

J.-P. BENZÉCRI

F. BENZÉCRI

Sources de programmes d'analyse de données en langage PASCAL : (III) : élaboration de tableaux divers (IIIF) : sous-tableaux, juxtapositions, présentation de tableaux ternaires

Les cahiers de l'analyse des données, tome 22, n° 4 (1997), p. 363-374

http://www.numdam.org/item?id=CAD_1997__22_4_363_0

© Les cahiers de l'analyse des données, Dunod, 1997, tous droits réservés.

L'accès aux archives de la revue « Les cahiers de l'analyse des données » implique l'accord avec les conditions générales d'utilisation (<http://www.numdam.org/conditions>). Toute utilisation commerciale ou impression systématique est constitutive d'une infraction pénale. Toute copie ou impression de ce fichier doit contenir la présente mention de copyright.

NUMDAM

Article numérisé dans le cadre du programme
Numérisation de documents anciens mathématiques

<http://www.numdam.org/>

**SOURCES DE PROGRAMMES
D'ANALYSE DE DONNÉES EN LANGAGE PASCAL:
(III) : ÉLABORATION DE TABLEAUX DIVERS
(IIIF) : SOUS-TABLEAUX, JUXTAPOSITIONS,
PRÉSENTATION DE TABLEAUX TERNAIRES**

[SOURCES PASCAL (IIIF)]

J.-P. & F. BENZÉCRI

**0 Introduction : fonctions des programmes 'soustab'; 'juxtab';
'wITxJ'**

L'objet de la présente note est de publier, avec un très bref commentaire, trois programmes dont les fonctions, pour simples qu'elles soient, ne sont pas commodément assurées par les logiciels usuels. Le lecteur appréciera par lui-même nos outils! nous serions comblés si, en les améliorant, il contribuait à créer ce qu'on a souvent appelé: le tableur de l'avenir.

De façon précise, 'soustab' a pour fonction de créer des sous-tableaux d'un tableau rectangulaire de nombres réels ou entiers, donné en format binaire (.w ou .z). On retrouve les deux modes de désignation de sous-ensembles, par dialogue ou par fichier, déjà utilisés en analyse de correspondance ('qori': IB) et pour les cumuls de blocs de lignes et colonnes (IIIE)

Le programme 'juxtab' a deux fonctions: juxtaposition proprement dite, i.e. construction du tableau: $I \times (J_1 \cup J_2)$, à partir des tableaux $I \times J_1$ et $I \times J_2$; et superposition, i.e. construction de $(I_1 \cup I_2) \times J$, à partir de $I_1 \times J$ et $I_2 \times J$.

L'objet du programme 'wITxJ', dont le nom même évoque une présentation rectangulaire d'un parallélépipède, est de faire communiquer entre elles les diverses présentations rectangulaires d'un tableau ternaire; en offrant à l'utilisateur trois voies: {L, C, D}:

L : transformer: $(I \times T) \times J$ en: $(T \times I) \times J$ (ou réciproquement); et calculer, s'il y a lieu, un tableau de marge: $I \times J$, ou: $T \times J$;

C: transformer: $I \times (J \times T)$ en: $I \times (T \times J)$ (ou réciproquement);

D: transformer: $(T \times I) \times J$ en $(T \times J) \times I$; avec la restriction que $\text{card}(T \times J)$ et $\text{card}I$ soient $\leq \text{jmax}=300$.

On notera que la notice [ÉTAT MacSAIF] (in CAD, Vol.XVIII, n°3; 1993) décrit une version de 'wITxJ' où manque la voie: D.

1 Le programme 'soustab' d'extraction de sous-tableaux

```

program soustab;
uses memtypes, quickdraw, osintf, toolintf, sane, uver5, uens5;
var finu:file of integer;ft:text;
    suf, sgu, sqg, nomba, nomf, chaine, filtre:string;
    rps, rpf, rpb, rpc, rpu, chk, carac, rpfcr:char;
    i, ia, ii, ic, j, a, b, c, u, carui, caruj, caruk, carqi, carqj, carqk, erl, ie, frci:integer;
    sli, slj, slk:pzgi;gai, gaj, gak:ptni;
    tampi:array[1..titab]of integer;tam3:array[1..3]of integer;f1:longint;
    lmt,adj:string[9];ttri:stringptr;pp:ptr;
    psgl:pzgl;
procedure nommer;          begin
    rpc:='N';erl:=0;sgu:='';
    while not ((rpc='O') or (erl=6)) do begin rpc:='O';
    write('le nom du tableau à lire est ');readln(nomba);
    suf:='.z';nomf:=concat(nomba, suf);
    if (verif(@nomf)=1) then begin
        suf:='.w';nomf:=concat(nomba, suf) end;
    rpc:=dialof(stringptr(@nomf));
    if (rpc='N') then erl:=erl+1 else begin writeln(nomf);
        write('ce nom est-il confirmé oui(O) ou non(N) ');
        readln(rpc) end end;
    c:=length(nomba);
    if (3<c) then sgu:=copy(nomba,c-2,3) else sgu:='';
    if (sgu='Fac') then nomba:=copy(nomba,1,c-3) else sgu:='';
end;

