プログラミング基礎'94 ＃4

久野　靖*

1994.7.25

第 3 回資料の後半を先送りしたので、今回はそこから始まります。

11　手続き

さて、前の節でやったように、PAD ではある項目以下を別の場所に書くことができるから、1 枚の図があまりに込み入って手に負えなくなることは避けられる。

プログラムの上でもこれと同様のこと…つまり、ひとまとまりの操作に名前をつけて独立して記すことができれば、同様の利益 (つまりメインプログラムが長過ぎてよくわからなくなるのが防げる) があるはずである。まさにそのために、プログラミング言語では「手続き」ないし「サブルーチン」が用意してある。

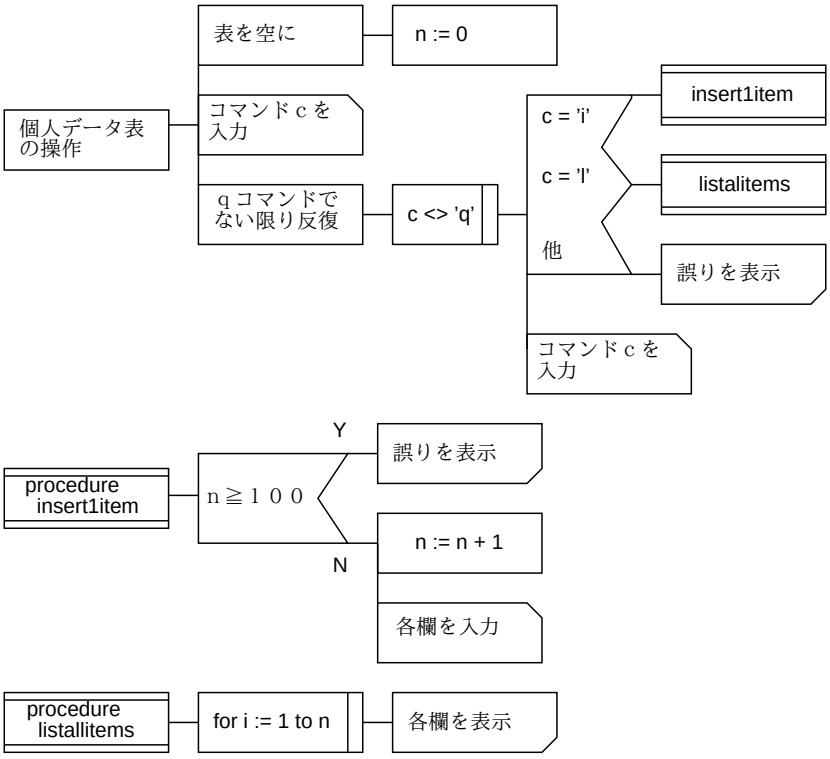

図 1: 表のコマンドによる操作 (手続き版)

これに対応する PAD を図 1 に示す。実は先の PAD とほとんど同じで、ただし前に「切り継ぎ」になっていた箱が今度は上下に線が入っている。これは「手続き呼び出し」と「手続き定義」を

*筑波大学大学院経営システム科学専攻

示している。手続きというのは PAD に沿って考えれば、ある箱以下の PAD ひとまとまりを名前をつけて別途記したものである。そして、プログラムの上では対応して、ある一連の動作ひとまとまりに名前をつけて別途記したものである。図 1 を Pascal に直したものを以下に示す。

```
program table2(input, output);
type str = array[1..20] of char;
var  name: array[1..100] of str;
     age:  array[1..100] of integer;
     weight: array[1..100] of real;
     n: integer;
     c: char;

procedure insert1item;
begin
  if n >= 100 then begin
    writeln('table overflow.')
  end
  else begin
    n := n + 1;
    write(' name> '); readln(name[n]);
    write(' age>  '); readln(age[n]);
    write(' weight> '); readln(weight[n])
  end
end;

procedure listallitems;
var i: integer;
begin
  for i := 1 to n do begin
    writeln(name[i]:12, age[i]:2, weight[i]:8:2)
  end
end;

begin
  n := 0;
  write('Command> '); readln(c);
  while c <> 'q' do begin
    if c = 'i' then begin insert1item end
    else if c = 'l' then begin listallitems end
    else begin
      writeln('unknown command: ', c);
    end;
    write('Command> '); readln(c)
  end
end.
```

つまり、手続きの定義はこれまでのプログラム (主プログラム) のミニチュア版で

```
procedure 手続き名;
var ....
begin
  ....
end;
```

という形をしている (var 部はあってもなくてもよいが、あった場合にそこで用意した変数はこの手続きの実行中にだけ使うことができる)。そして、定義が終った後で普通の文の場所に

手続き名;

というのがあるとそこで手続き定義に書かれている内容が実行される (これを手続き呼び出しという)。

それでどうですか? プログラムの内容自体は変わらないものの、ずっと読みやすくなったでしょう? PAD を適宜分割すると読みやすくなるのと同様に、プログラムの上でも適当な大きさの動作ひとまとまりを名前をつけて別に記すことで読みやすくなる。

