Cindyscriptでの整列演算

久保　康幸 *

On sort operator of Cinsyscript

Yasuyuki Kubo *

Abstract

I compared how to arrange by multiple conditions with the sort operator of Cindyscript.

1．はじめに

KeTCindy で成績処理のページ [1] では，得点は高い順（降順），同じ得点の学生は学籍番号を昇順に並べるため reverse を使ったり，100 から引いた得点でソートしたりしている．ここで，疑問は，なぜ一度に複数の条件で並べ替えなかったのかということである．

用意した cdy ファイルにデータを入力して，動作を確認すると，CindyScript のソート演算子は，指定された条件以外については，元の順序を維持していると言うことである．用意した３つの条件を順に使ってソートすることにより，それが確認できた．

この性質を利用して，[1] の例では，用意したデータが始めに学籍番号の若い順（昇順）に並んでいるから，ソートを１つの条件で行なっていると考えられる．これは，「学籍番号の若い順に並んでいる」という暗黙の条件を利用している．今回は，暗黙の条件を仮定しないで，複数の条件をすべて明示した命令でのソートを目指す．つまり，初めのデータが学籍番号の若い順番になってなくとも，得点と学籍番号という２つの条件でソートしたいなら，もし，学籍番号の若い順番にソートされている可能性が高いデータであっても，学籍番号によるソートをしてから，得点による並び替えをするということで，目的の順番に並び替えることが出来る．

この論文では，複数条件を明示的に利用したソートの方法を比較した．なお，動作を確認したのは，Cinderella（version 2.9 build 1898）に付属の CindyScript であり，マニュアルは，その Cinderella のメニューから呼び出される Cinderella マニュアルのリンク先にあるローカルファイルとしての CindyScript のマニュアル (Page last modified on Wednesday 17 of August, 2016.) である．日本語に翻訳する前のマニュアル（英語版）は，ローカルファイルとしてのマニュアル (Page last modified on Monday 14 of July, 2014.) を参照した．最新版を確認するためサイトの接続を試みたが，以前は接続できたサイトが，現在は接続できない．

表 1　Orgdata

Orgdata="

学籍番号	受検者	第１回	第２回	第３回
160101	氏名 01	60	75	63
160102	氏名 02	60	85	60
160103	氏名 03	60	65	33
160104	氏名 04	40	75	42
160105	氏名 05	40	85	45
160106	氏名 06	40	65	37
160107	氏名 07	50	75	41
160108	氏名 08	50	85	43
160109	氏名 09	50	65	69
160110	氏名 10	60	75	63
160111	氏名 11	60	85	60
160112	氏名 12	60	65	33
160113	氏名 13	40	75	42
160114	氏名 14	40	85	45
160115	氏名 15	40	65	37
160116	氏名 16	50	75	41
160117	氏名 17	50	85	43
160118	氏名 18	50	65	69

";

＊ 総合教育科

平成 29 年 11 月 30 日受理

2．並び替えデータの準備

上のようなデータ（表1）を用意した．

このデータは，第1回の得点が3人ずつ60点，40点，50点として得点が高い順でも低い順でもないようにした．それぞれの3人は，75点，85点，65点ずつとして3人の中でも，得点が高い順でも低い順でもないようにした．最初の9人をコピーして，3回の得点が同じ受験者が複数いるようにした．それらは学籍番号か氏名によってのみ区別される．

用意した表1のデータを CindyScript が扱う型に直すため，KETCindy の命令で，

```
datalist=Tab2list(Orgdata);
data=datalist_(2..length(datalist));
```

を実行したものが表2の data である．

CindyScript の演算子 sort をデータ data に対して実行する．なお，2行目の命令は，項目の行を除外して並べ替えるため datalist の2行目以降を data とするという内容である．

表2　data（項目行を除外）

```
data="
160101　氏名 01　60　75　63
160102　氏名 02　60　85　60
160103　氏名 03　60　65　33
160104　氏名 04　40　75　42
160105　氏名 05　40　85　45
160106　氏名 06　40　65　37
160107　氏名 07　50　75　41
160108　氏名 08　50　85　43
160109　氏名 09　50　65　69
160110　氏名 10　60　75　63
160111　氏名 11　60　85　60
160112　氏名 12　60　65　33
160113　氏名 13　40　75　42
160114　氏名 14　40　85　45
160115　氏名 15　40　65　37
160116　氏名 16　50　75　41
160117　氏名 17　50　85　43
160118　氏名 18　50　65　69
";
```

3．複数の条件による並び替え

用意した data を第1回の得点（第3番目の値）で昇順に並べ，それが同じなら，第2回の得点（第4番目の値）で昇順に並べる．そして，それらが同じ受検者を学籍番号（第1番目の値）の逆順（降順）に並べることで，ソート演算子 sort の動作を確認した．その方法を下に紹介する．

3．1　ソートを繰り返す

[1] の例から確認できるように，CindyScript のソート演算子は，指定された条件以外については，元の順番を維持しているから，複数の条件を逆の順番に適用してソートを繰り返せば良い．

