Seega DO ja END-i ning PROC ja END-i vahele jääv programmiosa "trepitakse" 2-3 positsiooni võrra sisse, kusjuures iga END kirjutatakse täpselt kohakuti talle vastava DO või PROC-iga. Selle reegli järgimine suurendab programmi loetavust eriti märgatavalt.

Niisiis võib programmi kirjutamist alustada näiteks 2. positsioonist - ühele reale programmi nimi kooloniga, teisele "PROC OPTIONS (MAIN);". DECLARE-lauseid alustatakse juba 4. positsioonist, samuti järgnevaid GET- ja omistamislauseid. Kui tuleb IF- või DO-lause, toimub "treppimine" - valiku või korduse sisse jääv osa kirjutatakse juba alates 6. positsioonist. Mida rohkem valikuid ja kordusi üksteise sees, seda sügavamale "trepitakse". Lõpuks tuleb "trepp" muidugi tagasi ja programmi lõpulauset "END programmi nimi;" alustatakse järjekordse rea 2. positsioonist.

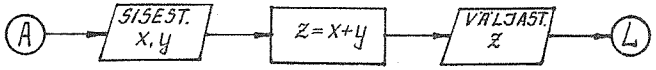
Ei tohi unustada ka kommentaare, kuid need lisatakse juba vabamas vormis, et tõsta veelgi programmi loetavust.

Näited PL-programmide kirjutamise ja vormistamise kohta leiame järgmisest osast.

3. NÄITEID PL-PROGRAMMIDEST

3.1. Lihtsaimad programmid

Näide 3.1. Leida kahe reaalarvu summa.



Joon. 3.1

Tähistame liidetavad x- ja y-ga ning summa z-ga. Algoritmi plokk-skeem on toodud joonisel 3.1 ja vastav PL-programm alljärgnevalt.

```
/* * * * * *
*     KAHE ARVU *
*       SUMMA   *
* * * *     */
```

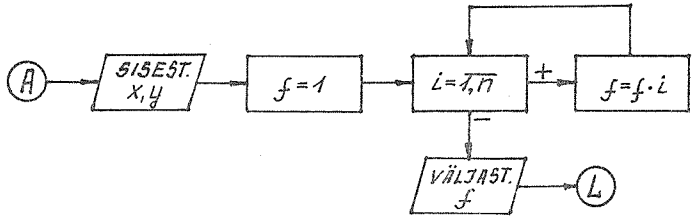


```
SUMMA:
PROC OPTIONS (MAIN);
    DCL (X, Y, Z) FLOAT;
    GET LIST (X, Y);
    Z=X+Y;
    PUT LIST (Z);
END  SUMMA;
```

Ka järgnevates näidetes on programmi päises kommenteeritud vaid tema eesmärki. Konkreetse õppeülesande lahendamisel peavad üliõpilased lisama oma nime, matrikli numbri ja õpperühma, näiteks

```
/* * * * * *
*     ANTS SAAR     *
*      830275       *
*      LI - 11      *
* *       * * *
*     KAHE ARVU     *
*      SUMMA        *
* * * * * */
```

Näide 3.2. Arvutada täisarvu n faktoriaal. Faktoriaali arvutamine seisneb tsüklilises korrutamises, mistõttu ta on esitatav korduse ehk iteratsiooni abil (joon. 3.2). Vastav PL-programm on järgmine:



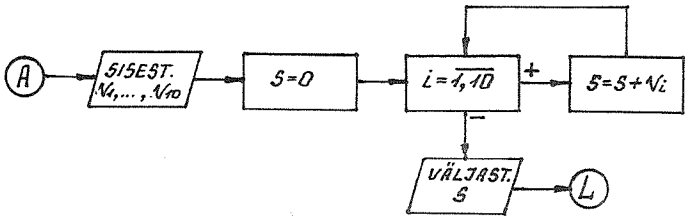
Joon. 3.2

```
                  /*   *   *   *   *
                  *     FAKTORIAALI *
                  *     ARVUTAMINE  *
                  *   *   *   *   */

FAKTOR:
PROC OPTIONS (MAIN);
   DCL FAC FLOAT,          /* FAKTORIAAL */
      (I, N) FIXED;
   GET LIST (N);
   FAC=1;                  /* ALGVÄÄRTUSTAMINE */
   DO I=1 TO N;
       FAC=FAC * I;
   END;
   PUT LIST (FAC);       /*  N! VÄLJASTAMINE */
END FAKTOR;
```

