JELENTÉS AZ ALGOL 60 ALGORITMIKUS NYELVRŐL[1]

A cikk szerzői, az ALGOL-bizottság tagjai: PETER NAUR (szerkesztő), J. W. BACKUS, L. F. BAUER, J. GREEN, C. KATZ, J. MCCARTHY, A. J. PERLIS, H. RUTISHAUSER, K. SAMELSON, B. VAUQUOIS, J. H. WEGSTEIN, A. van WIJNGAARDEN, M. WOODGER.

Fordította: RÉVÉSZ GYÖRGY[2]

A fordító előszava

Az elektronikus számológépek programozásához, ill. számítási eljárások leírásához az utóbbi években számtalan különböző formulanyelvet dolgoztak ki. Az ALGOL nyelv megalkotóinak az volt a céljuk, hogy olyan közös algoritmikus nyelvet hozzanak létre, amelyen megérthetik egymást a programozó szakemberek és amely egyúttal — fordító program segítségével — különböző elektronikus számológépek programozására is alkalmas.

A jelen cikk annak a jelentésnek a fordítása, amely a nyelvet az 1960 januári konferencián kidolgozott formában ismerteti. (Innen az ALGOL 60 elnevezés, megkülönböztetésül egy előbbi, 1958-ban kidolgozott alaktól.) A jelentést, amely rögzíti az ALGOL 60 „nyelvtanát", máris több nyelvre lefordították. A fordítások különböző matematikai folyóiratokban jelentek meg, helyenként magyarázó jellegű kiegészítésekkel és pl. az orosz nyelvű fordításhoz kritikai megjegyzéseket is fűztek.[3]

Az ALGOL nyelv fogalmainak magyar fordítása során jelentős gondot okozott, hogy helyenként egy egész új magyar nyelvű terminológia bevezetésére volt szükség. Az ebben, valamint a fordítás egészének átnézésében és javításában nyújtott segítségért köszönettel tartozom Dr. KALMÁR László akadémikusnak, valamint BÉKÉSSY András, DÖMÖLKI Bálint és FREY Tamás kollégáknak.

Budapest, 1961. október 6.

---

[1] „Report on the Algorithmic Language ALGOL 60" című cikk fordítása. (*Communications of the ACM* **3** (1960) 5, 299—314.)

A magyar nyelvre való fordításnál tekintetbe vettük a „Chiffres" c. folyóiratban megjelent francia nyelvű fordítást is. (A fordító megjegyzése.)

[2] MTA Számítástechnikai Központ.

[3] *Журнал вычислительной математики и математической физики* **1** (1961) 2, 308—342.

BEVEZETÉS

Előzmények

Annak az előzetes jelentésnek közzététele után[4,5] amely az ALGOL algoritmikus nyelvet egy 1958-ban, Zürichben tartott konferencián kidolgozott alakban ismertette, sokan érdeklődtek ez iránt a nyelv iránt.

Egy Mainzban, 1958 novemberében tartott tájékozódó összejövetel után 1959 februárjában, Koppenhágában jött össze egy ALGOL-konferencia, amelyre mintegy negyven érdekelt jött össze Európa különböző országaiból. Itt egy „gépi munkacsoport" alakult avégett, hogy a nyelv gépi alakját egészen a lyukszalag-kódig kidolgozzák. Ugyancsak ennek a konferenciának eredményeként kezdte meg a koppenhágai számológép-központ (Regnecentralen) egy, a további viták fórumaként szolgáló ALGOL Bulletin kiadását, Peter Naur szerkesztésében. Az 1959 júniusában Párizsban ülésező ICIP konferencián az ALGOL-bizottság számos megbeszélést tartott. Ezek a megbeszélések feltártak néhány félreértést a csoport feladatával kapcsolatban, amely eredetileg elsősorban az ALGOL-nyelv definíciója volt, de ugyanakkor világossá tették azt is, hogy ez a próbálkozás széles körű elismerésre talál. A viták eredményeképpen elhatározták, hogy 1960 januárjában nemzetközi találkozót tartanak az ALGOL nyelv megjavítása és egy végleges megfogalmazás kidolgozása végett. Egy európai ALGOL konferencián, amelyet 1959 novemberében Párizsban tartottak, és amelynek mintegy ötven résztvevője volt, az 1960 januári konferenciára hét európai képviselőt választottak, akik a következő szervezeteket képviselték: *Assotiation Francaise de Calcul*, *British Computer Society*, *Gesellschaft für Angewandte Mathematik und Mechanik*, és *Nederlands Rekenmachine Genootschap*. A hét képviselő egy végső előkészítő ülést tartott Mainzban, 1959 decemberében.

Eközben az Egyesült Államokban mindazok, akik módosításokat vagy helyesbítéseket kívántak javasolni az ALGOL nyelvvel kapcsolatban, felszólítást kaptak, hogy megjegyzéseiket a Communications of the ACM című folyóiratnak küldjék meg, közzététel végett. Ezek a megjegyzések szolgáltak azután az ALGOL módosítására vonatkozó megfontolások alapjául. Két szervezet, a SHARE és az USE, ALGOL munkabizottságot alakított és mindkét szervezet képviseltette magát az ACM programozási nyelvekkel foglalkozó bizottságában. Az ACM bizottság 1959 novemberében Washingtonban ülésezett és minden megjegyzést, amelyet csak az ALGOL nyelvre vonatkozólag az ACM folyóiratnak megküldtek, figyelembe vett. Ugyanitt hét képviselőt is választottak az 1960 januári nemzetközi konferenciára. Ez a hét képviselő 1959 decemberében, Bostonban tartott végső előkészítő megbeszélést.

Az 1960 januári konferencia

A tizenhárom küldött[6], aki Angliából, Dániából, Franciaországból, Németországból, Hollandiából, Svájcból és az Egyesült Államokból jött össze, Párizsban, 1960 január 11-től 16-ig ülésezett.

---

[4] Preliminary Report — International Algebraic Language, *Comm. ACM* **1**, No. 12(1958), 8.

[5] Report on the Algorithmic Language ALGOL by the ACM Committee on Programming Languages and the GAMM Committee on Programming, edited by A. J. Perlis and K. Samelson, *Numerische Mathematik* Bd. **1**, S. 41—60 (1959).

[6] William Turanskit, az amerikai csoport tagját, éppen az 1960 januári konferencia előtt autó ütötte el és életét vesztette.

Az ülést megelőzően Peter Naur egy új jelentés-tervezetet készített a korábbi előzetes jelentés és az előkészítő üléseken elhangzott javaslatok anyagából. Ezt a párizsi konferencia a végleges jelentéshez tárgyalási alapnak fogadta el, majd a konferencia e jelentés minden egyes fejezetét megvitatta és megállapodásra törekedett. Az alábbi jelentés a bizottság elgondolásait tartalmazza, illetőleg ezeknek az elgondolásoknak azt a részét, amelyben valamennyien megállapodtak.

Az előzetes ALGOL jelentéshez hasonlóan a nyelv háromféle alakját, „szintjét" különböztetjük meg, mégpedig a hivatkozási nyelvet, a publikációs nyelvet és a gépi reprezentánsokat.

A HIVATKOZÁSI NYELV

1. Ez a bizottság munkájának tárgyát képező nyelv.
2. Ez a definiáló nyelv.
3. E nyelv jelöléseit a kölcsönös megértés kényelmére való törekvés szabja meg, nem pedig valamely számológép megkötöttségei vagy valamely kódrendszer vagy a tiszta matematikai jelölésrendszer.
4. Ez a nyelv az alap és vezérfonal a fordító programok készítői számára.
5. Ez a nyelv a vezérfonal minden gépi reprezentáns számára.
6. A publikációs nyelvről bármely helyileg alkalmasnak talált gépi reprezentánsra való átíráshoz ez a nyelv a vezérfonal.
7. Ezt a nyelvet használják maguk az ALGOL nyelvről szóló főbb közlemények.

A PUBLIKÁCIÓS NYELV

1. A publikációs nyelv a kézírásnak és a nyomdatechnikának megfelelően a hivatkozási nyelvhez képest változtatásokat enged meg (pl. indexeket, közöket, exponenseket, görög betűket).
2. Ez a nyelv számítási eljárások leírására és közlésére használható
3. A publikációs nyelvben használatos jelzések országonként különbözhetnek egymástól, de biztosítani kell, hogy e jelölések kölcsönösen egyértelmű megfelelkezésben legyenek a hivatkozási nyelvvel.

A GÉPI REPREZENTÁNSOK

1. A nyelvnek minden egyes ilyen alakja a hivatkozási nyelv egy-egy leszűkítése, amit csak az tesz szükségessé, hogy a szabványos beviteli berendezések csak korlátolt számú jel használatát teszik lehetővé.
2. A gépi reprezentánsok mindegyike egy-egy adott gép jelkészletét használja és ezt a nyelvet érti meg az illető gépre kidolgozott fordító program.
3. Minden ilyen reprezentánshoz tartoznia kell speciális átírási szabályzatnak a publikációs vagy a hivatkozási nyelvről való átírás számára.

Ami a hivatkozási nyelvről egy a publikálásra alkalmas nyelvre való átírást illeti, többek között a következő szabályokat javasoljuk:

Hivatkozási nyelv	Publikációs nyelv
Index-zárójelek: [ ]	A zárójelbe foglalt sorrész süllyesztése és a zárójelek elhagyása.

Hatványozás: ↑	A kitevő szokásos emelése.
Kerek zárójelek: ( )	Tetszőleges alakú kerek, szögletes, kapcsos zárójelek.
Tizes alap jelzése: $_{10}$	A tizesnek és az őt követő egész számnak emelése és a tizes elé a szorzásjel betoldása.

A HIVATKOZÁSI NYELV LEÍRÁSA

Ami egyáltalán elmondható, az elmondható világosan is; amiről pedig beszélni nem lehet, arról hallgatni kell.

Ludwig Wittgenstein

1. A nyelv szerkezete

Amint a bevezetésben említettük, az algoritmikus nyelvnek három fajta alakja van, — a hivatkozási nyelv, a publikációs nyelv és a gépi reprezentánsok —, és a következők a hivatkozási nyelvre vonatkozólag írják le a kifejtendőket. Ezt úgy értsük, hogy minden egyes objektumot, amely e nyelvben értelmezve van, a szimbolumoknak egy adott készlete segítségével ábrázolunk és csak a szimbolumok megválasztásában térhet el ettől a másik két fajta reprezentáció. A szerkezet és a tartalom minden reprezentánsban meg kell, hogy egyezzék.

Az algoritmikus nyelv célja számítási eljárások leírása. A számítási szabályok leírására szolgáló alapfogalom az aritmetikai kifejezés közismert fogalma, amely aritmetikai kifejezés számokat, változókat és függvényeket, mint alapelemeket tartalmaz. Az ilyen kifejezésekből kiindulva „aritmetikai kompozíciós szabályok" alkalmazásával explicit formulákat, „értékadó utasítás"-oknak nevezett önálló nyelvi egységeket alakítunk.

Avégett, hogy a számítási eljárás menete demonstrálható legyen, bizonyos nem-aritmetikai utasítások bevezetése is szükséges, ezek pl. elágazásokat (alternatívákat) vagy bizonyos számítási utasítások ismételtetését írhatják elő. Mivel pedig ilyen utasítások végrehajtásához szükséges, hogy egyes utasítások másokra utaljanak, az utasításokat cimkékkel láthatjuk el. Utasítások sorozata úgynevezett utasítás-zárójelek segítségével egyetlen „összetett-utasítássá" kapcsolható össze.

Az utasításokat „deklarációk" támasztják alá, amelyek nem számítási utasítások, hanem tájékoztatást nyújtanak a fordítónak (fordítóprogramnak) az utasításokban szereplő objektumok előfordulásáról és bizonyos tulajdonságairól, mint pl. egy változó lehetséges értékeinek típusáról, a számok egy „tömb"-jének (pl. mátrixnak) a dimenziójáról, sőt esetleg pl. azoknak a szabályoknak összességéről, amelyek egy függvényt értelmeznek. Minden deklaráció egy összetett utasításhoz tartozik és csak arra érvényes. Olyan összetett utasítást, amely deklarációkat is tartalmaz, „blokk"-nak nevezünk.

Programnak nevezünk egy *önálló* összetett utasítást, vagyis egy olyan összetett utasítást, amely nem része más összetett utasításnak és nem használ fel olyan összetett utasításokat, amelyeket nem tartalmaz.[7]

Az alábbiakban ismertetjük az ALGOL nyelv szintaxisát és szemantikáját.[8]

1.1 A szintaktikus leírás formulanyelve

Az ALGOL nyelv alább következő ismertetése két részből, mégpedig ún. „szintaktikus" és „szemantikus" leírásból áll. A szintaktikus leírás (a formális definíciók rendszere) tartalmazza a kifejezések, utasítások és deklarációk szerkesztésének szabályait — kiindulva az alapszimbolumokból. A szemantikus leírás az ilyen módon szerkesztett kifejezések, utasítások és deklarációk jelentésére és használatának módjára nézve ad felvilágosítást.

A szemantikus leírás a közönséges nyelvet használja, a szintaktikus leírás pedig a formularendszerek felépítésének pl. a matematikai logikában szokásos módszereivel történik.

