tanulmányok

53/1976

MTA Számitástechnikai és Automatizálási Kutató Intézet Budapest

MAGYAR TUDOMÁNYOS AKADÉMIA
SZÁMITÁSTECHNIKAI ÉS AUTOMATIZÁLÁSI KUTATÓINTÉZETEE

MP Ø.2 MAKROPROCESSZOR ÁLTALÁNOS ISMERTETÉSE

Irta:

FARKAS ERNŐ

Tanulmányok 53/1976.

A kiadásért felel:

DR VÁMOS TIBOR

ISBN 311 024 6

768146 MTA KESZ Sokszorosító. F. v.: Szabó Gyula

TARTALOMJEGYZÉK

I.	BEVEZETÉS AZ MP Ø.2-BE	4
II.	MÜKÖDÉS	8
III.	A MAKRODEFINICIÓ FORMÁJA	10
IV.	AKTUÁLIS MINTASOROZAT KIJELÖLÉSE	12
V.	A MINTA FORMÁJA ÉS ILLESZKEDÉSE	14
VI.	A MAKRO TÖRZSE	17
VII.	MAKROKITERJESZTÉS	18
VIII.	A BELSŐ VÁLTOZÓK	21
IX.	A TÖMBVÁLTOZÓ	24
X.	STRING VÁLTOZÓ	26
XI.	AZ UGRÓ DIREKTIVÁK	28
XII.	SZÓTÁRKEZELÉS	30
XIII.	INPUT LEHETŐSÉGEK	32
XIV.	OUTPUT LEHETŐSÉGEK	34
XV.	ÜZEMMÓDVÁLTÓ DIREKTIVÁK	36
XVI.	A MUNKA BEFEJEZÉSE	37
	F Ü G G E L É K	40
XVII.	A MAKRO AUTÓKÓD	40
XVIII.	MAKRO AUTÓKÓD DIREKTIVÁK	45

AJÁNLOTT OLVASÁSIMÓD

Először csak a Ø. pontokat olvassuk, majd mindig egy nagyobb sorszámu pontig olvassuk ujra.

A szövegben előforduló ' jelek: szövegrészeket elhatároló idézőjelek, "nyomdatechnikai" jelek külön betütipus hiányában.

I. BEVEZETÉS AZ MP Ø.2 -BE

Ø

Az MP Ø.2 egy, az IDOS operációs rendszer részét képező szövegkezelő eszköz. Müködésének alapelve a mintaillesztés. A vizsgált szöveget soronként a szabványos és a felhasználó által megadott mintákkal hasonlitja össze. Minden mintához egy, illeszkedés esetén érvényessé váló intézkedés tartozik. Az intézkedés általában a vizsgált sornak, egy, több, esetleg nulla soros üres szöveggel való helyettesitése.

A minták mintasorozatokat alkotnak. A minta felismerése ugy történik, hogy a vizsgált szöveg egy sorát összehasonlitjuk egy mintasorozat egyes elemeivel, egymás után mindaddig, amig megegyezést nem találunk.

Egy mintának a vizsgálati láncban elfoglalt helyét, a mintát, valamint a hozzárendelt intézkedést tartalmazó leirást a továbbiakban makronak, magát az intézkedést a makro törzsének nevezzük. Azt a folyamatot pedig, melynek során mindezt a felhasználó megadja, makrodefiniciónak nevezzük.

A processzor munkáját irányitó szabványos intézkedések a direktivák.

A gyakorlati feladatok egy részében szükség van a feldolgozás állapotát jelző, számértéket, esetleg szöveget tartalmazó tárolókra. Ezen tárolók aktuális tartalmára való hivatkozás bármely soron belül megengedett. Az illeszkedés vizsgálatát ezek behelyettesitése előzi meg.

1

A fentiekből is látható, hogy az MP Ø.2 kevéssé alkalmas arra, hogy több soros mintákat felismerjen, illetve arra, hogy több sorból egy sort csináljon, bár körmönfont programozással erre is lehetőséget nyujt.

Fokozott óvatosságot igényel az is, amikor valamely sorrészletet akarunk felismerni és kicserélni, (pl. változót, számot stb.), nehogy ott is "felismerjük", ahol csak valami részeként fordul elő.

2

Mire használható tehát az MP Ø.2 makroprocesszor?

- Közönséges makrokiterjesztés:
  A szöveg bizonyos mintáju sorait makrohivásoknak tekintjük és helyükbe a megfelelő hosszabb szövegrészeket beirjuk. A makrotörzs, program értelmezés esetén nyilt szubrutin, amely minden híváskor bemásolódik, adatok, szövegek stb. esetén pedig egy tömör formáju rövidités kifejezése.
- Kihagyás:
  Bizonyos mintáju sorokat kihagyunk a szövegből. Ezt ugy érjük el, hogy a kihagyandó sorokat üres törzsü makroval ismerjük fel, azaz nulla sorral helyettesitjük. Elsősorban file-ok tömöritésére használjuk.
- Kigyüjtés:
  Bizonyos mintáju sorokat (esetleg átalakitott formában) kiadunk. A többi sort pedig elhagyjuk.
- Egyszerü forditás:
  Az input szöveg,minden sorához szükségszerüen tartoznia kell egy mitának, és ennek megfelelően a sor valamivel helyettesitődik. Ha a sorra egyetlen minta sem illik, hibajelzést kapunk.

3

Ezek persze csak a legegyszerübb esetek; ugyanezen feladatok bonyolultabb feltételek esetén is előfordulhatnak. A továbbiakban néhány, az MP Ø.2-re jellemző, de már bonyolultabb alapmegoldásra hivjuk fel a figyelmet.

- Szelektiv felismerés:
  Lehetőségünk van arra, hogy ne az összes definiált makro illeszkedését vizsgáljuk, hanem csoportokra osztva mindig csak egy csoportét és ezt a felismerendő mintasereget automatikusan változtassuk. Például egy programnak az adatmezejében és a programmezejében érvényes feldolgozási szabályokat szétválaszthatjuk ás ezzel a végrehajtás sebességét is gyorsitjuk és esetleg hibákat előzhetünk meg.
- Többszintes makrofeldolgozás:
  A processzor, miután a megfelelő mintáju sort a megfelelő sorokkal helyettesitette, az igy kapott sorokat mégegyszer átnézi, hogy nem talál-e köztük ismét valamelyik mintához illeszkedőt, és ha igen, a megfelelő helyettesitést ismét elvégzi. Igy lehetőségünk van arra, hogy a makro kifejtését több fázisban oldjuk meg, amely áttekinthetőbb makrok létrehozását, a több makroban közös rész makroval történő leirását biztositja.
- Iterativ felbontás makroval:
  Gyakran állunk szemben olyan feladattal, hogy egy változó hosszuságu sort kell részekre (elemekre) bontanunk. Ennek tipikus megoldása az, hogy a sort a makroprocesszor egy elemmel rövidebb sorra és az utolsó elemre bontja fel. A maradékra ilyenkor ujra alkalmazható a felbontást végző minta és helyettesités, igy végül a sor elemekre bomlik.

- A makrokészlet dinamikus bővitése:
  Az amivel a sort helyettesitjük, bármi lehet, nevezetesen egy ujabb makro definiciója is.

