TARTU RIIKLIK ÜLIKOOL

Ü.Kaasik, I.Kull

Programmeerimine ALGOLis

TARTU 1972

TARTU RIIKLIK ÜLIKOOL

Ü.Kaasik, I.Kull

Programmeerimine ALGOLis

TARTU 1972

E e s s õ n a

Käesolev õppevahend on mõeldud kõigile neile üliõpilastele, kelle õppeplaanis esineb programmeerimise (või arvutite ja programmeerimise) kursus.

Õppevahendi esimeses osas (koostanud I. Kull) antakse ülevaade algoritmikeelest ALGOL–60. Keele esitamisel kasutatakse nn. induktiivset defineerimismoodust, mis traditsiooniliste Backuse valemitega võrreldes on seni teenimatult tagaplaanile jäänud. Käsitletakse (paragrahvis 1) ka mõningaid algoritmikeelte esitamise ning õpetamise teoreetilisi ja metoodilisi küsimusi.

Vaatamata ALGOLi laialdasele kasutamisele ja õpetamisele on vastavaid ulatuslikumaid ja süstematiseeritud ülesandekogusid ilmunud seni väga vähe. Õppevahendi teise osaga püütaksegi seda lünka osaliselt täita. Ülesandeid on valitud ALGOLi kõikide olulisemate konstruktsioonide kohta. Rida ülesandeid on loogilise iseloomuga, programmiosade analüüsi, vigade avastamise jms. kohta, mis peaksid kaasa aitama keele sügavamale omandamisele. Et lahenduste esitamine oleks tunduvalt paisutanud käesoleva väljaande mahtu, siis pole neid kogule lisatud. Konsultatsioonivõimalusi arvestades ei tohiks see puudus raskendada õppevahendi kasutamist.

Autorid avaldavad tänu retsensentidele - M. Koidule ja R. Tombergile mitmete väärtuslike märkuste eest.

A L G O R I T M I K E E L E A L G O L - 6 0 E S I T U S

§ 1. Algoritmikeeltest ja nende ülesehituse põhimõtetest

Programmjuhtimisega arvutite (ehk raalide) võimsuse kasv ning nende üha laialdasem rakendamine tõi endaga kaasa vajaduse programmeerimise mehhaniseerimise ja automatiseerimise järele. Neil eesmärkidel loodi väga mitmesuguseid meetodeid ning süsteeme (standardprogrammide süsteemid, kompileerivad ja interpreteerivad programmid, silumisprogrammid jne.), mis 50-ndate aastate keskpaigu viisid välja nn. algoritmikeelte konstrueerimisele. Niisuguseid algoritmide (programmide) kirjapanemiseks määratud formaliseeritud keeli iseloomustavad järgmised tunnused.

1. Algoritmikeel on oma konstruktsioonide üldisema iseloomu tõttu inimesele algoritmide kirjapanemiseks ülevaatlikum ja hõlpsam, võrreldes arvuti käskude süsteemiga.

2. Algoritmikeele konstruktsioonid on nende spetsiaalse defineerimisviisi tõttu arvutil suhteliselt kergesti analüüsitavad.

3. Algoritmikeeled ei ole oluliselt seotud kasutatavate arvutite tüüpidega.

Algoritmikeele kasutamisel koostab inimene kõigepealt ülesande lahendusalgoritmi selles keeles. Seejärel teisendatakse (transleeritakse) algoritm arvutis spetsiaalse programmi - translaatori - abil arvuti käskudes esitatud programmiks ja alles pärast seda asutakse ülesande lahendamisele transleeritud programmi järgi. Et programm transleeritakse automaatselt, siis võib tõepoolest kõnelda programmeerimise automatiseerimisest. Tunnuse 3 tõttu on mõtet ka algoritmikeeltes esitatud programmide ulatuslikul publitseerimisel, nende vahetamisel arvutuskeskuste vahel jne. Nagu öeldud, eeldab algoritmikeele kasutamine vastava translaatori olemasolu. Translaatori koostamine on küll üsna komplitseeritud töö, kuid see tuleb teostada ühe kindla keele ja arvutitüübi korral ainult ühekordselt.

Üks esimesi algoritmikeeli - FORTRAN loodi USA-s aastail 1954-56. Keele ALGOL (Algorithmic Language) esimene variant valmis Euroopa ja USA matemaatikute ühistööna 1958. aastal, põhivariant ALGOL-60 aga 1960. aastal, kuid ka seda on hiljem veel modifitseeritud ja täiendatud. See keel on muutunud algoritmikeele klassikaliseks näidiseks ja ta on ka kõige laialdasemalt kasutatav.

Praegusajal ulatub algoritmikeelte koguarv sadadesse. Neid võib liigitada mitmesugustel alustel (näit. keele süntaksi, semantika, eesmärkide, kasutamisvaldkondade vms. põhjal). Esitame mõned algoritmikeelte olulisemad rühmad (alljärgnev jaotus pole välistav):

1) keeled matemaatiliste algoritmide esitamiseks: FORTRAN, ALGOL-60, ALGAMS, CPL jt.

2) keeled majanduslike algoritmide kirjapanemiseks: COBOL, TABSOL, АЛГЭК, АЛГОЛ-КОБОЛ jt.

3) keeled masintõlkeks ning muudeks lingvistilisteks eesmärkideks: COMIT, BCL jt.

4) keeled mõtlemise modelleerimiseks ja heuristiliseks programmeerimiseks: IPL - IV, IPL - V, LISP jt.

5) keeled informatsiooni otsimiseks: RECOL jt.

6) keeled nähtuste ja protsesside informatsioonilisteks modelleerimiseks: SIMULA, SIMSCRIPT, SOL, PL/1, ALGOL-68 jt.

Ka Eesti NSV-s on loodud mitmeid algoritmikeeli, peamiselt ALGOLi baasil (MALGOL, VELGOL), samuti keeli spetsiaalsete eesmärkide tarvis, nagu näit. APROKS (tehnoloogiliste protsesside kirjeldamiseks metallitöötlemisel).

Algoritmikeelte, nagu üldse formaliseeritud keelte esitamisel on vaja täpset vahet teha nn. objektkeele ja metakeele vahel. Objektkeeleks nimetatakse keelt, mida (antud käsitluses) esitatakse, mis on uurimise, õpetamise vms. objektiks. Metakeeleks nimetatakse keelt, mida kasutatakse kui vahendit objektkeele esitamiseks, uurimiseks või õpetamiseks. Nii on näiteks inglise keele õpetamisel eesti õppekeelega koolis inglise keel objektkeeleks, metakeeleks aga (enamasti) eesti keel. Käesolevas õppevahendis on objektkeeleks ALGOL-60.

Algoritmikeelte korral eristatakse veel keele teatavaid

erimodifikatsioone. Keele põhikuju nimetatakse etalonkeeleks; etalonkeele modifikatsiooni, mis on kohandatud publitseerimiseks, nimetatakse publikatsioonikeeleks; keele konkreetseks esituseks nimetatakse etalonkeele modifikatsiooni, mis saadakse konkreetse arvutitüübi ja translaatori iseärasusi arvestades.

Formaliseeritud keelte teoorias kasutatakse mõningaid traditsioonilise keeleteaduse termineid, nagu näiteks süntaks ja semantika. Süntaksi all mõistetakse reeglite hulka, millest lähtudes saab konstrueerida kõik vastava keele väljendid (ja ainult need). Semantikaks nimetatakse grammatika seda osa, kus fikseeritakse vastavate keelekonstruktsioonide sisuline tähendus.

Algoritmikeelte süntaks ja semantika tuleb esitada täpselt ja ühemõtteliselt. Semantika fikseerimisel tuginetakse vastava keele süntaksile, kasutatakse matemaatilist sümboolikat ning loomulike keelte (näit. eesti keele) mõningaid väljendusvahendeid. Süntaksi täpseks fikseerimiseks võib kasutada näiteks nn. induktiivset definitsiooni, spetsiaalseid metakeele (metalingvistilisi) valemeid [1] või nn. generatiivset grammatikat [2].

---

[1] Selle mooduse esitas J.W. Backus 1959.a. ALGOL-60 ametlikus väljaandes on keele süntaks esitatud selliste valemitega.

[2] Generatiivse grammatika formaalset aparatuuri on esimestena käsitlenud A.Thue (1914) ja E.Post (1943). Hiljem on seda lähtekohta kasutanud loomulike keelte uurimisel N. Chomsky (1957) jt., millest praegusajal on kujunenud ulatuslik matemaatilis-lingvistiline uurimissuund (vt.[4],[5]). Tähelepanuväärivа faktina võib märkida, et vana-india keelemehe Pāṇini (V saj. e.m.a.) sanskriti keele grammatika on üles ehitatud tänapäeva strukturaalse lingvistika ja generatiivse grammatika põhimõtetel.

Esitame näitena neljandsüsteemi täisarvu definitsiooni ülalnimetatud kolmel viisil.

A. Induktiivne definitsioon:

1. 0, 1, 2 ja 3 on märgita täisarvud;

2. kui α ja β on märgita täisarvud, siis $\alpha\beta$ on märgita täisarv;

3. kui α on märgita täisarv, siis α, $+\alpha$ ja $-\alpha$ on täisarvud.

Kirjutis "$\alpha\beta$" tähendab siin konstruktsioonide α ja β ühendust (konstruktsiooni α sümbolitele kirjutatakse vahetult järele β sümbolid).

B. Backuse metalingvistiliste valemite kasutamisel on vaja spetsiaalseid metamärke. Nendeks on märk "::=" (tähenduses "... on definitsiooni kohaselt"), märk "|" (sidesõna "või" tähenduses) ja sulud "⟨", "⟩" (kirjutised nendes sulgudes kuuluvad metakeelde).

Et Backuse valemite korral esitatakse iga konstruktsioon või mõiste otseselt vaid ühe valemiga (see kitsendus pole põhimõttelist laadi), siis tuleb siin induktiivse definitsiooni punktid 1 ja 2 ühendada. Täisarvu definitsiooni saame seega kujul:

⟨märgita täisarv⟩ ::= 0 | 1 | 2 | 3 |
⟨märgita täisarv⟩⟨märgita täisarv⟩

⟨täisarv⟩ ::= ⟨märgita täisarv⟩ | + ⟨märgita täisarv⟩ |
- ⟨märgita täisarv⟩

C. Generatiivne grammatika $G = (A, \Xi, R, \sigma)$ koosneb neljast komponendist, kus A tähendab nn. põhisümbolite (objektkeele tähtede) tähestikku, Ξ - abitähtede tähestikku, R - keelereeglite hulka (keelereeglid esitatakse kujul $P \to Q$, mis lubab genereerimisel sõna P asendada sõnaga Q; sõnad P ja Q võivad olla ka ühetähelised, s.t. tähed) ja σ - objektkeele sõnade genereerimise lähtesümbolit.

Antud näite puhul on hulk A järgmine:

$$A = \{ 0, 1, 2, 3, +, - \},$$

hulgad Ξ ja R on sobiv valida kujul

$$\Xi = \{ \sigma, \xi \} ,$$

$$R = \{ \sigma \to \xi , \sigma \to +\xi , \sigma \to -\xi , \xi \to \xi\xi , \xi \to 0, \xi \to 1, \xi \to 2, \xi \to 3 \} .$$

Objektkeele sõna genereeritakse sümbolist σ lähtudes R reeglite abil, kusjuures protsessi üksiksammud eraldatakse märgiga $\Rightarrow$.

Arvu +203 võib genereerida näiteks järgmiselt:

$$\sigma \Rightarrow +\xi \Rightarrow +\xi\xi \Rightarrow +\xi\xi\xi \Rightarrow +2\xi\xi \Rightarrow +20\xi \Rightarrow +203.$$

Vaadeldava grammatika kõikides reeglites $P \to Q$ on sõna P ühetäheline ja meil on tegemist generatiivse grammatika erijuhu, nn. kontekstivaba grammatikaga, mis osutub sisuliselt samaväärseks kahe eelmise defineerimisviisiga. Oluline on märkida, et induktiivse definitsiooni, Backuse metalingvistiliste valemite ja kontekstivaba grammatika kasutamisel saab algoritmiliselt kindlaks teha, kas vaadeldav sõna kuulub mingisse kindlasse liiki või mitte. Selle asja-

olu tõttu ongi algoritmikeele kõiki konstruktsioone võimalik arvutis analüüsida.

ALGOL-60 süntaksi esitamisel on seni kasutatud enamasti Backuse valemeid (vt. näit. [1], [2], [3] ja [7]). Propedeutilise käsitlusviisi näitena pakuvad huvi [12] ja [13], kus viimases on programmide koostamist ja analüüsimist selgitatud graafiliste skeemide (blokk-skeemide) abil. Generatiivse grammatika aparatuuri pole algoritmikeelte süntaksi esitamisel seni veel nähtavasti kasutatud. Selle moodu-se tugevaks küljeks on süntaksi üleskirjutuse suur kompaktsus. Raskusi võib tekkida aga vastava kirjutise ja fraasi-de genereerimise protsessi formaliseeritud iseloomu tõttu, mistõttu see moodus sobib ainult ettevalmistatud lugejale.

Käesolevas õppevahendis esitatakse ALGOL-60 süntaks induktiivse definitsiooni abil. Induktiivne definitsioon on oma ülevaatlikkuse ja väiksema formaliseerituse astme tõttu kõigiti sobiv moodus algoritmikeelte õpetamisel, eriti aga esimesel tutvumisel vastavate keeltega. Spetsiaalsete metamärkidena kasutame siin märki "|" (või) ning metasulge "$\langle$" ja "$\rangle$". Defineerimisel vajalike metakeele tähtedena kasutame gooti tähestiku suurtähti, üksikutel juhtudel kreeka tähestiku väiketähti. Et gooti tähestik on enamikule lugejaist vähem tuttav, esitame selle:

$\mathfrak{A}$ - A; $\mathfrak{B}$ - B; $\mathfrak{C}$ - C; $\mathfrak{D}$ - D; $\mathfrak{E}$ - E; $\mathfrak{F}$ - F; $\mathfrak{G}$ - G;

$\mathfrak{H}$ - H; $\mathfrak{I}$ - I; $\mathfrak{J}$ - J; $\mathfrak{K}$ - K; $\mathfrak{L}$ - L; $\mathfrak{M}$ - M; $\mathfrak{N}$ - N;

$\mathfrak{O}$ - O; $\mathfrak{P}$ - P; $\mathfrak{Q}$ - Q; $\mathfrak{R}$ - R; $\mathfrak{S}$ - S; $\mathfrak{T}$ - T; $\mathfrak{U}$ - U;

$\mathfrak{V}$ - V; $\mathfrak{W}$ - W; $\mathfrak{X}$ - X; $\mathfrak{Y}$ - Y; $\mathfrak{Z}$ - Z.

Kasutades eri ärifte, kirjagarnituure ja -kraade on võimalik objekt- ja metakeele märke nii valida, et nad graafiliselt ei ühti. Et rotaprindiväljaande puhul on selline moodus aga raskendatud, siis on käesolevas õppevahendis objekt- ja metakeele tähtede (ladina tähed ja kirjavahemärgid) graafiline ühtimine paratamatu. Konteksti põhjal selgub küll kohe, kas on tegemist objekt- või metakeelega. Mõningatel juhtudel kasutamegi objekt- ja metakeele eristamiseks metasulge " < " ja " > ". Et rõhutada konstruktsiooni lõpus asuva kirjavahemärgi kuulumist metakeelde (ja mitte konstruktsiooni enda juurde), asetame selle vastavast konstruktsioonist vähemalt kolme täheruumi kaugusele.

Lõpuks juhime tähelepanu veel ühele võimalusele algoritmikeelte ülesehitamiseks. See seisneb sobivate küllalt üldiste metamõistete kasutamises, nagu näiteks mõisted: täht, tähestik, sõna (resp. sõna teatavas tähestikus), mittetühi sõna (sõna, milles on vähemalt üks täht), tühisõna (sõna, milles pole ühtegi tähte), loend (teatavate elementide jada, kus järjekord on oluline), kogum (teatavate elementide jada, kus järjekord pole oluline), eraldav konstruktsioon, ümbritsev konstruktsioon. Kasutada tuleb ka sõnade ühendamise tehet ja hulgateoreetilisi tehteid. Kui need mõisted on juba defineeritud, siis osutuvad kõik spetsiaalsed ALGOLi konstruktsioonid defineeritavateks nende abil ilma induktiivse definitsiooni, Backuse valemite vms. ulatuslikuma kasutamiseta.

§ 2. Algoritmikeele ALGOL-60 tähestik

Algoritmikeele ALGOL-60 (edaspidi lihtsalt ALGOL) tähestik koosneb 116 tähest e. põhisümbolist. Nii nagu loomulike keelte tähestikus, nii esinevad ka algoritmikeelte tähestikus nn. liittähed. Keeles ALGOL on nendeks teatavad ingliskeelsed sõnad. Täpsema vahetegemise huvides kirjutatakse need liittähed tavaliselt poolpaksus kirjas (näit. keele ametlikus väljaandes). Trükitehnilistel põhjustel tõmbame aga käesolevas õppevahendis sellistele liittähtedele joone alla. ALGOLi tähestik on järgmine:

1. numbrid

0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9

2. ladina tähed

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 |
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

3. tõeväärtused (e. loogilised väärtused)

true | false

4. aritmeetiliste tehete märgid

+ | - | × | / | ÷ | ↑

5. loogiliste tehete märgid

∧ | ∨ | ⊃ | ≡ | ¬

6. võrdlusmärgid

$$> \mid \geq \mid < \mid \leq \mid = \mid \neq$$

7. järgnevusmärgid

if | **then** | **else** | **go to** | **for** | **do**

8. sulud

(|) | [|] | ' | ' | **begin** | **end**

9. eraldajad

, | . | 10 | : | ; | := | ␣ | **step** | **until** | **while** | **comment**

10. spetsifikaatorid (e. spetsifitseerivad märgid)

string | **label** | **value**

11. kirjeldajad (e. kirjeldavad märgid)

own | **Boolean** | **integer** | **real** | **array** | **switch** | **procedure**

Keele õpetamise, samuti testide esitamise ja analüüsimise huvides on otstarbekohane, et tähtedel oleksid ka oma nimed. Numbrite, ladina tähtede, tehte- ja võrdlusmärkide, sulgude ning eraldajate (v.a. ingliskeelsed sõnad) puhul kasutame nende tavalisi nimetusi. Väljaspool ALGOLi või spetsiaalselt ALGOLi tähenduses vähem kasutatavate märkide

$$/ \mid \div \mid \uparrow \mid ' \mid ' \mid \cdot \mid 10 \mid := \mid ␣$$

nimetused valime vastavalt järgmised: jagamismärk, täisarvulise jagamise märk, astendamismärk, algav rivisulg, lõppev rivisulg, kümnendpunkt, kümne aste, omistamismärk, lünk ehk tühik.

Tähti, mis pärinevad inglise keelest, võib nimetada kui vastavaid ingliskeelseid sõnu või ka nende eestikeelsete (ALGOLi seisukohast kohandatud) vastete järgi.

Esitame vastete loetelu:

true	- tõene	until	- kuni
false	- väär	while	- seni kui
if	- kui	comment	- kommentaar
then	- siis	string	- rivi
else	- vastupidisel juhul	label	- märgis
go to	- mine	value	- väärtus
for	- jaoks	own	- enda päralt
do	- täida, teosta	Boolean	- loogiline
begin	- algav operaatorsulg, algus	integer	- täisarv
end	- lõppev operaatorsulg, lõpp	real	- reaalarv
step	- samm	array	- massiiv
		switch	- lüliti
		procedure	- protseduur

§ 3. Arvud

Arvude defineerimisel keeles ALGOL kasutatakse ametlikus väljaandes mitmeid abimõisteid (näit. märgita täisarv, täisarv, kümnendmurd jne.), millest mõnel on ka iseseisev tähtsus. Esitamegi esmalt arvu traditsioonipärase definitsiooni (induktiivse definitsiooni kujul):

1. Iga number on märgita täisarv;

2. kui $\mathcal{A}$ ja $\mathcal{B}$ on märgita täisarvud, siis $\mathcal{A}\mathcal{B}$ on märgita täisarv;

3. kui $\mathcal{A}$ on märgita täisarv, siis $\mathcal{A}$, $+\mathcal{A}$ ja $-\mathcal{A}$ on täisarvud;

4. kui $\mathcal{A}$ on märgita täisarv, siis $.\mathcal{A}$ on kümnendmurd;

5. märgita täisarvud ja kümnendmurrud on kümnendarvud; kui $\mathcal{A}$ on märgita täisarv ja $\mathcal{D}$ on kümnendmurd, siis $\mathcal{A}.\mathcal{D}$ on kümnendarv;

6. kui $\mathcal{L}$ on täisarv, siis $_{10}\mathcal{L}$ on kümnendjärk;

7. kümnendarvud ja kümnendjärgud on märgita arvud; kui $\mathcal{C}$ on kümnendarv ja $\mathcal{F}$ on kümnendjärk, siis $\mathcal{C}\mathcal{F}$ on märgita arv;

8. kui $\mathcal{O}\mathcal{J}$ on märgita arv, siis $\mathcal{O}\mathcal{J}$, $+\mathcal{O}\mathcal{J}$ ja $-\mathcal{O}\mathcal{J}$ on arvud.

Arvu definitsiooni saab anda ka teisiti, mõnevõrra väiksemat abimõistete hulka kasutades. Olles defineerinud täisarvu (p. 1. - 3.), võib jätkata näiteks järgmiselt:

4'. kui $\mathcal{A}$ ja $\mathcal{B}$ on märgita täisarvud ning $\mathcal{L}$ on täisarv, siis

$$\mathcal{A} \mid .\mathcal{B} \mid \mathcal{A}.\mathcal{B} \mid {}_{10}\mathcal{L} \mid .\mathcal{B}\,{}_{10}\mathcal{L} \mid \mathcal{A}\,{}_{10}\mathcal{L} \mid \mathcal{A}.\mathcal{B}\,{}_{10}\mathcal{L}$$

on märgita arvud;

5'. kui $\mathcal{D}$ on märgita arv, siis $\mathcal{D}$, $+\mathcal{D}$ ja $-\mathcal{D}$ on arvud.

Selles definitsioonis pole vajadust abimõistete "kümnendmurd", "kümnendarv" ega "kümnendjärk" järele.

Arvud jaotatakse ALGOLis kahte tüüpi. Täisarvud (defineeritud punktides 1. - 3.) on arvud tüüpi integer. Kõik teised arvud on reaalarvud ehk arvud tüüpi real. Tähendame, et reaalarvu mõiste keeles ALGOL on oluliselt kitsam kui matemaatilises analüüsis.

Toome nüüd mõningaid arvude näiteid keeles ALGOL ühes nende semantika selgitamisega. Arvud ALGOLis

+04 | -5906 | 00.870 | -62.04 | $+_{10}+07$ | $+3_{10}15$ | $-19.24_{10}-02$

tähendavad tavalises matemaatilises sümboolikas arvusid (samas järjekorras):

4 | -5906 | 0,87 | -62,04 | 10^7 | $3\cdot10^{15}$ | $-19,24\cdot10^{-2}$.

Neid näiteid silmas pidades pole raske esitada ka mõningaid arvude lihtsustamise reegleid keeles ALGOL. Näiteks

$\pm 0\mathcal{A}.\mathcal{B}0_{10}\pm 0\mathcal{L}$ on samaväärne arvuga $\pm\mathcal{A}.\mathcal{B}_{10}\pm\mathcal{L}$,

$\pm\mathcal{A}.\mathcal{B}_{10}\pm 0$ on samaväärne arvuga $\pm\mathcal{A}.\mathcal{B}$,

$\pm 1_{10}\pm\mathcal{L}$ on samaväärne arvuga $\pm_{10}\pm\mathcal{L}$,

$\pm 0_{10}\pm\mathcal{L}$ on samaväärne arvuga .0 ,

kus $\mathcal{A}$, $\mathcal{B}$ ja $\mathcal{L}$ tähistavad märgita täisarve.

Arvude lihtsustamiseks on mõnikord otstarbekohane "nihutada" ka kümnendpunkti, muutes samal ajal kümnendjärku. Näit. $1.5_{10}-1$ on samaväärne arvuga $.15_{10}+0$, mis on

omakorda samaväärne arvuga .15 . Tähendame, et arvu lihtsustamisel peab säilima mitte ainult tema väärtus, vaid ka tüüp. Nii näiteks võib arvu 5.00 esitada lihtsamalt kujul 5.0, kuid mitte kujul 5, sest viimane arv on tüüpi integer , lähtearv aga tüüpi real .

§ 4. Identifikaatorid ja muutujad

Spetsiaalset konstruktsiooni, mis on ette nähtud muutujate, märgiste ja protseduuride identifitseerimiseks (s.o. vastavate objektide samasuse või erinevuse kindlakstegemiseks), nimetatakse identifikaatoriks. Seega täidab identifikaator sama ülesannet nagu pärisnimi.

Identifikaator defineeritakse järgmiselt:

1. Iga ladina täht on identifikaator;

2. kui $\mathcal{A}$ on identifikaator ja $\mathcal{B}$ on kas ladina täht või number, siis $\mathcal{A}\mathcal{B}$ on identifikaator.

Niisiis on identifikaator mittetühi sõna ladina tähtedest või ladina tähtedest ja numbritest, kus esimeseks sümboliks peab olema ladina täht. Identifikaatoriteks on näiteks järgmised sõnad:

A | m3gT07 | sin | summa | norm12 | pindala

Soovi korral on identifikaatoreid võimalik valida nii,

et neis kajastuks vastava muutuja vms. matemaatiline konstruktsioon (summa, korrelatsioon jne.) või sisuline tähendus (saagikus, pindala jne.).

Muutujaid võib liigitada kahest seisukohast - indeksite olemasolust või muutujate tüübist lähtudes.