Az ALGOL az a tárgy-nyelv, amelyet le akarunk írni. Ehhez egy másik nyelvre, az úgynevezett „metanyelvre" van szükségünk, amellyel a tárgynyelvet leírjuk. Az ALGOL szintaxisának szabályait a metanyelv formuláival fejezzük ki.[9]

E metanyelv alapszimbolumai a következők:

1. A „metazárójel": ⟨ ⟩. A metazárójelben álló szimbolumok a metanyelv változói, amelyeknek értékei az ALGOL nyelv bizonyos szimbolumai vagy szimbolum-sorozatai. Például ⟨ab⟩, ⟨azonosító⟩, ⟨összetett utasítás⟩.
2. A függőleges elválasztó vonal: | . A felsorolásokban használatos „vagy" kötőszó rövidítéséül szolgál.
3. A „négypont-egyenlő" jel: : : = . A meghatározás, definíció jelzésére szolgál. Azt jelenti, hogy a baloldalon álló metanyelvi szimbolumot a jobboldali kifejezések értelmezik.

Például ⟨ab⟩ : : = ⟨c⟩ | ⟨d⟩ azt jelenti, hogy az ⟨ab⟩ metaváltozó definíció szerint lehet ⟨c⟩ vagy ⟨d⟩, ahol ⟨c⟩ és ⟨d⟩ már korábban definiált metanyelvi változók.

A metanyelv használja a „láncképzés" (juxtapozíció) műveletét. Tekintsük például a következő definíciót:

$$\langle x \rangle : : = \langle a \rangle \langle b \rangle$$

Ez azt jelenti, hogy az ⟨x⟩ metaváltozó definíció szerint megengedett értékei a már korábban definiált ⟨a⟩ és ⟨b⟩ egy-egy lehetséges értékének a megadott sorrendben vett párjából állhatnak.

---

[7] Amikor azt mondjuk, hogy az aritmetika pontossága általában nincsen rögzítve, vagy pedig azt, hogy egy bizonyos eljárás eredménye nincs értelmezve, akkor ez úgy értendő, hogy egy program csak akkor definiál egy számítási eljárást teljes mértékben, ha kiegészítő információ rögzíti a kívánt pontosságot, a használandó aritmetika fajtáját és a szükséges műveletek menetét minden ilyen, a számítás során előforduló esetben.

[8] Az alábbi fejezetrészt a francia fordításhoz hasonlóan mi is kibővítettük. (A fordító megjegyzése.)

[9] Lásd: J. Backus: The syntax and semantics of the proposed international algebraic language of the Zürich ACM—GAMM conference. ICIP, Paris, June 1959.

A láncképzés műveletét az ALGOL leírásában *rekurzív* definíciókra is felhasználjuk. Ennek illusztrálására szolgál a következő példa:

⟨ab⟩ : : = (| [| ⟨ab⟩ (| ⟨ab⟩ ⟨d⟩

A (jel és a [jel a tárgynyelv egy-egy szimbóluma. ⟨ab⟩ és ⟨d⟩ a metanyelv változói. Tegyük fel, hogy ⟨d⟩ már definiálva van és ⟨d⟩ lehetséges értékei a számjegyek. A fenti formula definiálja ⟨ab⟩ -t. Eszerint ⟨ab⟩ lehetséges értéke

vagy a (szimbólum
vagy akár a [szimbólum
vagy akár az ⟨ab⟩ változónak egy a definíció szerint megengedett értéke *és* utána egy (szimbólum
vagy akár az ⟨ab⟩ változónak egy a definíció szerint megengedett értéke és utána a ⟨d⟩ változó egy megengedett értéke. Az ⟨ab⟩ változónak lehetséges értékei például a következők:

(73 ((4
[86 ((2
[(((1 (37(
(1 2 3(
(((

A defmíció azonban nem engedi meg például a következőket:

(8 [4 ((
3 (((78

Hogy megkönnyítsük a metanyelvi változók megkülönböztetésére használt szimbólumok emlékezetben tartását, megnevezésükre olyan szavakat választottunk, amelyek *megközelítőleg* kifejezik az illető változó természetét. Amikor más helyen, szöveg között fordul elő egy ilyen szó, akkor is a megfelelő szintaktikus definícióval megszabott értelemben használjuk. Könnyebbség kedvéért egyes formulák több helyen is szerepelnek a szintaktikus leírásban.

Definíció:
⟨üres⟩ : : =
(vagyis az üres szimbólum-lánc).

2. Alapjelek, azonosítók, és idézetek. Alapfogalmak.

A hivatkozási nyelv a következő alapjekelből épül fel:
⟨alapjel⟩ : : = ⟨betű⟩ | ⟨számjegy⟩ | ⟨logikai érték⟩ | ⟨elhatároló jel⟩

2.1. Betűk

⟨betű⟩ : : = a|b|c|d|e|f|g|h|i|j|k|l|m|n|o|p|q|r|r|s|t|u|v|w|x|y|z|
A|B|C|D|E|F|G|H|I|J|K|L|M|N|O|P|Q|R|S|T|U|V|W|X|Y|Z

Ehhez az ábécéhez hozzávehetünk még tetszőleges más jeleket vagy el is hagyhatunk belőle jeleket, feltéve, hogy (az első esetben) a hozzávett jelek nem

azonosak valamely más alapjellel (számjeggyel, logikai értékkel vagy elhatároló jellel).

A betűknek nincsen önálló jelentésük, azonosítók és idézetek képzésére használjuk őket.[10] (Lásd 2.4. Azonosítók, 2.6. Idézetek).

2.2.1. Számjegyek

⟨számjegy⟩ : : = 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9

A számjegyeket számok, azonosítók és idézetek képzésére használjuk.

2.2.2. Logikai értékek

⟨logikai érték⟩ : : = **true** | **false**

A logikai értékeknek rögzített jelentése van, a nyilvánvaló jelentés (igaz ill. hamis).

2.3. Elhatároló jelek

⟨elhatároló jel⟩ : : = ⟨művelet jele⟩ | ⟨elválasztójel⟩ | ⟨zárójel⟩ |
⟨deklarátor jel⟩ | ⟨specifikáló alapjel⟩

⟨művelet jele⟩ : : = ⟨aritmetikai művelet jele⟩ | ⟨reláció jele⟩ |
⟨logikai művelet jele⟩ | ⟨vezérlőjel⟩

⟨aritmetikai művelet jele⟩ : : = + | − | × | / | ÷ | ↑

⟨reláció jele⟩ : : = < | ≤ | = | ≥ | > | ≠

⟨logikai művelet jele⟩ : : = ≡ | ⊃ | ∨ | ∧ | ¬

⟨vezérlőjel⟩ : : = **go to** | **if** | **then** | **else** | **for** | **do**[11]

⟨elválasztójel⟩ : : = , | . | $_{10}$ | : | ; | := | ␣ | **step** | **until** | **while** | **comment**

⟨zárójel⟩ : : = (|) | [|] | ‘ | ’ | **begin** | **end**

⟨deklarátor jel⟩ : : = **own** | **Boolean** | **integer** | **real** | **array** | **switch** | **procedure**

⟨specifikáló alapjel⟩ : : = **string** | **label** | **value**

Az elhatároló jeleknek rögzített jelentésük van. Ez a jelentés a legtöbbjüknél nyilvánvaló, a többinek pedig a megfelelő helyen megadjuk a jelentését.

---

[10] A hivatkozási nyelvben önálló alapjelként szerepelnek bizonyos szavak. (Lásd 2.2.2. és 2.3.) Ezeknek az alapjeleknek betűsorozatokból álló más objektumoktól való megkülönböztetésére az alapjelként használt szavakat *aláhúzzuk*. Ezzel azt jelöljük, hogy az aláhúzott betűcsoport semmilyen összefüggésben sincsen azokkal az egyes betűkkel, amelyekből áll. Nyomdatechnikailag sokszorosított ALGOL-szövegekben megengedett az alapjelként használt szavaknak aláhúzás helyett *vastagbetűs szedéssel* való megkülönböztetése. Ekkor azonban ALGOL-szövegeken belül a vastagbetűs szedés más célra nem használható. (Fordító megjegyzése.)

[11] A **do** jel a ciklusutasításokban használatos. Semmi kapcsolata sincs az előzetes jelentésben szereplő **do**-val, amelyet nem vettünk fel az ALGOL 60 nyelvbe.

Az olyan tipográfiai sajátosságoknak, mint amilyen a szóköz vagy új sor kezdése, a hivatkozási nyelvben semmiféle jelentésük sincsen, ilyesmik azonban szabadon használhatók az olvasás megkönnyítésére.

Avégett, hogy a program jelei közé tetszőleges szövegeket is felvehessünk, a következő megállapodásokat tesszük az ilyen „kommentárokra" vonatkozólag:

Az alapjelek következő sorozata:	*egyenértékű a következővel:*
; **comment** ⟨bármilyen jelsorozat, amely nem tartalmazza a ; jelet⟩	;
begin comment ⟨bármilyen jelsorozat, amely nem tartalmazza a ; jelet⟩	**begin**
end ⟨bármilyen jelsorozat, amely nem tartalmaz sem *end*, sem *else*, sem ; jelet⟩	**end**

Egyenértékűség itt azt jelenti, hogy a jobboldali oszlopban szereplő három szimbólum bármelyike, bármely előfordulásában — az idézeteket kivéve — helyettesíthető tetszőleges olyan jelsorozattal, amelynek szerkezete a baloldali oszlop megfelelő sorában van megadva, ez a helyettesítés azonban a program végrehajtására semilyen befolyással sincsen.

2.4. Azonosítók.

2.4.1. Szintaktikus definíció

⟨azonosító⟩ : : = ⟨betű⟩ | ⟨azonosító⟩ ⟨betű⟩ | ⟨azonosító⟩ ⟨számjegy⟩

2.4.2. Példák

q

nap

V17a

TU 104 súlya

MARGIT

2.4.3. Szemantikus definíció.

Az azonosítóknak nincs egyszer s mindenkorra rögzített jelentésük. Skaláris változók, tömbök, címkék, kapcsolók és eljárások azonosítására szolgálnak és tetszés szerint választhatjuk meg őket. (Mindamellett lásd a 3.2.4. Szabványos függvények fejezetet.)

Ugyanaz az azonosító nem szolgálhat két különböző mennyiség megjelölésére, csak abban az esetben, ha a programban szereplő deklarációk szerint különböző hatáskörük van. (Lásd a 2.7. Mennyiségek, kategóriák és hatáskörök, valamint az 5. Deklarációk c. fejezetet.)

2.5. Számok

2.5.1. Szintaktikus definíció

⟨természetes szám⟩ : : = ⟨számjegy⟩ | ⟨természetes szám⟩ ⟨számjegy⟩
⟨egész szám⟩ : : = ⟨természetes szám⟩ | + ⟨természetes szám⟩ |
— ⟨természetes szám⟩
⟨valódi tizedestört⟩ : : = . ⟨természetes szám⟩
⟨kitevőrész⟩ : : = $_{10}$ ⟨egész szám⟩
⟨tizedesszám⟩ : : = ⟨természetes szám⟩ | ⟨valódi tizedestört⟩ |
⟨természetes szám⟩ ⟨valódi tizedestört⟩
⟨előjel nélküli szám⟩ : : = ⟨tizedesszám⟩ | ⟨kitevőrész⟩ |
⟨tizedesszám⟩ ⟨kitevőrész⟩
⟨szám⟩ : : = ⟨előjel nélküli szám⟩ | + ⟨előjel nélküli szám⟩ |
— ⟨előjel nélküli szám⟩

2.5.2. Példák

0	—200.084	—.083$_{10}$—02
177	+07.43$_{10}$8	—$_{10}$7
.5384	9.34$_{10}$+10	$_{10}$—4
+0.7300	2$_{10}$—4	+$_{10}$+5

2.5.3. Szemantikus definíció

A tizedes számok jelentése a szokásos. A kitevőrész tíznek egy egészkitevős hatványaként megadott skálatényező.

2.5.4. Típusok

Az egész számok típusa **integer** (egész típusúak), az összes többi számoké **real** (valós). (Lásd az 5.1. Típusdeklaráció c. fejezetet).

2.6. Idézetek

2.6.1. Szintaktikus definíció

⟨egyszerű idézet-mag⟩ : : = ⟨alapjekekből álló tetszőleges olyan sorozat, amely nem tartalmaz sem ‘ sem ’ jelet⟩ | ⟨üres⟩
⟨idézet-mag⟩ : : = ⟨egyszerű idézet-mag⟩ | ‘ ⟨idézet-mag⟩’
⟨idézet-mag⟩ ⟨idézet-mag⟩
⟨idézet⟩ : : = ‘ ⟨idézet-mag⟩’

2.6.2. Példák

‘5k,,—‘[[[‘ ∧ = / :’Tt”
‘ . . Ez ␣ egy ␣ ‘ idézet”

2.6.3. Szemantikus definíció

Avégett, hogy a nyelvet alkalmassá tegyük az alapjelek tetszőleges sorozatának feldolgozására, bevezettük a ‘ ill. ’ idézőjeleket. A ␣ szimbólum közt jelöl, ennek a jelnek idézeteken kívül nincs jelentése. Az idézetek bizonyos eljárások aktuális paramétereiként használhatók (lásd a 3.2. Függvénykifejezések c. és a 4.7. Eljárásutasítások c. fejezetet).

2.7. Mennyiségek, kategóriák és hatáskörök

A mennyiségek következő kategóriáit különböztetjük meg: skaláris változók, tömbök, cimkék, kapcsolók és eljárások.