Vaatamata sellele, et FAC on täisarvuline muutuja, kasutatakse kirjelduses võtmesõna FLOAT. Nimelt võimaldab arvutisisene andmete kujutamine FLOAT-muutujatele tunduvalt laiemat väärtustediapasooni kui FIXED-muutujatele. Faktoriaal võib aga omandada väga suuri täisarvulisi väärtusi.

Siin ja ka järgmistes näidetes on plokk-skeemi tähistused sageli lühemad ja tinglikumad kui programmides olevad identifikaatorid. Seda põhjusel, et plokk-skeemi põhiülesanne on selgelt esitada algoritmi struktuur, detailid aga täpsustatakse programmeerimiskeele vahenditega.

Näide 3.3. On antud 10 elemendist koosnev reaalarvude jada. Arvutada selle jada elementide summa.



Joon. 3.3

Tähistame jada järgmiselt: $v_1, v_2, \ldots, v_{10}$. Loomulik on neid andmeid vaadelda massiivina. Summa arvutamine on tsükliline protsess, mistõttu liitmistehe kirjutatakse ainult üks kord – iteratsioonis (joon. 3.3). PL-programm on järgmine:

```
        /*   *   *   *   *   *   *
         * JADA ELEMENTIDE SUMMA *
         *       ARVUTAMINE      *
         *   *   *   *   *   *   */

ELSUM:
PROC OPTIONS (MAIN);
   DCL   V (1:1Ø) FLOAT,          /* JADA */
         SUMMA FLOAT,
         I FIXED;
   GET LIST (V);
   SUMMA=Ø;
   DO I = 1 TO 1Ø;
      SUMMA=SUMMA + V(I);
   END;
   PUT LIST (SUMMA);
END ELSUM;
```

PL/I-s on olemas sisefunktsioon SUM, kus argumendiks massiivi identifikaator. See võimaldab parasjagu leida massiivi kõigi elementide summa, mistõttu viimase programmi saab kirja panna ka lühemalt:

```
          /*   *   *   *   *   *   *
           * JADA ELEMENTIDE SUMMA *
           *       ARVUTAMINE      *
           * SISEFUNKTSIOONI 'SUM' *
           *       ABIL            *
           *   *   *   *   *   *   */

ELSUM1:
PROC OPTIONS (MAIN);
  DCL (V(1 : 1Ø), SUMMA) FLOAT;
```

```
   GET LIST (V);
   SUMMA=SUM(V);
   PUT LIST (SUMMA);
END ELSUM1;
```

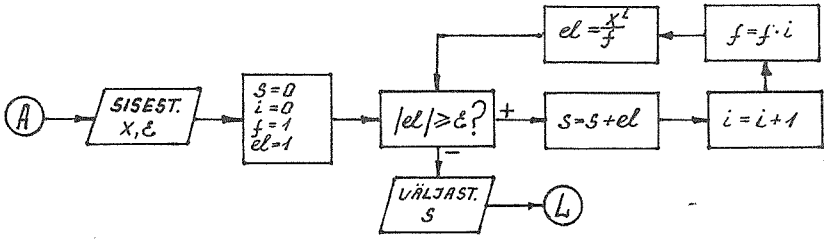
Ei maksa siiski loota, et PL/I-s on olemas sisefunktsioonid igaks juhtumiks.

3.2. Arvutuslikud programmid

Järgnevates programmides esitatakse mitmed tüüpilised juhtumid aritmeetiliste andmete töötlemisel.

Näide 3.4. Arvutada lõpmatu rea $1+x+\frac{x^2}{2!}+\frac{x^3}{3!}+\ldots$ summa etteantud täpsusega ε.

