TARTU ÜLIKOOL
Arvutuskeskus

VIRTUAALSÜSTEEMI PROTSEDUURIDEKEELED

Programme kõigile

Koostanud A.Jaeger
Ü.Kaasik
E.Saks

TARTU 1989

Kinnitatud matemaatikateaduskonna nõukogus

31. augustil 1989. a.

Käesolev väljaanne kujutab endast metoodilist juhendmaterjali nendele, kes ühtsusseeria arvutite virtuaalsüsteemis soovivad luua ja kasutada protseduure. Järgnev käsitlus toetub sarjas "Programme kõigile" juba ilmunud väljaannetele "Virtuaalarvutid" (1985), "Virtuaalarvutite kasutamine" (1987) ja "Virtuaalsüsteem ühtsusseeria arvuteil" (1989), milledes avaldatud materjalide tundmist lugejailt enamasti ka eeldatakse.

ЯЗЫКИ ПРОЦЕДУР ВИРТУАЛЬНОЙ СИСТЕМЫ.
Программы для всех.
Составители Андрес Я э г е р, Юло К а а з и к, Энно С а к с.

На эстонском языке.
Тартуский университет.
ЭССР, 202400, г. Тарту, ул. Юликооли, 18.
Vastutav toimetaja J. Tapfer.
Paljundamisele antud
Formaat 60x84/16.
Rotaatoripaber.
Masinakiri. Rotaprint.
Tingtrükipoognaid 5,58.
Arvestuspoognaid 4,43. Trükipoognaid 6,0.
Trükiarv 300.
Tell. nr. 642.
Hind 15 kop.
TÜ trükikoda. ENSV, 202400 Tartu, Tiigi t. 78.

1. P R O T $ E D U U R I D E K E E L E X E C 2

1.1. Sissejuhatus

Teatavasti tuleb iga vähegi keerulisema ülesande lahendamiseks arvutil üldiselt sooritada kindlas järjekorras terve hulk tegevusi. Selliseid tegevuste järjendeid nimetatakse sageli protseduurideks. Protseduuris sisalduvaid tegevusi võib näiteks virtuaalsüsteemi kasutades täita ka vahetult terminaalilt antavate CP- ja/või PTS-käskude (-alamkäskude) ning vajalike andmete sisestamise teel, kuid protseduuri korduval täitmisel muutub see tülikaks. Sellepärast ongi virtuaalsüsteemis protseduuride esitlevaks kirjeldamiseks olemas spetsiaalsed nn. protseduuridekeeled, nende täitmiseks aga vastavad interpretaatorid ehk protseduuritõõtlejad. Protseduurid säilitatakse lähtekeelsena EXEC-tüüpi PTS-failides ning interpreteeritakse protseduuride igakordsel täitmisel uuesti.

Virtuaalsüsteemis on kasutusel kokku kolm protseduuridekeelt (EXEC, EXEC2 ja REXX) koos vastavate interpretaatoritega. Nendest enamkasutatavad on kaks viimasenanimetatut, mis leiavadki käsitlemist käesolevas väljaandes. Esimeses peatükis võtame vaatlusele piisavaid võimalusi pakkuva, kuid suhteliselt lihtsama protseduuridekeele EXEC2.

1.2. Keele elemendid

Keeles EXEC2 kirjutatud protseduur (ehk lihtsalt "protseduur") kujutab endast teatavate põhisümbolite järjendit. Põhisümbolid jagatakse harilikult kolme rühma (tähed, numbrid ja piirajad) ning nad on koondatud järgmisse tabelisse:

Tähed	Numbrid	Piirajad
A B C D E F G H I J	0 1 2 3	+ - / = ' & ?
K L M N O P Q R S T	4 5 6 7	. : % ! ¬ > <
U V W X Y Z Ц # @	8 9	* ( ) _ ␣

(peale nende tohib aga tekstikonstantides ja kommentaarides kasutada veel ka koodi ДКОИ teisi sümboleid).

Protseduuri loogilisi kirjeid nimetame ridadeks: protseduur on ridade järjend. Reaks võib sealjuures olla kas CP- või PTS-käsk, protseduuridekeele direktiiv, korrektori (või mõne muu alamkeskkonna) alamkäsk, kommentaar (tärniga algav rida) või andmed. Protseduurirea maksimaalpikkus on 255 sümbolit. Rida võib märgendada kuni seitsmesümbolilise nimega (identifikaatoriga), mille ette tuleb kirjutada miinusmärk.

Enamasti moodustatakse protseduurid PTS-failidena virtuaalarvuti A-kettale korrektoriga XEDIT, kuid neid võib sisestada ka kaartsisendilt. Protseduur käivitatakse PTS-käsuga EXEC või lihtsalt protseduurinime (s.t. vastava PTS-faili nime) alusel. Näiteks protseduuri PROC1 käivitamiseks ja parameetrite PAR1, PAR2, PAR3 etteandmiseks sisestatakse kas

EXEC PROC1 PAR1 PAR2 PAR3 või PROC1 PAR1 PAR2 PAR3

Arvudena võib protseduurides kasutada vaid täisarve vahemikust -2147483648 kuni +2147483647. Kõikide protseduuris kasutatavate muutujate nimed peavad algama märgiga &. Muutujanimedena võib kasutada näiteks järgmisi konstruktsioone:

&X &PIKKUS

&3.1415927 &UPPER_LIMIT

Protseduuris võib kasutada ka aritmeetilisi avaldisi: tühikute vahel olevate pluss- ja/või miinusmärkidega ühendatud täisarvmuutujate ja/või täisarvude järjendeid, näiteks

3 - 4 + -11 + &ARV

Aritmeetilise avaldise ainsaks või viimaseks ükskliikmeks tohib olla veel ka täisarvulise väärtusega funktsioon. Näiteks aritmeetilise avaldise

2 + &LENGTH OF &TEKST

väärtuseks on muutuja &TEKST väärtuseks omistatud sõne pikkus (sümbolite arv) pluss kaks (vt. lk. 8).

Protseduuride kirjutamisel on veel oluline teada, kuidas toimub protseduuriridade töötlemine interpreteerimisel. Iga rida kõigepealt skaneeritakse: eemaldatakse eest ja lõpust tühikud, vahepealseist jäetakse aga alles vaid üks. Nii saadakse tühikuteta sõnede ehk sõnade järjend, mille edasisel töötlemisel asendatakse (paremalt alates) sõnades leiduvad muutujad nende väärtustega. Saadud rida analüüsitakse süntaktiliselt ning vigade puudumisel täidetakse.

Protseduuridekeele EXEC2 põhikonstruktsioonideks on erimuutujad, standardfunktsioonid ning direktiivid. Sellises järjekorras neid järgnevates osades ka käsitletakse.

1.3. Erimuutujad

Erimuutujateks nimetatakse niisuguseid muutujaid, milledele väärtused omistatakse automaatselt vastavat protseduuri töötleva interpretaatori poolt.

Programmeerija võib erimuutujaid (s.t. nende väärtusi) kasutada oma protseduuris igal pool, kus ta seda vajalikuks peab. Peale selle on küll üldiselt võimalik enamiku erimuutujate väärtusi protseduuri täitmise käigus ka muuta, kuid sellega tuleb olla ettevaatlik.

Kõigepealt vaatleme selliseid erimuutujaid, mille väärtused annavad informatsiooni antud hetkel täidetava protseduuri kohta. Nii on erimuutuja &0 väärtuseks täidetava protseduuri nimi ehk täpsemalt EXEC2-interpretaatorile üle antud käsurea esimene sõna. Näiteks protseduuri XQ käivitamisel koos parameetri 3 etteandmisega kas käsuga

```
EXEC2 XQ 3
```

või siis vahetult nime

```
XQ 3
```

alusel saab erimuutuja &0 väärtuseks protseduurinime XQ.

Erimuutujate &FILENAME, &FILETYPE ja &FILEMODE väärtuseks on aga vastavalt täidetavat protseduuri sisaldava faili nimi, tüüp ja režiim. Seejuures erineb erimuutuja &0 väärtus erimuutuja &FILENAME väärtusest vaid sel juhul, kui protseduur käivitati sünonüümi abil.

Mitu erimuutujat väärtustatakse protseduurile ülekantavate suurustega. Nii saab erimuutuja &CMDSTRING väärtuseks vastava protseduuri käivitanud käsurea alates protseduurini-

mest. See sisaldab peale sõnade nii vahepealseid kui ka lõputühikuid. Erimuutuja &ARGSTRING seevastu aga väärtustatakse protseduurile ülekantava parameetritereaga (selles reas puudub äsjavaadeldud erimuutujaga &CMDSTRING võrreldes vaid esimene sõna - protseduurinimi). Parameetritereas esinevad üksikparameetrid (sõnad) omakorda omistatakse nende esinemise järjekorras erimuutujatele &1, &2, &3 jne. Ülekantud üksikparameetrite arvu saame aga teada erimuutuja &N (sel muutujal on ka nimi &INDEX) väärtusena.

Osa erimuutujaid annavad informatsiooni protseduuri täitmise hetkeseisu kohta. Erimuutuja &LINE (ka &LINENUM) väärtuseks on antud hetkel EXEC2-interpretaatori poolt täidetava (töödeldava) protseduurirea number, erimuutuja &FROM (ka &GOTO) väärtuseks aga selle protseduurirea number, millelt anti viimane direktiiv &GOTO (vt. lk. 20). Selle protseduurirea numbri, millelt käivitati antud momendil täidetav tarbijaalamprotseduur või -funktsioon, annab meile aga erimuutuja &LINK väärtus. Antud hetkeks kujunenud protseduuride sisalduvustaseme numbri saame teada erimuutuja &DEPTH väärtusena (põhiprotseduuri korral on selleks väärtuseks üks). Erimuutuja &COMLINE väärtus näitab, mitmendalt protseduurirealt anti viimane käsk või alamkäsk.

Lõpuks nimetame veel kaht erimuutujat &BLANK ja &RC (ka &RETCODE), milledest esimese väärtuseks on üks tühik, teise väärtuseks aga viimasena täidetud käsu (alamkäsu) või alamprotseduuri lõpukood. Erimuutuja &RC väärtuse kontrollimise teel saab otsustada eelmise käsu (alamkäsu) või alamprotseduuri täitmise edukuse üle.

1.4. Standardfunktsioonid

Protseduuridekeele EXEC2 standardfunktsioonid algavad kõik funktsiooni määrava võtmesõnaga, millele kindlasti peab järgnema sõna "OF". Alles seejärel näidatakse nõutavad argumendid ehk parameetrid, aga nimetatud komponendid tuleb üksteisest eraldada vähemalt ühe tühikuga. Standardfunktsiooni sisaldava protseduurirea töötlemine interpretaatori poolt toimub ikka nii, nagu see on kirjeldatud punktis 1.2.

Kõigepealt vaatleme sõnadega (tühikuteta sõnedega) manipuleerimise võimalusi. Sõna pikkuse (s.t. tema sümbolite arvu) teadasaamiseks võib kasutada funktsiooni üldkujuga

&LENGTH OF sõna

Sõna algusest või lõpust mingi arvu sümbolite eraldamist võimaldab funktsioon üldkujuga vastavalt kas

&LEFT OF sõna arv

või

&RIGHT OF sõna arv

Sõnast mistahes osa (ehk alamsõne) eraldava funktsiooni üldkuju on järgmine

$$\left\{\begin{matrix}\text{\&PIECE}\\ \text{\&SUBSTR}\end{matrix}\right\}\ \text{OF sõna arvi}\ \left[\begin{matrix}\text{arv2}\\ *\end{matrix}\right].$$

Siin arvi määrab eraldatava alamsõne esimese sümboli järjekorranumbri lähtesõnas (järjekorranumbrid algavad ühest), arv2 aga eraldatavate sümbolite arvu (tärni näitamisel ning vaikimisi sõna lõpuni).

Sõna tüübi (kas arv või mitte) teadasaamiseks on standardfunktsioon järgmise üldkujuga

{&DATATYPE / &TYPE} OF sõna

Kui sõna on tõlgendatav arvuna, siis funktsiooni väärtuseks saadakse sümbolühend "NUM", teksti korral aga "CHAR".

Sõnade mitmel viisil ühendamist saab organiseerida järgmiste standardfunktsioonide abil.

Etteantud sõnade konkatenatsiooni võimaldab moodustada standardfunktsioon üldkujuga

{&CONCATENATION / &CONCAT} OF sõna1 sõna2 ...

Sõnadest tekstirea moodustamiseks on olemas koguni kaks standardfunktsiooni üldkujudega

&LITERAL OF sõna1 sõna2 ...

ja

&STRING OF sõna1 sõna2 ...

Tekstirea moodustamisel säilitavad need standardfunktsioonid kõik eelnevad ja vahepealsed tühikud. Erinevus nende vahel seisneb selles, et esimene standardfunktsioon (&LITERAL OF) ei asenda sõnades esinevaid muutujaid nende väärtustega, teine (&STRING OF) aga küll.

Järgnevalt vaadeldavad standardfunktsioonid on ette nähtud fikseeritud sõna leidmiseks kas etteantavast sõnast või siis sõnade jadast.

Sõna esimese sisaldumise leidmine etteantud sõnest toimub standardfunktsiooniga, mille üldkuju on

&LOCATION OF sõna sõne

Funktsiooni väärtuseks on kas sõna esimese sümboli järjekorranumber teises argumendis (numbrid algavad ühest) või null.

Näidatud sõna järjekorranumbri (numbrid algavad ühest) sõnade jadas annab meile järgmise üldkujuga funktsioon

&POSITION OF sõna sõna1 sõna2 ...

Etteantud järjekorranumbriga (arv) sõna eraldab sõnade jadast aga standardfunktsioon, mille üldkuju on järgmine

&WORD OF sõna1 sõna2 ... arv

Teatavasti on vaadeldava protseduuridekeele aritmeetilistes avaldistes lubatud kasutada liitmis- ja lahutamistehteid. Peale nende on aga vastavaid standardfunktsioone kasutades võimalikud veel ka korrutamine ja täisarvuline jagamine (kus murdosa jäetakse lihtsalt ära).

Kahe või enama arvu korrutamise teeb võimalikuks järgmise üldkujuga standardfunktsioon

{&MULTIPLICATION / &MULT} OF arv1 arv2 [arv3 ...]

Täisarvulist jagamist võimaldava standardfunktsiooni üldkuju on aga järgmine

{&DIVISION / &DIV} OF jagatav jagaja

Lõpuks vaatleme kolme n. ö. üksikut standardfunktsiooni.

Lõputühikute eemaldamist muutuja väärtuseks omistatud sõnest võimaldab standardfunktsioon, mille üldkuju on

&TRIM OF sõne

Ühise prefiksiga ja vaid numbriliste sufiksite poolest erinevate sõnade (sellised on näiteks erimuutujate nimed &1, &2 jne.) järjendit võimaldab genereerida standardfunktsioon, mille üldkuju on

&RANGE OF sõna i j

Siin esimese parameetriga antakse ette ühine prefiks, kahe järgnevaga aga arvulise sufiksi alg- ja lõppväärtus (muutumise samm on ühs). Nii näiteks erimuutujate &1, &2, ..., &12 genereerimine ja muutujale &M omistamine toimub järgmiselt:

```
&M = &RANGE OF & 1 12
```

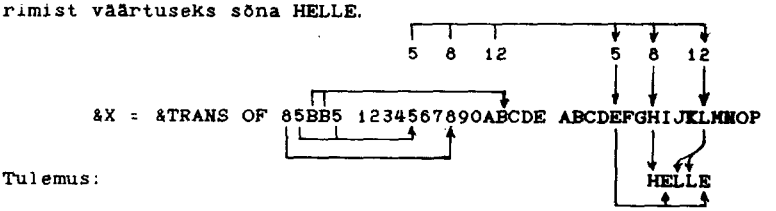
Muutuja &M väärtuseks saab siin järjend &1 &2 &3 ... &12 .

Antud sõna ümberkodeerimist võimaldava standardfunktsiooni üldkuju on järgmine

```
{&TRANSLATION}
{&TRANS      } OF sõna1 [sõna2 [sõna3]]
```

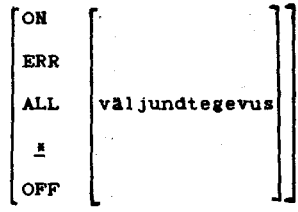
Selle standardfunktsiooni toimel modifitseeritakse esimesena näidatud sõna (sõna1) teise ja kolmanda argumendi (sõna2 ja sõna3) abil järgmisel viisil.

Esimese sõna sümboleid vaadeldakse järjest, kusjuures igaüht nendest võrreldakse sümbolitega teisest sõnast. Kui sümbol ei ühti mitte ühegagi teisest sõnast, siis jäetakse see sümbol esimeses sõnas muutmata. Kui aga sümbol esimesena ühtib teise sõna i-nda sümboliga, siis asendatakse ta i-nda sümboliga kolmandast sõnast. Kui kolmas sõna hoopis puudub või sisaldab i-st vähem sümboleid, siis vaadeldav esimese sõna sümbol asendatakse tühikuga. Tulemuse pikkus ühtib seega igal juhul esimese sõna pikkusega. Näiteks järgmise omistamisdirektiivi toimel omandab muutuja &X pärast ümberkodeerimist väärtuseks sõna HELLE.



1.5. Direktiivid

Interpretaatori töörežiimide kehtestamine. Iga keeles EXEC2 kirjutatud protseduuri esimene rida peab sisaldama direktiivi &TRACE. Selle direktiiviga saab kehtestada või tühistada nn. jälgimisrežiimi, mis võimaldab protseduuri täitmise kohta informatsiooni saamist. Direktiivi üldkuju on



Siin esimene operand määrab jälgimisrežiimi järgmiselt:

ON - jälgitakse kõiki protseduuris täidetavaid PTS- ja CP-käske ja alamkäske ning nullist erinevaid lõpukoode;

ERR - jälgitakse ainult vea põhjustanud käske ja alamkäske;

ALL - jälgitakse kõiki käske ja protseduuridirektiive;

* - säilib varempandud jälgimisrežiim (ka vaikimisi);

OFF - tühistatakse varempandud jälgimisrežiim.

Teise operandina võib direktiivis näidata veel nn. väljundtegevuse, mille esitamiseks on järgmised kolm võimalust.

&PRINT [sõna1 [sõna2 ...]] - teatab, et kogu jälgimisinfo tuleb väljastada ekraanile, kusjuures võtmesõna &PRINT järele kirjutatud informatsioon lisatakse (pärast muutujate asendamist vastavate väärtustega) jälgimisinfo ette.

&COMMAND sõna1 [sõna2 ...] - teatab, et iga jälgimisinforea ette tuleb lisada (pärast muutujate asendamist nende väärtustega) kogu võtmesõna &COMMAND järel näidatud infor-

matsioon ning selliselt saadud rida täita omaette käsuna. Kirjutades näiteks protseduuri algusesse direktiivi

&TRACE ALL &COMMAND EXECIO 1 DISKW TRACE DATA A (STRING

saame kogu jälgimisinformatsiooni failis identifikaatoriga TRACE DATA A (ekraanile jälgimisinfot nüüd ei väljastata).

&SUBCOMMAND alamkeskkond [sõnal [sõna2 ...]] - teatab, et iga jälgimisinforea ette lisatakse alamkeskkonna nimele järgnev informatsioon (muutujad asendatakse nende väärtustega) ja nii saadud rida täidetakse alamkeskkonna alamkäsuna.

Protseduuri töötlemisel kasutab interpretaator üldiselt virtuaalarvuti mälus asuvat puhvrit. Selles salvestatakse näiteks märgendite asukohad, täidetavate tsüklite koopiad jne., mis oluliselt kiirendab interpretaatori tööd. Puhvri suurust mõõdetakse üldiselt sellesse mahtuvate protseduuriridade arvuga. Interpretaatori käivitamisel on puhvri suuruseks kolmkümmend kaks rida. Seda saab aga töö käigus ka muuta, kasutades selleks järgmise kujuga direktiivi

$$\&\text{BUFFER} \left\{ \begin{matrix} n \\ * \end{matrix} \right\}$$

Selle direktiivi alusel vabastatakse senine puhver ning tarbe korral moodustatakse ka uus. Ainsa operandina tuleb siin näidata uue puhvri ridade arv. Kui see on null, siis uut puhvrit ei moodustata. Tärni näitamisel võetakse aga puhvriks kogu vaba mälu.

Maksimaalse efektiivsuse saavutamiseks peaks puhver mahutama protseduuri pikima tsükli. Kui aga tsüklist pöördutakse tarbijafunktsioonide ja/või -alamprotseduuride poole, siis peaks puhver ka neid mahutama.

