# 企業データを統合するための名寄せ処理技法

古隅 弘樹
兵庫県立大学経済学部

# Data Matching Techniques on Merging Company Information

Hiroki FURUZUMI
School of Economics, University of Hyogo

## 要旨

名簿情報の統合処理に必要な「名寄せ」を行うための SAS プログラミング技法について報告する。例えば、企業情報を収録した複数のデータベースをその名簿情報（企業名称、所在地、電話番号、等）を用いて統合するためには、各名簿データの企業情報を突合して重複登録されたレコードとユニークなものに分離する作業が必要になる。

文字情報を照合するための事前処理として、相互のデータセットにおけるデータ構造やフォーマットの違いを考慮するデータクレンジングの必要があるのは当然であるが、もっとも労力を要するのがデータにおける表記揺れの対処である。例えば、半角と全角、ひらがなとカタカナ、漢字の異体字表記などである。他にも、企業名称では「(株)ＡＢＣ」と「株式会社エー・ビー・シー」、住所の番地表記では「一丁目二番三号」と「１－２－３」といった表記揺れが挙げられる。人の目による判別では難しい問題ではないが、プログラミングによる文字列の照合では、たった一文字の違いが照合結果を左右してしまう。このような表記揺れに対処するためのプログラミング技法を紹介する。また、企業名称に事業所名称等が含まれる場合や、所在地における建物名称の有無など、完全一致の照合によるレコードの接合（データステップにおける merge および by ステートメントによる処理）では扱えない、部分一致や先頭一致による照合・接合方法や、照合用のキー文字列の作成方法についても述べる。

キーワード：　文字列照合技法、名寄せ、DBCS 関数、正規表現、PRX 関数

## 1. はじめに

複数のデータテーブルや異なるデータベースで管理されている顧客データや業務データを統合する際には、相互に重複して格納されている冗長な情報を抽出して排除する必要がある。これらの統合によって新たな情報の獲得を目的とする場合においても同様であるが、相互のレコードを何らかの条件で接合する必要がある。それぞれのデータに共通の識別番号が付与してあるか、相互を接合するための識別番号の対応表（いわゆるリンクコンスタント）が提供される場合の対処は容易であるが、そうでない場合にはそれぞれのデータ項目を照合して接合する必要があり、次の２つの接合方法がある。１つは特定の属性項目（例えば世帯データにおける居住地域、世帯構成や類型、家計の収支、等）の一致状況から同一とみなして接合する手法であり、

これを統計的照合という。もう１つは名簿情報の属性（名称、所在地、電話番号、等）が一致するものを接合する手法であり、これを完全照合という。本論文で取り扱うのは後者の完全照合の手法であり、以下では企業名簿情報を用いたいわゆる名寄せ処理を行うための SAS プログラミングの技法について報告する。

## 2. 事前処理

相互のデータセットを照合して接合する作業を始める前に、相互のデータセットにおけるデータ構造やフォーマットの違いを考慮するデータクレンジングの必要がある。例えば、数値や日付の書式、郵便番号や電話番号を比較する場合の括弧やハイフンの有無、住所データの分割状況、などを一方あるいは照合の都合に合わせて統一する必要がある。

また、文字列照合を行う場合、比較する項目において使用されている文字の分布を調査するのが望ましい（プログラム例１を参照）。それによって、想定外の記号や全角・半角の違いなど、当該項目における不適切な入力データや事前処理の不具合が明らかになることが多い。その上で、以下のような事柄について考慮する必要がある。

- 空白文字、区切り記号などの除去の必要性
- 長音記号、句読点などの記号表記
- 文字の半角・全角
- 英字の大文字・小文字
- カタカナ・ひらがな
- 漢字表記における異体字・機種依存文字・外字
- 促音および拗音の大書き・小書き、等

SAS で全角文字列データを処理するためには、DBCS 関数を利用する必要があるなど、シングルバイト（半角）文字列を処理する場合に比べて扱いが複雑になるが、以下のような関数をうまく使うことである程度は対応可能である。

| |
|---|
| kcompress 関数<br>空白文字を除去する。文字を指定することも可能。<br>使用例<br>変数 text に出現するカギ括弧や引用符を除去する<br>text = kcompress(text, ″「」()【】[]“” ‘’ ″); |
| ktranslate 関数<br>文字単位の変換を行う。１文字を別の１文字に置き換えたい場合に使用する。<br>使用例<br>変数 text に出現する小書きされたカタカナを大書きに変換する<br>text = ktranslate(text, ″アイウエオヤユヨツケ″, ″ァィゥェォャュョッヶ″);<br>変数 text に出現する漢字表記を統一する<br>text = ktranslate(text, ″辺辺弁弁弁二二二二″, ″邊邉辨瓣瓣辯弍弐貳貳″); |
| tranwrd 関数<br>文字列単位の置換を行う。<br>使用例<br>変数 text に出現する文字列「株式会社」を「(株)」に置換する<br>text = tranwrd(text, ″株式会社″, ″（株）″); |

これらの関数を組み合わせれば、半角文字と全角文字の変換も可能である。（プログラム例２を参照）

## 3. 企業名称の照合

企業名称を照合する際の問題は、商号に含まれる「～株式会社」のような会社形態の有無や、「～（株）」のような省略表記の使用による表記揺れが挙げられる。単純にこれらを除去してしまうと、株式会社と合名会社で異なるのに同名と判定されたり、これらが名称の前後どちらに付くかで企業が異なるため判別できないといった弊害が出てしまう。また、平成 18 年に改正された会社法の施行に伴う有限会社から株式会社への移行期をまたぐような情報を照合する場合には、企業名称における会社形態と前後の位置を名称と分けて照合するといった工夫が必要となる。会社形態の表記統一や除去については tranwrd 関数で対処可能である。