Rea summa saab esitada kujul: $s=\sum_{i=0}^{\infty}\frac{x^i}{i!}$. Et algoritm peab olema lõplik, tuleb piirduda lõpliku arvu liikmetega, mille määrabki täpsus: summeerimise võib lõpetada, kui $\left|\frac{x^i}{i!}\right|<\varepsilon$. Algoritmi plokk-skeem on toodud joonisel 3.4, kus kompaktsuse suurendamiseks on algoritmi jadalise osa sõltumatud tegevused viidud ühte plokki. Vastav PL-programm näeks välja järgmine:



Joon. 3.4


```
/*    *    *    *    *    *
*     LOPMATU REA SUMMA     *
*        ARVUTAMINE         *
*    ETTEANTUD TAPSUSEGA    *
*    *    *    *    *    *  */

AFRIDA:
PROC OPTIONS (MAIN);
  DCL (X,REA_SUMMA, ELEMENT, IFAC) FLOAT,
       EPSILON FLOAT,       /*  TAPSUS */
       I FIXED;
  GET LIST (X, EPSILON);
 /*      ALGVAARTUSTAMISED  */
  REA_SUMMA = 0;
  I=0;
  IFAC=1;
  ELEMENT=1;
/*    *    *    *    *      */
  DO WHILE (ABS(ELEMENT)>= EPSILON);
     REA_SUMMA = REA_SUMMA + ELEMENT;
     I=I+1;
     IFAC=IFAC*I;
     ELEMENT = X**I/IFAC;
  END;
  PUT LIST (X,EPSILON,REA_SUMMA);
END AFRIDA;
```

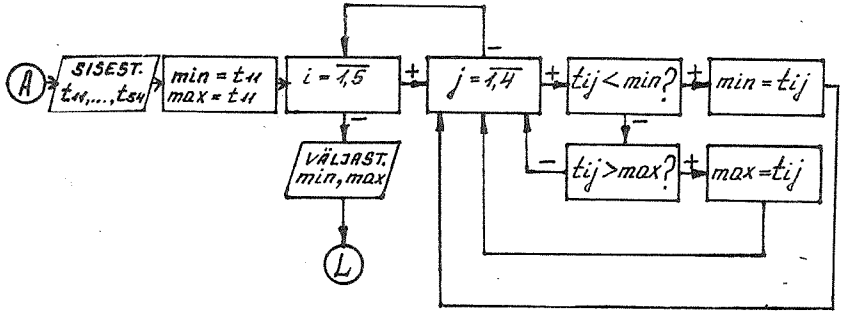
Siin ja edaspidi on programmeeritud ka olulisemate sisendandmete väljastamine. Nii on programmi listingus koos kõik vajalikud andmed - nii sisendandmed kui ka nendele vastavad väljundandmed (tulemused).

Näide 3.5. On antud tabel, mis koosneb 5 reast ja 4 veerust ning mille elemendid on täisarvud. Leida minimaalne ja maksimaalne väärtus selles tabelis.

Antud tabelit võib vaadelda 5 x 4-maatriksina:

$$\begin{pmatrix} t_{11} & t_{12} & \ldots & t_{14} \\ t_{21} & t_{22} & \ldots & t_{24} \\ \ldots & \ldots & \ldots & \ldots \\ t_{51} & t_{52} & \ldots & t_{54} \end{pmatrix}$$

Loomulikuks andmestruktuuriks on kahe indeksiga massiiv. Lahendusalgoritmi plokk-skeem on toodud joonisel 3.5. Et juba esimeses iteratsioonis oleks, millega võrrelda, tuleb otsitavad min ja max algväärtustada. Siin on algväärtuseks valitud tabeli esimene element t_{11}.