Võttes aluseks indeksite olemasolu, jagunevad muutujad lihtmuutujateks (e. indeksita muutujateks) ja indeksiga muutujateks. Lihtmuutujat tähistatakse identifikaatori abil, indeksiga muutujat aga konstruktsiooniga

$$\mathcal{A}[\mathcal{L}_1, \mathcal{L}_2, \langle \ldots \rangle, \mathcal{L}_n] \qquad , n \geq 1,$$

kus $\mathcal{A}$ on identifikaator (identifikaatorit $\mathcal{A}$ nimetatakse vastava indeksiga muutuja identifikaatoriks) ja $\mathcal{L}_1, \mathcal{L}_2, \ldots, \mathcal{L}_n$ $(n \geq 1)$ aritmeetilised avaldised (nn. indeksavaldised). Aritmeetilise avaldise definitsiooni esitame alles paragrahvis 6; praegu märgime ainult niipalju, et aritmeetilisteks avaldisteks on muuhulgas arvud ja muutujad. Metakeele kirjutis "$\langle \ldots \rangle$" tähendab "ja nii edasi", metakeelne tähis (indeks) "n" märgib indeksavaldiste $\mathcal{L}_1, \mathcal{L}_2, \ldots, \mathcal{L}_n$ arvu.

Lihtmuutujate näideteks sobivad eespool toodud identifikaatorid, välja arvatud "sin", mis on standardse protseduuri identifikaatoriks. Nimelt ei tohi muutuja identifikaatorina kasutada standardsete protseduuride tähistamiseks reserveeritud identifikaatoreid (vt. § 5).

Indeksiga muutujad on näiteks

a[3] | hind[eps[i,5.8]] | T3m[a+b,c[k],d[i,j]] ,

kus esimesed kaks on ühe indeksiga, viimane aga kolme indeksiga muutuja. Tähendame, et indeksavaldisena võivad omakorda esineda indeksiga muutujad, mille indekseid (definitsiooni kohaselt) põhimuutuja indeksite hulka ei loeta. Indeksavaldiste lõppväärtusteks on alati täisarvud. Kui indeksavaldise $\mathcal{L}_i$ väärtus pole täisarv, siis muudetakse see automaatselt täisarvuks eeskirja

$$\text{entier}(\mathcal{L}_i + 0{,}5)$$

alusel [3].

Indeksiga muutujat, kui seda vaadeldakse indeksite kõikvõimalike väärtustega määratud arvuhulgana (resp. tõeväärtuste hulgana), nimetatakse massiiviks, vastavat identifikaatorit aga massiivi identifikaatoriks.

Tüübi seisukohalt jagunevad muutujad täisarvulisteks, reaalarvulisteks ja loogilisteks muutujateks ehk vastavalt muutujateks tüüpi integer, real ja Boolean . Muutujate tüübid fikseeritakse muutujate kirjeldamisel, mida käsitleme paragrahvis 15.

---

[3] Kirjutisega entier(x) tähistatakse arvu x mitte ületavat suurimat täisarvu. Tähis entier($\mathcal{L}_i$+0.5) tähendab seega arvule $\mathcal{L}_i$ lähimat täisarvu, kusjuures .5-ga lõppevad arvud (näit. 1.5) ümardatakse ülespoole.

§ 5. Protseduurid ja rivid

Terminiga "protseduur" tähistatakse keeles ALGOL kaht konstruktsiooni - funktsiooni (mis on ette nähtud mingi arvulise või loogilise väärtuse arvutamiseks) ja protseduurioperaatorit (mis võimaldab algoritmiliselt töödelda mistahes arvulist, loogilist või alfabeetilist informatsiooni, samuti informatsiooni sisestada või väljastada).

Teatud sagedamini kasutatavaid protseduure saab programmis kasutada ilma täiendava informatsioonita - nende nn. standardsete protseduuride puhul on vajalikud algoritmid arvutis juba salvestatud. Kõikide teiste protseduuride kasutamise korral tuleb aga programm varustada kirjeldustega, kus esitatakse vastava protseduuri tööks vajalik algoritm. Kirjeldus tuleb tuua vastava programmi (bloki) kirjelduste osas, mis on programmi põhiosast (põhiprogrammist) eraldatud. Põhiprogrammis kujutab protseduur seega instruktsiooni teatava programmiosa (alamprogrammi) poole pöördumiseks, kusjuures fikseeritakse faktiliste parameetrite ehk protseduuri argumentide väärtused. Protseduur võib olla ka ilma ühegi faktilise parameetrita (näit. juhul, kui protseduuri faktilised parameetrid programmis puht-graafiliselt ei muutu).

Käesolevas paragrahvis vaatleme protseduuri esitamist põhiprogrammis. Protseduuri kirjeldusi käsitleme paragrahvis 16.

Parameetrita protseduur esitatakse identifikaatori abil, mida selles konstruktsioonis nimetatakse protseduuri identifikaatoriks; parameetritega protseduuri võib esitada kujul

$$\mathcal{A}(\mathcal{L}_1, \mathcal{L}_2, \langle\cdots\rangle, \mathcal{L}_n) \qquad ,n \geq 1,$$

kus $\mathcal{A}$ on identifikaator (vastava protseduuri identifikaator) ja $\mathcal{L}_1, \mathcal{L}_2, \ldots, \mathcal{L}_n$ faktilised parameetrid. Faktilisteks parameetriteks võivad olla järgmised konstruktsioonid: avaldised (defineeritakse paragrahvides 6, 7 ja 11), massiivide identifikaatorid (§4), protseduuride identifikaatorid (§5), lülitite identifikaatorid (§11) ja rivid (§5).

Koma asemel võib faktiliste parameetrite loendis parameetrite eraldamiseks kasutada ka konstruktsiooni kujul

$$)\mathcal{L}:(\qquad ,$$

kus $\mathcal{L}$ on mittetühi sõna ladina tähtedest. Tähendame, et parameetrite eraldamise moodus ega eraldajate (sõna $\mathcal{L}$) graafiline kuju ei mõjusta protseduuri täitmist.

Esitame mõningaid protseduuride näiteid (põhiprogrammis):

```
Fookus | arctan(z) | u7K(5.7₁₀-3,a[i],b+c,g(x)) |
Pln2(a,b)sk:(c)raadiused:(r,R) |
Print('kasum=␣',y)                                ,
```

kus esimesena toodud protseduur on ilma parameetriteta, neljandas on parameetrite eraldamiseks kasutatud moodust ")$\mathcal{L}$:(" (kahes kohas), viiendas on üheks faktiliseks parameetriks rivi.

Ühe ja sama funktsiooni väärtused on kõik üht tüüpi suurused, kas tüüpi integer, real või Boolean . Funktsiooni väärtuste tüüpi nimetatakse ka vastava funktsiooni tüübiks. Et protseduurioperaatoril võib olla palju resultaate, mis võivad olla ka eri tüüpi, siis protseduurioperaatori puhul tüübist ei kõnelda.

Protseduuri süntaksi definitsioonist endast ei selgu, kas on tegemist funktsiooni või protseduurioperaatoriga. See selgub aga kohe vastava konstruktsiooni kontekstist. Funktsioon saab esineda ainult kas aritmeetilise või loogilise avaldise koostisosana. Protseduurioperaator on aga omaette operaator, mis (enamasti) eraldatakse programmi teistest operaatoritest semikooloni abil. Nii näiteks programmilõigus

⟨...⟩ a + f(x,y) ⟨...⟩

f(x,y) on funktsioon, sest ta esineb aritmeetilise avaldise osana; programmilõigus

⟨...⟩ ; F(p,q,r); ⟨...⟩

on F(p,q,r) aga protseduurioperaator, sest ta esineb iseseisva operaatorina (semikoolonite vahel).

Standardseteks funktsioonideks soovitatakse võtta alljärgnevad funktsioonid, reserveerides nende jaoks kindlad identifikaatorid, mida ei tohi kasutada teises tähenduses. Seda loetelu võib ALGOLi konkreetsetes esitustes ka täiendada.

abs(*E*) , mis tähendab *E* absoluutväärtust;

entier($\mathscr{L}$) - tähendab avaldise $\mathscr{L}$ väärtust mitte ületavat suurimat täisarvu;

sign($\mathscr{L}$) - tähendab avaldise $\mathscr{L}$ märki, täpsemini

$$\text{sign}(\mathscr{L}) = \begin{cases} +1 & \text{, kui } \mathscr{L} > 0, \\ 0 & \text{, kui } \mathscr{L} = 0, \\ -1 & \text{, kui } \mathscr{L} < 0; \end{cases}$$

sqrt($\mathscr{L}$) = $+\sqrt{\mathscr{L}}$, defineeritud ainult $\mathscr{L} \geq 0$ korral;

sin($\mathscr{L}$) ja cos($\mathscr{L}$) nende tavalises tähenduses;

arctan($\mathscr{L}$) - funktsiooni Arctan peaväärtus radiaanides, s.t. $-\pi/2 < \text{arctan}(\mathscr{L}) < \pi/2$;

exp($\mathscr{L}$) = $e^{\mathscr{L}}$ (e - naturaallogaritmide alus);

ln($\mathscr{L}$) - naturaallogaritm avaldise $\mathscr{L}$ (> 0) väärtusest.

Funktsioonid entier ja sign on tüüpi integer, teised standardfunktsioonid aga tüüpi real .

Standardseteks protseduurioperaatoriteks soovitatakse võtta järgmised (vt. [2] lk. 113):

inreal($\mathscr{N}$,$\mathscr{M}$) - muutujale $\mathscr{M}$ omistatakse kanalilt (sisendseadmelt) nr. $\mathscr{N}$ toodav arvuline väärtus;

outreal($\mathscr{N}$,$\mathscr{A}$) - aritmeetilise avaldise $\mathscr{A}$ väärtus väljastatakse kanali nr. $\mathscr{N}$ kaudu;

inarray($\mathscr{N}$,$\mathscr{L}$) - identifikaatoriga $\mathscr{L}$ massiivi elementidele omistatakse kanalilt nr. $\mathscr{N}$ toodavad arvulised väärtused;

outarray($\mathcal{N}$,$\mathcal{L}$) - identifikaatoriga $\mathcal{L}$ massiivi elemendid väljastatakse kanali nr.$\mathcal{N}$ kaudu; kahel viimasel juhul järjestatakse massiivi elemendid leksikograafiliselt: näit. a[1,2] on elemendist a[2,1] eespool;

insymbol($\mathcal{N}$,$\mathcal{L}$,$\mathcal{M}$) - kanalilt nr.$\mathcal{N}$ toodavat sümbolit kontrollitakse, kas ta on ALGOLi täht või mitte. Jaataval juhul leitakse selle tähe järjekorranumber rivis $\mathcal{L}$ ja omistatakse see number muutuja $\mathcal{M}$ väärtuseks. Kui see täht rivis $\mathcal{L}$ ei esine, omistatakse muutujale $\mathcal{M}$ väärtus 0. Kui kontrollitav sümbol pole ALGOLi täht, omistatakse muutujale $\mathcal{M}$ teatav negatiivne väärtus;

outsymbol($\mathcal{N}$,$\mathcal{L}$,$\mathcal{A}$) - rivist $\mathcal{L}$ leitakse sümbol, mille järjekorranumber võrdub avaldise $\mathcal{A}$ väärtusega, ja väljastatakse see kanali nr. $\mathcal{N}$ kaudu.

length($\mathcal{L}$,$\mathcal{M}$) - rivi $\mathcal{L}$ sümbolite arv omistatakse muutujale $\mathcal{M}$.

Kolmel viimasel juhul ei võeta järjekorranumbrite määramisel arvesse kõige välimisi rivisulgusid.

Peale toodud protseduurioperaatorite kasutatakse käesolevas õppevahendis informatsiooni väljastamiseks veel standardset protseduurioperaatorit kujul

print($\mathcal{L}_1$, $\mathcal{L}_2$, $\langle\cdots\rangle$, $\mathcal{L}_n$) ,n ≥ 1,

kus faktilisteks parameetriteks võivad olla aritmeetilised ja loogilised avaldised ning massiivide identifikaatorid. Nimetatud operaatori korral trükitakse välja vastavate avaldiste ja massiivide elementide väärtused.

Ülalesitatud standardsete protseduurioperaatorite abil võib konstrueerida mitmeid teisi informatsiooni vahetust võimaldavaid operaatoreid. Et nimetatud operaatorid on seotud kasutatavate sisend- ja väljundseadmete iseärasustega, siis tuleb neid protseduure täpsustada keele ALGOL konkreetses esituses.

Lõpuks esitame veel konstruktsiooni "rivi", mis on ette nähtud opereerimiseks alfabeetilise informatsiooniga (vt. näit. protseduurioperaatoreid insymbol, outsymbol ja length). Et sellise informatsiooniga opereerimist hõlbustada, on rivi konstruktsioon hierarhiline: rivi võib sisaldada rivisid mitmel eri tasemel, samuti saame rivide ühendamisel jälle moodustada rivi. Rivi defineeritakse järgmiselt:

1. kui $\mathcal{A}$ on sõna ALGOLi tähtedest, mis ei sisalda rivisulge (" ' " ja " ' "), siis $\mathcal{A}$ on puhas rivi;

2. iga puhas rivi on lahtine rivi;

3. kui $\mathcal{L}$ ja $\mathcal{L}$ on lahtised rivid, siis $\mathcal{L}\mathcal{L}$ on lahtine rivi; kui $\mathcal{L}$ on lahtine rivi, siis ' $\mathcal{L}$ ' on lahtine rivi;

4. kui $\mathcal{L}$ on lahtine rivi, siis ' $\mathcal{L}$ ' on rivi.

Tähendame, et lahtise rivi mõiste hõlmab nii puhta rivi kui ka rivi mõiste. Näiteks kirjutistest

```
Tartu, ␣ 15. sept. | ' kuupalk ␣ =' x[t]'␣ rbl.' |
''pindala ␣' ' S = (a × h)/2 ''
```

on esimene puhas rivi, kolmas aga rivi (mis sisaldab kaks madalama tasemega rivi), kusjuures kõik need kolm konstruktsiooni on lahtised rivid.

Rivid esinevad ainult protseduuride faktiliste parameetritena. Sümbol "␣" tähendab lünka (tühikut), keeles ALGOL seda väljaspool rivi (resp. lahtist rivi) ei kasutata.

§ 6. Aritmeetilised avaldised

Keeles ALGOL on üldse kolme liiki avaldisi: aritmeetilised, loogilised (vt. §7) ja märgiseavaldised [4] (vt. §11). Aritmeetilise avaldise mõiste ALGOLis on aritmeetilise avaldise traditsioonilistest mõistetest mõnevõrra laiem: see hõlmab ka arvuliste väärtustega funktsiooni mõiste. Aritmeetiline avaldis võib peale selle sisaldada ka tingimusi (loogilisi avaldisi), mis võib viia mõttele, et loogiline avaldis tuleks defineerida enne aritmeetilist avaldist [5].

---

[4] Selle asemel on kasutatud ka terminit "nimeline avaldis" (vt. näit. [3]).

[5] Ringkäik nende mõistete defineerimisel on kõrvaldatav näit. järgmiselt: peale aritmeetilise avaldise definitsiooni p. 1–4 esitatakse loogilise avaldise definitsioon ja alles seejärel aritmeetilise avaldise definitsioon p. 5–6.

Et aga loogilise avaldise koosseisus võivad omakorda esineda aritmeetilised avaldised, siis alustame siiski aritmeetilistest avaldistest.

Aritmeetilise avaldise defineerime järgmiselt [6]:

1. Märgita arv, muutuja tüüpi real või integer, funktsioon tüüpi real või integer on aritmeetilised lihtavaldised;

2. kui $\mathcal{A}$ on aritmeetiline lihtavaldis, siis $(+\mathcal{A})$ ja $(-\mathcal{A})$ on aritmeetilised lihtavaldised;

3. kui $\mathcal{A}$ ja $\mathcal{B}$ on aritmeetilised lihtavaldised, siis

$$(\mathcal{A}\uparrow\mathcal{B}) \mid (\mathcal{A}\times\mathcal{B}) \mid (\mathcal{A}/\mathcal{B}) \mid (\mathcal{A}\div\mathcal{B}) \mid (\mathcal{A}+\mathcal{B}) \mid (\mathcal{A}-\mathcal{B})$$

on aritmeetilised lihtavaldised;

4. iga aritmeetiline lihtavaldis on aritmeetiline avaldis;

5. kui $\mathcal{A}$ on aritmeetiline lihtavaldis, $\mathcal{L}$ on aritmeetiline avaldis ja $\mathcal{P}$ on loogiline avaldis, siis

$$\underline{\text{if}}\ \mathcal{P}\ \underline{\text{then}}\ \mathcal{A}\ \underline{\text{else}}\ \mathcal{L} \qquad (1)$$

on aritmeetiline avaldis;

6. kui $\mathcal{L}$ on aritmeetiline avaldis, siis $(\mathcal{L})$ on aritmeetiline lihtavaldis.

---

[6] Käesolev aritmeetilise avaldise definitsioon erineb vastava mõiste definitsioonist ametlikus väljaandes (siin puudub rida ebaolulisi abimõisteid).

Sulgude kasutamine aritmeetilise avaldise definitsioonis on tingitud tehete järjekorra määramise vajadusest. Lisakokkulepetega tehete eelisjärjekorra kohta (ühes definitsiooni vastava täiendamisega) võib aga sulgude arvu tunduvalt vähendada. Määrame tehete eelisjärjekorra järgmiselt:

1. astendamine ($\uparrow$),

2. korrutamine ($\times$), jagamine (/) ja täisarvuline jagamine ($\div$),

3. liitmine (+) ja lahutamine (-).

Järjekorra naabertehete vahel määrab kõigepealt sulgude asetus, mille järel sooritatakse tehted eelisjärjekorda arvestades (sama järku tehete korral tuleb need sooritada loomulikus järjekorras vasakult paremale). Samaaegselt tuleb definitsiooni p. 3 täiendada selliselt:

3'. kui $\mathcal{A}$ ja $\mathcal{B}$ on aritmeetilised lihtavaldised, siis

$$\mathcal{A} \uparrow \mathcal{B} \mid \mathcal{A} \times \mathcal{B} \mid \mathcal{A} / \mathcal{B} \mid \mathcal{A} \div \mathcal{B} \mid \mathcal{A} + \mathcal{B} \mid \mathcal{A} - \mathcal{B}$$

on aritmeetilised lihtavaldised.

Otstarbekohane on aritmeetilise avaldise definitsiooni täiendada veel järgmise kokkuleppega: kui ($\mathcal{A}$) on aritmeetiline lihtavaldis, siis võib aritmeetilises avaldises kirjutise "else($\mathcal{A}$)" asendada kirjutisega "else $\mathcal{A}$" ; kui peale selle $\mathcal{A}$ pole tingimuslik aritmeetiline avaldis (s.t. pole kujul (1)), siis võib kirjutise "then($\mathcal{A}$)" asendada kirjutisega "then $\mathcal{A}$" .

Esitame nüüd mõningaid aritmeetiliste avaldiste näiteid:

$18.3_{10} + 4$ | a[i + 5] | exp(a + 0.5) |

5.7 × x ↑ 2 − 14.9 × y ↑ 3 + 8.6 × z ↑ 4 |

if x ≥ 0 then x else −x |

if a > 0 then 1 else if a < 0 then −1 else 0 .

Toodud näidetest kujutab neljas avaldis hulkliiget $5,7x^2 - 14,9y^3 + 8,6z^4$; 5. ja 6. avaldis on kujul (1), sisaldades loogilisi avaldisi $x \geq 0$, $a > 0$ ja $a < 0$.

Edasi selgitame aritmeetilise avaldise väärtuse arvutamist juhul, kui avaldis on kujul (1):

$$\underline{\text{if}}\ \mathscr{P}\ \underline{\text{then}}\ \mathscr{A}\ \underline{\text{else}}\ \mathscr{L} \quad .$$

Kõigepealt tuleb leida loogilise avaldise $\mathscr{P}$ tõeväärtus. Kui $\mathscr{P}$ on tõene, taandub avaldise (1) väärtuse arvutamine $\mathscr{A}$ väärtuse arvutamisele; kui $\mathscr{P}$ on väär, taandub avaldise (1) väärtuse arvutamine $\mathscr{L}$ väärtuse arvutamisele. Kui avaldistel $\mathscr{A}$ ja $\mathscr{L}$ on analoogiline struktuur nagu avaldisel (1), tuleb nende puhul rakendada sama põhimõtet.

Näide 1. Leiame aritmeetilise avaldise

$$\underbrace{\text{if}\ a > 0}_{\mathscr{P}}\ \underbrace{\text{then}\ 1}_{\mathscr{A}}\ \underbrace{\text{else}\ \overbrace{\text{if}}^{\mathscr{P}_1}\ \overbrace{a < 0}^{\ }\ \overbrace{\text{then}\ -1}^{\mathscr{A}_1}\ \overbrace{\text{else}\ 0}^{\mathscr{L}_1}}_{\mathscr{L}} \qquad (2)$$

väärtuse $a = 2$, $a = -3$ ja $a = 0$ korral.

Kui $a = 2$, siis tuleb $\mathscr{P}$ (ehk $a > 0$) tõesuse tõttu vaadelda avaldist $\mathscr{A}$, mis annab avaldise (2) väärtuseks 1.

Kui $a = -3$, siis tuleb $\mathscr{P}$ vääruse tõttu vaadelda

avaldist $\mathcal{L}$. Et $\mathcal{V}_1$ (ehk $a < 0$) on tõene, saame avaldise (2) väärtuseks -1.

Kui $a = 0$, siis $\mathcal{V}$ on väär. Avaldises $\mathcal{L}$ leiame kõigepealt, et $\mathcal{V}_1$ on väär ja seega saame avaldise (2) väärtuseks 0.

Muuhulgas tähendame, et avaldis (2) on samaväärne funktsiooniga sign(a).

Näide 2. Leiame aritmeetilise avaldise

$$\text{if } x > 3 \text{ then } \overbrace{(\text{if } x \geq 100 \text{ then } \text{sqrt}(x) \text{ else } 1/x)}^{\mathcal{A}} \quad \underbrace{\text{else if } x < (-5) \text{ then } \text{abs}(x) \text{ else } x}_{\mathcal{L}} \tag{3}$$

väärtuse $x = -2$, $x = 125$, $x = 20$ ja $x = -7$ korral.

Kui $x = -2$, siis leitakse avaldise (3) väärtus avaldisest $\mathcal{L}$, kust $x < (-5)$ vääruse tõttu saame x ehk konkreetselt -2.

Kui $x = 125$, siis leitakse avaldise (3) väärtus avaldisest $\mathcal{A}$, kust $x \geq 100$ tõesuse tõttu saame sqrt(x) ehk sqrt(125).

Kui $x = 20$, siis leiame avaldisest $\mathcal{A}$, et avaldise (3) väärtus on 1/20.

Kui $x = -7$, siis leiame avaldisest $\mathcal{L}$, et avaldise (3) väärtus on abs(-7) = 7.

Lõpuks peatume veel küsimusel, kuidas avaldub aritmee-

tiliste tehete tulemuse tüüp tehtest osavõtvate suuruste tüüpide kaudu.

1. Korrutis, summa ja vahe on tüüpi integer siis ja ainult siis, kui mõlemad tehtest osavõtvad suurused on tüüpi integer .

2. Jagatis pole defineeritud juhul, kui jagaja võrdub nulliga. Kui jagatis on defineeritud, on ta alati tüüpi real.

3. Täisarvuline jagatis (÷) defineeritakse ainult juhul, kui mõlemad tehtest osavõtvad suurused on täisarvud ja jagaja on nullist erinev. Tehte tulemus defineeritakse järgmiselt

$$m \div n = sign(m/n) \times entier(abs(m/n))$$

ja on alati tüüpi integer .

4. Aste pole defineeritud kolmel juhul: null astmel null korral (kus null on kas tüüpi integer või real), kui null on negatiivsel astmel ja kui negatiivne arv on mittetäisarvulisel astmel.

Täisarv mittenegatiivsel täisarvulisel astmel annab astme tüübi integer, kõik teised juhud, mil aste on defineeritud, annavad astme tüübiks real.

Näiteid aritmeetiliste tehete kohta (vastuse tüüp nähtub ka vastuse kujust):

2 × 4 = 8 (tüüp integer),

1.5 × 6 = 9.0 (tüüp real),

5 + 12 = 17 (tüüp integer).

12 - 4.0 = 8.0 (tüüp real),

12/3 = 4.0 (tüüp real),

$10/0_{13}3$ pole defineeritud,

24 ÷ 5 = 4 (tüüp integer),

24 ÷ (-5) = -4 (tüüp integer),

18 ÷ 3.5 pole defineeritud,

5 ↑ 3 = 125 (tüüp integer),

2 ↑ 0 = 1 (tüüp integer),

2.0 ↑ 0 = 1.0 (tüüp real),

$3.5 \uparrow (-2) = \left\langle \frac{1}{3,5^2} \right\rangle$ (tüüp real),

$0_{10}2 \uparrow (-2)$ pole defineeritud,

4.5 ↑ (-3.5) arvutatakse valemi

a ↑ r = exp(r × ln(a)) põhjal (tüüp real),

$0_{10}3 \uparrow 3.5 = 0.0$ (tüüp real),

0.0 ↑ (-2.2) pole defineeritud,

(-6) ↑ 2.0 pole defineeritud.

§ 7. Loogilised avaldised

Loogilise avaldise mõiste keeles ALGOL on laiem loogilise valemi mõistest lausearvutuses. See sisaldab endas niisuguseid konstruktsioone nagu aritmeetiline avaldis ja loogiline funktsioon ning on sisuliselt samaväärne arvutatava (rekursiivselt loendatava) predikaadi mõistega.

Defineerime loogilise avaldise järgmiselt [7]:

1. Tõeväärtused <u>true</u> ja <u>false</u>, muutujad tüüpi <u>Boolean</u> ja funktsioonid tüüpi <u>Boolean</u> on loogilised lihtavaldised;

2. kui $\mathcal{A}$ ja $\mathcal{B}$ on aritmeetilised lihtavaldised, siis

$$(\mathcal{A} = \mathcal{B}) \mid (\mathcal{A} \geq \mathcal{B}) \mid (\mathcal{A} \leq \mathcal{B}) \mid (\mathcal{A} \neq \mathcal{B}) \mid (\mathcal{A} < \mathcal{B}) \mid (\mathcal{A} > \mathcal{B})$$

on loogilised lihtavaldised;

3. kui $\mathcal{L}$ on loogiline lihtavaldis, siis $\neg\mathcal{L}$ on loogiline lihtavaldis;

4. kui $\mathcal{L}$ ja $\mathcal{M}$ on loogilised lihtavaldised, siis

$$(\mathcal{L} \wedge \mathcal{M}) \mid (\mathcal{L} \vee \mathcal{M}) \mid (\mathcal{L} \supset \mathcal{M}) \mid (\mathcal{L} \equiv \mathcal{M})$$

on loogilised lihtavaldised;

5. kõik loogilised lihtavaldised on loogilised avaldised;

6. kui $\mathcal{L}$ on loogiline lihtavaldis ja $\mathcal{E}$ ning $\mathcal{F}$ on loogilised avaldised, siis

$$\underline{\text{if}}\ \mathcal{E}\ \underline{\text{then}}\ \mathcal{L}\ \underline{\text{else}}\ \mathcal{F} \qquad (1)$$

on loogiline avaldis;

7. kui $\mathcal{E}$ on loogiline avaldis, siis $(\mathcal{E})$ on loogiline lihtavaldis.