Egy mennyiség hatáskörét a következőképpen definiáljuk: A skaláris változók, tömbök, kapcsolók és eljárások hatásköre azon utasítások összessége, amelyekben a megfelelő mennyiség azonosítójára vonatkozó deklaráció érvényes, cimkék számára pedig azoknak az utasításoknak az összessége, amelyeket (végrehajtásban) olyan utasítás követhet, amelyben az illető cimke szerepel.

2.8. Értékek és típusok

Érték egy rendezett számhalmaz (speciális esetben egyetlen szám), vagy logikai értékek rendezett halmaza (speciális esetben egyetlen logikai érték), vagy egy cimke.

A szintaktikus egységek közül némelyekhez értéket rendelünk. Általában ezek az értékek a program végrehajtása során változnak. A kifejezéseknek és elemeiknek az értékét a 3. fejezetben definiáljuk. Egy tömbnév értéke a megfelelő indexes változók értékeinek rendezett halmaza (lásd a 3.1.4.1. alfejezetet).

A különböző „típusok” (**integer, real, Boolean**) alapjában véve értékek tulajdonságait jelölik.[12] A szintaktikus egységekhez tartozó típusok ezen egységek értékeire vonatkoznak.

3. Kifejezések

Az ALGOL nyelvben a számítási eljárásokat leíró programok fő alkatrészei az aritmetikai, a logikai és a helymegjelölő kifejezések. E kifejezések alkotó részei, — néhány elhatároló jeltől eltekintve — logikai értékek, számok, változók, függvénykifejezések, aritmetikai műveletek jelei, relációk jelei, logikai jelek és vezérlőjelek. Minthogy mind a változók, mind a függvénykifejezések szintaktikus definíciójában előfordul a kifejezés fogalma is, ezért a kifejezések és alkotórészeik definíciója szükségképpen rekurzív.

⟨kifejezés⟩ : : = ⟨aritmetikai kifejezés⟩ | ⟨logikai kifejezés⟩ |
⟨helymegjelölő kifejezés⟩

3.1. Változók

3.1.1. Szintaktikus definíció

⟨változónév⟩ : : = ⟨azonosító⟩
⟨skaláris változó⟩ : : = ⟨változónév⟩

[12] Az „**integer**-típusú”, „**real**-típusú”, „**Boolean**-típusú” kifejezés jelentése rendre : egész-típusú, valós-típusú, logikai-típusú. (A fordító megjegyzése.)

⟨indexkifejezés⟩ : : = ⟨aritmetikai kifejezés⟩
⟨indexlista⟩ : : = ⟨indexkifejezés⟩ | ⟨indexlista⟩, ⟨indexkifejezés⟩
⟨tömbnév⟩ : : = ⟨azonosító⟩
⟨indexes változó⟩ : : = ⟨tömbnév⟩ [⟨indexlista⟩]
⟨változó⟩ : : = ⟨skaláris változó⟩ | ⟨indexes változó⟩

3.1.2. Példák

epsilon
det A
a 17
Q [7,2]
x [sin n × pi/2, Q [3,n,4]]

3.1.3. Szemantikus definíció

A változó egy értéknek adott jelölés. Ezt az értéket felhasználhatjuk különböző kifejezésekben más értékek képzésére, továbbá tetszés szerint változtathatjuk „értékadó utasítások" segítségével. (Lásd a 4.2. Értékadó utasítások c. fejezetet). Az egyes változók értékeinek típusát abban a deklarációban adjuk meg, amely magára a változóra (lásd az 5.1. Típusdeklaráció c. fejezetet) vagy a megfelelő tömbnévre vonatkozik (lásd az 5.2. Tömbdeklaráció c. fejezetet).

3.1.4. Indexek

3.1.4.1. Az indexes változók olyan értékeket jelölnek, amelyek többdimenziós tömbök elemei (lásd az 5.2. Tömbdeklaráció c. fejezetet). Az indexlistában szereplő minden egyes aritmetikai kifejezés az indexes változó egy-egy indexpozícióját foglalja el és így egy indexnek nevezzük. Az indexek teljes listáját szögletes „indexzárójelbe" [ ] foglaljuk. Azt, hogy egy indexes változó a megfelelő tömb melyik elemére vonatkozik, mindig az index aktuális értéke határozza meg (lásd a 3.3. Aritmetikai kifejezések c. fejezetet).

3.1.4.2. Minden indexpozíció úgy hat, mintha egész-típusú változó volna és az index kiszámítása azzal egyenértékű, mintha ennek a fiktív változónak adnánk értéket (lásd a 4.2.4. fejezetet). Az indexes változó értéke csak akkor van definiálva, ha az indexkifejezések értéke a tömb megfelelő indexkorlátai közé esik (lásd az 5.2. Tömbdeklaráció c. fejezetet).

3.2. Függvénykifejezések

3.2.1. Szintaktikus definíció

⟨eljárásnév⟩ : : = ⟨azonosító⟩
⟨aktuális paraméter⟩ : : = ⟨idézet⟩ | ⟨kifejezés⟩ | ⟨tömbnév⟩ |
⟨kapcsolónév⟩ | ⟨eljárásnév⟩
⟨szó⟩ : : = ⟨betű⟩ | ⟨szó⟩ ⟨betű⟩
⟨paraméterelválasztó jel⟩ : : = , |) ⟨szó⟩ : (

⟨aktuális paraméter-lista⟩ : : = ⟨aktuális paraméter⟩ |
⟨aktuális paraméter⟩ ⟨paraméterelválasztó jel⟩ ⟨aktuális paraméter⟩
⟨aktuális paraméter-rész⟩ : : = ⟨üres⟩ | (⟨aktuális paraméter-lista⟩)
⟨függvénykifejezés⟩ : : = ⟨eljárásnév⟩ ⟨aktuális paraméter-rész⟩

3.2.2. Példák

sin (a—b)
J (v + s,n)
R
S (s—5) Hőmérséklet: (T) Nyomás: (P)
Keresd ki (' : =') Tárold: (Q)

3.2.3. Szemantikus definíció

A függvénykifejezések meghatározott numerikus vagy logikai értéket definiálnak, amely a megfelelő eljárásdeklaráció által adott szabályoknak (lásd az 5.4. Eljárásdeklaráció c. fejezetet) az aktuális paraméterek megadott halmazára való alkalmazásával adódik. Azokat a szabályokat, amelyek az aktuális paraméterek megadására vonatkoznak, a 4.7. Eljárásutasítások c. fejezetben adjuk meg. Nem minden eljárás definiál szükségképpen függvénykifejezést (lásd az 5.4.4. Függvénykifejezések értéke c. fejezetet).

3.2.4. Szabványos függvények

Néhány azonosítót fenntarthatunk az analízis olyan gyakran használt szabványos (standard) függvényei számára, amelyeket eljárás formájában gondolunk megadva. Ajánljuk, hogy a fenntartott lista tartalmazza a következőket:

abs (K)	a K kifejezés abszolut értéke számára
sign (K)	a K értékének előjele számára (+1, ha K > 0; —1, ha K < 0; 0, ha K = 0).
sqrt (K)	a K kifejezés értékének négyzetgyöke számára
sin (K)	a K értékének színusza számára
cos (K)	a K értékének koszínusza számára
arctan (K)	a K arkusztangensének főértéke számára
ln (K)	a K természetes logaritmusa számára
exp (K)	a K exponenciális függvénye számára.

Ezeket a függvényeket úgy tekintjük, hogy egyaránt alkalmazhatók akár egész típusú, akár valós típusú argumentum esetében.Mindegyikük valós típusú eredményt szolgáltat, a sign K függvényt kivéve, amelynek értéke mindig egész típusú. Egyes speciális reprezentációkban előfordulhat, hogy ezek a függvények explicit deklarálás nélkül is felhasználhatók (lásd az 5. Deklarációk c. fejezetet).

3.2.5. Átalakító függvények

Bármely mennyiség- vagy kifejezéspár között definiálhatunk átalakító függvényeket. Ajánljuk, hogy egyikük a szabványos függvények között is előforduljon, mégpedig az
entier (K)
függvény, amely egy valós típusú kifejezést egész típusúba alakít át és annak a legnagyobb egész számnak az értékét rendeli hozzá, amely nem nagyobb, mint a K kifejezés értéke.

3.3. Aritmetikai kifejezések

3.3.1. Szintaktikus definíció

⟨additív művelet jele⟩ : : = + | —
⟨multiplikatív művelet jele⟩ : : = × | / | +
⟨elsődleges kifejezés⟩ : : = ⟨előjel nélküli szám⟩ | ⟨változó⟩ |
⟨függvénykifejezés⟩ | (⟨aritmetikai kifejezés⟩)
⟨tényező⟩ : : = ⟨elsődleges kifejezés⟩ | ⟨tényező⟩ ↑ ⟨elsődleges kifejezés⟩
⟨tag⟩ : : = ⟨tényező⟩ | ⟨tag⟩ ⟨multiplikatív művelet jele⟩ ⟨tényező⟩
⟨feltétlen aritmetikai kifejezés⟩ : : = ⟨tag⟩ | ⟨additív művelet jele⟩ ⟨tag⟩ |
⟨feltétlen aritmetikai kifejezés⟩ ⟨additív művelet jele⟩ ⟨tag⟩
⟨feltétel-rész⟩ : : = **if** ⟨logikai kifejezés⟩ **then**
⟨aritmetikai kifejezés⟩ : : = ⟨feltétlen aritmetikai kifejezés⟩ | ⟨feltétel-rész⟩
⟨feltétlen aritmetikai kifejezés⟩ **else** ⟨aritmetikai kifejezés⟩

3.3.2. Példák

Elsődleges kifejezések:

7.395_{10}—8
összeg
w [i + 2,8]
cos (y + z × 3)
(a—3/y + vu ↑ 8)

Tényezők:

omega
összeg ↑ cos (y + z × 3)
7.394_{10}—8 ↑ w [i + 2,8] ↑ (a—3/y + vu ↑ 8)

Tagok:

U
omega × összeg ↑ cos (y + z × 3) /7.394_{10}—8 ↑
w [i + 2,8] ↑ (a—3/y + vu—8)

Feltétlen aritmetikai kifejezés:

U—Yu + omega × összeg ↑ cos (y + z × 3) /7.394_{10}—8
↑ w [i + 2,8] ↑ (a—3/y + vu—8)

Aritmetikai kifejezések:

w × u—Q (S + Cu) ↑ 2

if q > 0 **then** S + 3 × Q/A **else** 2 × S + 3 × q

if a < 0 **then** U + V **else if** a × b > 17 **then** U/V **else**
if k ≠ y **then** V/U **else** 0

a × sin (omega × t)

$0.57_{10}12 \times a\,[N \times (N—1) / 2{,}0]$

(A × arctan (y) + z) ↑ (7 + Q)

if q **then** n—1 **else** n

if a < 0 **then** A/B **else if** b=0 **then** B/A **else** z

3.3.3. Szemantikus definíció

Az aritmetikai kifejezés egy numerikus érték kiszámításának szabályát jelenti. A feltétlen aritmetikai kifejezések esetében ezt az értéket úgy kapjuk meg, hogy a kifejezésben szereplő elsődleges kifejezések aktuális numerikus értékein elvégezzük a kijelölt aritmetikai műveleteket (lásd részletesebben a 3.3.4. fejezetben).

Egy elsődleges kifejezés aktuális numerikus értéke nyilvánvaló, ha az egy szám. Változók esetében a szóban forgó numerikus érték a *pillanatnyi* érték (amelyet a szó dinamikus értelmében *utoljára* kapott), függvénykifejezések esetében pedig az az érték, amely a megfelelő eljárásdeklaráció által definiált számítási szabályoknak a kifejezésben megadott aktuális paraméterek pillanatnyi értékeire való alkalmazásával adódik (lásd az 5.4. Eljárásdeklaráció c. fejezetet). Végül, egy zárójelbe foglalt aritmetikai kifejezés értékét rekurzív elemzés útján kell kiszámítani a másik három fajta elsődleges kifejezés értékei segítségével.

Az általánosabb aritmetikai kifejezések feltételeket is tartalmaznak. Az ezekben szereplő logikai kifejezések (lásd a 3.4. Logikai kifejezések) aktuális értékei alapján az általános aritmetikai kifejezésben előforduló különböző feltétlen aritmetikai kifejezések közül egyet kell kiválasztani. A kiválasztás a következő módon történik: balról jobbra haladva egymás után kiszámítjuk a feltételekben szereplő logikai kifejezések értékét, mindaddig, amig **true** (igaz) értékét nem találunk, és akkor az egész aritmetikai kifejezés értéke egyenlő lesz az első olyan aritmetikai kifejezés értékével, amelyik ez után a logikai kifejezés után áll. (A *leghosszabb*, e helyen található aritmetikai kifejezést kell számításba vennünk.) Az

else ⟨feltétlen aritmetikai kifejezés⟩

szerkezet pedig egyenértékű az

else if true then ⟨feltétlen aritmetikai kifejezés⟩

szerkezettel.

