A FORTRAN-PZ programozási nyelv

egyetemi doktori értekezés

Készitette:

PUSKÁS ALBERT

JÓZSEF ATTILA TUDOMÁNYEGYETEM TERMÉSZETTUDOMÁNYI KARA

1974.

B e v e z e t é s

A dolgozat a FORTRAN nyelv egy változatát, a FORTRAN - PZ nyelvet definiálja. A PZ jelet a "Programozási nyelv zászlós-ábrákkal való definiálása" rövidítésére használjuk. Célunk - a változat kidolgozásával - az volt, hogy megadjuk egy olyan programozási nyelvet, mely véleményünk szerint elegendő arra, hogy a számitástechnikai alapképzésben résztvevők - számitógéppel csak közvetett kapcsolatban lévők - számára egy programozási nyelv felépitése, szerkezete ismertté és könnyen elsajátithatóvá váljon.

E cél érdekében - az 1. pontban - röviden áttekintjük a programozási nyelvek kialakulását, fejlődését. Rámutatunk viszonylagos bonyolultságukra, melyekből adódik azon nem csekély hátrányuk, hogy nehezen tanulhatók és tanithatók.

A 2. pontban megkisérelјük összefoglalni azon indokainkat, melyek szükségessé tették a FORTRAN-PZ nyelv megkonstruálását, létrehozását. Ezen indokainkat elsősorban a megtanulhatóság, a megtanithatóság főproblémája köré csoportositjuk, figyelembe véve, hogy a 70-es években a számitástudomány már annyira része ill. eszköze az alkalmazott tudományoknak, hogy a leendő felhasználók a közép - vagy alsóbb foku iskolákban megtanulhatják a szükséges alapismereteket éppugy, mint a matematikai alapokat, tehát az ujabb generációknak egyáltalán nem lehet idegen egy egyszerü programozási nyelv notációja és megfogalmazás módja. [12].

Kalmár László akadémikus "An intuitive representation of context - free languages" cimü cikkében [4] hangsulyozza, hogy egy programozási nyelv szintaxisának megadására - az általa használt - zászlós ábrák előnyösen használhatók oktatói segédeszközként a programozási nyelvek tanitásában. Ezzel egyetértve, a FORTRAN-PZ nyelv szintaxisát mi is zászlósábrákkal adjuk meg, igy a 3. pontban a zászlós-ábrázolási módról szólunk.

A 4. pontban adjuk meg a nyelv definicióját, szabályait. Mivel ez a változat része az ASA FORTRAN nyelvnek, igy a két nyelv eltérései megfogalmazhatók, mint az ASA FORTRAN megszoritásai, igy ezen eltérésekre is e pontban mutatunk rá.

Végül az 5. pont tartalmazza - a 4. pont függelékeként - a FORTRAN-PZ nyelv szintaxisát leiró zászlósábrákat s a kategóriák jegyzékét.

1. A programozási nyelvek fejlődésének rövid áttekintése

"A programozási nyelvek - hasonlóan minden más nyelvhez - az információátvitel célját szolgálják valamely adó és vevő között (természetesen kódolt formában). A programozási nyelvek esetében az adó a feladatot leiró ember, a vevő pedig a feldolgozást végző elektronikus számitógép. Ez a körülmény a programozási nyelvek számos sajátosságát meghatározza, és egyben korlátozza lehetőségeiket. Mivel adóként az ember szerepel, célszerü lenne az emberéhez minél közelebb álló nyelvet alkalmazni az információátvitelre. Ugyanakkor biztositani kell, hogy a vevő (a számitógép) képes legyen dekódolni és értelmezni minden számára szóló kódolt információt. Ez a két követelmény némiképpen ellentmond egymásnak, minthogy a gép számára közvetlenül érthető nyelv (saját belső kódrendszere) igen távol áll az emberi nyelvektől" [1].

A számitógépek programozásának első időszakában a programok felirása a gép kódrendszerében történt. Ez a tény erősen akadályozta alkalmazhatóságukat és elterjedésüket egyaránt annak ellenére, hogy a gépi kódrendszerek is jelentős fejlődésen mentek keresztül. Igény merült fel tehát olyan közvetitő nyelvek iránt, melyek közelebb állnak a természetes nyelvekhez; valamint az iránt, hogy a számitógépeket alkalmassá lehessen tenni ilyen közvetitő nyelv-(ek)-en közölt információk

megértésére.

Ezen cél elérése érdekében az lett volna várható, hogy az ember és a gép közötti kommunikálás eszköze egyetlen "világnyelv" lesz, de egyáltalán nem ez történt, hanem meglepően nagyszámu mesterséges gépi nyelv fejlődött ki, amelyek között vannak világnyelvek és vannak szerény dialógusok [12].

A szerteágazó programozási nyelvek rövid áttekintését az alábbi csoportositás szerint végezzük el:

a) gépi szintü programozás,

b) assembly nyelvek,

c) speciális programozási nyelvek

i) eljárás - orientált nyelvek,

ii) probléma - orientált nyelvek,

d) általános célu programozási nyelvek.

a) Gépi szintü programozás

A számitógépek programozása megjelenésüktől kb 1952-ig közvetlenül gépi kódjukban történt. Ez a tény leszükitette alkalmazási területüket s hátráltatta elterjedésüket is. A gépi kódban való programirás megnehezitette a programozók munkáját, a számitógépeket alkalmazni kivánók problémájukat ily módon nem tudták megfogalmazni, végül egyaránt nehézséget támasztott a kutatási eredmények cseréjében is.

Előre lépést jelentett ezen időszak végén a szubrutinmódszer kidolgozása, mely tulajdonképpen rögzitett utasitás-sorozatokat jelentett és ezeket a számitások végrehajtása során nevükre való hivatkozással lehetett

alkalmazni.

b) Assembly nyelvek

A szubrutinok használatának megkönnyitésére irányuló törekvések vezettek el az assembly nyelvek kialakulásához. Kialakulásukhoz vezetett az a tény is, hogy tulajdonképpen a programozók szinte sohasem használták a programok felirásánál a gépi kódokat, hanem a müveleti kódokat mnemonikus kódokkal, a valódi cimeket szimbolikus illetve relativ cimekkel helyettesitették; az olyan utasitások megjelölésére pedig, melyekre más utasitásokban hivatkoztak cimkéket használtak.

Az assembly nyelvekben használt szimbolikus jelölésmód nagy mértékben megkönnyitette a programozást, de megnövekedett a számitógép munkája. A szimbolikusan irt programot ugyanis át kellett kódolni a gép belső nyelvére és az elrendezést (a cimzést) is végre kellett hajtani. Igy kifejlődésük csak akkor vált általánossá, amikor lehetőség nyilott az átkódolási munkák automatizálására. Bár az assembly nyelvek csak igen egyszerü programozási nyelvek voltak, az utasitásokat strukturálisan nem változtatták meg, szerepük mind a mai napig megmaradt.

A fenti nyelv-tipusok még igen szoros kapcsolatban voltak a gép utasitásrendszerével, a bennük használt program irásmód a gépikód-programozás kényelmetlen jelölésrendszeréhez alkalmazkodott, ezért jelentős lépésnek számitott a programozásban a gépi utasitás-

strukturától eltérő formákat is megengedő, a matematikai jelölésmódhoz közelebb kerülő programozási nyelvek megjelenése.

c) Speciális programozási nyelvek

i) Eljárás-orientált nyelvek

Az eljárás-orientált nyelvek körébe azokat a nyelveket soroljuk, melyek végrehajtandó eljárásokat irnak le a feldolgozandó adatokra és az alkalmazandó eljárásokra vonatkozó fogalmak és eszközök segitségével [1].

Kialakulásuk az ötvenes évek első felére tehető, de a hatvanas évekig a kidolgozott programozási nyelvek többnyire a konkrét géphez kapcsolódóak, gépi orientáltságuak voltak. Bár világos volt a kidolgozók előtt a nyelvek információtranszformátor szerepe és a számitógépnek másodlagos jelentősége, mégis a matematikai nyelvészeti szempontokat, eszközöket csupán a terminológiák kölcsönvétele erejéig alkalmazták.

A matematikai nyelvészeti eszközök fokozott figyelembe vételekor alakulhattak ki az eljárás-orientált nyelvek. Az elsősorban numerikus és algoritmizálható feladatok végrehajtására alkalmas programozási nyelvek közül legelterjedtebbek a FORTRAN (1957) és az ALGOL-60 (1960).

A programozási nyelveken felirt programok adott jelkészletből (alapjelekből) meghatározott szabályok szerint létrehozott egységek (szavak, kifejezések, mondatok) összessége, amely egységek a nekik tu-

lajdonitott értelmet hordozzák. A nyelv azon szabályait, amelyek alapján helyes programokat irhatunk a nyelv szintaxisának; amelyek pedig megmondják, hogy egy szintaktikusan helyes program milyen akciók végrehajtását jelenti a nyelv szemantikájának nevezzük.

Az emlitett két nyelv között mind jelkészletükben, mind szabályaikban lényeges különbségek, de ugyanakkor hasonlóságok is vannak. A ma általánosan használt FORTRAN (1964) lényeges módositásokkal jött létre az ALGOL-60 hatására. Nem célunk sem a két nyelv egymással való összehasonlitása, sem részletes taglalásuk, csupán kiragadunk néhányat előnyeik - illetve hátrányaik közül.

Mindkét nyelvben az aritmetikai és logikai kifejezések a matematikában szokásos jelölésmód szerint épülnek fel. Az ALGOL-60 előnye, hogy fejlett ciklusszervező, eljárás- és függvényeljárás-utasitásokkal, sajátos blokkstrukturával rendelkezik; hátránya az input-output (I/O) utasitások hiánya. Az 1962. évi módositott ALGOL jelentés ad szabványos I/O eljárásokat, de ezek is elsősorban numerikus feladatokra és nem adatfeldolgozásra (rekordok, file-ok) alkalmasak. A FORTRAN lényeges előnyeiként a változók automatikus deklarálása és a fejlett, szabványos I/O rendszerek; hátrányaiként pedig a feltételes kifejezések hiánya és a kevésbé fejlett ciklusszervező, eljárás- és függvényeljárás-utasitások emlithetők.

A numerikus problémák megoldására szolgáló nyel-

vek mellett kialakultak az ugynevezett szimbólum-manipulációs nyelvek is a hatvanas évek során. Ezek szimbolikus adatokon végzett müveletek, eljárások (forditóprogramok, szövegforditás és elemzés, mesterséges inteligencia vizsgálatok, stb.) leirására szolgálnak. Ilyenek pl. a listafeldolgozás céljára kidolgozott LISP - (1960) nyelvek; a gépi forditás céljaira kidolgozott COMIT - (1957) és SNOBOL - (1964) nyelv, valamint a folyamatirányitási feladatok megoldására szolgáló INDAC-nyelv.

ii) Probléma-orientált nyelvek

A probléma-orientált nyelvek körébe azokat a nyelveket soroljuk, melyek a megoldandó problémát annak terminológiájával, kifejezéseivel és utasitásaival irják le. Elsősorban a probléma megfogalmazására szolgálnak és nem a megoldási algoritmus közlésére. Pl. egy program során elegendő egy adott tipusu differenciálegyenletet leirni, a megoldást egy beépitett algoritmus (pl. Runge-Kutta módszer) szerint keresi meg a program. Közülük egyet - a gazdasági, adatfeldolgozási területen jól alkalmazható - COBOL-nyelvet emlitjük meg. Megalkotóit kettős cél vezette. Egyrészt, hogy a nyelv szinte korlátlan lehetőségeket adjon a bonyolult adatstrukturák felépitésére, mozgatására, valamint az adattároló, be- és kiviteli eszközök kezelésére; másrészt, hogy a nyelv az élő nyelvhez közeli alaku legyen, mivel általában az adatfeldolgozási munkákat alacsonyabb képesitésü emberek végzik. Éppen ez utóbbi cél miatt

is e nyelv (nyelvek) igen távol áll a gépi nyelvektől, igy gépre forditásuk is komplikáltabb s a növekvő igények ellenére sem terjedt el kellően.

d) Általános célu programozási nyelvek

Az előzőekben tárgyalt nyelvek megalkotásánál bár cél volt az általánosságra való törekvés, mégis ezek a probléma megoldásoknak csak egy-egy meghatározott területén váltak célszerüen alkalmazhatóvá. Ezért a számitástechnikai tapasztalatok felhalmozódásával ujabb igények és lehetőségek merültek fel a különböző célu programozási nyelvek szintézisének megteremtésére.

Általános célu programozási nyelv kidolgozására való törekvés ismét több "helyen" folyt, más-más szempontok figyelembe vétele mellett. Általános célu programozási nyelvek elsősorban az eljárás-orientált nyelveket kisérlik meg általánositani. Probléma-orienált nyelveknél ilyen irányu általánositás nem várható. Itt három általános célu nyelvet emlitünk, az IBM kezdeményezésére a PL/1, az IFIP kezdeményezésére az ALGOL-68 és a speciális felhasználási üzemmódra készült APL.

A PL/1 nyelv a numerikus és adatfeldolgozási feladatok végrehajtására szolgál, igy tulajdonképpen a FORTRAN, az ALGOL-60 és a COBOL programozási nyelvek kombinációjának tekinthető. Célszerüen egyesiti e három nyelv előnyeit, és küszöböli ki hátrányaikat. Jelenleg a legelterjedtebb programozási nyelv. Előnye a nyelvnek, hogy három önálló un. subset-ből tevődik össze: numerikus feladatok, adatfeldolgozási feladatok és

karaktermanipulációs feladatok megoldására alkalmas részekből. Igy az egyes felhasználóknak elegendő feladatuktól függően egyik vagy másik rész elsajátitása, megtanulása. Elterjedésében ez a tény igen nagy szerepet játszott ill. játszik.

