2 集合の集合

集合とはものの集まりである。ということは、集合をいくつか集めたものも新たな集合になる。たとえば集合 A, B, C, D があるとき、ここから A, C, D を集めた $U = \{A, C, D\}$ もまた集合である。この場合、A は集合 U の要素なので $A \in U$ であることに注意しよう。

具体例で示そう。$A = \{1, 2, 3\}$, $B = \{4, 5\}$ のとき $U = \{A, B\} = \{\{1, 2, 3\}, \{4, 5\}\}$ であるから、$\{4, 5\} \in U$ である。ややこしいのは、$V = \{A, 4, 5\}$ とした場合 $U \neq V$ であることだ。$\{\{1, 2, 3\}, \{4, 5\}\}$ と $\{\{1, 2, 3\}, 4, 5\}$ は、一見同じようだが違うのである。それは、単純に要素数を数えても分かる。U は要素数が 2 であるのに対し、V の要素数は 3 だからである。

PowerShell にこの区別ができるだろうか。

[ps script]
```
PS C:\Users\Yours > $a = 1,2,3
PS C:\Users\Yours > $b = 4,5
PS C:\Users\Yours > $u = $a,$b
PS C:\Users\Yours > $u
1
2
3
4
5
```

`$u` と打ち込むと上のようにデータが列挙されるので、$U = \{1, 2, 3, 4, 5\}$ に見える。`$u` の作り方から $U = \{\{1, 2, 3\}, \{4, 5\}\}$ でなくてはならないはずなのに。`"$u"`と打ち込んでみよう。

[ps script]
```
PS C:\Users\Yours > "$u"
System.Object[] System.Object[]
```

あれれ？　何やら妙な表示が出たぞ。実は、この表示こそが $U = \{\{1, 2, 3\}, \{4, 5\}\}$ を示している。表示の詳しい説明は省くけれど、見た目は一続きの数字の列でも 2 つの配列に分かれていることを意味している。ちなみに`$u[1]` と入力すると、そのことが分かる。

[ps script]
```
PS C:\Users\Yours > $u[1]
4
5
```

配列は 0 番めから数えるので、`$u[1]` は 2 番めの要素である $\{4, 5\}$ を確かに示しているのだ。多少の不便

はあるものの、とりあえず集合らしい処理はしてくれるようである。もちろん表示の仕方を工夫すれば、見た目を整えることは可能である。ただ、この旅はそれが目的ではないので、内部で集合らしい処理がされていれば良しとしよう。

さて、集合の集合を披露したところで、集合の和についても話しておこう。もう一度 $A = \{1,2,3\}$, $B = \{4,5\}$ を持ち出すことにする。

集合の集合とは $U = \{A, B\}$ のように、“集合を集めて” 新たな集合とすることであった。それに対して集合の和とは、“集合の要素を集めて” 新たな集合とすることである。すなわち $A = \{1,2,3\}$, $B = \{4,5\}$ に対する集合の和は $\{1,2,3,4,5\}$ となる。$\{\{1,2,3\},\{4,5\}\}$ とは微妙に違うことに注意してほしい。このように**和集合**を作る場合は、$V = A \cup B$ のように表す[*1]。要素を列挙する場合は $A \cup B = \{1,2,3,4,5\}$ と書く。

ここに $C = \{2,3\}$ という集合があるとき、$A \cup C$ はどうなるだろう。それは $\{1,2,3,2,3\}$ ではなく $\{1,2,3\}$ である。集合の大事な考えなので何度でも言うが、どんなものが集まっているかということが集合の視点であるから、重複は考えに入れないのである。そのことを **PowerShell** で確認しておこう。

[ps script]
```
PS C:\Users\Yours > $a = 1,2,3
PS C:\Users\Yours > $c = 2,3
PS C:\Users\Yours > $a+$c | sort -uniq
1
2
3
```

PowerShell で $\cup$ に似たことをするには、`+` で連結したものを`-uniq` をつけて `sort` すればよい。また、**PowerShell** において `+` は、計算以外の結合に使うこともある。たとえば`"a"+"1" = a1` みたいに。ただし、この旅では扱わない処理である。悪しからず。

ところで、集合の重複を知りたいこともある。たとえば A と C には重複している要素があるから、**共通部分**として取り出すことができる。その場合は、$A \cap C = \{2,3\}$ と書くことにする[*2]。このことを **PowerShell** で実現する場合は、ちょっとだけ手間をかけなくてはならない。

[ps script]
```
PS C:\Users\Yours > $c | foreach {if ($a -contains $_) {$_}}
2
3
```

何をしたかというと、まず`$c` から取り出した要素—これは変数`$_`に入る—を、パイプラインによって

[*1] $\cup$ は “カップ” と読む。
[*2] $\cap$ は “キャップ” と読む。

foreach 文に回している。その$_が$a に含まれれば$_を出力するのだ。これで、結果的に$a と$c に共通の要素を取り出せるのである。

また、A と C の関係は $C \subset A$ でもあるが、実は $C \subseteq A$ と書いてもよい。このことは、不等号の使い方を $3 < 5$ としても $3 \leq 5$ としてもよいことに似ている。具体的な数を書くと $3 \leq 5$ は少々おかしいように感じるが正しいのである。$3 \leq 5$ は、$3 < 5$ または $3 = 5$ が成り立つことを意味するので、“または” の一般的解釈—“どちらか一方”—に照らして正しいからだ。

この手の記号は、明確でないものを考えるときに威力を発するものだ。A さんの手にある硬貨を集合 A の要素、B さんの手にある硬貨を集合 B の要素としよう。A さんが B さんに、自分の手にあるすべての硬貨が B さんの手にあるかどうか尋ねたとする。「1 円玉はありますか？」「10 円玉はありますか？」...。すべての質問に B さんが「ある」と答えれば少なくとも $A \subseteq B$ であることが分かる。B さんは、A さんが質問した以外の硬貨を持っているかもしれないからだ。同じように B さんも A さんに尋ねて、すべての質問に「ある」が返ってくれば、やはり少なくとも $A \supseteq B$ であることが分かる。この状況は何を意味するのだろう。実は $A = B$ であることに気づくだろうか。そう、$A \subseteq B$ かつ $A \supseteq B$ は $A = B$ と**同値**なのだ。それは、$a \leq b$ かつ $a \geq b$ ならば $a = b$ であることと同じである。

□ **集合と集合の関係**

- {{ あれ }, { これ }, ... } は集合 { あれ }, { これ }, ... を要素とする集合、{ あれ, これ, ... } は “あれ”, “これ”, ... を要素とする集合である
- 集合 A と集合 B の和集合は $A \cup B$ で表す
- 集合 A と集合 B の共通部分は $A \cap B$ で表す
- 集合 A, B においてで $A \subseteq B$ かつ $A \supseteq B$ あることと $A = B$ であることは同値である