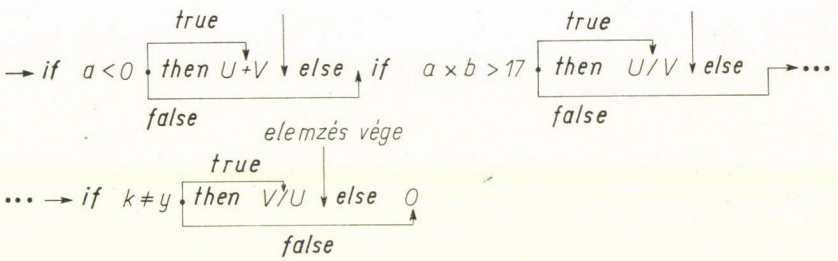
Példaképpen tekintsük a következő aritmetikai kifejezést:

if a < 0 **then** U + V **else if** a × b > 17 **then** U/V **else if** k ≠ y **then** V/U **else** 0

Nevezzük ezt az aritmetikai kifejezést rövidség kedvéért K-nak Feltesszük, hogy a K kifejezésben szereplő változók pillanatnyi értéke ismert A K kifeje-

zésben négy feltétlen aritmetikai kifejezés szerepel: U + V, U/V, V/U, és 0, e négy kifejezés általában négy különböző értéke közül az egyik és csak az egyik lesz K értéke, és hogy melyik lesz, azt a feltételekben szereplő logikai kifejezések értéke dönti el. Ha a < 0, vagy másszóval, ha az $a < 0$ logikai kifejezés **true** (igaz) értékű, akkor K értéke U + V értékével lesz egyenlő. Ha $a < 0$ nem igaz, vagyis **false** értékű, akkor meg kell néznünk, hogy az **else** elhatároló jel után mi áll. A jelen esetben az **else** után álló leghosszabb aritmetikai kifejezés újból egy (általános, feltételes) kifejezés. Ki kell tehát számítani ennek az utóbbinak feltétel-részében levő $a \times b > 17$ logikai kifejezés értékét. Ha ez az érték **true**, akkor K értéke U/V értékével egyenlő, ha pedig **false**, akkor újból az **else** után álló kifejezés vizsgálandó. Így tehát, az előbbi mintára, ha $k \neq y$ ($a\ k \neq y$ értéke **true**), akkor K értéke V/U értéke, ha pedig $k = y$ ($a\ k \neq y$ értéke **false**), akkor az **else** utáni kifejezés vizsgálandó. Most, utoljára, az **else** elhatároló jel után **feltétlen** aritmetikai kifejezés áll, mégpedig egy szám, a nulla. Ez a szerkezet a szemantikus definíció utolsó megjegyzése szerint egyenértékű az **else if true then** 0 szerkezettel, másszóval, ha az **else** után feltétlen aritmetikai kifejezés áll, és az elemzés egészen odáig eljut, mert egyetlen logikai kifejezés sem volt **true** értékű, akkor K értéke ennek az utolsó, feltétlen aritmetikai kifejezésnek értékével lesz egyenlő, tehát a jelen példában nulla.

Az elemzés menetét szemléltetik az alábbi ábra nyilai:



3.3.4. Műveleti jelek és típusok

A feltétel-részekben szereplő logikai kifejezésektől eltekintve a feltétlen aritmetikai kifejezések alkotórészeinek valós- vagy egész-típusúaknak kell lenniök (lásd az 5.1. Típus-deklaráció c. fejezetet). Az alapműveletek jeleinek a jelentését és azoknak a kifejezéseknek a típusát, amelyeket ezek segítségével nyerhetünk, az alább következő szabályok adják meg.

3.3.4.1. A +, a —, ill. a $\times$ műveleti jelek jelentése, mint szokásos, az összeadás, a kivonás ill. a szorzás művelete. A kifejezés egész-típusú lesz akkor, ha mindkét operandus egész-típusú, minden más esetben valós-típusú.

3.3.4.2. A ⟨tag⟩ / ⟨tényező⟩ ill. ⟨tag⟩ ÷ ⟨tényező⟩ szerkezetű műveletek mindketten osztást jelentenek, mégpedig a tagnak a tényező reciprokával való szorzása értelmében, figyelembe véve az elsőbbségi szabályokat (lásd a 3.3.5. fejezetet). Így pl.

a/b × 7/(p—q) × v/s

jelentése

$((((a \times (b^{-1})) \times 7) \times ((p—q)^{-1})) \times v) \times (s^{-1})$

A / műveleti jel az egész és valós típusok mind a négy lehetséges kombinációjára értelmezve van és *mindig* valós típusú eredményt szolgáltat. A ÷ műveleti jel (aritmetikai osztás jele) csak egész-típusú operandusokra van értelmezve és egész-típusú eredményt ad, még pedig a következőt:

a ÷ b = sign (a/b) × entier (abs (a/b))

(lásd a 3.2.4. és a 3.2.5. fejezetet).

3.3.4.3. A ⟨tényező⟩ ↑ ⟨elsődleges kifejezés⟩-szerkezetű művelet hatványozást jelent, amelyben a ⟨tényező⟩ az alap, az ⟨elsődleges kifejezés⟩ pedig a kitevő. Így például

2 ↑ n ↑ k jelentése $(2^n)^k$, míg

2 ↑ (n ↑ k) jelentése viszont 2^{n^m}.

Jelöljünk i-vel egy egész-típusú, r-rel egy valós-típusú és a-val egy akár egész-, akár valós-típusú számot. Az eredményt és típusát az alábbi szabályok szerint kapjuk:

$a \uparrow i$. Ha $i > 0$: $a \times a \times a \times \ldots \times a$ (i-szer); az eredmény a típusával egyező típusú.
Ha $i = 0$ és ha $a \neq 0$: 1; a-val egyező típusú.
Ha $i = 0$ és ha $a = 0$: nincs értelmezve
Ha $i < 0$ és ha $a \neq 0$: $1/(a \times a \times \ldots \times a)$,
(A nevezőben i számú tényező van); az eredmény valós-típusú.
Ha $i < 0$ és ha $a = 0$: nincs értelmezve.

$a \uparrow r$. Ha $a > 0$: exp ($r \times$ ln (a)); az eredmény valós-típusú.
Ha $a = 0$ és ha $r > 0$: 0.0; az eredmény valós-típusú.
Ha $a = 0$ és ha $r \leqq 0$: nincs értelmezve.
Ha $a < 0$: semmilyen kitevővel sincs értelmezve.

3.3.5. A műveletek elsőbbségi szabályai
Egy kifejezésben szereplő műveletek végrehajtása általában balról jobbra halad, azonban figyelembe kell venni az alábbi kiegészítő szabályokat:

3.3.5.1. A 3.3.1. fejezetben adott szintaktikus definíciók szerint a következő elsőbbségi szabályok érvényesek:

első a hatványozás: ↑
második a multpilikatív művelet: ×, /, vagy ÷,
harmadik az additív művelet: + vagy —.

3.3.5.2. Valamely kezdő zárójel és a megfelelő végzárójel között levő kifejezés külön számítandó ki és ezt az értéket használjuk fel a további számításokhoz. E szabály alapján valamely kifejezésben az egyes műveletek végrehajtásának sorrendjét megfelelően elhelyezett zárójelek segítségével tetszőlegesen szabhatjuk meg.

3.3.6. Valós mennyiségek aritmetikája

A valós típusú számokat és változókat a numerikus analízis értelmében vegyük, vagyis úgy, hogy értékük szükségképpen csak korlátozott pontossággal tekinthető definiáltnak. Ugyancsak ide értendő, hogy egy aritmetikai kifejezés kiszámított értéke bizonyos mértékben eltérhet az exakt matematikai értéktől. Nem adjuk meg azonban pontosan a követendő aritmetikai számításmódot,[13] és ezt úgy értjük, hogy különböző gépi reprezentánsok különböző módon számíthatják ki egy és ugyanannak az aritmetikai kifejezésnek értékét. A pontatlanságok következményeit a numerikus analízis módszereivel kell számon tartani. Ez az ellenőrzés a leirandó eljárás részét kell, hogy képezze és ugyancsak az algoritmikus nyelv segítségével fejezendő ki.

3.4. Logikai kifejezések

3.4.1. Szintaktikus definíció[14]

⟨reláció jele⟩ : : = < | ≦ | = | ≧ | > | ≠

⟨reláció⟩ : : = ⟨aritmetikai kifejezés⟩ ⟨reláció jele⟩ ⟨aritmetikai kifejezés⟩

⟨elsődleges logikai kifejezés⟩ : : = ⟨logikai érték⟩ | ⟨változó⟩ | ⟨függvénykifejezés⟩ | ⟨reláció⟩ | (⟨logikai kifejezés⟩)

⟨logikai tényező⟩ : : = ⟨elsődleges logikai kifejezés⟩ | ¬ ⟨elsődleges logikai kifejezés⟩

⟨logikai tag⟩ : : = ⟨logikai tényező⟩ | ⟨logikai tag⟩ ∧ ⟨logikai tényező⟩

⟨diszjunkciós kifejezés⟩ : : = ⟨logikai tag⟩ | ⟨diszjunkciós kifejezés⟩ ∨ ⟨logikai tag⟩

⟨implikációs kifejezés⟩ : : = ⟨diszjunkciós kifejezés⟩ | ⟨implikációs kifejezés⟩ ⊃ ⟨diszjunkciós kifejezés⟩

⟨feltétlen logikai kifejezés⟩ : : = ⟨implikációs kifejezés⟩ | ⟨feltétlen logikai kifejezés⟩ ≡ ⟨implikációs kifejezés⟩

⟨logikai kifejezés⟩ : : = ⟨feltétlen logikai kifejezés⟩ | ⟨feltétel-rész⟩ ⟨feltétlen logikai kifejezés⟩ **else** ⟨logikai kifejezés⟩

3.4.2. Példák

$x = -2$

$Y > V \vee z < Q$

$a + b > -5 \wedge z - d > q \uparrow 2$

$p \wedge q \vee x \neq y$

---

[13] Tehát az ALGOL nyelven leírt eljárás nem rögzíti pl. azt, hogy valós típusú változók értékét hány értékes jegynyi pontossággal kell megadnunk vagy más változók értékéből kiszámítanunk. (A fordító megjegyzése.)

[14] Itt az egyes szintaktikus egységek magyar megnevezésének rendszerében erősen eltértünk attól a rendszertől, amely a megfelelő angol kifejezések magyar fordításából adódnék, amint ez a tárgymutatóból is látható, ahol a különböző szintaktikus egységek angol megnevezése is megvan. Ez az eltérés azért látszik indokoltnak, mert az angol megnevezések egyrészt nem fedik a matematikai logikában szokásos szóhasználatot, másrészt nem illeszkednek az aritmetikai kifejezéseknél alkalmazott megnevezés-rendszerhez sem. (A fordító megjegyzése.)

$g \equiv \neg a \wedge b \wedge \neg c \vee d \vee e \supset \neg f$
if k < l **then** s > w **else** h ≦ c
if if if a **then** b **else** c **then** d **else** f **then** g **else** h < k

3.4.3. Szemantikus definíció

A logikai kifejezések mindegyike egy logikai érték meghatározására szolgáló szabályt jelent. A kiszámítási szabályok alapelve hasonló az aritmetikai kifejezések számára a 3.3.3. fejezetben megadott szabályokéhoz. Példaképpen tekintsük a következő logikai kifejezést:

if if if a **then** b **else** c **then** d **else** f **then** g **else** h < k

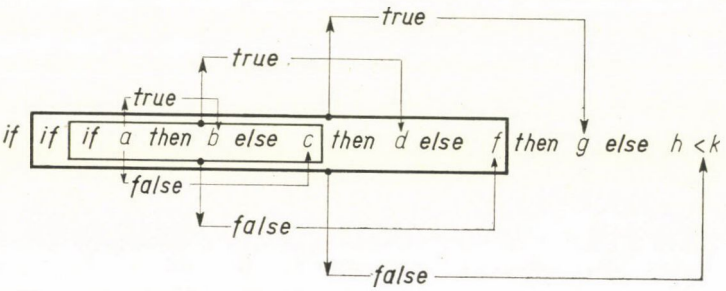
Feltesszük, hogy az a, b, c, d, f, g logikai változók és a h < k logikai kifejezés (reláció) pillanatnyi értéke ismert. Jelöljük rövidség kedvéért L-lel a példának választott logikai kifejezést. Az első **if** mutatja, hogy L általános, feltételes kifejezés, mert egy feltétel-résszel kezdődik. A feltétel:

if if a **then** b **else** c **then** d **else** f,

tehát L értéke g értékével vagy pedig h < k értékével lesz egyenlő, aszerint, amint a feltétel értéke **true** vagy pedig **false**. A feltétel, mint látható, újból általános, feltételes logikai kifejezés, amelynek feltétel-részében az

if a **then** b **else** c

feltétel szerepel. Ennek a feltételnek logikai értékét kell először kiszámítanunk és ez az érték b értékével vagypedig c értékével lesz egyenlő, aszerint, amint a értéke **true** ill. **false**. Miután ez utóbbi feltétel értékét kiszámítottuk, az előbbi (összetettebb) feltétel értékének kiszámítása következik és ennek pillanatnyi értéke dönt végül arról is, hogy L értéke g értékével vagy h < k értékével lesz-e egyenlő. Ezt a felépítést illusztrálja a következő ábra:



3.4.4. Típusok

Azokat a változókat és függvénykifejezéseket, amelyek elsődleges logikai kifejezésekként szerepelnek, logikai típusúaknak (**Boolean**) kell deklarálnunk (lásd az 5.1. Típus-deklaráció és az 5.4.4. Függvénykifejezések értéke c. fejezeteket).