```

Dans la procédure 'nommer', l'utilisateur désigne un tableau de base dont on extraira des sous-tableaux; le format doit être: .z ou .w; dans le cas particulier d'un fichier de facteurs, le programme prévoit de conserver, dans les noms des sous-tableaux créés, la désinence: 'Fac'; sinon le suffixe propre au sous-tableau sera placé immédiatement à la fin de: 'nomba'.

```

procedure ouvrir;begin
    reset(finu,concat(nomba, sgu, suf));writeln(concat(nomba, sgu, suf));
    for a:=1 to titab-1 do read(finu,tampi[a]);writeln(ttri^);
    read(finu,carui);read(finu,caruj);
    if (ibmax<carui) then new(sli) else sli:=pzgi(newptr(6*carui));
    gai:=ptni(newptr(2*carui));
    if (ibmax<caruj) then new(slj) else slj:=pzgi(newptr(6*caruj));
    gaj:=ptni(newptr(2*caruj));
    writeln('SIGLES des Colonnes');
    for j:=1 to caruj do begin
        for a:=1 to 3 do read(finu,tam3[a]);
        slj^[j]:=psgl^;write(sigler(psgl^):5);
        if (j mod 9=0) then writeln end;
    if not (caruj mod 9=0) then writeln;
    writeln('SIGLES des Lignes');
    for i:=1 to carui do if not eof(finu) then begin
        for a:=1 to 3 do read(finu,tam3[a]);
        sli^[i]:=psgl^;write(sigler(sli^[i]):5);
        for a:=1 to 2*caruj do read(finu,c);
        if (i mod 12=0) then writeln end;
    close(finu);
    if not (carui mod 12=0) then writeln;end;

```

La procédure 'ouvrir' affiche à l'écran la liste des sigles des lignes et celle des sigles des colonnes.

```

procedure lecnombre;begin ffl:=0;
  while not ((ord(carac) in [48..57]) or eof(ft)) do read(ft,carac);
  while (ord(carac) in [48..57]) do begin
    ffl:=(10*ffl)+(ord(carac)-48);
    if eof(ft) then carac:=' ' else read(ft,carac) end;end;
procedure enslister;begin
  for i:=1 to caruk do gak^[i]:=0;carqk:=0;
  chaine:=concat(nomba,sgq,sgu,'ens',chk);
  ie:=verif(stringptr(@chaine));
  if not (ie=2) then
    ensembler(caruk,@lmt,@adj,@slk^,@gak^);
  if (ie=2) then begin
    reset(ft,chaine);readln(ft,chaine);ia:=0;carac:=' ';
    while not eof(ft) do begin lecnombre;i:=ffl;ic:=0;
      if (i>10000) then begin i:=i mod 10000;ic:=1;end;
      if (ia=0) then ia:=i;
      if not ((i<=0) or (caruk<i)) then
        for ii:=ia to 1 do gak^[ii]:=1;
      ia:=0;if (ic=1) then ia:=i end;
      close(ft) end;
    for i:=1 to caruk do carqk:=carqk+gak^[i];end;

```

La procédure 'lecnombre, dont des variantes ont été vues (depuis IA§2.3), sert ici à lire des entiers sur un fichier de commande spécifiant un sous-ensemble. Par 'enslister', le programme cherche d'abord s'il existe un tel listage pour un des ensembles (chk='i', pour I; chk='j' pour J); le nom du listage devant être celui du tableau à créer, avec le suffixe: 'ensi' ou 'ensj'. Si le listage existe, on y lit, après une ligne de titre des numéros d'individus isolés ou des blocs (ceux-ci annoncés par un nombre de la forme: 10000+i; cf. IB§3.2.1). Sinon, le sous-ensemble est défini par la procédure: 'enssembler' de dialogue à l'écran: cf. IA§3.

```

procedure bloquer;begin
  rps:='N';
  while not (rps='O') do begin rps:='N';
    write('le sigle du sous tableau à créer est ');readln(chaine);
    if (4<length(chaine)) then sgq:=copy(chaine,1,4) else sgq:=chaine;
    for c:=1 to length(sgq) do if ((sgq[c]=':') or (sgq[c]=' ')) then sgq[c]:='1';
    if (sgq='') then sgq:='*';writeln(sgq);
    write('ce sigle est il confirme oui(O) ou non(N) ');
    readln(rps) end;
  frci:=0;rpfr:='N';

```

La procédure 'bloquer' a pour rôle de spécifier complètement le sous-tableau à créer. Un premier dialogue fixe le sigle (de longueur ≤4).

```

while not (rpfr='O') do begin
  write('faut-il ne considérer qu'une fraction des lignes O ou N ');
  readln(rpfr);if not (rpfr='O') then frci:=0;
  if (rpfr='O') then begin
    writeln('éliminer un demi(-2), un tiers(-3) ou un quart(-4)');
    write('ou ne garder qu'un demi(2), un tiers(3), un quart(4) ');
    readln(frci);if (frci=0) then frci:=2;
    if (frci<-4) then frci:=-4;if (4<frci) then frci:=4;
    if (frci=-1) then frci:=-2;if (frci=1) then frci:=2;
    if (0<frci) then writeln('      on ne gardera que 1/',frci);
    if (frci<0) then writeln('      on éliminera 1/',abs(frci)) end;
  write('le choix est' il confirmé O ou N ');readln(rpfr) end;