商業登記における企業名称では拗音や促音の小書き表記を大書きするケースが多く見られる。また、平成 14 年の商業登記規則等の改正により、以前は使えなかった英数字や記号（「＆」、「ー」、「・」、等）が使えるようになった。「＆」と「アンド」、「ＡＢＣ」と「エービーシー」のように、英字をカタカナに統一するといった対処はある程度はできるが、「カンパニー」と「ＣＯＭＰＡＮＹ」のようにカタカナと英字の表記揺れについては、「とうふ」と「豆腐」のようにひらがな（またはカタカナ）と漢字の表記揺れ同様で、これらの組合せをいずれかの表記に統一する定義を逐一 tranwrd 関数で記述する必要があり、対応が難しいものもある。

## 4. 正規表現と文字列照合

文字列処理に長けたスクリプト言語（Perl、Ruby など）では以前から正規表現によるパターンマッチングを行うことができるが、SAS でもサポートされるようになった。ここでは Perl の正規表現によるマッチング機能の一部が利用できる PRX 関数を紹介するが、シングルバイト文字を前提としているため、全角文字列の処理には工夫が必要である。これは SAS の PRX 関数の実装が Shift-JIS エンコーディングを正式にサポートしていない古いバージョン（Perl 5.6.1）をベースに開発されていることが原因と考えられる。

例えば、数字が１つ以上連続する文字列を意味する正規表現パターンは、半角文字列であれば /¥d+/ または /[0-9]+/ と記述可能であるが、全角の場合に /[０-９]+/ と記述するとエラーにはならないが意図する動作は望めない。prxdebug 関数でデバッグモードをオンにして照合過程をログ出力してみるとよく分かるが、これは正規表現パターンがシングルバイトで解釈され、照合対象の文字列についても同様にシングルバイトで照合されているのが原因である。上述の表現はシングルバイトで記述すると /[¥x82¥x4F-¥x82¥x58]+/ と同等であり、結果的に全角文字の２バイトのうちの１バイトが不適切な処理によって破壊されたり、それによって文字化けが発生するといった事態を引き起こしかねない。シングルバイトで処理されていることを考慮すれば、先程の全角数字列を検出するパターンは、 /(０|１|２|３|４|５|６|７|８|９)+/ あるいは /(¥x82[¥x4F-¥x58])+/ と記述すれば意図した照合動作を期待できる。

シングルバイトで処理されることに起因する弊害は他にもある。それは隠れた「¥」（¥x5C）が引き起こすやっかいな問題である。Perl では特殊な意味を持つ記号であり、「数字 (¥d)」や「水平タブ (¥t)」、「改行 (¥n)」などエスケープシーケンスで使用されている。「¥¥」と重ねることで特殊な意味を打ち消すことができ、「¥」という文字自体を表すことができる。Windows 用の日本語版 SAS では日本語文字コードはシフト JIS で処理しており、全角文字の中には 2 バイト目に半角の「¥」と同じ¥x5C を持つものがいくつかある。

これらの文字を正規表現の記述の中でそのまま用いると、２バイト目（¥x5C）が「¥」と解釈され、正規表現パターンがコンパイルできないといった弊害が出る。対処方法は単純で、これらの文字の直後に「¥」を入れることでその特殊な意味を打ち消すことができ、適切に処理してもらえるようになる。

いくつかの PRX 関数についての使用例を以下に挙げる。

prxmatch 関数
　正規表現パターンにマッチした位置を返す。
使用例
　変数 text に出現する全角漢数字表現の開始位置を取得する。マッチしなければ 0。

```
position = prxmatch("/(〇|一|二|三|四|五|六|七|八|九|十¥)+/", text);
```

prxchange 関数
　置換処理を行う
使用例
　変数 text に出現する全てのカギ括弧の開閉を二重カギ括弧の開閉に置き換える。

```
text = prxchange("s/「(.+)」/『$1』/", -1, text);
```

prxparse 関数
　正規表現パターンをコンパイルする。同じパターンを繰り返し使う場合に便利。
prxnext 関数
　順次検索を行う。
使用例
　変数 text に出現する全角算用数字表現を検索開始位置(start)から最後(-1)まで探索し、その出現位置を pos、マッチした文字列の長さを len に格納する。

```
regid = prxparse("/(０|１|２|３|４|５|６|７|８|９)+/");
call prxnext(regid, start, -1, text, pos, len);
```

prxdebug 関数
　デバッグモードを切り替える。オンにしておけば照合過程がログ出力される。
使用例

```
call prxdebug(1); /* on */        call prxdebug(0); /* off */
```

［参考表］下位バイトが¥x5C の全角文字（主要なものを抜粋）

| | | | | | | |
|---|---|---|---|---|---|---|
| 【815C】― | 【835C】ソ | 【895C】噂 | 【8B5C】欺 | 【8C5C】圭 | 【8D5C】構 | 【8E5C】蚕 |
| 【8F5C】十 | 【905C】申 | 【915C】曾 | 【925C】箪 | 【935C】貼 | 【945C】能 | 【955C】表 |
| 【965C】暴 | 【975C】予 | 【985C】禄 | 【995C】兔 | 【9C5C】彌 | 【9D5C】拿 | 【9E5C】杤 |