3.4.5. A műveletek jelei

A relációk a **true** (igaz) értéket veszik fel, ha a megfelelő reláció a benne szereplő kifejezésekre teljesül, ellenkező esetben pedig a **false** (hamis) értéket.

A ¬ (nem), ∧ (és), ∨ (vagy), ⊃ (implikáció) és ≡ (ekvivalencia) logikai jelek jelentését a következő művelet-táblázat mutatja:

L 1	**false**	**false**	**true**	**true**
L 2	**false**	**true**	**false**	**true**
¬ L 1	**true**	**true**	**false**	**false**
L 1 ∧ L 2	**false**	**false**	**false**	**true**
L 1 ∨ L 2	**false**	**true**	**true**	**true**
L 1 ⊃ L 2	**true**	**true**	**false**	**true**
L 1 ≡ L 2	**true**	**false**	**false**	**true**

3.4.6. A műveletek elsőbbségi szabályai

Egy logikai kifejezésben szereplő műveletek végrehajtása általában balról jobbra halad, figyelembe kell azonban venni az alábbi kiegészítő szabályokat:

3.4.6.1. A 3.4.1. fejezetben megadott szintaktikus definícióknak megfelelően a következő elsőbbségi szabályok érvényesek:

elsők az aritmetikai kifejezések a 3.3.5. fejezet szerint
másodlagosak: $<$, $\leqq$, $=$, $\geqq$, $>$, $\neq$
harmadlagos: ¬
negyedleges: ∧
ötödleges: ∨
hatodlagos: ⊃
hetedleges: ≡

3.4.6.2. A zárójelek használata a 3.3.5.2. fejezetben mondottak szerint értendő.

3.5. Helymegjelölő kifejezések

3.5.1. Szintaktikus definíció

⟨cimke⟩ : : = ⟨azonosító⟩ | ⟨természetes szám⟩
⟨kapcsolónév⟩ : : = ⟨azonosító⟩
⟨kapcsolókifejezés⟩ : : = ⟨kapcsolónév⟩ [⟨indexkifejezés⟩]
⟨feltétlen helymegjelölő kifejezés⟩ : : = ⟨cimke⟩ | ⟨kapcsolókifejezés⟩ |
(⟨helymegjelölő kifejezés⟩)
⟨helymegjelölő kifejezés⟩ : : = ⟨feltétlen helymegjelölő kifejezés⟩ |
⟨feltételrész⟩ ⟨feltétlen helymegjelölő kifejezés⟩ **else**
⟨helymegjelölő kifejezés⟩

3.5.2. Példák

17

p 9

válassz [n—1]

város [**if** y $<$ 0 **then** N **else** N—1]

if Ab $<$ c **then** 17 **else** q [**if** w $\leqq$ 0 **then** 2 **else** n]

3.5.3. Szemantikus definíció

Minden helymegjelölő kifejezés olyan szabályt jelent, amely egy utasítás cimkéjének meghatározására szolgál (lásd a 4. Utasítások c. fejezetet). A kiszámítás alapelve ismét teljesen hasonló az aritmetikai kifejezésekéhez (lásd a 3.3.3. fejezetet). Az általános esetben a feltétel-részekben levő Boole-kifejezések értékrendszere egy feltétlen helymegjelölő kifejezést választ ki. Ha ez cimke, akkor máris megvan a helymegjelölő kifejezés értéke, az eredmény. Ha pedig kapcsolókifejezés, akkor a megfelelő kapcsoló-deklarációra utal (lásd az 5.3. Kapcsoló deklaráció c. fejezetet) és a kapcsoló deklarációjában felsorolt helymegjelölő kifejezések közül egyet kiválaszt, mégpedig azt az egyet, amelynek a kapcsolólistán balról jobbfelé vett sorszáma megegyezik a kapcsolókifejezés indexkifejezésének pillanatnyi numerikus értékével. Minthogy az így kiválasztott helymegjelölő kifejezés ismét lehet egy további kapcsolókifejezés, azért ez a cimkemeghatározási eljárás nyilván rekurzív.

3.5.4. Indexkifejezések

Az indexkifejezés értékének kiszámítása az indexes változók indexkifejezéseinek kiszámításához hasonlóan történik (lásd a 3.1.4.2. fejezetet).

Egy kapcsolókifejezés értéke csak abban az esetben van definiálva, ha a hozzátartozó indexkifejezés értéke az 1, 2, . . . , n természetes számok valamelyikével egyenlő, ahol n a kapcsolólista elemeinek a száma.

3.5.5. Természetes számok mint cimkék

Ha természetes számot használunk cimkének, akkor a szám elején álló esetleges nullák nem befolyásolják a cimke értékét, azaz pl. 00217 ugyanazt a cimkét jelenti, mint 217.

4. Utasítások

Az algoritmikus nyelvben az operatív egységeket *utasítások*nak nevezzük. Ezeket az utasításokat általában abban a sorrendben kell végrehajtani, ahogy egymás után vannak írva, ezt a sorrendet azonban megváltoztathatják az *átirányító utasítások*, amelyek explicite megadják az utánuk következő utasítást, másrészt pedig a *feltételes utasítások*, amelyek bizonyos utasítások átugrását eredményezhetik.

Avégett, hogy az utasításoknak ezt a *dinamikus* sorrendjét meghatározzuk, egyes utasításokat cimkékkel láthatunk el.

Mivel utasássorozatokat *összetett utasítások*ba és *blokkok*ba foglalhatunk össze, az utasítások szintaktikus definíciója szükségképpen rekurzív. Továbbá, mivel a deklarációk, amelyeket az 5. fejezet ismertet, lényeges szerepet játszanak a szintaktikus szerkezetben, az utasítások szintaktikus definíciójában feltesszük, hogy a deklarációk már definiálva vannak.[15]

4.1. Összetett utasítások és blokkok

4.1.1. Szintaktikus definíció

⟨cimkétlen alaputasítás⟩ : : = ⟨értékadó utasítás⟩ | ⟨átirányító utasítás⟩ | ⟨üres utasítás⟩ | ⟨eljárásutasítás⟩

[15] Vagyis az utasítások és deklarációk fogalmát szimultán rekurziv definícióval kell megadni. (A fordító megjegyzése).

⟨alaputasítás⟩ : : = ⟨cimkétlen alaputasítás⟩ | ⟨cimke⟩ : ⟨alaputasítás⟩
⟨feltétlen utasítás⟩ : : = ⟨alaputasítás⟩ | ⟨ciklusutasítás⟩ |
⟨összetett utasítás⟩ | ⟨blokk⟩
⟨utasítás⟩ : : = ⟨feltétlen utasítás⟩ | ⟨feltételes utasítás⟩
⟨összetett utasítás vége⟩ : : = ⟨utasítás⟩ **end** | ⟨utasítás⟩ ;
⟨összetett utasítás vége⟩
⟨blokkfej⟩ : : = **begin** ⟨deklaráció⟩ | ⟨blokkfej⟩ ; ⟨deklaráció⟩
⟨cimkétlen összetett utasítás⟩ : : = **begin** ⟨összetett utasítás vége⟩
⟨cimkétlen blokk⟩ : : = ⟨blokkfej⟩ ; ⟨összetett utasítás vége⟩
⟨összetett utasítás⟩ : : = ⟨cimkétlen összetett utasítás⟩ |
⟨cimke⟩ : ⟨összetett utasítás⟩
⟨blokk⟩ : : = ⟨cimkétlen blokk⟩ | ⟨cimke⟩ : ⟨blokk⟩

Ez a szintaktikus definíció következőképpen illusztrálható: Jelöljünk tetszőleges (általában különböző) utasításokat U-val, deklarációkat D-vel, cimkéket C-vel, akkor a fenti két főbb szintaktikus egységnek, az összetett utasításnak és a blokknak az alakja a következő lesz:

összetett utasítás:
C : C : . . . : C : **begin** U ; U ; . . . ; U **end**

blokk:
C : C : . . . : C : **begin** D ; D ; . . . ; D ; U ; U . . . ; U **end**

Meg kell jegyeznünk, hogy minden U utasítás maga is összetett utasítás vagy blokk lehet.

4.1.2. Példák

Alaputasítások:

a : = p + q
go to Budapest
KEZDET : FOLYTATÁS : w : = 7.993

Összetett utasítás:

begin x : = 0 ; **for** y : = 1 **step** 1 **until** n **do** x : = x + A [y];
if x < q **then go to** STOP **else if** x > w—2 **then go to** S;
Aw : St : W : = x + bob **end**

Blokk:

Q: **begin integer** i, k; **real** w;
for i : = 1 **step** 1 **until** n **do**
for k : = i + 1 **step** 1 **until** m **do**
begin w : = A [i, k]; A [i, k] : = A [k, i];
A [k, i] : = w **end** i-re és k-ra
end blokk Q

4.1.3. Szemantikus definíció

Minden egyes blokk automatikusan bevezet egy új jelölésrendszerszintet. Ez a következőképpen értendő: minden azonosítónak, amely a blokkon belül előfordul, megfelelő deklaráció útján (lásd az 5. deklarációk c. fejezetet) *lokális* (helyi) jelleget adhatunk az illető blokkra vonatkozólag, ami annyit jelent, hogy (a) a blokkon belül az ilyen azonosítóval megnevezett mennyiség a blokkon kívül nem létezik és (b) bármely mennyiség, amely ugyanezzel az azonosítóval van megnevezve a blokkon kívül, hozzáférhetetlen a blokk belsejéből.

Az olyan azonosító (kivéve a cimkék azonosítóit), amely szerepel egy blokkban, de nem vonatkozik rá e blokk deklarációinak egyike sem, nem-lokális jellegű erre a blokkra nézve, tehát ugyanazt a mennyiséget képviseli a blokkon belül is, amelyet a blokkon kívüli közvetlenül következő szinten. A cimkék kivételt képeznek a mondott szabály alól: mindig lokális jellegűek arra a blokkra nézve, amelyben szerepelnek.

Mivel egy blokk bármely utasítása maga is újabb blokk lehet, a „lokális" és „nem-lokális" fogalmakat rekurzív módon értsük. Egy-egy azonosító, amelyik az A blokkra nézve nem lokális, lehet akár lokális, akár nem-lokális egy olyan B blokkra nézve, amelynek A egy utasítása.

4.2. Értékadó utasítások

4.2.1. Szintaktikus definíció

⟨baloldal⟩ : : = ⟨változó⟩ : =

⟨baloldali változólista⟩ : : = ⟨baloldal⟩ | ⟨baloldali változólista⟩ ⟨baloldal⟩

⟨értékadó utasítás⟩ : : = ⟨baloldali változólista⟩ ⟨aritmetikai kifejezés⟩ | ⟨baloldali változólista⟩ ⟨logikai kifejezés⟩

4.2.2. Példák

S : = p [0] : = n : = n + 1 + s

n : = n + 1

A : = B/c—v—q × S

S [v, k + 2] : = 3 — arctan (s × zeta)

V : = Q > Y ∧ Z

4.2.3. Szemantikus definíció

Az értékadó utasítások arra szolgálnak, hogy egy vagy több változóhoz valamely kifejezés értékét rendeljék hozzá. Az általános esetben ez a hozzárendelési eljárás a következő három lépésben történik:

4.2.3.1. A baloldal változóiban szereplő esetleges indexkifejezések értékét balról jobbfelé haladva rendre kiszámítjuk.

4.2.3.2. Kiszámítjuk a baloldal utolsó : = jele után álló kifejezés értékét.

4.2.3.3. E kifejezés értékét adjuk valamennyi, a baloldali változólistán szereplő változónak, indexes változó esetén a 4.2.3.1. lépésben kiszámított indexérték mellett.

4.2.4. Típusok

A baloldali változólistán szereplő minden egyes változót azonos típusúnak kellett, hogy deklaráljunk. Ha a változók **Boolean** típusúak (logikai változók), akkor az értékadó utasításban szereplő kifejezésnek is logikai kifejezésnek kell lennie. Ha a változók valós- vagy egész-típusúak, akkor ez a kifejezés aritmetikai kifejezés kell, hogy legyen. Ha pedig ennek az aritmetikai kifejezésnek a típusa különbözik a változókétól, akkor az érték-hozzárendelésbe bele kell értenünk a megfelelő átalakító függvény automatikus behivását. A valós-típusról az egész-típusra való áttérést úgy kell értenünk, hogy a megfelelő átalakító függvény az

entier (K + 0.5)

eredményt szolgáltatja, ahol K a kifejezés értékét jelenti.

4.3. Átirányító utasítások

4.3.1. Szintaktikus definíció

⟨átirányító utasítás⟩ : : = **go to** ⟨helymegjelölő kifejezés⟩

4.3.2. Példák

go to S

go to kijárat [n + 1]

go to Város [**if** y < 0 **then** N **else** N + 1]

go to if Ab < c **then** 17 **else** q [**if** w < 0 **then** 2 **else** n]

4.3.3. Szemantikus definíció

Az átirányító utasítás megbontja az utasításoknak a felírásuk sorrendjében definiált sorrendjét, oly módon, hogy a benne szereplő helymegjelölő kifejezés értéke segítségével megadja a rákövetkező utasítást. E szerint a következő végrehajtandó utasítás az lesz, amelynek a cimkéje megegyezik e helymegjelölő kifejezés értékével.

4.3.4. Korlátozások

Mivel a cimkék lokálisak, ezért az átirányító utasítások nem vezethetnek egy blokkon kívüli helyről a blokk belsejébe.