```

Un second dialogue propose à l'utilisateur de restreindre, à une fraction, le sous-ensemble des lignes conservées: ainsi, en analyse discriminante, on peut distinguer entre échantillon de base et échantillon d'épreuve (cf. IIA).

```
lmt:='colonne';caruk:=caruj;chk:='j';gak:=gaj;slk:=slj;
enslister;carqj:=carqk;writeln('carqj =',carqj:4);
lmt:='ligne';caruk:=carui;chk:='i';gak:=gai;slk:=sli;
enslister;carqi:=carqk;writeln('carqi =',carqi:5) end;
```

Et 'bloquer' se termine par deux appels à 'enslister' (cf. *supra*), pour fixer l'ensemble des lignes et l'ensemble des colonnes retenues. On notera que, s'il y a lieu, la restriction à une fraction s'applique à l'ensemble ainsi spécifié.

```
procedure creer;var finq:file of integer;
begin
  if not(frci=0) then begin
    if (0<frci) then for i:=1 to carui do
      if not((i mod frci)=1) then gai^[i]:=0;
    if (frci<0) then for i:=1 to carui do
      if ((i mod (-frci))=1) then gai^[i]:=0;end;
  rewrite(finq,concat(nomba,sgq,sgu,suf));
  writeln(concat(nomba,sgq,sgu,suf));
  for u:=1 to 64 do begin b:=tampi[u];
    write(finq,b) end;
  c:=0;
  if (sgu='Fac') then for j:=1 to caruj do
    if ((gaj^[j]=1) and (j<abmax+1)) then begin c:=c+2;
      write(finq,tampi[63+(2*j)]);
      write(finq,tampi[64+(2*j)]) end;
    for u:=65+c to titab-1 do write(finq,c);
    write(finq,carqi);write(finq,carqj);
    reset(finu,concat(nomba,sgu,suf));
    for u:=1 to titab+1 do read(finu,c);
    for j:=1 to caruj do begin read(finu,a,b,c);
      if (gaj^[j]=1) then write(finq,a,b,c) end;
    for i:=1 to carui do if (gai^[i]=1) then begin
      for u:=1 to 3 do begin read(finu,tam3[u]);write(finq,tam3[u]) end;
      writeln(sigler(psg1^));
      for j:=1 to caruj do begin read(finu,a);read(finu,b);
        if (gaj^[j]=1) then begin write(finq,a);write(finq,b) end end end
      else for a:=1 to 3+(2*caruj) do read(finu,c);
    writeln;close(finu);close(finq);end;
```

Comme son nom l'indique, 'creer' crée le fichier du sous-tableau spécifié par 'bloquer'. D'abord, en cas de restriction à une fraction ($frci \neq 0$), l'ensemble, vers lequel pointe: gai^i , des lignes à conserver, est restreint par la condition arithmétique appropriée. Ensuite, est ouvert en écriture le fichier: finq, du sous-tableau à créer; et on en écrit l'en-tête, d'après les informations lues en tête du tableau de base, par la procédure 'ouvrir'. Le cas d'un fichier de facteurs se distingue en ce qu'il faut écrire, comme une suite continue, les valeurs propres afférentes aux facteurs conservés (cf. IB§2.3.1). On rouvre alors en lecture le fichier: finu, du tableau de base, afin d'y copier les sigles et valeurs qui sont conservées dans le sous-tableau (les données de toute nature étant comme de règle, lues et comme des paires d'octets, sous le format: integer: cf. IA§2.5.1).

```

begin; rpu:='O';benzecri;
adj:='à garder';ttri:=stringptr(@tampi);psgl:=pzgl (@tam3);
while not (rpu='N') do begin
  nommer;rpu:='N';
  if (rpc='O') then begin ouvrir; rpf:='O';
    while not (rpf='N') do begin
      bloquer;
      creer;
      write('faut-il créer un autre sous-tableau de ',nomba,sgu,' O ou N ');
      readln(rpf) end;
      dispose(s1i);dispose(gai);dispose(s1j);dispose(gaj);
      write('faut-il créer des sous-tableaux d'un autre tableau O ou N ');
      readln(rpu) end end;end.

