第５章

商品分類の産業分類への変換

ー変換エラーデータ処理のプログラムー

黒子正人

はじめに

IO20 産業分類コード（以下、IO20 と略記）にもとづく世界貿易マトリクスを作成することに先立ち、元データとなる AID-XT で使われている SITC 商品分類コード（以下 SITC と略記）を IO20 に変換する必要がある。参考文献［２］では、変換の際エラーになったデータの処理をどのように行うかをめぐって２つの方式、すなわち、均等配分方式と金額加重配分方式を取り上げて概要を述べている。ここではさらに具体的な処理手順とプログラムの詳細について説明する。

１．均等配分による変換

１.１　拡張変換表の作成処理

ここでは、表１のような４つの処理を順番に行う。なお、以下の説明で登場するファイル名は拡張子で内容を見分けることが出来る。*.ex は、PL/I プログラムをコンパイル、実行するためのスクリプトファイルである。*.pli は PL/I で書かれたソースプログラムファイルである。*.ksh は、本来は Korn shell のために書かれたスクリプトファイルを示すが、ここでは Syncsort というソートを行うパッケージソフトウェア用に書かれたスクリプトファイルを示す。

ここでは、p110.pli と p4.pli のソースプログラムリストを元に、処理内容を説明する。p20.ksh と s4.ksh は単純なソート（整列）処理なので説明は省略する。

表１　拡張変換表の作成処理

スクリプト名	プログラム名	処理内容
p110.ex	p110.pli	オリジナル変換表の SITC を 1～5 桁にする。
p20.ksh	-	キー（SITC, IO20）の昇順にソートする。
p4.ex	p4.pli	拡張変換表を作成する。
s4.ksh	-	キー(SITC, IO20(21)) の昇順にソートする。

なお、以下に掲げるプログラムリストはコンパイル時に出力されるリストをもとにしたため、行番号が付いている。行番号が飛んでいるところがあるが、これは適宜、空行や不要なコメントを削除したものである。また行番号が無いところは１行の長さが長いために改行を入れたところである。

p110.pli の処理を説明する。オリジナルの SITC/IO20 変換表（18～21 行）を読み込んで（46 行）、IX で SITC に含まれる改行コードを取り去る。（69 行）次に、DgtCut1 で、SITC が空白でない場合に限り、SITC の 2 桁目から 5 桁目までの 4 桁分を空白にして（85 行）、OutD で出力する。次に DgtCut2 で、DgtCut1 と同じように SITC の 3 桁目から 5 桁目までの 3 桁分を空白にして出力する。つぎの DgtCut3 では、4 桁目から 2 桁分、次の DgtCut4 では、5 桁目から 1 桁分を空白にして出力する。次の DgtCut5 ではもし 5 桁目が空白でない場合（つまり 5 桁コードだった場合）はそのまま書き出す。

```
 1 /*-------------------------------------------------------
 2       Program Name: p110.pli
 3       Language    : PL/I
14  -------------------------------------------------------*/
15      p110: proc options(main);
16      dcl  (@TmCP entry       (char(15),fixed bin(31))
17            )                 external,
18            1 rec_i based(ptr),  /* オリジナル変換表*/
19              2 ic            pic'99',  /* IO20 */
20              2 fil           char(6),  /* 空白 */
21              2 sc            char(5),  /* SITC */
22
23            1 rec_o,              /* 出力レコード */
24              2 ic            pic'99',  /* IO20 */
25              2 fil           char(6),  /* 空白 */
26              2 sc            char(5),  /* SITC */
27
```

```
28              s2              char(5), /* SITC作業域 */
29              ptr             pointer,
30              (m,n1,n2,n3,n4,n5,i)    fixed bin(31) init(0);
31
32            on attn stop;
33            call INpD;
34              call @TmCP('input  cnt            ',m );
35              call @TmCP('output cnt of 1dgt cnv ',n1);
36              call @TmCP('output cnt of 2dgt cnv ',n2);
37              call @TmCP('output cnt of 3dgt cnv ',n3);
38              call @TmCP('output cnt of 4dgt cnv ',n4);
39              call @TmCP('output cnt of 5dgt cnv ',n5);
40          /*-------------------------------------------**
41          |                 INpD                      |
42          **-------------------------------------------*/
43          INpD:       proc;
44          dcl f               file record;
45            on endfile(f);
46            read file(f) set (ptr);
47            do while(^endfile(f));
48                 m=m+1;
49                  rec_o=rec_i, by name;
50                  call IX;
51                  call DgtCut1;
52                  call DgtCut2;
53                  call DgtCut3;
54                  call DgtCut4;
55                  call DgtCut5;
56                  read file(f) set (ptr);
57            end;
58          end INpD;
60          /*-------------------------------------------**
61          |                 IX                        |
64          **-------------------------------------------*/
65          IX: proc;
66                i = index(rec_i.sc,'0A'x,1);
67                if i^=0 then do;
68                        i=i-1;
69                        s2=substr(rec_i.sc,1,i);
70                end;
71                else   s2=rec_i.sc;
73          end IX;
75          /*-------------------------------------------**
76          |                 DgtCut1                    |
81          **-------------------------------------------*/
82          DgtCut1:    proc;
```

```
 83              if substr(s2,1,1)^=' ' then do;
 84                 rec_o.sc = s2;
 85                 substr(rec_o.sc,2,4)=' ';
 87                 call OutD;
 88                 n1 = n1 + 1;
 90              end;
 92           end DgtCut1;
 93           /*-------------------------------------------**
 94           |                 DgtCut2                   |
 99           **-------------------------------------------*/
100           DgtCut2:       proc;
101              if substr(s2,2,1)^=' ' then do;
102                 rec_o.sc = s2;
103                 substr(rec_o.sc,3,3)=' ';
105                 call OutD;
106                 n2 = n2 + 1;
108              end;
110           end DgtCut2;
111           /*-------------------------------------------**
112           |                 DgtCut3                   |
117           **-------------------------------------------*/
118           DgtCut3:       proc;
120              if substr(s2,3,1)^=' ' then do;
121                 rec_o.sc = s2;
122                 substr(rec_o.sc,4,2)=' ';
124                 call OutD;
125                 n3 = n3 + 1;
127              end;
129           end DgtCut3;
130           /*-------------------------------------------**
131           |                 DgtCut4                   |
136           **-------------------------------------------*/
137           DgtCut4:       proc;
138              if substr(s2,4,1)^=' ' then do;
139                 rec_o.sc = s2;
140                 substr(rec_o.sc,5,1)=' ';
142                 call OutD;
143                 n4 = n4 + 1;
145              end;
147           end DgtCut4;
148           /*-------------------------------------------**
149           |                 DgtCut5                   |
153           **-------------------------------------------*/
154           DgtCut5:       proc;
157              if substr(s2,5,1)^=' ' then do;
158                 rec_o.sc = s2;
```

```
160                 call OutD;
161                   n5 = n5 + 1;
163               end;
165         end DgtCut5;
167         /*-------------------------------------------**
168         |              OutD
169         **-------------------------------------------*/
170         OutD:                   proc;
171         dcl    fo               file record;
172                   write file(fo) from (rec_o);
173         end OutD;
175         end p110;
```