4.3.5. Nem értelmezett értékű kapcsolókifejezéshez irányító utasítás.

Minden olyan átirányító utasítás, amelyben a helymegjelölő kifejezés egy definiálatlan értékű kapcsolókifejezés, egyenértékű az üres utasítással.

4.4. Üres utasítások

4.4.1. Szintaktikus definíció

⟨üres utasítás⟩ : : = ⟨üres⟩

4.4.2. Példa

C: **begin** . . . ; János: **end**

4.4.3. Szemantikus definíció

Az üres utasítás semilyen működést nem hajt végre. Felhasználható egy cimke elhelyezésére.

4.5. Feltételes utasítások

4.5.1. Szintaktikus definíció

⟨feltétel-rész⟩ : : = **if** ⟨logikai kifejezés⟩ **then**

⟨feltétlen utasítás⟩ : : = ⟨alaputasítás⟩ | ⟨ciklusutasítás⟩ |
⟨összetett utasítás⟩ | ⟨blokk⟩

⟨feltételesen végrehajtandó utasítás⟩ : : = ⟨feltétel-rész⟩
⟨feltétlen utasítás⟩ | ⟨cimke⟩ : ⟨feltételesen végrehajtandó utasítás⟩

⟨feltételes utasítás⟩ : : = ⟨feltételesen végrehajtandó utasítás⟩ |
⟨feltételesen végrehajtandó utasítás⟩ **else** ⟨utasítás⟩

4.5.2. Példák

if x > 0 **then** n : = n + 1

if v > u **then** V: q : = n + m **else go to** R

if s < 0 ∨ P ≦ Q **then** AA: **begin if** q < v **then** a : = v/s **else** y: = 2 × a
end else if v > s **then** a : = v—q **else if** v > s—1 **then go to** S

4.5.3. Szemantikus definíció

A feltételes utasítások bizonyos utasítások végrehajtását vagy pedig az átugrását eredményezik, a bennük szereplő logikai kifejezések tényleges pillanatnyi értékeitől függően.

4.5.3.1. Feltételesen végrehajtandó utasítás

A feltételesen végrehajtandó utasításban szereplő feltétlen utasítás végrehajtásra kerül, ha a feltételben szereplő logikai kifejezés értéke **true** (igaz), ellenkező esetben ennek az utasításnak végrehajtása kimarad és a következő utasítás jön sorra.

4.5.3.2. Feltételes utasítás

A szintaktikus definíció szerint a feltételes utasítások kétfélék lehetnek, és a következőképpen illusztrálhatók:

if L 1 **then** U 1 **else if** L 2 **then** U 2 **else** U 3; U 4,

illetőleg

if L 1 **then** U 1 **else if** L 2 **then** U 2 **else if** L 3 **then** U 3; U 4

Itt L 1, L 2 és L 3 logikai kifejezéseket, U 1, U 2, U 3 pedig feltétlen utasításokat jelentenek. U 4 a feltételes utasítást követő utasítás.

Egy feltételes utasítás végrehajtása a következőképpen írható le: A feltételekben szereplő logikai kifejezéseket balról jobbfelé haladva sorban kiszámítjuk, amíg csak egy **true** értékűt nem találunk. Az erre a logikai kifejezésre következő feltétlen utasítást végrehajtjuk. Ha ez az utasítás nem határozza meg explicite a rákövetkezőt, akkor a következő végrehajtandó utasítás az U 4 lesz, vagyis az egész feltételes utasítást követő utasítás. Így azt mondhatjuk, hogy az **else** elhatároló jelnek az a hatása, hogy az előtte álló utasításra következő utasításként az egész feltételes utasítás után álló utasítást adja meg.

A következő szerkezet

else ⟨feltétlen utasítás⟩

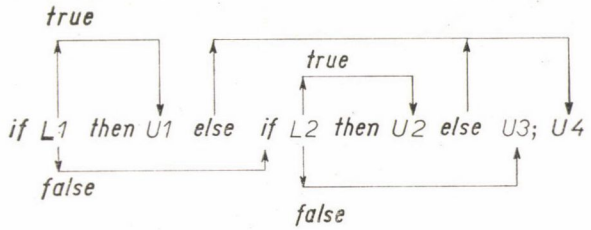
egyenértékű az

else if true then ⟨feltétlen utasítás⟩

szerkezettel.

Ha a feltétel-részekben levő logikai kifejezések egyike sem **true** értékű, akkor az egész feltételes utasítás hatása annyi, mint az üres utasításé.

Szemléltetés céljából hasznos lehet az alábbi rajz:



4.5.4. Átirányító utasítás egy feltételes utasítás belsejébe

Egy feltételes utasítás belsejébe irányító utasításnak a hatása az **else** elhatároló jel hatásáról mondottakból azonnal következik.

4.6. Ciklusutasítások

4.6.1. Szintaktikus definíció

⟨ciklusLista-elem⟩ : : = ⟨aritmetikai kifejezés⟩ | ⟨aritmetikai kifejezés⟩ **step** ⟨aritmetikai kifejezés⟩ **until** ⟨aritmetikai kifejezés⟩ | ⟨aritmetikai kifejezés⟩ **while** ⟨logikai kifejezés⟩

⟨ciklusLista⟩ : : = ⟨ciklusLista-elem⟩ | ⟨ciklusLista⟩, ⟨ciklusLista-elem⟩

⟨ciklusKezdet⟩ : : = **for** ⟨változó⟩ : = ⟨ciklusLista⟩ **do**

⟨ciklusutasítás⟩ : : = ⟨ciklusKezdet⟩ ⟨utasítás⟩ | ⟨cimke⟩ : ⟨ciklusutasítás⟩

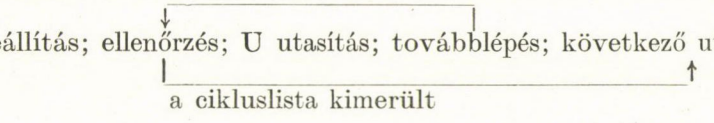
4.6.2. Példák

for q : = 1 **step** s **until** n **do** A [q] : = B [q]

for k : = 1, V 1 × 2 **while** V 1 < N **do**
for j : = I + G, L, 1 **step** 1 **until** N, C + D **do**
A [k, j] : = B [k, j]

4.6.3. Szemantikus definíció

A ciklusKezdet az utána következő U utasításnak nullaszoros vagy többszöri végrehajtását írja elő. Ezenfelül a benne szereplő változónak (a továbbiakban: a ciklus paraméterének) különböző értékeket ad. Ez az eljárás a következő ábrával illusztrálható:



Ebben a szerkezetben a „kezdőérték beállítása" a cikluslista első elemének a ciklus paraméteréhez való hozzárendelését jelenti. A „továbblépés" azt jelenti, hogy a paraméterhez a cikluslista következő elemének értékét rendeljük hozzá. Az „ellenőrzés" eldönti, hogy az ilyen módon kapott új érték a ciklus paramétere által felvehető értékek között van-e. Ha igen, akkor a cikluskezdet után álló utasítás végrehajtása, ha nem, akkor az egész ciklusutasítás után következő utasítás végrehajtása következik.

4.6.4. A cikluslista-elemek.

A cikluslista egy szabályt ad meg a ciklus paraméterének sorra adandó értékek kiszámítására. Az értékeknek ez a sorozata a cikluslista elemeiből adódik oly módon, hogy egyenként, a leírásuk sorrendjében vesszük őket sorra. A cikluslista-elemek háromféle fajtája által előállított értéksorozatot, valamint az U utasítás megfelelő végrehajtását az alábbi szabályok adják meg.

4.6.4.1. Aritmetikai kifejezés

Ez a cikluslista-elem egyetlen értéket határoz meg, mégpedig a kifejezésnek közvetlenül az U utasítás megfelelő végrehajtása előtt kiszámított értékét.

4.6.4.2. A **step** és az **until** jeleket tartalmazó elemek

Az A **step** B **until** C alakú elemek hatása, ahol A, B és C aritmetikai kifejezések, legtömörebben más ALGOL-utasítások segítségével írható le, mégpedig:

```
    V := A
C 1 : if (V—C) × sign (B) > 0 then go to Az elem kimerült;
    U utasítás;
    V := V + B; go to C1;
```

Itt V jelöli a cikluskezdetben szereplő ciklus-paramétert, „Az elem kimerült" pedig V értékének a cikluslista következő eleme alapján történő kiszámítására, vagy ha a vizsgált elem a lista utolsó eleme volt, akkor a program következő utasításának a végrehajtására utal.

4.6.4.3. A **while** szót tartalmazó elemek

Az E **while** F alakú elemek hatása, ahol E aritmetikai, F pedig logikai kifejezés, tömören ugyancsak más ALGOL-utasítások felhasználásával írható le:

```
C 3 : V := E;
      if ¬ F then go to Az elem kimerült;

      U utasítás;
      go to C 3;
```

ahol a jelölések ugyanazok, mint a 4.6.4.2. alfejezetben.

4.6.5. A ciklus paraméterének értéke a ciklusból való kilépéskor

Az U (összetettnek feltételezett) utasításból egy átirányító utasítás következtében való kilépéskor a ciklus paraméterének értéke ugyanaz lesz, mint közvetlenül az átirányító utasítás végrehajtása előtt volt.

Ha ellenben a kilépés a cikluslista kimerülése miatt történt, akkor a ciklus paraméterének értéke a kilépés után definiálatlan.

4.6.6. Ciklusutasítás belsejébe irányító utasítás

A ciklusutasításon kívülről egy a ciklusutasításon belül szereplő cimkére irányító utasítás nincs értelmezve.

4.7. Eljárásutasítások

4.7.1. Szintaktikus definíció

⟨aktuális paraméter⟩ : : = ⟨idézet⟩ | ⟨kifejezés⟩ | ⟨tömbnév⟩ | ⟨kapcsolónév⟩ | ⟨eljárásnév⟩

⟨szó⟩ : : = ⟨betű⟩ | ⟨szó⟩ ⟨betű⟩

⟨paraméterelválasztó jel⟩ : : = , |) ⟨szó⟩ : (

⟨aktuális paraméter-lista⟩ : : = ⟨aktuális paraméter⟩ | ⟨aktuális paraméterlista⟩⟨paraméterelválasztó jel⟩⟨aktuális paraméter⟩

⟨aktuális paraméter-rész⟩ : : = ⟨üres⟩ | (⟨aktuális paraméter-lista⟩)

⟨eljárásutasítás⟩ : : = ⟨eljárásnév⟩ ⟨aktuális paraméter-rész⟩

4.7.2. Példák

Nyom (A) Rendszám: (7) Eredmény: (V)

Transzponálás (W, v + 1)

Absmax (A, N, M, Yy, I, K)

Skalárszorzat (A [t, P, U], B [P], 10, P, Y)

4.7.3. Szemantikus definíció

Az eljárásutasítás egy eljárástörzs végrehajtását (a programba való behívását) eredményezi (lásd az 5.4. Eljárásdeklaráció c. fejezetet). Ha ez az eljárástörzs ALGOL-nyelven írt utasitás, akkor végrehajtása a következő műveleteknek a programon való elvégzésével lesz egyenértékű:

4.7.3.1. Érték-hozzárendelés (értékkel való behívás)

Minden formális paraméter, amely az eljárásdeklaráció fejének az érték-részében szerepel, a megfelelő aktuális paraméter értékét kapja (lásd a 2.8. Értékek és típusok c. fejezetet) és ezeket a hozzárendeléseket úgy tekintsük, mint amelyek explicite megtörténnek az eljárástörzsbe való belépés előtt. E formális paramétereket a továbbiakban az eljárás-törzsre nézve lokálisnak tekintjük.

4.7.3.2. Név-helyettesítés (névvel való hívás)

Minden olyan formális paramétert, amely az érték-részben nincs feltüntetve, az eljárástörzsben a megfelelő aktuális paraméterrel helyettesítünk, miután az utóbbit, ahol csak ezt a szintaktikus szabályok megengedik, zárójelbe tettük. Ezen eljárás során az eljárástörzsbe bekerülő és az eljárástörzsben már meglevő azonosítók közötti esetleges ütközéseket úgy küszöböljük ki, hogy szisztematikusan megváltoztatjuk az érintett formális vagy lokális azonosítókat.

4.7.3.3. A törzs behelyettesítése és végrehajtása

Végül a fentiek szerint módosított eljárástörzset az eljárásutasítás helyébe helyettesítjük és végrehajtjuk.

4.7.4. Az aktuális paraméterek formális paramétereknek való megfeleltetése.

Az eljárásutasítás paraméterlistájának mindenesetre ugyanannyi elemet kell tartalmaznia, mint amennyi az eljárásdeklaráció fejében, a formális paraméterek listáján van. Azt, hogy valamely aktuális paraméter melyik formális paraméternek felel meg, a paraméterlistán való helyzete határozza meg: ahányadik helyen áll az aktuális paraméter a paraméter-listán, annyiadik formális paraméternek felel meg.

4.7.5. Megszorítások

Ahhoz, hogy egy ilyen módon definiált eljárásutasításnak legyen értelme, nyilván szükséges, hogy azok a 4.7.3.1. ill. 4.7.3.2. alfejezetekben definiált műveletek, amelyeket az eljárástörzsön végrehajtunk, egy korrekt ALGOL-utasításra vezessenek.

