/; a lebegőpontos számok kerekítettek és 8 jegyre pontosak voltak.

OUTPUT I — Nyomtasd ki az I változó 5 utolsó bitjének értékét karakterként

SPACES I — Hagyj ki I számú szóközt az előző nyomtatatás után

LINE, LINES I — Emelj 1, illetve I számú sort az előző nyomtatás után

TITLE <tetszőleges szöveg>
Írd ki a <> között levő szöveget

Háttértár használat

Ezek az utasítások erősen gépfüggők, az alábbiak Elliott 803-ra érvényesek: a felírás, illetve a visszaolvasás 64 gépi szavas (39 bpW) blokkokban történt; I és J helyén konstans is állhatott.

FILM(I) ALLOW WRITE
FILM(I) PREVENT WRITE
Engedélyezi, illetve megtiltja a felírást az I-edik (>0) mágnesszalag-egységen

FILM(I) SEARCH J — Az I-edik mágnesszalag-egység J-edik (>0) blokkjára áll

FILM(I) BLOCK NUMBER TO J
Beolvassa az I-edik mágnesszalag-egység aktuális (soron következő) blokkjának címét a főtár J című szavába

FILM(I) TO J — Beolvassa az I-edik mágnesszalag-egység aktuális blokkját a főtárba, a J című szótól kezdve

FILM(I) FROM J — Kiír 64 gépi szót a főtár J című szavától kezdve az I-edik mágnesszalag aktuális blokkjába

Minden írási / olvasási művelet után automatikusan a soron következő blokk lesz aktuális!

JUMP IF FILM(I) IS SEARCHING@K

JUMP UNLESS FILM(I) IS SEARCHING@K

Az L címkére ugrik, ha az I-edik mágnesszalag kereső állapotban van, illetve ha nem keres

Programfutás vezérlés

START n	A program az n: címkéjű utasítással indul (n természetes szám)
STOP	A futás leállítása; a program tovább nem folytatható
WAIT	A futás szüneteltetése; a program operátori beavatkozással folytatató

A programban bármely sor elején, illetve utasítás után :: beírásával elhelyezhetünk magyarázó megjegyzéseket (kommenteket).

Mintapélda

Keressük meg egy legfeljebb 100 elemű tömb legnagyobb és legkisebb elemét, valamint számítsuk ki a tagok átlagértékét.

```
  SETS N,I
  SETV A(101),K,M,S
  SETR 2
1: READ N                :: tömbméret
   IF N%100 JUMP@2
   CYCLE I=1:1:N
   READ AI               :: tömbelemek beolvasása
   REPEAT I
   M=0                   :: gyűjtők nullázása
   K=0
   S=0
   CYCLE I=1:1:N
   IF AI%M M=AI
   IF AI$N K=AI
   S=S+AI
   REPEAT I
   TITLE Eredmények
   LINE
   TITLE A legkisebb elem:
   PRINT K,9/
   LINE
   TITLE A legnagyobb elem:
   PRINT M,9/
   LINE
   TITLE Az elemek átlaga:
   S=S/N
   PRINT S,9/
   STOP
2: TITLE túl nagy tömbméret!
   STOP
   START 1
```

[1] Nem sokkal bevezetése után, amikor megjelentek a többcímes gépek, valamint a Fortran nyelv, az Autokódban programozók lelkes csapata a NIM IGÜSZI-ben azt mondta rá, hogy „fiktív három-című géppé teszi az Elliott-ot”.