---

[7] Ka selles definitsioonis ei kasuta me mitmeid ametlikus väljaandes esinevaid ebaolulisi mõisteid.

Nii nagu aritmeetilise avaldise korral määrame ka siin sulgude arvu vähendamiseks tehete eelisjärjekorra. Lepime kokku, et kõigepealt tuleb sooritada aritmeetilised tehted, kusjuures nende tehete vaheline eelisjärjekord (punktid 1. - 3., § 6) jääb siin kehtima. Seejärel tulevad:

4. võrdlustehted $= \mid \geq \mid \leq \mid \neq \mid < \mid >$ (kõik sama järku tehted);

5. eitus ($\neg$);

6. konjunktsioon ($\wedge$);

7. disjunktsioon ($\vee$);

8. implikatsioon ($\supset$);

9. ekvivalents ($\equiv$).

Samaaegselt tuleb definitsiooni punkte 2 ja 4 täiendada nii, et võrdlustehteid ja binaarseid loogilisi tehteid võib kirjutada ka ilma ümarsulgudeta.

Esitame mõningaid loogiliste avaldiste näiteid:

$\underline{\text{true}} \mid A[i] \mid (A \supset B) \equiv \neg(C \vee \neg D) \mid$

$(a + b > 0) \wedge (\neg X \equiv Y[k]) \mid G(A \vee \neg B, C, d + 5) \mid$

$\underline{\text{if}}\ A \equiv B\ \underline{\text{then}}\ F(A) \supset \underline{\text{false}}\ \underline{\text{else}}\ x + y \leq z \wedge \neg A$

Nendes näidetes eeldame, et väikesed ladina tähed tähistavad arvulisi muutujaid, suured ladina tähed aga loogilisi muutujaid või loogilisi funktsioone.

Vaatleme nüüd, kuidas leitakse loogiliste avaldiste väärtusi. Kõigepealt esitame loogiliste tehete definitsioonid

A	B	$A \wedge B$	$A \vee B$	$A \supset B$	$A \equiv B$
false	false	false	false	true	true
false	true	false	true	true	false
true	false	false	true	false	false
true	true	true	true	true	true

A	$\neg A$
false	true
true	false

Näide 1. Loogilise avaldise

$$(A \supset B) \equiv \neg(C \vee \neg D)$$

tõeväärtus on A = D = true ja B = C = false korral false, A = true ja B = C = D = false korral aga true.

Tingimusliku loogilise avaldise (1)

$$\text{if } \mathcal{C} \text{ then } \mathcal{L} \text{ else } \mathcal{F}$$

väärtuste arvutamine toimub põhimõtteliselt samuti nagu tingimusliku aritmeetilise avaldise puhul. Kõigepealt tuleb leida loogilise avaldise $\mathcal{C}$ tõeväärtus. Kui $\mathcal{C}$ on tõene, leitakse avaldise (1) tõeväärtus avaldisest $\mathcal{L}$, vastupidisel juhul aga avaldisest $\mathcal{F}$. Keerulisematel juhtudel tuleb ka siin eeskirja korduvalt rakendada.

Näide 2. Arvutame loogilise avaldise

$$\text{if } \underbrace{x > 0}_{\mathcal{C}} \text{ then } \underbrace{A \supset B}_{\mathcal{L}} \text{ else } \underbrace{A \vee \neg B}_{\mathcal{F}} \qquad (2)$$

tõeväärtuse juhtudel

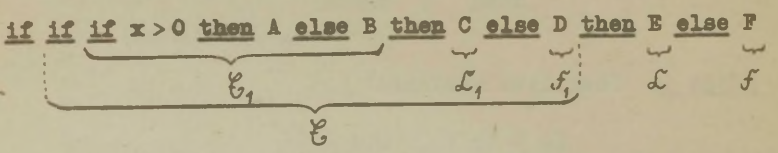
1) x = 3.5, A = true, B = false;

2) x = -2, A = false, B = false.

Juhul 1) on $\mathcal{E}$ tõene ja avaldisest $\mathcal{L}$ (ehk A ⊃ B) saame avaldise (2) tõeväärtuseks false.

Juhul 2) on $\mathcal{E}$ väär ja avaldis $\mathcal{F}$ (ehk A ∨ ¬B) annab avaldise (2) tõeväärtuseks true.

Näide 3. Loogilise avaldise



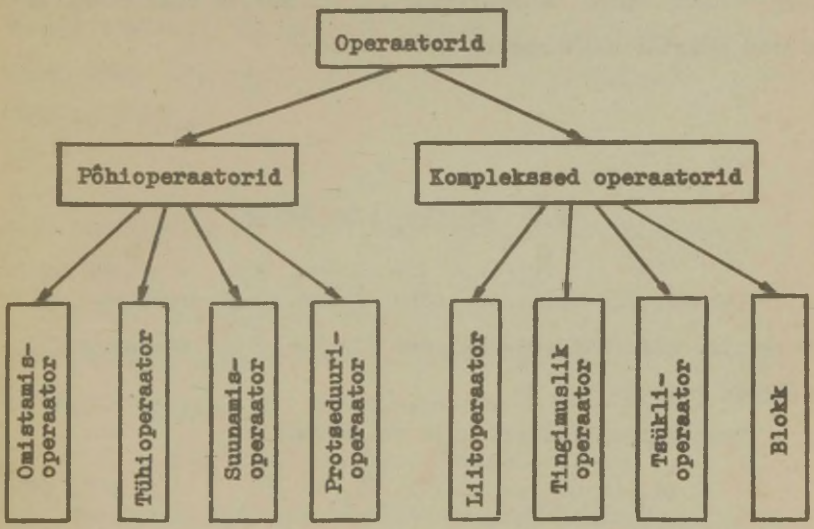
tõeväärtus on näit. x = 4, A = D = F = true, B = C = E = = false korral true; kui aga x = -5 (muutujate A, ..., E väärtused samad), siis false.

§ 8. Operaatorite liigitus

Arvud, identifikaatorid, muutujad, funktsioonid ja avaldised on keele ALGOL konstruktsioonid, mis võivad esineda ainult operaatorite koosseisus. Operaatorid ise on keele ALGOL suhteliselt iseseisvad üksused. ALGOL-programm kujutabki endast operaatorite lõplikku jada, millele lisanduvad mitmesugused kirjeldused. Programmi võib liigendada ka blokkide ta-

semel, kuid blokk ise kuulub samuti operaatorite hulka.

Kuigi blokil on teiste operaatoritega võrreldes mõningaid olulisi erinevusi (suurem iseseisvus võrreldes teiste operaatoritega), liigitatakse operaatorid kahte rühma siiski komplekssuse seisukohast. Nimelt liigitatakse operaatorid selle järele, kas nende koosseisus võivad esineda teised operaatorid või mitte. Esimesi operaatoreid nimetatakse komplekssseteks operaatoriteks, teisi põhioperaatoriteks. Operaatorite üksikasjalik liigitus on toodud joonisel 1.



Joonis 1.

Programmide koostamiseks piisab tegelikult juba põhioperaatoritest ja blokist. Liitoperaatori, tingimusliku operaatori ja tsüklioperaatori kasutamisel võib aga programme lühendada ja muuta neid ülevaatlikumaks.

Tähendame, et kõiki operaatoreid võib märgistada (varustada märgistega), mis lubab reeglina suunata otseselt nende täitmisele (vt. § 11). Märgiste olemasolu seisukohalt võib iga operaatorite liiki või rühma jaotada veel kaheks, millega saame mõisted, nagu märgistatud ja märgistamata operaator, märgistatud ja märgistamata põhioperaator, märgistatud ja märgistamata omistamisoperaator jne. Et operaatori töö märgistamisest ei sõltu ning märgistamine ise mingeid probleeme ei valmista, siis käsitleme üksikuid operaatoreid märgistamata kujul. Järgnevates paragrahvides vaatlemegi üksikuid märgistamata operaatorite liike.

§ 9. Omistamisoperaator

Omistamisoperaator on ette nähtud mingi arvulise või loogilise väärtuse omistamiseks kas ühele või samaaegselt mitmele muutujale.

Omistamisoperaatori kuju on järgmine:

$$\mathcal{A}_1 := \mathcal{A}_2 := \langle \ldots \rangle \; \mathcal{A}_n := \mathcal{L} \qquad , n \geq 1, \quad (1)$$

kus $\mathcal{L}$ on kas aritmeetiline või loogiline avaldis ja metalingvistilised sümbolid $\mathcal{A}_1, \mathcal{A}_2, \ldots, \mathcal{A}_n$ tähistavad muutujaid. Avaldise $\mathcal{L}$ liik ja muutujate $\mathcal{A}_1, \mathcal{A}_2, \ldots$ $\ldots, \mathcal{A}_n$ tüüp peavad olema kooskõlas. Kui $\mathcal{L}$ on loogiline avaldis, peavad muutujad $\mathcal{A}_1, \mathcal{A}_2, \ldots, \mathcal{A}_n$ olema tüüpi Boolean, aritmeetilise avaldise korral võivad muutujad

$\alpha_1, \alpha_2, \ldots, \alpha_n$ olla tüüpi integer kui ka tüüpi real.

Omistamisoperaatorite näiteid:

$$a := 5.7 \mid x[i/2 + 1] := i := m + 3.4 \mid$$

$$A[i] := B[i + j] := C := D[i] \supset C \quad .$$

Omistamisoperaatori puhul tuleb tähelepanu pöörata veel järgmistele asjaoludele.

1. Kui muutuja tüüp ei ühti omistatava arvu tüübiga või üldisemalt kõneldes - aritmeetilisest avaldisest $\mathcal{L}$ saadava arvu tüübiga, siis muutujale omistatav arv teisendatakse automaatselt. Täisarvu teisendamisel reaalarvuks probleeme ei teki. Reaalarvulise väärtuse $\mathcal{L}$ teisendamisel täisarvuks saadakse

$$\text{entier}(\mathcal{L} + 0.5),$$

s.t. reaalarvule $\mathcal{L}$ lähim täisarvuline väärtus, kusjuures .5-ga lõppevad arvud ümardatakse ülespoole.

Seega, kui esimeses toodud näidetest a on reaalarvuline muutuja, siis omistatakse talle väärtus 5.7. Kui aga a on tüüpi integer, siis omistatakse talle väärtus

$$\text{entier}(5.7 + 0.5) = \text{entier}(6.2) = 6 \quad .$$

Meenutame, et sama teisendusreeglit kasutatakse ka indeksavaldiste korral (indeksavaldiseks võib olla mistahes aritmeetiline avaldis, indeksi konkreetseks väärtuseks peab aga olema täisarv).

2. Omistamisoperaatoris on oluline veel arvutuste järjekord. Kui see poleks täpselt kindlaks määratud, siis võiks

aritmeetiliste avaldiste väärtuste omistamisel saada erinevaid tulemusi. Tehete järjekord määratakse järgmiselt:

a) kõigepealt arvutatakse omistamisoperaatori (1) kõikide muutujate $\mathcal{A}_1, \mathcal{A}_2, \ldots, \mathcal{A}_n$ indeksavaldised, kasutades vajaduse korral ülalesitatud teisenduseeskirja;

b) seejärel arvutatakse omistamisoperaatori (1) paremal pool seisva avaldise $\mathcal{L}$ väärtus;

c) edasi omistatakse muutujatele $\mathcal{A}_1, \mathcal{A}_2, \ldots, \mathcal{A}_n$ (kus indeksid on juba leitud) avaldise $\mathcal{L}$ väärtus, rakendades vajaduse korral ülalesitatud teisenduseeskirja.

Näide. Selgitame, missugused väärtused ja millistele muutujatele omistatakse operaatori

$$x[i/2 + 1] := i := m + 3.4$$

töö tulemusena, kui m ja i algväärtused on m = 10 ja i = 6 (ning i on tüüpi integer ja x[j] tüüpi real.

Arvutamine toimub järgmiste sammudega:

1. indeksavaldise i/2 + 1 väärtus tuleb 4;
2. aritmeetilise avaldise $\mathcal{L}$ väärtus tuleb 13.4;
3. arvutades muutujate tüüpe toimub omistamine

$$x[4] := 13.4 \quad \text{ja} \quad i := 13 \quad .$$

Kui oleksime arvutanud teises järjekorras, näiteks kõigepealt $\mathcal{L}$ (13.4), seejärel i (13) ja alles seejärel indeksavaldise (8), oleksime saanud hoopis omistamise x[8] := 13.4. Toodud näidetest ilmneb arvutuste järjekorra täpse fikseerimise vajalikkus omistamisoperaatori puhul.

§ 10. Tühioperaator

Tühioperaatorit võib defineerida süntaktiliselt (nii nagu defineerisime arvu, identifikaatori, muutuja ja teised ALGOLi konstruktsioonid), kuid seda võib mõista ka laiemas - semantilises mõttes.

Tühioperaator on süntaktiliselt defineeritav kui tühisõna. Arvestades seda, et vahetult järgnevad operaatorid eraldatakse teineteisest semikooloni abil, võib tühioperaatorit esitada näiteks programmilõigu

```
x := a ; ; y := b ;
```

abil, kus tühioperaator on kahe omistamisoperaatori vahel.

Tühioperaatori tööd on võimalik kirjeldada ainult eitavate väljendite abil. Oluline on märkida, et tühioperaator 1) ei muuda programmis ühegi muutuja väärtust, 2) ei muuda programmi operaatorite täitmise loomulikku järjekorda, 3) ei väljasta informatsiooni. Neid kolme omadust aluseks võttes võib tühioperaatori defineerida semantilises mõttes kui operaatori, mis rahuldab tingimusi 1) - 3). Semantilise tühioperaatori näiteks võib olla operaator

```
x := x
```

või suunamisoperaator (vt. § 11), mis suunab vahetult järgneva operaatori juurde, nagu juhul

```
go to Q ; Q : y := 5 ;
```

Et tühioperaator mingit otsest ülesannet programmis ei

täida, võib tekkida mulje, et see operaatoriliik pole üldse vajalik. Üldiselt kõneldes pole tõepoolest vajadust tühioperaatoreid programmi koosseisu lülitada. Kuid mõnel puhul on tühioperaatori kasutamisel siiski ka oma mõte, näiteks juhul, kus teda kasutatakse kui sobivat operaatorit märgistamiseks ja suunamiseks. Mitmetel juhtudel on tühioperaatorite kasutamisel hõlpsamini võimalik parandada vigu perforeeritud programmis. Teoreetilises käsitluses on tühioperaatori mõistel aga kindel koht, sest selle abil on sageli võimalik lihtsamini selgitada teiste operaatorite tööd.

§ 11. Suunamisoperaator ja märgiseavaldis

Operaatorid täidetakse programmis loomulikus järjekorras: pärast mingi operaatori tööd täidetakse temast vahetult paremal asuv operaator; kui samas reas temast paremal pool operaatorit ei leidu, tuleb edasi minna järgmise rea vasakult esimese operaatori täitmisele. Sageli on aga operaatorite täitmise loomulikku järjekorda vaja muuta, milleks keeles ALGOL leidub spetsiaalne konstruktsioon - suunamisoperaator.

Suunamisoperaatorit võib esitada kujul

go to $\mathfrak{M}$,

kus $\mathfrak{M}$ on märgiseavaldis. Seega taandub suunamisoperaatori konstruktsioon märgiseavaldise konstrueerimisele.

Märgiseavaldise defineerime järgmiselt:

1. Lihtsaks märgiseavaldiseks on märgis ehk märgend (süntaksi järgi võib märgiseks olla identifikaator või märgita täisarv) ja lüliti (süntaktiliselt - ühe indeksiga muutuja);

2. iga lihtne märgiseavaldis on märgiseavaldis;

3. kui $\mathcal{L}$ on loogiline avaldis, $\mathcal{L}$ - lihtne märgiseavaldis ja $\mathcal{M}$ - märgiseavaldis, siis

$$\underline{\text{if}}\ \mathcal{L}\ \underline{\text{then}}\ \mathcal{L}\ \underline{\text{else}}\ \mathcal{M} \qquad (1)$$

on märgiseavaldis;

4. kui $\mathcal{M}$ on märgiseavaldis, siis $(\mathcal{M})$ on lihtne märgiseavaldis.

Suunamisoperaatorite näiteid:

$$\underline{\text{go to}}\ Q \mid \underline{\text{go to}}\ 29 \mid \underline{\text{go to}}\ S[i + j] \mid$$
$$\underline{\text{go to}}\ \underline{\text{if}}\ x \geq 0\ \underline{\text{then}}\ 5\ \underline{\text{else}}\ R[2 \times n + 1] \quad .$$

Suunamisoperaatori töö kirjeldamisel tuleb kõigepealt tähendada, et märgiseavaldis määrab põhimõtteliselt teatava märgise, mille järgi toimubki suunamine. Suunamiseks peab vastav märgis esinema ka programmi vajalikus kohas (operaatori ees, mille juurde suunatakse) [8]. Märgise leidmiseks tuleb märgiseavaldist analüüsida samadel põhimõtetel nagu aritmeetilist või loogilist avaldist. Analüüsi tulemusena saame

---

[8] Siinkohal mainime, et väljastpoolt blokki ja tsüklioperaatorit pole võimalik suunata vastavate operaatorite alamoperaatorite juurde (vt. §§ 12, 14 ja 15).

mõnel juhul kohe märgise (millega analüüs lõpeb), mõnel juhul aga lüliti. Et leida märgist lüliti korral, on vaja täiendavat informatsiooni. Seda saame vastava lüliti kirjeldusest, mis peab asuma vaadeldava programmi (bloki) kirjelduste osas.

Lüliti kirjeldus defineeritakse järgmiselt:

$$\underline{\text{switch}}\ \mathcal{P} := \mathcal{M}_1, \mathcal{M}_2, \langle\cdots\rangle, \mathcal{M}_k \qquad ,k \geq 1,$$

kus $\mathcal{P}$ on vastava lüliti identifikaator ja $\mathcal{M}_1, \mathcal{M}_2, \ldots$ $\ldots, \mathcal{M}_k$ märgiseavaldised. Omistamismärgist paremal pool seisvat osa lülitikirjelduses nimetatakse lülituste loendiks.

Märgise määramine lüliti korral toimub järgmiselt:

1. Kõigepealt arvutatakse lüliti indeksavaldise väärtus (samuti nagu indeksiga muutuja puhul).

2. Seejärel otsitakse vastava lüliti kirjelduses lülituste loendist märgiseavaldis, mille järjekorranumber ühtib indeksi selle väärtusega. Numeratsiooni alustame kõige vasakpoolsemast avaldisest (järjekorranumbriga 1).

3. Kui leitud märgiseavaldis osutub märgiseks, on märgise määramise protsess sellega lõppenud; kui aga saame keerulisema märgiseavaldise, siis tuleb menetlust jätkata samade põhimõtete alusel.

4. Kui indeksavaldise väärtus on mittepositiivne või suurem märgiseavaldiste arvust k lülituste loendis, siis pole märgis määratud ja suunamisoperaator töötab sel juhul kui tühioperaator.

Näide. Olgu antud suunamisoperaator

go to S[i + j] (2)

ja lüliti S kirjeldus

switch S := 12, P3, if x > 0 then Tk else U, K[j] ;

Kui suunamisoperaatori (2) täitmise ajal i = j = 0, siis on indeksavaldise i + j väärtus 0 ja märgis pole määratud; kui näit. i = 1 ja j = 0, tuleb i + j = 1 ja lülituste loendist tuleb võtta esimene märgiseavaldis - märgis 12; kui näit. i = 0 ja j = 2, saame märgise P3; kui näit. i = 2 ja j = 1, saame kolmanda märgiseavaldise, mis olenevalt muutuja x väärtusest annab kas märgise Tk või U; kui näit. i = j = 2 saame neljanda märgiseavaldise - lüliti K[j], millest konkreetse märgise määramine nõuab lüliti K kirjelduse kasutamist. Niisiis töötab suunamisoperaator (2) vaadeldud juhtudel nagu tühioperaator või suunamisoperaator

go to 12 | go to P3 | go to Tk või go to U | go to K[j] .

Et tähistada koht programmis, kuhu on vaja suunata (juhtimine edasi anda), tuleb vastavad märgised 12, P3, Tk, U jne. paigutada vajalike operaatorite ette. Märgiste asetamist operaatorite ette nimetatakse operaatorite märgistamiseks, vastavat operaatorit nimetatakse sel juhul märgistatud operaatoriks. Operaator (mis võib olla ka märgistatud) ja vastav märgis eraldatakse teineteisest kooloniga. Samas blokis ei tohi sama märgisega märgistada rohkem kui üht operaatorit (muidu pole suunamine üheselt määratud). Küll võib

aga üht operaatorit märgistada mitme erineva märgisega. Märgistamine ei muuda operaatori liiki ega funktsioneerimist.

Märgistatud operaatorite näiteid:

M : x := 5;

L : Tk : A[t] := B := C := D[n] ⊃ E;

Teine esitatud operaatoritest on märgistatud kahe märgisega L ja Tk.

§ 12. Liitoperaator ja blokk

Liitoperaator moodustatakse lõplikust operaatorite jadast selle asetamisel operaatorsulgudesse begin ja end. Liitoperaatori kuju on seega järgmine:

begin $\mathscr{L}_1$; $\mathscr{L}_2$; ⟨···⟩ ; $\mathscr{L}_n$ end ,

kus $\mathscr{L}_1$, $\mathscr{L}_2$, ..., $\mathscr{L}_n$ (n ≥ 1) tähistavad operaatoreid. Liitoperaatori alamoperaatorid võivad olla mistahes liiki (kaasaarvatud liitoperaator ja blokk).

Liitoperaatori näiteid:

begin end | begin x := 27.3 end | begin y := a[i]; end |
begin x := i; go to if z ≥ 0 then M else N; M : u := x+j end

Esimeses näites on liitoperaatori ainukeseks alamoperaatoriks tühioperaator; kolmandas näites on kaks alamoperaatorit, teine neist tühioperaator.

Liitoperaatori moodustamine ei muuda alamoperaatorite eneste täitmist ega nende täitmise järjekorda liitoperaatori raames. Nimetatud konstruktsiooni mõte seisneb selles, et mitmest operaatorist moodustada üht või siis tingimuslikust operaatorist saada mittetingimuslikku operaatorit. Vajadus niisuguste konstruktsioonide järele tekib näiteks tingimusliku operaatori (vt. § 13) ja tsüklioperaatori (vt. § 14) kasutamisel.

Liitoperaatorile sarnaneb veel teine konstruktsioon - blokk. Süntaktiliselt erineb blokk liitoperaatorist ainult selle poolest, et blokk algab kirjeldustega, millele järgnevad operaatorid. Bloki struktuur on seega järgmine:

begin $\mathcal{D}_1$; $\mathcal{D}_2$; $\langle\ldots\rangle$; $\mathcal{D}_m$; $\mathcal{L}_1$; $\mathcal{L}_2$; $\langle\ldots\rangle$; $\mathcal{L}_n$ end ,

kus $\mathcal{D}_1$; $\mathcal{D}_2$, ..., $\mathcal{D}_m$ ($m \geq 1$) tähistavad kirjeldusi ja $\mathcal{L}_1$, $\mathcal{L}_2$, ..., $\mathcal{L}_n$ ($n \geq 1$) operaatoreid.

Näiteid blokkide kohta saame kergesti konstrueerida suvalisest liitoperaatorist, lisades viimasele mingi kirjelduse. Nii näiteks, silmas pidades, et real x on muutuja x kirjeldus (vt. § 15) saame sellised blokid

begin real x; end | begin real x; x := a end

Kui üks blokk on teise koostisosa, kõneldakse vastavalt hõlmatavast ja hõlmavast blokist, samuti kasutatakse terminit alamblokk.

Kuigi bloki ja liitoperaatori puht-graafiline erinevus ei ole suur, on sisulised (semantilised) erinevused nende kahe konstruktsiooni vahel olulised.

Blokk on programmi terviklik ja suhteliselt iseseisev osa. Blokis võib kasutusele võtta omaette tähised sõltumata programmi teistes osades kasutatavatest tähistest. See saavutatakse muutujate, lülitite ja protseduuride kirjeldamisega, millega need n.ö. lokaliseeritakse antud bloki raamides (seda küsimust vaatleme üksikasjalikumalt paragrahvis 15). Liitoperaatori puhul võib näiteks väljastpoolt liitoperaatorit suunata ükskõik missuguse selle alamoperaatori juurde. Bloki puhul on see aga võimatu: seal algab töö ikka esimesest operaatorist - blokk töötab kui kindel terviklik konstruktsioon.

Siinkohal võiks peatuda ka programmi mõistel ja selle tegelikul kirjapanekul keeles ALGOL. Sisuliselt tähendab programm mingi ülesande lahendamise algoritmi, mis on esitatud keeles ALGOL, kujutades endast selle keele teksti. Keelekonstruktsioonide seisukohalt võib programmiks olla kas blokk või liitoperaator, mis pole ühegi operaatori alamoperaatoriks ega kasuta ühtegi teist temast väljaspool asuvat operaatorit.

ALGOL-programmide praktilises kirjapanekus, aga samuti parema ülevaatlikkuse huvides on lepitud kokku järgnevas:

1) ALGOLi teksti on võimalik viia üle uuele reale mistahes põhisümbolist alates (ilma mingi poolitusmärgita). Üleviimisel ei tohi sümboleid korrata. Rea algusesse ja lõppu jäetav vaba ruum ei mõjusta programmi täitmist. See kokkulepe lubab programmi esitada ülevaatlikumalt (näit. alustada programmi eri etappe uuelt realt) ning on aluseks ka kokku-

leppele operaatorsulgude kirjutamise kohta.