II. MÜKÖDÉS

Ø

Az MP Ø.2 tevékenységeit 3 részre oszthatjuk:

- makrokészlet betöltése, amely meghatározza, hogy milyen mintáju sort mivel kell helyettesiteni (program betöltés)
- makrokiterjesztés, azaz a minták felismerése és a helyettesités elvégzése
- egyéb tevékenységek

Ennek megfelelően az input szövegben lehetnek:

- makrodefiniciók
- szövegsorok (ezeket kell a mintákkal egybevetni és helyettesiteni)
- direktivák (a processzor munkáját irányitó parancsok, amelyeket a processzor azonnal végrehajt)

A makrodefinició egy többsoros direktivának tekinthető. A többi direktiva egysoros. Az input sorok közül a direktivákat az különbözteti meg a többiektől, hogy '&' jellel kezdődnek. Ez figyelmezteti a processzort arra, hogy most nem feldolgozandó sor, hanem végrehajtandó parancs következik.

1

A direktivák és szövegsorok felváltva, tetszőleges sorrendben következhetnek egymás után.

2

Kialakult azonban az a gyakorlat, hogy a makrokészletet (programot) és az adatokat külön file-(ok)ba gyüjtjük és végrehajtáskor elõször az egyik, majd a másik file-t olvastatjuk el a processzorral.

3

Bonyolultabb esetben a teljes makrokészlet (program) betöltése a munka legelején nem célszerü (vagy nem is lehetséges). Külön program és külön adatfile(-ok) alkalmazása ilyenkor is racionális.

A makroprocesszor lehetõséget nyujt arra is, hogy egy adott pillanatban az addig definiált makrokat töröljük és a makroprocesszort ezzel alapállapotba helyezzük. (A makrok törlésével együtt &BEGIN AT 1 hatás is létrejön.) Erre a célra szolgál a

&RESET

direktiva.

III. A MAKRODEFINICIÓ FORMÁJA

Ø

A m a k r o d e f i n i c i ó

Egy fejből: &MACRO NO 'N' NEXT 'K'

Egy mintából: Pl: IZE ?! /?!/ !!

Egy törzsből: Pl: VALAMI 3

A + 4 - 2
STB.....

Egy farokból: &END OF MACRO

áll, amelyek közül a törzs üres is lehet, azaz elmaradhat. 'N' egy egész szám 2 és 255 között, ez a makro száma. Amikor a direktivákban a makrora hivatkozunk, ezt a számot kell használnunk.

A makroknak sorrendet kell definiálnunk. Ez a sorrend mondja meg, hogy a makrok mintáit a processzor milyen sorrendben próbálja rá a bejövő szövegsorokra. A sorrend definiálása ugy történik, hogy megmondjuk, hogy adott makro után melyik másik makro következzen NEXT 'K' .
A láncot k = 1-el, vagy K = Ø-val kell zárnunk.
Az 1 azt jelenti, hogy az adott sort változatlanul kiadja, a Ø azt, hogy a sort kiirja a display-re unmatchable illeszthetetlen hibaüzenet kiséretében, ahol a felhasználó egyszerü esetekben kijavithatja, vagy 'ETX' illetve 'HOME' leütésével befejezheti a futtatást.

1

K = ∅ -t akkor célszerü a lánc végére irni, ha a makroprogram olyan, hogy minden szövegsort valamely makronak fel kell ismernie, tehát a vizsgált sor elvileg nem juthat a lánc végére. A display-re kiiródó sor ilyenkor arra utal, hogy - szándékunkkal ellentétben - az illető sort egyik makro sem ismerte fel.

2

Kissé komplikáltabb feladat esetén a makrot általában nem egy, hanem több láncba szervezzük, és hol az egyik, hol a másik láncot használjuk.

3

Ha ugyanazt a makrot több láncban is fel akarjuk használni, akkor ujra kell definiálni különböző számokkal. A gyakorlatban ez ritkán fordul elő. Gyakoribb viszont az, hogy két makrolánenak ugyanolyan a vége, csak az eleje különbözik. Ilyenkor a végeket egyesithetjük olymódon, hogy az első közös makro két különböző makronak lesz a rákövetkezője.

IV. AKTUÁLIS MINTASOROZAT KIJELÖLÉSE

Ø

A sorozat első elemét, amelyre a mintaösszehasonlitást el kell kezdenünk, a

&BEGIN AT 'N'

direktivával jelöljük ki, ahol 'N' a kivánt makro sorszáma. Egy adott időpillanatban mindig egy makrosorozat van érvényben, és mindaddig érvényben is marad, amig ujabbat ki nem jelöltünk.

&BEGIN AT 1

állapotot tételezi fel, azaz mindent kiad változatlanul. Ez az állapot addig marad érvényben, amig a makrokat nem definiáltuk és meg nem mondtuk, hogy melyik a legelső.

1

Előfordul az az eset, hogy valamelyik azonositásra váró sorral nem ugy akarunk bánni, mint általában a többivel, azaz más mintákkal akarjuk ebybevetni, mint a többit. Ezt a

&MATCH WITH 'N'

direktivával érhetjük el, amely esetben az egyetlen utána jövő sor vizsgálata az 'N' sorszámu makroval indul, de utána ismét a &BEGIN AT-tel kijelölt standard makrosorozat lép életbe.
Vigyázat: A kijelölés csak a következő sorra érvényes, az ennek kiterjesztéséből kapott sorokra már nem!

A '&MATCH WITH' direktivát leggyakrabban a következő három esetben használjuk:

- Ha egy adott tipusu sor után egy adott másik tipusunak kell jönnie.
- Ha egy adott tipusu sor után valamilyen sorok nem jöhetnek.

Mindkét esetben az első sorhoz tartozó makro törzset egy megfelelő '&MATCH WITH' direktivával fejezzük be.

A harmadik eset a leggyakoribb: Ez esetben

- Egy makro törzsben egy olyan sort hozunk létre, amelyről tudjuk, hogy még további felbontást igényel bizonyos makrok szerint. Ez esetben az előtte levő sorban mondjuk meg egy ' MATCH WITH' direktivával, hogy melyek ezek a makrok.

A '&MATCH WITH' direktiva tulajdonképpen egy-egy átkapcsoló, és visszakapcsoló '&BEGIN AT' direktivapár helyettesitésére szolgál. Több sor esetén ilyen egyszerüsités nem lehetséges.

V. A MINTA FORMÁJA ÉS ILLESZKEDÉSE

Ø

A makrok mintái, amelyek alapján az azonositást végezzük, három elemből állhatnak:

- Kulcsszavakból
- Kötött paraméterek jeleiből
- Szabad paraméterek jeleiből

Kulcsszónak számit minden olyam jelsorozat, amelyben nincs kötött vagy szabad paraméter jel, továbbá automatikusan kulcsszónak számit a sor eleje és a sor vége.
(Vigyázat: a space-k is!)

Kötött paramétert jelöl egy '!' sorozat. A kötött paraméter csak kulcsszó előtt vagy kulcsszó után állhat és olyan hosszu paramétert jelöl, ahány '!'-ből áll a sorozat. (Ezért hivják kötöttnek.)