## 5. 所在地情報の照合

所在地の情報は照合を始める前の事前処理の段階から考慮しなければならないことが多い。例えば、複数の項目（都道府県、市区町村、町丁字番地号、建物名、等）に分割して入力されている場合に、項目区分（２区分、３区分）が異なることや、分割区分の揺らぎ（地方自治体の市区町村区分と入力項目区分のあいまいな分割)、政令指定都市など一部の都市で都道府県名の入力が省略されていないか、などである。

所在地の情報管理においては、正規化（コード化）することで住所の表記揺れによる弊害をなくす工夫がなされているが、このような方策がとられていない場合や、データによって異なる正規化手法が使われている場合には相互の住所に関する文字列情報を照合する必要がある。

所在地の表現に限らないが、さまざまな情報の記述において数字表現が出現する。例えば、「２３０４」という数字は「二千三百四」、「２千３百４」、「二三〇四」のような表記パターンがある。このような表記が所在地表記、とりわけ番地表記において多用されることから、漢数字と算用数字（アラビア数字）の表記の混在を解消して表記を統一させなければ文字列照合は成功しない。

漢字表記の問題（例えば、万と萬、十と拾、など）や、半角と全角の問題については、事前処理で述べたような表記統一の処置を施すことで対処できる。その上で、全角算用数字に表記統一を行うプログラムを作成してみたので紹介する（プログラム例３を参照)。なお、日本語の漢数字表記（とりわけ所在地や番地の数

字表記）では、４桁ごとに位（万、億、兆、･･･）があり、それぞれの位のなかで倍数（十、百、千）が繰り返し使われることを前提としてプログラムを作成しているため、欧米式に３桁毎に千単位で区切る表記（例：１２千３４５、等）には対応していない。

所在地の照合における問題は数字表現だけではない。例えば、「１－２－３」という番地表記は「１丁目２番３号」や「１丁目２番地の３」など、種々のパターンで表記される。このような番地表記についても統一させる必要がある（プログラム例４を参照）。

表記を統一する方法は情報を可能な限り損なわずに照合の成功率を高める方法であるが、すべての表記パターンについて適切な変換処理を行うことは高度なパターンマッチング技術が要求される。そこで、多少の情報の正確さを損なうことになるが、重要な部分をうまく残してマッチングの確率を高めるための照合用キーを作成するアプローチもある。例えば、ひらがな・カタカナはカタカナに統一し、漢数字・算用数字・ローマ数字（全角）は算用数字に統一、住所の番地表記も数字だけ（１－２－３は１２３）にするなどである。このように作成した照合用キーが複数の項目（例えば企業名称と所在地）で一致する場合には、非常に高い確率で両者は同一企業であると言える。

また、マンションや建物名称が番地情報の中に含まれる場合とそうでない場合について照合して一致するかを判定するのは、建物名称が別掲されている場合を除き、そのままでは困難である。他にも大字・小字の有無など、パターンマッチングでは対応できないものが多くある。そこで、郵便物のバーコードのように郵便番号７桁の情報と番地から抽出した数値情報を併せて照合する手法も考えられる。参考までに全角英数文字だけを抽出して出現順に結合するプログラムを紹介する（プログラム例５を参照）。

## 6. 接合処理

2 つのデータセットをレコード（オブザベーション）単位で接合する手順としては、最も厳しい照合条件から徐々に条件を緩和しながら照合し、接合できたレコードは以降の照合作業において除外するようにすれば、以降の照合件数を少なくして照合効率を上げることができ、レコード間の多重（1 対多、多対 1、多対多）接続も抑えることができる。照合用データセットの準備としては、クレンジングを施したオリジナルの情報に近いものから、可能な限り表記統一を施したもの、ある程度の精度を落とした照合用キー、と段階的な照合用データを企業名称や所在地などの各項目で作成しておくと、きめ細かく照合作業をコントロールできる。

照合条件を指定してレコードをマッチングさせる作業において、data ステップの merge 文を使用する、いわゆるマッチマージを行う場合は、by 文で指定した変数がすべて一致する条件は指定できるが、先頭一致や部分一致、and/or 条件を含めて柔軟な条件の指定が容易なのは SQL 文であり、ここでは proc sql によるプログラミングを紹介する（プログラム例６を参照）。

## 7. さいごに

日本語版 Windows SAS では、日本語文字列を Shift-JIS エンコーディングで処理しているが、マルチバイトエンコーディング特有の処理が必要である。インターネットの普及に伴う世界各地の言語処理の必要性と計算機の処理能力向上によって、Unicode に代表される多言語対応文字コードが現在の主流となりつつある。また、Shift-JIS は従来の JIS 第 1・第 2 水準漢字（JIS X 0208）をベースとしているが、1998 年に JIS 補助漢字（JIS X 0212）の策定、2000 年には JIS 第 3・第 4 水準（JIS X 0213）が策定され、その 2004 年改定版を Windows Vista がサポートし、2010 年に内閣告示された常用漢字表の改定に対応するための JIS 規格改定が必至となっ