p4.pli の処理を説明する。これは、p110.pli で出力されたファイルを SITC,IO20 のキー順にソートされたファイルを入力ファイルとして、拡張変換表を作成するプログラムである。基本的には、コントロールキーが変わる毎に処理を行うコントロールブレークであり、26 行からの主処理（INpD）においてそれをみることができる。ここでは二重ループのコントロールブレークをしている。最初のループ（35 行）では SITC が変わるまで do while ループ内の処理を行う。そのループ内側にある 2 番目の do while ループ（37 行）は SITC か IO20 が変わるまで SumP を繰り返す。SumP では、IO20 毎の頻度を k2 に累積する。(72 行) 2 番目の do while ループがブレークすると OutD2 に移り、拡張変換表の該当位置に頻度を書き出す。（83 行）さらに 1 番目の do while ループがブレークすると 45 行の OutD1 に移る。ここで拡張変換表の頻度合計を計算して、(91～95 行) ファイルへ出力する。なお、各ループ処理のコントロールキーの初期化を、SetVC1、SetVC2 の各サブプロシージャで行っている。

```
1         /*-------------------------------------------**
2         |                  p4.pli                     |
3         **-------------------------------------------*/
4         p4: proc options(main);
5         dcl  (@TmCP entry      (char(15),fixed bin(31))
6               )                external,
7             1 rec_i,          /* 入力レコード */
8               2 ic            pic'99',     /* IO20 */
9               2 fil           char(6),     /* 空白 */
10              2 sc            char(5),     /* SITC */
11
12            1 rec_o,          /* 拡張変換表レコード */
```

```
13              2 sc            char(5),     /* SITC */
14              2 fil2          char(1) init(' '),    /* 空白 */
15              2 fq(21)        pic'9999',   /* 頻度テーブル(x21) */
16
17            ic2               pic'99',     /* IO20 作業域 */
18            sc2               char(5),     /* SITC 作業域 */
19            (m,n,k1,k2,i)     fixed bin(31) init(0);
21            on attn stop;
22            call INpD;
23            call @TmCP('inp_d',m);
24            call @TmCP('out',n);
26          /*-------------------------------------------**
27          |                  INpD                     |
28          **-------------------------------------------*/
29          INpD:         proc;
30          dcl f                 file record;
31            on endfile(f);
32            read file(f) into (rec_i);
33            do while(^endfile(f));
34              call SetVC1;
35              do while ( (rec_i.sc=sc2) & (^endfile(f)) );
36                call SetVC2;
37                do while (  (rec_i.sc=sc2) &
38                            (rec_i.ic=ic2) &
39                            (^endfile(f) ) );
40                  call SumP;
41                  read file(f) into (rec_i);
42                end;
43                call OutD2;
44               end;
45               call OutD1;
46             end;
48          end INpD;
50          /*-------------------------------------------**
51          |                  SetVc1                     |
52          **-------------------------------------------*/
53          SetVC1:       proc;
54             sc2=rec_i.sc;
55             k1=0;
56             rec_o.fq=0;
57          end SetVC1;
59          /*-------------------------------------------**
60          |                  SetVc2                     |
61          **-------------------------------------------*/
62          SetVC2:       proc;
63             ic2=rec_i.ic;
```

```
64          k2=0;
65        end SetVC2;
67        /*-------------------------------------------**
68        |                  SumP                      |
69        **-------------------------------------------*/
70        SumP:         proc;
71          m=m+1;
72          k2=k2+1;
73        end SumP;
75        /*-------------------------------------------**
76        |                  OutD2                      |
77        **-------------------------------------------*/
78        OutD2:        proc;
79                k1=k1+k2;
80                if ic2>20 then do;
81                      put skip list(sc2,ic2);
82                  end;
83                else  fq(ic2)=k2;
84        end OutD2;
86        /*-------------------------------------------**
87        |                  OutD1                      |
88        **-------------------------------------------*/
89        OutD1:        proc;
90        dcl  fo       file;
91                i=1;
92                do while (i<=20);
93                  fq(21)=fq(21)+fq(i);
94                  i=i+1;
95                end;
96                rec_o.sc=sc2;
97                write file(fo) from (rec_o);
98               n=n+1;
99        end OutD1;
101       end p4;
```

1.2　オリジナル変換表による変換処理（p1s.pli プログラム）

p1s.pli は、AID-XT 基礎データを読み込み、まず該当報告国のデータのみを対象に、その SITC でオリジナル変換ファイルをサーチし該当 SITC があった場合、IO20 に変換してパスデータを書き出し、無い場合は、エラーデータに書き出すという処理である。特徴としては@Bsch というバイナリサーチ（二分検索）のサブプログラムを使用している点がある。これを用いて該当報告国表、オリ