Kaks direktiivi on seotud suurtähtede režiimi kehtestamisega. Esimene nendest kehtestab suurtähtede režiimi terminaalilt sisestamisel ning esitatakse üldkujul

$$\&\text{CASE}\left[\begin{matrix}\underline{\text{U}}\\ \underline{\text{M}}\end{matrix}\right]$$

Siin U näitamisel operandina (ja ka vaikimisi) muudetakse järgnevate direktiividega &READ (vt. lk. 16) loetud tekstis kõik väiketähed suurteks, M näitamisel aga mitte.

Teine vaadeldavatest direktiividest kehtestab suurtähtede režiimi aga muutujatele. Selle direktiivi üldkuju on

$$\&\text{UPPER}\left\{\begin{matrix}\text{ARGS}\\ \text{VARS [muut1 [muut2 ...]]}\end{matrix}\right\}$$

Siin ARGS näitamisel operandina teisendatakse kõik väiketähed suurtähtedeks erimuutujate &1, &2 jne., teisel juhul aga näidatud tarbijamuutujate muut1, muut2 jne. väärtustes.

Tarbe korral saab interpretaatorile kehtestada veel ka nn. lõikamisrežiimi, millega määratakse kindlaks protseduuri sisaldavast failist töötlemiseks loetavate protseduuriridade pikkus. Vastava direktiivi üldkuju on järgmine

$$\&\text{TRUNC}\left[\begin{matrix}\underline{\text{K}}\\ \underline{\text{x}}\end{matrix}\right]$$

Ainsa operandiga määratakse siin lõikamisveerg. Tärni korral (ja vaikimisi) on selleks rea maksimaalpikkus 255. Selle režiimi kasutamisel tuleb aga arvestada, et ta ei kehti terminaalilt loetud või direktiividele &BEGPRINT (ka &BEGTYPE) ja &BEGSTACK (vt. lk. 18) järgnevatele ridadele. Lõikamine toimub enne muutujate asendamist vastavate väärtustega (pärast asendamist üle lõikamisveeru ulatuvat rida ei lõigata).

Muutujatele väärtuste omistamine toimub omistamisdirektiiviga, mille üldkuju on

$$\text{muutujanimi} = \left\{\begin{array}{l}\text{arv}\\ \text{sõna}\\ \text{aritmeetiline avaldis}\\ \text{funktsioon}\end{array}\right\}$$

Muutuja väärtuseks saab seega omistada kas arvu, ühesõnalise teksti (mitmesõnalise, s.t. tühikuid sisaldava teksti omistamiseks tuleb kasutada standardfunktsioone &LITERAL OF või &STRING OF), aritmeetilise avaldise väärtuse või funktsiooni väärtuse. Funktsioonina võib seejuures näidata nii standardfunktsiooni kui ka tarbijafunktsiooni.

Omistamisdirektiivi kirjutamisel peab arvestama eeskätt seda, et tühikutega tuleb ümbritseda nii võrdusmärk kui ka kõik aritmeetilises avaldises esinevad tehtemärgid. Võrdusmärgist vasakul mitmest muutujast koosneva järjendi näitamisel asendatakse esialgu kõik muutujad (välja arvatud vasakpoolseim) paremalt alates nende väärtustega. Asendamiste tulemusena saadud muutujale omistatakse nüüd võrdusmärgist paremal näidatud väärtus.

Erimuutujatest &i saab (ümber)väärtustada üksnes neid, mis on antud hetkel määratud, s.t. mille korral i<&INDEX ehk mis algväärtustati juba protseduurile ülekantavate parameetritega. Seega võimaldab omistamisdirektiiv protseduuri sees muuta vaid varemväärtustatud erimuutujaid. Vaadeldavaid erimuutujaid saab aga protseduuris siiski nii algväärtustada kui ka muuta direktiiviga, mille üldkuju on

&ARGS [sõna1 [sõna2 ...]]

Siin argumentidena näidatud sõnad omistatakse vastavalt erimuutujate &i, &2 jne. väärtusteks, kusjuures erimuutuja &N (ehk &INDEX) väärtuseks saab nüüd direktiivis &ARGS näidatud sõnade arv. Kui see arv on väiksem varemväärtustatud erimuutujate arvust, siis ülejäänute väärtused kustutatakse. Argumentide puudumisel kustutatakse kõigi erimuutujate &i väärtused ja erimuutuja &INDEX väärtuseks saab null.

Kui soovitakse ümber määrata vaid osa erimuutujatest, siis võib seda teha hariliku omistamisdirektiivi abil. Teine võimalus erimuutujate osaliseks muutmiseks seisneb selles, et direktiivis &ARGS näidatakse nende erimuutujate nimed, millede väärtusi muuta ei soovita. Näiteks direktiiviga

&ARGS A &2 &3 B &5 C

omistatakse uued väärtused A, B, C vastavalt muutujatele &1, &4 ja &6, kuid &2, &3 ja &5 säilitavad oma väärtused.

Sisestamine ja väljastamine. Andmete sisestamiseks magasinist (vt. lk. 25) või terminaalilt (kui magasin on tühi) toimub direktiiviga, mille üldkuju on

```
&READ [ n
      | 1
      | *
      | ARGS
      | STRING muut
      | VARS [ muut1 [ muut2 ... ] ]
             [ *      [ * ...     ] ] ]
```

Siin ridade arvu n (vaikimisi üks) või tärni (määramata arv ridu) näitamisel võetakse loetud read interpretaatori poolt täitmisele nii, nagu oleksid nad osa täidetavast protseduu-

rist. Selline ridade lugemine ja täitmine lõpeb kas näidatud arvu ridade töötlemise järel või siis loetute seas direktiivide &BEGPRINT (ka &BEGTYPE), &BEGSTACK, &GOTO, &SKIP, &LOOP või &EXIT esinemisel. Ajutiselt katkestatakse lugemine-täitmine tarbija alamprotseduuri või funktsiooni poole pöördumisel, kuid see jätkub pärast naasmist.

Operandina võtmesõna (ARGS, STRING või VARS) näitamisega saab loetava rea või selle sõnad omistada muutuja(te)le.

Võtmesõna ARGS toimel loetakse rida ning selles sisalduvad sõnad omistatakse erimuutujatele &1, &2 jne. Vastavalt muudetakse ka erimuutuja &N (ehk &INDEX) väärtust.

Võtmesõna STRING toimel omistatakse loetud rida kui sõne näidatud muutujale. Seejuures tühikuid ei eemaldata ning reas esinevaid muutujaid nende väärtustega ei asendata.

Võtmesõna VARS toimel omistatakse loetud reas esinevad sõnad näidatud muutujatele. Kui seejuures sõnade arv ületab muutujate arvu, siis ülejäänud sõnu ignoreeritakse; vastupidisel juhul aga väärtustamata jäänud muutujad kustutatakse (seega muutujateta direktiivi &READ VARS saab kasutada rea kõrvaldamiseks sisendvoost). Muutujanime asemel tärni näitamisel jäetakse vastav sõna reas lihtsalt kõrvale.

Direktiivi &READ kasutamisel tuleb arvestada, et võtmesõnade ARGS ja VARS toimel loetud rida skaneeritakse sõnadeks (eelnevad, järgnevad ja liigsed vahepealsed tühikud eemaldatakse), kuid sõnades muutujaid väärtustega ei asendata. Koos võtmesõnadega VARS või STRING näidatud muutujanimedega toimitakse samuti kui omistamisdirektiivis vasakul pool võrdusmärki asuvatega, s.t. muutujad asendatakse nende väärtus-

tega (paremalt alates) kuni äärmise vasakpoolseni, millele siis omistatakse kas loetud sõna(d) või terve rida.

Sõltumata direktiiviga &TRUNC kehtestatud režiimist interpretaator direktiivi &READ toimel loetud ridu ei lõika. Ühtlasi tuleb meelde tuletada, et terminaalilt loetava rea maksimaalpikkus on 130 ja magasinist loetaval 255 sümbolit.

Informatsiooni ekraanile väljastamiseks on kaks võimalust. Esimese neist realiseerib direktiiv üldkujuga

$$\left\{\begin{matrix}\text{\&TYPE}\\ \text{\&PRINT}\end{matrix}\right\} \text{[sõna1 [sõna2 ...]]}$$

Siin operandidega määratud rida enne väljastamist skaneeritakse ja muutujad asendatakse väärtustega.

Skaneerimata ning ilma muutujate asendamiseta read saab aga väljastada järgmist üldkuju omava direktiiviga

$$\left\{\begin{matrix}\text{\&BEGTYPE}\\ \text{\&BEGPRINT}\end{matrix}\right\} \left[\begin{matrix}n & \left[\begin{matrix}K\\ *\end{matrix}\right]\\ 1 & \\ * & \\ \text{märgend} & \end{matrix}\right]$$

rida1

rida2

.....

märgend

Siin esimese operandina näidatakse kas direktiivile vahetult järgnevate väljastamisele kuuluvate ridade arv n (vaikimisi üks, tärni näitamisel faili lõpuni) või siis märgend, milleni väljastamine toimub (märgendatud rida jääb väljastamata). Teise operandiga saab määrata väljastamisele kuuluvate sümbolite arvu reas (tärni näitamisel ja vaikimisi terve rida).

Ridade skaneerimise ning muutujate asendamise mõttes on äsjavaadeldutega analoogilised veel informatsiooni magasini kandmise direktiivid, milledest esimese üldkuju on

&STACK [[FIFO / LIFO] [sõna1 [sõna2 ...]]]

Siin esimene operand määrab rea magasini kandmise viisi: kas FIFO (First In First Out) või siis LIFO (Last In First Out). FIFO toimel (ja vaikimisi) kantakse rida magasini põhja ning loetakse hiljem magasinist (näiteks direktiiviga &READ) pärast kõikide eelnevalt magasini kantud ridade lugemist. LIFO toimel seevastu kantakse rida magasini tippu ning loetakse hiljem välja enne kõiki teisi (varem magasini kantud) ridu.

Skaneerimata ja muutujate asendamiseta read saab magasini kanda direktiiviga, mille üldkuju on

&BEGSTACK [n / 1 / * / märgend [K [FIFO / LIFO]]]

rida 1

rida 2

.....

märgend

Siin jääb kehtima kõik varem direktiivi &BEGTYPE kirjelduses, aga samuti ka äsja võtmesõnade FIFO ja LIFO kohta öeldu. Lisada tuleb siiski veel seda, et pärast direktiivide &BEGTYPE (ehk &BEGPRINT) ja &BEGSTACK alusel ridade väljastamist jätkub protseduuri töö kas realt numbriga n+1 või siis märgendit sisaldavast reast.

Nii erimuutujate ℓi kui ka tarbijamuutujate hetkeväärtusi väljastab kontrollimiseks ekraanile direktiiv üldkujuga

$$\&DUMP \left\{\begin{array}{l}ARGS\\ VARS\ [muut1\ [muut2\ \ldots]]\end{array}\right\}$$

Siin võtmesõna ARGS toimel väljastatakse kõigi väärtustatud erimuutujate väärtused. Näidates aga võtmesõna VARS ja muutujanimed saame ekraanil nende muutujate väärtused. Nii erikui ka tarbijamuutujate väärtused esitatakse ekraanil kujul:

muutujanimi = väärtus

Protseduuri täitmise juhtimise direktiividest lihtsaimad on kaks tingimusteta suunamise direktiivi. Esimene neist

$$\&GOTO \left\{\begin{array}{l}reanumber\\ märgend\end{array}\right\}$$

annab juhtimise näidatud reanumbriga või märgendiga reale. Seejuures märgendi kasutamisel tuleb pikemates protseduurides võimaluse korral vältida tagasisuunamisi, sest märgendit otsib interpretaator töödeldavast reast alates kuni protseduuri lõpuni ja jätkab alles seejärel protseduuri algusest.

Teine nimetatud direktiividest üldkujuga

$$\&SKIP \left[\begin{array}{l}n\\ \underline{1}\end{array}\right]$$

määrab lihtsalt protseduuri täitmisel vahelejäetavate ridade arvu n (vaikimisi 1). Positiivse n korral antakse juhtimine edasi, negatiivse n korral aga tagasi (vahelejäetavate hulka jääb siis ka direktiivi &SKIP sisaldav rida).

Tingimusliku direktiivi üldkuju vaadeldavas keeles on

$$\&IF\ sõna1\ võrdlusmärk\ sõna2 \left\{\begin{array}{l}käsk\\ direktiiv\end{array}\right\}$$

Tingimuse moodustab siin kolme esimese operandiga määratud võrdlus. Kui tingimus on rahuldatud (tõene), siis täidetakse vastav käsk (alamkäsk) või direktiiv, vastasel korral tuleb aga täitmisele järgmine protseduuririda. Tingimuses võib kasutada järgmisi võrdlusmärke (sulgudes esitamisviis keeles EXEC2): võrdne (= või EQ), mittevõrdne (¬= või NE), väiksem (< või LT), väiksem-võrdne (<= või ¬> või LE või NG), suurem (> või GT) suurem-võrdne (>= või ¬< või GE või NL).

Vaadeldava direktiivi kasutamisel tuleb hoolitseda selle eest, et iga võrdluses osalev muutuja oleks tingimusliku direktiivi täitmise momendiks kindlasti väärtustatud, sest vastasel korral võib tekkida süntaksiviga. Kui näiteks muutuja &M on väärtustamata, siis omandab direktiiv

```
&IF &M = * &GOTO -GO
```

täitmisel järgmise, ilmselt vigase kuju

```
&IF = * &GOTO -GO
```

Kõige kindlama tee sellise vea vältimiseks annab mingi sümboli (näiteks punkti) lisamine võrdluses osalevate suuruste ette. Näiteks eelmise direktiivi võiks kirjutada nii

```
&IF . &M = . * &GOTO -GO
```

Tsüklite organiseerimiseks saab vaadeldavas protseduuridekeeles kasutada direktiivi, mille üldkuju on

$$\&\text{LOOP}\ \left\{\begin{matrix}\text{n}\\ \text{märgend}\end{matrix}\right\}\ \left\{\begin{matrix}\text{m}\\ *\\ \left\{\begin{matrix}\text{WHILE}\\ \text{UNTIL}\end{matrix}\right\}\ \text{tingimus}\end{matrix}\right\}$$

Siin esimese operandiga määratakse kas tsüklisse kuuluvate (järgnevate) ridade arv n või siis viimase tsüklisse kuuluva

rea märgend. Järgneva(te) operandi(de)ga saab määrata tsükli kordamiste arvu m (tärn jätab selle arvu määramata) või siis tsükli täitmise tingimuse (võrdlusmärk kahe sõna vahel), mida kontrollitakse enne tsükli iga täitmist. Tingimusele eelnev võtmesõna täpsustab tingimuse kontrollimist järgmiselt: WHILE korral täidetakse tsüklit seni kuni tingimus on rahuldatud (täitmine lõpetatakse siis, kui võrdlus saab vääraks); UNTIL korral täidetakse tsüklit seni kuni tingimus pole rahuldatud (täitmine lõpeb kui võrdlus on tõene).

Põhiprotseduuriga samasse faili kuuluva alamprotseduuri poole saab pöörduda järgmise üldkujuga direktiivi abil:

$$\&CALL \left\{ \begin{matrix} reanumber \\ märgend \end{matrix} \right\} [sõna1 \ [sõna2 \ \ldots]]$$

Siin esimene operand näitab alamprotseduuri algusrea numbri või märgendi, järgnevad aga kannavad alamprotseduurile üle parameetrid (millega väärtustatakse erimuutujad &i).

Alamprotseduur peab lõppema direktiiviga &RETURN, mis tagastab juhtimise põhiprotseduuri direktiivi &CALL sisaldava rea järele. Seejuures taastatakse erimuutujate &1, &2 ... ning &N (ehk &INDEX) väärtused. Selle direktiivi üldkuju on

&RETURN [sõna]

Siin operandi kasutatakse üksnes tarbijafunktsioonina välja kutsutud alamprotseduuri korral funktsiooni väärtuse (protseduuri ainsa tulemuse) tagastamiseks. Tarbijafunktsiooni võib välja kutsuda omistamisdirektiivis järgmisel viisil:

$$\left\{ \begin{matrix} reanumber \\ märgend \end{matrix} \right\} OF \ [sõna1 \ [sõna2 \ \ldots]]$$

kus parameetrite osas kehtib alamprotseduuride kohta öeldu.

Vaadeldav keel võimaldab programmeerijal protseduuris ette määrata ka omapoolse tegevuse mingi PTS-käsu (-alamkäsu) täitmisel tekkiva vea korral, s.t. kui lõpukood (RC) on nullist erinev. Seda saab korraldada direktiiviga üldkujul

&ERROR tegevus

Siin näidatud tegevus laieneb kõikidele protseduuris täidetavatele PTS-käskudele (-alamkäskudele) ja kehtib kas järgmise direktiivini &ERROR või siis protseduuri lõpuni. Eelmise direktiiviga määratud tegevuse saab tühistada operandita direktiivi &ERROR abil. Märgime lõpuks veel seda, et pärast direktiiviga &ERROR näidatud tegevuse täitmist antakse juhtimine tagasi vea põhjustanud käsu (alamkäsu) järel paiknevale reale (muidugi siis, kui tegevus ei muuda juhtimist).

Protseduuri täitmise lõpetab direktiiv üldkujuga

EXIT [lõpukood]
     [0   ]

kus lõpukoodina võib otse või siis muutuja vahendusel näidata arvu vahemikust -2147483648 kuni 2147483647 .

Käskude täitmise juhtimine. Teatavasti saab protseduurides täita nii CP- kui ka PTS-käske ja alamkäske, aga samuti ka EXEC- või MODULE-tüüpi failidena vormistatud tarbijaprotseduure või programme. Seejuures CP-käskudele peab kindlasti ette kirjutama tunnusena "CP", protseduurid tuleb aga käivitada PTS-käsu EXEC vahendusel, mille esimese operandiga näidatakse vastava EXEC-faili nimi, järgnevate operandidega aga ülekantavad parameetrid. PTS-käskude ja alamkäskude ning programmide (kui MODULE-tüüpi failide) täitmine protseduuris organiseeritakse aga vastavate direktiividega.

Vaatleme kõigepealt direktiivi, mille üldkuju on

&PRESUME [&COMMAND
           &SUBCOMMAND keskkond]

See direktiiv kehtestab (alam)keskkonna, milles tuleb täita kõik protseduuri järgnevad käsud (või alamkäsud). Võtmesõna &COMMAND toimel (ja vaikimisi) antakse protseduuris esinevad käsud täitmiseks põhisüsteemile (enamasti dialoogmonitorsüsteemile PTS). Seega piisab nii PTS-käskude kui ka programmide (MODULE-tüüpi failide) täitmiseks nende nime näitamisest (lisades tarbe korral ka ülekantavad parameetrid). Võtmesõna &SUBCOMMAND koos alamkeskkonna nimega (näiteks XEDIT) kehtestab vastava alamkeskkonna (näiteks korrigeerimiskeskkonna). Kui pärast vaadeldavat direktiivi esineb kehtestatud alamkeskkonna jaoks tundmatu (alam)käsk, siis antakse vastav lõpukood (-3) ning väljastatakse veateade.

Kaks järgnevat direktiivi annavad võimaluse vastavalt kas PTS-käsu või programmi täitmiseks kehtivas alamkeskkonnas või siis mingi alamkäsu täitmiseks PTS-keskkonnas. Nendest direktiividest esimese üldkuju on

&COMMAND nimi [parameetrid]

kus esimese operandina näidatakse kas PTS-käsu või käivitatava programmi (MODULE-tüüpi faili) nimi, millele järgnevad tarbe korral parameetrid.

Teine vaadeldavatest direktiividest võimaldab PTS-keskkonnas täita mingi alamkeskkonna alamkäsku. Esimese operandina näidatakse siin alamkeskkond, järgnevatega aga alamkäsu nimi ja tarbe korral parameetrid. Direktiivi üldkuju on

&SUBCOMMAND keskkond nimi [parameetrid]

2. MAGASINID JA PUHVRID

2.1. Sisend- ja väljundmagasin

Vaadeldav süsteem kasutab tegelikult kaht magasini: sisend- ja väljundmagasini. Programmeerimise seisukohalt pakub neist seejuures rohkem kasutamisvõimalusi just esimene.

Sisendmagasini võib informatsioon (käsud, andmed) sattuda kas terminaalilt sisestamisel või protseduuri (programmi) täitmisel näiteks direktiivi &STACK või PTS-käsu EXECIO (vt. lk. 29) toimel. Vastavalt jaotubki sisendmagasin kaheks osaks - puldipuhvriks ja programmimagasiniks. Sõltuvalt sellest, kust saabuvad sisendmagasini käsu- või andmeread, pannakse nad kas puldipuhvrisse või programmimagasini, kus nad koos varemkantud ridadega moodustavad lugemisele kuuluva ridade jada. Puldi sisendmagasini paigutatavate ridade maksimaalarv sõltub seejuures virtuaalarvuti põhimälu mahust.