```

Le programme principal est très simple. La locution: 'à garder'; est fixée pour les appels de 'ensembler'; deux pointeurs reçoivent leur type; et l'on entre dans la boucle générale: while not (rpu='N')...; le tableau de base est désigné par 'nommer'; l'en-tête et les sigles en sont lus par 'ouvrir'; et l'on entre dans la boucle: while not (rpf='N')... pour en créer autant de sous-tableaux qu'on le veut; chaque sous-tableau étant spécifié par 'bloquer'

2 Le programme 'juxtab' de juxtaposition et superposition de tableaux

```

program juxtab;
uses memtypes, quickdraw, osintf, toolintf, sane, uver5;
var fu1, fu2, fiq: file of integer;
    suf, sgu, sgq, sfc, lico, coli, nomba, nomb2, nomf, chaine: str255;
    rps, rpf, rpb, rpc, rpu, rpo, rvr: char;
    i, j, a, b, c, u, car1i, car2i, carqi, car1j, car2j, carqj, dif, tro, erl, ilu: integer;
    tampi, tambi: array[1..titab] of integer; tam3: array[1..3] of integer;
    ttri, ttsi: stringptr; pp: ptr; psgl: pzgl;
begin ; rpu:='O';benzecri;
ttri:=stringptr(@tampi);ttsi:=stringptr(@tambi);
psgl:=pzgl (@tam3);
writeln('ce programme crée un tableau à partir de 2 autres');
writeln('par juxtaposition(J) ou superposition(V)');
while not (rpu='N') do begin rpo:='N';
  while not (rpo in ['J','V']) do begin
    write('faut il juxtaposer(J) ou superposer(V) 2 tableaux ');
    readln(rpo); if not (rpo in ['J','V']) then
      writeln('ERREUR la réponse doit être J ou V') end;
    if (rpo='J') then lico:='ligne' else lico:='colonne';
    if (rpo='V') then coli:='ligne' else coli:='colonne';
    if (rpo='J') then begin
      write('faut il vérifier la concordance des sigles des lignes O ou N ');
      readln(rvr); end;

```

Le programme consiste en une boucle générale: while not (rpu='N')..., qui offre d'abord à l'utilisateur le choix entre juxtaposition et superposition; avec, dans le premier cas, vérification possible de la concordance des sigles des lignes entre les deux tableaux (car il se peut que, dans les deux tableaux: $I \times J1$ et $I \times J2$, les éléments du même ensemble I soient désignés par des sigles différents: e.g. s'il s'agit de données afférentes à deux périodes différentes).

On désigne ensuite les deux tableaux à combiner; et le sigle du tableau à créer; non sans exiger que le nom de celui-ci soit distinct de celui de chacun

```

rpc:='N';erl:=0;
while not ((rpc='O') or (erl=6)) do begin rpc:='O';
write('le nom du premier tableau à prendre est ');readln(nomba);
suf:='.z';nomf:=concat(nomba,suf);
if (verif(@nomf)=1) then begin
suf:='.w';nomf:=concat(nomba,suf) end;
rpc:=dialof(stringptr(@nomf));
if (rpc='N') then erl:=erl+1 else begin writeln(nomf);
write('ce nom est il confirmé oui(O) ou non(N) ');
readln(rpc) end end;
if (rpc='O') then begin rpc:='N';erl:=0;
while not ((rpc='O') or (erl=6)) do begin rpc:='O';
write('le nom du deuxième tableau à prendre est ');readln(nomb2);
if (nomb2=nomba) then begin rpc:='N';
writeln('ERREUR les 2 tableaux doivent être distincts') end
else begin nomf:=concat(nomb2,suf);
rpc:=dialof(stringptr(@nomf)) end;
if (rpc='N') then erl:=erl+1 else begin writeln(nomf);
write('ce nom est il confirmé oui(O) ou non(N) ');
readln(rpc) end end end;
if (rpc='O') then begin rpc:='N';
while not (rpc='O') do begin rpc:='N';
write('le suffixe du tableau a créer est ');readln(chaine);
if (4<length(chaine)) then sqq:=copy(chaine,1,4) else sqq:=chaine;
for c:=1 to length(sqq) do if ((sqq[c]='.') or (sqq[c]=' ')) then sqq[c]:='1';
if (sqq='') then sqq:='*';
if (concat(nomba,sqq)=nomb2) then
writeln('ERREUR le tableau créé doit être distinct des 2 autres')
else begin writeln(sqq);
write('ce sigle est il confirmé oui(O) ou non(N) ');
readln(rpc) end end end;
if not (rpc='O') then rpo:='N';

```

des deux dont il se compose. Puis on ouvre les tableaux à combiner, afin de vérifier la compatibilité des dimensions (cardJ1= cardJ2; ou cardI1=cardI2); et le maximum imposé au nombre des colonnes ou des lignes.

```

if (rpc='O') then begin
reset(fiu1,concat(nomba,suf));writeln(concat(nomba,suf));
for a:=1 to titab-1 do read(fiu1,tampi[a]);
writeln('titre du premier tableau:');writeln(tttri^);
read(fiu1,carli);read(fiu1,carlj);
reset(fiu2,concat(nomb2,suf));writeln(concat(nomb2,suf));
for a:=1 to titab-1 do read(fiu2,tambi[a]);
writeln('titre du deuxième tableau:');writeln(tttsi^);
read(fiu2,car2i);read(fiu2,car2j);tro:=0;
if (rpo='J') then begin dif:=carli-car2i;
if (jmax<carlj+car2j) then tro:=1 end;
if (rpo='V') then begin dif:=carlj-car2j;
if (ipmax<carli+car2i) then tro:=1 end;
if not (dif=0) then
writeln('ERREUR les tableaux n'ont pas le même nombre de ',lico,'s');
if not (tro=0) then
writeln('ERREUR le tableau créé aurait trop de ',coli,'s');
if not ((dif=0) and (tro=0)) then begin
rpc:='N';rpo:='N';close(fiu1);close(fiu2) end end;