2) Kui liitoperaator või blokk on küllalt lühike, siis püütakse teineteisele vastavad operaatorsulud kirjutada ühele reale (eespool toodud näidetes on seda kokkulepet arvestatud).

3) Kui liitoperaator või blokk on tekstireast pikem, siis on otstarbekohane teineteisele vastavad operaatorsulud kirjutada üksteise alla (ühele joonele või ühte veergu). Kogu vastava liitoperaatori või bloki tekst kirjutatakse nendest sümbolitest paremale poole.

Mitmekordsete operaatorsulgude puhul võib kokkuleppeid 2) ja 3) rakendada korduvalt. Tähendame, et kokkulepped 2) ja 3) pole süntaksireeglid ja pole järelikult kohustuslikud. Vastavaid operaatorsulge võib soovi korral ära märkida ka teisiti - märgise ja kommentaari (vt. § 17) abil.

Näiteid:

```
if i ≤ n then
begin if a[i] > 0 then
    j := j + 1
end else
begin print(j); b`:=
    a[j]↑2; go to M
end
```

```
A : begin x := i;
      B : begin go to if z ≥ 0
              then M else N
          end B;
          M : u := x + j
    end A;
```

Nendes näidetes on arvestatud kokkulepet 3), kusjuures teineteisele vastavate operaatorsulgude korral on kasutatud

(meta)punktiirjoont. Teises näites on operaatorsulgude vastavuse tähistamiseks kasutatud ka objektkeele konstruktsioone - märgist ja kommentaari (A ja B).

§ 13. Tingimuslik operaator

Konstruktsioon "tingimuslik operaator" on ette nähtud selleks, et võimaldada programmi täitmisel operaatoreid vahele jätta, vastavalt teatud tingimuste kehtimisele või mittekehtimisele. Kasutades tingimusliku operaatori alamoperaatoritena mitmesugust liiki operaatoreid, võib sel viisil tingimuslikult sooritada omistamisi ja suunamisi, välja jätta programmiosi ja blokke. Tingimusliku operaatori paindliku struktuuri tõttu saab seda kõike teha sageli palju lihtsamini kui teiste konstruktsioonide abil.

Tingimuslikul operaatoril on kaks erikuju:

$$\underline{\text{if}}\ \mathscr{L}\ \underline{\text{then}}\ \mathscr{A}_1 \qquad (1)$$

ja

$$\underline{\text{if}}\ \mathscr{L}\ \underline{\text{then}}\ \mathscr{A}_2\ \underline{\text{else}}\ \mathscr{L} \qquad ,(2)$$

kus $\mathscr{L}$ tähendab loogilist avaldist, $\mathscr{A}_1$ - mittetingimuslikku operaatorit, $\mathscr{A}_2$ - põhioperaatorit, liitoperaatorit või blokki ja $\mathscr{L}$ - mistahes operaatorit [9].

---

[9] Juhime tähelepanu sellele, et tähe "else" ette ei või kunagi asetada semikoolonit (vt. teine ja kolmas näide).

Tingimusliku operaatori näitеid:

```
if i ≤ n then go to Q |
if x[j] ≥ 0 then m := m + 1 else n := n + 1 |
if z > x[i] then begin i := i + 1; go to P end else if i ≠ 0
then z := x[i]                                              .
```

Tingimusliku operaatori töö toimub järgmiselt. Nii kuju (1) kui ka (2) korral kontrollitakse kõigepealt loogilise avaldise $\mathcal{L}$ kehtivust. Kuju (1) puhul täidetakse $\mathcal{L}$ kehtivuse korral operaator $\mathcal{A}_1$, vastupidisel juhul töötab operaator kui tühioperaator. Kuju (2) puhul täidetakse $\mathcal{L}$ kehtivuse korral operaator $\mathcal{A}_2$, vastupidisel juhul aga operaator $\mathcal{L}$. Kui tingimuslik operaator on keerulisema struktuuriga, tuleb nimetatud eeskirja kasutada ka tema alamoperaatorite puhul.

Pärast alamoperaatorite $\mathcal{A}_1$, $\mathcal{A}_2$ või $\mathcal{L}$ täitmist (eeldusel, et need ei andnud suunamist mõnele teisele operaatorile) täidetakse tingimuslikule operaatorile (1) või vastavalt (2) vahetult järgnev operaator.

Tingimusliku operaatori alamoperaatoreid on võimalik märgistada ja nende juurde suunata nii samast operaatorist kui ka väljastpoolt. Sel juhul toimub tingimusliku operaatori töö samuti nagu vastavate tingimuste kehtimisel, mispuhul see alamoperaator oleks töösse lülitunud. Nii näiteks programmilõigus

$$\underline{\text{if}}\ \mathcal{L}\ \underline{\text{then}}\ \text{M} : \mathcal{A}\ \underline{\text{else}}\ \text{N} : \mathcal{L}\ ;\ \mathcal{P}$$

töötavad go to M korral operaatorid

$$\mathcal{A}\ ;\ \mathcal{P}$$

(eeldades, et $\mathcal{A}$ ei sisalda suunamisoperaatorit), ja go to N korral operaatorid

$$\mathcal{L}\ ;\mathcal{N}$$

(eeldades, et $\mathcal{L}$ ei sisalda suunamisoperaatorit).

§ 14. Tsüklioperaator

Kuigi keeles ALGOL on mitmeid võimalusi tsüklite programmeerimiseks, on selleks otstarbeks siiski ette nähtud ka spetsiaalne konstruktsioon - tsüklioperaator. Tsüklioperaatori üldkuju on

$$\underline{\text{for}}\ \mathcal{A} := \mathcal{B}_1, \mathcal{B}_2, \langle\cdots\rangle\ , \mathcal{B}_n\ \underline{\text{do}}\ \mathcal{L} \qquad ,n \geq 1,$$

kus $\mathcal{A}$ (tsükli parameeter) võib olla mistahes muutuja tüüpi integer või real; $\mathcal{L}$ võib olla mistahes operaator; loendit "$\mathcal{B}_1, \mathcal{B}_2, \langle\cdots\rangle\ , \mathcal{B}_n$" nimetatakse tsükli loendiks, selle liikmeid $\mathcal{B}_i$ $(1 \leq i \leq n)$ - tsükli loendi elementideks.

Tsükli loendi elemente on kolme liiki:

1) $\mathcal{N}$ - aritmeetiline avaldis;

2) $\mathcal{N}_1$ step $\mathcal{N}_2$ until $\mathcal{N}_3$ (samm-element ehk "step"-element), kus $\mathcal{N}_1$, $\mathcal{N}_2$ ja $\mathcal{N}_3$ on aritmeetilised avaldised;

3) $\mathcal{N}$ while $\mathcal{E}$ ("while"-element ehk seni-kui-element), kus $\mathcal{N}$ on aritmeetiline ja $\mathcal{E}$ loogiline avaldis.

Eri liiki elementide järjekorra kohta loendis kitsendusi ei ole.

Tsüklioperaator võib näiteks olla järgmine:

for i := 1, i + 3, 5 step 2 until 15, 20, i - 1
while i ≥ 16 do S := S + a[i] .

Selle tsüklioperaatori loendis on viis elementi: esimene, teine ja neljas element on aritmeetilised avaldised, kolmas on samm-element ning viies "while"-element.

Tsüklioperaator töötab järgmiselt. Tsükli loendi elemendid annavad tsükli parameetrile väärtusi, kusjuures parameetri iga sellise väärtuse korral täidetakse operaator $\mathcal{L}$. Elemendid annavad väärtusi loendis toodud järjekorras (alustades esimesest) - kui üks element on ammendatud, minnakse üle järgmisele elemendile, kuni kogu loend on ammendatud [10].

Üksikud elemendid töötavad selliselt:

1. Aritmeetiline avaldis $\mathcal{V}$ annab tsükli parameetrile ainult ühe väärtuse - avaldise $\mathcal{V}$ väärtuse. Seega töötab see element "ühekordselt", nii nagu programmilõik:

$\mathcal{O}$:= $\mathcal{V}$; $\mathcal{L}$; go to ⟨ uus element ⟩ ;

2. Samm-element töötab nagu programmilõik:

$\mathcal{O}$:= $\mathcal{V}_1$; L : if ($\mathcal{O} - \mathcal{V}_3$) × sign($\mathcal{V}_2$) > 0
then go to ⟨ uus element ⟩ ; $\mathcal{L}$; $\mathcal{O}$:= $\mathcal{O} + \mathcal{V}_2$; go to L;

---

[10] Erijuhtudel võib toimuda väljumine tsüklioperaatorist ka teisiti.

3. "While"-element töötab nagu programmilõik:

L : $\mathcal{A} := \mathcal{P}$; if $\neg \mathcal{C}$

then go to < uus element > ; $\mathcal{L}$; go to L;

Metakeele väljend "go to < uus element > " tähistab üleminekut tsükli loendi järgmise elemendi juurde või tsüklioperaatori töö lõppu, kui vaadeldav element oli loendis viimane.

Nii näiteks töötab eespool esitatud tsüklioperaator järgmiselt (eeldame, et töö algul S = 0):

i := 1 ; S := a[1] ; i := 4 ; S := S + a[4] ;
i := 5 ; S := S + a[5]; i := 7 ; S := S + a[7] ;
i := 9 ; S := S + a[9]; <jne.> ,

millega saame summa

S = a[1] + a[4] + a[5] + a[7] + a[9] + a[11] + a[13] +
+ a[15] + a[20] + a[19] + a[18] + a[17] + a[16].

Tsüklioperaatori korral võib operaator $\mathcal{L}$ olla muuhulgas ka tsüklioperaator, millega saame n.ö. mitmekordse tsükli. Parema ülevaate saamiseks kirjutatakse selline operaator sageli kujul, kus vastavad tsükli parameetrid ja loendid on üksteise all, seega näit. kujul

for i := 1 step 1 until m do
for j := 1 step 1 until n do S := S + a[i, j] .

Antud juhul on tegemist kahekordse tsükliga, kus välimine tsükkel teostatakse parameetri i, sisemine parameetri j

järgi. Arvestades tsüklioperaatori töö üldiseloomustust, muutub sisemine parameeter kiiremini: iga i korral teostatakse sisemine tsükkel j järgi.

Lõpuks teeme tsüklioperaatori kohta veel kaks märkust.

1. Väljastpoolt tsüklioperaatorit pole võimalik suunata tsüklioperaatori mingi alamoperaatori juurde.

2. Tsüklioperaatori töö võib lõppeda kahel põhimõtteliselt erineval viisil: tsükli loendi elementide ammendamisega või siis suunamisega sellest tsüklioperaatorist väljapoole. Viimasel juhul peab operaatori $\mathcal{L}$ koosseisu kuuluma vastav suunamisoperaator. Esimesel juhul ei säili tsükli parameetri väärtus ja seda ei saa kasutada väljaspool vaadeldavat tsüklioperaatorit, teisel juhul aga parameetri (viimane) väärtus säilib ja seda saab järgnevates arvutustes soovi korral kasutada.

§ 15. Muutujate kirjeldamine.

Globaalsed ja lokaliseeritud muutujad.

Märgiste lokalisatsioon

Kirjelduste eesmärgiks on anda kirjeldatavate konstruktsioonide kohta täiendavat informatsiooni, mis on vajalik kas programmi transleerimiseks või salvestusseadmete ökonoomseks jaotamiseks. Kirjeldusi on üldse kolme liiki:

1. muutujate kirjeldused, mis jagunevad omakorda lihtmuutujate kirjeldusteks ja indeksiga muutujate e. massiivide kirjeldusteks;

2. lülitikirjeldused (vt. § 11);

3. protseduurikirjeldused, mis jagunevad funktsioonikirjeldusteks ja protseduurioperaatori kirjeldusteks.

Kirjeldused asuvad vastavate blokkide alguses enne bloki operaatoreid, kusjuures kirjelduste järjekord pole oluline.

Muutujate puhul vaatleme kõigepealt lihtmuutujate kirjeldusi. Kui $\mathcal{A}_1$, $\mathcal{A}_2$, ..., $\mathcal{A}_n$ $(n \geq 1)$ on lihtmuutujate identifikaatorid, siis

$$\underline{\text{real}}\ \mathcal{A}_1,\ \mathcal{A}_2,\ \langle \cdots \rangle\ ,\ \mathcal{A}_n$$

on vastavate lihtmuutujate kirjeldus, millega need kirjeldatakse kui reaalarvulised muutujad. Seevastu kirjeldusega

$$\underline{\text{integer}}\ \mathcal{A}_1,\ \mathcal{A}_2,\ \langle \cdots \rangle\ ,\ \mathcal{A}_n$$

teatatakse, et vastavad muutujad on täisarvulised ning kirjeldusega

$$\underline{\text{Boolean}}\ \mathcal{A}_1,\ \mathcal{A}_2,\ \langle \cdots \rangle\ ,\ \mathcal{A}_n\ ,$$

et nad on loogilised muutujad.

Kirjeldused eraldatakse üksteisest semikooloni abil.

Näiteks lihtmuutujate kirjeldustega

<u>real</u> a, B1, summa; <u>integer</u> c, d; <u>Boolean</u> A, B;
<u>real</u> b, C

teatatakse, et a, b, B1, C ja summa on reaalarvulised muutujad, c ja d täisarvulised muutujad ning A ja B loogilised muutujad. Neid kirjeldusi on võimalik anda ka kompaktsemal kujul, võttes kõigi reaalarvuliste muutujate kirjeldused kokku üheainsa sümboli real abil.

Bloki alguses toodud kirjelduste alusel reserveeritakse vastav arv mälupesi kirjeldatud muutujate salvestamiseks. Peale selle teatavad aga bloki alguses toodud kirjeldused veel, et vastavad muutujad on vajalikud ainult selle bloki piires. Kui blokist väljutakse hõlmavasse blokki, siis kustutatakse automaatselt kõik selles blokis kirjeldatud suurused ja vastavaid pesi kasutatakse juba teiste muutujate salvestamiseks. Seega mingis blokis kirjeldatud muutujat saab kasutada ainult selles blokis, mitte aga hõlmavas blokis.

```
A :          B :
{            {   real x;     }          }
```

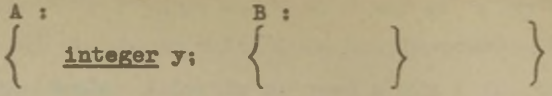
Joonis 2.

Nii näiteks saab joonisel 2 skemaatiliselt kujutatud programmis [11] muutujat x kasutada ainult blokis B, mitte aga hõlmavas blokis A.

Kui mingi muutuja on kirjeldatud hõlmavas blokis A ega ole kirjeldatud hõlmatavas blokis B (vt. joon. 3), siis on

---

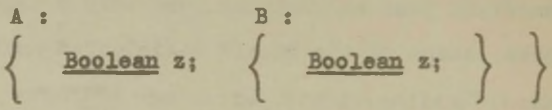
[11] Joonistel 2 - 7 on ülevaatlikkuse ja kompaktsuse huvides operaatorsulud asendatud loogeliste sulgudega.



Joonis 3.

ta kasutatav nii blokis A kui ka blokis B, sealjuures ühes ning samas tähenduses ja väärtuses. Sel juhul öeldakse, et vastav muutuja on selles piirkonnas (blokkides A ja B) globaalseks muutujaks.

Kui mingi muutuja on kirjeldatud nii hõlmavas kui ka hõlmatavas blokis (vt. joon. 4), siis vaadeldakse seda



Joonis 4.

muutujat erinevates blokkides mitte kui üht, vaid kui kaht eri muutujat - ka sel juhul, kui neil on samad kirjeldajad. Niisiis esineb blokis A tegelikult muutuja z_A, blokis B aga z_B, kusjuures blokis A pole võimalik kasutada muutujat z_B, blokis B aga muutujat z_A. Seega on vastava muutuja kehtivus ja kasutatavus piiratud vastava bloki raamidega. Sel puhul ütleme, et vastav muutuja on lokaliseeritud vastavates blokkides. Viimase juhu korral tuleb veel arvestada, et hõlmavast blokist A hõlmatavasse blokki B ülieminekul z_A väärtust ei kustutata (kuigi teda ei saa kasutada blokis B) ja seda saab kasutada blokki A tagasipöördumisel. Seevastu ülieminekul blokist B blokki A

muutuja z_B väärtus kustutatakse ja seda pole blokki B tagasipöördumisel järelikult võimalik enam kasutada.

Toome kahe viimase juhu kohta konkreetse näite, kus muutuja y on globaalne, x aga blokkides A ja B lokaliseeritud:

```
A : begin integer x, y;
        x := 1; y := x + 10;
        B : begin real x;
                x := y/2; y := x + y;
                print(x,y)
            end;
        x := x + 1; if y < 20 then go to B;
        print(x)
    end
```

Arvutuste käiku selles programmis võib fikseerida järgmiselt:

Blokis A : $x_A = 1$, $y = 11$;

B : $x_B = 5.5$, $y = \{ 5.5 + 11 = 16.5$ ümardades $\} = 17$,
trükitakse 5,5 ja 17;

A : $x_A = 2$;

B : $x_B = 8.5$, $y = 26$,
trükitakse 8.5 ja 26;

A : $x_A = 3$,
trükitakse 3.

Mõningatel juhtudel on vaja, et hõlmatavas blokis lokaliseeritud muutuja väärtus säiliks pärast ülemineku hõl-

mavasse blokki ja seda saaks kasutada hõlmatavasse blokki tagasipöördumisel. Selline olukord saavutatakse spetsiaalse kirjeldaja own lisamisega vastava muutuja kirjelduse ette hõlmatavas blokis. Süntaktiliselt on võimalik sümbolit own kasutada kõikide muutujakirjelduste puhul (vastava kirjelduse ees). Täiendades näidet jooniselt 4 tuleb vastavad kirjeldused sel korral esitada joonisel 5 näidatud kujul.

```
A :                          B :
{  Boolean z;   {  own Boolean z;          }   }
```

Joonis 5.

Sama eesmärki võib aga realiseerida ka teisiti - tähistada bloki B muutuja z näit. zB abil (eeldades, et zB ei esine blokis A) ja kirjeldada muutuja zB juba hõlmavas blokis A.

Muutujate lokaliseerimise põhimõtted kehtivad ka indeksiga muutujate ehk massiivide korral. Otstarbekohane on kõigepealt defineerida massiivikirjelduse segment:

$$\mathcal{A}_1,\ \mathcal{A}_2,\ \langle\ldots\rangle\ ,\ \mathcal{A}_m[\mathcal{B}_1\colon \mathcal{L}_1,\ \mathcal{B}_2\colon \mathcal{L}_2,\ \langle\ldots\rangle\ ,\ \mathcal{B}_n\colon \mathcal{L}_n]\ ,$$

kus $\mathcal{A}_1,\ \mathcal{A}_2,\ \ldots,\ \mathcal{A}_m$ $(m \geq 1)$ on massiivide identifikaatorid ja $\mathcal{B}_1,\ \mathcal{B}_2,\ \ldots,\ \mathcal{B}_n$ ning $\mathcal{L}_1,\ \mathcal{L}_2,\ \ldots,\ \mathcal{L}_n$ $(n \geq 1)$ aritmeetilised avaldised. Selles massiivikirjelduse segmendis on kõik massiivid (identifikaatoritega $\mathcal{A}_1,\ \mathcal{A}_2,\ldots$ $\ldots,\ \mathcal{A}_m$) n-dimensionaalsed, kus i-s indeks α_i võib muutuda ainult piirides

$$\text{entier}(\mathcal{B}_i + 0.5) \leq \alpha_i \leq \text{entier}(\mathcal{L}_i + 0.5).$$

Seejuures peab veel eeldama, et $\mathcal{G}_i \leq \mathcal{L}_i$ (i = 1, 2, ..., n), sest vastasel juhul on massiivikirjelduse segmendis süntaktiline viga.

Mistahes lõplik arv massiivikirjelduse segmente $\mathcal{M}_j$ (j = 1, 2, ..., k; k ≥ 1) lubab konstrueerida massiivikirjelduse ühel järgmisest neljast kujust:

real array $\mathcal{M}_1$, $\mathcal{M}_2$, ⟨...⟩ , $\mathcal{M}_k$,

array $\mathcal{M}_1$, $\mathcal{M}_2$, ⟨...⟩ , $\mathcal{M}_k$,

integer array $\mathcal{M}_1$, $\mathcal{M}_2$, ⟨...⟩ , $\mathcal{M}_k$,

Boolean array $\mathcal{M}_1$, $\mathcal{M}_2$, ⟨...⟩ , $\mathcal{M}_k$.

Neist kaks esimest kirjeldust kirjeldavad kõik vastavad massiivid kui reaalarvuliste elementidega massiivid, kolmas kirjeldab nad kui täisarvuliste elementidega massiivid, neljas kui tõeväärtuste massiivid.

Näiteks võivad bloki alguses olla massiivikirjeldused

integer array A, B[1 : 10], C[a : a + 5, -3 : b + c]; real array X[0 : 2, m - n : m + n], Y[0 : 2, m : m + n] .

Igal sisenemisel vaadeldavasse blokki arvutatakse uuesti indeksite muutumise piirid massiivikirjeldustes ja reserveeritakse nendele muutujatele vajalik arv pesi. Indeksite muutumise piire ei ole võimalik muuta antud bloki töö käigus. Sellest tuleneb lisanõue: kui indeksite muutumise piire määravates aritmeetilistes avaldistes esineb muutujaid, siis peavad need muutujad olema kirjeldatud antud blokki hõlmavas

blokis. Nii peavad esitatud näite puhul muutujad a, b, c, m ja n olema kirjeldatud hõlmavas blokis.

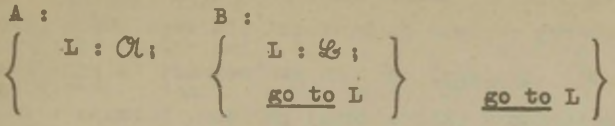
Kõneldes paragrahvis 12 liitoperaatori ja bloki erinevusest märkisime, et väljaspool blokki asuva suunamisoperaatoriga pole võimalik bloki sisse suunata. Siinkohal on otstarbekohane seda küsimust selgitada üksikasjalikumalt.

1. Juhul, kui vaadeldavas blokis on suunamisoperaator go to L (või sellega samaväärne konstruktsioon) ning mingi üks selle bloki operaator on varustatud märgisega L, annab operaator go to L juhtimise sellele operaatorile.

2. Juhul, kui vaadeldavas blokis on suunamisoperaator go to L (või sellega samaväärne konstruktsioon), aga selles blokis pole ühtegi operaatorit märgisega L, tuleb minna vahetult järgmisesse hõlmavasse blokki ja jätkata tööd märgisega L märgistatud operaatorist. Kui hõlmavas blokis sellist operaatorit pole, tuleb edasi vaadata jälle järgmist hõlmavat blokki. Kui antud programmis üheski hõlmavas blokis pole operaatorit märgisega L, siis töötab vaadeldav suunamisoperaator go to L kui tühioperaator.

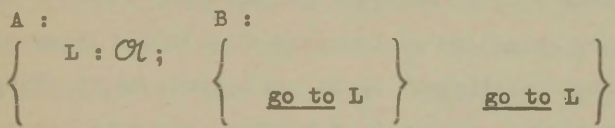
Samu reegleid on võimalik sõnastada ka teisiti, kasutades (üldistades) objektide lokaliseerituse mõistet ka märgiste puhul. Mingi operaatori ees seisev märgis loetakse automaatselt lokaliseerituks minimaalses blokis, mis sisaldab seda märgist (täpsemini: märgise esinemist). Nii näit. on joonisel 6 esitatud juhul märgised L lokaliseeritud blokkides A ja B ning blokis B asuv suunamisoperaa-

tor go to L ei saa suunata bloki A operaatori $\mathcal{A}$ juurde.



Joonis 6.

Teistsugune on olukord aga joonisel 7 esitatud juhul, kus L on bloki B suhtes globaalne märgis ja bloki B suunamisoperaatorile go to L seega kättesaadav.



Joonis 7.

§ 16. Protseduurikirjeldused

Soovides kasutada mittestandardseid protseduure, tuleb programmis (vastavate blokkide alguses) esitada ka nende kirjeldused. Et funktsiooni ja protseduurioperaatori kirjeldused on põhimõtteliselt ühesugused, siis käsitleme neid koos.

Funktsioonikirjelduse üldine kuju on

$\mathcal{A}$ procedure $\mathcal{L}$ $\mathcal{L}$,

protseduurioperaatori puhul aga

procedure $\mathcal{G}\,\mathcal{L}$,

kus $\mathcal{A}$ määrab vastava funktsiooni väärtuse tüübi (kas real, integer või Boolean), $\mathcal{G}$ on protseduuri (kirjelduse) päis ja $\mathcal{L}$ protseduuri (kirjelduse) põhiosa. Viimane määrabki üksikasjalikult, mida antud protseduuri puhul tuleb teha.

Protseduuri põhiosa esitamiseks on keeles ALGOL ette nähtud kaks põhimõttelist võimalust. Esiteks võib teda kirjutada ALGOLi operaatorina (erijuhul blokina). Täiendavaid üksikasju selle juhu kohta anname hiljem.

Teiseks võib protseduuri põhiosa kirja panna suvalise keele tekstina, millel ei tarvitse olla ALGOLi konstruktsioonidega mingit olulisemat seost. Seda protseduuri põhiosa $\mathcal{L}$ esitamise moodust nimetatakse koodiks. Enamasti kasutatakse sel korral konkreetse arvuti käskude süsteemis (ehk nn. masinakoodis) kirjapandud programmiosa, millest nimetatud moodus ka oma nime on saanud. Koodina on otstarbekohane või vajalik kirja panna need protseduuri põhiosad, milles kas kasutatakse arvuti väljund- või sisendseadmeid, vahetatakse informatsiooni üksikute salvestusseadmete vahel, töödeldakse alfabeetilist informatsiooni, kasutatakse rivisid, opereeritakse pesa üksikute kahendkohtadega vms. Et koodi kasutamine on seotud konkreetse arvutiga (resp. ALGOLi konkreetse esitusega mingi arvutitüübi jaoks), siis selle kohta etalonkeeles üksikasjalikumaid juhiseid ei ole võimalik anda.