Szabad paramétert jelöl egy '?' . A '?' jelentése "bármi".

(Az előzőekből következik, hogy a szabad paraméter két kulcsszó, illetve a hozzájuk tartozó kötött paraméterek között állhat.)

Azt mondjuk, hogy egy szövegsor illeszkedik egy mintához, ha a két sor egymásra fektethető oly módon, hogy a minta kulcsszavai a sorban előforduló ugyanolyan kulcsszavak fölé kerülnek; a kötött paraméterek jelei olyan karaktersorozatok föle, amelyek ugyanolyan hosszuak, a szabad paraméterek jelei pedig olyan karaktersorozatok föle, amelyek tetszőleges hosszuak (esetleg üresek).

Külön kérhető - az illeszkedés kiegészitő feltételeként - annak vizsgálata, hogy a kötött paraméterjeleknek megfelelő szövegrész zárójelet ne tartalmazzon, a szabad paraméterjeleknek megfelelő karaktersorozatokban pedig a kezdő - és

végzárójelek a helyes zárójelezés szabályai szerint legyenek elhelyezve.

Természetesen egy minta többféleképpen is illeszkedhet egy szövegsorhoz. A makroprocesszor az illeszkedést midig jobbról kisérli meg, igy az első lehetséges illeszkedéskor a kulcsszavak a lehető legjobboldalibb elhelyezkedésüek. (Ez az aritmetikai kifejezések feldolgozásánál hasznos.)

l

A makroprocesszornak definiálhatunk olyan mintát is, amelyben csak paraméterjelek vannak (a sor eleje, sor vége triviális kulcsszavakat leszámitva) . Az ilyen sor minden olyan sorra illik, amely a kötött paramétereknek megfelelő számu karaktert tartalmaz. Ilyenkor vigyázzunk arra, hogy a makro kifejtése során ne keletkezhessen olyan sor, amire a mintát ujra ráhuzhatjuk, mert ettől a processzor végtelen ciklusba esik. Ennek lehetősége bármilyen mintáju makro esetében fennáll, de különösen a kulcsszó nélküli minták esetén gyakori hiba, ezért az ilyen jellegü makrok mintájában gyakran alkalmazunk a mintát pontositó, töltelék kulcsszavakat.

A mintákra semmilyen megkötés nincs. Olyan mintát azonban ne alkalmazzunk, amelynek első karaktere '&' vagy ':' , mert az ilyen karakterrel kezdődő sorokat nem szövegsoroknak, hanem direktiváknak, illetve feltétlen output soroknak tekinti a makroprocesszor.

2

A makroprocesszor végeredményben osztályokba sorolja a szövegsorokat aszerint, hogy a sornak milyen a mintája. A több osztályba besorolható sor abba az osztályba kerül, amelyiknek a mintáját korábban néztük, azaz amelyik a vizsgálatra kijelölt makrodefiníció sorozatban előbb áll.

Gyakran előfordul, hogy két olyan mintánk van, ahol az egyik minta a másik mintának része:

Pl: IF ? THEN ?

és

IF ? THEN ? ELSE?

Ilyenkor két megoldás lehet: vagy előre vesszük a sorban a hosszabb mintát, és ha nem jó, akkor nézzük a rövidebbet, vagy a rövidebb minta szerint ismerjük fel a sort, és az adott paramétert tovább bontjuk egy belső makroval.

3

Egy-egy szövegsor osztályra jellemző mintát általában hosszabb és rövidebb formában adhatunk meg, csak az a lényeg, hogy jól elkülönitse a minta az egyik osztályt a másiktól. A futási idő csökkentése szempontjából azonban előnyös ha az egyes mintákat minél részletesebben (minél több kulcsszóval) a adjuk meg.

Ennek több oka is van: Egyik az, hogy a sikertelen próbálkozások esetén hamarabb kiderül, hogy a szövegsor nem illeszkedhet a mintához. A másik pedig, hogy a kulcsszavak vizsgálata gyorsabb.

Az a legelőnyösebb a makroprocesszornak, ha két sor különböző kulcsszóra végződik.

VI. A MAKRO TÖRZSE

ø

A makro törzse a minta és a farok között helyezkedik el. Hossza lehet nulla is. A törzs különböző tipusu sorokból állhat: Tartalmazhat szövegsorokat, direktivákat és teljes makrodefiniciókat is. Ezekkel a sorokkal a makrodefinició folyamán semmi nem történik (tehát nem kerülnek végrehajtásra). A törzs beolvasása alatt a processzor csak azt vizsgálja, hogy a törzset befejező &END OF MACRO megérkezett-e már. Amikor a befejező sor megjött, a makrotörzs nyilvántartásba vétele megtörténik.

3

Minden makrodefinciónak egyetlen egységes tömböt kell képeznie az input szövegben.

A &MACRO NO 'N' NEXT 'K'

és &END OF MACRO

Ugy viselkednek a sorok szintjén, mint egy-egy kezdő és végzárójel. Helytelenül zárójelezett szöveget pedig a makroprocesszor nem ért meg, vagy rosszul értelmez.

Az ilyen hibák esetén a makroprocesszor általában azt jelzi vissza, hogy kifutott az input file-ból. Ez rendszerint annak a jele, hogy bezáratlan makrodefinició keletkezett.

VII. MAKROKITERJESZÉTS

Ø

Amikor a makroprocesszor a szövegsorban egy makro mintáját felismeri, akkor azt a sort elhagyja és a megfelelő makrotörzsbeli sorokat adja ki helyette. Ezt az eljárást makrokiterjesztésnek nevezzük.

A makrotörzs sorai tartalmazhatnak hivatkozásokat az egyes paraméterjelöléseknek megfelelő szövegrészekre. Mind a szabad, mind a kötött paraméterek számára, ugyanis elhelyezkedésük sorrendje - balról jobbra - egy sorszámot határoz meg. A paraméterre ezzel a sorszámmal lehet hivatkozni. Egy mintasorban legfeljebb 9 paraméter szerepelhet.

A makrokiterjesztés a következő fázisokban megy végbe:

A makrotörzset behozza, és közben figyeli,hhogy talál-e benne ^1, ^2 ... ^9 alaku jelpárokat. Ha igen, akkor helyettesiti azzal a jelsorozattal, amelyet a minta felismerésekor talált a balról számitott ugyanolyan sorszámu paraméter helyén.

Az igy keletkezett szöveget bepakolja egy stackbe olyan módon, hogy a stack tetejére a törzs első sora kerül.

A stackből kiveszi egynként a sorokat, helyettesiti a

^A,... ^Z,... ^$,... ^* stb.

alaku jelpárokat (lásd részeltesebben a makrováltozóknál), és ujra összehasonlitja az aktuális makrominta sorozattal. Ha a sor direktiva volt, végrahajtja. Ha a sor makrohivás volt, a fenti eljrását végrehajtja rajta, ha egyik sem outputra kerül.

Amikor a stack kiürült, az olvasását folytatja az előzőleg használt inputról.

1

Mint az a korábbiakból is következik, készithető olyan makrokészlet, amelynek hatására a stack nem ürül ki (sőt esetleg egyre telitődik), erre vonatkozólag a makroprocesszor nem nyujt védelmet, de a rendelkezésre álló memóriaterület telitődése esetén

NO SPACE !*

hibajelzést ad.