```

Toutes vérifications étant faites, on entre dans la boucle de création proprement dite: if (rpc='O') then begin... D'abord est ouvert en écriture le fichier du nouveau tableau (dont le nom est fixé en tenant compte du cas de la

superposition de deux tableaux de facteurs; comme on l'a fait, dans 'soutab', pour un sous-tableau). Et l'on inscrit le titre et les dimensions du nouveau tableau. Puis l'on copie les données proprement dites (sigles et nombres) en distinguant les deux cas de la juxtaposition et de la superposition.

```

if (rpc='O') then begin
  nomf:=concat (nomba,sgq,suf);sfc:='';
  if ((rpo='V') and (suf='.w')) then begin
    a:=length(nomba);b:=length(nomb2);
    if (a>3) then sfc:=copy(nomba,a-2,3);
    if ((b>3) and (sfc='Fac')) then sfc:=copy(nomb2,b-2,3);
    if (sfc='Fac') then nomf:=concat(copy(nomba,1,a-3),sgq,sfc,suf) end;
  rewrite(fig,nomf);
  writeln('création du tableau ',nomf);
  for a:=1 to titab-1 do write(fig,tampi[a]);
  if (rpo='J') then begin carqi:=carli;carqj:=carlj+car2j end;
  if (rpo='V') then begin carqi:=carli+car2i;carqj:=carlj end;
  write(fig,carqi);write(fig,carqj) end;
if (rpo='J') then begin i:=0;
  for a:=1 to 3*carlj do begin read(fiu1,c);write(fig,c) end;
  for a:=1 to 3*car2j do begin read(fiu2,c);write(fig,c) end;
  while not ((rpo='N') or (i=carqi)) do begin i:=i+1;
    for a:=1 to 3 do read(fiu1,tam3[a]);sgu:=sigler(psgl^);
    for a:=1 to 3 do read(fiu2,tam3[a]);
    if ((sgu=sigler(psgl^)) or (rvr='N')) then begin
      writeln(lico,i:5,' sigle ',sgu);
      for a:=1 to 3 do write(fig,tam3[a]);
      for a:=1 to 2*carlj do begin read(fiu1,c);write(fig,c) end;
      for a:=1 to 2*car2j do begin read(fiu2,c);write(fig,c) end end
    else begin
      writeln('ERREUR sigles différents ',lico,i:5,': ',sgu,' et ',sigler(psgl^));
      rpo:='N' end end end;
if (rpo='V') then begin j:=0;writeln('sigles des ',lico,'s');
  while not ((rpo='N') or (j=carqj)) do begin j:=j+1;
    for a:=1 to 3 do read(fiu1,tam3[a]);sgu:=sigler(psgl^);
    for a:=1 to 3 do read(fiu2,tam3[a]);
    if (sgu=sigler(psgl^)) then begin
      write(sgu:5);if (j mod 16 =0) then writeln;
      for a:=1 to 3 do write(fig,tam3[a]) end
    else begin writeln;
      writeln('ERREUR sigles différents ',lico,j:5,': ',sgu,' et ',sigler(psgl^));
      rpo:='N' end end end;
if (rpo='V') then begin
  writeln;writeln('lignes du tableau 1');i:=0;
  while not(eof(fiu1) or (i=carli)) do begin i:=i+1;
    for a:=1 to 3 do read(fiu1,tam3[a]);
    write(sigler(psgl^):5);if (i mod 16 =0) then writeln;
    for a:=1 to 3 do write(fig,tam3[a]);
    for a:=1 to 2*carqj do begin read(fiu1,ilu);write(fig,ilu) end end;
  writeln;writeln('lignes du tableau 2');i:=0;
  while not(eof(fiu2) or (i=car2i)) do begin i:=i+1;
    for a:=1 to 3 do read(fiu2,tam3[a]);
    write(sigler(psgl^):5);if (i mod 16 =0) then writeln;
    for a:=1 to 3 do write(fig,tam3[a]);
    for a:=1 to 2*carqj do begin read(fiu2,ilu);write(fig,ilu) end end;
  end;
if (rpc='O') then begin close(fiu1);close(fiu2);close(fig);
  if (rpo='N') then détruire(nomf);end;
writeln;write('faut-il combiner d'autres tableaux O ou N ');
readln(rpu) end;end.

```


On notera que si l'on rencontre des sigles discordants (discordance pour les colonnes, dans le cas d'une superposition; ou pour les lignes, si cela est demandé, pour une juxtaposition), le caractère: rpo, est mis à 'N', et le fichier ouvert en écriture est finalement détruit.