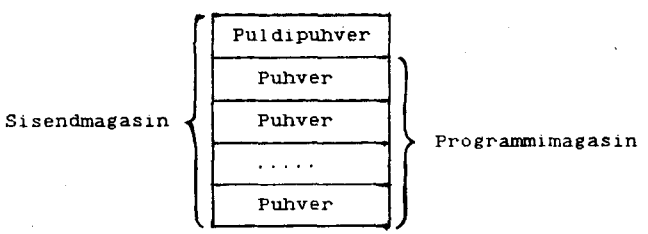
Sisestamisel näiteks direktiiviga &READ vaadatakse kõigepealt läbi programmimagasin, seejärel aga puldipuhver. Kui need on tühjad, siis toimub rea lugemine otse terminaalilt.

Kui terminaalilt käsu- või andmerea sisestamisel ilmub ekraani olekupiirkonda (ekraani parempoolses alaservas) teade NOT ACCEPTED, siis annab see tunnistust puldipuhvri "ületäitumisest". Sellisel juhul tuleb oodata puldipuhvri vabanemist ja alles pärast seda jätkata sisestamist. Analoogiliselt sisestamisega pannakse terminaalile väljastatavad andmed süsteemi poolt puldi väljundmagasini, kust nad väljastatakse ekraanile enne järjekordset lugemist sisendmagasinist.

2.2. Programmimagasini kasutamine

Mõlemad käesolevas väljaandes vaadeldavad protseduuridekeeled - nii EXEC2 (vt. peatükk 1) kui ka REXX (vt. peatükk 4) sisaldavad teatavasti direktiive informatsiooni vahetamiseks programmimagasiniga. Peale selle pakub aga veel ka dialoogmonitorsüsteem PTS mitmeid programmimagasiniga manipuleerimise võimalusi vastavate (allpool vaatlusele võetavate) PTS-käskude vahendusel.

Siiani rääkisime programmimagasinist kui ühest tervikust. Tegelikult aga saab selle omakorda jaotada veel mitmeks puhvriks, s.t. anda programmimagasinile mitmetasemelise struktuuri. Kogu sisendmagasini struktuuri võime seega esitada järgneval joonisel toodud viisil:



Uue puhvri moodustamiseks programmimagasini tuleb kasutada järgmist operandideta PTS-käsku

MAKEBUF

Uue puhvri *tasemenumbri* tagastab käsk MAKEBUF lõpukoodina, mille väärtus protseduuridekeele EXEC2 või REXX korral omistatakse vastavalt kas erimuutujale &RC või siis RC . Protseduuris kõik enne esimese käsu MAKEBUF andmist programmimagasini kantud andmed moodustavad seal nn. null taseme puhvri.

Mitmetasemelise programmimagasini korral algab ridade lugemine alati kõige kõrgema taseme, s.t. hiljem moodustatud puhvrist. Alles siis, kui sellest on loetud kõik read, alustatakse lugemist temale järgnevast (s.t. tasemelt järgmisest madalamast) puhvrist jne.

Operandideta käsk

SENTRIES

annab teada programmimagasinis sisalduvate ridade koguarvu (üle kõikide puhvrite), mis samuti tagastatakse lõpukoodina nagu käsu MAKEBUF korral.

Siinkohal peab eriti alla kriipsutama veel seda, et kahe äsjavaadeldud PTS-käsu eripära (andmete tagastamist lõpukoodina) tuleb arvestada protseduuris veatöötlusdirektiivi &ERROR kasutamisel.

Käsuga MAKEBUF moodustatud puhvrid saab vabastada käsuga, mille üldkuju on

DROPBUF [tasemenumber]

Selle käsu täitmisel vabastatakse näidatud tasemenumbriga puhver koos kõikide temast kõrgemat tasemenumbrit omavatega (kui neid leidub).

Nii kogu sisendmagasini (kõikide puhvrite) kui ka väljundmagasini vabastamine toimub operandideta käsuga

DESBUF

Kuna see käsk puhastab ka väljundmagasini, siis tuleb eelnevalt anda operandideta PTS-käsk CONWAIT võimaldamaks kõikide ridade täielikku väljastamist.

Käesolevas punktis toodud PTS-käskude kasutamist illustreerib järgmine näiteprotseduur:

```
&TRACE
&STACK BUFORIDA1
&STACK BUFORIDA2
MAKEBUF
&TYPE &RC
&STACK BUF1RIDA1
&STACK BUF1RIDA2
&READ VARS &R1
&TYPE &R1
DROPBUF 1
&READ VARS &R2
&TYPE &R2
CONWAIT
DESBUF
&EXIT
```

Protseduuri alguses kantakse programmimagasini (tase 0) read BUFORIDA1 ja BUFORIDA2. Seejärel moodustatakse uus puhver (tase 1), mille tasemenumber väljastatakse ka ekraanile (&TYPE &RC). Edasi kantakse programmimagasini (nüüd järelikult juba puhvrisse tasemega 1) read BUF1RIDA1 ja BUF1RIDA2. Järgnevalt loetakse puhvrist ja väljastatakse ekraanile üks rida (BUF1RIDA1) ning vabastatakse puhver tasemega 1 (millega ühtlasi kustutatakse ka vastavasse puhvrisse jäänud rida BUF1RIDA2). Sellele järgneva lugemise ja väljastamise tulemusena saame ekraanile rea BUFORIDA1. Kõige lõpuks täidetavate PTS-käskude CONWAIT ja DESBUF abil kustutatakse magasinist viimane sinna veel jäänud rida (BUFORIDA2) ning sellega protseduuri täitmine ka lõpeb.

2.3. Informatsiooni vahetamine perifeerseadmetega

Käesolevas punktis vaatluse alla võetav PTS-käsk EXECIO võimaldab sisestada kirjeid PTS-failist või virtuaalkaartsisendilt (programmi)magasini, kanda sinna CP-käskude täitmise tulemusi, väljastada ridu magasinist PTS-faili, virtuaalkaartväljundile või -trükiseadmele jms. Käsu üldkuju on

```
                                  ⎧DISKR fn ft [fr [kn]][(FINIs ri r2]
                                  ⎪CARD [(ri r2]
                                  ⎪CP    [(ri r2 STring rida]
                                  ⎪DISKW  fn ft fr [kn [f  [p]]]
                                  ⎪    [(FINIs r2 CAse [U] STring rida]
                                  ⎪                   [M]
EXECIO {ka} {*} {0}               ⎪PUNCH [(r2 CAse [U] STring rida]
                                  ⎪                [M]
                                  ⎪PRINT [([CC {Kood} r2
                                  ⎪            {DATA}
                                  ⎪                CAse [U] STring rida]
                                  ⎪                     [M]
                                  ⎩EMSG [(r2 CAse [U] STring rida]
                                                  [M]
```

Lühendiga "ri" on siin kokkuvõtlikult tähistatud režiime

```
⎡⎡{FInd }      ⎤⎡      ⎡ni n2⎤⎤  ⎡LIFO⎤
⎢⎢{LOcate} või ⎥⎢Zone  ⎢    ⎥⎥  ⎣FIFO⎦ [SKip]
⎣⎣{Avoid }    ⎦⎣      ⎣1  *⎦⎦
```

Analoogiliselt lühend "r2" tähistab režiime

```
⎡Margins ⎡ni n2⎤⎤ [STRIP][NOTYPE]
⎣        ⎣1  *⎦⎦
```

Esimese kohustusliku parameetrina tuleb käsus EXECIO näidata infovahetuses osalevate kirjete arv (üldkujus tähistatud ka). Selle asendamisel tärniga toimub magasinist kirjete väljastamine kuni tühikirjeni (seda ei väljastata), magasini sisestamine aga faili lõpuni. Koos režiimiga STRING võib tärni seejuures kasutada ainult käsufunktsiooni CP korral. Esimese parameetrina nulli (0) näitamisel seevastu infovahetusoperatsiooni ei täideta. Kui aga seejuures on kehtestatud režiim FINIS, siis toimub ainult faili sulgemine.

Teise kohustusliku parameetri abil võib käsufunktsiooniks määrata kas sisestamise kettafailist (DISKR) või kaartsisendilt (CARD) või siis režiimiga STRING näidatud CP-käsu täitmisel saadud informatsiooni magasini kandmise (CP).

Magasinist väljastamisega seotud käsufunktsioonideks on kas väljastamine kettafaili (DISKW), kaartväljundile (PUNCH) või trükiseadmele (PRINT), samuti teadete väljastamine terminaalile (EMSG).

Kõikide väljastamisfunktsioonide korral võib väljastatava rea näidata režiimi STRING abil. Selle puudumisel toimub aga väljastamine magasinist. Kui ka magasin on tühi (või tühjaks saanud), siis väljastatakse vahetult terminaalilt sisestatavad read (kirjed).

Käsu EXECIO ülejäänud parameetreid ja režiime vaatleme allpool seotuna juba konkreetsete käsufunktsioonidega.

Kirjete failist magasini sisestamisel (DISKR) näidatakse failinimi, -tüüp ja tarbe korral ka -režiim (vaikimisi on selleks tärn, s.t. faili otsitakse kõigilt kättesaadavatelt ketastelt). Järgneva parameetriga kn saab määrata kirjenumb-

ri, millest alustada sisestamist (vaikimisi üks). Seejuures tuleb arvestada, et kui käsk EXECIO antakse juba avatud faili jaoks (avamine toimub esimese selle faili jaoks antud käsu EXECIO täitmisel), siis arvestatakse seda kirjenumbrit viimasena töödeldud kirjele järgnevast alates. Režiimi FINIS näitamisel suletakse fail pärast antud käsu täitmist.

Kõikide sisestamisfunktsioonide (DISKR, CARD, CP) korral võib kasutada ka käsu üldkujus lühenditega "r1" ja "r2" tähistatud režiime. Vaatleme neid siinkohal veidi lähemalt.

Esimene nendest võimaldab pärast nõutud arvu kirjete magasini sisestamist nende seast vajaliku (võtmega määratud) kirje otsimist. Võti kujutab endast seejuures eraldajatega (näiteks kaldjoontega) piiratud sõne. Võtmele eelnev võtmesõna täpsustab otsimist järgmiselt: FIND toimel otsitakse võtit kirje (või režiimiga ZONE määratud tsooni) algusest, LOCATE toimel kogu kirjest (tsoonist) ja AVOID toimel otsitakse sellist kirjet, mis ei sisalda näidatud võtit.

Ükskõik millise äsjakirjeldatud otsimisviisi järgi kirje leidmisel kantakse programmimagasini algusesse kaks rida, millest esimene sisaldab antud operatsiooniga töödeldud kirjete arvu ning leitud kirje järjekorranumbri, teine aga esimese nõutud võtit sisaldava (või mittesisaldava) kirje enda. Kui nõutud kirjet ei leidu, siis väljastatakse veateade ning infot magasini ei kanta. Otsimistsooni (vaikimisi kogu kirje) saab vajaduse korral täpsustada režiimiga ZONE, millele järgneva kahe arvuga (n1 ja n2) määratakse tsooni algus- ja lõpp-positsiooni numbrid. Järgnev režiim LIFO või FIFO (ka vaikimisi) määrab kirjete magasini kandmise viisi. Võtmesõ-

naga SKIP näidatav režiim määrab ridade (kirjete) vahelejätmise ilma infovahetusoperatsiooni täitmata.

Võtmesõna MARGINS koos algus- ja lõpp-positsiooninumbreid näitavate arvudega (n1 ja n2) määrab infovahetuses osaleva rea- või kirjeosa (vaikimisi kogu kirje). Infovahetuses osalevatest ridadest (kirjetest) algus- ja lõputühikute eemaldamist saab organiseerida võtmesõna STRIP abil kehtestatava režiimiga. Võtmesõna NOTYPE abil saab tühistada infovahetuse veateadete terminaalile väljastamise.

Käsufunktsiooni CP korral tuleb võtmesõna STRING järel eraldi reana näidata see CP-käsk, mille poolt väljastatavat infot soovitakse saada programmimagasini. Näiteks CP-käsu QUERY DASD poolt väljastatava info saame magasini käsuga

```
EXECIO * CP (STRING QUERY DASD
```

Väljastamisfunktsiooni DISKW korral tuleb käsus näidata selle faili identifikaator (fn, ft, fr), millesse väljastatavad read kantakse. Identifikaatori järel võib näidata vastavalt kirjenumbri (vaikimisi uue faili jaoks 1ks, olemasoleva jaoks aga viimasele kirjele järgneva number), kirjeformaadi F või V (ka vaikimisi) ning uue faili jaoks kirjepikkuse (vaikimisi F-kirjete jaoks 80, V-kirjete jaoks aga 255 baiti). Režiimidest võib lisaks varasemast juba tuntutele näidata veel CASE, millele järgneva sümboli U abil saab kehtestada väiketähtede suurtähtedeks teisendamise enne infovahetusoperatsiooni täitmist. Sümboli M näitamisel (või ka vaikimisi) seevastu teisendamist ei toimu.

Informatsiooni virtuaalkaartväljundile, -trükiseadmele või teate terminaalile väljastamiseks tuleb käsufunktsiooni-

na näidata vastavalt kas PUNCH, PRINT või EMSG. Tarbe korral võib veel lisada vajalikud režiimid varasemast juba tuttavate seast (MARGINS, STRIP, NOTYPE, CASE, STRING).

Trükiseadmele väljastamisel saab režiimi CC abil täpsustada veel trükijuhtsümboli. Selle võib anda kas võtmesõna CC järel otseselt (kood) või siis võetakse selleks väljastatava kirje esimene sümbol (näidata DATA).

Käsu EXECIO kasutamisel tuleb silmas pidada, et varemloodud kettafaili kirjutamisel peavad kirjeformaat ja pikkus sobima (muidu antakse veateade ning kirjeid ei väljastata). Olemasoleva kettafaili kirjeid võib ka asendada uutega kirjenumbriga kn määratud kirjest alates. Kui seejuures on tegemist V-kirjetega, siis peavad asendavate kirjete pikkused täpselt ühtima asendatavate omadega, sest vastasel juhul kas väljastatakse veateade ja asendamist ei toimu (800-baidiliste plokkidena formatiseeritud virtuaalketta korral) või siis kustutatakse kõik asendatavatele järgnevad kirjed (1K-, 2K- või 4K-baidiliste plokkide korral).

Ühe faili kirjete sisestamisel/väljastamisel mitme käsuga EXECIO pole iga kord tarvis faili sulgeda (s.t. näidata režiimi FINIS). Sellisel juhul iga järgmise käsu EXECIO alusel sisestatakse/väljastatakse kirje alates eelmise käsuga viimasena töödeldud kirjele järgnevast. Seejuures tuleb aga arvestada, et kui käskude EXECIO vahel täidetakse mingi sellesama failiga manipuleeriv CP- või PTS-käsk, siis selle järel fail suletakse. Niisugustel juhtudel tuleb järgnevas käsus EXECIO kindlasti otseselt näidata selle kirje järjekorranumber, millest alates soovitakse infovahetust jätkata.

2.4. Näiteprotseduurid

Käesolevas punktis esitame kaks terviklikku protseduuri illustreerimaks nii protseduuridekeele EXEC2 kui ka PTS-käsu EXECIO kasutamise võimalusi. Näidetes esinevate PTS-käskude kohta võib informatsiooni saada varem sarjas "Programme kõigile" ilmunud väljaannetest "Virtuaalarvutid" ja "Virtuaalsüsteem ühtsusseeria arvuteil" (täiendavad seletused on esitatud tärniga algavate kommentaaridena).

Vaadeldavatest protseduuridest esimene nimega XFILENM võimaldab A-kettale uue faili moodustamist terminaalilt sisestatavatest kirjetest või ka kirjete lisamist olemasoleva faili lõppu. Protseduuri käivitab käsurida kujul

```
XFILENM [ fn   [ ft   [kf  [ kp  ]]]]
        [ PROC [ EXEC [ V   [ 255 ]]]]
```

kus parameetritena võib ette anda failinime, -tüübi, kirjeformaadi ning kirjepikkuse.

Protseduuri tekst on järgmine:

```
&TRACE
* PARAMEETRITE KONTROLLIMINE JA
* VAIKIMISIVAARTUSTE OMISTAMINE
&IF &N <= 4 &GOTO -L&N
&PRINT PARAMEETRITE ARV ON VALE - &N
&EXIT 24
-L0 &ARGS PROC EXEC V 255
&GOTO -L4
-L1 &ARGS &1 EXEC V 255
&GOTO -L4
```

```
-L2 &ARGS &1 &2 V 255
&GOTO -L4
-L3 &ARGS &1 &2 &3 255
-L4
* FAILI OLEMASOLU KONTROLLIMINE
&OLEMAS = &BLANK
&COMMAND STATEW &1 &2 A
&IF &RC ¬= 0 &GOTO -ER
&PRINT FAIL '&1 &2 A' ON JUBA OLEMAS. SOOVITE
                    KIRJEID LISADA (TYHIRIDA) / EI (E)
&READ VARS &V
&IF . &V ¬= . &EXIT
&OLEMAS = *
&GOTO -OK
-ER &IF &RC = 28 &GOTO -OK
&PRINT VIGA KASU "STATEW" TAITMISEL ( RC = &RC )
&EXIT 10&RC
-OK
* FAILI MOODUSTAMINE VQI TAIENDAMINE
&PRINT SISESTAGE KIRJED. LQPETAMISEKS TYHIRIDA
&IF . &OLEMAS = . * &GOTO -OLEMAS
&COMMAND EXECIO * DISKW &1 &2 A 1 &3 &4 (FINIS
&GOTO -ER1
-OLEMAS
&COMMAND EXECIO * DISKW &1 &2 A (FINIS
-ER1
&IF &RC ¬= 0 &EXIT 100&RC
&EXIT
```

Järgmisena vaadeldav näiteprotseduur nimega XYCOPY kopeerib ühest A-ketta failist teise kirjed alates sõnet x sisaldavast kuni sõnet y sisaldavani (viimane kaasa arvatud). Erinevalt analoogilisest PTS-käsust COPYFILE ei esitata siin etteantavatele sõnedele x ja y mitte mingisuguseid kitsendusi. Protseduuri käivitamiseks tuleb nüüd sisestada järgmise struktuuriga käsurida:

```
XYCOPY fn ft x y fn
```

Siin kahe esimese parameetriga näidatakse lähtefaili nimi ja tüüp, kahe järgnevaga vastavad sõned, viimasega aga uue faili nimi (tüüp jääb samaks mis lähtefailil).

Protseduuri tekst on järgmine:

```
&TRACE
&PAR = &BLANK
* PARAMEETRITE ARVU KONTROLLIMINE
&IF &N = 5 &GOTO -OK
&TYPE PARAMEETRITE ARV ON VALE - &N
&EXIT 24
-OK
&COMMAND MAKEBUF
* FAILI OLEMASOLU KONTROLLIMINE
&COMMAND STATEW &5 &2 A
&IF &RC ¬= 0 &GOTO -E
&TYPE FAIL ON OLEMAS: KAS LISADA LQPPU (AP),
        KIRJUTADA YLE (REP) VQI LQPETADA (TYHIRIDA
&READ VARS &PAR
&IF . &PAR = . &GOTO -EXIT
&GOTO -OK1
```

```
-E &IF &RC = 28 &GOTO -OK1
&TYPE VIGA KASUS "STATEW" ( RC = &RC )
&RCC = &RC
&GOTO -EXIT
-OK1
* PTS-KASKE "EXECIO" KASUTATAKSE SIIN ESIMESE
* KIRJE NUMBRI NING KIRJETE ARVU MAARAMISEKS
* PTS-KASU "COPYFILE" JAOKS
&T = &CONCAT OF / &3 /
&COMMAND EXECIO * DISKR &1 &2 A (LOCATE &T
&IF &RC ¬= 0 &GOTO -ERROR
&READ VARS * &M1
&T2 = &CONCAT OF / &4 /
&COMMAND EXECIO * DISKR &1 &2 A (LO &T2 FINIS
&IF &RC  = 0 &GOTO -ERROR
&READ VARS * &M2
&M = &M2 - &M1 + 1
&COMMAND COPY &1 &2 A &5 &2 A (FROM &M1 FOR &M &PAR
&IF &RC ¬= 0 &RCC = 110&RC
-EXIT
&COMMAND DROPBUF
&EXIT &RCC
-ERROR
&RCC = 100&RC
* JARGNEV KASK "EXECIO" ON FAILI
* SULGEMISEKS TQRKE KORRAL
&COMMAND EXECIO 0 DISKR &1 &2 A (FINIS
&GOTO -EXIT
```


10

3. P A N E E L I S Ü S T E E M I S T

3.1. Sissejuhatus

Paneelisüsteem on määratud informatsiooni vahetamiseks protseduuri (programmi) ja tarbija vahel terminaali täisekraanide kaupa. Järgnevalt käsitleme paneelisüsteemi kasutamise lihtsamaid võimalusi protseduuridekeeles EXEC2.