2

A makroprocesszor a szövegsorok beolvasása után (történjen az akár kivülről, akár a stackből) a mintaillesztés kipróbálása előtt megvizsgálja azt, hogy a sor ':'-tal kezdődik-e. Ha igen, akkor a sort azonnal kiadja az outputra. (A ':'-ot természetesen elhagyja).

A ':'-ot különösen a makrotörzsben alkalmazzuk gyakran, célszerü, ha a törzsben azokat a sorokat, amelyeket változatlan formában akarunk kivinni, kettősponttal jelölünk meg, igy azok minden vizsgálat nélkül azonnal outputra kerülnek. Ezáltal feleslegessé tesszük, hogy a processzor olyan minták illeszkedését vizsgálja, amelyekről ugyis tudjuk, hogy nem jöhettek szóba.

Vigyázat: A paraméterek és a makrováltozók helyettesitése ekkor is megtörténik, csak a mintaillesztés marad el!

3

Ha egy makro törzsében egy másik makrodefinició van, akkor a belső makro törzsében a külső makro paramétereire

^1,^2 ...^9

a belső makro paraméterire pedig

^:1,^:2,...^:9

formában hivatkozhatunk.

A makroprocesszor ugy müködik, hogy amikor a külső makrot kifejti, akkor helyettesiti a

^1,^2, ... ^9

hivatkozásokat, és ugyanakkor a ^: karakterpárból ^karaktert csinál, igy a belső makro törzsében, amikor a feldolgozás során sor kerül rá, a paraméterekre való hivatkozások már a szokásos formát öltötték.

A paraméterekre való hivatkozások természetesen ugyanigy oldhatók meg (megfelelő számu ':' beiktatásával) több szint esetén is.

VIII. A BELSŐ VÁLTOZÓK

Ø

A makroprocesszor a makrok definiálásán, felismerésén és hivásán kivül más tevékenységeket is végrehajt. Ezeknek a tevékenységeknek a megértéséhez be kell vezetnünk a makrováltozó fogalmát.

A makroprocesszor belsejében 26 olyan munkarekesz van, amelyben számokat helyezhetünk el. A rekeszek tartalmát megváltoztathatjuk, illetve bizonyos ellenőrzéseket végezhetünk rajtuk. A rekeszeket A-tól Z-ig az angol ABC betüivel jelöljük. A makrováltozók kezdőértéke zérus.

Lehetőségünk van arra, hogy bármely beolvasott sorban hivatkozzunk ezen értékek valamelyikére. A behelyettesités (a makrominta és a makrotörzs kivételével) közvetlenül a sor beolvasását követően történik meg.

A hivatkozás formája ^A ... ^Z alaku.

Ha a beolvasott sorban ilyen karakterpár fordul elő, a beolvasás után ennek a karakterpárnak a helyére a makrováltozó decimális értéke kerül annyi karakterben, ahány az elhelyezéshez szükséges.

A szövegsorokban lehetőségünk van arra, hogy

^$

alakban hivatkozzunk az aktuális input file-ból olvasott sor sorszámára. Ennek behelyettesitése a makrováltozókkal egyidejüleg történik.

A makrováltozóknak a következő direktivákkal adhatunk értékeket:

```
&'V' = 'E1'
&'V' = 'E1'+'E2'
&'V' = 'E1'-'E2'
&'V' = 'E1'*'E2'
&'V' = 'E1'/'E2'
&'V' = 'E1':'E2'
```

'V' a makrováltozót jelölő betü helyett áll, 'E1' és 'E2' egy-egy érték. Az értéket (A ^- hivatkozások kifejtése után) az alábbi jelsorozat - tipusok reprezentálhatják:

- Számjegyek (Pl. ' 187 ')
  Ha a szám nem fér el két byte-ban, vagy hibás

  Ekkor IN NUMBER hibajelzést kapunk.

- Egy betü (Pl. ' A ')
  Belső változókra való hivatkozás, a változó pillanatnyi értékét jelenti.

- Idézőjelek közötti látható karakter
  (Pl. ' "#" ') az érték a karakter ascii kódja.

- Idézőjelek közötti látható karakterpár
  (Pl. ' "XY" ') az érték nagy helyértékü byte-ja az első, kis helyértékü byte-ja a második karakter ascii kódja.
  (De nem lehet a látható karakter *,/,:,+, mert összetéveszthető a müveleti sellel)

- "üres" STRING (' ')
  értéke zérus.

Valamennyi esetben a jelsorozatot esetleg megelőző, illetve követő szóközöket a processzor figyelmen kivül hagyja.

A szám negativ, ha legfelső bitje 1. a számábrázolás a gépi ketbyte-os egész számok ábrázolásával azonos. A müveleteket is ennek megfelelően, a megfelelő gépi utasitásokkal végzi.

'+' az összeadást, '-' a kivonást, '*' a szorzást, '/' az egész osztás hányadosát, ':' az egész osztás maradékát jelöli.

A müveletek a tulcsordulást és az átvitelt figyelmen kivül hagyják.

1

Bár a makrováltozókra való hivatkozás formailag a paraméterre történő hivatkozáshoz hasonlit, külön felhivjuk rá a figyelmet, hogy a makrováltozók más időpontban, más algoritmus szerint helyettesitődnek.

A mintasorban a karakterpár változatlanul megmarad.
A törzsben előforduló karakterpár akkor helyettesitődik, amikor a stackből ujra elolvassuk a sort (kivéve természetesen az egymásba ágyazott makrok esetét, amikor csak a belső makro törzsének ujraolvasása esetén kerül helyettesitésre).

3

A makrováltozók természetesen nem csak szövegsorokban, hanem direktivákban is előfordulhatnak. Igen gyakori az a megoldás, hogy a szöveg alapján makrokat generáltunk és ezeket a makrokat a belső változók segitségével sorszámozzuk, illetve, hogy a '&BEGIN AT' , illetve '&MATCH WITH' direktivában makrováltozót használunk.

IX. A TÖMBVÁLTOZÓ

Ø

A makroprocesszor müködése során nem csak makrováltozókban tárolhatunk értékeket, hanem lehetőség van arra is, hogy egy tömböt rezerválhassunk egész értékek tárolására. Erre a célra szolgál a

```
& RESERVE 'K'
```

direktiva. A direktiva egy 'k' hosszuságu tömböt foglal le. A tömbbe a

```
&#,'X'='E'
```

és

```
'V'= ,'X'
```

direktivával irhatunk be, illetve olvashatunk ki. (Itt 'X' egy 'K'-nál kisebb értéket jelöl, 'E' egy tetszőleges érték lehet, 'V' pedig egy makrováltozót jelölő betü.)

1

A lefoglalt tömb a következő makrodefinicióig, illetve '& RESET' direktiváig érvényes.

2

A processzor ellenőrzi, hogy csak az érvényes tömbhatárig történhet beirás. A tömbelemek kezdőértéke nulla.