Az ALGOL-68 ma a legáltalánosabb nyelv, tulajdonképpen mindent tud, amit a hetvenes évek számitástechnikája és számitástudománya megkövetelhet. Kidolgozóit azon cél vezette, hogy segitségével könnyen leirhatók legyenek a különféle algoritmusok és a leirt algoritmusokat a legkülönbözőbb számitógépen végre lehessen hajtani. Az ALGOL-68 az információ-feldolgozás általános fogalmain alapuló uj nyelv, nem pedig valamely korábbi nyelv kiterjesztése. Az ALGOL-60 nyelvtől vett fogalmakat magasabb szinten tartalmazza. A gyakorlati használatban való elterjedését éppen tul általános volta, bonyolultsága akadályozza.

Az APL nyelv elsősorban a felhasználók igényének kielégitését szolgálja. A nyelv a felhasználók által használt irásmódhoz közelálló notációt alkalmaz, ezek könnyen elsajátithatók és egyszerü formában közölhetők a számitógéppel, ugyanakkor nagy teljesitményü rendszerként is alkalmazható. A nyelv utasitásai definiciós és végrehajtási üzemmódban használhatók. A végrehajtási üzemmódban a programozó által közölt utasitások azonnal végrehajtódnak. Bár e nyelv un. dialógus nyelv, mégis kevésbé elterjedt, mint a PL/1, mivel használata terminál-rendszerhez és nagy teljesit-

ményü központi számitógéphez kötött.

Végül megemlitjük a programozási nyelvek várható két fejlődési irányát. Egyik irány a természetes nyelvekhez való közelités, mig a másik az általános célu (univerzális) nyelvek magasabb szinten való továbbfejlesztése.

2. A FORTRAN-PZ nyelv megalkotásának indokai

A FORTRAN nyelv változatának kidolgozásához elsősorban oktatási tényezők vezettek el. Az alábbiakban ezekről szólunk.

A számitástechnikai oktatás során a gépi programozás alapjául nyilván a gépi kódu programozás szolgál. A programozáshoz szükséges alapfogalmak, fogalmak, program készitési módok ismertetése itt válik lehetővé. Az egyszerü programozás azonban a gépi utasitásrendszerrel szinte lehetetlen. A gépi kódu utasitások formája igen messze áll a hagyományos irásmódtól: a számitások során hagyományosan algebrai kifejezéseket használunk, a gépi kódu utasitások inkább a függvény-irásmódot alkalmazzák (a müveleti kód a függvény-név, a cimek az argumentumok); vagy mig a hagyományosan irt számitási eljárás kifejezések (több müveleti jel és zárójel használatával) értékének kiszámitásával hajtódik végre, addig a gépi kódu program egy-egy müveleti jelnek megfelelő utasitásokra bomlik. Mindezek ellenére a gépi programozás oktatásában a gépi kódu programozás elengedhetetlen. Véleményünk szerint e hátrányok kiküszöbölése a mnemonikus irásmód segitségével némileg lehetséges. Egyetemi és főiskolai előadások során sokkal hatékonyabbnak véljük a fiktiv gépi utasitásrendszerrel való programirást, mint valamely konkrét gépi utasitásrendszerével. Bár több egyetem és főiskola az utóbbi utat választja,

a tanárképző főiskolákon - elsősorban a szegedin - mi mégis a Kalmár-féle fiktiv gép utasitásrendszerének egy speciális változatát alkalmaztuk az 1963-ban tantervbe iktatott "A matematika modern alkalmazásai" c. tantárgy előadásai során.

Ma már - a hetvenes években - nem állhat meg a gépi programozás oktatása - még az alapfoku képzésben sem - a gépi kódok szintjén, szükség van egy magasabb szintü programozási nyelv oktatására is.

Az 1. pontban ismertetett programozási nyelvek programirásmódja már lényegesen közelebb került a hagyományos irásmódhoz. A programozási nyelvekben felhasznált konstansok, operanduszok, változók, alaputasitások és vezérlő utasitások formája és strukturája a szokásos matematikai formához és strukturához már nagyon hasonló, de különösen strukturális felépitésében még lényegesen különböző is.

Az előbb emlitett programozási nyelvi elemeken kivül vannak a programozási nyelvekben olyan általános szabályok, utasitás-csoportok, technikai fogások, amelyek egyrészt a programozó munkáját könnyitik meg, másrészt a számitógépet látják el a végrehajtáshoz szükséges információkkal. Vannak továbbá az egyes nyelveknek - saját céljaira fenntartott - különleges jelentésü kulcsszavai, melyek általában angol szavak, ezek a nyelv alapjelei közé tartoznak és jelentésüket rendszerint magukban hordozzák. Ezen szabályok - és a kulcsszavak is - a programozási nyelveket idegen-

szerüvé, bonyolulttá, nehézkessé teszik a megértés szempontjából és igy formalizmusuk ismét távol kerül a matematikai jelölésmódtól.

Ezek miatt az okok miatt is szükségességét véltük - a programozási nyelvek alapelemeinek szükebb megtartása mellett - egy könnyebben megérthető, könnyebben megtanulható és könnyebben oktatható nyelv kidolgozását.

Másik - nem kevésbé fontos - indokunk a számitástechnikai program megvalósitásának következménye.

A számitástechnikai program szerinti szakemberképzés intézményes oktatási részét a Müvelődésügyi Minisztérium vállalta magára. A képzés koncepciójának, célkitüzéseinek és programjának kidolgozására Számitástechnikai Bizottságok jöttek létre. Az MM Pedagógusképző Osztálya által létrehozott Albizottság dolgozta ki a pedagógusképző intézményekre - igy a tanárképző főiskolákra is - a számitástechnikai képzés irányelveit. Ezen irányelveket meghatározta az a tény, hogy a végzett matematika szakos hallgatók fogják oktatni az általános iskolákban azt a matematikát, mely hivatott arra is, hogy a tanulók a számitástechnika elemeivel megismerkedjenek. Igy hallgatóinkat a főiskolákon - az irányelvek szerint - általános alapképzésben kell részesiteni, általános alapképzésen értve "olyan számitástechnikai ismeretanyag közlését, melyre támaszkodva a népgazdaság legkülönbözőbb területein dolgozó szakemberek szakmai tevékenységüket a számitástechni-

ka adta lehetőségek ismeretében tudják elvégezni" [MM Számitástechnikai Irányelvei. 1970.] . Az irányelveket két dolog determinálta: egyrészt az általános iskolai számitástechnikai képzés anyaga, másrészt a főiskola uj tanterve adta lehetőségek [10].

Az emlitett irányelvek alapján az 1970/71. tanévtől bevezetett uj főiskolai tantervben a matematika szakos hallgatók számára egy, a számitástechnikai alapképzést szolgáló uj tárgy - a "Numerikus és gépi módszerek" c. tárgy - került bevezetésre. E három féléves tárgy programja - többek között - előirja egy programozási nyelv ismertetését, szabad választást engedve az ALGOL és a FORTRAN között [9]. A szegedi Tanárképző Főiskolán két éven át a FORTRAN - az 1973/74. tanévben, mivel a módositott tanterv miatt a tárgy oktatása két évfolyamon is folyt, pedig mindkét - nyelv került előadásra. Az ezen anyagrészre szánt kevés óraszám (előadásra mintegy 18-20 óra, gyakorlatra pedig 8-10 óra áll rendelkezésre) indokolja, hogy nem kerülhet sor egy programozási nyelv teljes ismertetésére. Az előadásokon, a gyakorlatokon és a vizsgákon szerzett többéves tapasztalat összegyüjtésével, felhasználásával hoztuk létre e szükitett változatot, mely véleményünk szerint a fent emlitett hallgatóság - de az alapképzésben bárhol részt vevők - számára elegendő egy programozási nyelv elemeinek és struktúrájának megismerése céljából.

Az uj tanterv egy másik lehetőséget is ad a hall-

gatók számitástechnikai képzésére a tantervben szereplő kötelezően választható kollégiumok keretén belül. E két féléves tárgyak időben követik a már emlitett Numerikus és gépi módszerek három félévét. A kötelezően választható kollégiumok számitástechnikával foglalkozó része igy lehetővé teszi, hogy egy programozási nyelv megadását metanyelvi módszerrel is tárgyalhassuk. Ilyen kollégiumok tananyagába is beépithető e szükitett változat. Bár ennek oktatási sikerességére vonatkozó helyi tapasztalataink még nincsenek - hiszen egy ilyen kollégiumot csak az 1974/75. tanévre hirdettük meg (az 1973/74. tanévvégi maximális érdeklődés már is sikernek tekinthető) - azonban figyelembe véve a JATE matematikus képzése során szerzett tapasztalatokat dolgoztuk ki a nyelv szintaxisát zászlós-ábrák segitségével. Erről az 1974-ben rendezett visegrádi - számitástechnikai oktatással kapcsolatos - konferencián részben beszámoltunk [11].

Végül választ kivánunk adni arra a kérdésre, hogy miért éppen a FORTRAN programozási nyelvet választottuk az oktatásra szánt programozási nyelv kidolgozásának alapjául.

Egyrészt válaszul szolgálnak erre a kérdésre az előző bekezdésben már érintett oktatási tapasztalatok, másrészt az ALGOL nyelv középiskolás szinten történő, valamint a FOKAL nyelv alapismereteinek tanitására már ismeretesek kisérletek és törekvések [5]. Természetesen ismertek a FORTRAN nyelvnek is szükitett vált-

zatai [7, 8], de ezek inkább gépi (TPA) specialitásoknak tekinthetők.

Mindezek mellett egyik legfőbb érvünk a FORTRAN programozási nyelven irt programok blokkséma struktúrájában rejlik. A mi hallgatóink képzésének végső célja az általános iskolai tanári pályára való felkészités. Az általános iskolában a korszerü matematikaoktatásban a számitástechnikai képzés legfőbb célja pedig, hogy a matematikai feladatok kapcsán a tanulók betekintést kapjanak az algoritmikus gondolkodásmódba. Az algoritmusok szemléltetésére legjobban pedig a folyamatábrák, blokksémák szolgálnak. Célunk elérésénél tehát döntő szerepet játszanak a számitási algoritmusok folyamatábrái. A változatban megtartva a FORTRAN programok "egy utasitás egy sor" programirásmódját a folyamatábráról a program könnyen felirható, és viszont a felirt programok könnyen olvashatóvá válnak.
Utóbbi érvünk szemléltetésére bemutatunk néhány példát:

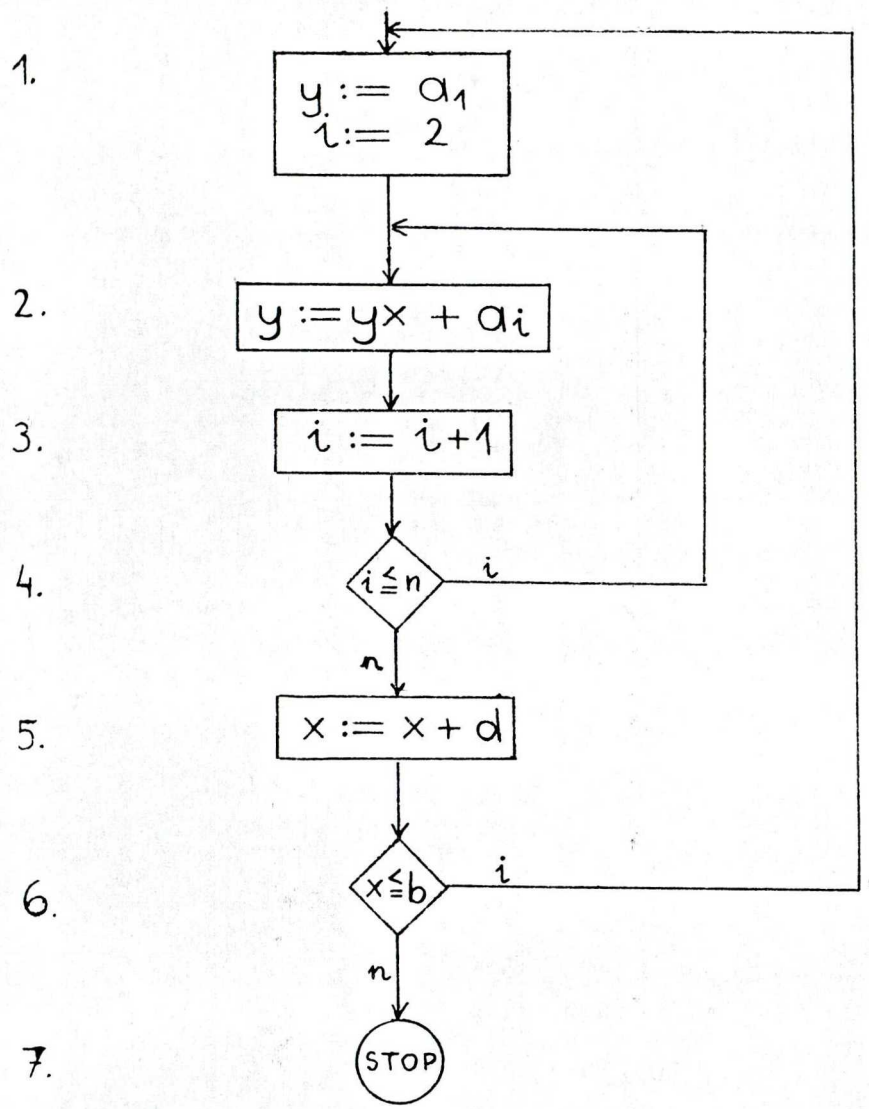
i) Legyen adva egy

$$y = \sum_{i=1}^{n} a_i \, x^{n-i} \qquad (n > 1, \text{ egész szám})$$

polinom és számitsuk ki az [x, b) intervallumon d lépésközzel felvett helyettesitési értékeit. Rajzoljuk fel a számitási algoritmus folyamatábráját és irjuk fel FORTRAN-PZ nyelven programját (a programirásnál a deklaráló, input-output utasitásoktól eltekintünk).