Paneelisüsteemi abil saab kõik protseduurile vajalikud andmed esitada kindla kujundusega ekraani ehk paneeli loogilistel väljadel ning sisestada need korraga näiteks sisestamisklahvile vajutamise teel. Seejuures sisaldab paneel enamasti veel vajalikud juhendid, küsimused ja võimaluse korral ka vastuste variandid (töö toimub seega dialoogi vormis).

Paneeli loogilistel väljadel olev informatsioon omistatakse sisestamisel automaatselt standardnimedega (või protseduuris määratud nimedega) muutujatele ja on seega kohe kasutatav protseduuri edaspidisel täitmisel. Samuti aga protseduuris muutujatele omistatud väärtused ilmuvad paneeli ekraanile väljastamisel vastavatele paneeliväljadele.

Üldiselt koosneb paneele kasutava protseduuri koostamine järgmistest etappidest. Kõigepealt planeeritakse infovahetusele vastav(ad) ekraanikujundus(ed), kusjuures eeskujuks on näiteks "paberliku asjaajamise" blanketid. Seejärel moodustatakse spetsiaalse alamsüsteemi abil paneel(id) ja salvestatakse A-kettale. Lõpuks koostatakse moodustatud paneeli(de)ga informatsiooni vahetav protseduur, kasutades selleks spetsiaalseid direktiive ja erimuutujaid.

3.2. Paneeli moodustamine

Uue paneeli moodustamine või olemasoleva muutmine toimub PTS-käsuga PANEL2, mille ainsa operandina võib näidata paneeli nime. Kogu edasine töö toimub dialoogina standardekraanide (-paneelide) vahendusel.

Operandita käsu PANEL2 täitmisel ilmub ekraanile alguspaneel, millel tuleb valida paneelinimi ning vajutada sisestamisklahvile. Seejärel (kui käsus PANEL2 oli nimi, siis kohe) ilmub dimensioonipaneel mõõtude näitamiseks (standardselt 24 80-sümbolilist rida). Jätkamiseks vajutatakse klahvile PF2, mis väljastab moodustamispaneeli, kus vasakul serval on reanumbrid, ülal aga veerunumbrid (rea tagumiste positsioonide ekraanile saamiseks tuleb uuesti vajutada klahvile PF2). Vabaks jääval ekraaniosal moodustab tarbija nüüd paneeli, pidades seejuures silmas järgmist. Näiteks mistahes teksti ees peab (standardmäärangu alusel) olema eraldajana kas märk "[", või "]", iga andmevälja ees aga allkriips "_". Andmevälja pikkuse näitamiseks võib kasutada ühte või mitut täitesümbolit "#". Eraldajate ja täitesümbolite hetkeväärtused on esitatud ekraani viimasel real, kus neid võib tarbe korral ka muuta, valides vastavatel väljadel lihtsalt uued märgid ning vajutades sisestamisklahvile.

Paneeli kujundamisel saab kasutada ka nn. erialamkäske, mille valimiseks tuleb vajutada klahvile PF6. Et aga andmed alamkäskude kohta saab ekraanile tuua klahvile PF7 (HELP) vajutamise teel, toome siin vaid lühiülevaate:

A n m - jätab rea number n järele m tühja rida;

BO - toob ekraanile paneeli lõpuosa (viimase reani);

CE n1 [n2] - tsentreerib read numbritega n1 - n2;

CO n1 n2 [n3] - kopeerib read n2 - n3 rea n1 järele;

DE n1 n2 - eemaldab read n1 - n2, n2 = * korral lõpuni;

DI - toob paneeli ekraanile nii nagu protseduuris;

DU n [m] - dubleerib *m* korda rida *n* (vaikimisi m=1);

M n1 n2 [n3] - nagu CO, kuid read n2 - n3 eemaldatakse;

T - toob ekraanile paneeli alguse;

n - paneb rea number n ekraani algusesse.

Kui paneel või selle osa on kujundatud (paneeli hetkeseisu võib võtmele PF4 vajutamise teel kettale salvestada ja tööd jätkata), siis asutakse paneelivälju kirjeldama. Klahvile PF3 vajutamisel ilmub ekraanile kirjeldamispaneel, mille ülaservas helendab vastava rea see väli, mida antud hetkel saab kirjeldada. Kirjeldamisel võib määrata välja heleduse, valitava informatsiooni tüübi jne. (vajalikud andmed välja kirjeldamiseks on ekraanil olemas). Uus vajutus klahvile PF3 annab võimaluse järgmise välja kirjeldamiseks. Väljade kirjeldamise käigus võib klahvile PF2 vajutamise teel tagasi minna moodustamispaneelile (näiteks selleks, et alamkäsu DI abil kontrollida kujunduse sobivust, teha mingeid parandusi-täiendusi vms.). Kui paneeli paljude väljade seast soovitakse (ümber) kirjeldada vaid ühte konkreetset, siis tuleb moodustamispaneelil juhtida kursor vastava välja eesolevale eraldajale ning vajutada klahvile PF3.

Kujundamise lõpetamiseks toob klahv PF5 ette lõpupaneeli, mille abil salvestatakse paneel A-kettale kahe failina: nende ühine nimi on paneelinimi, tüüp aga PANEL ja PCB.

3.3. Protseduuri kirjutamine

Paneelide kasutamiseks on protseduuridekeeles EXEC2 olemas erivahendid, milliste tutvustamisel piirdume siin ühe võimaliku, praktikas proovitud kasutamisviisi esitamisega.

Informatsiooni vahetamiseks paneeli ja protseduuri vahel annab süsteem kõikidele (loogilistele) väljadele paneelil nende esinemise järjekorras nimed, näiteks andmeväljadele D1, D2, ... ja tekstiväljadele T1, T2, ... Andmeväljadele kantud info omistatakse seejuures automaatselt vastavatele erimuutujatele &D1, &D2, ... ning nendele muutujatele protseduuris omistatud väärtused kantakse paneeli järjekordsel ekraanile väljastamisel vastavatele andmeväljadele.

Peale vaadeldute on paneelisüsteemis kasutatavad veel mitmed erimuutujad, milledest siinkohal tutvustame järgmisi.

Käsu USE (vt. lk. 42) täitmisel väärtustatakse erimuutujad &DFIELDS ja &TFIELDS andme- ja tekstiväljade arvuga. Erimuutuja &RSTATUS saab aga väärtuse ENTER, PA3, PF1, ..., PF12 vastavalt sellele, millise klahviga (sisestamisklahviga, klahviga PA3 või ühega funktsionaalklahvidest) jätkati protseduuri tööd pärast paneeli ekraanile väljastamist. Informatsiooni selle kohta, kus viimati asus kursor, saab erimuutujate &RCURSOR ja &RCURSOROFFSET vahendusel. Neist esimene sisaldab kas vastava paneelivälja nime või tühiku (kui kursor polnud ühelgi väljal), teine aga nihke vastava välja alguse või absoluutnihke ekraani alguse suhtes.

Järgnevalt vaatleme käske ja alamkäske, mida läheb tarvis paneeli kasutavas protseduuris.

Paneelisüsteemi initsialiseerimiseks tuleb protseduuri alguses näidata operandideta PTS-käsk EUDEXEC2.

Selleks, et luua side paneelisüsteemi ja EXEC2-interpretaatori vahel peab enne paneelisüsteemi alamkäskude kasutamist protseduuris kirjutama direktiivi

&PRESUME &SUBCOMMAND DISPLAY

Enne paneeli ekraanile väljastamist tuleb ta eelnevalt kirjeldada alamkäsuga, mille üldkuju on

USE [PANEL] paneelinimi

Varemkirjeldatud paneeli väljastab ekraanile operandideta alamkäsk DISPLAY, mis kannab paneeli vastavatele väljadele ka erimuutujate &D1, &D2, ... hetkeväärtused.

Pärast paneeli väljastamist saab protseduuri täitmist jätkata vajutusega ühele juba eespool nimetatud klahvidest. Jätkamisel väärtustatakse ühtlasi ka erimuutujad &RSTATUS, &RCURSOR ja &RCURSOROFFSET, paneeli andmeväljadel olev info aga omistatakse vastavalt erimuutujatele &D1, &D2, ...

Alamkäsuga CURSOR võib enne paneeli ekraanile väljastamist määrata kursori asukoha ekraanil. Alamkäsu üldkuju on

$$\text{CURSOR} \begin{bmatrix} \text{väljanimi [nihe]} \\ \text{absoluutnihe} \end{bmatrix}$$

Siin operandina näidatakse kas vastava välja nimi (näiteks D3, T1) ja tarbe korral ka nihe välja alguse suhtes või siis absoluutnihe ekraani alguse suhtes. Operandi puudumisel pannakse kursor esimese kaitsmata (NOPROTECT) andmevälja algusesse, kui sellist pole, siis ekraani vasakusse ülanurka.

Töötamise varem alamkäsuga USE kirjeldatud paneeliga lõpetab operandideta alamkäsk TERMinate.

3.4. Näiteid paneelide kasutamise kohta

Paneelide kasutamise lihtsama näitena esitame alljärgnevalt protseduuri nimega XYCOPYP. Nii selle protseduuri juhendi esitamiseks kui ka vajalike andmete sisestamiseks moodustame paneeli nimega XYCOPAN, mille kirjeldus üldjoontes (arvestades trükitehnilisi moonutusi) on järgmine:

```
]**** . . . P R O T S E D U U R  " X Y C O P Y P " . . . ***

        ]PROTSEDUUR "XYCOPYP" KOPEERIB YHEST A-KETTA FAILIST
        ]TEISE KOIK KIRJED ALATES SQNET 'X' SISALDAVAST KUNI
        ]SQNET 'Y' SISALDAVANI (VIIMANE KAASA ARVATUD)
                ]VALIGE NQUTUD ANDMED JA VAJUTAGE SISESTAMIS-
                ]KLAHVILE (LQPETAMISEKS KLAHVILE "PF3"):

        ]KOPEERITAVA FAILI NIMI: _########
        ]KOPEERITAVA FAILI TYYP: _########

]SQNE 'X': _### . . . . . .                                    ##
]SQNE 'Y': _### . . . . . .                                    ##

                ]UUE FAILI NIMI: _########

]********** . . . . . .                                        ***
```

Protseduur XYCOPYP organiseerib selle paneeli abil andmete etteandmise lk. 36-37 toodud näiteprotseduurile XYCOPY, mis teatavasti kopeerib ühest A-ketta failist teise kirjed

alates sõnet x sisaldavast kuni sõnet y sisaldavani (viimane kaasa arvatud). Protseduuri käivitamisel tuleb näidata kopeeritava faili nimi ja tüüp, vastavad sõned ning uue faili nimi. Paneelil valitavad andmed omistatakse teatavasti automaatselt erimuutujatele &D1, &D2, &D3, &D4 ja &D5, mida ongi kasutatud parameetrite edastamiseks PTS-käsuga EXEC väljakutsutavale (meile juba tuttavale) protseduurile XYCOPY.

Protseduri XYCOPYP tekst on järgmine:

```
&TRACE
-DISP1
&COMMAND EUDEXEC2
&PRESUME &SUBCOMMAND DISPLAY
USE PANEL XYCOPAN
-DISP
DISPLAY
&IF &RC = 0 &GOTO -OK
&EXIT 10&RC
-OK
&IF . &RSTATUS = . PF3 &EXIT
&IF . &RSTATUS = . ENTER &GOTO -ENT
&GOTO -DISP
-ENT
&COMMAND EXEC XYCOPY &D1 &D2 &D3 &D4 &D5
&IF &RC = 0 &GOTO -DISP1
&EXIT 100&RC
```

Näitena paljude paneelide kasutamisest terves protseduuridesüsteemis soovitame tutvuda TÜ Arvutuskeskuses välja töötatud dialoogmonitori PTS kasutamise süsteemiga XAJEX.

Süsteem asub käesoleval ajal virtuaalarvuti CMS virtuaalkettal aadressiga 319. Kui vastava tarbija virtuaalarvuti eelprotseduur (fail PROFILE EXEC A) ei sisalda nimetatud virtuaalketta hõivamist, siis tuleb terminaalilt sisestada (või tarbe korral eelprotseduuri lülitada) järgmised käsud:

```
CP LINK CMS 319 319 RR RC
AC 319 P
```

Edasi võib süsteemi käivitamiseks käsureal valida lihtsalt süsteemi põhiprotseduuri nime XAJEX, mille tulemusena ilmub ekraanile põhimenüü (süsteemi automaatseks käivitamiseks tuleb eelprotseduuri lõppu lisada PTS-käsk EXEC XAJEX). Kogu järgnev töö toimub paneelide vahendusel dialoogina.

Mitmeid süsteemi XAJEX põhimenüüs esinevaid tegevusi saab käivitada ka autonoomselt, s.t. väljaspool põhiprotseduuri. Esitamegi allpool tähtsamate autonoomselt (enamasti nime valimisega käsurealt) käivitatavate alamprotseduuride loetelu koos lühiseletusega:

XAJEXSYS - informatsioon virtuaalarvuti kohta;

XAJEXPTS - dialoogmonitori PTS töörežiimide ja funktsioonide kontrollimine/muutmine;

XAJEXNF - uue faili moodustamine terminaalilt sisestatavatest andmetest;

XAJEXOS - OS-andmekogumite ülekandmine PTS-failideks;

XAJDISK - virtaalketaste kontrollimine, hõivamine ning vabastamine;

XAJEX2FL - virtuaalketta kõikide failide loendi saamine koos menüüga (võib anda ka koos parameetritega fn, ft, fr (vaikimisi A1) nii nagu see toimub PTS-käsu FLIST korral).

4. P R O T S E D U U R I D E K E E L R E X X

4.1. Põhimõisted

Keeles REXX (= Restructured EXtended eXecutor) kirjutatud protseduur, samuti nagu keeles EXEC2 kirjutatud protseduurid, tuleb vormistada failina tüübinimega EXEC. Ühine on nendele keeltele ka see, et protseduure interpreteeritakse lausete kaupa. Keeles REXX kirjutatud protseduur peab algama kommentaariga, mis kujutab endast sümbolitepaaridega /* ja */ ümbritsetud sõnet (võib hõivata ka mitu rida).

Protseduuridekeel REXX kasutab järgmisi sümboleid:

tähed: A - Z ja a - z; numbrid: 0 - 9;

piirajad: + - * / % ! & = ¬ > < .

: ; ( ) ' # ¤ , " _ ? ␣

REXX-keelne protseduur on lausete järjend, kus iga lause on omakorda lekseemide järjend. Kasutatatakse järgmisi lekseemitüüpe: kommentaar, sõne, kuueteistkümnendsõne, nimi, tehtemärk ja eraldaja. Iseloomustame neid tüüpe lähemalt.

Sõne kujutab endast sümbolite järjendit pikkusega kuni 250 sümbolit, mis on ümbritsetud apostroofide või jutumärkidega. Lubatud on kasutada ka tühja sõnet (sõne pikkus võrdub nulliga). Kui sõnele vahetult järgneb täht X (või x), siis on tegemist kuueteistkümnendsõnega. Kui aga sõnele vahetult järgneb avav sulg, siis loetakse seda sõnet funktsiooni nimeks. Sõnedeks on näiteks järgmised kirjutised:

'DATE' ehk "DATE" "YTLE: ""TERE""" ehk 'YTLE: "TERE"'

'PARM="PARAMEETRID"' "ПОД'ЕЗД" ehk 'ПОД''ЕЗД'

(paneme tähele, et apostroofidega ümbritsetud sõnes tuleb iga sinna kuuluv apostroof kujutada kahe järjestikuse apostroofina; analoogiliselt toimitakse jutumärkidega).

Järgmised kirjutised osutuvad kuueteistkümnendsõnedeks:

'ABCD'X "1aF2E"x "3 4E 5f"X

(viimases näites on kuueteistkümnendnumbrid lugemise hõlbustamiseks lõpust alates paarikaupa tühikuga eraldatud).

Nimedeks on tähtedest, numbritest ja erisümbolitest #, @, allkriips ning punkt koosnevad sõned maksimaalpikkusega 250 sümbolit. Interpretaator tõlgendab kõik nimedes esinevad väiketähed suurtähtedena. Nime võib kasutada kas märgendina, keele REXX võtmesõnana või konstandina. Kui aga nimel ühtki nimetatud tähendustest ei ole, siis on tegemist muutujaga, millele võib omistada väärtusi. Kui muutujale pole väärtust omistatud (või see on tühistatud), siis on muutuja algväärtuseks tema nime moodustav sõne.

Protseduuridekeel REXX sisaldab lisaks tavalistele muutujatele veel kolm erimuutujat, millele omistatakse väärtusi teatud käskude või direktiivide täitmise tulemusena. Mõningase ettevaatusega võib erimuutujaid küll kasutada ka tavaliste muutujatena, kuid seda pole soovitatav siiski teha, sest nii läheksid kaotsi nende erimuutuja-omadused.

Erimuutuja RC väärtusena säilitatakse viimasena täidetud (alam)käsu (aga ka direktiivi SIGNAL ON SYNTAX) lõpukoodi. Erimuutuja RESULT väärtuseks omistatakse tagastatava avaldise väärtus alamprotseduurist naasmisel (see avaldis kuulub võtmesõnaga RETURN või EXIT algava direktiivi koosseisu). Erimuutuja SIGL väärtusena säilitatakse REXX-keelse

protseduuri reanumber, kust toimus viimane suunamine märgendi järgi (kas direktiiviga SIGNAL või pöördumisena sisemise alamprotseduuri poole).

Nimed jaotatakse konstantideks ning liht- ja liitnimedeks. Konstant on kirjutis, mis algab numbri või punktiga ja mille väärtust ei saa muuta. Konstandi ees võib veel olla ka märk (+ või -). Konstandid on näiteks järgmised kirjutised:

+37 827.23 12E-5 -.88

Lihtnime esimene sümbol ei tohi olla number ning ta ei tohi sisaldada oma koosseisus punkti. Liitnimi ei alga samuti numbri ega punktiga, kuid ta peab sisaldama vähemalt ühe punkti. Liitnime osa esimese punktini (kaasa arvatud) nimetatakse nime juureks. Juurele järgnevad kas märgita konstandid, lihtnimed või tühinimed (pikkusega null), mis omavahel eraldatakse punktidega. Liitnimed on näiteks kirjutised:

DATA.3 ARRAY.I.J ARBIT_FILE..ONE.2

Kui protseduur töötleb liitnime, siis alati toimub juurele järgnevas osas sisalduvate muutujate asendamine nende väärtustega. Liitnimed on tõlgendatavad massiivielementidena, ehkki massiivi REXX-keel kirjeldada ei võimalda. Olgu märgitud, et uusi väärtusi võib omistada mitte ainult liht- ja liitnimele, vaid ka nime juurele. Viimasel juhul toimub sama väärtuse omistamine kõigile nendele antud hetkel eksisteerivatele liitnimedele, millel on näidatud ühine juur.

Keeles REXX saab sooritada mitmesuguseid tehteid: konkatenatsiooni, võrdlusi, aritmeetilisi ja loogilisi tehteid.

Konkatenatsiooni tehtemärgina kasutatakse kas tühikut või kaht järjestikust hüüumärki. Esimesel juhul toimub kahe

sõne ühendamine selliselt, et nendevaheline tühik säilib, teisel juhul aga mitte. Näiteks konkatenatsioonitehete

'QUERY' 'DISK' ja 'ABC' !! 'D'

tulemusteks on vastavalt sõned 'QUERY DISK' ja 'ABCD'.

Aritmeetiliste tehete märke on seitse:

+ liitmine; / jagamine;

- lahutamine; % täisarvuline jagamine;

* korrutamine; // jäägi leidmine.

** astendamine;

Liitmis- ja lahutamistehted on kasutatavad nii unaarsetena kui ka binaarsetena, ülejäänud tehted ainult binaarsetena. Märgime, et kui jäägi leidmisel on jagatava väärtus negatiivne, siis tuleb jääk mittepositiivne. Astendamistehe on lubatud vaid täisarvulise astendaja korral ning jäägi leidmisel peavad mõlemad operandid olema täisarvulised.

Võrdlustehete tulemuseks on tõeväärtus "tõene" või "väär", mida REXX-keele interpretaator tõlgendab vastavalt täisarvuna 1 või 0. Kasutada tohib järgmisi võrdlustehteid:

= võrdne; > suurem; < väiksem;

¬= ehk /= ehk <> ehk >< mittevõrdne;

>= ehk ¬< suurem-võrdne ehk mitte väiksem;

<= ehk ¬> väiksem-võrdne ehk mitte suurem.