ている。Shift-JIS の 2 バイト空間で新たな文字需要をカバーするのは限界に達しており、システムにおける日本語エンコーディング環境は見直すべき時期がきている。SAS における正規表現によるパターンマッチングでは Perl5.6.1 をベースとして開発された PRX 関数を用いたが、Perl5.8 以降のバージョンで関数が実装されたならば、UTF-8 をベースとしたマルチバイトエンコーディングの文字列処理に対応することになる。UTF-8 と Shift-JIS のコードマッピングの問題があるにせよ、SAS における将来の実装を期待したい。

漢数字を含む数値表現を算用数字に統一するプログラムを紹介したが、他の数値表現としてはローマ数字があり、全角ローマ数字（機種依存文字）、英字、大文字・小文字、算用数字による代用、等の表記揺れ（例：Ⅸ、ix、ＩＸ、ｉｘ、9）がある。全角ローマ数字の場合を除き、他の英字表記との区別が困難なため統一変換処理は難しく、変換処理の対象外とした。

所在地情報の照合を阻害する他の要因としては、市区町村合併や政令指定都市への移行により、市区町村番号（JIS X 0402）の変更、住所表記（新たな町域、区割り）の変更、場合によっては郵便番号の変更を伴う新旧住所表記の混在や、登記情報の番地と郵便住所の番地は一致しないことが多いなど、対応情報の入手とそれに基づく変換処理が必要となる場合もある。

データの接合作業では、照合条件の設定および順序について精査する必要がある。各照合レベルにおける接続件数やその内訳（1 対 1 接続か否か）をチェックし、必要に応じて条件設定を見直すべきである。また、緩和した条件下での接続や、接続したが 1 対 1 でない組合せについては後でチェックリストを作成して目視確認をする必要がある。また、目視確認によって得られた知見を照合プログラムにフィードバックするなどのアフターケアが必要である。

## 参考文献


[1] 周防節雄・古隅弘樹・宮内環 共著「法人企業統計調査と事業所・企業統計調査の統合データによる企業データベース:1983～2005 年」（特集 ミクロ経済データによる統計解析－日本の法人企業の構造）、統計数理 57(2)、277-303、2009

[2] 法務省 HP「商号にローマ字等を用いることについて」、http://www.moj.go.jp/MINJI/minji44.html

[3] 法務省 HP「会社法の施行に伴う会社登記についてのＱ＆Ａ」、http://www.moj.go.jp/MINJI/minji92.html

[4] 日本郵便 HP「郵便番号・バーコードマニュアル」、http://www.post.japanpost.jp/zipcode/zipmanual/index.html

[5] Microsoft HP, ‘Windows Codepage: 932 (Japanese Shift-JIS)’, http://msdn.microsoft.com/ja-jp/goglobal/cc305152

[6] Ken Lunde 著、小松章・逆井克己訳、『CJKV 日中韓越情報処理』、オライリージャパン、2002 年

[7] ‘Functions and CALL Routines: Perl Artistic License Compliance’, SAS 9.2 Documentation, SAS(R) 9.2 Language Reference: Dictionary

[8] 「SAS9 の新機能と移行について　－Perl 正規表現を利用する関数」、SAS Technical News Autumn 2004

[9] 斎藤靖・小山裕司 他共著、Computer Today ライブラリ-34『新 Perl の国へようこそ－Perl5 対応版』、サイエンス社、1996 年

[10] 「特集 SAS からはじめる SQL」、SAS Technical News Spring 2011

[11] 深沢千尋著『文字コード「超」研究』、ラトルズ、2003 年

[12] 文字研究会編『新常用漢字表の文字論』、勉誠出版、2009 年

[13] Microsoft HP「Windows Vista:JIS X 0213:2004 対応と新日本語フォント‘メイリオ’について」
http://www.microsoft.com/ja-jp/windows/products/windowsvista/jp_font/default.aspx

# 付録

## プログラム例１．　文字の出現頻度を調査する

①１文字ずつ切り出してデータセットへ出力し、proc freq で出現頻度を集計する方法。
（入力）work.inputdata の変数 text
（出力）work.charlog

```
data charlog;
  set inputdata;
  length _obs _col 4 _key $2 _code $4;
  keep _:;

  if text=' ' then return;
  _obs=_N_;
  len=klength(text);

  do i=1 to len;
    _col=i;
    _key=ksubstr(text, i, 1);
    _code=put(_key, $hex.);
    if substr(_code,1,1) not in ('8','9','E','F')
      then _code=substr(_code,1,2); /*半角文字*/
    output;
  end;
run;


proc freq data=charlog;
  tables _code*_key /list missing;
run;
```

②切り出した１文字をキーとして出現頻度を hash オブジェクトに登録し、結果をデータセットに出力する方法。
（入力）work.inputdata の変数 text
（出力）work.charfreq

```
data _null_;
  set inputdata  end=eod;
  length _code $4 _key $2 _freq 8;

  if _N_=1 then do; /* hash オブジェクトの定義 */
    declare hash h(hashexp: 16, ordered: 'yes');
    rc=h.defineKey('_key');
    rc=h.defineData('_code', '_key', '_freq');
    rc=h.defineDone();
    call missing(_key, _freq);
  end;

  if text=' ' then return;
  len=klength(text);
  do i=1 to len;
    _key=ksubstr(text, i, 1);
    _code=put(_key, $hex.);
    if substr(_code,1,1) not in ('8','9','E','F')
      then _code=substr(_code,1,2); /*半角文字*/
    rc=h.find();
    if rc=0 then do; /*既出文字の場合*/
      _freq+1; rc=h.replace();
    end;
    else do; /*初出文字の場合*/
      _freq=1; rc=h.add();
    end;
  end;
  if eod then rc=h.output(dataset: 'charfreq'); /*出力*/
run;

proc print data=charfreq; run;
```