Alkalmazva az ismert Horner-elrendezést, a számitás

folyamat ábrája a következő lehet (az egyes blokkokat sorszámmal láttuk el):



A számitásnak megfelelő programot két módon is megadjuk: egyik csak IF utasitásokat használ fel, a másikban már DO ciklusszervezés is előfordul. A felirt programokban a megfelelő nagy betüket használtuk azo-

nositóknak, a megjegyzés rovatban a folyamatábra blokkjait tüntettük fel. A második program az összetettebb utasitás (DO utasitás) felhasználása miatti blokkösszevonódást is szemlélteti. (l. a következő oldalt.)

FORTRAN	JUHÁSZ GYULA TANÁRKÉPZŐ FŐISKOLA MATEMATIKAI TANSZÉKE SZEGED	PROGRAM: POLINOM HELYETTESÍTÉSI ÉRTÉKÉNEK KISZÁMÍTÁSA

KÉSZITŐ:

	CIMKE	UTASITÁSOK	MEGJEGYZÉS	
	1	6 7 72	73 80	
C		IF UTASITASSAL		1
	4	Y = A(1)	1 BLOKK	2
		I = 2	1 "	3
	1	Y = Y * X + A(I)	2 "	4
		I = I + 1	3 "	5
		IF(I - N) 1,1,2	4 "	6
	2	CONTINUE		7
		X = X + D	5 "	8
		IF(X - B) 4,4,3	6 "	9
	3	CONTINUE		10
		STOP	7 "	11
				12
				13
C		DO UTASITASSAL		14
	2	Y = A(1)	1 BLOKK	15
		DO 1 I = 2,N	1,3,4 "	16
		Y = Y * X + A(I)	2 "	17
	1	CONTINUE		18
		X = X + D	5 "	19
		IF(X - B) 2,2,3	6 "	20
	3	CONTINUE		21
		STOP	7 "	22
				23
				24
				25

Statisztikai problémák megoldása során igen gyakran van szükség a következő algoritmusra:

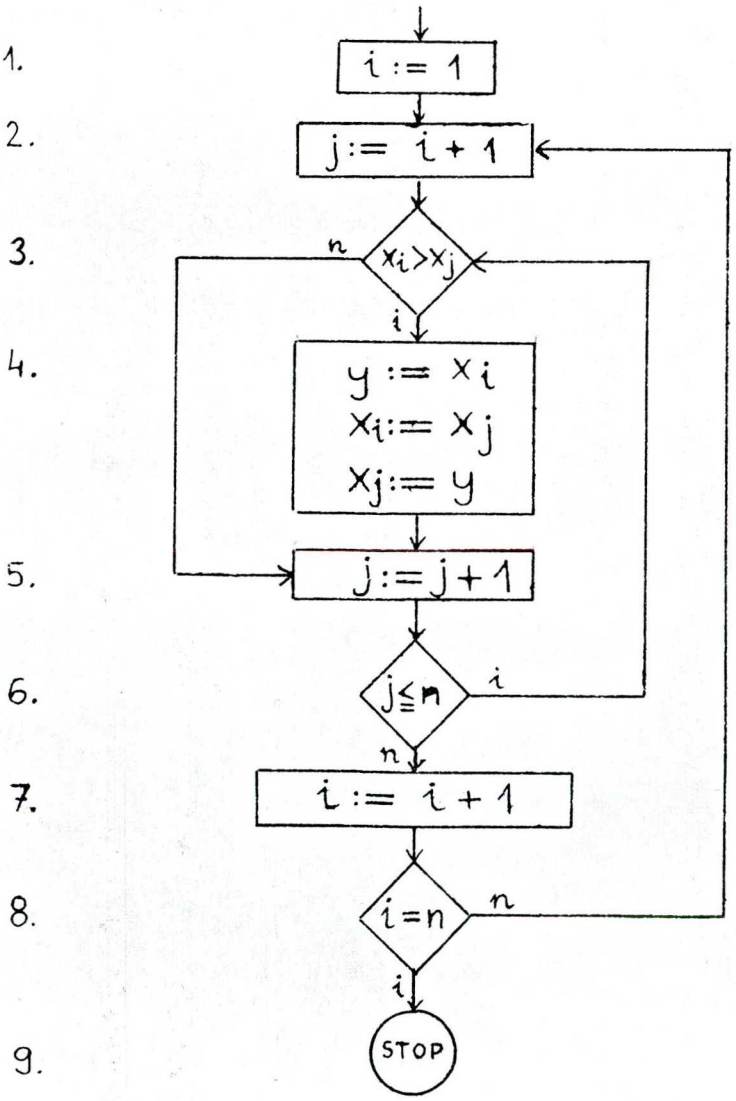
ii) adott $x_1, x_2, \ldots, x_n$ n elemü ($n > 2$) mintát rendezünk nem csökkenő sorozatba.

A feladat megoldására szolgáló algoritmus a következő lehet:

1) Ha $x_i > x_j$, ahol $j = i+1, i+2, \ldots, n$, akkor x_i-t felcseréljük x_j-vel, azaz $y = x_i$, $x_i = x_j$, $x_j = y$ cserét végrehajtjuk $j = i+1, i+2, \ldots, n$ esetén.

2) Az 1) alattiakat megismételjük $i = 1,2,\ldots,n-1$ esetén.

A leirt algoritmus blokksémája a következő lehet:



Irjuk fel az algoritmusnak megfelelő program-részletet, azonositóknak a megfelelő nagy betüket használva és a megjegyzés rovatban ismét feltüntetve a folyamatábra blokkjainak sorszámát. A program-részletet DO utasitással is felirva hasonló megjegyzést tehetünk, mint az előző példa során. (l. a következő oldalt.)

FORTRAN

JUHÁSZ GYULA TANÁRKÉPZŐ FŐISKOLA
MATEMATIKAI TANSZÉKE SZEGED

PROGRAM: MINTA ELEMEK RENDEZÉSE

KÉSZITŐ:

	CIMKE	UTASITÁSOK	MEGJEGYZÉS	
	1	6 7 72	73 80	
1	C	UTASITÁSSAL		1
2		I = 1	1 BLOKK	2
3	4	J = I + 1	2 "	3
4		IF(X(I).LE.X(J)) GOTO 3	3 "	4
5		Y = X(I)		5
6		X(I) = X(J)	4 "	6
7		X(J) = Y		7
8	3	J = J + 1	5 "	8
9		IF(J - N) 4,4,2	6 "	9
10	2	I = I + 1	7 "	10
11		IF(I - N) 4,5,5	8 "	11
12	5	CONTINUE		12
13		STOP	9 "	13
14				14
15	C	DO UTASITÁSSAL		15
16		M = N - 1	1,7,8	16
17		DO 1 I = 1, M	BLOKK	17
18		K = I + 1	2 "	18
19		DO 1 J = K, M	5,6 "	19
20		IF(X(I).LE.X(J)) GOTO 1	3 "	20
21		Y = X(I)		21
22		X(I) = X(J)	4 "	22
23		X(J) = Y		23
24	1	CONTINUE		24
25		STOP	9 "	25

iii) Legyen adva x_1, x_2, ..., x_n elemü minta ($n > 2$). Számitsuk ki a minta empirikus középértékét, szórás. és korrigált szórás négyzetét, valamint variációs együtthatóját.

Jelölje ezeket rendre $\bar{x}$, s, k és v, akkor kiszámitva az

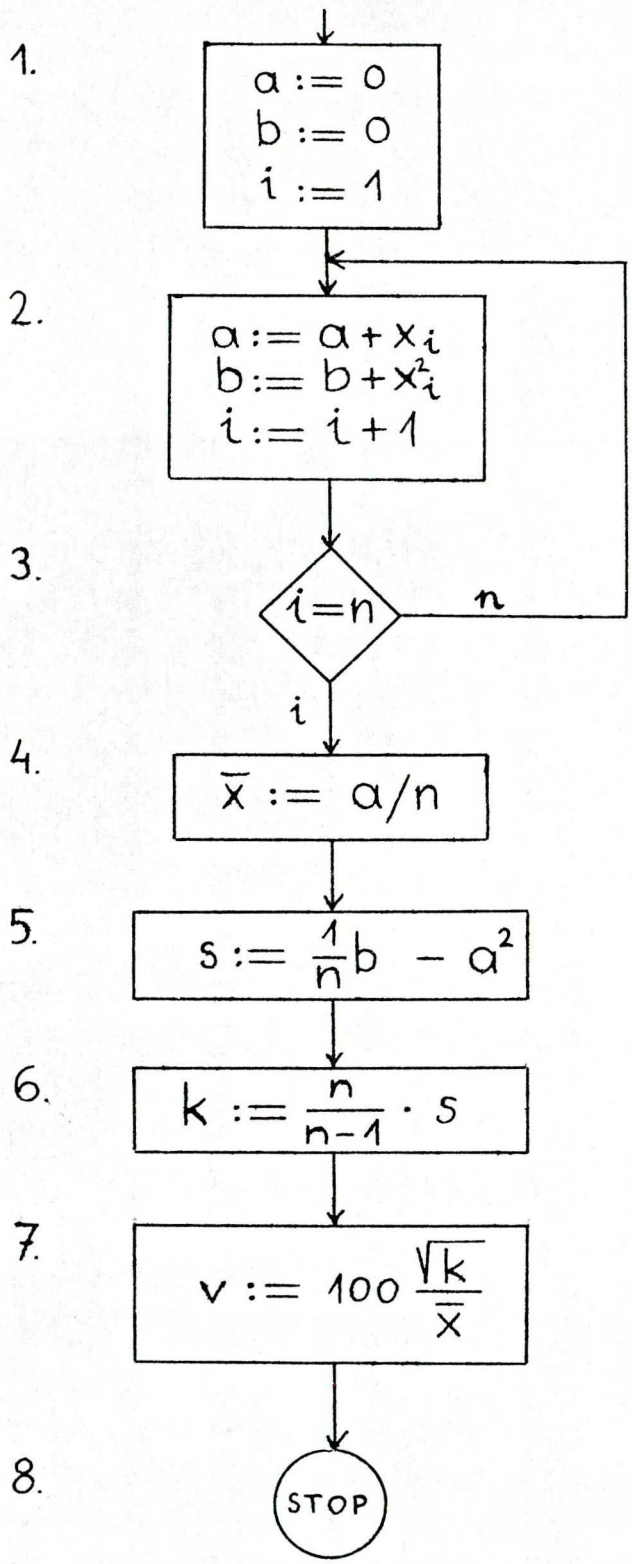
$$a = \sum_{i=1}^{n} x_i \qquad \text{és} \qquad b = \sum_{i=1}^{n} x_i^2$$

összegeket, adódik, hogy

$$\bar{x} = \frac{a}{n} \; ; \quad s = \frac{1}{n} b - a^2 ; \quad k = \frac{n}{n-1} \cdot s \quad \text{és}$$

$$v = 100 \frac{k}{\bar{x}} \cdot$$

Irjuk fel a számitás folyamatábráját és programrészletét:


1.
a := 0
b := 0
i := 1
2.
a := a + x_i
b := b + x_i^2
i := i + 1
3.
i=n
n
i
4.
x̄ := a/n
5.
s := 1/n b − a^2
6.
k := n/(n−1) · s
7.
v := 100 √k / x̄
8.
STOP

FORTRAN

JUHÁSZ GYULA TANÁRKÉPZŐ FŐISKOLA
MATEMATIKAI TANSZÉKE SZEGED

PROGRAM: MINTA KÖZEPE, SZÓRÁSA, KORRIGÁLT SZÓRÁSA, VARIÁCIÓS EGYÜTTHATÓJA

KÉSZITŐ:

CIMKE (1–5)	UTASITÁSOK (7–72)	MEGJEGYZÉS (73–80)
C	MINTA JELLEMZŐK	
	A = 0.0	1 BLOKK
	B = 0.0	
	I = 1	
1	A = A + X(I)	
	B = B + X(I)**2	2 "
	I = I + 1	
	IF(I - N) 1,1,2	3 "
2	X = A / N	4 "
	S = 1 / N * B - A**2	5 "
	K = N / (N - 1) * S	6 "
	V = 100.0 * SQRT(K) / X	7 "
	STOP	8 "

3. A kontextus-mentes nyelvek zászló-ábrázolásáról

A bevezetésben emlitettük, hogy Kalmár László akadémikus [4] tanulmánya alapján a FORTRAN-PZ nyelvet zászlós-ábrával adjuk meg. Ezért a dolgozat jelen pontjában összefoglaljuk a zászlósábrázoláshoz szükséges fogalmakat, és megadjuk azok értelmezését.

Mindenek előtt a kontextus-mentes nyelvek definiciójához jutunk el, majd leirjuk az ilyen nyelvek zászlós-ábrával való reprezentációját.