Edasi vaatleme, kuidas tuleb kirja panna protseduuri päis. Selle üldkuju on järgmine:

$\mathcal{L}_0$ $\mathcal{L}_1$ $\mathcal{L}_2$ $\mathcal{L}_3$,

kus $\mathcal{L}_0$ tähendab vastava protseduuri identifikaatorit, $\mathcal{L}_1$ on formaalsete parameetrite kogum, $\mathcal{L}_2$ väärtuste kogum ja $\mathcal{L}_3$ spetsifikatsioonide kogum. Osade $\mathcal{L}_1$, $\mathcal{L}_2$ ja $\mathcal{L}_3$ kohta tuleb muidugi omakorda anda veel üksikasjalikud konstruktsioonireeglid.

1. Mistahes protseduuri faktilised parameetrid (põhiprogrammis) ja formaalsed parameetrid (kirjelduses) on üldiselt erinevad objektid. Nii võib faktiliseks parameetriks olla aritmeetiline avaldis, formaalseks parameetriks võib olla aga ainult identifikaator. Nõutav on siinjuures see, et protseduuri faktiliste parameetrite arv võrduks formaalsete parameetrite arvuga. Vastavus faktiliste ja formaalsete parameetrite vahel määratakse järjekorranumbrite alusel. Nii vastab esimesele faktilisele parameetrile esimene formaalne parameeter jne. Seega peab formaalsete parameetrite hulk ilma parameetriteta protseduuri korral olema tühi (s.t. tühisõna), n parameetri (n ≥ 1) korral aga on selle kuju järgmine:

$$(\mathcal{P}_1,\ \mathcal{P}_2,\ \langle\cdots\rangle\ ,\ \mathcal{P}_n) \quad ,$$

kus $\mathcal{P}_i$ $(1 \leq i \leq n)$ on identifikaatorid. Koma asemel võib parameetreid eraldada ka teisiti (vt. § 5).

2. Vastavate faktiliste ja formaalsete parameetrite seostamiseks on põhimõtteliselt kaks võimalust: 1) formaalsele parameetrile antakse vastava faktilise parameetri väärtus (arvuline või loogiline), mis sellel on protseduuri

poole pöördumise momendil; 2) formaalne parameeter asendatakse tervikuna vastava faktilise parameetriga (seejuures asetatakse avaldised sulgudesse). Väärtuste kogumis loetletaksegi need formaalsed parameetrid, mille puhul kasutatakse esimest seostamise moodust. Ülejäänud formaalsete parameetrite puhul kasutatakse automaatselt teist moodust. Seega võib väärtuste kogum olla tühi (s.t. tühisõna) või kujul

$$\underline{\text{value}}\ \mathcal{P}_{i_1}\ ,\ \mathcal{P}_{i_2}\ ,\ \langle\cdots\rangle\ ,\ \mathcal{P}_{i_k}\ ;$$

kus $1 \leq i_1, i_2, \ldots, i_k \leq n$.

3. Spetsifikatsioonide kogum sisaldab programmi transleerimiseks vajalikku informatsiooni formaalsete parameetrite kohta. Spetsifikatsioonide kogumis pole vaja esitada kõiki formaalseid parameetreid, tingimata tuleb aga spetsifitseerida kõik need parameetrid, mis esinevad väärtuste kogumis. Spetsifikatsioonide kogum võib mõnel juhul olla ka tühi (näit. siis, kui väärtuste kogum on tühi). Kui spetsifikatsioonide kogum pole tühi, siis koosneb ta lõplikust arvust järgmise konstruktsiooniga spetsifikatsioonidest:

$$\gamma\ \mathcal{P}_{j_1},\ \mathcal{P}_{j_2},\ \langle\cdots\rangle\ ,\ \mathcal{P}_{j_1}\ ;$$

kus $1 \leq j_1, j_2, \ldots, j_l \leq n$, $\mathcal{P}_{j_1}, \mathcal{P}_{j_2}, \ldots, \mathcal{P}_{j_l}$ on formaalsed parameetrid ja γ tähendab spetsifitseerivat väljendit. Spetsifitseerivad väljendid võivad olla järgmised:

$$\underline{\text{array}} \mid \underline{\text{procedure}} \mid \underline{\text{label}} \mid \underline{\text{string}} \mid \underline{\text{switch}} \mid \langle\text{tüüp}\rangle \mid \langle\text{tüüp}\rangle\ \underline{\text{array}} \mid \langle\text{tüüp}\rangle\ \underline{\text{procedure}}\ ,$$

kus ⟨tüüp⟩ tähistab kas tähte real, integer või Boolean.

Spetsifikatsioonid näitavad formaalsete parameetrite liiki protseduurikirjelduse põhiosas ning nendega peavad olema kooskõlas vastavad faktilised parameetrid. Nii näiteks, kui teatav formaalne parameeter on spetsifitseeritud kui real, siis peab vastav faktiline parameeter olema aritmeetiline avaldis; formaalsele parameetrile, mis on spetsifitseeritud kui label, peab vastama märgiseavaldis; formaalsele parameetrile, mis on spetsifitseeritud kui string, peab vastama rivi jne.

Esitame nüüd näitena suuruse

$$S = \begin{cases} -1 & , \text{ kui } a_m + a_{m+1} + \ldots + a_n < 0; \\ \sqrt{a_m + a_{m+1} + \ldots + a_n} & , \text{ kui } a_m + a_{m+1} + \ldots + a_n \geq 0 \end{cases}$$

arvutamise protseduuri kirjelduse kahel kujul: kui protseduur on esitatud funktsioonina ja kui ta on esitatud protseduurioperaatorina (seejuures eeldame, et $m \leq n$). Põhiprogrammis olgu vastavad protseduurid näit. kujul

S(a, m, n) ja S(a, 10, m + n, R),

kus esimesel juhul omistatakse tulemuse väärtus funktsioonile enesele, teisel juhul aga selles spetsiaalselt ettenähtud parameetrile R.

Esimesel juhul võib protseduurikirjelduse esitada näiteks nii:

```
real procedure S(A, p, q); value p, q; integer p, q;
real array A;
begin integer i; real r;
      r := 0.0; for i := p step 1 until q do
      r := r + A[i];
      if r ≥ 0 then r := sqrt(r) else r := -1;
      S := r
end;
```

teisel juhul aga näiteks nii:

```
procedure S(A, p, q) resultaat : (T); value p, q;
integer p, q; real array A; real T;
begin integer i;
      T := 0.0; for i := p step 1 until q do
      T := T + A[i];
      if T ≥ 0 then T := sqrt(T) else T := -1
end;
```

Seoses esitatud protseduurikirjeldustega märgime veel, et formaalsed parameetrid ei tarvitse ja ei saagi ühtida vastavate faktiliste parameetritega (ülaltoodud teises näites on faktilisteks parameetriteks arv ja aritmeetiline avaldis). Pöördumisel protseduurikirjelduse poole selgitame kõigepealt, kuidas vastavaid faktilisi ja formaalseid parameetreid tuleb seostada. Et A pole toodud väärtuste hulgas, siis tuleb A protseduuri põhiosas igal pool asendada a-ga. Parameetrid p ja q on aga toodud väärtuste hulgas, mis tähendab tegelikult seda, et enne protseduuri põhiosa täitmist teostatakse omistamised p := m ja q := n või tei-

sel juhul vastavalt p := 10 ja q := m + n. Teisel juhul asendatakse veel formaalne parameeter T parameetriga R (sest T puudub väärtuste hulgas). Seejärel täidetakse protseduurikirjelduse põhiosa, mis antud näidetes kujutab blokki. Funktsiooni kirjelduse põhiosas peab esinema omistamisoperaator, kus vasakul pool omistamismärki on selle protseduuri identifikaator. Pärast protseduuri põhiosa täitmist pöördutakse tagasi põhiprogrammi (samasse kohta, kust väljuti), kusjuures funktsioonil S(a, m, n) või vastavalt parameetril R on juba vajalik väärtus.

Teeme veel mõned üldisema iseloomuga märkused seoses protseduurikirjeldustega.

1. Faktiliste parameetrite kaudu võib protseduurikirjelduse põhiossa juurde tulla uusi identifikaatoreid. Mõned juurdetulevad identifikaatorid võivad sealjuures ühtida protseduuri põhiosas lokaliseeritud identifikaatoritega. Tekib nn. "tähiste kollisioon", mis aga translaatori poolt automaatselt likvideeritakse protseduuri põhiosa vastavate tähiste muutmise teel.

Olgu näiteks antud järgmine protseduurikirjeldus:

```
procedure Summa (a, m, n, r); integer m, n; real r;
real array a;
begin integer i;
      r := 0.0;
      for i := m step 1 until n do
      r := a[i] + r
end;
```

Kui põhiprogrammis esineb protseduurioperaator Summa (b, 3, i + 10, S), siis täidetakse protseduurikirjelduse põhiosa nii:

```
begin integer i1;
      S := 0.0;
      for i1 := 3 step 1 until i + 10 do
      S := b[i1] + S
end;
```

2. Formaalsetele parameetritele, mis on toodud väärtuste kogumis, omistatakse protseduuri põhiosa poole pöördumisel kindlad väärtused (eelmistes näidetes p := m, q := n jne.). Et puht-graafiliselt samad identifikaatorid (p ja q) võivad aga esineda ka programmi sama bloki teistes kohtades, siis võib selline otsene omistamine rikkuda programmi tööd. Sellepärast mõistetakse protseduurikirjelduse põhiosa täitmist nii, et need formaalsetele parameetritele vastavad identifikaatorid automaatselt lokaliseeritakse protseduurikirjelduse põhiosas (olenemata sellest, kas protseduurikirjelduse põhiosa on blokk või mitte) ja alles seejärel omistatakse nendele parameetritele-identifikaatoritele põhiprogrammiga ettenähtud väärtused.

Tähendame veel seda, et formaalseid parameetreid, mis esinevad väärtuste kogumis, ei tohi kasutada omistamisoperaatorite $\mathcal{A} := \mathcal{L}$ vasakul pool.

3. Nagu juba eespool öeldud, võivad protseduurid ol-

la ka täiesti ilma parameetriteta. Sel korral puuduvad loomulikult nii faktilised kui ka formaalsed parameetrid ja protseduur põhiprogrammis koosneb ainult protseduuri identifikaatorist. Sellist protseduuri on otstarbekohane kasutada siis, kui näit. mingi suuruse arvutamine toimub ühtede ja samade faktiliste parameetrite puhul.

Olgu näiteks programmi paljudes kohtades vaja arvutada suurust

$$S = \sqrt{(\sum_{i=0}^{n} a_i)^2 - \sum_{i=0}^{n} a_i^2},$$

kus n on mingi fikseeritud naturaalarv. Vastav parameetriteta protseduur näeb põhiprogrammis välja kujul S, tema kirjeldus on aga kujul

```
real procedure S; begin integer i; real u, v;
                    u := v := 0.0;
                    for i := 0, i + 1 while i ≤ n do begin
                    u := a[i] + u; v := a[i]↑2 + v end;
                    S := sqrt(u ↑ 2 - v)
                  end;
```

4. Rekursiivseteks protseduurideks nimetatakse niisuguseid protseduure, mille kirjelduse põhiosas esineb pöördumine (kas vahetult või mingi teise protseduuri kaudu) sellesama protseduuri poole.

Näiteks faktoriaali arvutamise protseduuri kirjelduse võib esitada järgmisel rekursiivsel kujul ($n \geq 0$):

```
integer procedure faktoriaal (n); value n; integer n;
        faktoriaal := if n = 0 then 1 else
        n × faktoriaal (n - 1);
```

Et translaatori selle osa koostamine, mis transleerib protseduurikirjeldused arvuti käskude keelde, on rekursiivsete protseduuride korral küllaltki keeruline, siis pole paljude konkreetsete esituste puhul seda võimalust ette nähtud. Selline kitsendus keele väljendusvõimalusi oluliselt ei vähenda, sest iga rekursiivset protseduuri on võimalik alati esitada ka mitterekursiivsena. Faktoriaali arvutamise protseduuri on näiteks võimalik esitada järgmise mitterekursiivse protseduurina:

```
integer procedure faktoriaal (n); value n; integer n;
begin integer i, k; k := 1;
        for i := 1, i + 1 while i ≤ n do k := k × i;
        faktoriaal := k
end;
```

5. Peatume lõpuks veel ühel omapärasel nähtusel, mis võib esineda protseduuride koostamisel ja kasutamisel - nn. kõrvalefektil. Teatavasti on protseduuri ülesandeks kindlal viisil opereerida vastava protseduuri lähtesuurustega, näit. arvutada funktsioonide väärtused ja omistada need kindlatele identifikaatoritele. Kuid protseduuri põhiosas võivad peale formaalsete parameetrite esineda veel teised muutujad, mis võivad protseduuri põhiosa töö käigus omandada uusi väärtusi ja mis ei tarvitse olla lokaliseeritud

põhiosas. Sel puhul annab protseduur peale otseselt ettenähtud resultaatide veel kõrvaltulemusi, mis seisneb mõningate teiste muutujate (identifikaatorite) väärtuste muutumises. Äsjakirjeldatud nähtust nimetataksegi kõrvalefektiks.

Nii näiteks protseduuri

```
real procedure f(x, y); value x, y; real x, y;
begin a := a + 1;
    f := sqrt(x↑2 + y↑2)
end;
```

korral on protseduuri kõrvalefektiks muutuja a väärtuse suurendamine. Sellisel juhul ei ole aga mitte ükskõik, kas põhiprogrammis kirjutada näit.

```
z := a + f(b, c)   või   z := f(b, c) + a
```

, sest peale f(b, c) täitmist on muutunud ka a väärtus.

Ilmselt toob kõrvalefektide olemasolu protseduurides kaasa täiendavaid probleeme, kuid neid saab programmeerija ise arvestada. Pealegi on kõrvalefektil ka kasulikke rakendusi. Kõigi nende asjaolude tõttu pole seda nähtust keelest välja jäetud, kuigi soovitatakse teda kasutada vajaduse korral siiski ainult protseduurioperaatorite puhul.

§ 17. Kommentaarid

Esitame lõpuks veel ühe konstruktsiooni - kommentaari, mis võimaldab lisada selgitavaid märkusi programmi vastavates kohtades. Niisugused märkused on mõeldud ainult inimestele, jäävad programmi transleerimisel vahele ega mõjusta seega üldse arvutuste käiku. Kommentaare võib programmi paigutada vahetult pärast tähti " ; ", "begin" või "end" järgmisel viisil:

Pärast tähte	Kommentaari kuju
;	; comment α_1 ;
begin	begin comment α_1 ;
end	end α_2 ; end α_2 end end α_2 else ,

kus α_1 on kommentaari tekst, mis võib sisaldada kõiki ALGOLi tähti peale " ; ", mis on antud juhul kommentaari lõpu tunnuseks; tekst α_2 võib sisaldada kõiki ALGOLi tähti peale kolme tähe " ; " , "end" ja "else" , mis on antud juhul kommentaari lõpu tunnuseks.

Kommentaari koos märgisega võib kasutada ka operaatorsulgude vastavuse tähistamiseks, kasutades algava operaatorsulu ees näiteks märgist A ja lisades vastavale lõppevale operaatorsulule kommentaari A, näit.

A : begin ⟨...⟩ end A ;

Lõpuks tähendame, et operaatorsulu end järele kirjutamata jäänud semikoolon (kui sümbolile end ei järgne vahetult sümbolit else või end) võib põhjustada programmis vigu, sest sellele järgnevat operaatorit peetakse kommentaariks, jäetakse seetõttu transleerimata ning täitmata.

HARJUTUSÜLESANDED

1 Kasutades induktiivset definitsiooni, Backuse metalingvistilisi valemeid ja generatiivset grammatikat, defineerida järgmine sõnade hulk:

a) Tähestiku $A = \{a,b,c,d,e\}$ kõikide mittetühjade sõnade hulk.

b) Sõnade aa, aaaa, ..., $(aa)^n$, ... hulk[1], mida võib lühidalt esitada ka kui sõnade a^{2n} $(n\geq 1)$ hulka.

c) Sõnade $(abcd)^n$ $(n\geq 1)$ hulk.

d) Sõnade $a^{2m+3n+5k}$ $(m,n,k=0,1,\ldots)$ hulk.

e) Sõnade ab^na $(n\geq 0)$ hulk.

f) Sõnade a^nb^n $(n\geq 0)$ hulk.

g) Sõnade $a^{3m+5n}b^{3m+5n}$ $(m,n\geq 0)$ hulk.

h) Sõnade $a^{3m+7n}b^{5m+2n+1}$ $(m,n\geq 0)$ hulk.

i) Sõnade $a^{5p+2q+3r}b^{5t+1}c^{7p+q+4r}$ $(p,q,r,t\geq 0)$ hulk.

j) Sõnade $a^{m+9n}b^{3k+7b+2}c^{3p}d^{2k+1}e^{5m+2n+3}$ $(m,n,k,l,p\geq 0)$ hulk.

---

[1] Tähisega $(P)^n$ märgime sõnade ühendit PP...P, kus sõna P on n korda järjestikku kirjutatud. Kui sõna P koosneb ühest tähest, siis kirjutame vastava tähise ilma sulgudeta, näit. a^m, b^{2n+3} jne. $(P)^0$ on tühisõna.

2 Lähtudes tähestikust $B = \{0, I, /, -\}$ defineerida kolmel eri moodusel ratsionaalarv kui kahe täisarvu jagatis, kus nimetaja on positiivne. Sümboleid 0 ja I ei tarvitse tingimata vaadelda kahendsüsteemi numbritena.

3 Rooma numbrite süsteemis kasutatakse arvude 1, 5, 10, 50, 100, 500 ja 1000 tähistamiseks vastavalt numbrimärke I, V, X, L, C, D ja M. Täisarvud 1 - 9 kirjutatakse selles süsteemis järgmiselt: I, II, III, IV, V, VI, VII, VIII ja IX. Analoogiliselt tähistatakse kümnelised ja sajalised.

Kasutades induktiivset definitsiooni anda reeglid täisarvude 1 - 3999 kirjutamiseks rooma numbrite süsteemis.

Võttes kasutusele tähised X_i ja V_i vastavalt arvude 10^i ja 5.10^i $(i=0,1,\ldots)$ tähistamiseks on võimalik rooma numbrite süsteemi üldistada. Kasutades induktiivset definitsiooni anda reeglid mittenegatiivsete täisarvude kirjutamiseks rooma numbrite üldistatud süsteemis.

4 Olgu antud järgmine mehe- ja naisenime induktiivne definitsioon:

1. Ants, Endel, Ivar, Karl, Leo, Mati, Olaf ja Ülo on mehenimed;
2. Ene, Endla, Eva, Helmi, Laine, Leida, Maret, Reet ja Tamara on naisenimed;
3. kui $\mathcal{A}$ on mehenimi ja $\mathcal{B}$ on kas mehe- või naisenimi, siis $\mathcal{A}-\mathcal{B}$ on mehenimi;

4. kui $\mathcal{L}$ ja $\mathcal{P}$ on naisenimed, siis $\mathcal{L}-\mathcal{P}$ on naisenimi.

Selgitada, kas kirjutised Ene-Endel-Karl-Maret, Eva-Helmi-Reet, Ants-Leida-Ülo-Laine ja Mati-Mai-Tamara on mehe- või naisenimed.

Esitada toodud definitsioon Backuse metalingvistiliste valemite ning generatiivse grammatika abil (soovi korral muutes nimede loetelu punktides 1 ja 2).

5 Olgu antud tähestik $C = \{a,b,c,d,e,f\}$ ja induktiivne definitsioon:

1. a, b ja c on klassi K sõnad;
2. d, e ja f on klassi L sõnad:
3. kui $\mathcal{O}$ on klassi K sõna ja $\mathcal{G}$ on klassi L sõna, siis $\mathcal{O}\mathcal{G}$ on klassi K sõna;
4. kui $\mathcal{O}$ on klassi K sõna ja $\mathcal{G}$ on klassi L sõna, siis $\mathcal{G}\mathcal{O}$ on klassi L sõna.

Selgitada, missuguse klassi sõnad on dabe, cebe, bedac ja fecad.

Esitada toodud definitsioon Backuse metalingvistiliste valemite ning generatiivse grammatika abil.

6 Olgu antud tähestik $D = \{+,-\}$ ja definitsioon

⟨muster⟩ ::= + | ⟨muster⟩ - + | + ⟨muster⟩ | - ⟨muster⟩ -

a) Esitada see definitsioon generatiivse grammatika abil ning samuti induktiivse definitsioonina.

b) Selgitada, kas järgmised sõnad on "mustrid":

- - + - -
+ + - + +
- + - + - +
+ - + - - +

c) Koostada neli vähemalt kuuetähelist sõna tähestikus D, mis ei ole "mustrid".

7 Esitada "mustri" võimalikult lihtne definitsioon nii, et tähestiku D = {+,-} sõnad

+ - - + - +
+ + - + + +

aga samuti

+ - + +
+ - + - + +
+ - + - + - + + jne.
- - + -
- - + - + -
- - + - + - + - jne.

osutuksid "mustriteks", sõnad

- + - + -
+ + - + - +
+ - + - + - -

aga mitte.

8 Esitada definitsioon

⟨muster⟩ ::= +|++|+- | ⟨muster⟩ + | ⟨muster⟩ - |
⟨muster⟩ ⟨ muster⟩ | ⟨muster⟩ +- | ⟨muster⟩ -+

võimalikult lihtsal kujul.

9 Olgu antud generatiivne grammatika $G = (A, \Xi, R, \sigma)$, kus $A = \{a,b\}$, $\Xi = \{\xi, \sigma\}$, $R = \{\sigma \to ab,\ \sigma \to a\xi\sigma b,\ \xi \to b\sigma b,\ \xi\sigma \to b,\ \xi \to \}$. Kas sõnad abba ja ababbabbb on genereeritavad sellest grammatikast lähtudes?

10 Anda (üldkujuline) generatiivne grammatika keele $L = \{a^n b^n c^n \mid n \geq 1\}$ jaoks.

11 Kas järgmised konstruktsioonid kujutavad arve? Jaataval juhul selgitada arvu tüüp (real, integer) ja näidata täpsem liigitus (täisarv, kümnendmurd, kümnendarv, kümnendjärk, märgita arv, arv).

a) $34.6_{10}4 \mid +079 \mid {}_{10}{-09} \mid -_{10}{+27} \mid 17.6 \mid -_{10}{+2.8} \mid 352 \mid$
$-.43 \mid .99 \mid 3._{10}{-5} \mid .41_{10}{+15} \mid .7+_{10}2 \mid -54.9 \mid 13_{10}{-4}$

b) $.07_{10}{-24} \mid 19_{10}{+3} \mid -006 \mid 31.4 \mid +.93 \mid {}_{10}{+16} \mid 74 \mid .25 \mid$
$.6-_{10}3 \mid 52.8_{10}{-4} \mid 18._{10}{+2} \mid +75.9 \mid +_{10}{-43} \mid {}_{10}{-4.3}$

c) $074 \mid +074 \mid 58._{10}{-12} \mid 489_{10}{+2} \mid 68-._{10}{-14} \mid ._{10}{+33} \mid$
$25.7_{10}{+2.1} \mid .178 \mid -.178 \mid 834.05 \mid -834.05 \mid ._{10}{-1}$

d) $.263 \mid -.45 \mid 723.16 \mid -723.16 \mid +087 \mid 6.05_{10}{-4.0} \mid$
$-._{10}{+12} \mid 24_{10}{-3} \mid .5+_{10}09 \mid -.5_{10}09 \mid ._{10}{-27} \mid 3._{10}{-3}$

12 Lihtsustada maksimaalselt järgmiste arvude kirjutist (arvuline väärtus ja arvu tüüp peavad seejuures säilima).

a) $0.6_{10}{+12} \mid +1.0_{10}{-3} \mid -0_{10}{+5} \mid 3.0_{10}{+7} \mid 5.8_{10}{+0}$

b) $-.4_{10}+13 \mid +0.0_{10}-03 \mid -1.0_{10}+08 \mid -5.2_{10}+11 \mid +25.8_{10}-0 \mid 48.37_{10}-03$

c) $+0_{10}-4 \mid 6.0_{10}+7 \mid -1.0_{10}+2 \mid 3.9_{10}-0 \mid 0.8_{10}+14$

d) $+0.3_{10}-08 \mid -0.0_{10}+24 \mid +1.0_{10}-07 \mid +7.40_{10}-16 \mid -16.2_{10}+0 \mid 53.12_{10}-02$

13 Leida järgmiste tehete tulemused, mis peavad õigesti kajastama ka arvu tüüpi (mõningatel juhtudel võib tulemus olla defineerimata).

a)

$3 \times 7 =$	$4 + 3 =$
$2.4 \times 0 =$	$7 - 2.0 =$
$5.5 \times 2 =$	$14 \div 4 =$
$16/4 =$	$14 \div (-4) =$
$3/.0_{10}3 =$	$8 \div 2.0 =$

b)

$4 \times 5 =$	$9 - 2 =$
$3 \times 0.0 =$	$3 + 4.0 =$
$6.5 \times 4 =$	$16 \div 3 =$
$15/3 =$	$(-16) \div 3 =$
$5/.0_{10}2 =$	$6.0 \div 3 =$

c)

$.0_{10}2 \uparrow (-7) =$	$3 \uparrow 3.0 =$
$.0_{10}2 \uparrow 3.2 =$	$64 \uparrow 0 =$
$7.3 \uparrow 0 =$	$0.0_{10}-4 \uparrow 0 =$
$3.5 \uparrow (-2) =$	$3.4 \uparrow 2 =$
$9 \uparrow 3 =$	$(-5) \uparrow 2.0 =$

d)

$.0_{10}-3\uparrow 4.4 =$	$4.7\uparrow 2 =$
$6.4\uparrow 0 =$	$.0_{10}-3\uparrow(-4) =$
$59\uparrow 0 =$	$4.2\uparrow(-2) =$
$8\uparrow 4 =$	$0.0_{10}3\uparrow 0 =$
$(-4)\uparrow 3.0 =$	$4\uparrow 2.0 =$

14 Millised järgmistest konstruktsioonidest ei saa olla muutujad (mispärast)?

x↑2	Suu1,5
mass[1.5]	aA3b
sinx	A1[3a]
1A3B[1]	∝2
funktsioon	cos
a+3	175
B_{10}	ABC