（結果出力例）

企業名簿データから所在地の情報を入力として得られた文字の出現頻度（一部抜粋）

| _code | _key | _freq | _code | _key | _freq | _code | _key | _freq | _code | _key | _freq |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 8140 | □ | 76,313 | 82A0 | あ | 98 | 889F | 亜 | 14 | E056 | 澤 | 13 |
| 8144 | . | 44 | 82A2 | い | 561 | 88A1 | 娃 | 1 | E05F | 濱 | 1 |
| 8145 | ・ | 98 | 82A4 | う | 78 | 88A2 | 阿 | 265 | E06B | 猪 | 7 |
| 8148 | ？ | 1 | 82A6 | え | 12 | 88A4 | 愛 | 175 | E07C | 烟 | 1 |
| 814A | ゛ | 1 | 82A8 | お | 45 | 88A6 | 姶 | 11 | E0B7 | 犢 | 1 |
| 8158 | 々 | 245 | 82A9 | か | 137 | 88A7 | 逢 | 8 | E161 | 畉 | 2 |
| 815B | ― | 1,632 | 82AA | が | 310 | 88A8 | 葵 | 130 | E163 | 當 | 1 |
| 815C | ― | 43 | 82AB | き | 250 | 88A9 | 茜 | 18 | E18E | 瘤 | 2 |
| 815D | - | 12 | 82AC | ぎ | 30 | 88AB | 悪 | 1 | E1A1 | 癸 | 1 |
| 815E | ／ | 9 | 82AD | く | 151 | 88AE | 旭 | 325 | E1C1 | 眞 | 2 |
| 8169 | （ | 313 | 82AF | け | 16 | 88AF | 葦 | 3 | E1E8 | 礪 | 3 |
| 816A | ） | 312 | 82B1 | こ | 18 | 88B0 | 芦 | 60 | E24B | 祠 | 1 |
| 8175 | 「 | 1 | 82B2 | ご | 1 | 88B2 | 梓 | 7 | E250 | 祓 | 1 |
| 8176 | 」 | 1 | 82B3 | さ | 434 | 88B8 | 虻 | 15 | E27D | 竈 | 18 |
| 817C | ー | 57,928 | 82B4 | ざ | 6 | 88BB | 綾 | 80 | E294 | 笏 | 1 |
| 8190 | $ | 1 | 82B5 | し | 37 | 88BC | 鮎 | 3 | E2A2 | 箏 | 4 |
| 8194 | ＃ | 22 | 82B6 | じ | 54 | 88BE | 粟 | 46 | E2A5 | 筵 | 1 |
| 8195 | ＆ | 13 | 82B7 | す | 28 | 88C0 | 安 | 627 | E2AB | 筬 | 1 |
| 824F | ０ | 13,015 | 82B8 | ず | 12 | 88C1 | 庵 | 15 | E2C0 | 簔 | 1 |
| 8250 | １ | 63,971 | 82B9 | せ | 15 | 88C6 | 鞍 | 22 | E2C4 | 籠 | 26 |
| 8251 | ２ | 38,736 | 82BA | ぜ | 1 | 88C7 | 杏 | 1 | E2CA | 簗 | 9 |
| （・・・略・・・） | | | （・・・略・・・） | | | （・・・略・・・） | | | （・・・略・・・） | | |

## プログラム例２．半角文字と全角文字を変換するマクロプログラム

半角文字を全角文字に変換する

```
%macro jisx0201to0208(Arg);
  ktranslate(
    tranwrd( tranwrd( tranwrd(
    tranwrd( tranwrd( tranwrd(
    tranwrd( tranwrd( tranwrd(
    tranwrd( tranwrd( tranwrd(
    tranwrd( tranwrd( tranwrd(
    tranwrd( tranwrd( tranwrd(
    tranwrd( tranwrd( tranwrd(
    tranwrd( tranwrd( tranwrd(
    tranwrd( tranwrd(
  &Arg,
    'ｶﾞ','ガ'), 'ｷﾞ','ギ'), 'ｸﾞ','グ'),
    'ｹﾞ','ゲ'), 'ｺﾞ','ゴ'), 'ｻﾞ','ザ'),
    'ｼﾞ','ジ'), 'ｽﾞ','ズ'), 'ｾﾞ','ゼ'),
    'ｿﾞ','ゾ'), 'ﾀﾞ','ダ'), 'ﾁﾞ','ヂ'),
    'ﾂﾞ','ヅ'), 'ﾃﾞ','デ'), 'ﾄﾞ','ド'),
    'ﾊﾞ','バ'), 'ﾋﾞ','ビ'), 'ﾌﾞ','ブ'),
    'ﾍﾞ','ベ'), 'ﾎﾞ','ボ'), 'ﾊﾟ','パ'),
    'ﾋﾟ','ピ'), 'ﾌﾟ','プ'), 'ﾍﾟ','ペ'),
    'ﾎﾟ','ポ'), 'ｳﾞ','ヴ'),
    '□！”＃＄％＆’（）＊＋，－．／'||
    '０１２３４５６７８９：；＜＝＞？'||
    '＠ＡＢＣＤＥＦＧＨＩＪＫＬＭＮＯ'||
    'ＰＱＲＳＴＵＶＷＸＹＺ［￥］＾＿'||
    '｀ａｂｃｄｅｆｇｈｉｊｋｌｍｎｏ'||
    'ｐｑｒｓｔｕｖｗｘｙｚ｛｜｝〜'||
    '。「」、・ヲァィゥェォャュョッ'||
    'ーアイウエオカキクケコサシスセソ'||
    'タチツテトナニヌネノハヒフヘホマ'||
    'ミムメモヤユヨラリルレロワンﾞﾟ',
    ' !"#$%&''()*+,-./'||
    '0123456789:;<=>?'||
    '@ABCDEFGHIJKLMNO'||
    'PQRSTUVWXYZ[¥]^_'||
    '`abcdefghijklmno'||
    'pqrstuvwxyz{|}~'||
    '。「」、・ヲァィゥェォャュョッ'||
    'ｰｱｲｳｴｵｶｷｸｹｺｻｼｽｾｿ'||
    'ﾀﾁﾂﾃﾄﾅﾆﾇﾈﾉﾊﾋﾌﾍﾎﾏ'||
    'ﾐﾑﾒﾔﾕﾖﾗﾘﾙﾚﾛﾜﾝﾞﾟ')
%mend jisx0201to0208;
```