目的とする順番のため，まず，次の命令で学籍番号の逆順に並べる．学籍番号をそのまま比較して並べ替えると，番号の若い順になり変化しないので，

```
dt=sort(data,-1*(#_1));
```

これによって，得られるデータ dt は，次のようになる（表3）．

表3　dt（学籍番号を降順）

```
dt="
160118　氏名 18　50　65　69
160117　氏名 17　50　85　43
160116　氏名 16　50　75　41
160115　氏名 15　40　65　37
160114　氏名 14　40　85　45
160113　氏名 13　40　75　42
160112　氏名 12　60　65　33
160111　氏名 11　60　85　60
160110　氏名 10　60　75　63
160109　氏名 09　50　65　69
160108　氏名 08　50　85　43
160107　氏名 07　50　75　41
160106　氏名 06　40　65　37
160105　氏名 05　40　85　45
160104　氏名 04　40　75　42
160103　氏名 03　60　65　33
160102　氏名 02　60　85　60
160101　氏名 01　60　75　63
";
```

次に，第4番目の値で昇順に並べ替える．

```
dt=sort(dt,#_4);
```

という命令により，次のように dt が変化する（表４）．

dt は，4 番目の値で昇順に並び，4 番目の値が同じものは 1 番目の値で降順に並んでいる．

最後に最も優先度の高い 3 番目の値で昇順に並べる．今の dt に対して，さらに次の命令

```
dt=sort(dt,#_3);
```

を実行する．これで，次のように目的の順番になっている（表５）．

表４　dt（表３の dt を加工）

dt="

160118	氏名 18	50	65	69
160115	氏名 15	40	65	37
160112	氏名 12	60	65	33
160109	氏名 09	50	65	69
160106	氏名 06	40	65	37
160103	氏名 03	60	65	33
160116	氏名 16	50	75	41
160113	氏名 13	40	75	42
160110	氏名 10	60	75	63
160107	氏名 07	50	75	41
160104	氏名 04	40	75	42
160101	氏名 01	60	75	63
160117	氏名 17	50	85	43
160114	氏名 14	40	85	45
160111	氏名 11	60	85	60
160108	氏名 08	50	85	43
160105	氏名 05	40	85	45
160102	氏名 02	60	85	60

";

このように，ソートの条件を重ねるだけで，複数条件によるソートができる．おそらく，これが最も単純な考え方であるが，命令を繰り返すのでプログラムの評価は低いと思う．なお，マニュアルを見ても，元の順番をなるべく活かすとは明示していないが不安である．

３．２　数値を加える (1)

例えば，2 つの科目 A, B の得点が，それぞれ 100 点以下と決まっているなら，それぞれの数値は 3 桁以下である．科目 A の得点で並び替えたとき，同じ点数なら，さらに科目 B の得点を見て並び替えたいとするなら，A の得点に 0 を 3 つ続けて（つまり，得点を 100 倍して），B の得点を足す．その数値を比較すれば，まず A の得点で比較して，それが同じなら B の得点を見て並べることが出来る．これの欠点は，どんどん数値が大きくなることである．

それを回避する方法として，プログラムの確認における視認性の問題を考えなければ，A の得点を 1000 倍する代わりに B の得点の最大値より大きい数値（例えば 100）をかけて，B と加えることも出来る．この考え方でプログラムを書いた例を紹介することはやめて，代わりに，次のような例で済ます．

A, B どちらのリストにある値も，すべて 5 より小さな自然数とする．5 は，B のリストの最大値より大きく，5 進数で考えたとき，A からの値を前の桁，B からの値を後ろの桁という 2 桁の数として考えれば，A による値でソートした後，それらが等しい場合に B の値でソートすることになる．

表５　dt（表４の dt を加工）

dt="

160115	氏名 15	40	65	37
160106	氏名 06	40	65	37
160113	氏名 13	40	75	42
160104	氏名 04	40	75	42
160114	氏名 14	40	85	45
160105	氏名 05	40	85	45
160118	氏名 18	50	65	69
160109	氏名 09	50	65	69
160116	氏名 16	50	75	41
160107	氏名 07	50	75	41
160117	氏名 17	50	85	43
160108	氏名 08	50	85	43
160112	氏名 12	60	65	33
160103	氏名 03	60	65	33
160110	氏名 10	60	75	63
160101	氏名 01	60	75	63
160111	氏名 11	60	85	60
160102	氏名 02	60	85	60

";

３．３　数値を加える (2)

前の方法は，並び替えの条件が数値の場合に限られる．次のように修正することで，並び替えの条件が数値でない場合にも複数の条件によるソートができる．なお，もともとのソート演算子は，マニュアルによると「慣例として次の順序があります。ブール値 < 数 < 文字列 < リスト」となっており，数値以外も並び替えが出来る．数値でないものを複数