3

Lehtőség van arra, hogy az

&ARRANGE 'N'

direktivával a tömb első 'N' pozitiv értékü elemét sorbarendezzük. A sorbarendezés után a tömb első 'N' elemének értéke növekvő sorba rendeződik. A lerendezett sorozatot egy negativ érték zárja. A többszörös értékek nem tartják meg multiplicitásukat. 'N' szükségképpen kisebb 'K'-nal.

X. STRING VÁLTOZÓ

Ø

A makroprocesszorban a numerikus értékü változókon kivül van egy szöveg tipusu változó is, melyet '*'-gal jelölünk. Ebbe bármikor betárolhatunk egy maximum +2 karakter hosszu stringet a következő direktiva segitségével:

```
&*'STRING'
```

erre a 'STRING'-re

alakban hivatkozhatunk. Ennek behelyettesitése a numerikus makrováltozókkal egyidejüleg történik.

1

A string változóban a

```
&'V'*'C'
```

illetve

```
&'V'* 'STRING'
```

direktivával helyeztünk el egy karaktersorozatot. ('V' egy makrováltozó, 'C' egy karakter.)
Hatására a string változóban elhelyezésre kerül 'V'-ben meghatározott számu 'C' karakter.

A string kitételére csak akkor van szükség, ha a karakter space, ez ugyanis nem állhat a sor végén.

2

Egy tetszőleges string hosszának meghatározására módot nyujt a

```
&!V'<'STRING'
```

direktiva. Hatására a 'V' változóban tárolásra kerül a string hossza.

3

Például a

```
&A<^2
```

direktiva hatására az A változóba kerül a második paraméter hossza.

XI. AZ UGRÓ DIREKTIVÁK

Ø

A makrováltozók lehetőséget nyujtanak arra, hogy ugró direktivákat vezessünk be. Az ugrás azt jelenti, hogy a processzor a direktiva hatására adott számu sort olvas be, és ezekkel nem csinál semmit (lenyeli őket). A makroprocesszor a szövegben csak előre tud ugrani. Utána a munkát a következő sor rendes feldolgozásával folytatja.

Az ugró direktivák alakjai a következők:

```
&SKIP 'K' IF  'E' NEGATIV
&SKIP 'K' IF   E  POZITIV
&SKIP 'K' IF   E  ZÉRO
&SKIP 'K'
```

1

Ha 'K' negativ vagy zéró, sem ugrás, sem hibajelzés nem váltódik ki.

2

Az ugró direktivákat feltételes szöveggenerálásra használhatjuk.

Egy programot ugy irhatunk meg, hogy egyes részeit feltételtől függően vesszükbbe, egyes részeket több alternativában irunk meg. Majd a teljes programból, amely több lehetséges alternativa leirását tartalmazza, ugy generálunk egy konkrét programot, hogy egy kis programban beállitjuk a makrováltozók értékét, végigolvashatjuk a feltételes szö-

veget, és igy megkapjuk a végleges szöveget.

Hasonló elágazásokat tehetünk makrok törzsében is. Ha az egyik makro a makrováltozóba beállit egy értéket, akkor a másik makro ez alapján generálhat különböző szövegeket a makrohivás helyére.

3

Az átugrott sorokat a processzor egyáltalán nem vizsgálja. Igy, ha egy makrodefinició kezdősorát átugorjuk, akkor esetleg a törzs közepébe kerülünk és ez helytelen végrehajtást eredményez. Ez a hiba rendszerint ugy jelentkezik, hogy az

```
& END OF MACRO
```

sor

```
UNMATCHABLE
```

hibajelzés kiséretében iródik ki.

Ha viszont egy makro törzséből adunk egy akkora ugrást, amely a törzs méretét meghaladja, akkor a törzs többi sorát kihagyjuk, a további szövegből pedig még annyi sort, amennyi hiányzik. Ez egy legális trükk.

XII. SZÓTÁRKEZELÉS

∅

A makroprocesszor lehetőséget nyujt arra, hogy stringeket szótárba vegyünk. A

```
&'V'@'STRING'
```

direktiva hatására a processzor megvizsgálja, hogy az adott string előfordult-e már a szótárban, ha igen, a 'V' változóban megadja, hogy milyen sorszám alatt. Ha még nem szerepelt a szótárban, uj elemként beirja oda a soronkövetkező sorszámmal, és azt adja vissza a 'V' változóban.

Az igy elrakott szótári tételekre a makrováltozókhoz hasonlóan a

```
^@'V'
```

direktivával törölhetünk. Hatására a szótárból törlődnek a 'V'-vel megadott sorszámu és az annál magasabb sorszámu stringek.

1

A szótár a következő makrodefinicióig vagy &RESET direktiváig érvényes.

2

A szótárban való keresés is mitaillesztés, de - mivel azonos minták nem fordulhatnak elő - a keresés iránya érdektelen.

3

A szótár elemeihez értéket rendelhetünk olymódon, hogy

```
&RESERVE 'K'
```

direktivával lefoglalunk egy tömböt és a tömb szótári sorszámu elemébe rakjuk az értéket.

XIII. INPUT LEHETŐSÉGEK

Ø

A makroprocesszor bemenetét az idős operációs rendszer szövegfile-jai képezhetik. Egyszerre maximum 15 file jelölhető ki. A file-okra a kijelölés során kapott sorszámokkal hivatkozhatunk, a sorszámok Ø-tól 14-ig terjednek. A nullás sorszámu file tartalmazza a processzor számára irt főprogramot.

A főprogramból meghivhatjuk a

```
&FILE 'N'
```

direktiva segitségével a többi file-t. A direktiva hatására az input file olvasása megszakad és az adott file olvasása kezdődik meg. Ha a hivott file elfogyott, az olvasás automatikusan a hivó file-ből folytatódik. Ha a

```
&FILE
```

direktivát paraméterenélkül adjuk ki, akkor a legmagasabb sorszámu (az utolsó) file kijelölésére kerül sor.

A főprogram file-jának a "0" sorszámu file-nak & jellel kell kezdődnie. /Ezt az autókódnál magyarázzuk meg./

1

A

```
&FILE
```

direktivát makrotörzsben is kiadhatjuk, hatására természetesen csak a törzs esetleges további sorainak beolvasása után érvényesül.

2

Az MP Ø.2 file kezelő rendszere három input file pointerét tárolja.
Ezek:

- "Ø" sorszámu vezérlő file
- "hivó" file
- "hivott" (aktuális) file

File vége jelzés esetén a "hivó" lesz a "hivott" a "Ø" pedig a hivó. Ha a "Ø" file ad file vége jelzést,

END OF FILE

hibajelzést váltódik ki.

3

Ha egy file-t hivtunk, akkor lehetőségünk van arra, hogy a hivott file-ban egy

&RETURN

direktivát adjunk ki. Hatására az eredeti hivó file-nak adódik vissza a vezérlés. Ekkor viszont ebben adhatjuk ki a &RETURN direktivát, melynek hatására ismét a legutolsóra hivott filenak a '&RETURN'-t követő sornál folytatódik az olvasás.

XIV. OUTPUT LEHETŐSÉGEK

ø

A processzor munkája közben a keletkező output sorokat egy output filera helyezi el. A

&CLOSE 'V'

direktiva lehetőséget nyutj arra, hogy egy adott pillanatban az ddig keletkezett output file-t lezárjuk és a továbbiakban input file-ként használjuk. Hatására uj output file-t kezdünk, és a régit az eddigieknél eggyel nagyobb sorszám alatt az input file-ok között felhasználhatjuk.