全角文字（かな、英数字、記号）を半角文字に変換する

```
%macro jisx0208to0201(Arg);
  tranwrd( tranwrd( tranwrd( tranwrd( tranwrd(
  tranwrd( tranwrd( tranwrd( tranwrd( tranwrd(
  tranwrd( tranwrd( tranwrd( tranwrd( tranwrd(
  tranwrd( tranwrd( tranwrd( tranwrd( tranwrd(
  tranwrd( tranwrd( tranwrd( tranwrd( tranwrd(
  tranwrd( tranwrd( tranwrd( tranwrd( tranwrd(
  tranwrd( tranwrd( tranwrd( tranwrd( tranwrd(
  tranwrd( tranwrd( tranwrd( tranwrd( tranwrd(
  tranwrd( tranwrd( tranwrd(
  ktranslate(
    ktranslate(&Arg,
        ' !"#$%&''()*+,-./'||
        '0123456789:;<=>?'||
        '@ABCDEFGHIJKLMNO'||
        'PQRSTUVWXYZ[¥]^_'||
        '`abcdefghijklmno'||
        'pqrstuvwxyz{|}~'||
        '。「」、・ヲァィゥェォャュョッ'||
        'ーアイウエオカキクケコサシスセソ'||
        'タチツテトナニヌネノハヒフヘホマ'||
        'ミムメモヤユヨラリルレロワンﾞﾟ',
        '□！”＃＄％＆’（）＊＋，－．／'||
        '０１２３４５６７８９：；＜＝＞？'||
        '＠ＡＢＣＤＥＦＧＨＩＪＫＬＭＮＯ'||
        'ＰＱＲＳＴＵＶＷＸＹＺ［￥］＾＿'||
        '｀ａｂｃｄｅｆｇｈｉｊｋｌｍｎｏ'||
        'ｐｑｒｓｔｕｖｗｘｙｚ｛｜｝〜'||
        '。「」、・ヲァィゥェォャュョッ'||
        'ーアイウエオカキクケコサシスセソ'||
        'タチツテトナニヌネノハヒフヘホマ'||
        'ミムメモヤユヨラリルレロワンﾞﾟ'
    ),
      'ヲァィゥェォャュョッ'||
      'ｱｲｳｴｵｶｷｸｹｺｻｼｽｾｿ'||
      'ﾀﾁﾂﾃﾄﾅﾆﾇﾈﾉﾊﾋﾌﾍﾎﾏ'||
      'ﾐﾑﾒﾔﾕﾖﾗﾘﾙﾚﾛﾜﾝ',
      'をぁぃぅぇぉゃゅょっ'||
      'あいうえおかきくけこさしすせそ'||
      'たちつてとなにぬねのはひふへほま'||
      'みむめもやゆよらりるれろわん'
    ),
     'ガ','ｶﾞ'), 'ギ','ｷﾞ'), 'グ','ｸﾞ'),
     'ゲ','ｹﾞ'), 'ゴ','ｺﾞ'), 'ザ','ｻﾞ'),
     'ジ','ｼﾞ'), 'ズ','ｽﾞ'), 'ゼ','ｾﾞ'),
     'ゾ','ｿﾞ'), 'ダ','ﾀﾞ'), 'ヂ','ﾁﾞ'),
     'ヅ','ﾂﾞ'), 'デ','ﾃﾞ'), 'ド','ﾄﾞ'),
     'バ','ﾊﾞ'), 'ビ','ﾋﾞ'), 'ブ','ﾌﾞ'),
     'ベ','ﾍﾞ'), 'ボ','ﾎﾞ'), 'パ','ﾊﾟ'),
     'ピ','ﾋﾟ'), 'プ','ﾌﾟ'), 'ペ','ﾍﾟ'),
     'ポ','ﾎﾟ'), 'ヴ','ｳﾞ'), 'が','ｶﾞ'),
     'ぎ','ｷﾞ'), 'ぐ','ｸﾞ'), 'げ','ｹﾞ'),
     'ご','ｺﾞ'), 'ざ','ｻﾞ'), 'じ','ｼﾞ'),
     'ず','ｽﾞ'), 'ぜ','ｾﾞ'), 'ぞ','ｿﾞ'),
     'だ','ﾀﾞ'), 'ぢ','ﾁﾞ'), 'づ','ﾂﾞ'),
     'で','ﾃﾞ'), 'ど','ﾄﾞ'), 'ば','ﾊﾞ'),
     'び','ﾋﾞ'), 'ぶ','ﾌﾞ'), 'べ','ﾍﾞ'),
     'ぼ','ﾎﾞ'), 'ぱ','ﾊﾟ'), 'ぴ','ﾋﾟ'),
     'ぷ','ﾌﾟ'), 'ぺ','ﾍﾟ'), 'ぽ','ﾎﾟ')
%mend jisx0208to0201;
```