の条件で並び替えるとき，先の節の方法は，すぐには使えない．そこで，数値でないものの条件については，順位をつけて数値化すれば良い．なお，[1] では並び替えを利用して順位をつけている．並び替えをせずに順位をつけるのに，CindyScript に関数が用意されてないから，少し準備が必要だが，それについては，別の機会に紹介する．

３．４　文字列として扱う

並び替えの条件に文字列のリストを含む場合に，順位付けをして数値化するのは，煩雑である．もし，数値のリストがけた数が揃っているなら，逆に，数値の方を文字列化すれば良い．複数の条件によるソートをしたければ，それらの文字列を繋げれば良い．ただし，文字列のリストも文字列の長さが揃っていることが条件になる．

値を文字列に直す演算子 `text` と文字列の結合 `+` を組合せて，例えば，次のような命令を実行する．

```
dt=sort(data,text(#_3)+text(#_4)
+text(170000-(#_1)));
```

ここでの注意点は，学籍番号（第１番目の値）を降順にするため，170000 から引いていることである．学籍番号の最大値が 170000 より小さいことを利用して，逆の順番にした．

３．５　条件をリストにする

これまでに紹介した方法の不満を一気に解決する方法として，条件をリストにして比較する方法が考えられる．

まず，マニュアルの説明を引用する．

> CindyScript においては、すべての要素はどんな２つの要素でも比較できるような自然な順序を有します。２つの要素は等しいか、どちらかが大きくなります。実数では通常の数の順序です。文字列では辞書順です。複素数はまず実部の順序で並べ、実部が同じなら虚部の順序になります。 リストは、最初の要素を比較します。さらに、慣例として次の順序があります。
>
> ブール値 < 数 < 文字列 < リスト

なお，CindyScript での複素数は，`a+i*b` というように表したものを指す．演算子 `complex` により，ユークリッド座標系はガウスの複素数平面と同一視され，点 `[a,b]` は複素数 `a+i*b` に変換される。

これらを組み合わせて考えれば，ソートに使いたい２つの条件に使う項目の値が数値なら，いったんそれらをペアにして，複素数にしてから，並び替えをすれば良いことが分かる．

つまり，２つの条件をペアにして，複素数にして比較すれば，まず実部の順序で並べて，実部が同じなら虚部の順序になるのと同様に，まず最初の要素を比較して，最初の要素が同じなら，第２の要素の順序にするということで，目的とする順序になる．

しかし，この方法は，今回のような３つの条件を見て並び替えるのには使えない．

そこで，複素数に変換することなく，リストそのものを比較してみてはどうだろうか．ただし，マニュアルの説明のリストの部分を読む限り，条件をリストにすると，リストにした条件の第１の要素だけを見てソートすると考えられる．マニュアルには，条件をリストにした例はない．

念のため，英語版のマニュアルから，この部分を引用する．

> Two lists are compared by the first entry in which they differ.

英語版を読んでも，日本語版と異なる内容には受け取れない．

今回のデータで，リストを比較する方法を確認するため，次のような命令を実行した．

```
dt=sort(data,[#_3,#_4,-1*(#_1)]);
```

なお，３つ組みリストの３番目だけ`-1*(#_1)`というようにしたのは，第１番目の値を降順にするためである．文字列の結合の場合と違って，第１番目の値の最大値を気にすることなく，負の数にするだけで良い．

この命令の実行により，目的とする並び替え（表５と同じもの）を得た．

４．まとめ

今回，確認したなかでは，最後の方法が最も良いと考える．その次は，プログラムの行数が多くなるが，最初に紹介した命令を重ねる方法である．ただし，与えられたデータがある順番で並んでいることを期待せず，並び替えに使う条件は，すべて命令として明示しておくことにする．

また，どちらの方法も，マニュアル（ローカルファイル）に示されていない性質を用いている．Cinderella Japan 代表の入谷昭先生に会ったときに確認することにする．

今回，このように演算子の動作を確認するためにスクリプトを書くには，KETCindy によるデータ処

理に関する [2] および，その経験が役に立った．また，このような複数条件によるソートは，入谷先生のサイト [1] だけでなく，CDC を用いた実験授業 [3] でのデータ整理における要請もあった．

今回の動作確認は，KETCindy の上で行なった．最近の KETCindy については，[4] に紹介されているように R 対応が進められているが，演算子 `sort` は CindyScript 上のものなので，その影響はない．

参考文献および参考 URL

[1] 入谷昭：成績処理 (`https://sites.google.com/site/ketcindy/home/cdv/shi-yan`)

[2] Y. Kubo：Data processing with KETCindy, LNCS, vol.10407 IV,pp.240-250(2017).

[3] 西浦孝治ほか：共同研究成果報告書, 豊橋技科大高専連携教育プロジェクト 2016, (2017).

[4]「Ketpic.com」(`http://ketpic.com/`)

[5]「Cinderella」(`http://www.cinderella.de/tiki-index.php`)

[6]「CinderellaJapan」(`https://sites.google.com/site/cinderellajapan/`)