ジナル変換表を表引きしている。

```
 1        /*-------------------------------------------**
 2        # Program Name: p1s.pli
 3        # Language    : PL/I
12        **-------------------------------------------*/
13        p1s: proc options(main);
14        dcl  (@TmCP entry     (char(15),fixed bin(31)),
15             @Bsch entry      (char(*),*,fixed bin(31),
                                 fixed bin(31))
16             )               external,
17             (cc(2,100),c(2))  char(6),   /* 国テーブル */
18             (ss(2,5000),s(2)) char(5);   /* SITC/I020 変換表用 */
19        dcl 1 rec_i,  /*  入力(AID-XT)データ */
20              2 rc           char(6),   /* 報告国 */
21              2 d            char(1),   /* 輸出入区分 */
22              2 s            char(5),   /* SITC   */
23              2 pc           char(6),   /* 相手国 */
24              2 y            pic'99',   /* 報告年 */
25              2 v            pic'(12)-9',   /* 金額     */
26              2 qu           char(2),       /* 数量単位 */
27              2 q            pic'(12)-9',   /* 数量     */
28
29             1 rec_o,  /* 出力レコード  */
30              2 rc           char(6),
31              2 d            char(1),
32              2 io           char(2),   /* I020  */
33              2 s            char(5),
34              2 pc           char(6),
35              2 y            pic'99',
36              2 v            pic'(12)-9',
37              2 qu           char(2),
38              2 q            pic'(12)-9',
39             /*  該当報告国抽出用データ  */
41             1 rec_fc         based(ptr),
42              2 cc1          char(6),   /* 国コード１ */
43              2 fil1         char(1),   /* 空白       */
44              2 cc2          char(6),   /* 国コード２ */
45              2 fil2         char(1),   /* 空白       */
46              2 cn           char(20),  /* 国名       */
47             /*  SITC/I020 変換表データ  */
48             1 rec_fs        based(ptr2),
49              2 ss2          char(2),   /* I020 */
50              2 fil1         char(6),   /* 空白 */
51              2 ss1          char(5),   /* SITC */
52
```

```
53              ss22              pic'99' init(0),
54              (ptr,ptr2)                pointer,
55              (m,n,nc,ns,ne,p,ix)       fixed bin(31) init(0);
56
57              on attn stop;
58              call Setlni;
59              call INpCC;
60              call @TmCP('inp rp',nc);
61              call INpSS;
62              call @TmCP('inp cnv',ns);
63              call INpD;
64              call @TmCP('inp_d',m);
65              call @TmCP('out',n);
66              call @TmCP('cnv err',ne);
68            /*-------------------------------------------**
69            |                  Setlni                     |
70            **-------------------------------------------*/
71            Setlni:      proc;
72                cc=' ';
73                ss=' ';
74            end Setlni;
76            /*-------------------------------------------**
77            |                  INpCC                      |
78            **-------------------------------------------*/
79            InpCC:               proc;
80            dcl  fc              file;
81                 on endfile(fc);
82                 read file(fc) set(ptr);
83                 do while(^endfile(fc));
84                     c(1)=cc1;
85                     c(2)=cc2;
86                     call MkCC;
87                     read file(fc) set(ptr);
88                 end;
89            end INpCC;
91            /*-------------------------------------------**
92            |                  MkCC                       |
93            **-------------------------------------------*/
94            MkCC:        proc;
95                nc=nc+1;
96                 cc(*,nc)=c;
97            end MkCC;
98
99            /*-------------------------------------------**
100           |                  INpSS                      |
101           **-------------------------------------------*/
```

```
102         InpSS:                  proc;
103         dcl  fs                 file;
104              on endfile(fs);
105              read file(fs) set(ptr2);
106              do while(^endfile(fs));
107                  ix=0;
108                 ix=index(ss1,'0A'x,1);
109                  if ix^=0 then s(1)=substr(ss1,1,ix-1);
110                           else s(1)=ss1;
111                  s(2)=ss2;
112                  call MkSS;
113                  read file(fs) set(ptr2);
114              end;
115         end INpSS;
117         /*-------------------------------------------**
118         |                  MkSS                     |
119         **-------------------------------------------*/
120         MkSS:        proc;
121             ns=ns+1;
122              ss(*,ns)=s;
123         end MkSS;
125         /*-------------------------------------------**
126         |                  INpD                     |
127         **-------------------------------------------*/
128         INpD:        proc;
129         dcl  f                  file record;
130              on endfile(f);
131              read file(f) into (rec_i);
132              do while(^endfile(f));
133                  m=m+1;
135                  /*  相手国＝世界のみを処理する */
136                  if rec_i.pc="000000"
137                    then do ;
138                      rec_o=rec_i ,by name;
139                      call Ccnv;
140                    end;
142                  read file(f) into (rec_i);
143              end;
144         end INpD;
145
146         /*-------------------------------------------**
147         |                  Ccnv                     |
148         **-------------------------------------------*/
149         Ccnv:        proc;
151              /* reporting country code de nukidashi */
153              c(1)=rec_i.rc;
```

```
154             call @Bsch(c(1),cc(1,*),nc,p);
155             if p^=0
156               then do;
157                 call Scnv;
158               end;
160         end Ccnv;
161         /*-----------------------------------------------**
162         |                    Scnv                       |
163         **-----------------------------------------------*/
164         Scnv:         proc;
165             /* 数量単位に規定値以外がある場合は空白にする */
166             call QUcnv;
168             /* SITC 末尾がハイフン(-)の場合は空白にする */
170             if substr(rec_o.s,5,1)='-'
                  then substr(rec_o.s,5,1)=' ';
                /* SITC / IO20 変換処理 */
171             s(1)=rec_o.s;
172             p=0;
173             call @Bsch(s(1),ss(1,*),ns,p);
174             if p^=0
175               then do;
176                 ss22 = substr(ss(2,p),1,2);
177                 rec_o.io = ss22;
178                 call OutD;
179               end;
180               else do;
181                 call OutE;
182               end;
184         end Scnv;
185         /*-----------------------------------------------**
186         |                    QUcnv                      |
187         **-----------------------------------------------*/
188         QUcnv: proc;
190           if  rec_i.qu='B0'|
191               rec_i.qu='B3'|
192               rec_i.qu='B6'|
193               rec_i.qu='C0'|
194               rec_i.qu='G0'|
195               rec_i.qu='K0'|
196               rec_i.qu='L0'|
197               rec_i.qu='M0'|
198               rec_i.qu='M3'|
199               rec_i.qu='M6'|
200               rec_i.qu='N0'|
201               rec_i.qu='N3'|
202               rec_i.qu='P0'|
```

```
203             rec_i.qu='P3'|
204             rec_i.qu='P6'|
205             rec_i.qu='S0'|
206             rec_i.qu='U0'|
207             rec_i.qu='U3'|
208             rec_i.qu='U6'|
209             rec_i.qu='V0'|
210             rec_i.qu='V3'|
211             rec_i.qu='W0'|
212             rec_i.qu='W3'|
213             rec_i.qu='X3' then rec_o.qu=rec_i.qu;
214           else              rec_o.qu=' ';
215       end QUcnv;
218       /*-----------------------------------------**
219       |                  OutE                   |
220       **-----------------------------------------*/
221       OutE:        proc;
222       dcl fe       file;
223              rec_o.io = ' ';
224              write file(fe) from (rec_o);
225             ne=ne+1;
226       end OutE;
229       /*-----------------------------------------**
230       |                  OutD                   |
231       **-----------------------------------------*/
232       OutD:        proc;
233       dcl fo       file;
234              write file(fo) from (rec_o);
235             n=n+1;
236       end OutD;
238       end p1s;
```