Joon. 3.5

PL-programm oleks järgmine:

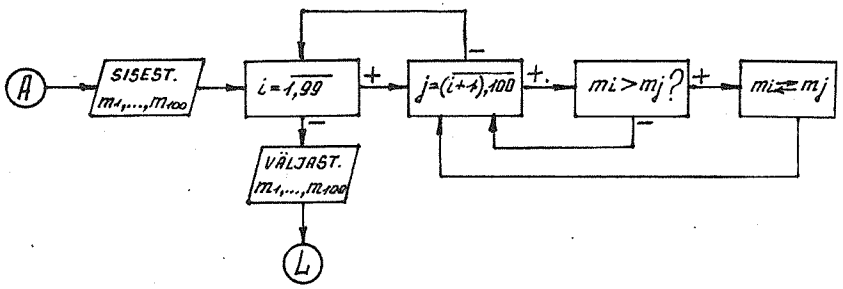
```
          /*   *   *   *   *   *
          * MIN. JA MAKS. VAARTUSE  *
          *  LEIDMINE TABELIST      *
          *   *   *   *   *   *   */
MINMAX:
PROC OPTIONS (MAIN);
  DCL  T (1:5, 1:4) FIXED,    /* TABEL */
       (MIN, MAX) FIXED,
       I FIXED,               /* REA NUMBER */
       J FIXED;               /* VEERU NUMBER */
  GET LIST (T);
  PUT LIST (T);
  MIN=T(1,1);                 /*   ALG-      */
  MAX=T(1,1);                 /* VAARTUSTAMINE */
```



```
       DO  I = 1 TO 5;
         DO J = 1 TO 4;
           IF T (I,J) < MIN   THEN
              MIN = T (I,J);
           ELSE  IF  T (I,J) > MAX  THEN
                    MAX = T (I,J);
         END;
       END;
       PUT SKIP LIST (MIN, MAX);
END MINMAX;
```

Näide 3.6. Järjestada 100 reaalarvust koosnev jada nii, et selle elemendid paikneksid väärtuste mittekahanevas järjekorras.

Sorteerimine ehk järjestamine mingi tunnuse alusel (siin reaalarvulise väärtuse alusel) on arvutis sagedamini lahendatavaid ülesandeid. On terve rida sorteerimisalgoritme. Joonisel 3.6 toodud algoritm on üks lihtsamaid, kuid mitte kuigi efektiivne. Vastav PL-programm on alljärgnev:

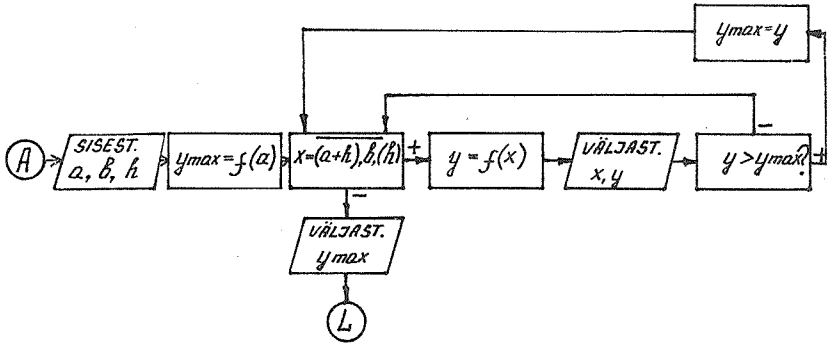


Joon. 3.6

```
/*   *   *   *   *   *
*  JADA JARJESTAMINE      *
*  ELEMENTIDE VAARTUSTE   *
*  MITTEKAHANEVAS JARJE-  *
*                         *
*       KORRAS            *
*   *   *   *   *   *   */
```

```
SORT:
PROC OPTIONS (MAIN);
   DCL  M (1:1ØØ) FLOAT,     /*       JADA        */
        MA FLOAT,            /* ABIMUUTUJA        */
        (I,J) FIXED;
   GET LIST (M);
   PUT LIST ('SORTEERIMATA JADA');
   PUT SKIP LIST (M);
   DO  I = 1 TO 99;
       DO  J = I+1  TO 1ØØ;
         IF  M (I) >  M(J)   THEN
             DO;   /* M(I)  ja  M(J)  VAHETAMINE */
                   MA = M(I);
                   M(I)=M(J);
                   M(J)=MA;
                  /*  *    *    *      *      *      */
             END;

       END;
    END;
    PUT SKIP LIST ('SORTEERITUD JADA');
    PUT SKIP LIST (M);
 END SORT;
```

Näide 3.7. Esitada funktsiooni $y = -x^2 + 0{,}5\,x - 1$ väärtuste tabel lõigul $x \in [a, b]$ sammuga $\Delta x = h$. Leida ka funktsiooni maksimaalne väärtus sel lõigul.