15 Arvutada järgmiste elementaarfunktsioonide väärtused (pidades silmas ka tulemuse tüüpi).

a)

abs(-3)	entier(14.1)
sqrt(25)	abs(0)
entier(-13.3 + 0.5)	sign(3.7)
sign(-.01)	exp(ln(3))

b)

abs(3 - 5)	entier(-5.3)
sqrt(12 + 24)	abs(4)
entier(13.3 - 0.5)	sign(-4.3)
sign(0.0)	exp(ln(5))

16 Esitada järgmised indeksiga muutujad tavalises kirjaviisis, kui on teada, et i = 3,4 ja k = 1,7.

a)

A[2×i+k,1.7+i×k]	E[i-3,k]
b[i]	f[k+0.1,-i+5]
C[5×k,3/i]	G[i-2,3+k×2,1]
d[i-2×k,4-2×k]	h[i-k,i+k↑2]

b)

A[7/k-1,2-i]	E[i×2-1.5,2-k]
b[i-3.5,2.1+k]	f[k]
C[i×3,3-k×2,5]	G[i/2,k↑2]
d[i×k,i]	h[i-3,2-k]

17 Kirjutada järgmine valem aritmeetilise avaldisena.

a) $\frac{m - 2A}{2m + A} \cdot \frac{m(n^2+1)}{m^2 + n^2} \cdot \sin\frac{\alpha}{2}$

b) $\frac{x + y - \sqrt{(x-y)^2 + 4z}}{2}$

c) $e^{\alpha_{i,j+k}} \sqrt{\cos^3\beta + \frac{2}{3}\gamma_n^5} + \frac{(|A|-|B|)^4}{28}$

d) $\frac{(|A|+|B|)^3}{25} \ln\alpha_{ij} - \frac{3}{4}\sqrt{\beta_{k,j+5}^{17} \sin^2\gamma_\psi}$

e) $\cfrac{a_1 \cdot a_2}{-a_3 + \cfrac{a_4}{a_5 - \cfrac{a_6}{a_7}}}$

f) $\cfrac{a_1 - \cfrac{a_2 - \cfrac{a_3}{a_4}}{a_5}}{a_6 + a_7}$

h) max(a,b)

i) min(a,b,c)

18 Esitada järgmised ALGOLis kirjutatud aritmeetilised avaldised tavalises kirjaviisis:

a) `(a + b↑2×c)/(d + c↑2)`
`(c/(a↑3 + b) - a/b)/(a + c/(a + b↑3))`

b) `-a×b↑2 + b×sin(a/b)×c + a↑b/c`
`(a[i] + ((c + d/b)/b)/b)/b`

c) `(a + b)↑(2↑(c + 3) + 3)`
`(a + sqrt(a↑2 + b↑2))/(b↑2 - (a - b)/(a + b))`

d) `((a + b)↑(a↑2) - b↑2)×(a↑2 - b)↑2`
`(a - b↑2/c)/(b + a/c↑2)`

19 Järgmiste valemite kirjutamisel aritmeetiliste avaldistena on tehtud vigu. Parandada need vead.

a) $\frac{\cos^2 2x - \sin^2(x+y)}{x^2 + 1}$ `(cos(2x)↑2 - sin(x+y)↑2)/x↑2+1`

$(a + b)^{c^2+d}$ `(a + b)↑c↑2 + d`

b) $\frac{\cos^2(x^2 + 1) - y}{\sqrt{x + y^2}}$ `cos↑2(x↑2+1) - y/sqrt(x+y)↑2`

$e^{x^2} + x^{e^{2x}}$ `exp(x)↑2 + x↑exp(2x)`

20 Missugused väärtused omandab aritmeetiline avaldis

a) if x ≥ 0.5 then (if x ≤ 10 then sqrt(x) else x/3)
else if x ≥ -12 then x↑2 - 5 else abs(x + 4)

kui x = 0; 3; -15 ja 15?

b) if x > $0.3_{10}2$ then (x - $0.3_{10}2$)↑2 else if x < (-2)
then 3 × x - 10 else if x ≥ 0 then 4×x↑2 else
a↑2 + 5×b

kui x = -3; -1; 4 ja 35?

c) if x ≥ $0.55_{10}2$ then (if x < $_{10}2$ then x - $0.5_{10}2$
else a↑3) else if x ≤ 10 then 4×x else x↑2/2 - 6

kui x = -3; 8; 57 ja 106?

d) if x ≥ 3.8 then (if x < $6.6_{10}1$ then sqrt(x) else
x - $_{10}2$)else if x < $-0.95_{10}1$ then x↑2 else x/2

kui x = -12; 2; 38 ja 75?

e) if x < 5.4 then (if x ≥ $-2.9_{10}1$ then x + $1.7_{10}1$
else sqrt(-x)) else if x ≥ $.78_{10}2$ then x - $_{10}2$
else x/3

kui x = -34; -15; 14 ja 82?

f) if x ≥ $-0.6_{10}-2$ then (if x ≤ $0.3_{10}2$ then x↑2 + 3.5
else x - 25) else if x ≥ $-15_{10}2$ then sqrt(x + $15_{10}2$
) else x + $15_{10}2$

kui x = $-1.6_{10}3$; $-1.4_{10}3$; 0.0 ja $4_{10}1$?

21 Arvutada aritmeetilise avaldise

if a > 0 then (if b > 0 then a + b else a - b)
else if c > 0 then a - c else b + c

väärtus, kui

a) a = 1; b = 3 ja c = 3.

b) a = 1; b = -5 ja c = 4.

c) a = -2; b = 5 ja c = 2,5.

d) a = -2; b = 10 ja c = -7.

22 Kirjutada aritmeetiline avaldis, mis:

omandab väärtuse $-x$, kui $x \in (-\infty, -15]$,
omandab väärtuse $\sin(-x)$, kui $x \in (-15, -10]$,
omandab väärtuse e^{x+1} , kui $x \in (-10, -3]$,
omandab väärtuse $\frac{2}{3}x^2 - 6$, kui $x \in (-3, +3)$,
omandab väärtuse $\sqrt{x - 3}$, kui $x \in [3, 10)$,
omandab väärtuse $\sin x$, kui $x \in [10, 15)$ ja
omandab väärtuse x , kui $x \in [15, +\infty)$.

23 Kirjutada järgmise kahe aritmeetilise avaldise summa, vahe, korrutis ja jagatis (igaüks kahel erineval viisil).

a) if x < 0 then a else b
ja
if y ≥ 3 then c else d

b) if x > 6 then 2 else 10
ja
if y ≤ 0 then -3 else 1/4

24 Kirjutada aritmeetiline avaldis, mille väärtused xy-tasandi joonisel näidatud piirkondades (rajajoon ei kuulu viirutusepoolsesse piirkonda) on defineeritud järgmiselt:

a) $-0{,}000029$ piirkonnas I (vt. joon. 1),

$\tan^2(c + \frac{2}{15}\pi)$ piirkonnas II,

c, kui $\alpha > 0$
$c + 1$, kui $\alpha \leq 0$ } piirkonnas III,

$c^2 + 1$ piirkonnas IV.

b) $43700 \cdot 10^{-5}$ piirkonnas I (vt. joon. 1),

$\cos\frac{\alpha+1}{2}$, kui $\gamma \leq 2$
$\sin\frac{\alpha-1}{2}$, kui $\gamma > 2$ } piirkonnas II,

$\sqrt{\alpha^2 + \frac{1}{2}xy}$ piirkonnas III,

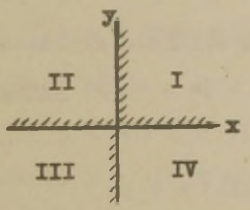
a_i piirkonnas IV.

c) $x + y$ piirkonnas I (vt. joon. 2),

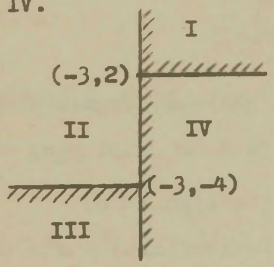
$A[4 - x]$ piirkonnas II,

$0{,}00091$ piirkonnas III,

$\sin(x - y)$ piirkonnas IV.



Joonis 1.



Joonis 2.

d) C[y + 3] piirkonnas I (vt. joon. 3),

tan x piirkonnas II,

0,000067 piirkonnas III,

A + x piirkonnas IV.

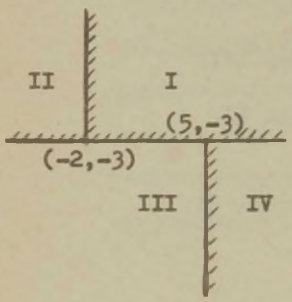
e) 10^{x-y} piirkonnas I (vt. joon. 4),

$\frac{a}{x - y}$ piirkonnas II,

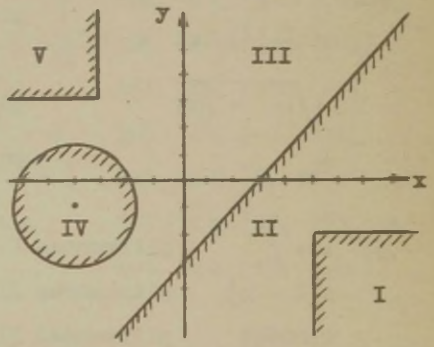
$x^2 - y^3$ piirkonnas III,

$\sqrt{10 + x + y}$ piirkonnas IV,

$\sin\frac{x}{y}$ piirkonnas V.



Joonis 3.



Joonis 4.

25 Määrata järgmiste aritmeetiliste avaldiste tüüp, kui a, b ja c on tüüpi real ning i ja j tüüpi integer.

a) i × sign(a ↑ b)

entier((a/b) - a + cos(c)) ↑ i ÷ j

j + entier(c) × sqrt(a ↑ i)

b) i ÷ entier(tan(a)/b) ↑ j ↑ 2

2 × i - sin(j)

i × (sign(a × c) - b) ↑ j

c) abs(a - i) ↑ sqrt(i)

(entier(a - b) + i) ↑ (-2)

i/j - exp(i) ↑ j - c ↑ i

26 Missugused väärtused omandab loogiline avaldis

a) if if A then x > y else x < y ↑ 2 then B else A ⊃ B

kui A = true, B = false, x = 5, y = 3 ja A = false, B = false, x = 3, y = 1?

b) if if x > y then A else B then A ⊃ B else x < y ↑ 2

kui x = 5, y = 3, A = false, B = true ja x = 3, y = 5, A = true, B = false?

c) if if x ≥ 0 then y < 3 else x > y then A ≡ B
else A ∧ B

kui x = 2, y = 1; x = 2, y = 4; x = -3, y = 2 ja x = -3, y = -5?

d) if if x ≠ y then x > y else x < 0 then x > 0
else y > 0

kui x = -2, y = -2; x = -2, y = -1; x = -1, y = 1; x = 0, y = 0; x = 1, y = -1; x = 1, y = 1 ja x = 1, y = 2?

27 Millele taandub loogiline avaldis

a) $\underline{\text{if}}\ x < -1\ \underline{\text{then}}\ A \wedge \underline{\text{true}}\ \underline{\text{else}}\ B \supset \underline{\text{false}}$

juhul kui $x = 5$ ja juhul kui $x = -3$?

b) $\underline{\text{if}}\ x \geq 0\ \underline{\text{then}}\ (\underline{\text{if}}\ y > 3\ \underline{\text{then}}\ \underline{\text{true}}\ \underline{\text{else}}\ B \supset A)$
$\underline{\text{else if}}\ y < -4\ \underline{\text{then}}\ A\ \underline{\text{else}}\ B$

juhul kui $x = 5,\ y = 4$; $x = 5,\ y = -2$ ja $x = -3,\ y = 4$?

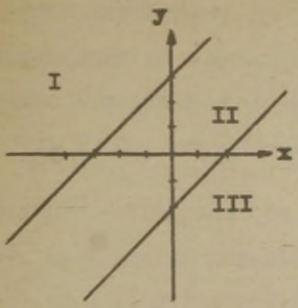
c) $\underline{\text{if if}}\ x \neq y\ \underline{\text{then}}\ x > y\ \underline{\text{else}}\ x < 0\ \underline{\text{then}}\ A \supset B$
$\underline{\text{else if}}\ y > 0\ \underline{\text{then}}\ A\ \underline{\text{else}}\ A \vee B$

juhul kui $x = -1,\ y = -1$; $x = 0,\ y = 0$; $x = 1,\ y = -1$ ja $x = 1,\ y = 2$?

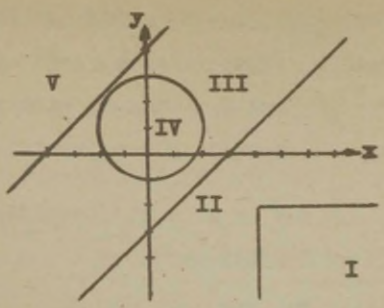
28 Kirjutada loogiline avaldis, mille väärtused xy-tasandi joonisel näidatud piirkondades on defineeritud järgmiselt:

a) $y \geq -x$ piirkonnas I (vt. joon. 5),
$(x \geq y) \supset A$ piirkonnas II,
$\underline{\text{false}} \supset (\underline{\text{true}} \wedge B)$ piirkonnas III.

b) $\underline{\text{true}}$ piirkonnas I (vt. joon. 6),
$x > -y$ piirkonnas II,
$x \geq y - 1$ piirkonnas III,
$(A \supset B) \equiv C$ piirkonnas IV,
$x^2 + y^2 > 3x + 2y^2$ piirkonnas V.

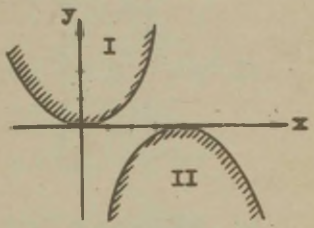


Joonis 5.



Joonis 6.

29 Kirjutada loogiline avaldis, mille väärtused määratakse järgmiselt: kui $y \geq 0$, siis on avaldise väärtus võrdne selle väite väärtusega, et punkt (x,y) kuulub piirkonda I (vt. joon. 7); kui aga $y < 0$, siis selle väite väärtusega, et punkt (x,y) kuulub piirkonda II. Joonisel näidatud piirkonnad I ja II määratakse ruutparaboolidega.



Joonis 7.

30 Kirjutada loogiline avaldis, mille väärtused määratakse järgmiselt:

a) kui $a > 0$, siis juhul $\begin{cases} \eta \geq 0{,}3 \text{ on väärtuseks } |x| < 1, \\ \eta < 0{,}3 \text{ on väärtuseks } |x-a| < 1; \end{cases}$

kui $a \leq 0$, siis juhul $\begin{cases} \eta > a \text{ on väärtuseks } \underline{\text{false}}, \\ \eta \leq a \text{ on väärtuseks } \underline{\text{true}}. \end{cases}$

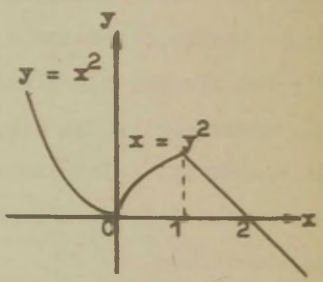
b) kui $x \geq 0$, siis avaldis on samaväärne väitega "punkt (x,y) kuulub ringi raadiusega 2 ja keskpunktiga (2;3)"; kui aga $x < 0$, siis avaldise väärtuseks on $y > -5$ korral $y > x$, vastupidisel juhul aga $x \neq y$.

c) kui $a \neq 0$, siis avaldis on samaväärne väitega "ruutvõrrandi $ax^2 + bx + c = 0$ mõlemad lahendid on reaalsed"; kui aga $a = 0$, siis avaldise väärtuseks on $b \neq 0$ korral true, vastupidisel juhul aga false.

3) Kirjutada omistamisoperaatorina muutuja y väärtuse arvutamise eeskiri, kui

a) $$y = \begin{cases} dx - \frac{b}{c}, & \text{kui } x \geq d, \\ x + \frac{d}{b}, & \text{kui } b \leq x < d, \\ bx - \frac{c}{d}, & \text{kui } x < b. \end{cases}$$

b) sõltuvus x ja y vahel on esitatud joonisel 8.



Joonis 8.

c) $y = b_i x + c_i$ poollõigul $x \in (a_i, a_{i+1}]$, kus $i = 1, 2, \ldots, 5$ ning on teada, et $a_1 < a_2 < \ldots < a_6$ ja $a_1 < x \leq a_6$.

d) $$y = \begin{cases} \ln(1 - ab), & \text{kui } ab < 0, \\ \ln \sqrt{ab + 1}, & \text{kui } a > 0 \text{ ja } ab > 0, \\ \frac{\ln(1 - a)}{\sqrt{ab + 1}}, & \text{kui } a < 0 \text{ ja } ab > 0, \\ 5{,}3, & \text{kui } ab = 0. \end{cases}$$

32 Missugused väärtused on täisarvulisel lihtmuutujal i ja reaalarvulisel indeksiga muutujal A[...] (leida ka indeksi väärtus) pärast järgmise programmiosa täitmist?

a) i := 4; A[2 × i/3 + 2] := i := i ↑ 2 + 7.5;

b) d := 3.5; i := 2; A[i + 2] := i := d + i;

c) i := 7; f := 5.4; A[i/2 + 1] := i := 2 × f - i;

d) i := 4; j := 10; A[i + j/3] := i := i + j + 5;

33 Näidata tehete järjekord järgmise omistamisoperaatori täitmisel:

a) H := m × (i + j) ↑ sin(r + p) ↑ 3/k;

b) A[2 × i + m] := - (m × i - cos(n))/p ↑ (r - q);

c) B := a × b/d + d × sin(a × b + d)/a ↑ b;

d) G[i - m/n] := i ↑ (b × cos(m - tan(n/i)) ↑ m);

e) i := 2 × (a - b + sin(a + cos(b - a)) ↑ 2);

34 Kirjutada omistamisoperaatorina loogilise muutuja L arvutamise eeskiri, kui

a) $$L = \begin{cases} \underline{\text{true}}, & \text{kui } d > b, \\ \underline{\text{false}}, & \text{kui } d \leq b. \end{cases}$$

b) $$L = \begin{cases} x > 0, & \text{kui } m = \underline{\text{false}}, \\ x = y, & \text{kui } m = \underline{\text{true}}. \end{cases}$$

c)
$$L = \begin{cases} \underline{\text{true}} & \text{, kui } x = 0 \text{ ja } y = 1, \\ y > 1 & \text{, kui } x = 0 \text{ ja } y \neq 1, \\ x < 0 & \text{, kui } x \neq 0 \text{ ja } y = 1, \\ \underline{\text{false}} & \text{ ülejäänud juhtudel.} \end{cases}$$

d)
$$L = \begin{cases} x < -1 & \text{, kui } x < 0, \\ x > 4 & \text{, kui } x > 3{,}5, \\ \underline{\text{false}} & \text{ ülejäänud juhtudel.} \end{cases}$$

35 Kirjutada omistamisoperaator, mis annab loogilisele muutujale S väärtuse true või false vastavalt sellele, kas saab või ei saa

a) sirglõikudest pikkustega x, y ja z moodustada kolmnurka.

b) risttahukat mõõtmetega x, y ja z pista läbi ristkülikulise avause mõõtmetega p ja r.

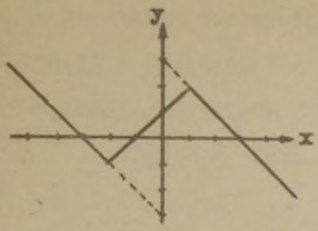
c) sirglõikudest pikkustega A, B, C ja D moodustada nelinurka.

d) ellipsist pooltelgedega a ja b välja lõigata ristkülikut mõõtmetega x ja y.

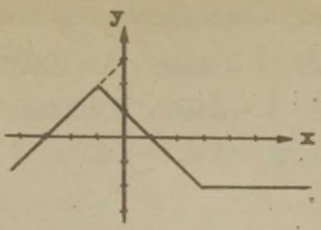
36 Esitada muutuja y väärtuse arvutamise eeskiri nii omistamisoperaatori kui ka tingimusliku operaatori kujul, kui sõltuvus x ja y vahel on antud

a) joonisega 9.

b) joonisega 10.



Joonis 9.



Joonis 10.

37 Esitada järgmine operaator tingimusliku operaatori kujul.

a) x := if y > z then a else b;

b) go to if x ≥ a then p else Q[i];

c) for i := 1 step 1 until n do S := S + 1/i;

d) x := if y ≥ 0 then sqrt(y) else if y < -5
then abs(y + 2) else y ↑ 2;

38 Kasutades lülitit koostada programmilõik, mis arvutab ja trükib välja avaldise

$$2x^2 - 3y^2 + 4xy - 16x + \frac{2}{5}y$$

väärtused muutujate x ja y väärtustel: x = 1, y = 2; x = 2, y = 3; x = 4, y = 1 ning x = 5, y = 4.

39 Kasutades lülitit koostada programmilõik, mis arvutab ja trükib välja loogilise avaldise

$$A \supset B \equiv C \vee A \supset \neg B$$

väärtused muutujate A, B ja C järgmistel väärtustel: A = = false, B = false, C = false; A = false, B = false, C = = true; A = false, B = true, C = false; A = true, B = false, C = false ning A = true, B = true, C = true.

40 Mida trükib välja järgmine programmilõik?

a)
```
begin switch R := M, L;
      x := 5; i := 0;
      M : i := i + 1;
      x := 3 × x + 1; print(x);
      go to R[i]; L :
end;
```

b)
```
begin switch R := M, N;
      S := 0; i := 1; M : S := S + a[i]↑i;
      i := i + 1; go to R[if i ≤ 6 then 1 else 2];
      N : S := abs(S); print(S)
end;
```

c)
```
begin integer x, y, z; x := y := 2; z := 0;
      P : y := x + y + z;
      begin integer x; x := 5;
            P : z := x + z; x := x + 1;
            if z ≤ x then go to P;
            print(x)
      end;
      print(x,z)
end;
```

d)
```
begin real x, y;
      y := 0.5;
      L : x := y + 1; y := x + y;
      begin real x;
            L : x := 2 × y + 1;
                y := (x + y)/2;
                if x < 9 then go to L;
                print(x,y);
      end;
      if x < 5 then go to L; print(x)
end;
```

e)
```
begin integer x, y, z; x := y := 2; z := 0;
      P : y := x + y + z;
      begin integer x; x := 5;
            P : z := x + z; x := x + 1;
            if z ≤ x then go to P
      end;
      Q : print(x,z);
      begin real z; z := 5.5; x := x + z;
            if x ≤ 2 × y then go to Q
      end;
      y := y + 1; if x ≤ 3 × y then go to Q
      print(x,y)
end;
```

f)
```
begin integer x,y; x := 2; y := 3;
      Q : x := x + y;
      begin integer x; x := 12;
            Q : y := 2 × y - 1; x := x + y/2;
            if y ≤ x/3 then go to Q;
            print(x,y)
      end;
      y := x + y; if y < 10 + x then go to Q;
      print(x,y)
end;
```

g)
```
begin real x; integer y; switch R := if y ≤ 50
      then M else P;
      y := 2; M : x := y + 3; y := 3 × y;
      begin integer x; real z;
            x := y; M : z := x + y; x := x + z;
            if z ≤ 30 then go to M;
            print(x,z)
      end;
      go to R[1]; P : print(x,y)
end;
```

h)
```
S := 5; for i := 3 step 4 until 15 do S := S + (
i - 2) ↑ 2; print(S);
```

i)
```
S := 7; for i := 2, i + 3 while i ≤ 11 do S := S
+ (i - 3) ↑ 3; print(S);
```

j) P := 1; S := 0;

for i := 1, i + 1 while i ≤ n do

begin b := a[i]; if b > 0 then P := P × b

else S := S + b

end; print(P); print(S);

41 Missugused väärtused on muutujatel a[j] ja i pärast järgmise programmilõigu tööd?

i := 5; for j := 1 step 1 until 40 do a[j] := 2 × j;

a[a[i] + 1] := i := a[i] + 4;

42 Koostada programmilõik suuruse

$$S = \frac{1 + x}{\pi} + \sum_{k=1}^{68} (-1)^k \frac{3\sin x - 1}{(2k - 1)^2 + x^2}$$

arvutamiseks, kasutades

a) suunamisoperaatorit ilma lülitita.

b) suunamisoperaatorit lülitiga.

c) tingimuslikku operaatorit.

d) tsüklioperaatorit.

43 Koostada programmilõik suuruste

$$S_i = \frac{x}{3a_{11}} + \sum_{j=1}^{n} \frac{a_{ij}x^2}{ja_{ij}x + 1}$$

arvutamiseks ja trükkimiseks, kasutades sisemises tsüklis tsüklioperaatorit, välimises aga

a) suunamisoperaatorit ilma lülitita.

b) suunamisoperaatorit lülitiga.

c) tingimuslikku operaatorit.

d) tsüklioperaatorit.

44 Koostada programmilõik, mis antud $n \times n$ maatriksi $C = (c_{ij})$ järgi arvutab ja trükib välja

a) peadiagonaali elementide summa.

b) kõrvaldiagonaali elementide summa.

c) ülalpool peadiagonaali olevate elementide summa.

d) allpool kõrvaldiagonaali olevate elementide summa.