1.3　拡張変換表による1回目の変換処理（p50s.pli プログラム）

ここでは、p1s.pli でエラーになったファイルを読み込んで１－１で作成された拡張変換表を引いてパスデータとエラーデータとを書き出す。この時点ではまだエラーデータの SITC は変化させない。

まず、表引き用の配列を初期化する。（SetIni）次に、拡張変換表を配列 ii(i,p) に格納する。(INpSS, MkSS)この配列の添字 p が各 SITC に対応する連番であり、添字 i は IO20 の 21 個の配列に対応する。 主処理(INpD)で p1.pli からのエラーデータを読み込み、Scnv で拡張変換表を表引きして該当 SITC があった場合に

はOutDでパスデータを出力し、無い場合にはOutEでエラーデータを出力する。OutDは多少長いサブプロシージャとなっている。ここでは21個の配列に格納された頻度の値を使用して IO20 のゼロより大きい値が入っているものに金額(v)と数量 (q)の数値を配分している。(138～147 行) 配分の仕方は 144 行のとおり、各IO20の頻度を頻度の合計値 (IO20の配列の21番目) で割った値に従って均等配分している。計算結果を四捨五入し (151～155 行)、金額か数量のどちらかがゼロ以上のデータを出力している。(157～159行)

```
 1        /*-----------------------------------------**
 2        # Program Name: p50s.pli
 3        # Language    : PL/I
14        **-----------------------------------------*/
15        p50s: proc options(main);
16        dcl   (@TmCP entry      (char(15),fixed bin(31)),
17              @Bsch entry       (char(*),*,fixed bin(31),
                                   fixed bin(31))
18              )                 external,
19              ii(21,5000)       pic'(4)9',
20              (ss(5000),sc)     char(5);
22        dcl 1 rec_i,
23              2 rc              char(6),
24              2 d               char(1),
25              2 io              char(2),
26              2 s               char(5),
27              2 pc              char(6),
28              2 y               pic'99',
29              2 v               pic'(12)-9',
30              2 qu              char(2),
31              2 q               pic'(12)-9',
32
33            1 rec_o like rec_i,
34
35            1 rec_e like rec_i,
36
37            1 rec_fs,
38              2 s_fs            char(5),
39              2 fil1            char(1),
40              2 i_fs(21)        pic'(4)9',
41
42            s2                  pic'99',
43            (fq,v2,q2)          float bin(53) init(0),
44            (m,n,ne,ns,p,ix,i)          fixed bin(31) init(0);
```

```
45
46              on attn stop;
47              call Setlni;
48              call INpSS;
49              call @TmCP('inp cnv',ns);
50              call INpD;
51              call @TmCP('inp_d',m);
52              call @TmCP('out',n);
53              call @TmCP('err',ne);
56
57          /*-------------------------------------------**
58          |                  Setlni                    |
59          **-------------------------------------------*/
60          Setlni:       proc;
61              ii=0;
62              ss=' ';
63               sc=' ';
64          end Setlni;
66          /*-------------------------------------------**
67          |                  INpSS                     |
68          **-------------------------------------------*/
69          InpSS:                proc;
70          dcl  fs               file;
71               on endfile(fs);
72               read file(fs) into(rec_fs);
73               do while(^endfile(fs));
74                   sc=s_fs;
75                   call MkSS;
76                   read file(fs) into(rec_fs);
77               end;
78          end INpSS;
80          /*-------------------------------------------**
81          |                   MkSS                     |
82          **-------------------------------------------*/
83          MkSS:         proc;
84              ns=ns+1;
85               ss(ns)=sc;
86               ii(*,ns)=i_fs;
87          end MkSS;
89          /*-------------------------------------------**
90          |                  INpD                      |
91          **-------------------------------------------*/
92          INpD:         proc;
93          dcl  f                file record;
94               on endfile(f);
95               read file(f) into (rec_i);
```

```
96              do while(^endfile(f));
97                      m=m+1;
98                       call Scv;
99                      p=0;
100                 read file(f) into (rec_i);
101             end;
102         end INpD;
103
104         /*-------------------------------------------**
105         |                   Scv                     |
106         **-------------------------------------------*/
107         Scv:proc;
108             sc=rec_i.s;
109             call Scnv;
110         end Scv;
111         /*-------------------------------------------**
112         |                   Scnv                    |
113         **-------------------------------------------*/
114         Scnv:       proc;
116             /* FS file search */
117             call @Bsch(sc,ss(*),ns,p);
118             if p^=0
119               then call OutD;
120               else call OutE;
122         end Scnv;
124         /*-------------------------------------------**
125         |                   OutD                    |
126         **-------------------------------------------*/
127         OutD:       proc;
128         dcl fo      file;
129             i=1;
130             do while (i<=20);
131               if  ii(i,p) > 0 then do;
132                     rec_o=rec_i ,by name;
133
134                     /* I020 code set */
135                     s2 = i;
136                     rec_o.io = s2;
137
138                     /* value, quantity culculate and set */
139                     v2 = rec_i.v;
140                     q2 = rec_i.q;
141
142                      /* divide proportionally by frequency */
143                     if ii(21,p) ^= 0
144                       then fq = ii(i,p) / ii(21,p);
```

```
145                       else fq=1;
146                     v2 = v2 * fq;
147                     q2 = q2 * fq;
148                     rec_o.v = v2;
149                     rec_o.q = q2;
150
151                      /* rounding */
152                     if v2 - rec_o.v >= 0.5
153                       then rec_o.v = rec_o.v + 1;
154                     if q2 - rec_o.q >= 0.5
155                       then rec_o.q = rec_o.q + 1;
156
157                     /* do not write record if v=0 and q=0  */
158                     if (( rec_o.v ^= 0) | ( rec_o.q ^= 0) ) then
159                         write file(fo) from (rec_o);
160
161                     n=n+1;
162               end;
163
164               i=i+1;
165
166          end;
167       end OutD;
169       /*-----------------------------------------**
170       |                  OutE                   |
171       **-----------------------------------------*/
172       OutE:        proc;
173       dcl  fe      file;
174            rec_o = rec_i, by name;
175              write file(fe) from (rec_o);
176            ne=ne+1;
177       end OutE;
178
179       end p50s;
```