Enne nende tehete sooritamist eemaldatakse operandidest äärmised tühikud ja seejärel võrdsustatakse operandide pikkused, lisades lühema operandi lõppu vajaliku arvu tühikuid. On aga veel kaks sõnede võrdlemiseks mõeldud võrdlustehet, kus eelnevalt operandide pikkusi ei võrdsustata ega ei eemaldata äärmisi tühikuid. Nendeks teheteks on järgmised:

== sõned on täielikult identsed;

¬== ehk /== sõned pole identsed.

Loogiliste tehete märgid on järgmised (operandidel tohivad olla vaid täisarvulised väärtused 0 ja 1):

& loogiline korrutamine;

| loogiline liitmine;

&& mitteekvivalents;

¬ eitus.

Neist tehetest esimesed kolm on binaarsed, viimane unaarne.

Tehete jaoks on keeles kehtestatud kindel prioriteet: kõrgema prioriteediga tehted sooritatakse madalama prioriteediga tehetest varem (kui sulgude lisamisega ei ole teisiti määratud). Tehteid sooritatakse järgmises prioriteetide kahanemise järjekorras:

1) unaarsed tehted (+, -, ¬);

2) astendamine (**);

3) korrutamine ja jagamised (*, /, %, //);

4) liitmine ja lahutamine (+, -);

5) konkatenatsioon (tühik, ||);

6) võrdlemised (vt. lk. 49);

7) loogiline korrutamine (&);

8) loogiline liitmine ja mitteekvivalents (|, &&).

Lekseemide eraldajatena on REXX-keeles kasutusel järgmised sümbolid: koolon, semikoolon, ümarsulud ja koma. Samuti on aga eraldajateks ka kõik tehtemärgid, välja arvatud arvu järgus esinev märk (+ või -). Peaks olema endastmõistetav, et nimetatud sümbolid loetakse eraldajateks muidugi ainult siis, kui nad ei kuulu ühegi sõne koosseisu.

4.2. Laused

Protseduuridekeeles REXX on kasutusel järgmised laseteliigid: tühilaused, märgendid, omistamislaused (ehk omistamisdirektiivid), direktiivid ja käsud ning alamkäsud. Üldiselt võib protseduuri ühele reale kirjutada kas ühe või ka mitu lauset, kusjuures reeglina peab iga lause lõppema semikooloniga. Real ainsana (või viimasena) paikneva lause lõpus võib aga semikooloni kirjutamata jätta (samuti märgendi järele semikoolonit ei panda). Sageli ei mahu pikemad laused ühele reale ära - sellisel juhul võib lause paikneda mitmel järjestikusel real, kusjuures jätkamise tunnusena tuleb jätkatava rea viimase sümbolina lisada koma.

Iseloomustame lauseid liikide kaupa lähemalt. Tühilause koosneb tühikutest ja/või kommentaaridest. Protseduuri täitmisel jäävad tühilaused vahele, kuid protseduuride silumisel võidakse ka nende kohta informatsiooni väljastada.

Märgend kujutab endast lihtnime, millele järgneb koolon (semikoolonit pole vaja lisada). Märgendada tuleb kõik sisemised alamprotseduurid ja samuti need laused, millele toimub suunamine direktiivi SIGNAL abil. Ühel lausel tohib vajaduse korral olla ka mitu märgendit.

Omistamislause tuleb protseduuris kirjutada kujul

```
muutuja = avaldis;
```

Siin tohib muutuja kohal olla kas liht- või liitnimi, aga ka liitnime juur (vt. lk. 48). Omistamislausete kasutamist selgitab järgmine protseduurikatkend (kus selgitused on kirjutatud omistamislausete järele kommentaaridena):

```
A=3;  B=4;  C='REXX'
A.B=C            /* A.4='REXX'                    */
A.REXX=5         /* A.REXX=5                      */
A.C='PROC'       /* A.REXX='PROC'                 */
C.C=A.REXX       /* C.REXX='PROC'                 */
X.A.B='FAIL'     /* X.3.4='FAIL'                  */
A.='END'         /* A.4='END' ja A.REXX='END'     */
```

REXX-keele direktiiv on teatava võtmesõnaga algav lause, mis võib omakorda sisaldada veel teisi võtmesõnu. Mõned direktiivid (niisugusteks nn. liitdirektiividekс on võtmesõnadega IF, DO ja SELECT algavad) võivad omakorda sisaldada teisi direktiive. Lähemalt käsitleme direktiive punktis 4.3.

Keeles REXX täidetakse käskudena PTS-käske, CP-käske ja käsu XEDIT alamkäske, kusjuures nende võtmesõnad tuleb vormistada sõnedena. Kõigi käskude täitmisel välja töötatav lõpukood omistatakse erimuutuja RC väärtuseks. Kui käsu parameetrid sisaldavad spetsiaalsümboleid (tärn, võrdusmärk, sulud jne.), siis tuleb nad vormistada sõnedena, sest muidu tõlgendataks neid lekseemidena (näiteks tärni korrutamismärgina). Ka konstantidena võetavad nimed tuleb vormistada sõnedena, vastasel juhul tõlgendataks neid muutujate nimedena.

Üldiselt üritab REXX-interpretaator igat käsku kõigepealt täita protseduurina (EXEC-tüüpi failina). Kui aga seda ei soovita (näiteks töökiiruse huvides), siis tuleb failide läbivaatamise järjekord muuta direktiiviga ADDRESS.

Alamprotseduurid võib jaotada sisemisteks ja välisteks. Sisealamprotseduur on REXX-keelse protseduuri koostisosaks, ta algab alamprotseduurile nime andva märgendiga ning lõpeb

võtmesõnaga RETURN seotud direktiiviga. Välisalamprotseduur kujutab endast iseseisvat protseduuri (EXEC-tüüpi faili).

Alamprotseduuri poole on võimalik pöörduda ühel kahest võimalikust viisist: alamprogrammina ja funktsioonina. Alamprogrammina pöördumine toimub direktiiviga, mis algab võtmesõnaga CALL ja millele järgneb alamprotseduuri nimi ning vajaduse korral veel üleantavate parameetrite loetelu. Funktsioonina pöördumine toimub avaldise koosseisus: tuleb näidata alamprotseduuri nimi, millele vahetult järgneb parameetrite loetelu (võib olla ka tühi!) sulgudes.

Olgu märgitud, et viimati kirjeldatud viisil pöördutakse ka standardfunktsioonide poole, loomulikult silmas pidades konkreetse standardfunktsiooni parameetrite tähendust.

Üldiselt hakatakse alamprotseduuri otsima sisealamprotseduuride hulgast ja leitud alamprotseduur võetakse ka täitmisele. Kui näidatud nimega sisealamprotseduuri ei leidu, siis jätkatakse otsimist välisalamprotseduuride hulgast. Kui aga töökiiruse huvides soovitakse konkreetse välisalamprotseduuri otsimist sisealamprotseduuride hulgast vahele jätta, siis tuleb selle alamprotseduuri nimi lihtsalt ümbritseda apostroofide või jutumärkidega. Olgu märgitud, et välisalamprotseduuri poole pöördumine (alamprogrammina) on teostatav ka käsu EXEC abil, näidates alamprotseduuri nime koos üleantavate parameetrite loeteluga.

Protseduuridekeel REXX võimaldab muuhulgas pöörduda ka ASSEMBLER-keeles või mõnes muus keeles kirjutatud välisprotseduuride poole (nii alamprogrammina kui ka funktsioonina). Lähemalt on neid küsimusi tutvustatud käsiraamatus [1].

4.3. Direktiivid

REXX-keele direktiiv on lause, mille esimene lekseem on teatud võtmesõna ja teine lekseem pole võrdusmärk (sest muidu oleks tegemist omistamislausega) ega koolon (muidu oleks tegemist märgendiga). Paljude direktiivide koosseisu võib kuuluda veel teisigi võtmesõnu. Võtmesõnade kirjutamiseks tohib kasutada nii suur- kui ka väiketähti - interpretaator teisendab kõik väiketähed suurtähtedeks. Võtmesõnadega IF, SELECT ja DO algavad direktiivid on liitdirektiivid, mis sisaldavad oma koosseisus veel teisi direktiive.

Käskude või alamkäskude täitmiseks vajaliku keskkonna saab kehtestada direktiiviga, mille üldkuju on

$$\text{ADDRESS} \begin{bmatrix} \text{keskk [avi]} \\ \text{[VALUE] av2} \end{bmatrix};$$

Siin parameeter keskk on nimi või sõne, mis määrab vajaliku keskkonna, avi aga avaldis, mille väärtus täidetakse selle keskkonna (alam)käsuna. Pärast käsu täitmist taastatakse senine keskkond, kui aga avi puudub, siis lihtsalt kehtestatakse näidatud keskkond. Võtmesõnale VALUE järgnev parameeter av2 on avaldis, mille väärtus näitab kehtestatava keskkonna nime (kui av2 on sulgudes, siis võib VALUE ära jätta). Kõigi parameetrite puudumisel taastatakse see keskkond, mis kehtis enne eelmist direktiivi ADDRESS.

Direktiiv ADDRESS võib kehtestada järgmised keskkonnad:

'PTS' - PTS-käske hakatakse otsima protseduuride (EXEC-tüüpi failide) hulgast, mitteleidmisel aga käskudest (direktiivi ADDRESS puudumisel võetakse see keskkond vaikimisi);

'COMMAND' - PTS-käske hakatakse esmalt otsima moodulite (MODULE-tüüpi failide) hulgast, mitteleidmisel aga käskude hulgast; samanimelise protseduuri või CP-käsu täitmiseks tuleb neile lisada vastavalt prefiks EXEC või CP;

'XEDIT' - kehtestatakse käsu XEDIT alamkäskude keskkond.

Märgime, et alamprotseduuri koosseisu kuuluv direktiiv ADDRESS ei mõjuta väljakutsuva protseduuri täitmist. Kehtiva keskkonna nime saab teada standardfunktsiooniga ADDRESS.

Arvude kujutamist ja aritmeetiliste avaldiste arvutamise täpsust juhib direktiiv üldkujuga

$$\text{NUMERIC} \left\{ \begin{array}{l} \text{DIGITS [avi]} \\ \text{FUZZ [av2]} \\ \text{FORM} \left[\begin{array}{l} \underline{\text{SCIENTIFIC}} \\ \text{ENGINEERING} \end{array} \right] \end{array} \right\};$$

Siin avaldise avi väärtus määrab tüvenumbrite maksimaalarvu arvutustulemustes (vaikimisi 9). Võtmesõnale FUZZ järgneva avaldise av2 väärtus näitab, kui mitme tüvenumbri võrra tuleb võrdlustehetes vähendada aritmeetiliste avaldiste arvutamise täpsust võrreldes avaldise avi väärtusega (vaikimisi on selleks väärtuseks null). Võtmesõna FORM toimel esitatakse arvutustulemused eksponentkujul, kusjuures võtmesõnaga SCIENTIFIC (ja vaikimisi) määratakse mantissi täisosaks täpselt üks nullist erinev number (erandiks on muidugi nulliga võrduv arv), võtmesõnaga ENGINEERING määratakse aga kolmega jaguv järk. Alamprotseduuris esinev direktiiv NUMERIC ei mõju väljakutsuvale protseduurile ja vastupidi.

Väiketähed teisendab suurtähtedeks direktiiv

```
UPPER nimi1 [nimi2 [nimi3 ...]];
```

Selles direktiivis näidatud muutujate väärtustes asendatakse väiketähed vastavate suurtähtedega. Seejuures võivad loetellu kuuluda nii liht- kui ka liitnimed, aga mitte juured. Kui loetelu mingi element pole muutuja nimi, siis suurtähtedeks ei teisendata ning kui on sisse lülitatud režiim SIGNAL ON NOVALUE, siis tekib situatsioon NOVALUE (vt. lk. 60).

Muutujate väärtuste tühistamise direktiiv on üldkujuga

DROP nimi1 [nimi2 [nimi3 ...]];

Väärtuse tühistamisel saab iga näidatud muutuja väärtuseks tema vaikimisi võetav algväärtus (sõne, mille moodustab muutuja nimi). Seejuures, kui direktiivis näidatakse nime juur, siis tühistatakse kõigi selle juurega muutujate väärtused.

Muutujate väärtusi sisestakse direktiiviga

PARSE [UPPER] { ARG | EXTERNAL | NUMERIC | PULL | SOURCE | VALUE [avaldis] WITH | VAR muutuja | VERSION } [šabloon];

Selles üldkujus on parameetritel järgmine tähendus:

UPPER - ühtlasi teisendatakse väiketähed suurtähtedeks;

ARG - analüüsida protseduurile antavaid parameetreid;

EXTERNAL - analüüsida puldipuhvrist sisestatavaid andmeid; kui aga puhvrist sisestada pole, siis tuleb andmed sisestada puldilt (puhvris sisalduvate ridade arvu saab teada standardfunktsiooni EXTERNALS abil);

NUMERIC - analüüsida andmeid direktiivi NUMERIC jaoks: esimene, teine ja kolmas väärtus vastavad direktiivi NUMERIC võtmesõnadele DIGITS, FUZZ ja FORM (näiteks võib sisestatavate sümbolite järjendiks olla: 9 0 SCIENTIFIC);

PULL - analüüsida programmimagasinist sisestatavaid andmeid; kui aga magasin on tühi, siis toimitakse vastavalt parameetrile EXTERNAL;

SOURCE - analüüsida täidetava protseduuri karakteristikuid järgmise formaadi kohaselt:

$$\text{PTS}\left\{\begin{matrix}\text{COMMAND}\\ \text{FUNCTION}\\ \text{SUBROUTINE}\end{matrix}\right\}\left\{\begin{matrix}\text{fn ft fr}\\ * \ *\end{matrix}\right\}\left\{\begin{matrix}\text{nimi}\\ ?\end{matrix}\right\}\text{keskk}$$

siin teine komponent määrab pöördumisviisi: COMMAND - tegemist põhiprotseduuriga; FUNCTION - pöörduti funktsioonina; SUBROUTINE - pöörduti alamprogrammina; komponendid fn, ft ja fr määravad protseduuri sisaldava faili (kui failitüüp ja -režiim pole määratavad, siis tärnid); komponent nimi tähendab protseduuri väljakutsumisel näidatud nime, näiteks failinime sünonüümi (kui aga seda nime määrata ei saa, siis küsimärk); komponent keskk näitab, millisest keskkonnast protseduuri poole pöörduti;

VALUE [avaldis] WITH - analüüsida antud avaldist (mille koosseisu ei tohi kuuluda muutuja nimega WITH);

VAR muutuja - analüüsida näidatud muutuja väärtust;

VERSION - analüüsida REXX-interpretaatori poolt moodustatud sõnet, mis käesoleva juhendi koostamise ajal on kujul:

... 3.20 14 Jan 1983

(interpretaatori versiooni number ja moodustamise kuupäev);

šabloon - omavahel trafarettidega (tühikud, sõned, arvud) eraldatud muutujate nimed; neile muutujaile omistatakse analüüsimisel saadud väärtused (täpsemalt vt. lk. 90).

Märgime, et direktiivide PARSE UPPER ARG [šabloon] ja PARSE UPPER PULL [šabloon] asemel võib kasutada vastavalt direktiive ARG [šabloon] ja PULL [šabloon].

Avaldise sõnena tõlgendatava väärtuse kirjutamine programmimagasini kõrgeimasse tasemesse (vt. lk. 26) otsejärjestuses FIFO (s.o. magasini lõppu) toimub direktiiviga

```
QUEUE [av];
```

Kirjutatava avaldise av pikkus ei tohi olla suurem kui 255 sümbolit; avaldise puudumisel kantakse magasini tühi rida.

Selle direktiiviga väga sarnane on direktiiv üldkujuga

```
PUSH [av];
```

Erinevus seisneb üksnes selles, et kirjutamine toimub nüüd pöördjärjestuses LIFO (s.o. magasini algusse).

Andmete väljastamine ekraanile toimub direktiiviga

```
SAY [av];
```

Väljastatava avaldise av väärtus võib olla suvalise pikkusega (vajaduse korral väljastatakse mitmele reale). Avaldise puudumisel väljastatakse tühi rida. Näiteks direktiivide

```
DATA=100; SAY DATA 'JAGADA 4-GA =' DATA/4;
```

toimel kantakse ekraanile tekst: 100 JAGADA 4-GA = 25 .

Alamprotseduuri poole alamprogrammina pöördumiseks kasutatakse direktiivi üldkujuga

```
CALL alampr [av1 [av2 ... av10]];
```

Siin parameetrid av1, ..., av10 on avaldised, mille väärtused väljakutsutavale protseduurile üle antakse (seal on nad

kättesaadavad direktiiviga ARG või PARSE ARG). Näidatud nimega alampr alamprotseduuri otsimist alustatakse sisealamprotseduuride hulgast ja leidmisel antakse talle juhtimine. Kui sisealamprotseduuride hulgas otsitavat ei leitud, siis jätkub otsimine välisalamprotseduuride hulgast. Kui parameeter alampr vormistada sõnena, siis alustatakse alamprotseduuri otsimist kohe välisalamprotseduuride hulgast, mis võib kiirendada alamprotseduuri leidmist. Märgime veel, et välisalamprotseduuri poole saab pöörduda ka PTS-käsuga EXEC.

Protseduuri täitmine lõpetatakse direktiiviga

EXIT [av];

Aritmeetilise avaldise av väärtus saab ühtlasi selle protseduuri lõpukoodiks (avaldise puudumisel on lõpukood null).

Alamprotseduuri töö lõpetamiseks ja väljakutsunud protseduuri naasmiseks võib kasutada direktiivi

RETURN [av];

Direktiivi täitmisel arvutatakse ühtlasi avaldise av väärtus ja omistatakse erimuutuja RESULT väärtuseks (avaldise puudumisel omistatakse sõne 'RESULT'). Põhiprotseduuris on direktiiv RETURN samaväärne direktiiviga EXIT. Kui alamprotseduuri poole pöörduti funktsioonina, siis ei tohi avaldist ära jätta, sest tema annabki funktsiooni leitava väärtuse.

Sisealamprotseduuri ja väljakutsuva protseduuri ühiste muutujate loetelu määratakse direktiiviga

PROCEDURE [EXPOSE nimi1 [nimi2 [nimi3 ...]]];

Nimede loetelu näitabki, millistele muutujatele tuleb pöördumisel garanteerida juurdepääs ka selles alamprotseduuris. Kui nime asemel näidata nime juur, siis on sisealamprotse-

duuris kättesaadavad kõik selle juurega liitnimed. Direktiiv PROCEDURE on soovitatav panna vahetult alamprotseduuri nime määrava märgendi järele. Selle direktiivi puudumisel on kõik alamprotseduuri muutujad kättesaadavad põhiprotseduuris ning vastupidi. Direktiivi kasutamist illustreerib näide:

```
/* V@LJAKUTSUV PROTSEDUUR */
J=1; X.1='A'; CALL PROC1;
SAY J K M;    /* V@LJASTATAKSE SYMBOLID: 1 7 M */
EXIT;
/* ALAMPROTSEDUUR */
PROC1: PROCEDURE EXPOSE J K X.J
SAY J K X.J;    /* V@LJASTATAKSE SYMBOLID: 1 K A */
K=7; M=3; RETURN;
```

Tingimusteta suunamiseks ja protseduuri täitmise käigus tekkivatele situatsioonidele reageerimiseks tuleb kasutada direktiivi üldkujuga

```
SIGNAL { mnimi                                          }
       { [VALUE] av                                     }
       { { ON  } { NOVALUE } }
       { { OFF } { SYNTAX  } }
       { { ON  } { ERROR   } }
       { { OFF } { HALT    } } ;
```

Parameeter mnimi teatab selle märgendi, kuhu tuleb juhtimine üle anda (märgendit otsitakse protseduuri algusest alates). Parameeter av on avaldis, mille väärtus annab nimetatud märgendi (kui avaldise esimeseks lekseemiks pole sõne ega nimi, siis võib võtmesõna VALUE ära jätta). Võtmesõnad ON või OFF tähendavad vastavalt, et järgneva võtmesõnaga näidatud situ-

atsiooni "püüdmine" kas lülitatakse sisse või tühistatakse. Situatsioon NOVALUE tekib siis, kui arvutatavas avaldises, direktiivis UPPER või direktiivis PARSE võtmesõna VAR järel esineb nimi, mis pole muutuja nimi. Situatsioon SYNTAX tekib juhul, kui protseduuri täitmisel avastatakse REXX-keele süntaksiviga. Situatsioon ERROR on PTS-käsu või CP-käsu täitmise lõppemine nullist erineva lõpukoodiga. Protseduuri täitmise kohene katkestamine käsuga HI tekitab situatsiooni HALT (vaikimisi on kõigi situatsioonide "püüdmine" tühistatud).