Először megadjuk a nyelvnek egy általános - általunk a továbbiakban használt - fogalmát:

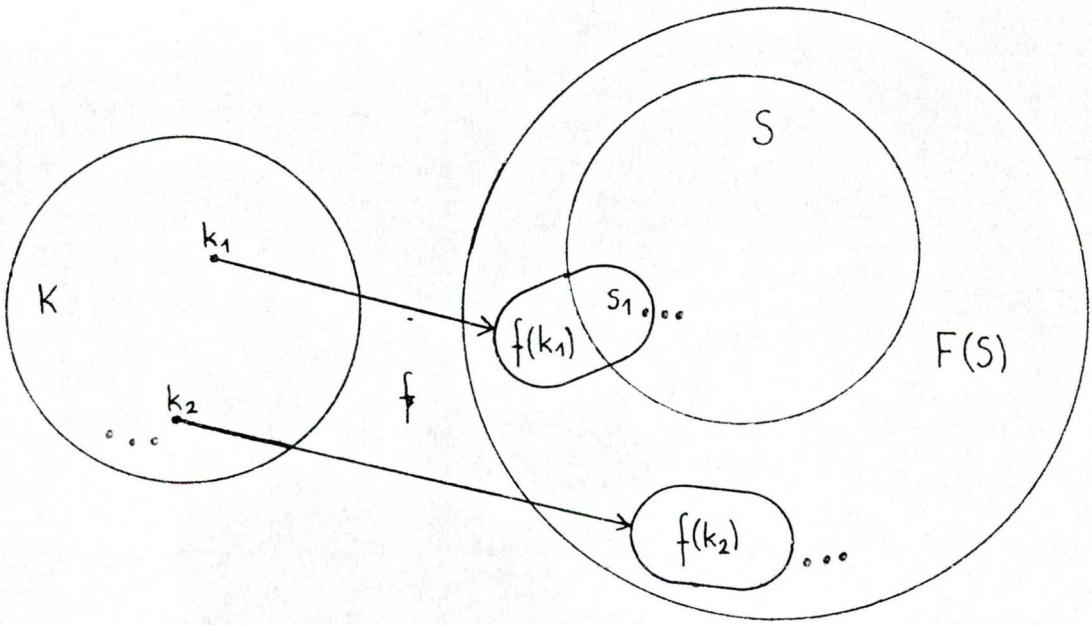
Definició: Egy L nyelv alatt egy $L = \{S, K, f\}$ rendezett hármast értünk, ahol

1) S és K idegen, nem üres, véges halmazok,

2) f pedig K egy leképezése az S által generált F(S) szabad félcsoport összes részhalmazainak halmazába.

Megjegyzés: az F(S) szabad félcsoport müvelete alatt a iuxtapozitio - (egymás után helyezés-)-t értjük.

A definiciót az alábbi ábra szemlélteti:



Bevezetjük a következő elnevezéseket:

1) Az S halmazt szókészlet (terminalis szótár)-nak, elemeit pedig szavaknak,

2) a K halmazt kategória-halmaznak, elemeit kategóriáknak nevezzük.

3) F(S) a szósorozatok (szóláncok) halmaza.

4) Végül ha valamely $k \in K$ kategóriára $f(k) \subseteq F(S)$ teljesül, akkor azt mondjuk, hogy f(k) a k kategóriához tartozó szósorozat (szósorozatok).

Ezekután a kontextus-mentes nyelv definiálásához szükséges néhány segédfogalmat vezetünk be.

Definició: egy G kontextus-mentes gramatikán egy $G = \{S, K, R\}$ rendezett hármast értünk, ahol

1) S és K idegen, nem üres, véges halmazok,

2) és R részhalmaza a $\{K \times F(S \cup K)\}$ halmaznak.

Az S és K halmazokat továbbra is szókészletnek

illetve kategória-halmaznak, az R halmazt pedig a szabályok-halmazának nevezzük. Az R halmaz $r = (k,\varphi)$ elemei (ahol $k \in K$ és $\varphi \in F(S \cup K)$) a szabályok. Egy $r = (k,\varphi)$ szabályt a továbbiakban $k:\varphi$ módon jelölünk.

Most már egy k kategória fogalom kifejtését a következő módon értelmezhetjük:

a) A k kategória közvetlen kifejtésének nevezzük az $F(S \cup K)$ halmaz egy φ elemét, ha $k:\varphi$ szabály.

b) A k kategória kifejtésének nevezzük az $F(S \cup K)$ halmaz egy φ elemét, akkor

1) ha φ közvetlen kifejtése k-nak, vagy

2) ha $\varphi = \varphi_1\varphi_2\varphi_3$ alaku, továbbá ha $\varphi' = \varphi_1 k' \varphi_3$ kifejtése k-nak és $k':\varphi_2$ szabály, akkor φ kifejtése k-nak.

c) Végül a k kategória terminális kifejtésén a k kategória azon kifejtését értjük, melynek elemei csak az $F(S)$ halmazból valók (azaz olyan sorozatok, melyeket csak szavakból képzünk).

A fenti segédfogalmak bevezetése után megadhatjuk a kontextus-mentes nyelv definicióját:

Akkor mondjuk, hogy egy $L = \{S, K, f\}$ nyelvet a $G = \{S, K, R\}$ kontextus-mentes gramatika generál, ha bármely $k \in K$ kategóriára $f(k)$ a k kategória összes terminalis kifejtésének halmaza.

Ezek után:

egy $L = \{S, K, f\}$ nyelv akkor kontextus-mentes, ha valamely kontextus-mentes gramatika generálja.

Most rátérünk a kontextus-mentes nyelvek zászlós-ábrázolására. Először a zászlós-ábra értelmezését adjuk meg, majd leirjuk a szükséges jelölésrendszert, végül megmondjuk, hogy egy nyelv hogyan reprezentálható velük.

Definició: Zászlós-ábrán egy $A = \{S,K,G,f_1,f_2,g_1,g_2\}$ rendezett hetest értünk, ahol

1) S és K idegen, nem üres, véges halmazok,

2) G egy véges, irányitott gráf,

3) f_1, f_2, g_1, g_2 pedig leképezések.

Jelölje C a G gráf csucspontjainak halmazát, és legyen C_1 és C_2 két idegen, nem üres részhalmaza C-nek. Ezek után f_1 és f_2 a C_1 és C_2 halmazok K-ra való leképezéseit jelentik.

Jelölje továbbá E a G gráf éleinek halmazát, és legyen E_1 és E_2 két idegen, E_1 nem üres, részhalmaza E-nek. Ezek után g_1 az E_1 halmaz leképezését S-re, g_2 pedig E_2 leképezését jelenti K-ba.

Mi - zászlós-ábráinknál - az alábbi elnevezéseket és szimbólum-rendszert alkalmazunk.

Az S halmaz s elemeit illetve a K halmaz k elemeit továbbra is szavaknak illetve kategóriáknak nevezzük.

A $c_1 \in C_1$ csucspontot, ha $f_1(c_1) = k \in K$, a k kategória kezdőpontnak, mig a $c_2 \in C_2$ csucspontot,

ha $f_2(c_2) = k \in K$, a k kategória végpontnak nevezzük.

Egy k kategória kezdő- és végpontját a G gráfban zászló-fejjel ábrázoljuk, mely balra ill. jobbra mutat aszerint, hogy kezdő ill. végpont jeléül szolgál. E zászló-fej olyan derékszögü háromszög, mely egyik befogóján áll és a derékszög e befogón jobbra illetve balra helyezkedik el:

k kategória

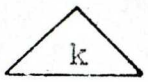
kezdőpont végpont

A zászlós-ábrák jobb áttekintése miatt a k kategória zászló-fejet nyéllel láthatjuk el:

Az $e_1 \in E_1$ élt ($g_1(e_1) = s \in S$) s szó élnek nevezzük. A szó éleket a következő módon jelöljük:

s - él

Végül az $e_2 \in E_2$ élt ($g_2(e_2) = k \in K$) k kategória élnek nevezzük. Egy k kategória élt olyan kettős zászló-fejjel ábrázolunk, mely balra is és jobbra is mutat. A kettős zászló-fej olyan derékszögü háromszög, mely átfogóján áll:



k - él

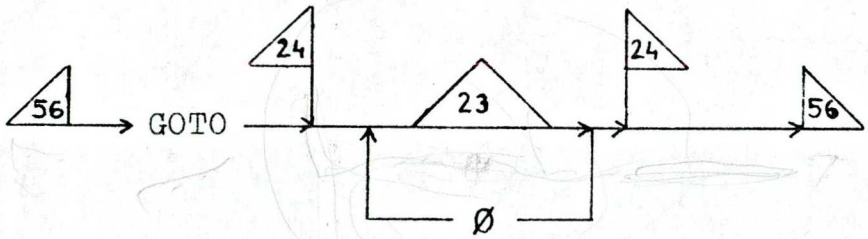
A FORTRAN-PZ nyelv ábrázolásakor a nyelv minden kategóriájához egy sorszámot is hozzárendelünk. Az ábrákon a kategóriákra a hozzájuk rendelt sorszámukkal hivatkozunk, azaz e sorszámokat irjuk be a zászló-fejekbe. A zászló-nyelek hosszának nem tulajdonitunk jelentőséget, csupán - esztétikai szempontból - az egy kategóriához tartozó kezdő- és vég zászló-fejekhez egyforma hosszuságu nyelet rajzolunk. A kettős zászló-fejekhez nyelet nem rajzolunk.

Az elmondottakat egy egyszerü ábrával szemléltetjük. A FORTRAN-PZ nyelv feltétel nélküli ugró utasitása

GOTO cimke

alaku. Rajzoljuk fel az utasitás zászlós-ábráját.

Legyenek 23. értékes számjegy, 24. cimke és 56. vezérlő utasitás kategóriák (23, 24, 56 $\in$ K), valamint GOTO és Ø szavak (GOTO, Ø $\in$ S), akkor egy lehetséges zászlós-ábra a következő lehet:



Az ábrából leolvasható, hogy a feltétel nélküli vezérlést átadó utasitás a fenti alaku, továbbá az is,

hogy cimkéül értékes számjeggyel kezdődő számjegy sorozat használható.

Az olyan zászlós-ábrát, mely k éleket nem tartalmaz (E_2 üres halmaz) <u>véges állapotu</u>-nak nevezzük (egy ilyet, egy $A = \{S, K, G, f_1, f_2, g_1\}$ rendezett hatos ad meg). Véges állapotu zászlós-ábrához a zászlós-ábra <u>deriváltjain</u> (leszármazottjain) keresztül juthatunk. Ha egy zászlós-ábra valamely k élét k <u>részgráfjával</u> helyettesitünk, akkor a zászlós-ábra egy deriváltját nyerjük. Egy zászlós-ábra bármely két $c_1 \in C_1$ és $c_2 \in C_2$ pontját összekötő G' gráfot G <u>részgráfjának</u> nevezünk, k részgráfról pedig akkor beszélünk, ha c_1 és c_2 pontokra teljesül, hogy $f_1(c_1) = f_2(c_2) = k$.

Ezek után

Definició: egy $L = \{S, K, f\}$ nyelvet az $A = \{S, K, G, f_1, f_2, g_1, g_2\}$ <u>zászlós-ábra által reprezentált nyelv</u>nek nevezünk, ha f minden $k \in K$ kategóriához mindazon f(k) szósorozatok halmazát rendeli, melyek a zászlós-ábra valamely deriváltjában, valamely k él mentén leolvashatók.

A fent leirt fogalmak és jelölésrendszer alapján reprezentáljuk a FORTRAN-PZ nyelvet az 5. pontban található zászlós-ábrákkal.

4. A FORTRAN-PZ nyelv, és eltérései az ASA FORTRAN nyelvtől

Jelen pontban a FORTRAN-PZ nyelv definicióját adjuk meg. Mondanivalónk egyrésze a zászlós-ábrák magyarázatául szolgál, másrésze azon konvenciók, kiegészitések felsorolása, melyeket a zászlós-ábrákon nem adhatunk meg, végül rámutatunk más FORTRAN nyelvektől [6, 7, 8] való eltéréseire.

A nyelv megalkotásánál - oktatási szempontoktól vezérelve - az ASA FORTRAN nyelvből csak a legszükebb fogalmakat használtuk fel, de éppen annyit, amennyi - véleményünk szerint - az alapfoku képzés során szükséges egy programozási nyelv strukturájának teljes megismeréséhez. Igyekeztünk továbbá minél kevesebb gépi korlátozást figyelembe venni.

A nyelvi változat elsősorban numerikus feladatok megoldására alkalmas. A megoldható numerikus feladatok köre nem haladja meg az alapképzésben részt vevők matematika ismereteit sem. Ez a cél is vezetett bennünket a nyelv megalkotása során.

Az alábbiakban a nyelv ismertetését ugy végezzük el, hogy az alapelemekből kiindulva jutunk el a program végső fogalmához. A tárgyalás során az éppen tárgyalt zászlós-ábrákra ( ) -be tett ábraszámokkal hivatkozunk.

4.1. A programirás szabályai

A nyelven leirt programokat programlapra irjuk. A programlap sorokból és 80 oszlopból áll. A 80 oszlop négy sávra oszlik:

1) az első sáv (1 - 5 oszlopok) cimkék megadására szolgál,
2) a második sáv (6. oszlop) az un. folytatás megjelölésére használjuk,
3) a harmadik sávba (7 - 72 oszlopok) az utasitásokat irjuk,
4) a negyedik sáv (73 - 80 oszlopokban lévő információ) nem kerül feldolgozásra, igy ide az utasitásokhoz szükséges megjegyzéseinket irhatjuk.

Egy oszlopba csak egyetlen jel (karakter) irható. Minden utasitást uj sorba kell irni. Ha egy utasitás 66 jelnél többet tartalmaz uj sorba kell folytatni. A folytatás tényét a 6. oszlopban jelöljük meg egy értékes számjeggyel (2), igy összesen 9 folytatás-sor megengedett.

Az első sáv első poziciójába irt C karakter azt jelöli, hogy az ilyen sor nem tartozik a programhoz. Az ilyen "comment" sorba magyarázatot irhatunk a program olvasója részére.

Az első sávba az utasitások cimkéjét irjuk. Cimke legfeljebb ötjegyü, értékes számjeggyel kezdődő számsorozat lehet (2).

Üres karaktert a programirásakor bárhol használhatunk.

4.2. Alapjelek

A nyelv alapjelei az 1. ábrán - a betük és számjegyek a 2. ábrán - találhatók.

Az ábra különösebb magyarázatra nem szorul. Csupán annyit jegyzünk meg, hogy az ASA FORTRAN több alapjelének hiánya már rámutat a szükités mértékére is.

A zérust különleges számjegyként szerepeltetjük, mivel több esetben (cimke, index, stb.) csak értékes számjegyek használatát engedjük meg.