A direktiva után egy makrováltozó neve is állhat (egy betü), ezt jelöltük 'V'-vel. Ebben az esetben a megfelelő makrováltozóba az uj input file sorszáma kerül. Ha nem áll semmi, a sorszám nem kerül letárolásra, ilyenkor paraméter nélküli

&FILE

direktivával érhetjük el legegyszerübben.

i

A

& PRINT'SZÖVEG'

direktiva hatására a szöveg a printeren kinyomtatásra kerül. A nyomtatandó szöveg a display képernyőjén is megjelenik.

2

Nyomkövetésre a következő direktiva nyujt módot:

&TRACE'SZÖVEG'

A direktiva hatására a szöveg kiirható a display-re és a printerre is. A direktivát tartósan beépitherjük programjainkba, nem szükséges, hogy a kész programból kivegyük, ugyanis a direktiva csak bizonyos kulcsállásnál müködik.

Ha a kezdőpulton 8ØØ1 van beállitva, akkor a képernyőreakerül, és automatikusan minden egyes további kiirás egy sorral feljebb csusztatja a szöveget. &ØØØ1 beállitása esetén a szöveg a printeren is kinyomtatásra kerül.

XV. ÜZEMMÓDVÁLTÓ DIREKTIVÁK

Ø

Alapállapotban a makroprocesszor a zárójelezés helyességét a kötött és szabad paraméterekben nem figyeli. A zárójelek figyelése a

&(

direktiva hatására kezdődik meg.

1

A makroprocesszor a tárolóhivatkozásokat (^ ...) a feldolgozandó adatszövegben is kifejti a beolvasás során. Ahhoz, hogy makroprocesszorral feladatleirás jellegü szövegeket is feldolgozhassunk, ezt a kifejtést le kell tudnunk tiltani. Erre a célra a / karaktert használhatjuk (^/...). Ennek a tiltásnak automatikus feloldását - A ^/ pár ^ -lal való helyettesitését - a

&/

direktiva segitségével érhetjük el.

3

Mindkét direktiva hatását az

&&

direktiva szünteti meg.

XVI. A MUNKA BEFEJEZÉSE

Ø

A makroprocesszor munkáját az

&FINISH

direktivára fejezi be. Hatására a legutolsó output file-ra implicit

&CLOSE - t

alkalmazunk, majd a makroporcesszor munkája véget ér és a könyvtáros jelentkezik és nevet kér az eredmény file-nak.

Lehetőségünk van arra, hogy a

&FINISH

direktivának egy vagy több (vesszővel elválasztott) makrováltozó nevét (betüt) adjunk meg paraméterként. Ennek hatására az utolsón kivül a változóknak megfelelő sorszámu input file-okat is könyvtárba vehetjük.
A könyvtáros ilyen esetben kiirja azt a makrováltozót jelölő betüt, amelyikre aktuálisan nevet kér.

1

Ha munka során hiba történik, a sor hibaüzenettel kiiródik. A hibaszöveget követően három szám iródik ki:

'A'/B'.*'C'*

formában.
'A' az aktuális input file sorszáma, 'B' az utoljára beolvasott sor sorszána ezen a file-on belül, mig 'C' az aktuális vizsgálati lánc első tagjának sorszáma.

Ha a cursor a kiirás után a számokat követően áll, a processzor munkája nem folytatható és tetszőleges karakter leütésére a rendszer jelentkezik be. Ha a cursor a hibásnak talált sor első karaktere alatt áll, a felhasználó a sort kijavithatja. A javitáshoz használhatja az összes látható karaktert, továbbá a 'RETURN' billenytyüt, amelynek hatására a cursor a sor elejére lép, a 'CR' és 'CL' cursor-mozgató billentyüket, az 'IC' 'DC' billentyüket beszurásra illetve kitörlésre. A begépelt sort 'ETX' karakterrel kell zárni. Hatására a cursortól jobbra álló karakterek törlődnek.

'HOME' leütésére a munka véget ér és a processzor megpróbálja az eredményt könyvtárba venni. 'ERASE' hatására a vezérlés az operációs rendszernek adódik át.

2

Amennyiben a munka során eredményfile-t nem állitunk elő (Pl: kigyüjtés printeren, felhasználói hibajelzés stb.), a makroprocesszor munkáját az

&EXIT

direktiva segitségével is leállithatunk. Hatására a display-ernyőn

EXIT

formális hibaüzenet jelenik meg.

3

Ha az eredményfile assembler forrásnyelvi program, további feldolgozásra közvetlenül átadható a rendszer assembler programjának. Erre a célra a

& START "A"

direktiva szolgál. Ennek hatására behivásra kerül az assembler, amely a keletkezett file-t fogja leforditani.

Assembler forrásnyelvi programrészlet (belövés alatt álló program) könyvtárazása a

& START "E"

direktiva segitségével váltható ki. Hatására az editor program a keletkezett file-t az assembler nyelvü könyvtárba veszi.

Ha az eredményfile makroprocesszor feladat-leirás (makrokészlet és a hozzátartozó vezérlő direktivák), ennek vezérlőprogramként történő elinditását a generáláshoz használt file-ok kitörlésével a

& START 'V'

direktiva teszi lehetővé. A 'V' változó tartalma, az elindítás után maximális sorszámmal rendelkező adatfile sorszáma kell, hogy legyen. A direktiva hatása egyenértékü a generált program könyvtárazásával, majd elinditásával, (A makrováltozók kezdőértéke is zérus!)

A

& START 'V' M

direktiva standard makro-autokod nyelvü feladat-leirás esetén valósitja meg az előbbi funkciót. Az elindítás előtt ilyenkor a program leforditása is megtörténik.

F Ü G G E L É K

XVII. A MAKRO AUTOKÓD

A makro-autokód egy olyan standard makrokészlet, amely a felhasználók számára lehetővé teszi, hogy a definiált makrokat egyszerübben rövidebb formátumban és a sorszámozás nélkül adhassák meg.

Egy makro-autokódban definiált makrokészlet a következő elemekből áll:

- makrodefiniciók
- cimkék
- direktivák
- kommentárok

Ebből a szövegből egy olyan szöveg keletkezik, amely MP/Ø.2-nek megfelelő makrodefinicióból, illetve direktivákból áll.

A kommentátorok a felhasználók megjegyzéseit tartalmazzák, és kihagyásra kerülnek.

A cimkék arra valók, hogy rajtuk keresztül a keletkezett makrok sorszámára hivatkozhassunk, amelyek konkrét értékét nem tudjuk, hiszen a sorszámozás automatikusan történik.

A trace szolgáltatás segitségével kérhető a cimkék számértékeinek kiiratása.

1. A makrodefinició formája

A makrodefinició egy mintasorból, és az azt követő törzs-sorokból áll. A makro végét az jelzi, hogy utána egy mintasor vagy direktiva jön.

A mintasor első karaktere "/", a harmadik karaktere ".", ez után következik a minta. A második karakter

különböző lehet, és eszerint különböző láncolású makrok keletkeznek.