Juhul, kui protseduuri täitmisel avastati üks nimetatud neljast situatsioonist, antakse juhtimine selle situatsiooniga samanimelise märgendiga varustatud lausele (märgendit otsitakse protseduuri algusest alates). Kui vastavat märgendit ei leitud, siis toimub selle protseduuri töö avariilõpetamine koos vastava veateate väljastamisega.

Situatsiooni avastamisel omistatakse vigase rea number ühtlasi erimuutuja SIGL väärtuseks, mida saab kasutada vea lokaliseerimiseks. Lisaks omistatakse situatsioonide ERROR ja SYNTAX korral veakood ka erimuutujale RC.

Et situatsioonide avastamisel oleks välditud protseduuri tsüklisse jäämine, siis pärast mingi situatsiooni avastamist tema "püüdmine" automaatselt tühistatakse ("püüdmise" taastamiseks tuleb see uuesti sisse lülitada).

Märgime veel, et alamprotseduuris olev direktiiv SIGNAL ei mõju väljaspool seda. Protseduuri dialoogsilumisel (direktiiviga TRACE) situatsioonide "püüdmine" tühistatakse.

Hargnemiste organiseerimiseks protseduuris saab kasutada liitdirektiivi üldkujuga

IF av [;] THEN [;] lausei [ELSE [;] lause2];

Kui selles direktiivis esineva loogilise avaldise av väärtus võrdub ühega, siis täidetakse lausei (millega ka direktiivi täitmine lõpeb), vastasel juhul täidetakse lause2 (kui see on olemas). Üldiselt võib lausei või lause2 olla suvaline, aga mitte tühi. Ainult semikoolonist koosneva tühilause asemel kasutatakse parameetriteta direktiivi NOP. Kui lausei ja lause2 on ühel real, siis peab lausei lõppema semikooloniga.

Tsüklite täitmist on kõige mugavam organiseerida liitdirektiiviga, mille üldkuju on

```
        [nimi:avi [TO av2] [BY av3] [FOR av4]]
        [FOREVER                              ]
DO      [av5                                  ];
        [WHILE av6                            ]
        [UNTIL av7                            ]
    [lausete järjend]
END [nimi];
```

Parameeter nimi:avi määrab siin tsükliloendaja nime ja temale algväärtust omistava avaldise avi. Võtmesõnale TO järgnev avaldis av2 määrab tsükliloendaja lõppväärtuse (kui see puudub, siis tuleb tsükli täitmine lõpetada teisiti). Võtmesõnale BY järgnev avaldis av3 määrab tsükliloendaja muutumise sammu (ka negatiivne samm on lubatud; vaikimisi võrdub samm ühega). Võtmesõnale FOR järgnev avaldis av4 määrab tsükli täitmiskordade maksimaalarvu (juhul, kui tsükkel ei lõpe juba varem mõne muu tingimuse järgi).

Parameetri FOREVER puhul on formaalselt tegemist lõputu tsükliga (tsüklist saab siiski väljuda direktiivide LEAVE ja

SIGNAL kaasabil). Avaldis av5 määrab tsükli täitmiskordade maksimaalarvu sel juhul, kui puudub tsükliloendaja.

Võtmesõnale WHILE järgneva loogilise avaldise av6 väärtus juhib tsükli kordamist järgmiselt: kui see väärtus on ükS, siis tsüklit korratakse, kui aga väärtus on null, siis kordamine lõpetatakse. Teatud mõttes vastupidine on võtmesõnale UNTIL järgneva loogilise avaldise av7 tähendus: kui tema väärtus on null, siis tsüklit korratakse, kui aga väärtus saab üheks, siis loetakse tsükkel ammendatuks.

Tsüklidirektiivide kasutamisel tuleb veel arvestada, et võtmesõnadega TO, BY ja FOR algavate konstruktsioonide omavaheline järjekord pole oluline. Kui võtmesõna DO järel puuduvad tsükli täitmist juhtivad lekseemid, siis on tegemist direktiivsulgudega DO; ...; END; (kus punktiiriga tähistatud protseduurilõik täidetakse ühekordselt). Niiviisi saab mistahes lausete järjendi muuta üheks lauseks ja kasutada seda seal, kus mitme lause kasutamine pole lubatud (näiteks direktiivis IF või SELECT). Võtmesõnale END järgnev nimi (kui ta on olemas) peab ühtima tsükliloendajaga. Tsükli täitmiskordade arv võib sõltuda ka direktiividest LEAVE ja ITERATE. Võtmesõnu TO, BY, FOR, FOREVER, WHILE ja UNTIL ei lubata kasutada tsüklidirektiivi koosseisus muutujate nimedena (väljaspool tsüklidirektiivi pole see keelatud).

Tsüklidirektiivi koosseisu kuuluvate lausete täitmisjärjekorra muutmist võimaldab direktiiv

ITERATE [nimi];

Parameeter nimi teatab selle tsükliloendaja nime, millisele tsüklile direktiiv mõjub (nime puudumisel mõjub kõige sise-

misemale tsüklile). Direktiivi toimel talle järgnevaid lauseid (kuni tsükli lõpuni) ei täideta, vaid hakatakse tsüklit uuesti kordama. Tuleb aga arvestada, et kui tsüklist toimub pöördumine mingisse alamprotseduuri, siis muutub see tsükkel seniks passiivseks ja direktiiv ITERATE pole seal kasutatav.

Tsükli täitmise katkestamist võimaldab direktiiv

```
LEAVE [nimi];
```

Parameeter nimi teatab selle tsükliloendaja nime, millisest tsüklist väljutakse (nime puudumisel väljutakse kõige sisemisemast tsüklist). Tsüklist väljumisel säilitab tsükliloendaja oma senise väärtuse. Kui aga tsüklis siseneti mingisse alamprotseduuri, siis ei saa seal direktiivi LEAVE kasutada (samuti tsüklis, mis moodustati direktiiviga INTERPRET).

Kui direktiiv IF võimaldab valida kahe lause täitmise vahel sõltuvalt teatava loogilise avaldise väärtusest, siis liitdirektiiv SELECT võimaldab valida suvalise arvu lausete täitmise vahel. Direktiivil on järgmine üldkuju:

```
SELECT;
  WHEN avi [;] THEN [;] lause1;
  WHEN av2 [;] THEN [;] lause2;
  . . . . . .
  WHEN avn [;] THEN [;] lausen;
  [OTHERWISE [;] [lausete järjend];]
END;
```

Selle liitdirektiivi täitmine kulgeb järgmiselt. Järjest arvutades võtmesõnadele WHEN järgnevate loogiliste avaldiste avi, av2, ... väärtusi, leitakse esimene, mis võrdub ühega. Seejärel täidetakse vastavale võtmesõnale THEN järgnev lau-

se, millega direktiivi SELECT täitmine on lõppenud. Kui aga osutub, et kõigi loogiliste avaldiste avi, . . . , avn väärtused võrduvad nulliga, siis täidetakse võtmesõnale OTHERWISE järgnev lausete järjend (juhul muidugi, kui see on olemas).

Võtmesõnade WHEN arv direktiivis SELECT võib olla suvaline. Direktiivi koosseisu tohivad kuuluda üldiselt mistahes laused, sealhulgas ka liitdirektiivid (IF, DO või SELECT), kuid mitte tühilause (võib kasutada direktiivi NOP). Direktiivi SELECT kasutamist illustreerib protseduurilõik:

```
'STATE' fn ft fr;
SELECT;
  WHEN RC = 0 THEN DO;
    'ERASE' fn ft fr;
    SAY 'FAIL LEITI JA EEMALDATI'; END;
  WHEN RC = 28 ! RC = 36 THEN SAY 'FAILI EI LEITUD';
  OTHERWISE; DO;
    SAY 'ETTEN@GEMATU LQPUKOOD' RC 'K@SUS STATE';
    EXIT RC; END;
END;
```

Avaldise tõlgendamist lause(te)na võimaldab direktiiv

```
INTERPRET av;
```

Siin arvutatakse avaldise av väärtus ja saadud sõnet tõlgendatakse ühe või mitme lausena (nagu oleksid laused viimasel juhul ümbritsetud direktiivsulgudega DO; . . . ; END;) ning ka täidetakse. Direktiivi INTERPRET kasutamist illustreerib järgmine protseduurilõik:

```
DATA='ABC'; INTERPRET DATA='4';
/* T@IDETAKSE OMISTAMISLAUSE ABC=4 */
```

Järgnev protseduurilõik väljastab kolm korda ekraanile teksti 'ALUSTAME' ja käivitab siis protseduuri PAKETID:

```
START="ALUSTAME";
DATA='DO 3; SAY START; END; "EXEC PAKETID"';
INTERPRET DATA;
```

REXX-keelse protseduuri silumisel (trasseerimisel) saab kasutada järgmist jälgimisdirektiivi:

```
                  [ [ ]  [All        ] ]
                  [ [ ]  [Commands   ] ]
                  [ [?]  [Errors     ] ]
                  [ [ ]  [Intermediates] ]
                  [ [ ]  [Labels     ] ]
TRACE             [ [ ]  [Negatives  ] ] ;
                  [ [ ]  [Normal     ] ]
                  [ [!]  [Off        ] ]
                  [ [ ]  [Results    ] ]
                  [ [ ]  [Scan       ] ]
                  [ arv                ]
                  [ [VALUE] av         ]
```

(võtmesõnade väiketähtedega kirjutatud osad võib ära jätta). Protseduure saab siluda kas pakk- või dialoogrežiimis. Pakkrežiimis algab pärast mingi lause silumist kohe järgmise silumine, dialoogrežiimis aga peatutakse iga lause järel ning puldilt võib sisestada suvalise direktiivi (silumist jätkab tühisisestus). Vaikimisi kehtestatakse pakkrežiim.

Küsimärk kehtestab silumise dialoogrežiimis (küsimärki ei eraldata järgnevast võtmesõnast tühikuga). Puldilt sisestatavas direktiivis TRACE küsimärk tühistab dialoogrežiimi.

Hüüumärk direktiivis TRACE (ka selle ja järgneva võtmesõna vahel pole tühik lubatud) kehtestab silumisrežiimi, mis jätab vahele kõik PTS- ja CP-käsud (erimuutuja RC seega alati null). Hüüumärk ei mõju puldilt sisestatavatele käskudele. Hüüumärki kasutatakse selliste protseduuride silumisel, kus käskude täitmisel võib olla ebasoovitavaid tagajärgi.

Direktiivi TRACE võtmesõnadel on järgmine tähendus:

ALL - jälgitakse kõikide lausete täitmist;

COMMANDS - jälgitakse PTS- ja CP-käskude täitmist;

ERRORS - väljastatakse teated nende PTS- või CP-käskude kohta, mille täitmine lõpeb nullist erineva lõpukoodiga;

INTERMEDIATES - jälgitakse kõikide lausete täitmist ja väljastatakse kõik vahetulemused, mis saadakse avaldiste arvutamisel või liitnimedes lõpukomponentide asendamisel;

LABELS - jälgitakse märgenditele suunamisi;

NEGATIVES - jälgitakse nende PTS- või CP-käskude täitmist, mis annavad negatiivse lõpukoodi;

NORMAL - sama tähendus mis võtmesõnal NEGATIVES;

OFF - tühistatakse protseduuri silumine (ka vaikimisi);

RESULTS - väljastatakse avaldiste väärtused ning direktiivides PARSE, PULL, ARG muutujatele omistatavad väärtused;

SCAN - siļumine ilma lauseid täitmata; sisemisse alamprotseduuri või direktiivi INTERPRET kuuluv direktiiv TRACE SCAN ei mõju väljapoole seda alamprotseduuri (direktiivi).

Direktiivis TRACE võib parameetrina näidata ka arvu (on vajalik vaid dialoogrežiimis silumisel puldilt sisestatuna, muidu ignoreeritakse). Kui arv on positiivne, siis dialoogrežiimis nii mitme lause täitmise järel peatusi ei tehta.

Kui aga arv on negatiivne, siis jäetakse selle absoluutväärtusega võrdne arv lauseid silumisel vahele. Näiteks direktiiv TRACE -100 teatab, et järgmised sada lauset silumisele ei kuulu ja nende täitmise järel peatusi ei tehta.

Võtmesõnale VALUE järgneva avaldise av väärtuseks peab olema üks direktiivi TRACE võtmesõnadest (mitte VALUE) või arv. Kui avaldise esimene lekseem pole sõne ega nimi, siis võib võtmesõna VALUE ära jätta.

Mingis alamprotseduuris kehtestatud silumismeetodid kehtivad ainult selle alamprotseduuri piires. Kehtivat silumisrežiimi saab küsida standardfunktsiooni TRACE abil.

Direktiiv TRACE kannab iga ekraanile väljastatava rea ette kolmesümbolilise prefiksi, mis näitab andmete liiki:

- - silutav lause protseduuris esitatud kujul;

+++ - silumisel väljastatav teade (nullist erinev lõpukood; küsimus, millele tuleb vastata; süntaksivea teade);

>>> - avaldise või muutujale omistatav väärtus;

>.> - fiktiivsete muutujate (mis on direktiivis PARSE, ARG või PULL tähistatud punktiga) väärtused.

Direktiiv TRACE INTERMEDIATES kasutab veel prefikseid:

>C> - liitnimi pärast lõpukomponentide asendamist;

>F> - funktsiooni väärtus;

>L> - sõne või nimi, mis pole muutuja;

>O> - binaarse tehte tulemus;

>P> - unaarse tehte tulemus;

>V> - muutujale omistatud väärtus.

Silumisel väljastatakse ka rea esimese lause reanumber, mis võimaldab jälgida lausete täitmise järjekorda.

4.4. Standardfunktsioonid

Protseduuridekeele REXX standardfunktsiooni poole pöördutakse avaldise koosseisus: näidates funktsiooni nime, mille järel argumentide loetelu komadega eraldatult ja sulgudega ümbritsetult (loetelu võib olla tühi, kuid sulud peavad olema). Standardfunktsiooni leidmise kiirendamiseks esitatakse tema nimi sõnena (muidu algaks otsimine sisealamprotseduuride hulgast). Standardfunktsiooni nime võib kirjutada suur- või väiketähtedega (viimased teisendab interpretaator suurtähtedeks) ja argumentideks tohivad olla sobivat tüüpi avaldised. Näiteks pikkust määrav argument peab olema mittenegatiivne täisarv, täitesümboliks tohib olla vaid ühesümboliline sõne. Arvudega opereerivad standardfunktsioonid eeldavad, et direktiividega NUMERIC DIGITS ja NUMERIC FUZZ on kehtestatud väärtused 9 ja 0 (nagu vaikimisi).

Andmete teisendamine. Muutuja väärtust võimaldab teada saada standardfunktsioon

```
VALUE(nimi)
```

(ühtlasi teisendatakse väiketähed suurtähtedeks). Kui argument nimi pole muutuja nimi, siis on funktsiooni väärtuseks argument kui sõne. Nimi võib olla ka liitnimi, kus lõpukomponendid asendatakse nende väärtustega. Standardfunktsiooni VALUE kasutamist selgitab protseduurilõik:

```
DROP A3; A33=7; J=3; N=J;
A='VALUE'(N);          /* A=3   */
B='VALUE'('A'J);       /* B=A3  */
C='VALUE'('A'J!!J);    /* C=7   */
```

Arvu esituse muutmiseks (tühikute või nullide lisamine, ümardamine, liigsete tühikute eemaldamine, eksponentkuju muutmine) saab kasutada standardfunktsiooni

FORMAT(arv[,[t-pikk][,[m-pikk][,[j-pikk][,pos]]]])

Esimene argument määrab teisendatava arvu. Kui muid argumente antud ei ole, siis teisendatakse arvu nii, nagu oleks ta liidetud nulliga ja tulemus esitatud vastavalt direktiivile NUMERIC DIGITS. Argument t-pikk määrab täisosa pikkuse, mis ei tohi olla väiksem kui lähtearvus (eesnullideta). Vajaduse korral eemaldatakse eesnullid või lisatakse tühikud. Teise argumendi puudumisel arvu täisosa esitus ei muutu. Argument m-pikk (võib võrduda ka nulliga) määrab murdosa pikkuse. Kui see on lähtearvu murdosa pikkusest väiksem, siis toimub arvu ümardamine; kui aga suurem, siis lisatakse lõppu vajalik arv nulle. Kolmanda argumendi puudumisel arvu murdosa esitus ei muutu. Argument j-pikk määrab numbrite arvu eksponentkuju järgus, kusjuures see väärtus peab konkreetse arvu esituseks piisav olema (kuid mitte suurem kui 9). Kui näidata neljanda argumendi väärtus võrdsena nulliga, siis arvu eksponentkujul ei esitata. Viimane argument pos näitab kümnendpunkti asukoha eksponentkuju mantissis.

Standardfunktsiooni FORMAT kasutamise näiteid:

```
A='FORMAT'('1.73',4,3)          /* A='   1.730'      */
B='FORMAT'('-.76',4,1)          /* B='  -0.8'        */
C='FORMAT'('1.73',4,0)          /* C='   2'          */
D='FORMAT'('12345.73',,,2,2)    /* D='1.234573E+04'  */
E='FORMAT'('12345.73',,3,,0)    /* E='1.235E+4'      */
F='FORMAT'('12345.73',,,3,6)    /* F='12345.73'      */
```

Sõne teisendab kümnendkujule standardfunktsioon

C2D(sõne[,pikkus])

Teise argumendi pikkus puudumisel tõlgendatakse sõnet märgita arvuna. See argument määrab sõnes arvesse võetavate sümbolite arvu: kui ta erineb sümbolite tegelikust arvust sõnes, siis lisatakse sõnele vasakult vajalik arv nulle '00'X või "lõigatakse" vasakult ära üleliigsed sümbolid. Kui pikkus on antud ja sõne esimene bitt võrdub ühega, siis tõlgendatakse sõnet negatiivse arvu täiendkoodina.

Toome näiteid standardfunktsiooni C2D kasutamise kohta:

Funktsioon	Väärtus	Funktsioon	Väärtus
'C2D'('09'X)	9	'C2D'('FF81'X,2)	-127
'C2D'('81'X)	129	'C2D'('FF81'X,1)	-127
'C2D'('A')	193	'C2D'('FF7F'X,1)	127
'C2D'('A'X)	10	'C2D'('F081'X,2)	-3967
'C2D'('81'X,1)	-127	'C2D'('F081'X,1)	-127
'C2D'('81'X,2)	129	'C2D'('0031'X,0)	0

Sõne teisendab kuueteistkümnendkujule standardfunktsioon

C2X(sõne)

Argumendina võib anda ka kuueteistkümnendsõne (toimub sama-susteisendus). Standardfunktsiooni C2X kasutamise näiteid:

```
A='C2X'('72S')          /* A = 'F7F2E2' */
B='C2X'('0123'X)        /* B = '0123'   */
```

Täisarvu teisendab sümbolkujule standardfunktsioon

D2C(arv[,pikkus])

Argumendi pikkus puudumisel ei tohi arv olla negatiivne ning tulemusest eemaldatakse eesnullid. See argument näitab tulemuse pikkust sümbolites, kusjuures negatiivne arv teisendatakse täiendkoodi. Kui saadava sõne pikkus on näidatust suurem, siis "lõigatakse" liigne osa vasakult ära, kui väiksem, siis lisatakse vasakule nulle (positiivsel juhul '00'X, negatiivsel 'FF'X). Toome selle kohta näiteid (esitades tulemused kuueteistkümnendkoodidena):

Funktsioon	Väärtus	Funktsioon	Väärtus
'D2C'(9)	'09'X	'D2C'(257,1)	'01'X
'D2C'(129)	'81'X	'D2C'(-127,1)	'81'X
'D2C'(129,1)	'81'X	'D2C'(-127,2)	'FF81'X
'D2C'(129,2)	'0081'X	'D2C'(12,0)	''

Täisarvu teisendab kümnendkujust kuueteistkümnendkujule standardfunktsioon

D2X(arv[,pikkus])

Et see on äsjavaadelduga sarnane, siis piirdume näidetega:

Funktsioon	Väärtus	Funktsioon	Väärtus
'D2X'(129)	'81'	'D2X'(257,2)	'01'
'D2X'(129,1)	'1'	'D2X'(-127,2)	'81'
'D2X'(129,2)	'81'	'D2X'(-127,4)	'FF81'
'D2X'(129,4)	'0081'	'D2X'(12,0)	''

Kuueteistkümnendsõne teisendab sõneks standardfunktsioon

X2C(sõne)

Argumentsõne iga kaks kuueteistkümnendnumbrit teisendatakse üheks sümboliks, näiteks 'X2C'('F7F2A2') väärtus on '72s'.