4.3. Konstansok

A nyelvben szám és logikai konstansok fordulhatnak elő. Az ASA FORTRAN ötféle, a TPA FORTRAN csak számkonstans használatát engedi meg. Utóbbitól eltérően mi a logikai konstansok használatát is megengedjük, szükségességét érezve ennek a matematika korszerü oktatása miatt is.

A konstansok tipusai tehát INTREGER, REAL és LOGICAL lehet.

Az INTEGER és a LOGICAL konstansokra a FORTRAN nyelveknél szokásos konvenciók érvényesek (4), (5).

A REAL tipusu számok a tizedes tört alakban irt racionális számok. Irásuk általában öt részből áll: előjel, egész rész, tizedespont, törtrész, kitevőrész (4).

A kitevőrész további részekre bomlik: a kitevő jele (E betü), a kitevő előjele és a kitevő.

Általános alakuk tehát:

$$\pm\ e\ .\ t\ E\ \pm\ k$$

ahol e, t, k számjegy-sorozatok. Irásmódjukra az alábbi megszoritásokat tesszük:

tizedespontnak mindig kell szerepelni,

a zérus egész részt nem szükséges kiirni,

az egész és törtrész közül legalább az egyiknek szerepelni kell,

a kitevőrész hiányozhat,

a tizedespontot nem tartalmazható kitevőnek mindig van előjele.

4.4. Változók, azonositók

A nyelvben aritmetikai és logikai változókat használunk. A változókra való hivatkozás azonositójukkal történik.

Az azonositó (2) betüvel kezdődő, betüt vagy számjegyet tartalmazó karaktersorozat. Az egyes FORTRAN nyelvek megszabják - általában 6 karakterben - a karaktersorozat hosszát is, mi erre nézve nem teszünk megszoritást. Azonositóul nem használjuk az angol szóalapjeleket, illetve az ezekkel kezdődő karaktersorozatokat.

Az azonositók első betüjével automatikusan dek-

laráljuk (2), hogy milyen tipusu változók megjelölésére használjuk az azonositót a program során a következő módon:

I, J, K, M, N kezdőbetüvel egész tipusu,

L kezdőbetüvel logikai és

a többi betüvel kezdődő azonositókat valós tipusu változók neveként használjuk. Az automatikustól eltérő deklarációról a 4.11. pontban szólunk.

Az aritmetikai változók (6) skaláris (11) és indexes változók (3) lehetnek.

Az indexes változók - fogalmuk a matematika indexes változóival azonos - egy és két indexüek lehetnek. Indexként csak értékes számjegyet és egésznek deklarált betüt használhatunk. Némely FORTRAN változatok kettőnél kevesebb, némelyek több index használatát is megengedik. Mi elegendőnek véljük a maximálisan két index használatát, igy ugyanis a vektorok elemein kivül mátrix-elemeket is tudunk használni.

Az indexes változók deklarálásáról később szólunk.

4.5. Kifejezések, függvények

A kifejezések aritmetikai és logikai kifejezések lehetnek.

Aritmetikai kifejezéseket (4) kapunk akkor, ha operanduszokat aritmetikai müveleti jelekkel kapcsolunk össze. A kifejezésben szereplő operanduszok számkonstansok, változók, függvénykifejezések és zárójelbe

helyezett aritmetikai kifejezések lehetnek.

Egy aritmetikai kifejezés tipusa egész ill. valós aszerint, hogy a benne szereplő valamennyi operandusz egész ill. valós tipusu. Az aritmetikai müveletek csak azonos tipusu operanduszokon végezhetők el. Kivétel ez alól a szabály alól némileg a hatványozás: egész alapot csak egész kitevőre hatványozhatunk, ekkor az eredmény is egész tipusu; valós alap esetén valós tipust kapunk, de negativ számot valós kitevővel nem szabad, nullát pedig csak pozitiv egész kitevővel szabad hatványozni.

Az aritmetikai kifejezések értékének meghatározásakor a benne szereplő változók aktuális értékével végezzük el a kijelölt müveleteket a müveletek precedenciája és a bal-asszociáció alapján. A szabályok adta sorrendtől való eltérést zárójelek alkalmazásával érhetünk el. Egy zárójeles aritmetikai kifejezés mindig önálló operandusznak tekintendő. Zárójeles kifejezések tetszésszerinti mélységben egymásba skatulyázhatók és értékelésük belülről kifelé történik.

A logikai kifejezések (5) logikai konstansokból, változókból, aritmetikai kifejezések közötti relációkból, logikai müveleti jelekből és zárójelbe zárt logikai kifejezésekből épülnek fel.

A logikai kifejezések értékelése hasonlóan történik, mint az aritmetikai kifejezéseknél, a logikai müveletek precedenciája (sorrend: aritmetikai müveletek, relációk, NOT, AND és OR müvelet) és a bal-

asszociáció alapján.

A nyelv kétféle függvény-tipust használhat: az un. standard függvényeket és az utasitásfüggvényeket.

A standard függvények (1) közül az ALOG, COS, EXP, SIN és SQRT külső standard függvények csak REAL tipusu argumentummal rendelkezhetnek és értéküK is ilyen tipusu. Paraméterként tetszőleges aritmetikai kifejezést használhatunk, mely ujabb függvény hivatkozást is tartalmazhat.

Az utasitás függvények csak deklaráció után használhatók, igy ezekről a 4.11. pontban szólunk.

4.6. Utasitások

A nyelvben a programok utasitások sorozataként irhatók fel. Az utasitásokat funkciójuk szempontjából két csoportra osztjuk. Az egyik csoport a végrehajtható utasitások (15) csoportja, mely utasitások a forditás után a célprogramban is szerepelnek és a célprogram futása során hajtódnak végre, mig a másik csoport a forditóprogram számára ad információt s ezeket nem végrehajtható vagy deklarációs utasitásoknak (15) nevezzük.

Először a végrehajtható, majd a nem végrehajtható utasitásokat tárgyaljuk.

4.7. Értékadó utasitás

Az aritmetikai és logikai müveleti utasitások formája a nyelvben egységes, igy nem is különböztet-

jük meg őket, hanem egységesen értékadó utasitásoknak (6) nevezzük. Az értékadó utasitások formája:

Változó = kifejezés ,

vagy röviden V = k

Az utasitás végrehajtása során a k (aritmetikai vagy logikai) kifejezés aktuális értéke átadódik a baloldali V változónak, V korábbi értéke elvész. V és k tipusának meg kell egyezniök. Megengedett, hogy egyszerre több egyező tipusu változónak ugyanazt az értéket adjuk egyetlen értékadó utasitással.

4.8. Szervező utasitások

A programok fontos részei a szervező utasitások (7). Ezek a megállási-, üres- és I/O-utasitások.

A STOP utasitást a gép megállitására használjuk.

Az üres utasitás CONTINUE hatására nem történik semmi. Ez az utasitás mindig cimkével ellátott. Mi a ciklusok lezárására használjuk.

Az információ bevitelét a READ, kivitelét pedig a WRITE utasitással végezhetjük el (7). Mindegyik utasitáshoz tartozik egy FORMAT utasitás (8), mely az átvitel formáját határozza meg. Bár a FORMAT utasitás a nem végrehajtható utasitások csoportjába tartozik az I/O utasitásokkal való szoros kapcsolata

miatt mégis itt tárgyaljuk.

Az I/O utasitások alakja:

READ (p, c) L ,

WRITE (p, c) L ,

ahol p és c előjel nélküli egész számok, L pedig az I/O lista.

A p (periféria szám) a programozó által a különböző perifériás egységekhez rendelt értékes számjegy. A hozzárendelésre semmilyen korlátozást nem teszünk.

A c szám az I/O utasitáshoz rendelt FORMAT utasitás cimkéje.

Az I/O lista az átvitelre kerülő változók azonositóit tartalmazza, egymástól vesszővel elválasztva. A lista egyetlen elemből is állhat.

A FORMAT utasitás alakja:

c FORMAT (F)

ahol c az utasitás cimkéje, F pedig a FORMAT lista. Egyetlen FORMAT utasitás sem állhat cimke nélkül, de több I/O utasitáshoz is tartozhat ugyanaz a FORMAT utasitás.

A FORMAT lista elemei az átvitt változók alakját szabják meg, egymástól vesszővel vannak elválasztva. Egy összetartozó READ - FORMAT vagy WRITE - FORMAT utasitásnál az I/O és F lista elemei között pontos megfeleltetés van.

A FORMAT lista elemei (az un. specifikációk)

csak numerikus adatok átviteli alakját határozzák meg. Egész tipusu adatok átvitelére az I, valós tipusu adatok átvitelére pedig az F specifikációt használjuk. Több FORTRAN változat a numerikus specifikációk mellett un. szerkesztési specifikációkat is megenged. Ezeket mi nyelvünkbe nem épitettük be.

A FORMAT lista elemeinek alakja a következő lehet:

I specifikáció : I t

F specifikáció : F t.s

ahol t természetes szám, s pedig valamely számjegy. A specifikátorok hatásukat a következő módon fejtik ki:

a input esetén

I t jelentése: a változó INTEGER tipusu, a kártyalapon t oszlop van kijelölve a változó értékére.

F t.0 jelentése: a változó REAL tipusu, a kártyalapon t oszlop van kijelölve a változó értékére. Ekkor mindig s = 0 irandó, melynek nincs jelentősége.

b output esetén

I t jelentése: a változó INTEGER tipusu és értéke t hely-

re iródjon ki.

F t.s jelentése: a változó REAL tipusu és értéke t helyre iródjon ki, a tizedespontot s számjegy kövesse.

A specifikációknál ismétlési tényezőket nem engedünk meg.

4.9. Vezérlő utasitások

Vezérlő utasitásokként feltétel nélküli és feltételes vezérlő utasitásokat definiálunk (9). Mindkettő hatására a vezérlés valamely cimkével ellátott vagy cimke nélküli utasitásra adódik át.

Feltétel nélküli vezérlő utasitás alakja GOTO c, ahol c annak az utasitásnak a cime, melyre a vezérlés átadódik. A cimkével ellátott utasitás a vezérlő utasitás előtt és után is lehet.

Feltételes vezérlés átadásra a nyelv IF utasitásai szolgálnak. Definiálunk aritmetikai és logikai IF utasitásokat.

Az aritmetikai IF utasitás alakja:

$$\text{IF}\,(k)\ c_1, c_2, c_3,$$

ahol k egy aritmetikai kifejezés, c_1, c_2, c_3 pedig cimkék. Az ilyen utasitás hatására vezérlés átadás történik a c_1, c_2 illetve c_3 cimkével ellátott utasitásra aszerint, hogy a zárójelbe tett

aritmetikai kifejezés értéke negativ, nulla illetve pozitiv.

A logikai IF utasitás alakja:

IF (k) U ,

ahol k egy logikai kifejezés, U pedig egy utasitás. Az U utasitás nem viselhet cimkét és nem lehet logikai IF utasitás vagy ciklusutasitás. A logikai IF utasitás jelentése: ha a logikai kifejezés értéke igaz, akkor az U utasitás, mig ha hamis, akkor U végrehajtása nélkül a soronkövetkező utasitás hajtódik végre.

4.10. Ciklusutasitás

Igen fontos szerepet tölt be a programozásban a ciklusutasitás (10), mely olyan utasitáscsoportok összefogására szolgál, ahol az utasitás-csoportban szereplő változók különböző értékei mellett egymásután többször is végre kell hajtani a csoportban lévő utasitásokat.

A ciklusutasitást DO utasitásnak is nevezzük, általános alakja:

DO c p = k, v, n

ahol c cimke, p a ciklusparaméter, k a ciklusparaméter kezdő-, v a végértéke, n pedig a ciklusparaméter növekménye.

A c cimke a ciklusban végrehajtandó utasitáscsoport (ciklusmag) utolsó utasitásának cimkéje.

Mi mindig az üres utasitással (CONTINUE) zárjuk a ciklusok magját. A záró utasitás csak a DO utasitás után állhat.

A p ciklusparaméter csak egész tipusu skalárisváltozó lehet. A ciklusba lépéskor felveszi kezdőértékét. A ciklusban értékét nem szabad megváltoztatni.

A k és v értékei pozitiv egész számok (0-tól különböző) vagy egész változók lehetnek. Csak a $v > k$ esetet engedjük meg. Az n értéke csak értékes számjegy lehet. Ha $n = 1$, akkor megengedett a ciklusutasitás röviditett irásmódja is:

DO c p = k, v

A DO utasitás végrehajtása során a ciklusparaméter felveszi a k kezdőértéket, majd végrehajtódnak a magban szereplő utasitások, ezután a ciklusparaméter értéke az n növekmény értékével megnő és ha a paraméter értéke nagyobb mint a v végérték, akkor a záróutasitást követő utasitás, különben ujra a ciklusmag hajtódik végre.

A DO utasitás utáni első utasitásnak végrehajthatható utasitásnak kell lenni.

Megengedjük a ciklusok egymásba skatulyázását is. Az egymásba skatulyázás mértékére nem teszünk megkötést. A belső ciklus minden utasitásának a külső ciklus magjához kell tartozni (közös záróutasitásuk lehet). A belső és külső ciklus nem használhat azonos ciklusparamétert.

Nem definiáljuk a kiterjesztett hatáskör fogalmát, a cikluson kivülről tilos a ciklusba való vezérlésátadás.

4.11. Deklarációk

A deklaráló utasitások közül csak a tipusdeklarációt (11) engedjük meg. Mind a skaláris, mind az indexes változókat tipusdeklarációval deklaráljuk, ha az automatikus tipusdeklaráció céljainknak nem megfelelő.

Általános alakjuk:

$$T \quad v_1, v_2, \ldots, v_n$$

ahol T a tipus neve (INTEGER, REAL, LOGICAL), a v_i-k pedig azon változók azonositói - méretükkel ellátva - melyeknek tipusát elő akarjuk irni.

A skaláris változók közül csak az automatikus deklarálástól különböző változókat kell a fentiek szerint deklarálni. Az indexes változókat mindig deklarálni kell.