１.４　拡張変換表による２回目以降の変換処理（p5s.pli プログラム）

ここでエラー処理を最終的に完結させるプログラムである。p50s.pli の処理に加え、入力されるエラーファイルの SITC を末尾から一桁づつ削除していって拡張変換表を表引きする。p50s.pli と内容が似ており、Scv 以外の部分がほとんど同じなので、Scv 以外の部分はリストから割愛した。割愛した部分で p5s.pli と p50s.pli とが異なる点は、p5s.pli ではエラーデータは出力しないのでエラーデー

タに対応する記述が無い点だけである。(例えば rec_e, やエラー件数の表示など)

Scv は p50s.pli と大きく異なっている。ここでは、入力されるエラーデータの SITC を取り出し、最初の空白が何桁目に現れるか、あるいは無い (5 桁コード) かを調べる。(102 行) 次に、do while ループに入り、SITC の各桁が全て空白になる (SITC の最初の空白の位置が 1 になる。ix=1) か、あるいは、Scnv で拡張変換表に該当 SITC があった (p^=0) か、のいずれかの条件が満たされるまで (104 行)、SITC の末尾桁を空白にし (107～121 行)、その SITC で拡張変換表を表引きする処理 (Scnv。p50s.pli と同様。) を繰り返す。

```
  1       /*-------------------------------------------**
  2       # Program Name: p5s.pli
  3       # Language    : PL/I
  8       **-------------------------------------------*/
  9       p5s: proc options(main);
                :
           (  中略  )
                :
 95       /*-------------------------------------------**
 96       |                  Scv                      |
 97       **-------------------------------------------*/
 98       Scv:proc;
 99            sc=rec_i.s;
100
101            /* to set the location of space to variable ix  */
102              ix=index(sc,'20'x,1);
103
104            do while( (ix^=1) & (p=0)  );
105                 ix=index(sc,'20'x,1);
106
107              /* when location of space is not 0
108                     (containing 1-4 digits)
109                     omit the last digit  */
110
111                 if ix^=0 then do;
112                   substr(sc,ix-1,1)=' ';
113                end;
114
115              /* when location of space is  0
116                     (containing 5 digits)
117                     omit the last digit  */
```

```
118
119                     else do;
120                         substr(sc,5,1)=' ';
121                     end;
122
123                   /* search into converter file */
124                     call Scnv;
125
126                end;
127
128         end Scv;
                :
            (  中略  )
                :
185         end p5s;
```

２．金額加重配分による変換

金額加重配分方式のプログラムは均等配分方式のプログラムを元にして作成したため一部機能が同じところがあり、そのような部分は詳細を割愛する。一方、これらの方式は別々のディレクトリにプログラムファイルを保存しているため、仮に同じプログラム名でも全く別の内容であり関連は無いので、注意が必要である。

２.１　オリジナル変換表による変換処理

オリジナル変換表による変換処理は、均等配分方式とほとんど同じである。ただし、こちらではこの処理で出力されたパスデータから拡張変換表を作成するため、パスデータと同時に拡張変換表用のデータを別途出力している点だけが異なる。この拡張変換表用のデータは拡張変換表を作成するのに必要な項目のみをパスデータの項目から取捨選択したものである。

２.２　パスデータからの拡張変換表の作成

ここでは、パスデータと同時に出力された拡張変換表用のファイルを読み込んで、拡張変換表を出力する。処理ステップとしては表２の３つのプログラム

表2　パスデータからの拡張変換表の作成

スクリプト名	プログラム名	処理内容
p11.ex	p11.pli	オリジナル変換表の SITC を 1～5 桁にする。
p22.ksh	-	キー（報告国、輸出入区分、報告年、SITC、IO20、）の昇順にソートし金額を集計する。
p4.ex	p4.pli	拡張変換表を作成する。

を順番に行う。ここで、p11.pli の内容は入力データのフォーマットを除いて均等配分方式の p110.pli とほとんど同じなので割愛する。p22.ksh は、単にソートするだけでなく集計する機能、すなわち、金額を集計（summarize）して同一キーのレコードを 1 レコードとして出力する機能を含んでいる。これは Syncsort で /SUMMARIZE オプションを使用することにより実現している。p4.pli は以下にリストを掲げた。