Joon. 3.7

Algoritmi plokk-skeem on joonisel 3.7. Nii tabeli rea väljastamine kui ka funktsiooni maksimaalse väärtuse otsing toimuvad ühes ja samas korduses. PL-programm on järgmine:

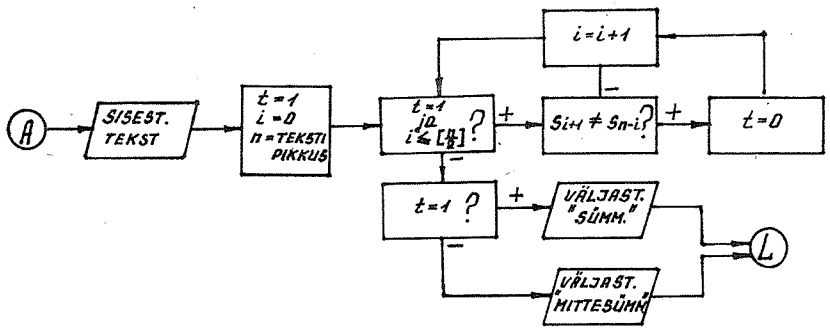
```
      /*    *    *    *    *    *    *
      *  FUNKTSIOONI VAARTUSTE TABELI   *
      *  JA MAKS. VAARTUSE LEIDMINE     *
      *  LOIGUL (A, B) SAMMUGA H        *
      *    *    *    *    *    *    *  */

FUNTAB:
PROC OPTIONS (MAIN);
    DCL (X, Y, A, B, H, YMAX)  FLOAT;
    GET LIST (A, B, H);
    PUT LIST ('FUNKTS, VAARTUSTE TABEL');
    YMAX = - A **2 + Ø.5 *A - 1;     /*ALGVAARTUSTAMINE*/
    DO  X = A + H   TO  B  BY  H;
        Y = - X**2 + Ø.5*X-1;
        PUT SKIP LIST (X,Y); /* TABELI REA TRYKK */
        IF Y>YMAX THEN
           YMAX = Y;
    END;
    PUT SKIP LIST (YMAX);
END FUNTAB;
```

3.3. Tekstitöötluse programmid

Järgnevates programmides on tegemist põhiliselt stringandmete töötlemisega.

Näide 3.8. Tekst paikneb ühel perfokaardil. Kontrollida, kas see tekst on sümmeetriline, s.t. tagurpidi lugedes sama kui edaspidi lugedes.

Tähistame selles ja ka järgnevates plokk-skeemides teksti i-ndat sümbolit s_i-ga. Veel on selles ülesandes vajalik sümmeetriatunnus (plokk-skeemis tähistatud t-ga), mis eristab 2 alternatiivi - "sümmeetriline" ja "mittesümmeetriline". Algoritm on toodud joonisel 3.8, PL-programm alljärgnevalt.



Joon. 3.8.