45 Koostada võimalikult lühike programmilõik, mis annab kokkuvõttes täpselt sama tulemuse nagu järgmine programmilõik:

a) for i := 1 step 1 until 4, 50, i - 5 while i ≥ 25, 6, 8, 10, 20, 18, 16, 14, 12, 5 do a[i] := b[i] + c[i];

b) S := 0; for i := 1, 2, 3, 40, i - 4 while i ≥ 16, 14 step - 2 until 6, 4, 5, 7 do S := S + i ↑ 2;

c) if s < 3 then go to 15; t := s - i; 15:
if s ≥ 5 ∨ s ≤ 6 then t := if s < 3 then s + i else t + 2 × i else t := s + t + i;

d) a := abs(b); i := s := 0; c := entier(a);
for i := i + 1 while i ≤ c do s := s + a;
c := (s + b × (a - c)) × sign(b); a := s;

e) for i := a↑2 while c ≠ a↑2 do c := a↑2;
c := c +(if a > b then -b↑2 else if a < b
then b↑2 else a↑2); c := sqrt(if a > b
then (c + 2 × b↑2)↑2 else c↑2);

f) 1: if x = y then go to 2; t := b - a;
2: if x < y then t := -t else if x > y then
3: t := t + a × 2 else t := a - b;

g) if if a < b then a = b else a ≥ b then
c := b - a else if a > b then c := a else
if a > b then c := a + b else c := b;

h) if x ≤ y then go to 1; d := if x > y then y else x;
1: if x < y then d := if x = y then x + y else y -
x else d := if x > y then d - x else 0;

i) w := 0; for z := if x > y then -y else -x step
if y ≥ x then y else x until if x < y then 2 × y
- x else 2 × x - y, 1 do w := w + z;

j) k := if a > 0 then 2 × a else -3 × a; l := if b ≥ 0
then b else -2 × b; if k < 0 then k := k + abs(a)
else k := k - abs(a); m := l + k + (if a ≤ 0 then
a else 0) + (if b > 0 then b - abs(b) else 2 × b
+ abs(b));

k) a := if if x < y then x = y else x ≥ y then (if x < y then y - x else x) else if x < y then y else x - y; b := a - (if x > y then x - y else if x = y then 0 else y - x); c := a + b; a := b := 0;

l) 1: t := 0; if a = x then go to 3 else
2: t := t + (if a ≥ x then c else b);
3: if a ≤ x then t := t + a else if a = x then
4: t := t - b else t := t - c; if a ≥ x then t := t + b;

m) m := entier(abs(m)); n := entier(abs(n));
if n ≠ 0 then for i := 0 step 1 until m do
begin m := m - n; if m = 0 then begin i := i + 1; go to 1 end else if m < 0 then go to 2 else begin m := m + n; go to 1 end 2: end; 1:

46 Koostada programmilõik, mis antud arvudest x_1, x_2, ..., x_n lähtudes leiab

a) suurima arvu ja selle indeksi.

b) vähima arvu ja selle indeksi.

c) nende arvude aritmeetilise keskmise.

d) positiivsete arvude korrutise.

e) eraldi positiivsete ja negatiivsete arvude summa.

f) eraldi positiivsete ja negatiivsete arvude aritmeetilise keskmise.

g) suuruse $\sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})^2}$, kus $\bar{x}$ on nende arvude aritmeetiline keskmine.

47 Koostada programmilõik, mis järjestab jada x_1, x_2, ..., x_n liikmed kasvavas järjekorras.

48 Olgu antud kaks massiivi x_1, x_2, ..., x_n ja y_1, y_2, ..., y_n. Koostada programmilõik, mis leiab

a) iga $k = 1, 2, \ldots, n$ korral suuruse

$$z_k = \sum_{i=1}^{k} x_i y_{k+1-i} .$$

b) astakkorrelatsiooni kordaja

$$\rho = 1 - \frac{6\sum_{i=1}^{n}(x_i - y_i)^2}{n(n^2 - 1)} .$$

c) niisuguste paaride (x_i, y_i) arvu, kus $x_i > y_i$.

d) suuruse

$$\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2 + \sum_{i=1}^{n}(y_i - \bar{y})^2} ,$$

kus $\bar{x}$ ja $\bar{y}$ on nende massiivide aritmeetilised keskmised.

49 Koostada programmilõik, mis antud $m \times n$ maatriksi $A = (a_{ij})$ järgi arvutab ja trükib välja

a) iga veeru elementide summad.

b) iga rea elementide summad.

c) kõikide elementide summa.

d) nende elementide a_{ij} summa, kus $i < j$.

50 Koostada programmilõik, mis arvutab massiivi x kuuluvate indeksiga muutujate väärtused, kui

a) $x_j = \max\limits_{1 \le k \le s} \sum\limits_{i=0}^{m} a_{ij} \cdot \left(\min\limits_{0 \le p \le t} \sum\limits_{r=1}^{q} b_{kr} c_{pr} \right)^i$ $(1 \le j \le n)$.

b) $x_i = \min\limits_{1 \le k \le n} \sum\limits_{j=1}^{m} \left(a_{ijk} \sum\limits_{p=1}^{r} b_{ijp} \left(\max\limits_{0 \le t \le s} c_{jt} \right)^p \right)$ $(1 \le i \le q)$.

c) $x_k = \sum\limits_{i=0}^{t} \left(\min\limits_{1 \le j \le n} a_{ij} \cdot \sum\limits_{p=0}^{m} b_{pk} \cdot \left(\max\limits_{1 \le r \le q} c_{ri} \right) p \right)$ $(0 \le k \le s)$.

d) $x_k = \max\limits_{0 \le i \le m} \sum\limits_{j=0}^{n} \left(a_{ijk} \sum\limits_{r=0}^{t} b_{krj} \left(\min\limits_{1 \le s \le p} c_{sj} \right)^r \right)$ $(1 \le k \le q)$.

e) $x_i = \max\limits_{0 \le j \le m} \sum\limits_{k=0}^{t} a_{ijk} \cdot \left(\sum\limits_{p=1}^{s} \min\limits_{0 \le r \le q} b_{rpi} c_{pjr} \right)^k$ $(1 \le i \le n)$.

f) $x_i = \sum\limits_{k=1}^{n} a_{ik} \cdot \max\limits_{0 \le j \le t} \sum\limits_{p=1}^{m} b_{pj} \cdot \left(\min\limits_{1 \le r \le n} c_r \right)^p$ $(0 \le i \le q)$.

51 Koostada programmilõik, mis iga $i = 1, 2, \ldots, 15$ ja $x = 1{,}1;\ 1{,}2;\ \ldots;\ 3{,}4$ väärtuse korral arvutab ning

trükib (koos i ja x väärtustega) suuruse

a) $S(i,x) = \frac{1+x}{2i} + \sum_{j=1}^{i} (-1)^j \frac{3\sin x - ix}{(2j-1)^2 + x^2}$.

b) $f(i,x) = 1 + \frac{x}{2} + \frac{x^2}{3} + \ldots + \frac{x^i}{i+1}$.

c) $h(i,x) = \frac{a_i x^i + a_{i-1}x^{i-1} + \ldots + a_1 x + a_0}{b_i x^i + b_{i-1}x^{i-1} + \ldots + b_1 x + b_0}$.

d) $R(i,x) = \frac{1}{\pi}\sqrt{i}\,x + \sum_{j=1}^{i} \frac{2x\cos^2 x - j}{(j+1)^2 + ix^2}$.

e) $T(i,x) = \frac{ix}{2\pi} + \sum_{j=1}^{34+2i} (-1)^j \frac{(2j+7)x - 4i^2}{(j+2i)(3j-5) + ix^2}$.

52 Esitada võimalikult kompaktsel kujul järgmiste massiivide kirjeldused:

massiivi identifikaator	tüüp	indeksite arv	indeksite muutumispiirkond		
			1. indeks	2. indeks	3. indeks
a	real	1	-3 ... 10		
b	Boolean	2	1 ... 8	1 ... 2	
c	integer	2	0 ... m	1 ... k	
d	real	3	1 ... 5	0 ... p	1 ... p+k
e	real	1	0 ... 13		
f	Boolean	2	1 ... 2	1 ... 8	
g	integer	2	0 ... m	1 ... k	
h	real	3	1 ... 5	0 ... p	0...k+p-1

53 Vähemalt millised kirjeldused peavad leiduma blokis, mis sisaldab järgmisi operaatoreid?

a) for s := if a[k] = b ∨ c then d else g[k] step g[a[k]] until 3 do p[s] := d = r; go to m[i];

b) if k[3] then go to kk[3] else p[i] := a ÷ b;
4: for f := a step 1 until b do r[f] := c/d[f];

c) for i := k, m + i while n[p] do a[i] := b[i];
go to if r[j] = h ⊃ g then s[t+i] else if u
then v else w[j];

d) if a = b ∨ c then go to if d[f] then h[g] else i
else j[k] := m ÷ n = p[q, r];

e) for p := u[c], p + i[c] while t[p, c] do go to
if k[p] = a ∨ b then r[p + n[p]] else if m[p]
then v[p] else pp;

f) p[m] := a - b ÷ if r ∧ x < 1 then b ↑ 2 else k[m];
go to if s[i] ∨ s[m] then ss[i] else k3;

54 Vormistada ülesannetes 51a, 51b, 51d ja 51e koostatud programmilõigud iseseisvate blokkidena.

55 Märkida välja kõik järgmises programmis esinevad süntaktilised vead ning kirjeldada nende vigade iseloomu (programmi sisule tähelepanu pööramata):

a) begin integer a, b; real c, d, j; Boolean g, t;
Boolean array h, hh(1:7); switch tp = t, 3, t3;
for j := if g = t ∨ c = d then a else c = j until
a, b do t3: h := a - j = b + d; t4: if h = true
then go to if c < a then tp[t] else t5 end;

b) begin integer i, a; real c; Boolean d, h; integer
array f, g[1:10]; switch t2t := tt, t2; for i =
3.5 step if f < c then 1.5 until 7,5 do begin t2:
g[i] := k + c ∨ d; go to if d = h then t2t[3]
else t3 end;

c) begin integer k, m; Boolean b; real array a[1:5],
c; switch pt = 3, 4, 5; pt[1]: for k := if c[i]<b
then 1 + if b then a else m step f until a[k];
3: go to if (k ∨ m) < 5 then pt else 6; end;

d) begin integer n, b, f; real k, i; Boolean n, nn;
integer array g,h[1:n]; switch sinna := a, 1, b;
for i := 1 step 1 until i ≤ n do g[i] := h[n-i]:=
i; a: for k := 1 step 1 while g[n] + 1 begin 1: m
:= b·f + if n = nn then h[k] - g[k]; go to if k =
= m ⊃ nn then sinna[4]else 2 + sinna[1];

e) begin integer i, n; real r; real array f[i:n];
for i := 1 step 1 until i = n do f[i] := i ↑ 2;
a: if n = 3 then go to b; begin real p; p := 0.0;
for i := 1, i + 1 while i < n do p := p + f[i];
b: go to a end; r := p ↑ 2; print(r) end;

f)
```
begin integer n; real x, y; Boolean k, t; switch
s := a, b; real array f[1:n]; j := 1; a: if n < 1
then go to c; begin integer j; for j := 1 step 1
until n do begin x := f[j]; b: y := f[j]/2 end; c:
go to if k = t then b else s[3] end end;
```

g)
```
comment: see on üks lootusetu programm
begin Boolean a, b;
integer i, s; real s1, 1s, s2;
real array x, a[1:10];
s := 1.5; 5:
for i := 1 step 1 until 20 do
a[i] := i • s;
s2: b := a[1] = a[2] end
if b = s↑4 then go to 5 else s1 end;
```

56 Kirjeldada funktsioon, mis jada $a_1, a_2, \ldots, a_n$ ja selle pikkuse n järgi leiab

a) suuruse $\sqrt{a_1^2 + a_2^2 + \ldots + a_n^2}$.

b) suuruse $\sqrt{a_m^2 + a_{m+1}^2 + \ldots + a_n^2}$, kus $m < n$ tuleb täiendavalt ette anda.

c) jada suurima ning vähima elemendi vahe.

d) positiivsete elementide arvu selles jadas.

e) märgimuutude arvu selles jadas.

57 Kirjeldada täisarvuline funktsioon, mis leiab

a) kahe antud naturaalarvu suurima ühisteguri.

b) binoomkordaja $\binom{n}{m}$.

c) antud naturaalarvu faktoriaali.

d) antud jada suurima elemendi järjekorranumbri.

58 Kirjeldada loogiline funktsioon, mis kontrollib, kas

a) ühe antud avaldise väärtus jagub teise antud avaldise väärtusega.

b) antud jada elemendid on kasvavas järjekorras.

c) antud jadas leidub etteantavast väärtusest suuremaid elemente.

d) kaks samadimensionaalset vektorit on ortogonaalsed.

e) kolm antud suurust saavad olla kolmnurga külgepikkusteks.

f) kaks oma küljepikkustega antud kolmnurka on sarnased.

g) kas antud polünoom $a_n x^n + a_{n-1} x^{n-1} + \ldots + a_1 x + a_0$ jagub kaksliikmega $x - a$.

h) antud $m \times n$ maatriksil (a_{ij}) on olemas sadulpunkt, s.t. kas

$$\max_i \min_j a_{ij} = \min_j \max_i a_{ij} .$$

59 Kirjeldada funktsioon, mis antud $m \times n$ maatriksi järgi leiab

a) selle maatriksi suurima ja vähima elemendi vahe.

b) selle maatriksi ja antud vektori korrutamisel saadava vektori pikkuse.

c) selle veeru järjekorranumbri, kus elementide summa on kõige suurem.

d) selle maatriksi ja tema transponeeritud maatriksi korrutamisel saadava maatriksi diagonaalelementide summa.

60 Kirjeldada funktsioon, mis leiab kolmnurga pindala tema

a) kolme külje pikkuste järgi.

b) tippude ristkoordinaatide järgi.

61 Kirjeldada rekursiivne funktsioon, mis arvu π korduva liitmise või lahutamise teel teisendab antud reaalarvulise suuruse poollõiku $[0, \pi)$ kuuluvaks.

62 Kirjeldada protseduur, mis jada $a_1, a_2, \ldots, a_n$ ja selle pikkuse n järgi

a) normeerib selle jada (kui vektori).

b) moodustab uue jada $a_n, a_{n-1}, \ldots, a_1$.

c) leiab jada suurima, vähima, suurima negatiivse ja vähima positiivse elemendi (vajaliku elemendi puudumise korral annab nulli).

63 Kirjeldada protseduur, mis kahest ühepikkusest jadast $a_1, a_2, \ldots, a_n$ ja $b_1, b_2, \ldots, b_n$ moodustab

a) jada $a_1, b_1, a_2, b_2, \ldots, a_n, b_n$.

b) jada $c_1, c_2, \ldots, c_n$, kus $c_i = \max(a_i, b_i)$.

c) jada $a_1, b_n, a_2, b_{n-1}, \ldots, a_n, b_1$.

64 Kirjeldada protseduur, mis polünoomi $a_n x^n + a_{n-1}x^{n-1} + \ldots + a_1 x + a_0$ kordajate jada $a_0, a_1, \ldots, a_n$ järgi moodustab

a) selle polünoomi tuletise kordajate jada.

b) selle polünoomi jagamisel vahega $x - a$ tekkiva jagatispolünoomi kordajate jada.

65 Kirjeldada protseduur, mis $m \times n$ maatriksi (a_{ij}) järgi moodustab

a) transponeeritud maatriksi (a_{ji}).

b) maatriksi $(p - a_{ij})$, kus $p = \max\limits_{i,j} a_{ij}$.

66 Kirjeldada protseduur, mis antud ruutmaatriksi A järgi moodustab

a) maatriksi $E - A^2$, kus E on ühikmaatriks.

b) sümmeetrilise maatriksi B nii, et $\vec{x}'A\vec{x} = \vec{x}'B\vec{x}$ (s.t. muudab ruutvormi maatriksi sümmeetriliseks).

c) vektori, mille i-ndaks elemendiks on nende elementide A_{kj} summa, kus kas $k = i$ ja $j \geq i$ või $k \geq i$ ja $j = i$.

d) vektori, mille i-ndaks elemendiks on maatriksi i-nda rea maksimaalne element.

67 Vähemalt millised kirjeldused peavad leiduma blokis, mis sisaldab järgmisi operaatoreid?

a) `if a = b ⊃ c(i) then go to f[k] else g[k] := p[r(i), i];`

b) `52: go to if s ∨ r = 5 then a[k] else 51;`
`ji[i, ij + j] := i - ii(j)↑2; iji(j,ji);`

c) `if s ≤ t ⊃ u then go to if p(5) then pp[5]else p5;`
`if a[r] = b then a[r] := f(x) else g(x);`

d) `for s := if f(x,y) then 1 else t - i step 1 until t + i do if g(s) = A[y] then go to s1; r(y); z := 2.5;`

e) `x[i + 2, i - 2] := i↑k - k↑i + pp(i); p1p(i + 2, k); a1: go to if zz > z × z then a5 else p[a];`

f) for i := 1 step if r[i] then 1 else -1
until p[k] do f[i] := g(i + s(i)); go to if
a ⊃ b=c then d[5] else d5;

g) if a(b) then e := if f ∨ g = h then i(j) else k
else go to if m(n) then p[r(s)] else t;

h) for i := 1 step if c then k else 2 until k↑2 do
a[i] := b[i] + f(i); if 1 = k ⊃ viies[i]= b[k]
then go to teine;

i) if a[b] then c(d) else if e = f(g) then h else
go to k[i]; for x := y(z) step 1 until z
do p[x] := g(x);

j) for a := b(m) while c(m) do if f[m] then g(m, p[m]
) else go to h[m + n]; if p[i] = r(m, i) then q;

k) if a[r(g)] then m[g]:= p(u) = q(u, u) else su;
go to if t[i]∨ c(i) = d[i] then k[i] else ki;

l) if a(b) then c(d) else if e[f] = g ⊃ h then go to
if i[j] = k(1) then m[n(p)] else r else s;

m) go to if a(b) = c[d] then g[h] else i[j]; if k(m,
n) then p(r, s) else if t then u else w(f);

68 Esitada järgmine funktsioonikirjeldus võimalikult lihtsal kujul:

15

```
real procedure d(a, b, c); value a, b, c; real a, b, c;
begin real f, g, k, l, m; f := a↑2; g := b↑2; l := if
f = g then (f + g) × a else f × b + g × a; if l = f × b
+ g × a then k := 3 × l else k := 2 × l; if a × f = g ×
b then begin m := 2 × a × f; go to 2 end; m := a × f; m
:= b × g + m; 2: d := m + k end;
```

69 Koostada võimalikult lihtne blokk (või operaator), millega saab mistahes programmis asendada järgmise bloki.

a)
```
begin real procedure ruut(a, b, c); begin c := if
m ≤ n then m else n; x := A[c-a, c-b]; z := A[a,
b]; ruut := x↑2 + z↑2 end; integer i, j, k; real
x, y, z; k := 1; B := A[1, k]; y := 0.0; for i :=
1 step 1 until m do for j := 1 step 1 until n do
begin x := ruut(i, j, k); y := y + sqrt(x) end; B
:= B + z; z :=  ruut(i, j, k) + ruut(j, i, k) end;
```

b)
```
begin integer procedure pann(a,b); pann := a :=
b↑2; integer a, b, c; integer procedure pada(a,b,
c); pada := c(a,b) + a + b↑2; a := 0;
for a := a while a = a↑2 do begin a := a + 1; d
:= pada(b, d, pann) end; d := d↑2 end;
```

c)
```
begin integer procedure tühi(a,b,c); begin integer
a; a := b↑2; c := c + 1; tühi := a + b end;
integer i, j; for i := 1 step 1 until n do a :=
tühi(j, i, a); b := tühi(a, b, a) + tühi(a, b, a)
end;
```

d) begin integer i, j; switch p := a, b; real procedure pool(a, b); begin integer k; k := a + b; a := k - a; pool := A[k - b, a]/2.0; b := k - b end; j := 0; a: j := j + 1; i := j; b: i := i + 1; A[i, j] := pool(i, j) + pool(i, j); go to p[if i < n then 2 else if j < n then 1 else 1 + n] end;

e) begin integer i, j, k; real c; real procedure sum(a, b); begin b := m - a; sum := y + c end; y := 0. 0; c := a[m, 1]; k := m + n-(if m > n then m else m); for i := 2 step 1 until k do begin y := sum(i, j); c := a[j + 1, i]; if i + k ≥ m + n then go to b end; b: c := sum(j, i) end;

f) begin integer i; integer procedure indeks(a, b); begin b := b + 1; indeks := a - b + 2 end; y := 0.0; i := 1; b: y := y + a[i, indeks(n, i)]; if i ≤ m then begin if m ≤ n then go to b else go to if i > n then c else b end; c: i := indeks(m, i) end;

g) begin integer i, j, k; integer procedure summa(p, q); begin p := p + 1; q := 1 + m + n - p; b := a[q - n, q - m]; summa := p + q end; real b; y := 0.0; i := 0; k := m - n; for j := 0 while i + m ≤ n + (if k > 0 then k + n else k + m) do begin j := summa(i, j); y := y + b end; end;

70 Funktsioonikirjeldust kasutades koostada programm, mis arvutab funktsiooni

$$F(x,a) = \frac{1}{4}\left[f(x,a) + f(-x,a) + f(x,-a) + f(-x,-a)\right]$$

väärtuste tabeli, kus

$$f(x,a) = ax + \sum_{i=0}^{10} \frac{\sin(ix + a) + \cos(ix + a)}{(i + 1)^2},$$

a omandab väärtused 0,4; 0,6; ...; 1,6 ja x väärtused -0,5; -0,45; ...; 2,5.

71 Funktsioonikirjeldust kasutades koostada programm, mis arvutab funktsiooni

$$F(x) = f(x)\sin x + f(-x)\cos x + \sqrt{f^2(-x) + x^2}$$

väärtuste tabeli, kus

$$f(x) = 5x^2 + e^{x-2}$$

ja x omandab väärtused -2,2; -2,1; ...; 1,5.

72 Pööramata tähelepanu järgmise programmi sisule, märkida välja kõik selles esinevad süntaktilised vead ja kirjeldada nende iseloomu.

a) begin real a, f; a := 5; integer r, i;
integer array l, g[1:r]; real procedure
choh(a, f) value: c) value a; integer array f, a;

```
procedure c; for f := 1 step 1 until n do a[f] :=
r↑c + f end ohoh i: if 1 := a then go to ii else
else f := ohoh(1)array:(f, ohoh) end end;
```

b)
```
begin integer k, l; real x, y; inreal(x);
procedure funny(a, b, c, d); value a;
array b, c: procedure d;
b[i] := c[i ÷ i]× d; a := 1.5;
funny := a/b[i] end funny
l: for k := if x < 0 then 1 else 3 until 5 do
x := funny(k, x, y, funny)↑x end end;
```

c)
```
begin integer i, n; real array a; real s; real
procedure f(x,y); value x, y; real f, x, y; f :=
x↑2 + 2 × y↑2; s := 0; for i := 1 step 1 until
n do begin a[i] := f(i, i × f(3, i)); s := s +
a[i] end end;
```

d)
```
begin integer n, j; real i, k, integer; real
array m; real procedure s(i, j); value i; integer
i; real array j; begin integer k; s := 0; for k
:= 1 step 1 until i do s := s + j[k]end; for j :=
1 step 1 until n do m[j] := s(j, m) end;
```

e)
```
begin integer i, n; real x, y, z; switch f := 1,2;
real procedure g(i); value i, real i; g := i↑2 -
3 × i↑3; real procedure h(ii); value ii; h := 5
+ i↑2; for i := 1 step 1 until n do begin y := 1
+ z(i); go to f[i - i ÷ 2]; print(i); 2: print(g(
y)/h(y)) end;
```

f) **begin real** rf, fr; **Boolean** f, g; **integer** i, ii;
real array s, p[1:5]; **switch** tuda := 3, ii, 5;
procedure nuda(r) **procedure**: f); **procedure** f;
if r > rf = fr **then** f; 3: i := 3.7; 5: nuda(i);
for i := 1 **step** 1 **until** 5, 0 **do** s[i] := p_i;
go to if f = rf **then** tuda **else** 3 **end**;

73 Arvestades seoseid [2]

$$B_1 = A_1 \supset \neg A_2\ ,\quad B_2 = (\neg A_3/A_4) \equiv \neg A_5\ ,$$

$$B_3 = A_2 \equiv \neg A_4\ ,\quad B_4 = A_3 \supset \neg A_1 \wedge A_5$$

koostada programm, mis loogiliste muutujate $A_1, \ldots, A_5$ kõigi tõeväärtuste korral arvutab järgmise loogilise avaldise tõeväärtuse:

a) $(B_1 \supset \neg B_4) \equiv \neg(B_3 \supset B_2)$.

b) $\neg(B_2 \vee (A_2 \supset B_4) \equiv B_3/(\neg B_1 \vee \neg A_5))$.

74 Koostada programm, mis leiab järgmise loogilise avaldise täieliku disjunktiivse normaalkuju ja täieliku konjunktiivse normaalkuju (sisuliselt tähendab see avaldise tõesus- ja vääruspiirkonna leidmist):

a) $A_1 \equiv (\neg A_2 \supset A_3 \vee \neg A_4)) \supset$
$((\neg A_3/(A_5 \wedge \neg(A_2 \vee A_6))) \supset A_4 \equiv \neg A_1)$

---

[2] Märk / tähistab siin Shefferi tehet: $X/Y = \neg(X \wedge Y)$.

b) $(A/\neg C) \supset \neg(B \equiv (\neg A \vee D)) \equiv$

$(((C \wedge \neg D)/(A \vee B)) \supset \neg B)$

75 Koostada programm, mis selgitab, kas loogilisest avaldisest (valemist)

$$(A \wedge B \supset C) \wedge (C \wedge D \supset \neg E) \wedge (\neg J \supset D \wedge E)$$

järeldub avaldis (valem)

a) $B \wedge \neg J \supset \neg A$.

b) $\neg C \supset \neg E \wedge \neg J$.

c) $B \vee C \supset D$.

d) $(B \supset \neg A) \vee (\neg J \supset \neg C)$.

76 Koostada programm, mis kuupäeva järgi leiab vastava nädalapäeva (sama aasta esimese jaanuari nädalapäeva teadmisel).

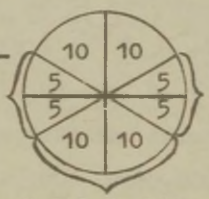
77 Koostada programm positsioonilises kümnendsüsteemis antud naturaalarvude 1-3999 esitamiseks rooma numbrite süsteemis. Rooma numbrimärke I, V, X, L, C, D, M võib sealjuures kodeerida vastavalt arvudega 1, 5, 10, 50, 100, 500 ja 1000. Sellist moodust kasutades tuleb näiteks arv MDCCCXCIV = 1894 kodeerida kujul 1000, 500, 100, 100, 100, 10, 100, 1, 5.