p4.pli は基本的な機能は、均等配分方式の p4.pli と同様である。こちらに固有な特徴は次の 3 点である。（1）入力データの形式が異なっている。（2）コントロールキーが増えている。（3）拡張変換表に格納する IO20 の配列の値をパーセント値にしている。

```
 1        /*
 2        #-------------------------------------------#
 3         Program Name: p4.pli
 4         Language    : PL/I
18        #-------------------------------------------#
19        */
21        p4: proc options(main);
22        dcl  (@TmCP entry     (char(15),fixed bin(31))
23             )                external,
24            1 rec_i,
25              2 rc           char(6),
26              2 d            char(1),
27              2 y            pic'99',
28              2 s            char(5),
29              2 io           char(2),
30              2 v            pic'(12)-9',
31
32             1 rec_o,
33              2 rc           char(6),
```

```
34              2 d             char(1),
35              2 y             pic'99',
36              2 s             char(5),
37              2 fq(20)        pic'(6)-9',
38
39            rc2               char(6),
40            d2                char(1),
41            y2                pic'99',
42            s2                char(5),
43            io2               pic'99',
44
45            weight_value(20) float bin(53),
46            weight_value_sum float bin(53),
47            (k1,k2,v2)        float bin(53) init(0),
48            (m,n,i)           fixed bin(31) init(0);
49
50            on attn stop;
51            call INpD;
52            call @TmCP('inp_d',m);
53            call @TmCP('out',n);
54
55          /*-----------------------------------------**
56          |                  INpD                   |
57          **-----------------------------------------*/
58          INpD:         proc;
59          dcl f                 file record;
60               on endfile(f);
61               read file(f) into (rec_i);
62               do while(^endfile(f));
63                 call SetVC1;
64                 do while ( (rec_i.rc=rc2) & (rec_i.d=d2) &
65                            (rec_i.y=y2)   & (rec_i.s=s2) &
66                            (^endfile(f)) );
67                   call SetVC2;
68                   do while ((rec_i.rc=rc2) & (rec_i.d=d2) &
69                             (rec_i.y=y2)   & (rec_i.s=s2) &
70                             (rec_i.io=io2) & (^endfile(f) ) );
71                     call SumP;
72                     read file(f) into (rec_i);
73                   end;
74                   call OutD2;
75                 end;
76                call Percent_calc;
77                call OutD1;
78               end;
80          end INpD;
```

```
82        /*-----------------------------------------------**
83        |                   SetVc1                       |
84        **-----------------------------------------------*/
85        SetVC1:      proc;
86           rc2=rec_i.rc;
87           d2=rec_i.d;
88           y2=rec_i.y;
89           s2=rec_i.s;
90           k1=0;
91           weight_value=0;
92           weight_value_sum=0;
93           rec_o.fq=0;
94        end SetVC1;
96        /*-----------------------------------------------**
97        |                   SetVc2                       |
98        **-----------------------------------------------*/
99        SetVC2:      proc;
100          io2=rec_i.io;
101          k2=0;
102       end SetVC2;
104       /*-----------------------------------------------**
105       |                   SumP                         |
106       **-----------------------------------------------*/
107       SumP:        proc;
108          m=m+1;
109          v2=rec_i.v;
110          k2=k2+v2;
111       end SumP;
113       /*-----------------------------------------------**
114       |                   OutD2                        |
115       **-----------------------------------------------*/
116       OutD2:       proc;
117              k1=k1+k2;
118              if io2>20 then do;
119                    put skip list(s2,io2);
120                end;
121              else  weight_value(io2)=k2;
122       end OutD2;
124       /*-----------------------------------------------**
125       |                   OutD1                        |
126       **-----------------------------------------------*/
127       OutD1:       proc;
128       dcl fo       file;
129              rec_o.rc=rc2;
130              rec_o.d=d2;
131              rec_o.y=y2;
```

```
132               rec_o.s=s2;
133               write file(fo) from (rec_o);
134              n=n+1;
135         end OutD1;
136         /*-----------------------------------------**
137         |                Percent_calc               |
138         **-----------------------------------------*/
139         Percent_calc:proc;
141              /* 合計を計算する */
143               i=1;
144               do while (i<=20);
145                 weight_value_sum =
                                  weight_value_sum + weight_value(i);
146                 i=i+1;
147               end;
149              /* 各ウェイトのパーセンテージを計算する */
151               i=1;
152               do while (i<=20);
153                 if weight_value_sum = 0 then
154                     weight_value(i) = 0;
155                 else
156                     weight_value(i) =
                         weight_value(i) / weight_value_sum * 100000 ;
158                 i=i+1;
159               end;
161              /* 出力レコードに転記 */
163               i=1;
164               do while (i<=20);
166                 fq(i) = weight_value(i);
168                 /* 小数点第１位を四捨五入 */
170                 if  ( weight_value(i) - fq(i) ) <  0.5
171                     then  fq(i) = fq(i) + 0;
172                 else      fq(i) = fq(i) + 1;
174                 i=i+1;
175               end;
177         end Percent_calc;
178         end p4;
```

２.３　拡張変換表での１回目の変換処理

ここでは、表３の２つの処理を順番に行う。

表3　拡張変換表での1回目の変換処理

スクリプト名	プログラム名	処理内容
p6.ksh	-	キー（報告国、輸出入区分、報告年、SITC、相手国、数量単位）の昇順にソートし、金額、数量を集計する。
p7.ex	p7s.pli	AID-XT データと拡張変換表とのマッチング。

p7s.pli プログラムは、AID-XT エラーデータと拡張変換表とのファイルマッチング（突き合わせ）処理を行う。ReadE で AID-XT エラーデータ読み込みを行い、ReadT で拡張変換表データの読み込みを行う。INpD が主処理ルーチンであり、ここで一般的なマッチングのパターンを使っている。

ところで、一般的なマッチングの手法を用いてトランザクションデータ（トランデータ）によってマスタデータを更新するアルゴリズムを簡単に記述すると、以下のようなものになる。

```
１．トランデータとマスタデータをそれぞれキー順にソートする。
２．マッチング処理
  call トランデータ読み込み
  call マスタデータ読み込み
  do while（トランデータとマスタデータが EOF になるまで）;
       if    （トランデータのキー＞マスタデータのキー）then
             call マスタデータ読み込み;
       else if（トランデータのキー＜マスタデータのキー）then
             call エラーデータ書き込み
             call トランデータ読み込み;
       else if（トランデータのキー＝マスタデータのキー）then do;
             call マスタデータ更新
             call マスタデータ読み込み;
             call トランデータ読み込み;
       end;
  end;
```

なお、このアルゴリズムではデータの読み込み処理でファイルの終端(EOF)に達したときには、キーに High-Value（0xFF）を入れることになっている。

INpD では基本的にはこのアルゴリズムに従っているが、若干変更したものとしている。上記の「トランデータ」は AID-XT データ、「マスタデータ」を拡

張変換表データと読み替えれば、変更点は、（AID-XT データのキー＝拡張変換表データのキー）の条件に入った場合、拡張変換表データの読み込みは行わないようにコメントアウトした（128 行）ことである。