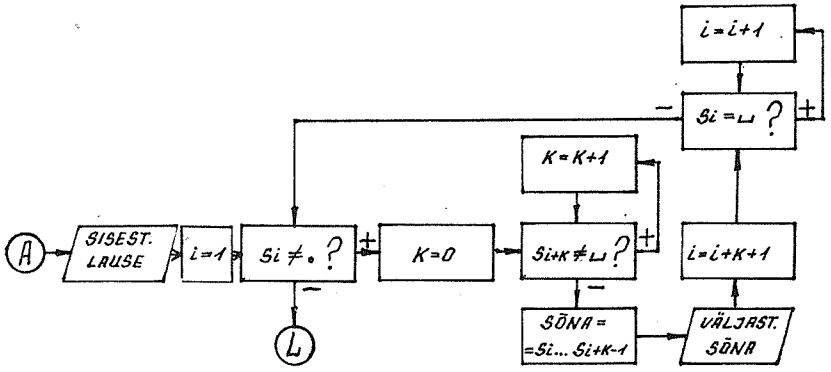
```
/*     *     *     *     *     *
*   TEKSTI SYMMEETRILISUSE       *
*          KONTROLL              *
*     *     *     *     *     */

SYMKONT:
PROC OPTIONS (MAIN);
   DCL TEXT CHAR (78)  VAR,
   (I, N) FIXED,
   SYMMEETRIA  FIXED;
   GET LIST (TEXT);
   SYMMEETRIA = 1;
   I = Ø;
   N = LENGTH (TEXT);   /* N - TEKSTI PIKKUS SYMBOLITES*/
   DO WHILE (SYMMEETRIA=1 & I<= FLOOR (N/2));
     IF SUBSTR (TEXT, I+1,1) ¬=SUBSTR (TEXT, N-I,1) THEN
        SYMMEETRIA   = Ø;      /* SYMM. RIKUTUD  */
     I = I+1;
   END;
   IF SYMMEETRIA = 1 THEN
      PUT LIST ('TEKST', TEXT, 'ON SYMMEETRILINE');
   ELSE
   PUT LIST ('TEKST', TEXT, 'EI OLE SYMMEETRILINE');
END SYMKONT;
```


Maksimaalne sümbolite arv ühel perfokaardil on 80, GET-LIST-lause eeldab, et muutuja väärtus sisestatakse stringkonstandi kujul, seega apostroofide vahel. Nii jääbki 78 sisulist sümbolit, mis kajastub ka muutuja TEXT-kirjelduses.

Näeme, et stringandmete töötlemisel osutub sisefunktsioon SUBSTR (vt. punkt 2.3) hädavajalikuks. Veel leiavad selles peogrammis kasutamist sisefunktsioonid LENGTH ja FLOOR.

Näide 3.9. Lause paikneb ühel perfokaardil. Ainsaks kirjavahemärgiks on lause lõppu tähistav punkt. Väljastada see lause kujul "igas reas üks sõna", kui lauses on iga sõna järel vähemalt üks tühik.

Algoritmi plokk-skeem on toodud joonisel 3.9, kus i



Joon. 3.9

tähistab jooksva sõna esimese sümboli numbrit lause tekstis, k aga jooksva sõna pikkust sümbolites.

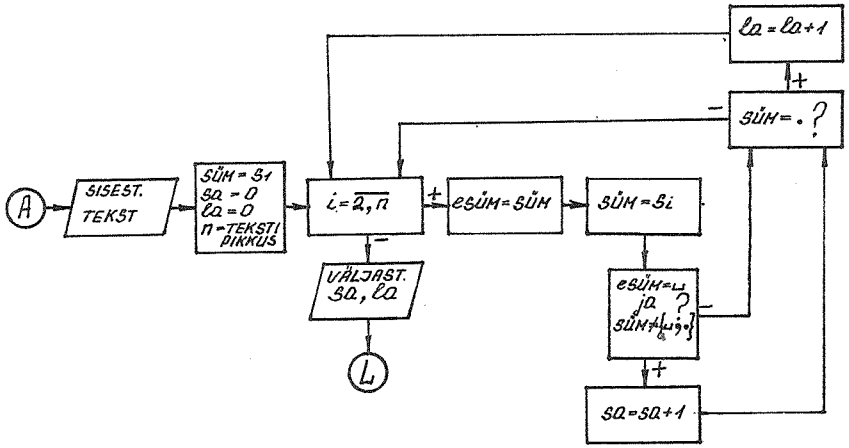
Vastav PL-programm on selline:

```
/*     *     *     *     *     *
*     LAUSE TRYKK KUJUL          *
*   'IGAS REAS YKS SONA'         *
*     *     *     *     *     */
```

```
LAUSETR:
PROC OPTIONS (MAIN);
  DCL  LAUSE CHAR (78) VAR,
       SONA CHAR  (3Ø) VAR,
       I FIXED,          /* SONA ESIMESE SYMBOLI NUMBER */
       K FIXED;          /*      SONA PIKKUS            */
  GET LIST (LAUSE);
  I=1;
  DO WHILE (SUBSTR (LAUSE,I,1) ¬ = '.');
     /*  SONA   LOPU  OTSING  */
     K=Ø;
     DO  WHILE (SUBSTR (LAUSE, I+K,1) ¬ = ' ');
         K=K+1;
     END;
      /*     *     *     *     */
      SONA = SUBSTR (LAUSE, I,K);
      PUT  SKIP  LIST (SONA);
      I=I+K+1;
      /*  UUE SONA  ALGUSE  OTSING  */
      DO WHILE (SUBSTR (LAUSE, I,1) = ' ');
         I=I+1;
      END;
       /*    *     *     *     */
   END;
END LAUSETR;
```