## プログラム例３．漢字を含む数字表現を全角算用数字に統一変換する

```
data _null_;
  array _mvals{3} _temporary_ ; /* （倍数の値）1:十, 2:百, 3:千 */
  array _uvals{3} _temporary_ ; /* （単位の値）1:万, 2:億, 3:兆 */
  length _kchar $2 _source $256 _knumstr _anumstr $100;
  input _source;

  if _N_=1 then do;
    retain _regid;
    _regid = prxparse('/(０|１|２|３|４|５|６|７|８|９|〇|一|二|三|四|五|六|七|八|九|十¥|百|千|万|億|兆)+/');
  end;

  /* 抽出処理（漢数字を含む数字部分） */
  _mstart=1;
  call prxnext(_regid, _mstart, -1, _source, _mpos, _mlen);
  do while(_mpos ^= 0 );
    _knumstr=substr(_source, _mpos, _mlen);
    do _i=1 to dim(_mvals); _mvals{_i}=0; end;
    do _i=1 to dim(_uvals); _uvals{_i}=0; end;
    _digit1=.;
    _errflg=0;

    /* 解釈処理（数字部分→数値） */
    _klen=klength(_knumstr);
    do _i=1 to _klen;
      _kchar=ksubstr(_knumstr,_i,1);
      _kpos=max(kindex('０１２３４５６７８９', _kchar),
                kindex('〇一二三四五六七八九', _kchar));
      if _kpos>0 then do;
        if _digit1=. then _digit1=0;
        _digit1=_digit1*10+_kpos-1;
      end;
      else do;
        _kpos=kindex('十百千', _kchar);
        if _kpos>0 then do;
          _mchk=0;
          do _j=1 to _kpos; _mchk+_mvals{_j}; end;
          if _mchk>0 then _errflg=1;
          if _digit1=. then _digit1=1;
          _mvals{_kpos}=_digit1;
          _digit1=.;
        end;
        else do;
          _kpos=kindex('万億兆', _kchar);
          if _kpos>0 then do;
            _uchk=0;
            do _j=1 to _kpos; _uchk+_uvals{_j}; end;
            if _uchk>0 then _errflg=1;
            do _j=1 to dim(_mvals);
              _uvals{_kpos}=_uvals{_kpos}+_mvals{_j}*10**_j;
            end;
            if _uvals{_kpos}=0 and _digit1=. then _errflg=1;
            _uvals{_kpos}+_digit1;
            if _uvals{_kpos}=0 or _uvals{_kpos}>=10000 then _errflg=1;
            _digit1=.;
            do _j=1 to dim(_mvals); _mvals{_j}=0; end;
          end;
          else errflg=1;
        end;
      end;
      if _errflg then leave;
    end;

/*  （次ページにつづく）  */
```

```
    /* 最終計算処理（倍数および単位数の合算） */
    if _errflg then do;
      put "NOTE: 数字変換をスキップしました。 " _knumstr;
    end;
    else do;
      _anum=0;
      do _i=1 to dim(_uvals);
        _anum=_anum+_uvals{_i}*10**(4*_i);
      end;
      do _i=1 to dim(_mvals);
        _anum=_anum+_mvals{_i}*10**_i;
      end;
      if _digit1 ne . then _anum=_anum+_digit1;

      /* 置換処理（漢数字→全角算用数字） */
      _anumstr=ktranslate(left(put(_anum, best32.)), "０１２３４５６７８９", "0123456789");
      if _mpos>=1 then do;
        if _mpos>1 then _source=cats(substr(_source, 1, _mpos-1), _anumstr, substr(_source, _mpos+_mlen));
        else _source=cats(_anumstr, substr(_source, _mpos+_mlen));
        _mstart=_mpos+length(_anumstr);
      end;
    end;

    /* 再処理の必要性の検証 */
    call prxnext(_regid, _mstart, -1, _source, _mpos, _mlen);
  end;
  put _source;
cards;
札幌市中央区南二十三条西十四丁目三十八番二百八号
十万千十、三兆四千五百億六十七、七八億九〇一万二
万が一、山口百恵、四万十川
;
```

【出力例】

```
札幌市中央区南２３条西１４丁目３８番２０８号
１０１０１０、３４５０００００００６７、７８０９０１０００２
万が１、山口１００恵、４００１０川
```

【注　意】

このプログラムでは兆の位まで対応しているが、64 ビット(倍精度)で正確に整数値を表現できるのは $2^{53}$ (約 9 千兆)までであり、それを超える値では奇数が表現できないといった不具合が出る。より大きい数の表現に対応させるためには、全体を数値に変換せず位毎に文字列化するようにプログラムを修正すればよい。なお、不適切な数字表現(例:十億万円)や、千単位で区切る表現(例:10千円)、小数点やコンマを含む表現には対応していない。