A változók dimenziója csak 0, 1 vagy 2 lehet.

A FORTRAN nyelvekben szokásos minden más deklaratiV utasitás hiányzik.

A deklaráció során adjuk meg a standard függvényeken kivül a programban felhasználandó függvényeket, az utasitás függvényeket is. Alakjuk:

$$F\,(p_1, p_2, \ldots, p_n) = k$$

ahol F a függvény azonositója, a p_i-k a függvény formális paraméterei ($i \geq 1$), amelyek skaláris változók, k pedig p_i-ket tartalmazó kifejezés.

4.12. Szegmensek

A nyelven felirt programok programegységekből, un. szegmensekből (15) állnak. Mi két féle szegmens használatát engedjük meg a fő- és eljárás szegmenst. E két szegmens közötti információcserét a formális és aktuális paraméterek segitségével valósitjuk meg. Nincs szükségünk tehát a FORTRAN-nyelvekben használt COMMON utasitásra.

A fő-szegmens szerkezeti felépitését - melyben a sorrend kötött - a 15. ábra mutatja:

1. szegmensnyitó utasitás,
2. deklaráló utasitások,
3. utasitás függvények,
4. végrehajtható utasitások,
5. FORMAT utasitások,
6. szegmenszáró utasitás.

Fő-szegmensnyitó utasitás a PROGRAM, záró utasitás az END utasitás.

Minden program egyetlen fő-szegmenst és tetszőleges számu eljárás szegmenst tartalmazhat.

Az eljárás szegmens szerkezeti felépitését a 13. ábra mutatja. Nyitó utasitásának alakja:

SUBROUTINE e $(f_1, f_2, \ldots, f_n)$

ahol e az eljárás neve, egy azonositó, az f_i-k pedig az eljárás formális paraméterei. Az eljárások formális paraméterei skaláris és indexes változók lehetnek. A formális paraméterek számára nem teszünk korlátozást.

Minden eljárás szegmenst egy fő-szegmenshez tartozó utasitás aktivizál. Az eljárás végrehajtása után a fő-szegmensbe való visszatérésre a RETURN utasitást használjuk. Az eljárás szegmenszáró utasitása is az END utasitás.

Az eljárás szegmensek aktivizálása (hivása) az eljárás utasitással (14) történik. Az eljárás utasitás alakja:

$$\text{CALL } e\ (a_1, a_2, \ldots, a_n)$$

ahol e a hivott eljárás neve, az a_i-k pedig az aktuális paraméterek.

Az aktuális paraméterek értékeivel hajtódik végre a hivott eljárás. Igy az eljárás formális paraméterei és az aktuális paraméterek között kölcsönös megfeleltetésnek kell lenni mind számukban, mind a paraméterek tipusaiban.

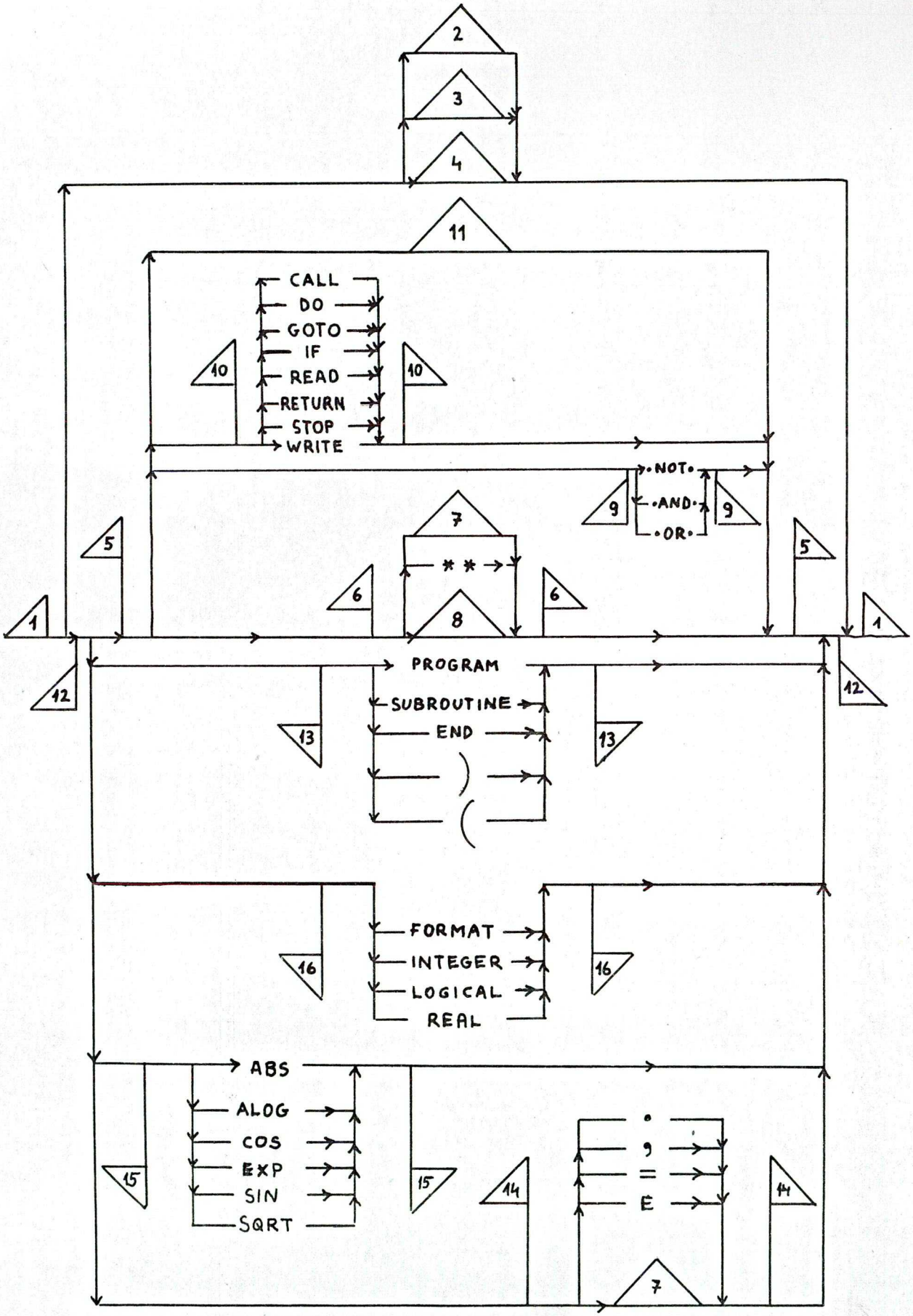
Megengedett, hogy az eljárás szegmensek további eljárás szegmenseket aktivizáljanak, de rekurziv szegmens aktivizálás nem lehetséges.

5. A FORTRAN-PZ nyelv zászlós-ábrái

Először felsoroljuk a FORTRAN-PZ nyelvnél használt kategóriákat sorszámaikkal együtt. Az egyes kategóriák után zárójelbe irt számok azon ábrák számát jelentik, melyeken a kategóriák kifejtését találjuk.

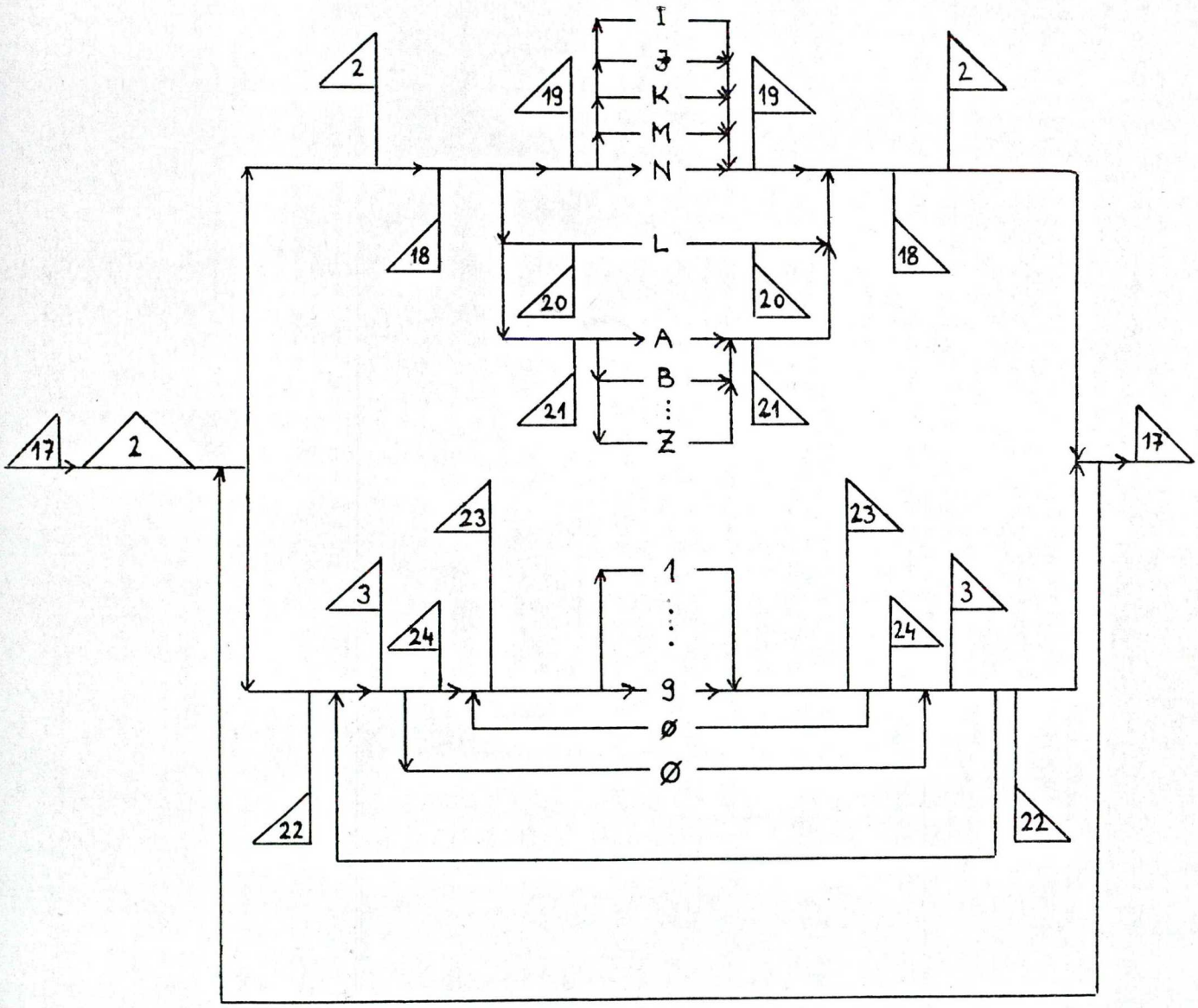
1. Alapjel (1)
2. Betü (2)
3. Számjegy (2)
4. Logikai érték (5)
5. Müveleti jel (1)
6. Aritmetikai müveleti jel (1)
7. Additiv müveleti jel (4)
8. Multiplikativ müveleti jel (4)
9. Logikai müveleti jel (1)
10. Vezérlő jel (1)
11. Reláció jel (5)
12. Elhatároló jel (1)
13. Zárójel (1)
14. Elválasztó jel (1)
15. Standard függvény jel (1)
16. Deklaráló jel (1)
17. Azonositó (2)
18. Automatikus deklaráció (2)
19. Egész betü (2)
20. Logikai betü (2)
21. Többi betü (2)
22. Természetes szám (2)
23. Értékes számjegy (2)
24. Cimke (2)
25. Előjel nélküli szám (4)
26. Tizedesszám (4)
27. Valódi tizedestört (4)
28. Kitevőrész (4)
29. Változó (6)
30. Skaláris változó (11)
31. Indexes változó (3)
32. Indexlista (3)
33. Indexkifejezés (3)
34. Függvénykifejezés (12)

71. Eljárás szegmens (13)
72. Eljárásfej (13)
73. Eljárásnév (13)
74. Formális paraméter rész (13)
75. Formális paraméter (13)
76. Formális paraméterlista (13)
77. Eljárástörzs (13)
78. Eljárásutasitás (14)
79. Aktuális paraméter rész (14)
80. Aktuális paraméterlista (14)
81. Aktuális paraméter (14)
82. Szegmens (15)
83. Fő-szegmens (15)