Ez minden eljárásutasításra azt a megszorítást jelenti, hogy az aktuális paraméterek kategóriája és típusa összeférjen a megfelelő formális paraméterek kategóriájával és típusával. Ennek az általános szabálynak néhány speciális esete a következő:

4.7.5.1. Idézetek nem lehetnek aktuális paraméterek olyan eljárásutasításokban, amelyek ALGOL 60-utasításból álló eljárástörzzsel rendelkező eljárásra hivatkoznak (lásd a 4.7.8. fejezetet).

4.7.5.2. Minden olyan formális paraméter, amely az eljárástörzsben szereplő értékadó utasításban baloldali változóként szerepel és nem értékével van behíva, csak olyan aktuális paraméternek felelhet meg, amely maga is változó (mint a kifejezés speciális esete).

4.7.5.3. Egy formális paraméter, amely az eljárástörzsben tömbnévként szerepel, csakis olyan aktuális paraméternek felelhet meg, amely egy ugyanolyan dimenziójú tömbhöz tartozó tömbnév. Továbbá, ha ez a formális paraméter az értékével van behíva, akkor a behívás folyamán létrejövő megfelelő lokális tömb ugyanolyan indexhatárokkal kell, hogy rendelkezzék, mint az aktuális tömb.

4.7.5.4. Egy az értékével behívott formális paraméternek általában nem felelhet meg kapcsolónév vagy eljárásnév, mert ezeknek nincsen értékük. (Kivételt képeznek azon olyan eljárások nevei, amelyeknek a formális paraméter-része üres (lásd az 5.4. fejezetet), és azok, amelyek függvénykifejezést definiálnak (lásd az 5.4. és a 4.1. fejezetet). Az ilyen eljárásnevek önmagukban is komplett kifejezések.)

4.7.5.5. Minden formális paraméterre nézve megszorítások állhatnak fenn a neki megfeleltetendő aktuális paraméter típusára vonatkozólag. (Ezek a megszorítások az eljárás fejének specifikációs részében meg lehetnek adva, de az is lehet, hogy nincsenek megadva.) Az ilyen megszorításokat az eljárásutasításban nyilván figyelembe kell venni.

4.7.6. A törzs nem-lokális mennyiségei

Definiálatlan az olyan eljárásutasítás, amely kívül van egy az eljárástörzsben nem-lokális mennyiség hatáskörén.

4.7.7. Paraméterelválasztó jelek

A paraméterelválasztó jeleket egyenértékűnek tekintjük. Nem kívánjuk, hogy az eljárásutasítás paraméterelválasztó jelei és az eljárás fejében

szereplő paraméterelválasztó jelek között megfelelkezés legyen, azonkívül, hogy a számuk egyezzék meg. Az az információ, amelyet a bonyolultabb fajta paraméterelválasztó jelekkel közölhetünk, teljesen tetszőleges.

4.7.8. Kód alakjában megadott eljárástörzs

Azok a megszorítások, amelyek egy az ALGOL-tól különböző kódban írt eljárástörzset behívó eljárásutasítással kapcsolatosak, nyilván a használt kód jellemzőitől, valamint a felhasználó céljaitól függnek, s ezért a hivatkozási nyelv keretein kívül esnek.

5. Deklarációk

A deklarációk a programban szereplő azonosítók bizonyos tulajdonságait definiálják. Egy deklaráció csak egy blokkon belül érvényes, e blokkon kívül az azonosítókat más célra használhatjuk fel (lásd a 4.1.3. fejezetet).

Dinamikus szempontból ez a következő szabályokat vonja maga után: egy blokkba való beléés alkalmával (ami csak a **begin** elhatároló jelen keresztül történhet, minthogy a belül levő címkék lokálisak a blokkban és így kívülről hozzáférhetetlenek) minden, a blokk számára deklarált azonosító azt a jelentést veszi fel, amely az adott deklaráció természetéből következik. Ha ugyanezek az azonosítók már más, külső deklarációkkal definiálva voltak, akkor ettől kezdve új jelentést vesznek fel. Másrészt azok az azonosítók, amelyek nincsenek a szóbanforgó blokkban külön deklarálva, megtartják korábbi jelentésüket.

A kilépéskor (amely vagy az **end** elhatároló jelen, vagy pedig egy átirányító utasításon keresztül történhet) minden olyan azonosító, amely a blokkra nézve deklarálva volt, elveszti blokkbeli jelentését.

Minden deklarációt kiegészítésképpen elláthatunk az **own** deklarátor jellel. Ennek hatására az adott blokkba való minden újabb belépés alkalmával az **own** jellel megjelölt mennyiségek újból felveszik azt az értéket, amellyel az adott blokkból való utolsó kilépéskor rendelkeztek, míg az így meg nem jelölt, de a blokkra nézve egyébként deklarált mennyiségek értéke (egyelőre) definiálatlan marad (amíg csak egy a blokkban levő utasítás értéket nem ad nekik). A címkék, az eljárásdeklarációk formális paraméterei és esetleg a szabványos függvények azonosítói kivételével (lásd a 3.2.4. és a 3.2.5. fejezetet) valamely programban szereplő minden azonosítót deklarálni kell. Egy és ugyanazon blokkfejben egyetlen azonosítót sem szabad egynél többször deklarálni.

Szintaktikus definíció

⟨deklaráció⟩ : : = ⟨típusdeklaráció⟩ | ⟨tömbdeklaráció⟩ |
⟨kapcsolódeklaráció⟩ | ⟨eljárásdeklaráció⟩

5.1. Típusdeklaráció

5.1.1. Szintaktikus definíció

⟨típuslista⟩ : : = ⟨skaláris változó⟩ | ⟨skaláris változó⟩, ⟨típuslista⟩

⟨típus⟩ : : = **real** | **integer** | **Boolean**

⟨típusjelzés⟩ : : = ⟨típus⟩ | **own** ⟨típus⟩

⟨típusdeklaráció⟩ : : = ⟨típusjelzés⟩ ⟨típuslista⟩

5.1.2. Példák

integer p, q, s

own Boolean Acryl, n

5.1.3. Szemantikus definíció

A típusdeklaráció azt a célt szolgálja, hogy bizonyos skaláris változók nevéül szolgáló azonosítókat adott típusúaknak jelentsünk ki. A valós-típusúaknak deklarált változók csak pozitív, nulla vagy negatív értékeket vehetnek fel, az egész-típusúak pedig csak pozitív vagy negatív *egész* értékeket, illetve a nullát. A logikai típusú változók értéke csak **true** vagy **false** lehet.

Az aritmetikai kifejezésekben minden olyan helyet, amelyet egy valós-típusúnak deklarált változó elfoglalhat, ugyanúgy elfoglalhat egy egész-típusúnak deklarált változó is.

Az **own** deklarátorjel értelmére vonatkozólag lásd fentebb az 5. fejezet negyedik bekezdését.

5.2. Tömbdeklarációk

5.2.1. Szintaktikus definíció

⟨alsó korlát⟩ : : = ⟨aritmetikai kifejezés⟩

⟨felső korlát⟩ : : = ⟨aritmetikai kifejezés⟩

⟨korlátpár⟩ : : = ⟨alsó korlát⟩ : ⟨felső korlát⟩

⟨korlátpár-lista⟩ : : = ⟨korlátpár⟩ | ⟨korlátpár-lista⟩, ⟨korlátpár⟩

⟨tömbszelet⟩ : : = ⟨tömbnév⟩ [⟨korlátpár-lista⟩] | ⟨tömbnév⟩, ⟨tömbszelet⟩

⟨tömblista⟩ : : = ⟨tömbszelet⟩ | ⟨tömblista⟩, ⟨tömbszelet⟩

⟨tömbdeklaráció⟩ : : = **array** ⟨tömblista⟩ | ⟨típusjelzés⟩ **array** ⟨tömblista⟩

5.2.2. Példák

array a, b, c [7 : n, 2 : m], s[—2 : 10]

own integer array A [**if** c < 0 **then** 2 **else** 1 : 20]

real array q [—7 : —1]

5.2.3. Szemantikus definíció

A tömbdeklaráció egy vagy egyidejűleg több azonosítóról mondja ki azt, hogy indexes változókból álló tömböt képvisel és megadja e tömbök dimenziószámát, az indexek korlátait és a változók típusát.

5.2.3.1. Indexkorlátok

Egy tömb indexkorlátai a tömbnevet követő első szögletes zárójelben [ ] vannak megadva egy „korlátpár-lista" formájában. E lista minden eleme egy indexnek az alsó, ill. a felső korlátját adja meg két aritmetikai kifejezés formájában, a két korlátot pedig egy kettőspont : választja el. A korlátpár-lista minden indexnek balról jobbra haladó sorrendben megadja a korlátait.

5.2.3.2. Dimenziószámok

A dimenziószámot a korlátpár-lista elemeinek a száma adja meg.

5.2.3.3. Típusok

Közös deklarációban szereplő minden tömbnek azonos típusa van. Ha nincsen megadva explicit típusjelzés, akkor a tömböket valós típusúaknak (**real**) kell tekinteni.

5.2.4. Alsó és felső korlátok

5.2.4.1. A korlátokat megadó aritmetikai kifejezések értékét ugyanúgy kell kiszámítani, mint az indexkifejezések értékeit (lásd a 3.1.4.2. fejezetet).

5.2.4.2. Alsó ill. felső korlátként szereplő kifejezések csak olyan változóktól és eljárásoktól függhetnek, amelyek nem-lokálisak arra a blokkra nézve, amelyre a tömbrdeklaráció szól. Ennélfogva valamely program legkülső blokkjában csak konstans indexkorlátokkal megadott tömbdeklarációk lehetnek.

5.2.4.3. A tömb csak akkor van definiálva, ha a felső indexkorlátok egyike sem kisebb a megfelelő alsó indexkorlátnál.

5.2.4.4. A korlátokat megadó kifejezéseket a blokkba való minden belépés alkalmával újból kiszámítjuk.

5.2.5. Az indexes változók azonossága

Az indexes változók azonossága nincsen kapcsolatban a tömbdeklarációban megadott indexkorlátokkal. Mindazonáltal, még akkor is, ha egy tömböt **own** jellel deklarálunk, a megfelelő indexes változók közül mindig csak azoknak az értéke van definiálva, amelyeknek az indexei a legutóbb kiszámított korláton belül esnek.

5.3. Kapcsolódeklarációk

5.3.1. Szintaktikus definíció

⟨kapcsolólista⟩ : : = ⟨helymegjelölő kifejezés⟩ | ⟨kapcsolólista⟩, ⟨helymegjelölő kifejezés⟩

⟨kapcsolódeklaráció⟩ : : = **switch** ⟨kapcsolónév⟩ : = ⟨kapcsolólista⟩

5.3.2. Példák

switch S : = S1, S2, Q [m], **if** v < —5 **then** S3 **else** S4

switch Q : = p1, w

5.3.3. Szemantikus definíció

A kapcsolódeklaráció a megfelelő kapcsoló értékeit határozza meg. Ezeket az értékeket a kapcsolólistán szereplő helymegjelölő kifejezések értékeiként egyenként adjuk meg. A helymegjelölő kifejezések mindegyikéhez egy-egy természetes szám, 1, 2, ... van rendelve, mely a lista elemeinek balról jobbfelé való leszámlálásával adódik. Az indexkifejezés egy adott értékének megfelelő kapcsolókifejezés értéke (lásd a 3.5. Helymegjelölő kifejezések c. fejezetet) nem más, mint annak a kapcsolólistán levő helymegjelölő kifejezésnek az értéke, amelyhez hozzárendelt egész szám megegyezik ezzel az értékkel.

5.3.4. A kapcsolólistán levő kifejezések értékének kiszámítása

Egy a kapcsolólistán levő kifejezés értékét mindannyiszor kiszámítjuk, valahányszor csak a listának arra az elemére történik hivatkozás, amelyben

ez a kifejezés előfordul és ehhez a számításhoz a kifejezésben szereplő változóknak a pillanatnyi értékét használjuk fel.

5.3.5. A hatáskörök befolyása

Értelmetlen egy kapcsolókifejezés értékére való hivatkozás olyan helyről, amely kívül esik az illető értékhez tartozó helymegjelölő kifejezésben szereplő valamely mennyiség hatáskörén.