/.. kezdetü sorból olyan makro keletkezik, amelynek sorszáma az előzőnél eggyel nagyobb, és NEXT-jében az előző makrora hivatkozik. (Az első makro sorszáma 2 és NEXT-jében 1-re hivatkozik.)

/Ø. kezdetü sorból, az előzőnél eggyel magasabb sorszámu makro keletkezik, amely NEXT-jében Ø-ra hivatkozik.

/1. kezdetü sorból, keletkező makro sorszáma eggyel magasabb lesz és NEXT-jében 1-re hivatkozik.

/!. kezdetü sorból (ahol ! egy cimke, azaz betü A-tól R-ig) olyan makro keletkezik, amely NEXT-jében a ! cimkéjü makrora hivatkozik. A ! értékének már definiáltnak kell lennie.

2. A törzshöz tartozó sorok ' ' (SPACE) karakterrel kezdődnek.

' .' karakterpárral kezdődő sorok tartalma bármi lehet és a pontot követő karakterek belekerülnek a makro törzsébe.

' /' karakterpárral kezdődő sorba egy direktivamnemonikja kerülhet. Helyette a makro törzsbe maga a direktiva kerül.

A direktivák mnemonikjait, valamint az egyéb sorok rövid, szemléletes magyarázatát lásd a függelékben.

A törzs-sorokat követő mintasor vagy direktiva hatása, mind maga a makro, mind az esetleg benne megnyitott makrok automatikusan lezárásra kerülnek.

3. Egy makrot cimkével jelölhetünk meg, majd sorszámára ezután ezzel hivatkozunk.

   A cimke formája:

   /!/ ahol a ! egy betü A-tól R-ig.
   A cimke függetlenül attól, hogy hol áll (az előző makro után, vagy annak törzsén belül, stb.), az utána következő első mintasorból keletkező makrot jelöli.

   A cimkére való hivatkozás ^! alakban történik (azaz tulajdonképpen makrováltozó hivás). Hivatkozás csak predefiniált cimkére lehetséges.

   Ha postdefinit cimkére akarunk hivatkozni, akkor előre meg kell állapitanunk a hivatkozni kivánt makro sorszámát, és a cimkének ezt az értéket kell adni a következő direktivával:

   /!/+? ahol A ? a kivánt és a pillanatnyi sorszám különbsége.

   Célszerü ilyenkor ellenőrzésképpen az adott mekro előtt a

   /!/ =

   cimkét elhelyezni. Hatására ellenőrzésre kerül, hogy az adott makronak ténley annyi lett-e a sorszáma, mint amennyinek definiáltuk. (Ha nem annyi, hibajelzést kapunk és kiiródik az eltérés.)

   A forditás során a cimkék számértéke &ØØØ1 kulcsállásnál kiirható a sornyomtatón.

   K ö z i s m e r t a z , h o g y a m a k r o p r o c e s s z o r a b e o l v a s o t t s o r o k b a n a R a l a k u m a k r o v á l t o z ó h i v á s o k a t a z o n n a l h e l y e t t e s i t i , e z é r t a z a u t ó k ó d h i v á s o k a t

^//R (t ö r z s ö n b e l ü l ^:R i s l e h e t).
a l a k b a n k e l l i r n u n k .

Az autókód leforditása során a hivatkozásokból a //
eltünik.

4. A direktivák "//" karakterpárral kezdődnek, majd utánuk egy direktiva mnemonikja áll, amely azonos a törzsben használtakkal.
Hatására makrodefinición kivül elhelyezett - tehát a makrokészlet beolvasásakor azonnal végrehajtásra kerülő - direktiva kerül beirásra.

A //M.? (&MACRO NO ...) és a //E (&END OF MACRO) direktivák használata tilos, ezeket maga az autókód helyezi el.

Különleges direktiva ezen kivül a //B, melynek hatására olyan &BEGIN AT direktiva keletkezik, amelyik a legutolsó definiált, illetve az utolsó /S/ cimkéju makrokra hivatkozik.

//RT direktiva hatására &RESET direktiva iródik a kimenő makrókészletbe és a makrok számozása elölről kezdődik.

5. A kommentár sorok *-gal, vagy ◊-sel kezdődhetnek. Kommentár sor az üres sor is.

6. Lehetőségünk van továbbá arra is, hogy az autókódban definiált makrokészletben olyan sorokat irjunk, amelyekről azt kivánjuk, hogy változatlanul menjenek át a keletkező makrokészletbe.
A /. kezdetü sor további része változatlanul megy át.

Több sor átadására szolgál a /" kezdősor, amely után következő sorokat az autókódforditó átmásolja a keletkező makrokészletbe. Az átmásolandó sorok végét egy #-ből álló sor jelzi, amely már nem másolódik át.

A /: kezdősor hatására az utána következő sorok ugy kerülnek átmásolásra, hogy eléjük még egy ':' is kerül. A másolás végét itt is a #-ból álló sor jelzi.

7. A makro-autókód használata az MP/Ø.2 alap-bejelentkezésekor olyan, hogy elsőként a makrokészletet, majd az adatfile-(okat) kell kijelölni.
Amennyiben adatfile-t nem jelölünk ki, eredményfileként a leforditott makrokészletet kapjuk.
Az autókódot fordító makrókészlet behivása automatikusan történik.

Ha Ø file nem & jellel kezdődik, a makroprocesszor autókódnak tekinti automatikusan, ellenkező esetben "leforditott" makrokészletnek tekinti.

XVIII. MAKRO AUTÓKÓD DIREKTIVÁK

......*?	COMMENT
......	COMMENT
......◊?	COMMENT
....../"	IDEZET
....../:	OUT IDEZET
......#	IDEZET VEGE
....../!/?	CIMKE
....../!/+?	PREDEFINITI CIMKE
....../!/=	PREDEFINITI CIMKE ELLENÖRZÉS
...../!/B	BEGIN AT "LAST"(^S)
....../!/RT?	MACRO SORSZÁM UJRA ELÖLRŐL
....../!,	TÖRZS LEZÁRÁS ... /^1 DIREKTIVA
....../ .?	^1
....../1.?	&MACRO NO :T NEXT 1
....../ϕ.?	&MACRO NO :T NEXT ϕ
	^1
....../..?	&MACRO NO :T NEXT :U
....	^1
....../!.?	&MACRO NO :T NEXT : 1
....	^2
..... /!=#,2	1= , 2
..... /!=?-?	&^1=^2-^3
..... /!=?+?	&^1=^2+^3
..... /!=?:?	&^1=^2/^3
..... /!=?*?	&^1=^2*^3
..... /!=?	&^1=^2
..... /SZ?,?	&SKIP^ 1 IF ^2 ZERO
..... /SN?,?	&SKIP^ 1 IF ^2 NEGATIV
..... /SP?,?	&SKIP^ 1 IF ^2 POSITIV
..... /S ?	&SKIP^ 1
..... /E	&END OF MACRO
..... /B!?	&BEGIN AT ^1^2
..... /M !?	&MATCH WITH ^1^2

```
...... /P:?          &PRINT 1
...... /T:?          &TRACE 1
...... /(            &(
...... //            &/
...... /&            &&
...... /*?           &*^1
...... /! !?         &^1 ^2^3
...... /!<?          &^1<^2
...... /!]           &^1]
...... /!@?          &^1 ^2
...... /EX           &EXIT
...... /ST!?         &START@^1^2
...... /A !?         &ARRANGE ^1^2
...... /#,?=?        &#,^1=^2
...... /RV!?         &RESERVE^1^2
...... /RT?          &RESET^1
...... /RN           &RETURN
...... /FN?          &FINISH ^1
...... /C?           &CLOSE ^1
...... /FL?          &FILE ^1
...... /M.?,?        &MACRO NO ^1 NEXT ^2
...... .?            ^1
```