Näide 3.10. Tekst paikneb 10 perfokaardil ja koosneb lausetest. Iga lause lõpeb punktiga, kusjuures lause sees võivad esineda ka komad. Enne iga sõna on vähemalt üks tühik. Loendada tekstis olevate sõnade arv ja lausete arv.

Teksti maksimaalne pikkus on 798 sümbolit (selgitus on näites 3.8). Lahendusalgoritmi plokk-skeem on joonisel 3.10 ning PL-programm alljärgnev:



Joon. 3.10

```
      /*    *    *    *    *    *
      *    TEKSTIS OLEVATE SONADE
      *    JA LAUSETE ARVU        *
             LOENDAMINE
      *       *    *    *    *    */

LOEND:
PROC OPTIONS (MAIN);
   DCL TEXT CHAR (798) VAR,
       SYMBOL CHAR (1),
       EELMINE_SYMBOL CHAR (1),
       SONADE_ARV FIXED,
       LAUSETE_ARV FIXED,
       (I,N) FIXED;
   GET LIST (TEXT);
   PUT LIST (TEXT);
   SYMBOL = SUBSTR (TEXT, 1,1);
   SONADE_ARV = Ø;
   LAUSETE_ARV = Ø;
```

```
   N=LENGTH (TEXT);
   DO I = 2 TO N;
      EELMINE_SYMBOL = SYMBOL;
      SYMBOL = SUBSTR (TEXT, I,1);
      IF EELMINE_SYMBOL = ' ' & SYMBOL ¬= ' ' &
        SYMBOL ¬= ',' & SYMBOL ¬= '.' THEN
          /*   UUE SONA ALGUS    */
        SONADE_ARV = SONADE_ARV + 1;
      IF SYMBOL  = '.' THEN
         /*    LAUSE LOPP      */
        LAUSETE_ARV = LAUSETE_ARV+1;
    END;
    PUT SKIP LIST (SONADE_ARV,LAUSETE_ARV);
END LOEND;
```

Nähtub, et mõnikord võivad IF-lausete (ja DO-WHILE-lausete) tingimused olla üsna keerukad ning võtta enda alla mitu rida programmi tekstis.

K I R J A N D U S

1. Программирование на ПЛ/I ОС ЕС. М., Статистика, 1979.
2. Гребенников Л.К., Лебедев В.Н. Решение задач на ПЛ/I в ОС ЕС. М., Финансы и статистика, 1981.
3. Скотт Р., Сондак Н. ПЛ/I для программистов. М., Статистика, 1977.
4. Jürgenson, R. Programmeerimine kolmanda põlvkonna arvutitele. Tln., Valgus, 1977.
5. Programmeerimiskeele PL/I põhimõisteid. Trt., TRÜ, 1978.
6. Sissejuhatus operatsioonisüsteemi OS kasutamisse. Tln., TPI, 1978.