3 Le programme 'wITxJ' de présentation d'un tableau ternaire

```

program wITxJ;
uses memtypes, quickdraw, osintf, toolintf, sane, uver5, ulire5, ucrire5;
var i, j, ii, jj, tt, carti, cari, cart, cartj, carj, er2: integer;
sgli, sg2j, sg3i, sg4j: pzqi; klji: kjl; paux: ptr;
nomba, nomf, titre, sig: string; rpc, rpq, rpf, rpm: char;
tlk, t2k: tk; sli, s2j, s3i, s4j: ptr; ft: text;
begin benzecri; rpc:='O';
while (rpc='O') do begin carti:=0;
  writeln('ce programme transforme ITxJ en TIXJ (ou réciproquement)');
  writeln('ou, encore, IxTJ en IxJT (ou réciproquement)');
  writeln('ou, encore, TIXJ en TJxI ');
  writeln('ou crée, par cumul de lig, un tab de marge binaire d'un tab ternaire');
  litab(s2j, @tlk, @nomba, @titre, rpc, carti, cartj, sli);
  if (rpc='O') then begin
    writeln('cari =', carti:5, ' ; carj =', cartj:5); rpx:='*';
    while not (rpq in ['C', 'D', 'L']) do begin
      writeln('faut-il traiter les blocs de lignes(L) ou de colonnes(C)');
      write('ou passer de TIXJ à TJxI (D) '); readln(rpq); end;
  end;
end;

```

Le programme, consiste en une boucle générale: while (rpc='O') do...; laquelle commence par la procédure 'litab' (IA§2) dont l'objet est de choisir et d'entrer en mémoire le tableau à transformer. Si le choix de l'utilisateur s'est fixé sur un tableau accessible, on entre dans la boucle: if (rpc='O') then..., laquelle comprend tous les traitements. L'utilisateur choisit d'abord entre trois variantes: ['C', 'D', 'L'], dont la fonction respective apparaîtra dans la suite. Avant tout traitement, le programme demande de préciser le format des données; et s'assure que celui-ci se prête à la transformation choisie.

```

if (rpq='L') then begin
  writeln('cari x carT = ', carti:5); er2:=0; rpc:='N';
  while not ((rpc='O') or (er2=6)) do begin
    writeln('ce programme transforme ITxJ en TIXJ (ou réciproquement)');
    writeln('il peut, également, créer le tableau de marge IxJ ou TxJ');
    writeln('on suppose que CHACUN des blocs consécutifs de lignes,');
    writeln('tel qu''il se présente au DÉPART, a pour cardinal carB');
    write('Le nombre carB est '); readln(cart);
    if (cart<1) then cart:=2; if (carti<cart) then cart:=carti div 2;
    if not (carti mod cart=0) then begin rpc:='N'; er2:=er2+1;
      write('ERREUR: carB, =', cart:5, ', ne divise pas ', carti:5, ' ');
      readln(rpf); end;
    if (carti mod cart=0) then begin
      write('ce nombre carB, =', cart:5, ' est il confirmé oui(O) ou non(N) ');
      readln(rpc); end; end; end;
end;

```

D'abord, pour 'L': l'ensemble des lignes est un ensemble produit; soit: $I \times T$: les lignes forment alors des blocs consécutifs de cardinal: $carB = carT$; et l'on doit vérifier que $carB$ divise le nombre total des lignes.

Il en est de même pour 'C'; à ceci près que les colonnes y prennent le rôle dévolues aux lignes pour 'L'.

```

if (rpq='C') then begin
  writeln('carJ x carT = ',cartj:5);er2:=0;rpc:='N';
  while not ((rpc='O') or (er2=6)) do begin
    writeln('ce programme transforme IxJT en IxTJ (ou réciproquement)');
    writeln('on suppose que CHACUN des blocs consécutifs de colonnes,');
    writeln('tel qu''il se présente au DÉPART, a pour cardinal carB');
    write('le nombre carB est ');readln(cart);
    if (cart<1) then cart:=2;if (cartj<cart) then cart:=cartj div 2;
    if not(cartj mod cart=0) then begin rpc:='N';er2:=er2+1;
      write('ERREUR: carB, =',cart:5,', ne divise pas ',cartj:5,' ');
      readln(rpf);end;
    if (cartj mod cart=0) then begin
      write('ce nombre carB, =',cart:5,' est il confirmé oui(O) ou non(N) ');
      readln(rpc);end;end;end;