A TANULMÁNYOK sorozatban eddig megjelentek:

1/1973 Pásztor Katalin: Módszerek Boole-függvények minimális vagy nem redundáns, {∧, ∨, ¬} vagy {NOR} vagy {NAND} bázisbeli, zárójeles, vagy zárójel nélküli formuláinak előállitására

2/1973 Вашкеви Иштван: Расчленение многосвязных промышленных процессов с помощью вычислительных машин

3/1973 Ádám György: A számitógépipar helyzete 1972 második felében

4/1973 Bányász Csilla: Identification in the Presence of Drift

5/1973* Gyürki J.-Laufer J.-Girnt M.-Somló J.: Optimalizáló adaptiv szerszámgépirányitási rendszerek

6/1973 Szelke E.-Tóth K.: Felhasználói Kézikönyv /USER MANUAL/ a Folytonos Rendszerek Szimulációjára készült ANDISIM programnyelvhez

7/1973 Legendi Tamás: A CHANGE nyelv/multiprocesszor

8/1973 Klafszky Emil: Geometriai programozás és néhány alkalmazása

9/1973 R. Narasimhan: Picture Processing Using Pax

10/1973 Dibuz Á.-Gáspár J.-Várszegi S.: MANU-WRAP hátlaphuzalozó, MSI- TESTER integrált áramköröket mérő, TESTOMAT-C logikai hálózatokat vizsgáló berendezések ismertetése

11/1973 Matolcsi Tamás: Az optimum-számitás egy uj módszeréről

12/1973 Makroprocesszorok, programozási nyelvek. Cikkgyüjtemény az NJSzT és SzTAKI közös kiadásában. Szerkesztette: Legendi Tamás

13/1973 Jedlovszky Pál: Uj módszer bonyolult rektifikáló oszlopok vegyészmérnöki számitására

14/1973 Bakó András: MTA kutatóintézeteinek bérszámfejtése számitógéppel

15/1973 Ádám György: Kelet-nyugati kapcsolatok a számitógépiparban

16/1973 Fidrich I.-Uzsoky M.: LIDI-72 listakezelő rendszer a Digitális Osztályon, 1972. évi változat

17/1974 Gyürki József: Adaptiv termelésprogramozó rendszer /APS/ termelőmühelyek irányitására

18/1974 Pikler Gyula: MINI-számitógépes interaktiv alkatrészprogramiró rendszer NC szerszámgépek automatikus programozásához

19/1974 Gertler J.-Sedlak J.: Software for process control

20/1974 Vámos T.-Vassy Z.: Industrial Pattern Recognition Experiment - A Syntax Aided Approach

21/1974 A KGST I. - 15-1.: "Diszkrét rendszerek automatikus vezérlése" c. témában 1973. februárban rendezett szeminárium előadásai

22/1974 Arató M.-Benczur A.-Krámli A.-Pergel J.: Stochastic Processes, Part I.

23/1974 Benkó S.-Renner G.: Erősen telitett mágneskörök számitógépes tervezési módszere

24/1974 Kovács György-Franta Lászlóné: Programcsomag elektronikus berendezések hátlaphuzalozásának tervezésére

25/1974 Járdán R. Kálmán: Háromfázisu tirisztoros inverterek állandósult tranziens jelenségei és belső impedanciája

26/1974 Gergely József: Numerikus módszrek sparse mátrixokra

27/1974 Somló János: Analitikus optimalizálás

28/1974 Vámos Tibor: Tárgyfelismerési kisérlet nyelvi módszerekkel

29/1974 Móricz Péter: Vegyészmérnöki számitási módszerek fázisegyensulyok és kémiai egyensulyok vizsgálatára

30/1974 Vassy Z. - Vámos T.: The Budapest Robot - Pragmatic Intelligence

31/1975 Nagy István: Frekvenciaosztásos középfrekvenciás inverterek elmélete

32/1975 Singer D., Borossay Gy., Koltai T.: Gázhálózatok optimális irányitása különös tekintettel a Fővárosi Gázmüvek hálózataira

33/1975 Vámos T.-Vassy Z.: Limited and Pragmatic Robot Intelligence

Mérő L.-Vassy Z.: A Simplified and Fastened Version of the Hueckel Operator for Finding Optimal Edges in Pictures

Галло В.: Программа для распознавания геометрических образов, основанная на лингвистическом методе описания и анализа геометрических структур

34/1975 László Nemes: Pattern Identification Method for Industrial Robots by Extracting the Main Features of Objects

35/1975 Garádi-Krámli-Ratkó-Ruda: Statisztikai és számitástechnikai módszerek alkalmazása kórházi morbiditás vizsgálatokban

36/1975 Renner Gábor: Elektromágneses tér számitása nagyhőmérsékletü anyagban

37/1975 Edgardo Felipe: Specification problems of a process control display

38/1975 Hajnal Andrásné: Nemlineáris egyenletrendszerek megoldási módszerei

39/1975 A.Abd El-Sattar: Control of Induction motor by three phase thyristor connections in the secondary circuit

40/1975 Gerhardt Géza: QDP Grafikus interaktiv szubrutinok a CDC 3300-GD'71 grafikus konfigurációra

41/1975 Arató M.-Benczur A.-Krámli A.-Pergel J.: Stochastic Processes, Part II.

42/1975 Arató Mátyás: Fejezetek a matematikai statisztikából számitógépes alkalmazásokkal

43/1975 Matavovszky Tibor - dr Pásztorné Varga Katalin: Programrendszer Boole-függvényrendszer együttes egyszerüsitésére vagy minimalizálására

44/1975 Bacsó Nándorné: Pneumatikus áramköri hazardok

45/1975 Varga András: Ellenpárhuzamos félvezetőpárokkal vezérelt aszinkronmotoros hajtások számitási módszerei

46/1976 Galántai Aurél: Egylépéses módszerek lokális hibabecslései

47/1976 Abaffy József: A feltétel nélküli függvényminimalizálás kvadratikus befejezésü módszerei

48/1976 Strehó Mária: Stiff tipusu közönséges differenciálegyenletek megoldásáról

49/1976 Gerencsér László: Nemlineáris programozási feladatok megoldása szekvenciális módszerekkel

50/1976 Treer Róbert: A syntax macro definition language.

51/1976 Bakó András: TIMER időredukciós programcsomag

52/1976 W.A. Potas: Computer Aided Design

Jelen dolgozat a 8.3.E Software készitési eszközök téma keretében készült.

A x-gal jelölt kivételével a sorozat kötetei megrendelhetők az Intézet könyvtáránál /Budapest, XIII. Victor Hugo u. 18-22./