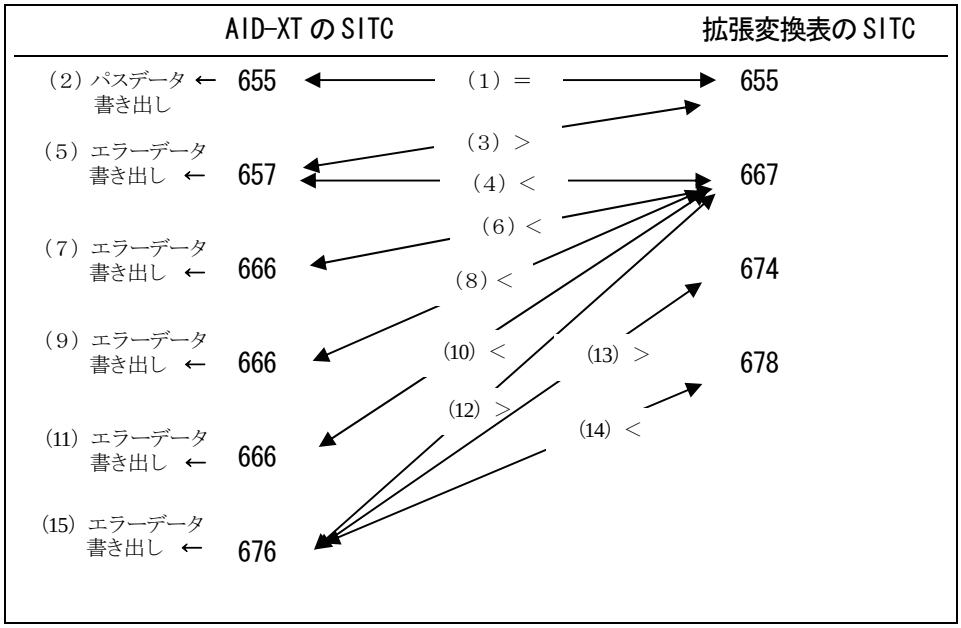
なぜこのような変更を行ったかを以下に説明する。表 4 は入力ファイルの AID-XT データのキーの中の SITC を左側に示し、拡張変換表の SITC を右側に示している。これ以外のキー項目（報告国、輸出入区分、報告年）は 2 つのファイル間で一致しているものとする。なお、この SITC は実際のデータの値ではなく、あくまでも例示するために作成したものである。

表 4　マッチングするデータの例

AID-XT の SITC	拡張変換表の SITC
655	655
657	667
666	674
666	678
666	
676	

表 4 は極端な例ともいえるが、報告国、輸出入区分、報告年をキーに加えた場合、実際の SITC の組合せにはこのような場合が存在する。すなわち、SITC=657 のように拡張変換表に該当する SITC が存在しない場合があり、さらに、SITC=666 のように AID-XT データにおいて、キーの重複がありうることである。これは、p7s.pli の前に実行される AID-XT データのソート・集計処理（p6.ksh）においてソート集計キーに数量単位を加えているからである。
表 4 の組合せの場合に正しくマッチングされた場合、表 5 のようにマッチングが行われ、エラーデータが出力される必要がある。表 5 は、矢印で結ばれた値がマッチングで比較されることを示す。カッコ内の数字が処理の順番であり、

表５　マッチングの例



比較された結果が不等号で示される。例えば（1）AID-XT データの SITC=655 と拡張変換表データの SITC=655 が比較され等しいので、(2) パスデータとして出力される。次に(3)AID-XT データのみを読み込み、AID-XT データの SITC=657 と拡張変換表データの SITC=655 が比較され AID-XT データの方が大きい（＞）ので、(4) 拡張変換表データ（SITC=667）が読み込まれて AID-XT データとキー比較を行い、拡張変換表データのキーが大きい（＜）ので、(5) AID-XT データをエラーデータとして書き出す。以下同様に繰り返す。

前述の変更点は例のように AID-XT データがソートキーに数量単位を含んだソート集計をしているために重複したキーを持つために､設定したものである。例では AID-XT データの SITC=666 が重複キーとなっているが、ここで仮に拡張変換表に SITC=666 が存在した場合、AID-XT データの SITC=666 の最初のデータのみがパスデータとなり、それ以外は読み飛ばされるかエラーデータとなってしまうことを考えれば明らかである。

```
1       /*-------------------------------------
3       # Program Name: p7s.pli
4       # Language    : PL/I
```

```
22        ------------------------------------------*/
23
24        p7s: proc options(main);
25        dcl    ( @TmCP entry    (char(15),fixed bin(31))
26               ) external;
27
28        dcl  fei      file record;      /* File error input */
29        dcl  fti      file record;      /* File table input */
30        dcl  fpo      file record;      /* File passed output */
31        dcl  feo      file record;      /* File error output */
32
33        /* p1.ex でエラーになったデータを入力する */
35        dcl  1 rec_error_input,
36                 2 rc          char(6),
37                 2 d           char(1),
38                 2 io          char(2),
39                 2 s           char(5),
40                 2 pc          char(6),
41                 2 y           char(2),
42                 2 v           pic'(12)-9',
43                 2 qu          char(2),
44                 2 q           pic'(12)-9';
45
46        /*  p4.ex で出力された変換用テーブル*/
48        dcl    1 rec_table_input,
49                 2 rc          char(6),
50                 2 d           char(1),
51                 2 y           char(2),
52                 2 s           char(5),
53                 2 fq(20)      pic'(6)-9';
54
55        /*  マッチングの結果マッチしたパスデータを出力する*/
57        dcl  1 rec_pass_output like rec_error_input;
58
59      /*  マッチングの結果マッチしなかったエラーデータを出力する*/
61        dcl  1 rec_error_output like rec_error_input,
62
63             io20_99                  pic'99' init(0),
64             fq_float                 float bin(53) init(0),
65             value_float              float bin(53) init(0),
66             quantity_float           float bin(53) init(0),
69          /* これらは厳密には High-value, Low-value ではないが,
70             char 属性では,0xFF,0x00 を正しく比較できないのでこの
               ようにした. */
72             high_value               char(14) init((14)'~'),
73             low_value                char(14) init((14)' '),
```

```
74
75              cnt_table_input          fixed bin(31) init(0),
76              cnt_error_input          fixed bin(31) init(0),
77              cnt_pass_output          fixed bin(31) init(0),
78              cnt_error_output fixed bin(31) init(0),
79
80              key_error_input          char(14) init(' '),
81              key_error_input_old      char(14) init(' '),
82              key_table_input          char(14) init(' ');
83
84              on attn stop;
85              call INpD;
86              call @TmCP( 'error inp cnt', cnt_error_input );
87              call @TmCP( 'table inp cnt', cnt_table_input );
88              call @TmCP( 'pass  out cnt', cnt_pass_output );
89              call @TmCP( 'error out cnt', cnt_error_output );
90
91          /*-------------------------------------------**
92          |                   INpD                     |
93          **-------------------------------------------*/
94          INpD:proc;
95          dcl  fei      file record;
96          dcl  fti      file record;
97          on endfile( fei );
98          on endfile( fti );
100         call ReadE;
101         call ReadT;
102
103         do while (^endfile( fei )) ;
104           if   ( key_error_input > key_table_input ) then do;
106              call ReadT;
109           end;
110           else if ( key_error_input < key_table_input ) then do;
111              call OutE;
117              call ReadE;
120           end;
121           else if ( key_error_input = key_table_input ) then do;
123              call OutD;
124              call ReadE;
128              /*call ReadT; */
132           end;
133         end;
134         end INpD;
135
136          /*-------------------------------------------**
137          |                   ReadE                    |
```

```
138         **-------------------------------------------*/
139         ReadE: proc;
140              key_error_input_old = key_error_input;
142              read file( fei ) into (rec_error_input);
143              substr( key_error_input, 1, 6) = rec_error_input.rc;
144              substr( key_error_input, 7, 1) = rec_error_input.d;
145              substr( key_error_input, 8, 2) = rec_error_input.y;
146              substr( key_error_input,10, 5) = rec_error_input.s;
148              if (^endfile( fei ))
149                     then cnt_error_input = cnt_error_input + 1;
150                     else key_error_input = high_value;
153         end ReadE;
155         /*-------------------------------------------**
156         |                  ReadT                     |
157         **-------------------------------------------*/
158         ReadT: proc;
159              read file( fti ) into ( rec_table_input );
160              substr( key_table_input, 1, 6) = rec_table_input.rc;
161              substr( key_table_input, 7, 1) = rec_table_input.d;
162              substr( key_table_input, 8, 2) = rec_table_input.y;
163              substr( key_table_input,10, 5) = rec_table_input.s;
165              if (^endfile( fti ))
166                     then  cnt_table_input = cnt_table_input + 1;
167                     else  key_table_input = high_value;
170         end ReadT;
172         /*-------------------------------------------**
173         |                  OutE                      |
174         **-------------------------------------------*/
175         OutE:  proc;
176         dcl  feo     file record;
177                rec_error_output = rec_error_input ,by name;
178                write file(feo) from (rec_error_output);
179              cnt_error_output = cnt_error_output + 1;
182         end OutE;
183         /*-------------------------------------------**
184         |                  OutD                      |
185         **-------------------------------------------*/
186         OutD:  proc;
187         dcl fpo  file record,
188             i  fixed bin(31) init(0); /* roop counter (up to 20) */
190         i=1;
191         do while (i<=20);
192             if  fq(i) > 0 then do;
193                 rec_pass_output = rec_error_input ,by name;
195                 io20_99 = i;
196                 rec_pass_output.io = io20_99;
```

```
198                 /* 金額数量を計算してセットする */
200                 fq_float = rec_table_input.fq(i) / 100000.0 ;
202                 value_float = rec_error_input.v;
203                 value_float = value_float *  fq_float;
204                 rec_pass_output.v = value_float;
206                 quantity_float = rec_error_input.q;
207                 quantity_float = quantity_float * fq_float;
208                 rec_pass_output.q = quantity_float;
210                /* 四捨五入 */
212                 if value_float - rec_pass_output.v >= 0.5
213                   then  rec_pass_output.v = rec_pass_output.v + 1;
215                 if quantity_float - rec_pass_output.q >= 0.5
216                   then rec_pass_output.q = rec_pass_output.q + 1;
218                 /* 金額 = 0 かつ 数量 = 0 のとき以外は書き出す */
220                 if ( ( rec_pass_output.v ^= 0) |
221                      ( rec_pass_output.q ^= 0) ) then do;
222                    write file(fpo) from (rec_pass_output);
223                    cnt_pass_output = cnt_pass_output + 1;
226                 end;
227               end;
229               i=i+1;
231         end;
232         end OutD;
233
234         end p7s;
```