78 **Koostada programm rooma numbrite süsteemis antud arvude esitamiseks positsioonilises kümnendsüsteemis. Soovi korral kasutada rooma numbrimärkide kodeerimiseks ülesandes 77 nimetatud moodust.**

79 **Koostada programm positsioonilises kümnendsüsteemis antud naturaalarvude 0 - 999 999 esitamiseks eesti keeles.**

Vajalikke arvsõnu: null, üks, kaks, ..., üheksa, kümme, üksteist, ..., üheksateist, kakskümmend, ..., üheksakümmend, sada, kakssada, ..., üheksasada ja tuhat võib kodeerida vastavalt arvudega 0, 1, 2, ..., 9, 10, 11, ..., 19, 20, ..., 90, 100, 200, ..., 900 ja 1000. Nii näiteks on arvu 57819 eestikeelne esitus sellise kodeerimisviisi korral 50, 7, 1000, 800, 19.

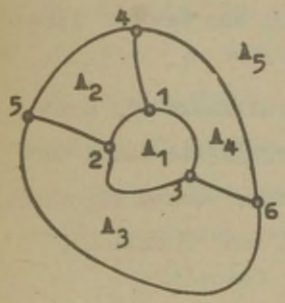
80 Koostada programm, mis teisendab ametlikes dokumentides kasutatava kellaaja (10.52, 19.17 jne.) tavalises kõnepruugis esitatud kellaajaks. Vajalikke termineid (veerand, pool, kolmveerand, minut, minutit, puudub, läbi, enne, pärast, lõunat) ja arvsõnu võib kodeerida arvudega. Täistunni osi väljendada joonisel 11 näidatud viisil.



Joonis 11.

81 Koostada etteantud normaalkaardi kõikide kahevärvilahendite leidmise programm.

Normaalkaardiks nimetatakse topoloogilist kaarti [3], kus igast sõlmpunktist väljub täpselt kolm serva. Normaalkaardi kahevärvilahendiks nimetatakse kaardi piirkondade värvimist kahe värviga nii, et iga piirkond on värvitud mingi ühe värviga ning mistahes sõlmpunkti kolm naaberpiirkonda pole kõik ühte värvi. Joonisel 12 on sõlmpunktid tähistatud numbritega 1, ..., 6, piirkonnad aga sümbolitega A_1, ..., A_6. Tähistades värve arvudega 0 ja 1 osutub selle kaardi üheks kahevärvilahendiks näiteks järgmine värvimine: $A_1 = 1$, $A_2 = A_3 = A_4 = 0$, $A_5 = 1$.



Joonis 12

Et värvilahendi värvide ümbervahetamine annab jälle värvilahendi, võib programm piirduda ainult poolte lahendite leidmisega.

82 Koostada programm, mis etteantud n korral leiab kõik algarvud $a \leq n$.

---

[3] Topoloogiliseks kaardiks nimetatakse tasandi jaotust (pidevate kõverate abil) lõplikuks arvuks piirkondadeks, kusjuures üks piirkond (joonisel 12 piirkond A_5) on tõkestamata.

83 Koostada programm, mis leiab etteantud täisarvude jada $a_1, a_2, \ldots, a_n$ iga paari suurima ühisteguri (kasutada Euklei dese algoritmi).

84 Koostada programm, mis etteantud n korral teeb iga arvu a $(2 \leq a \leq n)$ puhul kindlaks, kas see on täiuslik, küllane või puudulik.

Arvu a nimetatakse täiuslikuks, küllaseks või puudulikuks selle järgi, kas tema kõigi pärisjagajate summa (arv a ise välja arvatud) on vastavalt võrdne, suurem või väiksem arvust a (näit. arv 6 on täiuslik, arv 12 küllane, arv 4 aga puudulik).

85 Koostada programm, mis tähestiku

$A = \{ 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, m, *, \triangle, \nabla, (,) \}$

sõnade puhul teeb kindlaks, kas on tegemist valemiga või mitte, kusjuures valem defineeritakse järgmiselt:

1. 0, 1 ja 2 on konstandid;
2. kui γ on täisarv (lõplik sõna numbritest), siis $m\gamma$ on muutuja;
3. kõik konstandid ja muutujad on valemid;
4. kui α on valem, siis $*\alpha$ on valem;
5. kui α ja β on valemid, siis $(\alpha \triangle \beta)$ ja $(\alpha \nabla \beta)$ on valemid;
6. kui α on valem, siis (α) on valem.

Lihtsuse huvides võib sümbolid kodeerida arvudena.

86 Koostada tõlkimisalgoritm formaliseeritud keelest A keelde B.

Keel A defineeritakse järgmiselt:

1. "Abari" ja "Amadi" on nimisõnad;
2. "Adinir" on sihitu verb;
3. "Abikik" on sihiline verb;
4. "Ak" on sidesõna;
5. ⟨nimisõna⟩ ⟨sihitu verb⟩ on lihtlause, kusjuures nimisõna on aluseks;
6. ⟨nimisõna⟩ ⟨sihiline verb⟩ ⟨nimisõna⟩ on lihtlause, kusjuures esimene nimisõna on alus, teine sihitis;
7. ⟨lihtlause⟩ ⟨sidesõna⟩ ⟨lihtlause⟩ on liitlause.

Keel B defineeritakse järgmiselt:

1. "Begaso", "Bugaro", "Bemado" ja "Bumaro" on nimisõnad;
2. "Bedinor" ja "Begunor" on sihitud verbid;
3. "Bezikor" ja "Begukor" on sihilised verbid;
4. "Bel" ja "Beg" on sidesõnad;
5. ⟨sihitu verb⟩ ⟨nimisõna⟩ on lihtlause, kusjuures nimisõna on aluseks;
6. ⟨sihiline verb⟩ ⟨nimisõna⟩ ⟨nimisõna⟩ on lihtlause, kusjuures esimene nimisõna on alus, teine sihitis;
7. ⟨sidesõna⟩ ⟨lihtlause⟩ ⟨lihtlause⟩ on liitlause.

Keelte A ja B sõnade vastavus antakse järgmise tabeliga:

Keel A	Tõlkevaste määravad tingimused	Keel B
Abari	lihtlauses koos sõnaga "Adinir"	Begaso
	vastupidisel juhul	Bugaro
Amadi	lihtlauses koos sõnaga "Abari"	Bemado
	vastupidisel juhul	Bumaro
Adinir	lihtlauses koos sõnaga "Abari"	Bedinor
	vastupidisel juhul	Begunor
Abikik	lihtlauses koos "Amadi" kui alusega	Bezikor
	vastupidisel juhul	Begukor
Ak	esimeses lihtlauses on sihiline verb	Bel
	vastupidisel juhul	Beg

Lihtsuse huvides võib keeltes A ja B esinevad sõnad kodeerida arvudena.

87 Koostada programm, mis etteantud täisarvuliste liikmetega jada $x_1, x_2, \ldots, x_n$ järjestab ümber mittekahanevaks jadaks ja moodustab vastava variatsioonrea, s.t. leiab jada erinevad liikmed (kasvavas järjekorras) ning nende sagedused.

88 Koostada programm, mis etteantud variatsioonrea korral leiab keskmised

$$\bar{x}_k = \sqrt[k]{\frac{\sum x_i^k f_i}{\sum f_i}},$$

kus x_i tähistab rea üksikväärtust ja f_i vastavat sagedust; parameeter k muutub piirides $k = 1, 2, \ldots, m$.

89 Koostada programm, mis etteantud jada $x_1, x_2, \ldots$ $\ldots, x_n$ põhjal koostab nn. intervallitud variatsioonrea, s.t. väljastab (etteantud) intervallide piirid ühes vastavasse intervalli kuuluvate jada liikmete arvuga.

90 Koostada programm, mis etteantud variatsioonrea korral leiab mediaani

$$Me = x_{\langle me\rangle} + h\,\frac{\frac{1}{2}\sum f_i - f_{\langle s\rangle}}{f_{\langle me\rangle}},$$

kus $x_{\langle me\rangle}$ on nn. mediaanintervalli alumine piir, $f_{\langle me\rangle}$ - mediaanintervalli sagedus, $f_{\langle s\rangle}$ - mediaanintervallile eelnevate intervallide sageduste summa ja h - mediaanintervalli pikkus. Mediaanintervall määratakse võrratustega $f_{\langle s\rangle} < \frac{1}{2}\sum f_i$ ja $f_{\langle s\rangle} + f_{\langle me\rangle} \geq \frac{1}{2}\sum f_i$.

91 Koostada programm, mis etteantud variatsioonrea korral leiab moodi

$$Mo = x_{\langle mo\rangle} + h\,\frac{f_{\langle mo\rangle} - f_{\langle mo\rangle-1}}{(f_{\langle mo\rangle} - f_{\langle mo\rangle-1}) + (f_{\langle mo\rangle} - f_{\langle mo\rangle+1})},$$

kus $x_{\langle mo\rangle}$ on nn. moodintervalli alumine piir, $f_{\langle mo\rangle}$ - moodintervalli sagedus, $f_{\langle mo\rangle-1}$ ja $f_{\langle mo\rangle+1}$ tähistavad moodintervallile eelneva (resp. järgneva) intervalli sagedust, h tähistab moodintervalli pikkust. Moodintervall määratakse kui kõige suurema sagedusega intervall.

92 Koostada ühefaktorilise dispersioonanalüüsi programm. Lähteandmed antakse maatriksiga

$$X = \begin{pmatrix} x_{11} & \cdots & x_{1n} \\ - & - & - \\ x_{m1} & \cdots & x_{mn} \end{pmatrix},$$

kus x_{ij} tähistab teatava suuruse j-ndat mõõtmistulemust uuritava faktori i-nda väärtuse korral. Arvutada tuleb suurus

$$F = \frac{\frac{1}{m-1} \cdot n \sum_i (\bar{x}^{(i)} - \bar{x})^2}{\frac{1}{m(n-1)} \sum_i \sum_j (x_{ij} - \bar{x}^{(i)})^2},$$

kus $\bar{x} = \frac{1}{mn} \sum_i \sum_j x_{ij}$ ja iga $i = 1, 2, \ldots, m$ korral $\bar{x}^{(i)} = \frac{1}{n} \sum_j x_{ij}$.

93 Koostada programm, mis etteantud maatriksi

$$X = \begin{pmatrix} x_{11} & \cdots & x_{1n} \\ - & - & - \\ x_{m1} & \cdots & x_{mn} \end{pmatrix}$$

järgi leiab iga kahe rea korrelatsioonikordaja

$$\rho_{ik} = \frac{\sum_j (x_{ij} - \bar{x}_i)(x_{kj} - \bar{x}_k)}{\sqrt{\sum_j (x_{ij} - \bar{x}_i)^2 \cdot \sum_j (x_{kj} - \bar{x}_k)^2}},$$

kus $\bar{x}_i$ ja $\bar{x}_k$ tähistavad vastavate ridade aritmeetilisi keskmisi.

94 Koostada programm, mis suuruste x ja y eriväärtuste esinemissageduste tabeli

	y_1	...	y_j	...	y_l
n_1	n_{11}	...	n_{1j}	...	n_{1l}
...	...	...	...	...	...
x_i	n_{i1}	...	n_{ij}	...	n_{il}
...	...	...	...	...	...
x_k	n_{k1}	...	n_{kj}	...	n_{kl}

(n_{ij} tähendab eriväärtuste paari x_i, y_j esinemissagedust) järgi arvutab korrelatsioonikordaja

$$\rho = \frac{\sum_i \sum_j n_{ij}(x_i - \bar{x})(y_j - \bar{y})}{\sqrt{\sum_i n_i(x_i - \bar{x})^2 \cdot \sum_j n'_j(y_j - \bar{y})^2}},$$

kus $\bar{x}$ ja $\bar{y}$ on nende suuruste eriväärtuste aritmeetilised keskmised,

$$n_i = \sum_j n_{ij} \quad \text{ning} \quad n'_j = \sum_i n_{ij}.$$

95 Koostada programm teoreetilise sagedusjaotuse arvutamiseks ja selle võrdlemiseks empiirilise jaotusega χ^2-kriteeriumi alusel.

Algandmeteks on empiirilise jaotuse intervallide piirid

$$u_0, u_1, u_2, \ldots, u_n \quad (u_0 < u_1 < \ldots < u_n)$$

ja vastavad sagedused

$$f_1, f_2, \ldots, f_n .$$

Nende alusel leiame iga $i = 1, 2, \ldots, n$ korral teoreetilised normaaljaotuse sagedused

$$f_i^{(t)} = \frac{N}{\sigma\sqrt{2\pi}} \exp\left(-\frac{(x_i - \bar{x})^2}{2\sigma^2}\right) \cdot (u_i - u_{i-1})$$

ning suuruse

$$\chi^2 = \sum_i \frac{(f_i - f_i^{(t)})^2}{f_i^{(t)}} ,$$

kus

$$N = \sum f_i, \quad x_i = \frac{1}{2}(u_i + u_{i-1}),$$

$$\bar{x} = \frac{1}{N}\sum f_i x_i, \quad \sigma = \sqrt{\frac{1}{N}\sum f_i (x_i - \bar{x})^2} .$$

Ülesannet võib veel täiendada vastava χ^2 väärtuse tõenäosuse $P(\chi^2)$ leidmisega. Sel juhul tuleb algandmete hulka täiendada sobiva osaga (vastavalt vabadusastmete arvule $K = n - 2$) $P(\chi^2)$ väärtuste tabelist. Tõenäosuse täpsemaks arvutamiseks kasutada lineaarset interpolatsiooni.

96 Koostada programm kahe tundmatuga lineaarsete võrrandsüsteemide

$$a_i x + b_i y = c_i$$
$$a_i' x + b_i' y = c_i'$$

lahendamiseks $(i = 1, 2, \ldots, n)$.

97 Koostada programm ruutvõrrandite

$$a_i x^2 + b_i x + c_i = 0$$

lahendamiseks, kusjuures võib eeldada, et iga $i = 1, \ldots, n$ korral $a_i \neq 0$.

98 Koostada programm lineaarse võrrandisüsteemi

$$\begin{aligned}
a_{11}x_1 &= b_1 \\
a_{21}x_1 + a_{22}x_2 &= b_2 \\
a_{31}x_1 + a_{32}x_2 + a_{33}x_3 &= b_3 \\
&- - - - - - - - - - - - - - - \\
a_{n1}x_1 + a_{n2}x_2 + \ldots + a_{nn}x_n &= b_n
\end{aligned}$$

lahendamiseks.

99 Koostada programm võrrandi $f(x) = 0$ lahendamiseks lõigu poolitamise meetodil, eeldades, et võrrandil $f(x) = 0$ on etteantud lõigul $[a,b]$ üks lahend ja et $f(x)$ on lõigul $[a,b]$ pidev.

Meetod seisneb lähtelõigu $[a,b]$ ja protsessi käigus moodustatud osalõikude $[a_i, b_i]$ poolitamises, kusjuures poolitatud osalõikudest valitakse välja see pool, milles asub lahend ($f(x)$ väärtused otspunktides vastandmärgilised). Lõikude poolitamine toimub seni, kuni jõutakse osalõiguni $[a_j, b_j]$ pikkusega $b_j - a_j < 2\varepsilon$, kus ε on (ligikaudse) lahendi x^* lubatav viga. Lahendiks võetakse $x^* = \frac{1}{2}(a_j + b_j)$.

Programmis konkretiseerida funktsiooni $f(x)$ kuju.

100 Koostada programm integraali abil defineeritud funktsiooni

$$F(a,b) = \int_0^{\frac{\pi}{2}} \sin^{a-1}\varphi \cos^{b-1}\varphi \, d\varphi$$

väärtuste tabeli koostamiseks, kus a = 1,0; 1,1; ...; 1,5 ja b = 1,0; 1,1; ...; 1,5. Integraali arvutamisel kasutada

a) trapetsivalemit.

b) Simpsoni valemit.

101 Koostada programm algtingimusega diferentsiaalvõrrandi

$$\begin{cases} y' = f(x,y) \\ y(x_0) = y_0 \end{cases}$$

lahendamiseks Runge-Kutta neljandat järku meetodiga, s.t. kasutades valemeid

$$k_1 = hf(x_i, y_i),$$

$$k_2 = hf(x_i + \frac{h}{2}, y_i + \frac{k_1}{2}),$$

$$k_3 = hf(x_i + \frac{h}{2}, y_i + \frac{k_2}{2}),$$

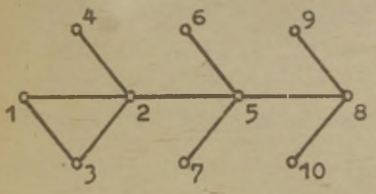
$$k_4 = hf(x_i + h, y_i + k_3),$$

$$k = \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4),$$

$$y_{i+1} = y_i + k,$$

kus i = 0, 1, ..., n. Programmi koostamisel tuleb konkretiseerida protseduurina kirjeldatava funktsiooni f(x,y) kuju.

102 Koostada programm, mis etteantud labürindi (graafi) ja etteantud kahe tipu korral leiab tee esimesest tipust teise ning sellise tee mitteolemasolu korral väljastab vastava tunnuse. Labürinti (vt. joon. 13) võib esitada kaarte



Joonis 13.

1	2	3	4	5
6	7	8	9	10
11	12	13	14	15
16	17	18	19	20
21	22	23	24	25

Joonis 14.

(naabertippude paaride) abil, näit. kujul (1,2), (1,3), (2,3) jne. või vastava maatriksi $A = (a_{ij})$ abil, kus

$$a_{ij} = \begin{cases} 1, & \text{kui tipud } i \text{ ja } j \text{ on naabertipud} \\ 0, & \text{vastupidisel juhul.} \end{cases}$$

Erijuhul võib labürinte esitada ka näiteks joonisel 14 näidatud viisil, kus üleminekud on lubatud nende naaberruutude vahel, kus puudub tõke (jäme joon). Algandmetes tuleb fikseerida kas lubatud üleminekud või tõkked.

Programmi võib koostada põhimõttel, et mingist tipust lähtuva mitme kaare korral valitakse alati näit. kõige väiksema järjekorranumbriga tipp. Ummiktee korral tuleb teha tagasikäik, kusjuures vastav kaar tähistatakse nii, et seda enam ei läbita. Sooritada ei tule ka käiku, mis viib juba läbitud tippu. Sel juhul tuleb talitada samuti nagu ummiktee korral.

103 Koostada programm, mis leiab etteantud graafi iga kahe tipu vahelised kaugused. Andmed graafi kohta esitatakse ruutmaatriksi $A = (a_{ij})$ kujul, kus a_{ij} $(i,j = 1, 2, \ldots, n)$ tähendab naabertippude i ja j vahelist kaugust. Kui aga i ja j pole naabertipud, siis $a_{ij} = a_{ji} = M$ (suvaline küllalt suur arv). Tippude vahelised kaugused annab maatriks $B^{(n)} = (b_{ij}^{(n)})$, mis leitakse seoseist

$$b_{ij}^{(p)} = \min_{k}(b_{ik}^{(p-1)} + a_{kj}) \quad (p = 1, 2, \ldots, n),$$

$$b_{ij}^{(o)} = a_{ij} .$$

104 Koostada programm mingi ühe ressursi optimaalse jaotamise ülesande lahendamiseks. Antud on sealjuures maatriks $A = (a_{ij})$, kus a_{ij} $(i = 1, 2, \ldots, m;\ j = 0, 1, \ldots, n)$ tähendab efekti, mis saadakse selle ressursi j ühiku suunamisel objektile i.

Ressursi optimaalse (suurimat efekti andva) jaotuse määramiseks leiame iga $k = 0, 1, \ldots, n$ korral suurused

$$b_k^{(p)} = \max_{r+s=k}(b_r^{(p-1)} + a_{p+1,s}),$$

kus $p = 1, 2, \ldots, m-1$ ja

$$b_k^{(o)} = a_{1k} .$$

Ühtlasi tuleb iga $b_k^{(p)}$ leidmisel fikseerida indeksi s see väärtus s_{kp}, mille korral maksimum saavutatakse.

Ressursi koguvaru v korral ($0 < v \leq n$) annab maksimaalse koguefekti suurus $b_v^{(m-1)}$. Ressursi optimaalse jaotuse saame fikseeritud indeksite s_{kp} abil: suurusele $b_v^{(m-1)}$ vastav $s_{v,m-1}$ näitab, mitu ühikut tuleb suunata objektile m; suurusele $b_{v-s_{v,m-1}}^{(m-2)}$ vastav s näitab, mitu ühikut suunata objektile m - 1 jne.

Programm peab väljastama maksimaalse koguefekti ja optimaalse jaotuse iga võimaliku varu $v = 1, 2, \ldots, n$ korral.

105 Koostada programm tingrefleksi kujundamise ja kustutamise protsessi modelleerimiseks.

Tingrefleksi modelleerimisel on meil tegemist järgmiste suurustega: A (tingimatu ärritaja), B (tingitud ärritaja), R (reaktsioon), c (tingrefleksi väljakujundatust iseloomustav suurus), c_0 (tingrefleksi väljakujundatuse läviväärtus), η (vahemikus $(-\gamma, \gamma)$ ühtlaselt jaotatud juhuslik suurus), δ (positiivne suurus, mille võrra c suureneb igal positiivsel ja väheneb igal negatiivsel üksikkatsel). Nende suuruste vahelised seosed ja c muutumine üksikkatsetel on antud järgmise tabeliga:

Ärritajad	Suuruse c muutumine	Reaktsioon
A = 1, B = 1	$c' = c + \eta + \delta$	R = 1
A = 1, B = 0	$c' = c + \eta$	R = 1
A = 0, B = 1	$c' = c + \eta - \delta$	$R = \begin{cases} 1, & \text{kui } c' \geq c_0 \\ 0, & \text{kui } c' < c_0 \end{cases}$
A = 0, B = 0	$c' = c + \eta$	R = 0

kus A, B ja R võrdumine ühega tähendab (vastavalt ärrituse või reaktsiooni) olemasolu, nulliga võrdumine aga puudumist.

Programmis võib eeldada vahemikus (-x, x) ühtlaselt jaotatud juhuslikke suurusi genereeriva standardse funktsiooni Juh(x) olemasolu. Katseseeria A_1, ..., A_n, B_1,... ..., B_n antakse igakordsel modelleerimisel ette.

106 Koostada programm fikseeritud mängimisviiside võrdlemiseks järgmises mängus.

Mäng toimub kahe poole X ja Y vahel. Tähistame nende poolte "löögivõimsusi" $x_1^{(t)}$ ja $y_1^{(t)}$, "kaitstavaid võimsusi" vastavalt $x_2^{(t)}$ ja $y_2^{(t)}$. Igal sammul t ($1 \le t \le n$) antakse mängija X võimsuste jaotus ehk löök $u_1(t)$, $u_2(t)$ ette algandmetega (sealjuures $u_1(t)$, $u_2(t) \ge 0$, $u_1(t) + u_2(t) \le 1$), mängija Y löök $v_1(t)$, $v_2(t)$ (kus samuti $v_1(t)$, $v_2(t) \ge 0$ ja $v_1(t) + v_2(t) \le 1$) arvutatakse aga programmiga, näit. võrdeliselt vastaspoole objektide võimsustega. Võimsuste muutumine ülemineku järgmisele sammule määratakse valemitega ($i = 1, 2$; $t = 1, 2, \ldots, n$):

$$x_i(t) = x_i(t-1) - q_i y_i(t) v_i(t),$$
$$y_i(t) = y_i(t-1) - p_i x_i(t) u_i(t),$$

kus kordajad p_1, p_2, q_1, $q_2 > 0$ ning algväärtused $x_1(0)$, $x_2(0)$, $y_1(0)$ ja $y_2(0)$ on ette antud. Mängija X mängimisviisi efektiivsust hinnatakse avaldisega

$$K = x_2(n) - y_2(n).$$

Programm peab m ($m > 1$) konkreetse mängimisviisi hulgast leidma parima (kus K on maksimaalne), väljastama maksimaalse K ja vastava mängukäigu.

107 Koostada programm, mis domineeritud strateegiate kõrvaldamise teel vähendab maatriksmängu dimensioone.

Kahe isiku (A ja B) nullsummamäng antakse ette maatriksiga

$$\begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ - & - & - & - \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{pmatrix},$$

kus a_{ij} tähendab mängija A võitu juhul kui A kasutab oma strateegiat i ning B oma strateegiat j. Mängija A strateegia k on domineeritud strateegia i poolt, kui iga $j = 1, \ldots, n$ korral $a_{kj} \leq a_{ij}$. Mängija B strateegia l on domineeritud strateegia j poolt, kui iga $i = 1, \ldots, m$ korral $a_{il} \geq a_{ij}$.

Programm peab moodustama mängu taandatud maatriksi ning kummagi mängija jaoks vektori, mille elemendid näitavad taandatud maatriksi strateegiate järjekorranumbreid lähtemaatriksis.

K i r j a n d u s

1. П. Наур (ред.). Алгоритмический язык АЛГОЛ-60. Пересмотренное сообщение. М., 1965.
2. М.И. Агеев. Основы алгоритмического языка АЛГОЛ-60. М., 1965.
3. Ü. Kaasik. Arvutid ja programmeerimine II. Tartu, 1967.
4. С. Гинзбург. Математическая теория контекстно-свободных языков. М., 1970.
5. А.В. Гладкий, И.А. Мельчук. Элементы математической лингвистики. М., 1969.
6. А.М. Вухтияров, Л.М. Зикевская, Г.Д. Фролов. Сборник задач по программированию. М., 1970.
7. Ü. Kaasik, A. Korjus, I. Kull. Programmeerimine. Tallinn, 1971.
8. R. Petroviča, L. Ulmanis. ALGOL-60 palīdz programmēt. Rīga, 1968.
9. U. Mereste. Ülesandeid statistika praktikumiks. Tartu, 1971.
10. M. Levin, S. Ulm. Arvutusmeetodite käsiraamat. Tallinn, 1966.
11. Г. Греневский. Кибернетика без математики. М., 1964.
12. А.Л. Брудно. АЛГОЛ. М., 1968.
13. L. Võhandu. Õpime programmeerima. "Horisont" 1-6, 1969.

S i s u k o r d

Э.Кавсик, И. Кулль
ПРОГРАММИРОВАНИЕ НА АЛГОЛе
На эстонском языке
Тартуский государственный университет
ЭССР, г. Тарту, ул. Пикооли, 18
Vastutav toimetaja M. Koit
==========
TRÜ rotaprint 1971. Paljundamisele antud 28.XII 1971. Trükipoognaid 8,62. Tingtrükipoognaid 8,02. Arvestuspoognaid 6,27. Trükiarv 1000. Paber 3ox42. 1/4. MB 07672.
Tell. nr. 1121.
Hind 45 kop.

Hind 45 kop.