```

Le cas 'D' est analogue; à la gestion près de l'espace. En effet, dans la version expéditive présentée ici, on procède avec un jeu de pointeurs pour lire, en mémoire centrale, non le tableau demandé, mais son transposé; et il revient à la procédure 'critab' (cf.IID§5) d'écrire sur disque le tableau demandé lui-même. C'est pourquoi est imposée la condition que: carTJ ne dépasse pas la

```

if (rpq='D') then begin
  writeln('nombre des lignes: carI x carJ = ',carti:5);er2:=0;rpc:='N';
  while not ((rpc='O') or (er2=6)) do begin rpc:='O';
    writeln('ce programme transforme TIxJ en TJxI ');
    writeln('on suppose que CHACUN des blocs consécutifs de lignes,');
    writeln('tel qu''il se présente au DÉPART, a pour cardinal carI');
    write('le nombre carI est ');readln(cari);
    if (cari<1) then cari:=2;if (carti<cari) then cari:=carti div 2;
    if (jmax<cari) then cari:=jmax;cart:=carti div cari;
    if not(carti mod cari=0) then begin rpc:='N';
      write('ERREUR: carI, =',cari:5,', ne divise pas ',carti:5,' ');
      readln(rpf);end;
    if (rpc='O') and (jmax<cart*cartj) then begin rpc:='N';
      write('IMPOSSIBLE: carTJ, =',cart*cartj:5,', dépasse ',jmax:5,' ');
      readln(rpf);end;
    if (rpc='N') then er2:=er2+1;
    if (rpc='O') then begin
      write('ce nombre carI, =',cari:5,' est il confirmé oui(O) ou non(N) ');
      readln(rpc);end;end;end;

```

valeur: jmax, fixée arbitrairement à 300, pour un tableau de pointeurs: tk (cf.IA§1.1). On a, simultanément, imposé que soit inférieur à: jmax, le nombre: cari, des colonnes du tableau demandé.

```

if (rpc='N') then begin rpq:='*';
  for j:=0 to cartj do dispose(t1k[j]);
  dispose(s1i);dispose(s2j);end;

```

Si le tableau de base choisi ne se prête pas aux transformations demandées, la boucle: if (rpc='O') then..., tourne court sur une libération de la mémoire centrale; et toutes les voies sont interdites par l'instruction: rpq:='*'. Sinon on passe à l'écriture de tableau, en distinguant selon les voies: ['L','C','D'].

Dans le cas: 'L', est prévu le calcul de tableaux de marge rectangulaires. Ce calcul est proposé de telle sorte que l'utilisateur ne soit pas induit en erreur par un notation, autre que la sienne des trois ensembles {I, J, T} en correspondance.

```

if (rpg='L') then begin
  writeln('Demande de calcul des tableaux de marge IxJ ou TxJ');
  write('faut-il cumuler les blocs de lig tels qu'ils sont donnés(D) ou non(N) ');
  readln(rpm);
  if not (rpm='N') then rpm:='D';
  write('faut-il cumuler les blocs de lig une fois regroupés(R) ou non(N) ');
  readln(rpf);
  if not (rpf='N') then rpf:='R';
  cari:=carti div cart;
  for j:=0 to cartj do klji[j]:=pti(tlk[j]);
  sgli:=pzgi(sli);sg2j:=pzgi(s2j);
  if (carti<5400) then begin
    s3i:=newptr(6*carti);sg3i:=pzgi(s3i);end
  else begin new(sg3i);s3i:=ptr(sg3i);end;

```

On notera qu'afin de permuter les sigles des lignes, est créé un nouveau tableau, désigné par le pointeur typé: sg3i (cf. IA§1.2).

```

for j:=1 to cartj do begin
  for i:=1 to carti do klji[0]^i:=klji[j]^i;
  for ii:=1 to cari do for tt:=1 to cart do
    klji[j]^(tt*cari+ii-cari):=klji[0]^(tt+(ii*cart)-cart);end;
  for ii:=1 to cari do for tt:=1 to cart do
    sg3i^(tt*cari+ii-cari):=sgli^(tt+(ii*cart)-cart);
  nomf:=concat(nomba, 'BI.w');
  critab(s3i, s2j, @tlk, @nomf, @titre, carti, cartj, 0);

```

Le regroupement, en blocs, du contenu de chaque colonne se fait, en prenant la colonne: 0 (zéro), comme espace auxiliaire.

```

for j:=1 to cartj do begin
  if (rpm='D') then for ii:=1 to cari do begin klji[0]^ii:=0;
    for tt:=1 to cart do
      klji[0]^ii:=klji[0]^ii+klji[j]^ii+(cari*tt)-cari);end;
  if (rpf='R') then for tt:=1 to cart do begin klji[0]^cari+tt:=0;
    for ii:=1 to cari do
      klji[0]^cari+tt:=klji[0]^cari+tt+klji[j]^ii+(cari*tt)-cari);end;
  if (rpm='D') or (rpf='R') then
    for i:=1 to (cari+cart) do klji[j]^i:=klji[0]^i);end;
  if (rpm='D') then begin
    nomf:=concat(nomba, 'CI.w');
    critab(s3i, s2j, @tlk, @nomf, @titre, cari, cartj, 0);end;
  if (rpf='R') then begin
    for j:=1 to cartj do t2k[j]:=@klji[j]^(1+cari);
    nomf:=concat(nomba, 'CBI.w');
    critab(sli, s2j, @t2k, @nomf, @titre, cart, cartj, 0);end;

```

Une fois écrit le tableau permuté, on procède, colonne après colonne, aux cumuls éventuels, pour les marges. Ici encore, on utilise la colonne zéro; de plus, dans les colonnes [1..cartj], les tableaux de marge prennent place l'un en-dessous de l'autre; et ils sont lus par 'critab' grâce à des pointeurs appropriés.