Kuueteistkümnendsõne teisendab arvuks standardfunktsioon

X2D(sõne[,pikkus])

Argumentsõne peab olema kas kuueteistkümnendsõne või kuueteistkümnendnumbritest koosnev sõne. Kui pikkus pole näidatud, siis tõlgendatakse sõnet märgita täisarvuna. Kui pikkus on näidatud ja erineb antud sõne pikkusest, siis lõigatakse liigsed sümbolid vasakult ära või lisatakse sinna nulle. Kui pikkus on näidatud ja sõne esimene bitt võrdub ühega, siis tõlgendatakse seda sõnet negatiivse arvu täiendkoodina. Toome vastavaid näiteid:

Funktsioon	Väärtus	Funktsioon	Väärtus
'X2D'('0E')	14	'X2D'('F081',4)	-3967
'X2D'('F81')	3969	'X2D'('F081',3)	129
'X2D'('C6 F0'X)	240	'X2D'('F081',2)	-127
'X2D'('81',2)	-127	'X2D'('F081',1)	1
'X2D'('81',4)	129	'X2D'('0031',0)	0

Aritmeetilised standardfunktsioonid. Arvu absoluutväärtuse leidmiseks kasutatakse standardfunktsiooni

ABS(arv)

Näiteks on nii funktsiooni 'ABC'('12.3') kui ka funktsiooni 'ABS'(-12.3) väärtuseks arv 12.3.

Maksimumi leidmiseks on määratud standardfunktsioon

MAX(arv1[,arv2 ... [,arv9]])

ja miinimumi leidmiseks standardfunktsioon

MIN(arv1[,arv2 ... [,arv9]])

Nagu nendest üldkujudest nähtub, tohib anda kuni üheksa argumenti. Suurema arvu argumente saab anda näiteks selliselt:

'MAX'(1,2,3,4,5,6,7,8,'MAX'(9,10,11,12,13)) .

Arvu märgi leidmine toimub standardfunktsiooniga

SIGN(arv)

Pärast ümardamist (direktiivi NUMERIC DIGITS järgi) leitakse arvu märk (1, -1 või 0) traditsiooniliselt.

Reaalarvu murdosa pikkust muudab standardfunktsioon

TRUNC(arv[,pikkus])

Arv ümardatakse ja vastavalt argumendile pikkus kas lõigatakse murdosast liigsed lõpunumbrid või lisatakse lõpunulle. Kui pikkus pole näidatud, siis loetakse ta nulliks (toimub murdosa ärajätmine). Näiteks funktsioonide 'TRUNC'(127.1,3) ja 'TRUNC'(127.09782,3) väärtused on 127.100 ja 127.097 .

Mittenegatiivse väärtusega juhusliku täisarvu saamiseks etteantud lõigul võib kasutada standardfunktsiooni

RANDOM([min][,[max][,lähtearv]])

Argument min teatab saadava juhusliku arvu vähima (ei tohi olla negatiivne; vaikimisi null), aga argument max suurima väärtuse (mis ei tohi olla üle 10000; vaikimisi 999).

Kui soovitakse iga kord saada üht ja sama juhuslike arvude jada (samalt lõigult), siis neist esimese jaoks tuleb näidata argument lähtearv. Kui aga lähtearv puudub, siis kasutatakse selleks jooksva kellaaja mikrosekundite arvu, mis-

tõttu igas pöördumistsüklis saadakse üldiselt erinev juhuslike arvude jada. Näiteks protseduurilõik

```
S='RANDOM'(1,6,3); DO 5; S=S 'RANDOM'(1,6); END; SAY S;
```

väljastab ekraanile juhuslike arvude jada 1 5 4 3 2 2.

Bitikaupa tehted. Bitikaupa loogiliseks korrutamiseks on keeles ette nähtud standardfunktsioon

```
BITAND(sõne1[,[sõne2][,täitesümbol]])
```

bitikaupa loogiliseks liitmiseks standardfunktsioon

```
BITOR(sõne1[,[sõne2][,täitesümbol]])
```

ja bitikaupa mitteekvivalentsiks standardfunktsioon

```
BITXOR(sõne1[,[sõne2][,täitesümbol]])
```

Argumentidega sõne1 ja sõne2 sooritatakse vastav tehe vasakult paremale lühema sõne lõpuni (tulemuse lõppu lisatakse pikema sõne ülejäänud sümbolid). Kui on antud täitesümbol, siis lisatakse neid lühema sõne lõppu. Puuduva sõne2 asemele võetakse tühisõne. Neid tehteid illustreerivad näited:

```
A='BITAND'('s','P');          /* A='b'    */
B='BITAND'('ABC');            /* B='ABC'  */
C='BITAND'('e','L4','0')      /* C='aM'   */
D='BITOR'('s','P*');          /* D='7*'   */
E='BITOR'('REXX',,'0');       /* E='9577' */
F='BITXOR'('d','%');          /* F='Y'    */
G='BITXOR'('REXX',,'40'X);    /* G='rexx' */
```

Jooksva teabe küsimine. Arvutis registreeritud kuupäeva annab standardfunktsioon

```
DATE([formaat])
```

Kuupäeva esitamise kuju määrava formaadina on lubatud kasutada järgmisi (mida tohib lühendada kuni esitäheni):

Century - tulemus kujul ddddd annab päeva viiekohalise järjekorranumbri alates 1. jaanuarist 1900;

Days - tulemus ddd annab päeva järjekorranumbri aastas;

European - tulemus dd/mm/yy annab kahekohalistena vastavalt kuupäeva, kuunumbri ja aastanumbri;

Julian-OS - tulemuseks yyddd (vt. formaadid D ja E);

Month - tulemuseks kuu nimi inglise keeles;

Orderer - tulemuseks yy/mm/dd (vt. formaat E);

Sorted - tulemuses yyyymmdd aastanumber neljakohaline;

USA - tulemuseks mm/dd/yy (vt. formaat E);

Weekday - tulemuseks nädalapäeva nimi inglise keeles.

Formaadi puudumisel on tulemus kujul dd mmm yyyy , kus mmm annavad kuu nimetuse kolm esitähte (inglise keeles).

Jooksva kellaaja teadasaamiseks ning taimeri sisse- ja väljalülitamiseks kasutatakse standardfunktsiooni

TIME([formaat])

Kellaaja esitamise kuju ja taimeri ümberlülitamist määrava formaadina tohib kasutada järgmisi (lühendatavad esitäheni):

Elapsed - tulemus kujul ssssssss.kkkkkk annab täissekundid ja murdosa mikrosekundites taimeri sisse- või väljalülitamise hetkest alates; kui taimer välja lülitatud, siis toimub sisselülitamine (protseduur alustab väljalülitatult);

Hours - tulemus hh annab täistunnid keskööst alates;

Long - tulemus hh:mm:ss.kkkkkk annab kellaaja (vastavalt tunnid, minutid, täissekundid ja mikrosekundid);

Reset - tulemus nagu formaadi E korral; kui taimer oli sisse lülitatud, siis lülitatakse ta välja;

Seconds - tulemus sssss annab sekundid keskööst alates.

Formaadi puudumisel saadakse kellaaeg kujul hh:mm:ss (vt. formaat L). Märgime veel, et jooksva kellaaja väljastamine ei sõltu direktiivist NUMERIC DIGITS.

Kehtiva keskkonna nime saab teada standardfunktsiooniga

ADDRESS()

Puldipuhvri ridade arvu annab standardfunktsioon

EXTERNALS()

Programmimagasini ridade arvu annab standardfunktsioon

QUEUED()

Puldi reapikkust saab teada standardfunktsiooniga

LINESIZE()

Nime tüüpi võimaldab määrata standardfunktsioon

SYMBOL(nimi)

Argumendiks olev nimi tuleb vormistada sõnena ja seal toimub väiketähtede asendamine suurtähtedega. Kui tegemist on liitnimega, siis asendatakse lõpukomponendid nende väärtustega. Funktsiooni SYMBOL väärtus on üks järgmistest tüüpidest:

VAR, kui argument osutub muutuja nimeks;

LIT, kui argument pole muutuja nimi või on konstant;

BAD, kui argument on kehtetu nimi.

Selle funktsiooni kasutamist illustreerib protseduurilõik:

```
DROP A. 3; J=3;
A='SYMBOL'('J')        /*  A = 'VAR'  */
B='SYMBOL'(J)          /*  B = 'LIT'  */
C='SYMBOL'('A. J')     /*  C = 'LIT'  */
D='SYMBOL'('*')        /*  D = 'BAD'  */
```

Virtuaalarvuti identifikaatori annab standardfunktsioon

USERID()

Sõne tüübi kindlakstegemiseks on standardfunktsioon

DATATYPE(sõne[,tüüp])

Teise argumendi puudumisel annab funktsioon väärtuse NUM või CHAR sõltuvalt sellest, kas sõne on konstant või mitte. Kui on antud kaks argumenti, siis kontrollitakse, kas sõne on seda tüüpi või mitte: funktsiooni väärtus vastavalt 1 või 0. Võib kontrollida järgmisi tüüpe (lühendatavad esitäheni):

Alphanumeric - sõne koosneb tähtedest ja numbritest;

Bits - sõne koosneb numbritest 0 ja 1;

Lowercase - sõne koosneb väiketähtedest;

Mixedcase - sõne koosneb tähtedest;

Number - tegemist on kümnendarvuga;

Symbol - tegemist on nimega või konstandiga;

Uppercase - sõne koosneb suurtähtedest;

Wholenumber - tegemist on täisarvuga (sealhulgas eksponentkujul), mis kooskõlas direktiiviga NUMERIC DIGITS;

X - tegemist on kuueteistkümnendarvuga (tohib kasutada ka väiketähti ja numbritepaare võib lõpust alates loetavuse parandamiseks ühe tühikuga eraldada).

Standardfunktsiooni DATATYPE kasutamise näiteid:

```
A='DATATYPE'('12.3','N')        /*  A = 1  */
B='DATATYPE'('12.3','W')        /*  B = 0  */
C='DATATYPE'('','M')            /*  C = 0  */
D='DATATYPE'('RIDA','L')        /*  D = 0  */
E='DATATYPE'('H2OK','S')        /*  E = 1  */
```

Kehtiva silumisrežiimi teadasaamiseks ja selle muutmiseks võib kasutada standardfunktsiooni

TRACE([režiim])

Argument näitab kehtestatava silumisrežiimi: võib näidata suvalise võtmesõna peale VALUE direktiivist TRACE (vt. lk. 67) kas koos vahetult eelneva küsimärgiga või ilma. Argumendi puudumisel silumisrežiimi ei muudeta ja funktsiooni väärtuseks on kehtiva silumisrežiimi nime esitäht (või koos eelneva prefiksiga). Funktsiooni kasutamist illustreerib näide:

```
TRACE A; ...;SAY TRACE('?E');
```

Ekraanile väljastatakse kehtiva silumisrežiimi nimi A ning kehtestatakse silumisrežiim E (täitmine dialoogrežiimis).

Protseduuri poole pöördumisel üleantavaid parameetreid, nende arvu ja olemasolu saab selgitada standardfunktsiooniga

```
ARG([number[,'Exists'
          ,'Omits' ]])
```

Argumentide puudumisel on funktsiooni väärtuseks parameetrite arv. Esimene argument näitab parameetri järjekorranumbri nende loetelus. Kui teine parameeter (mida võib lühendada kuni esitäheni) pole antud, siis funktsiooni väärtuseks on valitud parameetri väärtus (parameetri puudumisel tühisõne).

Argumendi EXISTS korral kontrollitakse näidatud järjekorranumbriga parameetri olemasolu ning vastavalt sellele on funktsiooni väärtus 1 või 0. Parameetri OMITS tähendus on vastupidine: näidatud järjekorranumbriga parameetri puudumisel on funktsiooni väärtus 1 ja olemasolu korral 0.

Märgime, et kui protseduuri poole pöörduti käsuga EXEC, siis tõlgendatakse parameetrite loetelu ühe parameetrina, kui aga funktsioonina või direktiiviga CALL, siis eraldi parameetritena (nad eraldatakse ka komadega). Juurdepääsu parameetritele võimaldavad ka direktiivid ARG ja PARSE ARG.

Näidatud koodiga veateate annab standardfunktsioon

```
ERRORTEXT(kood)
```

Argumendiks tohib siin olla ülimalt kaheKohaline naturaalarv. Kõigi veateadete loetelu on antud väljaandes [1]. Standardfunktsiooni ERRORTEXT kasutamist illustreerivad näited:

```
A='ERRORTEXT'(16)     /* A = 'LABEL NOT FOUND'      */
B='ERRORTEXT'(60)     /* B = '' (viga nr. 60 pole) */
```

Täidetava protseduuri rea annab standardfunktsioon

```
SOURCELINE([reanr])
```

Argumendi väärtus (ei tohi olla suurem ridade arvust protseduuris) näitab, mitmenda rea teksti nõutakse. Argumendi puudumisel on funktsiooni väärtuseks ridade arv protseduuris.

Sümbolite ja sõnedega opereerimine. Sõne pikkuse saab leida standardfunktsiooniga

```
LENGTH(sõne)
```

Näiteks funktsioonide 'LENGTH'('ABCDEFGH') ja 'LENGTH'('') väärtused on vastavalt kaheksa ja null.

Sõne rajastamist vasakult võimaldab standardfunktsioon

```
LEFT(sõne,pikkus[,täitesümbol])
```

Argument pikkus näitab saadava sõne sümbolite arvu. Liigsed sümbolid eemaldatakse sõne lõpust või lisatakse sinna täitesümboleid (vaikimisi tühikuid). Näiteks:

```
A='LEFT'('ABC D',8)          /*  A = 'ABC D   '  */
B='LEFT'('ABC D',8,'.')      /*  B = 'ABC D...'  */
C='LEFT'('ABC, DEF',7)       /*  C = 'ABC DE'    */
```

Sõne rajastamist paremalt võimaldab standardfunktsioon

```
RIGHT(sõne,pikkus[,täitesümbol])
```

Selle erinevusi eelmisest illustreerivad näited:

```
A='RIGHT'('ABC DEF',5)      /* A = 'C DEF'  */
B='RIGHT'('12',5,'0')       /* B = '00012'  */
```

Sõne tsentreerimist võimaldab standardfunktsioon

$$\left\{\begin{matrix}\text{CENTER}\\ \text{CENTRE}\end{matrix}\right\}(\text{sõne},\text{pikkus}[,\text{täitesümbol}])$$

Sõnet täiendatakse vasakult ja paremalt täitesümboliga (vaikimisi tühikuga) kuni näidatud pikkuseni. Kui näidatud pikkus on lähtesõne pikkusest väiksem, siis eemaldatakse liigsed sümbolid nii vasakult kui paremalt. Kui lisatavate (eemaldatavate) sümbolite arv on paaritu, siis lisatakse (eemaldatakse) paremal üks sümbol rohkem. Näiteid:

```
A='CENTER'('ABC',7)          /* A = '  ABC  '   */
B='CENTER'('ABC',8,'-')      /* B = '--ABC---'  */
C='CENTER'('LOGOFF',3)       /* C = 'OGO'       */
```

Äärmiste sümbolite eemaldamiseks on standardfunktsioon

$$\text{STRIP}(\text{sõne}[,\left[\begin{matrix}\text{'Leading'}\\ \text{'Trailing'}\\ \text{'Both'}\end{matrix}\right][,\text{sümbol}]])$$

Argument 'LEADING' ('TRAILING') käsib eemaldada sõne algusest (lõpust) näidatud järjestikused sümbolid (argumendi puudumisel tühikud). Argument 'BOTH' (ja vaikimisi) ühendab eelmised tegevused. Kasutamist illustreerib protseduurilõik:

```
A='STRIP'('  AB C ')          /* A = 'AB C'    */
B='STRIP'('  AB C  ','L')     /* B = 'AB C  '  */
C='STRIP'('012.7000',,'0')    /* C = '12.7'    */
D='STRIP'('  AB C ','T')      /* D = '  AB C'  */
```

Sõne positsiooni teises sõnes leiab standardfunktsioon

```
INDEX(sõne1,sõne2[,alguspos])
```

Argumendi sõne2 sisaldumist argumendis sõne1 otsitakse vasakult paremale, alates argumendiga alguspos määratud kohast argumendis sõne1 (vaikimisi algusest). Funktsiooni väärtuseks on leitud positsiooni number või null (kui sõne2 ei sisaldu). Funktsiooni kasutamist selgitab protseduurikatkend:

```
A='INDEX'('ABCDEF','BC',3)      /*  A = 0  */
B='INDEX'('ABCABC','BC',3)      /*  B = 5  */
```

Sõne positsiooni teises sõnes annab ka funktsioon

```
POS(sõne1,sõne2[,alguspos])
```

mis erineb eelmisest üksnes selles, et argumentide sõne1 ja sõne2 osad on vahetatud, nagu selgub protseduurikatkendist:

```
A='POS'('GH','ABC DEF GHI')     /*  A = 9  */
B='POS'('X','ABX CDX EFX',4)    /*  B = 7  */
```

Sõne positsiooni määramiseks on ka standardfunktsioon

```
LASTPOS(sõne1,sõne2[,alguspos])
```

mis erinevalt eelmisest otsib paremalt vasakule, alates argumendiga alguspos määratud kohast (vaikimisi sõne2 lõpust). Funktsiooni kasutamist selgitab protseduurikatkend:

```
A='LASTPOS'(' ','ABC DEF GHI')      /*  A = 8  */
B='LASTPOS'(' ','ABC DEF GHI',7)    /*  B = 4  */
```

Ühe sõne sümbolite esinemist teise sõne sümbolite seas saab kontrollida standardfunktsiooniga

```
VERIFY(sõne1,sõne2[,['M'][,alguspos]])
```

Siin toimub argumendi sõne1 sümbolite sisalduvuse kontrollimine argumendis sõne2. Kui sõne1 koosneb vaid sõne2 sümbolitest, siis on funktsiooni väärtus null, vastasel juhul saame funktsiooni väärtuseks sõne1 esimese sellise sümboli järjekorranumbri, mida pole sõne2 sümbolite hulgas. Kui funktsi-

oonis on aga antud veel ka argument 'M', siis saame funktsiooni väärtuseks sõnei esimese sellise sümboli järjekorranumbri, mis leidub sõne2 sümbolite hulgas.