5.4. Eljárásdeklarációk

5.4.1. Szintaktikus definíció

⟨formális paraméter⟩ : : = ⟨azonosító⟩

⟨formális paraméter-lista⟩ : : = ⟨formális paraméter⟩ |
⟨formális paraméterlista⟩ ⟨paraméterelválasztó jel⟩
⟨formális paraméter⟩

⟨formális paraméter-rész⟩ : : = ⟨üres⟩ | (⟨formális paraméter-lista⟩)

⟨azonosítólista⟩ : : = ⟨azonosító⟩ | ⟨azonosítólista⟩, ⟨azonosító⟩

⟨érték-rész⟩ : : = **value** ⟨azonosítólista⟩; | ⟨üres⟩

⟨specifikátor⟩ : : = **string** | ⟨típus⟩ | **array** | ⟨típus⟩ **array** | **label** | **switch** |
procedure | ⟨típus⟩ **procedure**

⟨specifikációs rész⟩ : : = ⟨üres⟩ | ⟨specifikátor⟩ ⟨azonosítólista⟩; |
⟨specifikációs rész⟩ ⟨specifikátor⟩ ⟨azonosítólista⟩;

⟨eljárás feje⟩ : : = ⟨eljárásnév⟩ ⟨formális paraméter-rész⟩; ⟨érték-rész⟩
⟨specifikációs rész⟩

⟨eljárástörzs⟩ : : = ⟨utasítás⟩ | ⟨kód⟩

⟨eljárásdeklaráció⟩ : : = **procedure** ⟨eljárás feje⟩ ⟨eljárástörzs⟩ | ⟨típus⟩
procedure ⟨eljárás feje⟩ ⟨eljárástörzs⟩

5.4.2. Példák (lásd még a jelentés végén levő példákat is)

```
procedure Nyom (a) Rendszám: (n) Eredmény: (s); value n; array a; integer
    n; real s;
    begin integer k;
    s : = 0;
    for k : = 1 step 1 until n do s : = s + a [k, k]
    end

procedure Transzponálás (a) Rendszám: (n); value n
    array a; integer n;
    begin real w; integer i, k;
    for i : = 1 step 1 until n do
        for k : = i + 1 step 1 until n do
        begin w : = a [i, k];
            a [i, k] : = a [k, i] ;
            a [k, i] : = w end
        end Transzponálás
procedure Absmax (a) méretek: (n, m) Eredmény: (y) Indexek: (i, k);
```

```
comment Az m-szer n-es a mátrix legnagyobb abszolút értékű elemének
    értékét adja y-nak, a hozzátartozó indexek értékét pedig i-nek és k-nak;
array a; integer n, m, i, k; real y;
begin integer p, q;
y : = 0;
for p : = 1 step 1 until n do for q : = 1 step 1 until m do
if abs ( a [ p, q ] ) > y then begin y : = abs (a [p, q ] ); i : = p; k : = q
    end
end Absmax
```

```
procedure Skalárszorzat (a, b) Rendszám: (k, p) Eredmény: (y);
    value k;
    integer k, p; real y, a, b;
    begin real s;
    s : = 0;
    for p : = 1 step 1 until k do s : = s + a × b;
    y : = s
    end Skalárszorzat

integer procedure Négyszögjel (u); real u;
Négyszögjel : = if 0 ≦ u ∧ u ≦ 1 then 1 else 0
```

5.4.3. Szemantikus definíció

Az eljárásdeklaráció a **procedure** jel után álló eljárásnévvel megjelölt eljárás meghatározására szolgál. Az eljárásdeklaráció fő alkotórésze egy (rendszerint összetett) utasítás vagy egy kódsorozat, az „eljárástörzs", amely eljárásutasítások vagy függvénykifejezések segítségével aktivizálható annak a blokknak más részeiből, amelynek az elején, a blokkfejben, az illető eljárás deklarációja van. Az eljárástörzshöz egy fejrész is tartozik, az „eljárás feje", amely a törzsben szereplő bizonyos azonosítókról megmondja (specifikálja), hogy formális paramétereket képviselnek. Valahányszor az eljárás aktivizálása megtörténik (lásd a 3.2. Függvénykifejezések és a 4.7. Eljárásutasítások c. fejezetet) az eljárástörzs minden formális paramétere vagy felveszi egy aktuális paraméter értékét, vagy ez utóbbival helyettesítődik. Az eljárástörzsben nem formális paraméterekként szereplő azonosítók lehetnek lokálisak vagy nemlokálisak a törzsre nézve aszerint, hogy deklarálva vannak-e a törzsben vagy sem. Azok, amelyek nem-lokálisak a törzsre nézve, ugyanakkor lehetnek lokálisak arra a blokkra vonatkozólag, amelynek elején az eljárásdeklaráció szerepel.

5.4.4. Függvénykifejezések értéke

Ahhoz, hogy egy eljárásdeklaráció függvénykifejezés értékét definiálja, szükséges, hogy az eljárástörzsben szerepeljen valamilyen értéknek a megfelelő eljárásnévhez való hozzárendelése (értékadó utasítás). Ezenkívül ennek az értéknek a típusát is deklarálni kell egy típusdeklarátorral, amely az eljárásdeklarációban a legelső szimbólum kell, hogy legyen.

Az eljárás nevének az eljárástörzsben való minden más előfordulása a megfelelő eljárás aktivizálást jelenti.

5.4.5. Specifikációk

Az eljárás fejében szerepelhet egy specifikációs rész, amely a formális paraméterek fajtáiról és típusairól ad tájékoztatást nyilvánvaló értelmű jelölések segítségével. Ebben a részben minden formális paraméter csak egyszer fordulhat elő és a név szerint behívott paraméterek (lásd a 4.7.3.2. fejezetet) feltüntetése nem feltétlenül szükséges.

5.4.6. Kódsorozat alakjában megadott eljárástörzs

Meg van engedve, hogy valamely eljárástörzset egy az ALGOL-tól különböző más nyelven adjunk meg. Mivel az a szándékunk, hogy az ilyen módon való alkalmazás kizárólag a gépi reprezentánsok kérdése legyen, az ilyen nyelvekre vonatkozó további szabályok nem adhatók meg a hivatkozási nyelv keretében.

Példák eljárásdeklarációra

1. PÉLDA

```
procedure euler (fgv, összeg, epsz, elég); value epsz, elég;
integer elég; real procedure fgv; real összeg, epsz;
comment Az euler-eljárás kiszámítja a fgv (i) összegét i = 0-tól végtelenig
egy megfelelően finomított Euler-transzformáció segítségével. Az összegezést
addig végzi, amíg a transzformált sor tagjainak abszolút értéke „elég"-szer
egymás után kisebb lesz mint „epsz". Tehát meg kell adnunk egy egyváltozós,
egész argumentumú „fgv" függvényt, egy „epsz" hibakorlátot és egy „elég"
természetes számot. Az eredmény az „összeg" összeg lesz. Az euler-eljárás
különösen hatékony lassan konvergáló vagy divergens alternáló sorokra;
begin integer i, k, n, t; array m [0 : 15 ]; real mn, mp, ds;
i := n := t := 0; m [0] := fgv (0); összeg: = m [0] / 2;
következő tag: i := i + 1; mn := fgv (i);
    for k := 0 step 1 until n do
        begin mp := (mn + m[k]) / 2; m [k] := mn; mn: = mp end
            közepelés;
    if (abs (mn) < abs (m [n] )) ∧ (n < 15) then
        begin ds := mn / 2; n := n + 1; m[n] := mn end ellenőrzés
            else ds := mn; összeg := összeg + ds;
            if abs (ds) < epsz then t := t + 1 else t := 0;
            if t < elég then go to következő tag
end euler
```

2. PÉLDA[16]

[16] Ez az RK eljárás néhány új gondolatot tartalmaz, amelyek hasonlóak S. GILL, „A process for the step by step integration of differential equations in an automatic computing machine" c. (Proc. Cambridge Phil. Soc., **47** (1951), 96. o.) és E. FRÖBERG, „On the solution of ordinary differential equations with digital computing machines" c. (Fysiograf. Sällsk. Lund, Förhd. **20**, 11. sz. [1950] 136—152) cikkében ismertetett gondolatokhoz. Tudnunk kell azonban, hogy ez az eljárás esetleg nem optimális a számítási idő és a kerekítési hibák szempontjából, és hogy a számológépen még nem próbálták ki.

```
procedure RK (x, y, n, FGV, epsz, eta, xV, yV, fi);
value x, y; integer n; Boolean fi; real x, epsz, éta, xV;
    array y, yV; procedure FGV;
```

comment: RK integrálja az $y'_k = f_k(x, y_1, y_2, \ldots, y_n)$, $(k = 1,2, \ldots, n)$ differenciálegyenlet-rendszert a Runge-Kutta-módszerrel, és a megfelelő integrációs lépésközt automatikusan keresi meg.

Paraméterek: az ismeretlen $y_k(x)$ függvények y [k] kezdeti értéke az x helyen. Az egyenletrendszer n rendszáma. Egy FGV (x, y, n, z) eljárás, amely az integrálandó rendszert, vagyis az f_k függvények összességét képviseli. Az „epsz" és „éta" hibahatárok, amelyek a numerikus integrálás kívánt pontosságát határozzák meg. Az integrációs intervallum vége, xV. Az yV kimenő paraméter a megoldást jelenti az xV pontban. A „fi" logikai változó, amelynek az RK egyszeri vagy első behívásakor mindig a **true** értéket kell adnunk, ha azonban az y függvényeket több közbülső $x_0, x_1, \ldots, x_n$ pontban kell meghatározni és ezért az eljárás ismételten behívásra kerül ($k = 0,1, \ldots, n-1$ esetén $x = x_k$-val és $xV = x_{k+1}$-gyel), akkor az eljárás rövidítése (számítási idő megtakarítása) céljából fi = **false** is alkalmazható. Az FGV eljárás bejárati paraméterei legyenek x, y és n, míg a z kijárati paraméter képviseli a $z[k] = f_k(x, y[1], y[2], \ldots, y[n])$ deriváltakat az x és az y-ok aktuális értékeire. Fellép egy „comp" nevű eljárás is, mint nem-lokális azonosító;

```
begin
    array z, y1, y2, y3 [1 : n ]; real x1, x2, x3, H;
Boolean ki; integer k, j; own real s, Hs;
procedure RK1L (x, y, h, xv, yv); real x, h, xv; array y, yv;
    comment RK1L egyetlen integrációs lépést végez el a Runge—Kutta-
módszer szerint az x és y [k] kezdeti értékek felhasználásával. Eredményül az
xv = x + h és az yv [k] kijárati paraméterek adódnak, ahol az utóbbi az xv
ponthoz tartozó megoldást jelenti. Fontos: Az n, FGV és a z paraméter nem-
lokális mennyiség az RK1L-ben;
    begin array w [1 : n], a [1 : 5]; integer k, j;
        a [1]: = a [2] : = a [5] : = h /2; a [3] : = a [4] : = h; xv : = x;
        for k : = 1 step 1 until n do yv [k] : = w [k] : = y [k];
        for j : = 1 step 1 until 4 do
    begin
        FGV (xv, w, n, z);
        xv : = x + a [j];
        for k : = 1 step 1 until n do
        begin w [k] : = a [k] + a [j] × z [k];
            yv [k] : = yv [k] + a [j + 1] × z [k]/3
        end k
    end j
end RK1L;
```

Program eleje:

```
    if fi then begin H: = xV—x; s: = 0    end
        else H: = Hs; ki: = false;
```

```
AA: if (x + 2.01 × H—xV > 0) ≡ (H > 0) then
      begin Hs: = H; ki: = true; H: = (xV—x)/2 end if;
      RK1L (x, y, 2 × H, xl, yl);
BB:  RK1L (x, y, H, x2, y2); RK1L (x2, y2, H, x3, y3);
       for k: = 1 step 1 until n do
      if comp (y1 [k], y3 [k], éta) > epsz then go to CC;
```

comment: comp (a, b, c) egy függvénykifejezés, amelynek értéke az a és b változók mantisszái különbségének az abszolút értékével egyenlő, miután e mennyiségek kitevőit egyenlővé tettük az eredetileg adott a, b, c paraméterek közül a legnagyobb kitevőjűnek a kitevőjével;

```
x: = x3; if ki then go to DD;
      for k: = 1 step 1 until n do y[k]: = y3 [k];
      if s = 5 then begin s: = 0; H: = 2 × H end if;
      s : = s + 1; go to AA;
CC: H: = 0.5 × H; ki: = false; x1: = x2;
      for k: = 1 step 1 until n do y1[k]: = y2 [k]; go to BB;
DD: for k: = 1 step 1 until n do yV [k]: = y3 [k]
end RK eljárás
```

A DEFINÍCIÓK, A FOGALMAK ÉS A SZINTAKTIKUS EGYSÉGEK TÁRGYMUTATÓJA

Az utalások a vonatkozó fejezet számával vannak megadva. Az utalások három csoportra oszlanak:

1. „def". A „def." rövidítés után álló fejezetszámok a szintaktikus definícióra vonatkoznak (ha egyáltalán van ilyen).

2. „szint." A „szint." rövidítés után álló felezetszámok a metanyelvi formulákban való előfordulásokat jelzik. Az olyan utalások melyek már a def-csoportban is előfordultak, itt nem ismétlődnek.

3. „szöveg." A „szöveg" szót követő fejezetszámok a szövegben előforduló definíciókra utalnak.

Mindazok az alapjelek, amelyek nem aláhúzott (vastagbetűs) szavak, a tárgymutató elején vannak összegyűjtve.

A példákban való előfordulásokat a tárgymutató mellőzi.

\+ , lásd: összeadás jele
— , lásd: kivonás jele
× , lásd: szorzás jele
/ , lásd: osztás jele
÷ , lásd: aritmetikai osztás jele
↑ , lásd: hatványozás jele
< , ≦, = , ≧, >, ≠, lásd: reláció jele
¬, ∧, ∨, ⊃, ≡, lásd: logikai művelet jele
, , lásd: vessző
. , lásd: tizedespont
10 , lásd: tizes alap

Az ALGOL nyelvben alapjelként használatos angol szavak jelentése

array	tömb, (mátrix, vektor, többdimenziós táblázat)
begin	kezdet
Boolean	Boole-féle (logikai)
comment	magyarázat
do	végezd el, tedd
else	máskülönben
end	vég
false	hamis
for	-ra, -re
go to . . .	menj . . .-hez
if	ha
integer	egész
label	címke
own	saját
procedure	eljárás
real	valós
step	lépj, lépés
string	lánc, füzér
switch	kapcsoló
then	akkor
true	igaz
until	-ig
value	érték
while	amíg csak