## プログラム例４．番地表記を統一する

```
data _null_;
  length _addr $100;
  input  _addr ;
  if _N_=1 then do;
    retain _regid ;
    _patnum="(０|１|２|３|４|５|６|７|８|９)+";
    _regid = prxparse('s/('||_patnum||')丁目('||_patnum||')番(地)?(の)?('||_patnum||')(号)?/$1－$3－$7/');
  end;
  call prxchange(_regid, 1, _addr);
  put _addr;
cards;
あさひが丘１０丁目２０番３０４号
あけぼの町３丁目１０番地１２３号
ゆうひが丘１２丁目３４番地の５６
;
```

【出力例】

あさひが丘１０－２０－３０４
あけぼの町３－１０－１２３
ゆうひが丘１２－３４－５６

## プログラム例５．全角英数文字を抽出して出現順に結合する

```
data _null_;
  length _source $256 _numstr $40;
  input  _source;
  if _N_=1 then do;
    retain _regid;
    _regid = prxparse('/(¥x82[¥x4F-¥x58¥x60-¥x79¥x81-¥x9A])+/');
  end;
  _mstart=1; _numstr="";
  call prxnext(_regid, _mstart, -1, _source, _mpos, _mlen);
  do while(_mpos ^= 0 );
    _numstr = cats(_numstr, substr(_source, _mpos, _mlen));
    call prxnext(_regid, _mstart, -1, _source, _mpos, _mlen);
    _mstart=_mpos+_mlen;
  end;
  put _numstr;
cards;
北区川島町９丁目２６－８３４－Ａ１０１
中央区清水通３丁目１４番地　じばさんビルＢ４０５号室
;
```

【出力例】

９２６８３４Ａ１０１
３１４Ｂ４０５

## プログラム例６．PROC SQL で照合処理を行う

```
options mprint;
libname mtest "c:¥temp¥mtest";

/* 照合処理 SQL 生成マクロ定義 */
%macro matching(lv, cond);
  create table mtest.linkab&lv. as select a.aid, b.bid from mtest.acorplist as a, mtest.bcorplist as b where &cond.;
  select count(*) into :lnkcnt from mtest.linkab&lv.;
  %if &lnkcnt = 0 %then %do;
    %put NOTE: Skip delete process. Lv.= &lv ;
  %end;
  %else %do;
    delete from mtest.acorplist where aid in (select unique aid from mtest.linkab&lv.);
    delete from mtest.bcorplist where bid in (select unique bid from mtest.linkab&lv.);
  %end;
%mend;

/* 照合実行 */
proc sql stimer noprint;
  %matching(01, %str(a.name    =b.name     and a.addr    =b.addr     and a.tel=b.tel))
  %matching(02, %str(a.name    =b.name     and a.addr    =b.addr                    ))
  %matching(03, %str(a.name_key=b.name_key and a.addr_key=b.addr_key and a.tel=b.tel))
  %matching(04, %str(a.name_key=b.name_key and a.addr_key=b.addr_key                ))
  /* (中略) */
  %matching(15, %str(a.name    =b.name     and (a.addr eqt b.addr or b.addr eqt a.addr)))
  /* (中略) */
  %matching(30, %str(a.name_key=b.name_key))
quit;

/* 照合結果を統合する instert 文実行用マクロ定義 */
%macro makeInsertSql(lvstart, lvend);
  %do lv=&lvstart %to &lvend;
    %let lvz=%sysfunc(putn(&lv, z2.));
    insert into mtest.linkab_log select unique bid, aid, &lv. as lv from  mtest.linkab&lvz.;
  %end;
%mend;

/* 照合結果の統合 */
proc sql stimer;
  create table mtest.linkab_log as select bid, aid, 1 as lv from mtest.linkab01;
  %makeInsertSql(2, 30)
  create table mtest.linkab_log_cnt as
    select l.aid, l.bid, lv, acount, bcount from mtest.linkab_log as l,
      (select aid, count(*) as acount from mtest.linkab_log group by aid) as a,
      (select bid, count(*) as bcount from mtest.linkab_log group by bid) as b
    where l.aid=a.aid and l.bid=b.bid
  ;
quit;

/* 照合レベル別接続結果件数の出力 */
proc freq data=mtest.linkab_log_cnt; tables lv; where acount=1 and bcount=1; run; /* 1対1の接続     */
proc freq data=mtest.linkab_log_cnt; tables lv; where acount>1 or  bcount>1; run; /* 1対1でない接続 */
```

【解　説】

ライブラリ mtest にある企業リスト A と企業リスト B の照合用データセット（acorplist, bcorplist）を接合する。これらのデータセットには、管理用 ID（aid, bid）、表記の統一処理済みの企業名称(name)および所在地(addr)、電話番号(tel)と、一致度合いを向上させるために加工した照合用キー（name_key, addr_key）等を収録しているとする。

照合処理は proc sql で行っている。照合条件が最も厳しいものから徐々に緩和していくように照合レベル（&lv）と条件（&cond）を定義する。１つの照合レベル毎にマッチした ID の組合せを保存し、当該 ID を持つレコードを照合用データセットから削除する。(必要なら照合用データセットのバックアップを取っておく)。

結果（ID の組合せ）を統合し、ID 毎の件数を数えることで相互のレコードが１対１でマッチしたか否かを判定できる。１対１接続でない組合せや照合レベルが一定以下のものについては、必要に応じて目視確認する必要がある。