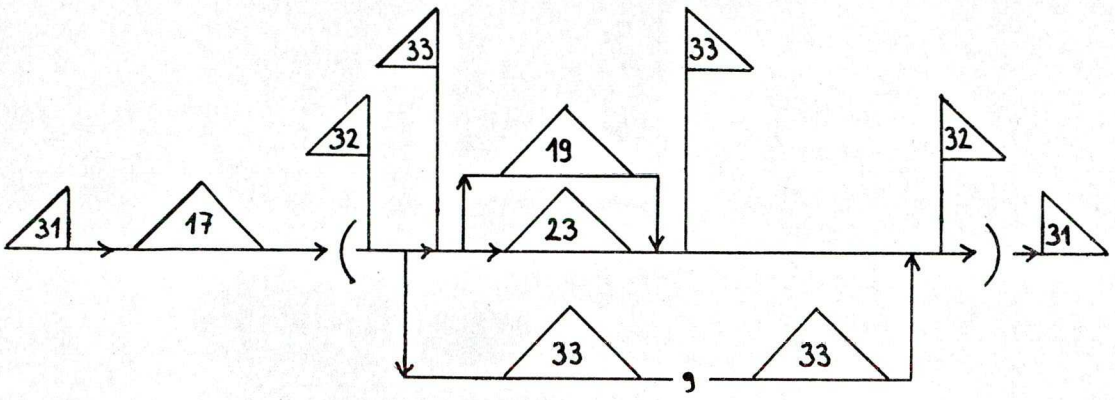
1. ÁBRA

1. ALAPJEL



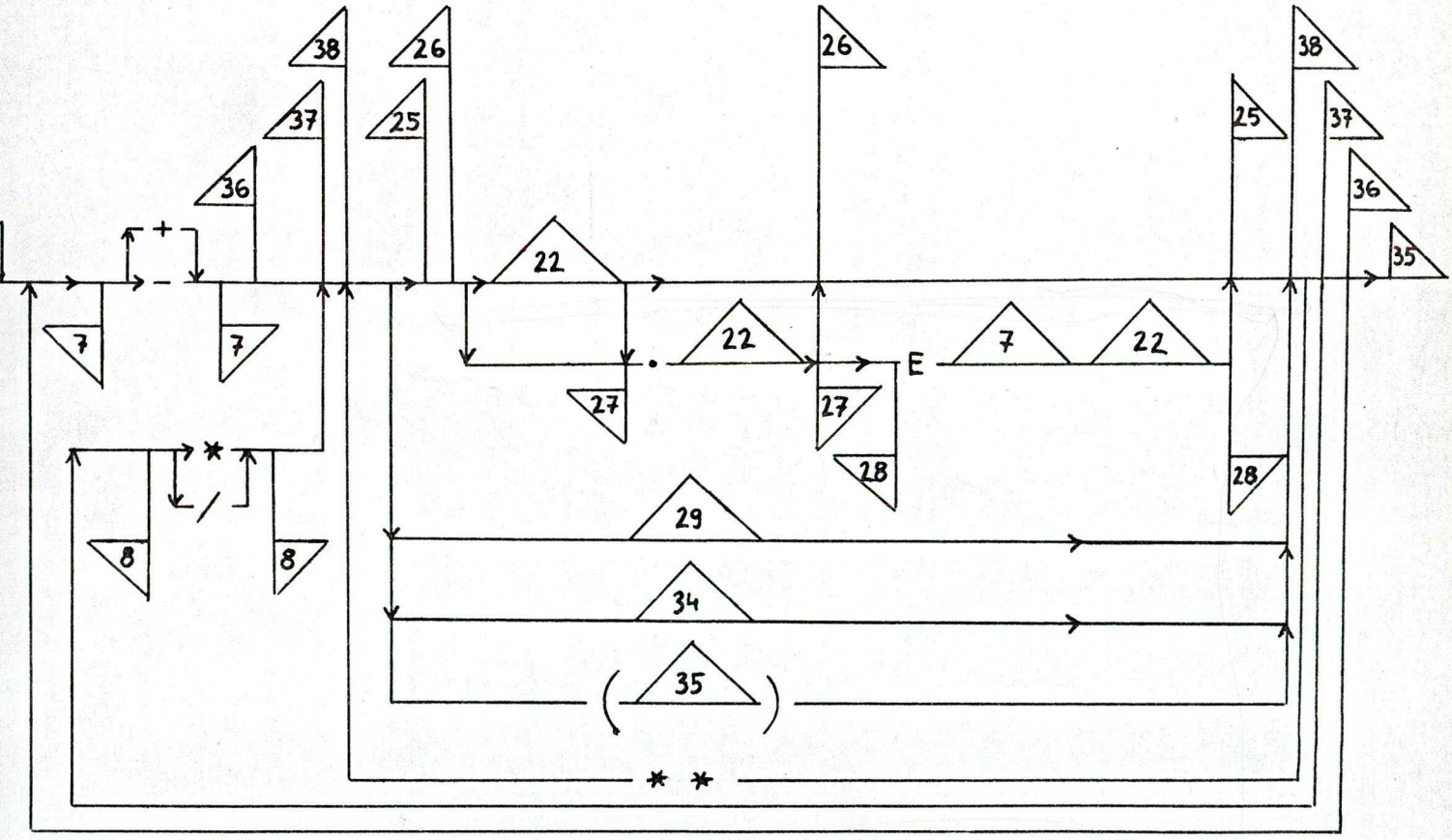
2. ÁBRA

17. AZONOSÍTÓ



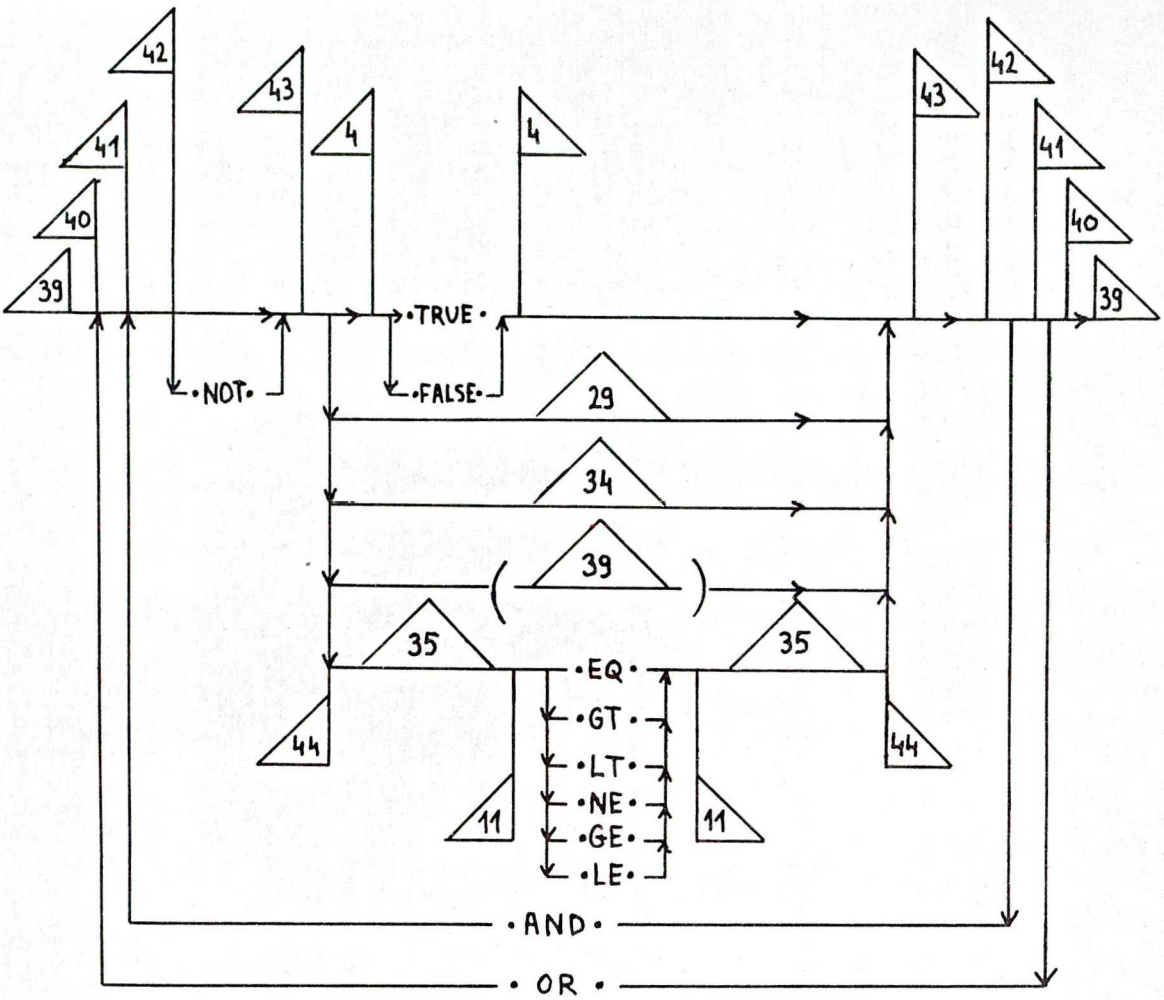
3. ÁBRA

31. INDEXES VÁLTOZÓ



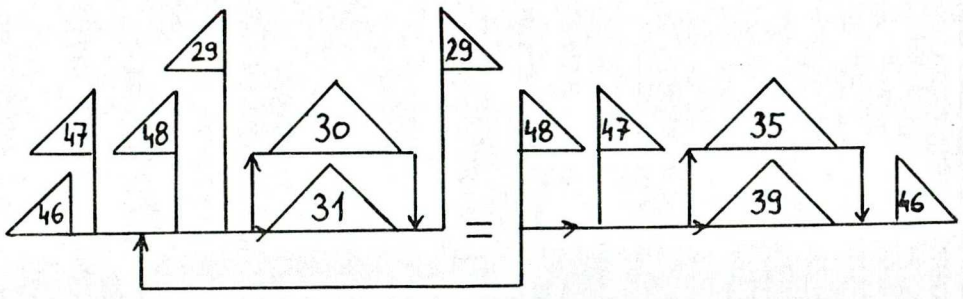
4. ÁBRA

35. ARITMETIKAI KIFEJEZÉS



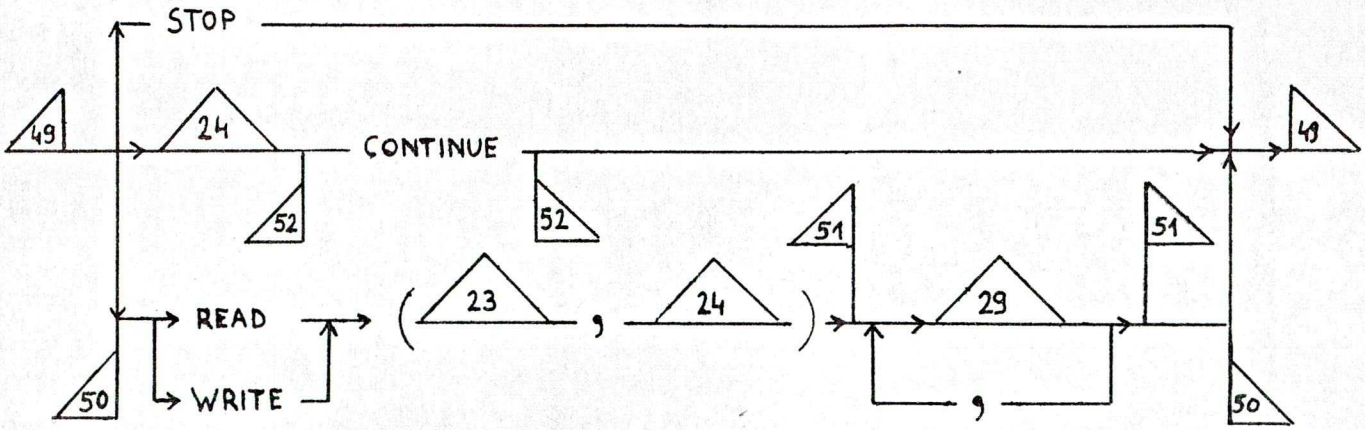
5. ÁBRA

39. LOGIKAI KIFEJEZÉS



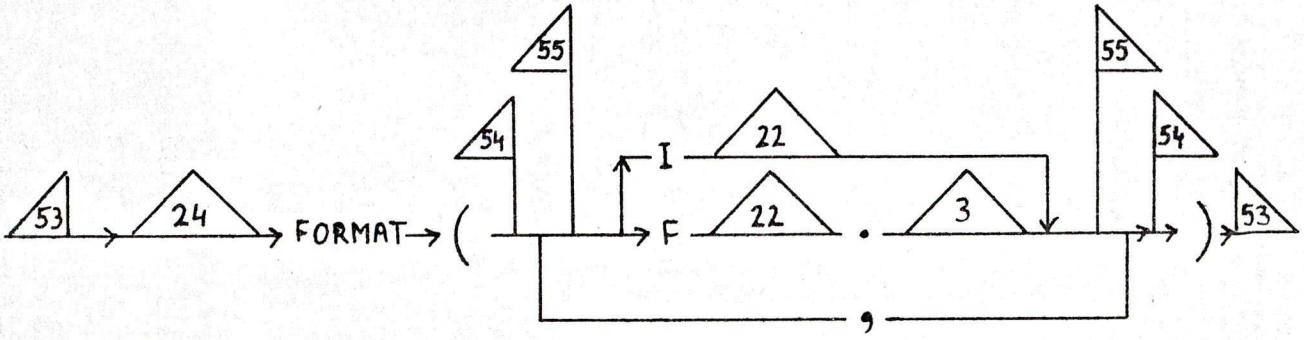
6. ÁBRA

46. ÉRTÉKADÓ UTASÍTÁS



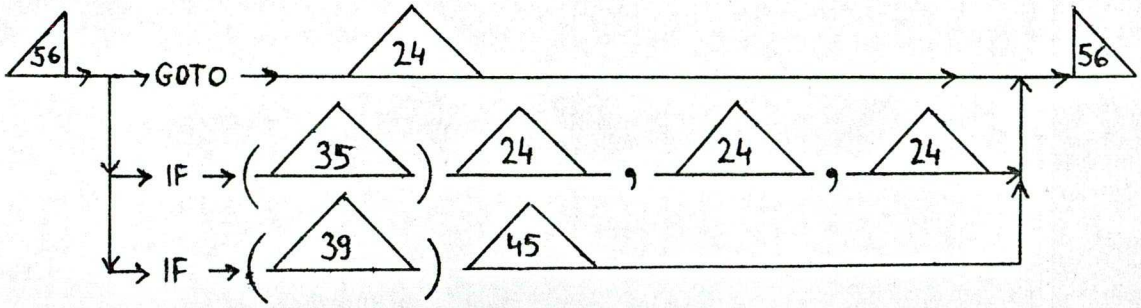
7. ÁBRA

49. SZERVEZŐ UTASÍTÁS



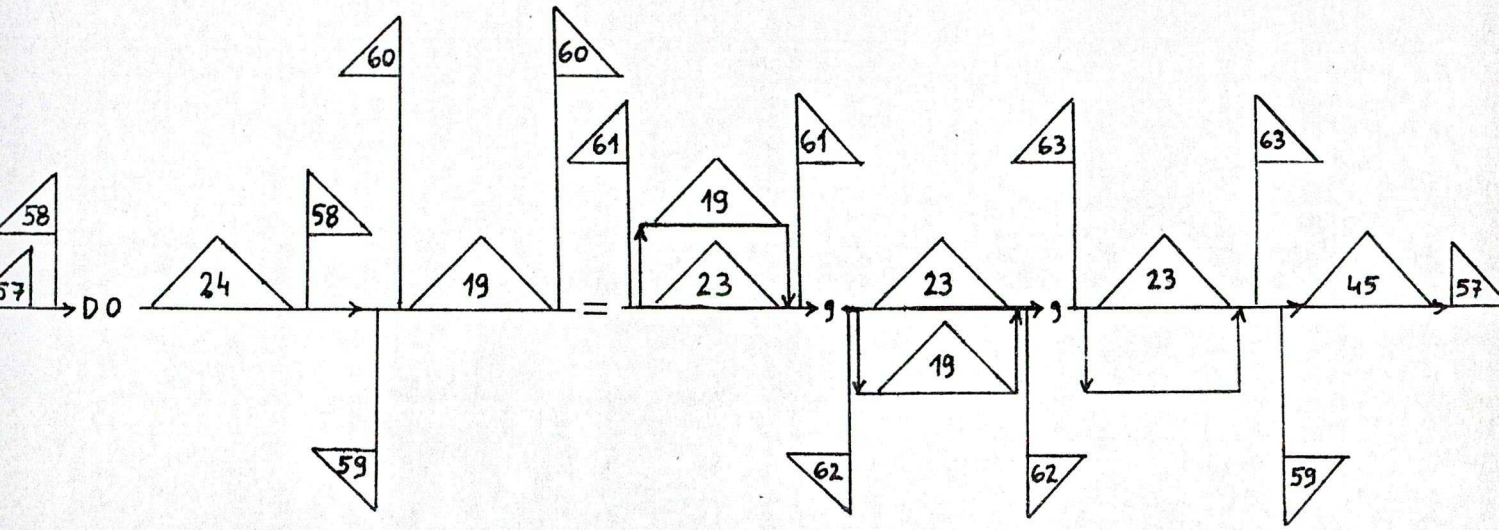
8. ÁBRA

53. FORMAT UTASÍTÁS



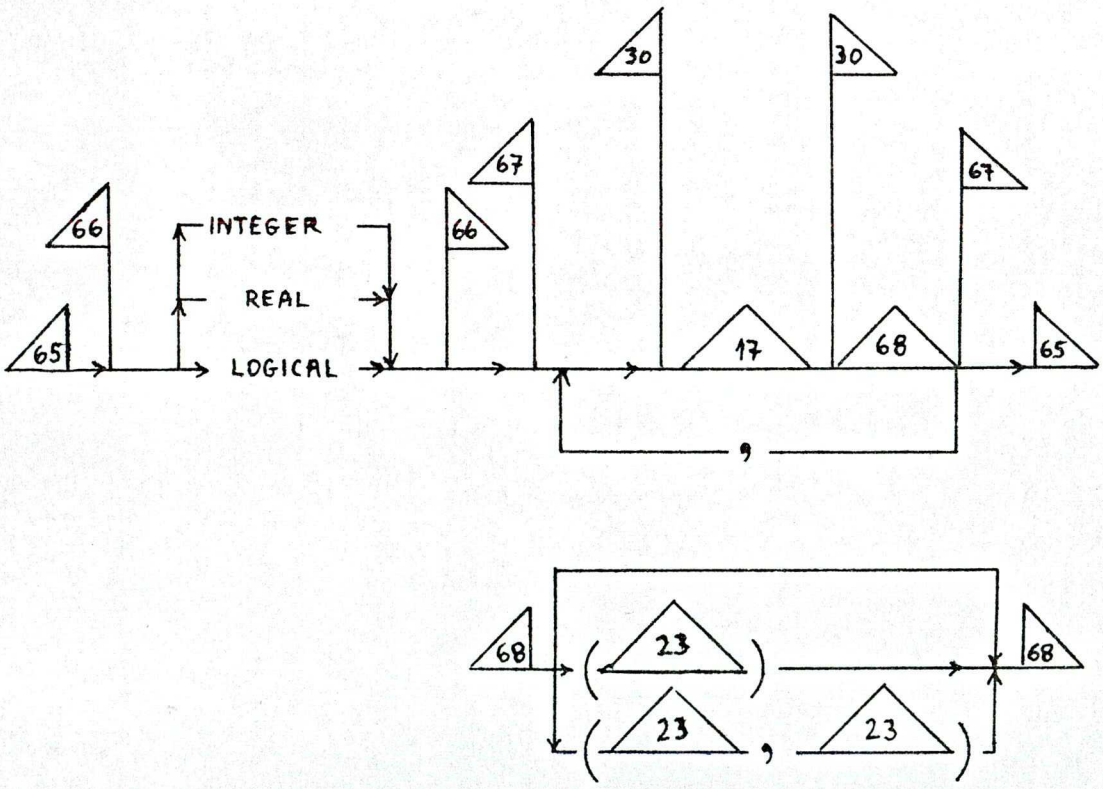
9. ÁBRA

56. VEZÉRLŐ UTASÍTÁS



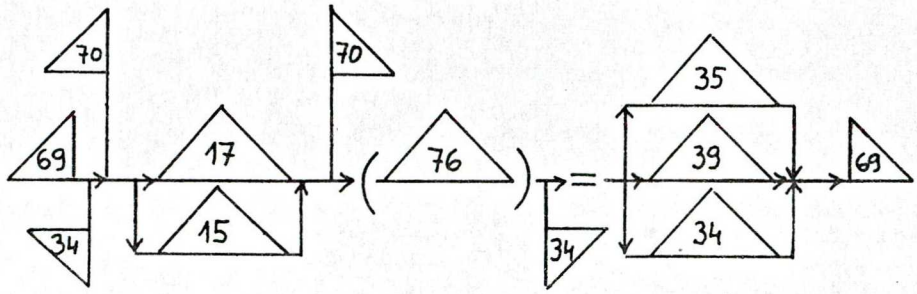
10. ÁBRA

57. CIKLUSUTASÍTÁS



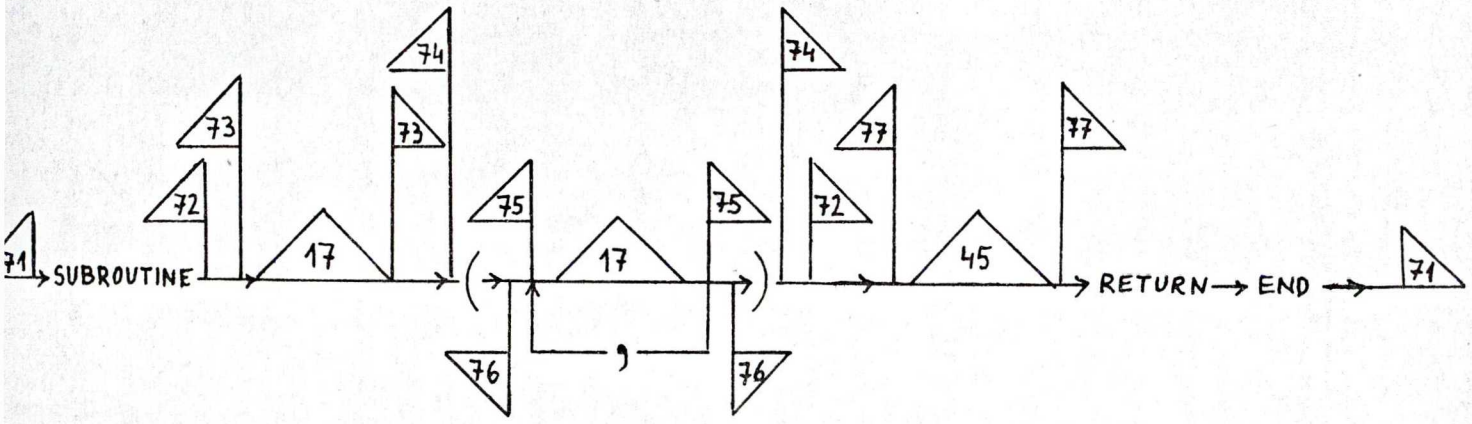
11. ÁBRA

65. TÍPUSDEKLARÁCIÓ



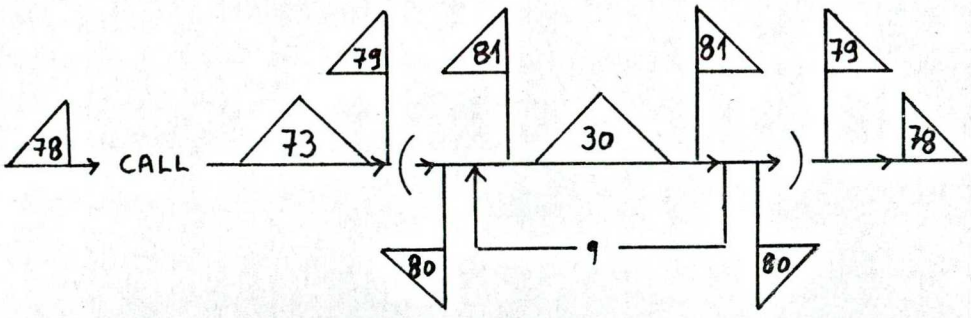
12. ÁBRA

69. UTASÍTÁS FÜGGVÉNY



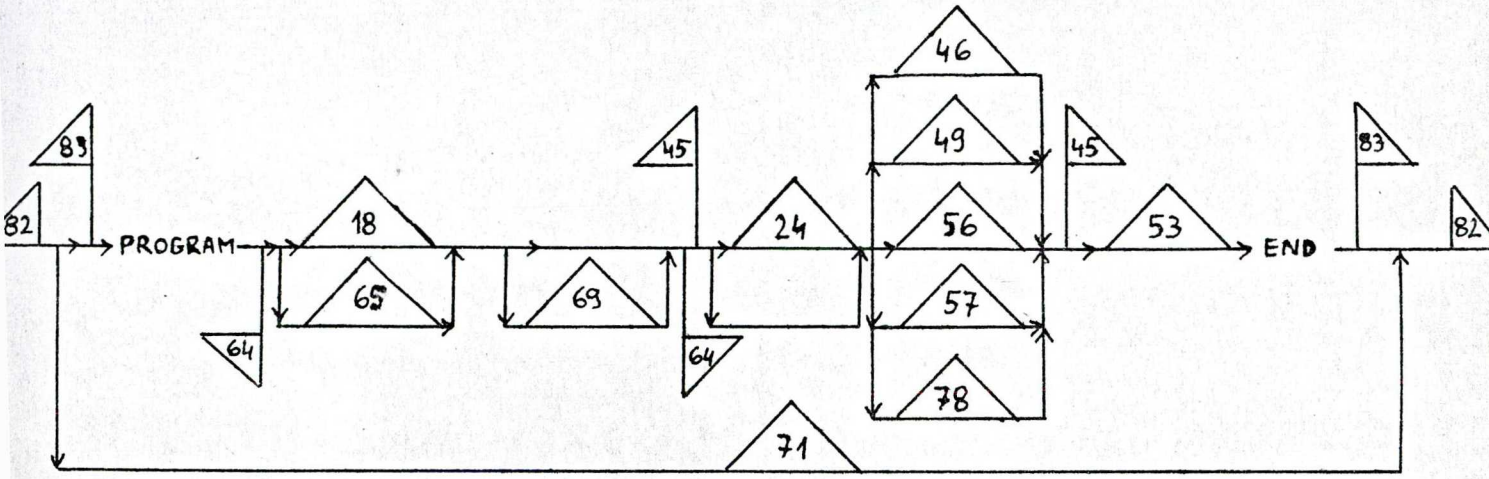
13. ÁBRA

71. ELJÁRÁS SZEGMENS



14. ÁBRA

78. ELJÁRÁSUTASÍTÁS



15. ÁBRA

82. SZEGMENS
83. FŐ-SZEGMENS
64. DEKLARÁCIÓ
45. UTASÍTÁS

I r o d a l o m j e g y z é k

1. Az elektronikus, digitalis számitógépek eddigi fejlődése és a várható fejlődés fő irányai, /Tanulmány/, Szeged, /1972/