Argument alguspos näitab, mitmendast sümbolist alates hakatakse argumendi sõnei sümboleid otsima argumendis sõne2 (vaikimisi esimesest). Kui sõnei pikkus on väiksem argumendist alguspos, siis saab funktsioon väärtuse null. Standardfunktsiooni kasutamist illustreerib protseduurilõik:

```
A='VERIFY'('123','12345678')                /* A = 0 */
B='VERIFY'('103','12345678')                /* B = 2 */
C='VERIFY'('AB4T','1234567890','M')         /* C = 3 */
D='VERIFY'('1P3Q4','1234567890',,3)         /* D = 4 */
E='VERIFY'('AB3CD5','1234567890','M',4)     /* E = 6 */
```

Sõne lühendamist kontrollib standardfunktsioon

```
ABBREV(sõne,lühend[,pikkus])
```

Kontrollitakse, kas argumendi lühend väärtus sõnena ühtib näidatud sõne esimeste sümbolitega või mitte - vastavalt sellele on funktsiooni väärtus 1 või 0. Argument pikkus näitab, kui mitme esimese sümboli kokkulangemist nõutakse (vaikimisi lühendi tegelik pikkus). Standardfunktsiooni kasutamist illustreerib protseduurikatkend:

```
A='ABBREV'('PRINT','PRI')           /*  A = 1  */
B='ABBREV'('PRINT','PRI',4)         /*  B = 0  */
C='ABBREV'('PRINT','',1)            /*  C = 0  */
D='ABBREV'('PRINT','')              /*  D = 1  */
E='PRN'; F='ABBREV'('PRINT',E,2)    /*  F = 1  */
```

Sõnede võrdlemist teostab standardfunktsioon

```
COMPARE(sõnei[,[sõne2][,täitesümbol]])
```

Enne võrdlemist lisatakse lühema sõne lõppu täitesümboleid (vaikimisi tühikuid; kui sõne2 puudub, asendab teda tühisõne). Kui võrreldavad sõned langevad kokku, siis saab funktsioon väärtuse null, vastasel juhul aga selle sümboli järjekorranumbri, kust alates kokkulangemine aset ei leia. Standardfunktsiooni kasutamist illustreerib protseduurilõik:

```
A='COMPARE'('ABC','ABC')          /* A = 0 */
B='COMPARE'('ABC','AK')           /* B = 2 */
C='COMPARE'('AB-- ','AB','-')     /* C = 5 */
```

Alamsõne väljaeraldamiseks on standardfunktsioon

SUBSTR(sõne,alguspos[,[pikkus][,täitesümbol]])

Argument alguspos näitab, mitmendast sõne sümbolist väljaeraldamine algab. Kolmanda argumendiga teatatakse leitava alamsõne pikkus (vaikimisi sõne lõpuni). Vajaduse korral lisatakse sõne lõppu veel täitesümboleid (vaikimisi tühikuid). Funktsiooni kasutamist illustreerib protseduurikatkend:

```
A='SUBSTR'('ABC',2)          /* A = 'BC'      */
B='SUBSTR'('ABC',2,6,'+')    /* B = 'BC++++'  */
```

Alamsõne eemaldamiseks sõnest on standardfunktsioon

DELSTR(sõne,alguspos[,pikkus])

Argument alguspos näitab, mitmendast sõne sümbolist alates tuleb sümboleid eemaldada (kui see on üle sõne pikkuse, siis sümbolid ei eemaldata). Argument pikkus teatab eemaldatavate sümbolite arvu (vaikimisi sõne lõpuni). Standardfunktsiooni kasutamist illustreerib protseduurikatkend:

```
A='DELSTR'('ABCD',3)          /* A = 'AB'      */
B='DELSTR'('ABCDE',6)         /* B = 'ABCDE'   */
C='DELSTR'('ABCDE',3,2)       /* C = 'ABE'     */
```

Ühe sõne sümbolite ülekatmiseks teise sõne sümbolitega teatud kohast alates saab kasutada standardfunktsiooni

OVERLAY(sõne1,sõne2[,[pos][,[pikkus][,täitesümbol]]])

Argumendi sõne1 sümbolitega asendatakse argmendi sõne2 sümbolid alates kolmanda argumendiga pos näidatud kohast (vaikimisi sõne2 algusest). Neljas argument näitab sõne1 tegelikult kasutatava osa pikkuse (vaikimisi kogu sõne1 pikkus). Vajaduse korral lisatakse lähtesõnede lõppu täitesümboleid (vaikimisi tühikuid). Standardfunktsiooni kasutamist illustreerib protseduurikatkend:

```
A='OVERLAY'(' ','ABCDEF',3)        /* A = 'AB DEF'    */
B='OVERLAY'('.','ABCDEF',3,2)      /* B = 'AB. EF'    */
C='OVERLAY'('123','ABC',5,4,'+')   /* C = 'ABC+123+'  */
```

Ühe sõne lisamiseks teise sõnesse on standardfunktsioon

INSERT(sõne1,sõne2[,[pos][,[pikkus][,täitesümbol]]])

Argument sõne1 lisatakse argumendi sõne2 koosseisu selle sümboli järele, mille järjekorranumbri määrab argument pos (vaikimisi sõne2 ette). Seejuures lisatavate sümbolite arv määratakse argumendiga pikkus (vaikimisi sõne1 pikkusega). Vajaduse korral lisatakse lähtesõnede lõppu täitesümboleid (vaikimisi tühikuid). Standardfunktsiooni kasutamist illustreerib protseduurikatkend:

```
A='INSERT'(' ','ABCDEF',3)         /* A='ABC DEF'     */
B='INSERT'('123','ABC',5)          /* B='ABC  123'    */
C='INSERT'('123','ABC',2,5,'+')    /* C='AB123++C'    */
D='INSERT'('123','ABC',1,2)        /* D='A12BC'       */
E='INSERT'('123','ABC',4,5,'+')    /* E='ABC+123++'   */
F='INSERT'('ALGUS',' LQPP')        /* F='ALGUS LQPP'  */
```

Sümbolite asendamist sõnes võimaldab standardfunktsioon

TRANSLATE(sõne1[,[sõne2][,[sõne3][,täitesümbol]]])

Argument sõne1 määrab teisendatava sõne. Ülejäänud argumentide puudumisel toimub vaid väiketähtede teisendamine suurtähtedeks. Sümbolite asendamine kulgeb järgmiselt: kontrollitakse sõne3 iga sümboli esinemist argumendis sõne1 (neid võib seal olla mitu või mitte ühtegi) ja asendatakse nad sama järjekorranumbriga sümboliga argumendist sõne2. Kui sõne2 puudub, siis võetakse vaikimisi tühisõne. Kui puudub sõne3, siis võetakse tema asemele sõne, mis tuleb standardfunktsiooni XRANGE('00'X,'FF'X) väärtuseks (vt. allpool). Seejuures, kui sõne2 ja sõne3 pikkused on erinevad, siis lisatakse sõne2 lõppu täitesümboleid (vaikimisi tühikuid). Standardfunktsiooni kasutamist illustreerib näide:

```
A='TRANSLATE'('ABCD','12','ABID','.') /* A='12C.' */
```

Sõne järjestikuseks kopeerimiseks on standardfunktsioon

COPIES(sõne,arv)

Argumendiga arv näidatakse, kui mitmest lähtesõne koopiast tuleb konkatenatsioon moodustada. Standardfunktsiooni kasutamist illustreerib protseduurilõik:

```
A='COPIES'('ABC',3)     /* A='ABCABCABC'  */
B='COPIES'('TYHI',0)    /* B=''           */
```

Sõne sümbolid järjestab ümber standardfunktsioon

REVERSE(sõne)

Näiteks 'REVERSE'('loodima') väärtus on sõne 'amidool'.

Järjestikuste koodidega sümbolitest koosneva sõne saab moodustada standardfunktsiooniga

XRANGE([algussümbol][,lõppsümbol])

Argument algussümbol määrab moodustatava sõne esimese sümboli (vaikimisi '00'X), argument lõppsümbol aga viimase sümboli (vaikimisi 'FF'X). Standardfunktsiooni kasutamist illustreerib protseduurilõik (kus tuleb tähele panna, et muutuja E väärtuse arvutamisel on lõppsümbol algussümbolist eespool):

```
A='XRANGE'('A','F')       /* A='ABCDEF'        */
B='XRANGE'('83'X,'87'X)   /* B='cdefg'         */
C='XRANGE'(,'04'X)        /* C='0001020304'X   */
D='XRANGE'('i','j')       /* D='iдефгхцйj'     */
E='XRANGE'('FE'X,'02'X)   /* E='FEFF000102'X   */
```

Sõnadega opereerimine. Protseduuridekeele REXX koosseisu kuulub terve rida standardfunktsioone, mis opereerivad üksiksõnade loetelust koosnevate sõnedega. Sellistesse loeteludesse kuuluvad sõnad tuleb üksteisest eraldada tühikutega, seetõttu ükski sõna tühikuid sisaldada ei saa.

Sõne koosseisu kuuluva sõna pikkuse leidmiseks saab kasutada standardfunktsiooni

WORDLENGTH(sõne,järjek)

Täisarv järjek näitab, mitmenda sõna pikkus sõnest leida tuleb. Kui sõna ei leitud, on funktsiooni väärtus null. Standardfunktsiooni kasutamist illustreerib näide:

```
A='WORDLENGTH'('PROTSEDUURIDEKEEL EXEC2',2) /* A=5 */
```

Sõna algupositsiooni sõnes leiab standardfunktsioon

WORDINDEX(sõne,järjek)

Täisarv järjek näitab, mitmenda sõna algupositsioon sõne algusest leida tuleb. Kui sõna pole, on funktsiooni väärtus null. Standardfunktsiooni kasutamist illustreerib näide:

```
A='WORDINDEX'('PROTSEDUURIDEKEEL EXEC2',2) /* A=19 */
```

Sõna väljaeraldamiseks sõnest on standardfunktsioon

WORD(sõne,järjek)

Täisarv järjek näitab, mitmes sellesse sõnesse kuuluv sõna välja eraldada tuleb (kui sõna ei leita, on väärtuseks tühisõne). Standardfunktsiooni kasutamist illustreerib näide:

```
W='WORD'('PROTSEDUURIDEKEEL REXX',2) /* A='REXX' */
```

Kui sõnest on tarvis välja eraldada mitu järjestikust sõna, siis on see teostatav standardfunktsiooniga

SUBWORD(sõne,järjek[,arv])

Siin argument arv teatab väljaeraldatavate sõnade arvu (vaikimisi eraldatakse sõne lõpuni). Funktsiooni väärtuseks saadavas sõnes eemaldatakse kõik äärmised tühikud, sõnadevahelised tühikud aga säilivad. Standardfunktsiooni kasutamist illustreerib protseduurikatkend:

```
A='SUBWORD'(' AB CD  123 567',2,2) /* A='CD  123' */
B='SUBWORD'('AB CD 123 567',3)     /* B='123 567' */
C='SUBWORD'('AB CD 123 567',5)     /* C='' */
```

Sõnade järjendi asukoha sõnes leiab standardfunktsioon

FIND(sõne,sõnad)

Argument sõnad määrab sõnade järjendi, kusjuures funktsiooni väärtuseks saadakse selle järjendi esimese sõna järjekorranumber sõnes. Kui sellist sõnade järjendit sõnes ei leidu, siis on funktsiooni väärtus null. Standardfunktsiooni kasutamist illustreerib protseduurikatkend:

```
A='FIND'('T@na on reede','on reede')    /* A=2 */
B='FIND'('T@na on reede','on laup@ev') /* B=0 */
```

Sõnade järjendi eemaldab sõnest standardfunktsioon

DELWORD(sõne,järjek[,arv])

Täisarv järjek näitab, mitmendast sõnast algab eemaldamine (kui seda sõna pole, siis eemaldamist ei toimu). Argument arv teatab eemaldatavate sõnade arvu (vaikimisi sõne lõpuni). Funktsiooni kasutamist illustreerib protseduurikatkend:

```
A='DELWORD'('MIS KELL ON?',2)    /* A='MIS'          */
B='DELWORD'('MIS KELL ON?',4)    /* B='MIS KELL ON?' */
C='DELWORD'('MIS KELL ON?',2,1) /* C='MIS ON?'      */
```

Sõne rajastamiseks saab kasutada standardfunktsiooni

JUSTIFY(sõne,pikkus[,täitesümbol])

Argument pikkus näitab saadava sõne pikkuse. Rajastamisel kõigepealt eemaldatakse sõnest äärmised tühikud ja asendatakse kõrvutised sisemised tühikud täitesümboliga (vaikimisi tühikuga). Seejärel eemaldatakse sõne lõpust liigsed sümbolid ja kui saadav sõne lõpeb tühikuga, siis eemaldatakse ka see tühik. Kui sõne osutub nüüd nõutust lühemaks, siis lisatakse sõnade vahele täitesümboleid (võimalikult ühtlaselt). Standardfunktsiooni kasutamist illustreerib protseduurilõik:

```
A='JUSTIFY'('ABC  DEF  EI',8,'+') /* A='ABC++DEF'  */
B='JUSTIFY'('ABC  DEF 123',11)   /* B='ABC  DEF 12' */
```

Sõne formatiseerimiseks on määratud standardfunktsioon

SPACE(sõne[,[arv][,täitesümbol]])

Formatiseerimine seisneb sõne äärmiste tühikute eemaldamises ja sõnes olevate sõnade eraldamises täitesümbolitega (vaikimisi tühikutega), mille arvu määrab argument arv (vaikimisi 1). Funktsiooni kasutamist iseloomustab protseduurikatkend:

```
A='SPACE'('ABC  DEF')            /*  A='ABC DEF'    */
B='SPACE'('   ABC DEF',3)        /*  B='ABC  DEF'   */
C='SPACE'('ABC  DEF  ',2,'+')    /*  C='ABC++DEF'   */
```

4.5. Sõnede analüüs informatsioonivahetusel

Võtmesõnadega PARSE, ARG ja PULL määratud informatsioonivahetusdirektiivide täitmisel toimub sõnede jaotamine osadeks, mis omistatakse vastavate muutujate väärtuseks. Selline osadeksjaotamine toimub muutujate nimedest ja trafarettidest koosneva šablooni abil. Iga muutuja võib šabloonis olla seotud kas vasak-, parem- või kahepoolsete trafarettidega, mis määravad positsioonide numbrid sõnes (trafarettide puudumisel võetakse positsioonide numbrid vaikimisi).

Trafarette on kahte liiki: literaalsed ja positsioonilised. Literaalne trafarett on sümbolite järjend, mida sõne analüüsimisel hakatakse temast otsima ja millele järgneb või millele eelneb eraldatav osasõne. Literaalse trafareti saab šabloonis näidata kas sõnena või ka ümarsulgudega ümbritsetud muutuja nimena, millele on omistatud vastav väärtus. Loetelus esimese literaalse trafareti korral toimub sõne läbivaatamine esimesest positsioonist alates, iga järgneva korral aga sealt, kus läbivaatamine viimati lõppes, seega vaadatakse analüüsitav sõne läbi ainult üks kord.

Toome näiteid literaalsete trafarettide kasutamise kohta. Esimeses näites jaotatakse analüüsitav sõne osadeks koma järgi, s.t. literaalse trafaretina kasutatakse sõnet ',':

```
PARSE VALUE 'KEVAD,SUVI,SYGIS,TALV',
      WITH W1 ',' W2 ',' W3 ',' W4;
```

Selle direktiivi täitmise tulemusena omistatakse muutujatele W1, W2, W3 ja W4 vastavalt väärtused 'KEVAD', 'SUVI', SYGIS' ja 'TALV'. Täpselt sama tulemuse annaks ka direktiivid

```
C=',';  PARSE VALUE 'KEVAD,SUVI,SYGIS,TALV',
          WITH W1 (C) W2 (C) W3 (C) W4;
```

Kui analüüsitavas sõnes trafaretiga määratud sümboleid ei leita, siis on trafaretiks sõne lõpp. Näiteks direktiivi

```
PARSE VALUE 'MINA-MEES,SINA-NAINE',
  WITH W1 '-' W2 ',' W3 ',' W4;
```

täitmine on samaväärne järgmiste omistamislausete täitmisega

```
W1='MINA'; W2='MEES'; W3='SINA-NAINE'; W4='';
```

(muutuja W4 jaoks väärtust pole, selleks võetakse tühisõne).

Kui šablooni kahe muutuja vahelt trafarett puudub, siis vaikimisi võetakse trafaretiks tühik (sellise tühiku kõrval olevad tühikud aga trafarettideks ei ole). Näiteks direktiiv

```
PARSE VALUE 'MINA-HEA   POISS' WITH W1 '-' W2 W3;
```

on samaväärne omistamislausetega:

```
W1='MINA'; W2='HEA'; W3='  POISS';
```

(kui selles näites oleks muutujate W2 ja W3 vahel esinenud trafarett ' ', siis oleks tulemus tulnud sama).

Kui vaikimisi võetavad trafaretid (s.o. tühikud) jaotavad sõne osadeks, mille arv ületab muutujate arvu šabloonis, siis viimase muutuja väärtuseks omistatakse sõne osa kuni tema lõpuni. Näiteks direktiivi

```
PARSE VALUE 'PROTSEDUURIDE KEEL REXX' WITH W1 W2
```

täitmine on samaväärne järgmiste omistamislausete täitmisega

```
W1='PROTSEDUURIDE'; W2='KEEL REXX';
```

Olgu märgitud, et kui šabloonide abil eraldatavat sõneosa pole tarvis omistada ühegi muutuja väärtuseks, siis tuleb muutuja nime asemel kasutada punkti (seega on punkt samaväärne fiktiivse muutujaga).

Nagu ülaltoodud näidetest ilmnes, on muutujate väärtustamine informatsioonivahetusel suures osas sarnane omistamislausete täitmisega. Seetõttu, kui on tarvis ühesuguseid väärtusi omistada tervele reale muutujatele, võib sabloonis kasutada nimede juuri (vt. lk. 48).

Vaatleme nüüd veel positsiooniliste trafarettide kasutamist. Positsiooniline trafarett määrab analüüsitavas sõnes sümboli absoluutse või relatiivse positsiooni, millega algab või lõpeb eraldatav sümbolite järjend (osasõne). Absoluutne positsioon esitatakse märgita positiivse täisarvuna, relatiivne positsioon aga märgiga (+ või -) varustatud täisarvuna. Absoluutne positsioon määrab sümboli järjekorranumbri sõne alguse suhtes, relatiivne positsioon aga nihke viimati määratud positsiooni suhtes (viimane võib olla määratud ka literaalse trafareti abil).

Erinevalt literaalsetest trafarettidest saab positsiooniliste trafarettide abil sõnet läbi vaadata ka tagasisuunas, mis võimaldab sama sõne mitmekordset töötlemist.

Vaatleme positsiooniliste trafarettide kasutamise näiteid. Direktiiv

PARSE VALUE 'T@NA ON TEISIP@EV' WITH W1 5 W2 9 W3;

määrab muutujatele W1, W2 ja W3 vastavalt positsioonid 1-4, 5-8 ja 9-17, s.t. nende muutujate väärtusteks omistatakse vastavalt sõned 'T@NA', ' ON ' ja 'TEISIP@EV'.

Direktiivide

A='123456789'; PARSE VAR A 3 W1 +3 W2 3 W3;

täitmine on samaväärne järgmiste omistamislausete täitmisega

W1='345'; W2='6789'; W3='3456789';

Direktiivi

```
PARSE VALUE 'AA123CCC456DD',
  WITH W1 '2' -1 W2 '4' +2 -1 W3;
```

täitmise tulemusena saavad muutujad W1, W2 ja W3 vastavalt väärtused 'AA1', '123CCC' ja '56DD'.

Kui positsiooniline trafarett määrab positsiooni väljapoole analüüsitavat sõnet, siis võetakse positsiooniks sõne lõpp. Positsiooniliste trafarettide abil on lihtne organiseerida ühe ja sama väärtuse omistamist erinevatele muutujatele. Näiteks direktiivi

```
PARSE VAR X 1 W1 1 W2 1 W3;
```

täitmise tulemusena omistatakse muutujate W1, W2 ja W3 uueks väärtuseks muutuja X senine väärtus.

Protseduuri poole pöördumisel direktiivi CALL abil (või ka funktsiooni poole pöördumisel) võib üleantavate parameetrite loeteluna kasutada mitut komadega eraldatud sõnet. Näiteks kui alamprotseduuri PROC poole pöördutakse direktiiviga

```
CALL PROC 5,'A+B',100;
```

siis selles alamprotseduuris on direktiivi

```
ARG S1,S2,S3;
```

täitmine samaväärne järgmiste omistamislausete täitmisega

```
S1=5; S2='A+B'; S3=100;
```

Kirjandus

1. ЕС ЭВМ. СВМ. ПДО. Язык процедур REXX. Описание языка. E1.00005-03 35 02.

Lisa

REXX-keele võtmesõnad

(viidas esinev täht E, D või S näitab, et võtmesõna on vastavalt erimuutuja, direktiivi või standardfunktsiooni nimi; järgnev arv osutab lehekülje numbrile käesolevas vihikus)

Võtmesõna	Viit	Võtmesõna	Viit
ABBREV	S 83	DELSTR	S 84
ABS	S 73	DELWORD	S 88
ADDRESS	D 54	DO	D 62
ADDRESS	S 77	DROP	D 56
ARG	D 56	D2C	S 71
ARG	S 79	D2X	S 72
BITAND	S 75	ERRORTEXT	S 80
BITOR	S 75	EXIT	D 59
BITXOR	S 75	EXTERNALS	S 77
CALL	D 58	FIND	S 88
CENTER	S 81	FORMAT	S 70
CENTRE	S 81	IF	D 62
COMPARE	S 83	INDEX	S 81
COPIES	S 86	INSERT	S 85
C2D	S 71	INTERPRET	D 65
C2X	S 71	ITERATE	D 63
DATATYPE	S 78	JUSTIFY	S 89
DATE	S 75	LASTPOS	S 82

Võtmesõna	Viit	Võtmesõna	Viit
LEAVE	D 64	SIGL	E 47
LEFT	S 80	SIGN	S 74
LENGTH	S 80	SIGNAL	D 60
LINESIZE	S 77	SOURCELINE	S 80
MAX	S 74	SPACE	S 89
MIN	S 74	STRIP	S 81
NOP	D 62	SUBSTR	S 84
NUMERIC	D 55	SUBWORD	S 88
OWERLAY	S 85	SYMBOL	S 77
PARSE	D 56	TIME	S 76
POS	S 82	TRACE	D 66
PROCEDURE	D 59	TRACE	S 78
PULL	D 58	TRANSLATE	S 86
PUSH	D 58	TRUNC	S 74
QUEUE	D 58	UPPER	D 55
QUEUED	S 77	USERID	S 77
RANDOM	S 74	VALUE	S 69
RC	E 47	VERIFY	S 82
RESULT	E 47	WORD	S 88
RETURN	D 59	WORDINDEX	S 87
REVERSE	S 86	WORDLENGTH	S 87
RIGHT	S 80	XRANGE	S 86
SAY	D 58	X2C	S 73
SELECT	D 64	X2D	S 73

S i s u k o r d