Quant aux sigles, chaque ligne d'un tableau de marge reçoit celui de la première ligne du bloc qu'on a cumulé pour la créer.

```

for j:=0 to cartj do dispose(tlk[j]);
dispose(sli);dispose(s2j);dispose(s3i);end;

```

Le traitement de: rpg='L', se termine en libérant la mémoire.

```

if (rpq='C') then begin
carj:=cartj div cart;
for j:=0 to cartj do klji[j]:=pti(tlk[j]);
sgli:=pzgi(s1i);sg2j:=pzgi(s2j);
s4j:=newptr(6*cartj);sg4j:=pzgi(s4j);
for jj:=1 to carj do for tt:=1 to cart do
  t2k[(tt*carj)+jj-carj]:=tlk[tt+(jj*cart)-cart];
for jj:=1 to carj do for tt:=1 to cart do
  sg4j^[(tt*carj)+jj-carj]:=sg2j^[tt+(jj*cart)-cart];
nomf:=concat(nomba,'BJ.w');
critab(s1i,s4j,@t2k,@nomf,@titre,carti,carj,0);
for j:=0 to cartj do dispose(tlk[j]);
dispose(s1i);dispose(s2j);dispose(s4j);end;

```

Pour 'C' le traitement est tout analogue à celui de 'L'; à ceci près que n'est pas proposé le calcul de tableau de marge binaire.

```

if (rpq='D') then begin
cart:=cart1 div cari;carj:=cart*cartj;
for j:=0 to cartj do klji[j]:=pti(tlk[j]);
sgli:=pzgi(s1i);sg2j:=pzgi(s2j);
for tt:=1 to cart do for jj:=1 to cartj do
  t2k[(tt*cartj)+jj-carj]:=@klji[jj]^[1+((tt-1)*cari)];
for ii:=1 to cari do begin
  sig:=concat('i',chr(48+(ii div 100)));
  sig:=concat(sig,chr(48+((ii div 10) mod 10)),chr(48+(ii mod 10)));
  sgli^[ii]:=zigler(sig);end;
for tt:=1 to cart do begin
  sig:=concat(chr(64+((tt div 20) mod 20)),chr(64+(tt mod 20)),'xx');
  for jj:=1 to cartj do begin
    sig[3]:=chr(96+((jj div 20) mod 20));
    sig[4]:=chr(96+(jj mod 20));
    if (sig[3]='') then sig[3]='0';if (sig[4]='') then sig[4]='0';
    sg2j^[(tt*cartj)+jj-carj]:=zigler(sig);end;end;
nomf:=concat(nomba,'BIJ.w');
critab(s1i,s2j,@t2k,@nomf,@titre,cari,carj,1);
for j:=0 to cartj do dispose(tlk[j]);
dispose(s1i);dispose(s2j);end;

```

Dans le passage de: $(T \times I) \times J$, à: $(T \times J) \times I$, (cas 'D'), on note que les colonnes du tableau: $I \times (T \times J)$, (transposé du tableau à créer: $(T \times J) \times I$), ne sont autres que les segments successifs des colonnes du tableau de base: ce qui permet de les désigner par un tableau de pointeurs.

Quant aux sigles, ceux des éléments de I sont pris, tels quels, dans le 1-er bloc de l'ensemble initial des lignes: $(T \times I)$. Pour $(T \times J)$, renonçant à extraire, de $(T \times I)$ et de J, des notations vraisemblables, on a choisi de mettre successivement le rang de t et celui de j; chaque nombre étant écrit en base vicésimale, avec deux capitales pour: t; et deux minuscules pour: j.

```

if (rpc='O') then begin write('faut-il poursuivre O ou N ');
  readln(rpc);end;end;end;
readln(xpf);end.

```

Si l'on a abouti à une création de tableau, l'utilisateur peut demander de poursuivre en traitant un autre tableau de base. Ainsi se termine la boucle de calcul: if (rpc='O')..., la boucle générale: while (rpc='O')...; et le programme lui-même.

N.B. Avec le présent Cahier, le dernier du Volume XXII, s'achève la publication des sources des principaux programmes d'analyse factorielle, de Classification Ascendante Hiérarchique, d'examen de codage et élaboration des données en tableaux. Manquent toutefois, notamment, les programmes d'analyse de séries chronologiques.

À la cartographie qui devrait constituer une partie (IV), il n'a été fait que de brèves allusions à propos du programme: 'zrang' (cf. II A, §§1.1.3.2 et 2.3).

Quant à l'élaboration et à l'analyse des textes, le présent cahier présente, en détail, le programme de tri: 'trigalat§' (V A). Du reste de la partie (V), l'article [SOURCES V A] offre un aperçu.