練習 4-1 上の表操作プログラムに、次のようなコマンドを追加せよ (前と同じ)。ただし各コマンドの動作は手続きとして分離すること。まず PAD を書き、続いて Pascal に直して動かせ。

(a) 名前を指定すると何番目の項目かを表示する。
(b) 指定した番号の項目を削除する。
(c) 指定した 2 つの番号の項目を入れ換える。
(d) 表を名前順にならべ替える。

練習 4-2 'l' コマンド以外のコマンドについても、票を操作した後で全要素を表示してくれるように改良せよ。(PAD、Pascal とも。)

12 手続きの引数

練習 4-2 で見たように、手続きは複数箇所から同じ内容のまとまった処理を利用したい場合にもきわめて有効である。(実はプログラミング言語に手続きなるものが導入されたもとの理由はこちらである。)

もっとも、プログラムを見やすく分割する、とういうことの方が現在では重要になっていると筆者は思うけれど。だから、いくらかでもまとまった概念の操作は手続きにしておくのがよい。

しかし、手続きの内容がどこから呼び出された場合でも「完全に同じ」であると少々不便である。たとえば「2 つの変数の内容を交換する」というのは充分一般的な命令列だと思うが、それをそのまま手続きにしてしまうと「特定の 2 つの変数を交換する」ものができてしまう。交換したい変数は手続き呼び出しの場所に応じて違うだろうから、それでは役に立たない。

これを解決するには、手続き呼び出しにおいて「**この変数とこの変数を**交換する」のように付加的な情報を渡すことができればよい。実際、ほとんどのプログラミング言語では (BASIC は駄目!) これができるようになっている。例えば Pascal で 2 つの (実数) 変数の内容を交換する手続きは次のように書くことができる。

```
procedure swapreal(var x:real; var y:real);
var z: real;
begin
  z := x; x := y; y := z
end;
```

これを呼び出すところでは

```
swapreal(a[i], a[i+1]);
```

のように、任意の実数型の箱 (変数でも配列要素でもよい) を指定する。すると、これらのされたものが swapreal の中では x や y という名前に「重ね合わせられて」実行される。これらの「重ね合わせのために用いられる名前のことを「仮引数」ないし「仮パラメタ」と呼び、呼び出す時に指定するものの方を「実引数」ないし「実パラメタ」と呼ぶ。実引数と仮引数は個数が一致し、また対応するものそれぞれの型が一致しないといけない。

あと 1 つ。上の例では仮引数の指定のところに「var」が書かれているが、これを略すこともできる。var つきの仮引数を「変数パラメタ」、var なしのものを「値パラメタ」と呼ぶ。その違いは、変数パラメタでは仮引数を書き換えると対応する実引数も変化する代りに実引数は変数の箱でなければならないのに対し、値パラメタでは仮引数の書き換えは実引数に影響しない代りに実引数として任意の式が書けるということである。

練習 4-3 前回やった、配列の内容を小さい順に並べ替えるプログラムを、swapreal を使うように書き直せ。

練習 4-4 パラメタとして実数変数と実数の値を渡すと、変数の内容を指定した値だけ増やす手続き increal を書け。

13 関数

既に見てきたように、平方根を求めるなどの計算はハードウェアの命令としてはなく、それアルゴリズムによって実現している。しかしプログラムの上では

```
sd := sqrt(v);
```

などのように短くそれらしく書けると嬉しい。そのためには、ちょうど手続きと同じように一連の計算を実行し、ただし最後に 1 つの値を結果として返すような機能があればよい。これを「関数」という。

Pascal では関数はほとんど手続きと同様の構文で定義する。ただし違うのは次の点である。

- procedure の代りに function と書く。
- 頭書きの末尾でその関数が返す型を指定する。
- 本体の中で、返したい値を関数名に代入する。

例えば整数の絶対値を計算する関数は次のようになる。

```
function iabs(x:integer): integer;
begin
  if x >= 0 then begin iabs := x end else iabs := -x end
end;
```

練習 4-5 次のような関数を作れ (まず PAD を書き、次に Pascal で)。

(a) 2 つの整数のうち大きい方 (等しければその値) を返す関数 imax2。
(b) 3 つの整数のうち最大のものを返す関数 imax3。
(c) 4 つの整数のうち最大のものを返す関数 imax4。
(d) 1 つの正整数を受け取りその階乗を返す関数 fact。

(e) 2 つの正整数を受け取りその最小公倍数を返す関数 `gcd`。

(f) n、r を受け取り、n 個のものから r 個選ぶ組合せを計算する関数 `comb`。

なお、関数を作った場合、それだけでは実行できないから、必要な引数を読み込み、関数で値を計算して書き出す主プログラムを用意する必要がある。この手の主プログラムを (関数を drive するのが目的だから) ドライバなどと呼ぶ。