２.４　拡張変換表での２回目以降の変換処理

ここでは、２－３で前述した１回目の変換処理からエラーデータとなっておりてきたAID-XTデータのSITCを末尾１桁を削除してマッチングをする処理で、繰り返し行う処理である。

表６　拡張変換表での２回目以降の変換処理

スクリプト名	プログラム名	処理内容
p51.ex	p5s.pli	AID-XTデータのSITCを1～5桁にする。
p61.ksh	-	キー（報告国、輸出入区分、報告年、SITC、相手国、数量単位）の昇順にソートし、金額、数量を集計する。
p71.ex	p7s.pli	AID-XTデータと拡張変換表とのマッチング。

内容は、AID-XT データの SITC の末尾を削除する処理 p51.ex を最初に行う。次に 1 回目の変換処理と同様にソート・集計処理（p61.ksh）、マッチング処理（p7s.pli）が続く。

p5s.pli は、入力データのフォーマットが異なるが処理内容は１－１で前述した p110.pli とほぼ同等である。また、p6.ksh と p7s.pli の内容は前述した２－３のものと同等である。

２.５　最終エラーデータの変換処理とパスデータのソート・集計処理

p8.ksh で最後に残ったエラーデータの SITC を暫定的に'99' などの未使用コードに置き換える。これは Syncsort の/DERIVEDFIELD オプションで実現している。そして p9.ksh でこれまでに出力されたパスデータと最終エラーデータをソート・集計してファイルを一本化して出力する。（表 7）

表 7　最終エラーデータの変換処理とパスデータのソート・集計処理

スクリプト名	プログラム名	処理内容
p8.ksh	-	IO20 に固定値 '99'を設定して出力する。
p9.ksh	-	パスデータ、最終エラーデータをキー（報告国、輸出入区分、IO20、SITC、相手国、報告年、数量単位）の昇順にソートし、金額、数量を集計する。

３．　おわりに

ここで取り上げたプログラムはいずれも基本的なプログラミング手法を若干アレンジしたもので特に技術的に難易度の高いものではない。そのため比較的応用がしやすく、商品分類の産業分類への変換以外の用途にも使用できるであろう。

一方これらは IO20 が 20 種類しかないコードだったので可能なプログラムだったといえる。いわば n 対 few の配分の処理である。仮に IO20 が数百、数千

種類あるようなコード体系だったとしたら、この処理手順では対応できない。これはより一般的な n 対 n の配分の問題として残されている。

また、これらのプログラムは全て PL/I で記述されている。実行時にも特にパフォーマンス上の問題は発生していない。しかし、コードの互換性や保守性を考慮して、C++や Java などの最近の言語で記述し直すことが必要となるだろう。

【参考文献】

［1］古河俊一、野田容助編『標準国際貿易商品分類と産業分類の対応関係』統計資料シリーズ No.80、アジア経済研究所、1998

［2］黒子正人、商品分類の産業分類への変換－変換エラーデータの処理－、（野田容助編『世界貿易マトリクス－国際産業連関表 24 部門にもとづいて－』、統計資料シリーズ No.84、アジア経済研究所、2002）