2. Belső gépi nyelvek, beleértve a magasszintü nyelveket, /Tanulmány/, Szeged, /1973/

3. FARKAS, E., Programozási nyelvek szintaxisa és szemantikája, azok definiciója és formalizálása, /MTA-SZTAKI Tanulmány/, Budapest, 12/1973. 105-120.

4. KALMÁR, L., An intuitive representation of context-free languages, COLING, The procudings of the International Conference on Computational Linguistics, Sánga-Säby, /1969/

5. KANÁSZ, L., Feladatlapok a FOKAL programnyelv alapismereteinek tanitására, /Előadás/, Visegrád, /1974/

6. LŐCS, Gy., - VIGASSY, J., A FORTRAN programozási nyelv, /Müszaki Könyvkiadó/, Budapest, /1970/

7. PÁLOS, J., TPA - MINI - FORTRAN programnyelv, /ESZK Tájékoztató/, Budapest, 11/1973. 120-157.

8. PÁLOS, J. TPA - 8K - FORTRAN, /ESZK Tájékoztató/, Budapest, 11/1973. 157-229.

9. PERGE, I., - PUSKÁS, A., Numerikus és gépi módszerek I-II., /Tankönyvkiadó/, Budapest, /1974/

10. PUSKÁS, A., Számitástechnikai képzés a tanárképző főiskolákon, /Előadás/, Szeged, /1971/.

11. PUSKÁS, A., A FORTRAN programozási nyelv egy változata, /Előadás/, Visegrád, /1974/

12. SZENTGYÖRGYI, ZS., Uj Bábel épül? I-II., /A programozási nyelvek fejlődése/ Természet Világa, 1974. 4., 6. sz. 173-177. és 263-266.

T a r t a l o m j e g y z é k