プログラミング基礎'94 ＃1

久野　靖*

1994.7.21

0　はじめに

この科目は既にあちこちで述べているように、プログラミングを経験したことのない人にプログラミングについて学んでもらうためのものである。それで、最初から水を掛けるようだが、わずか5日間、それも各2時間ちょっとの講義+演習だけで自分の役に立つ (たとえばアンケートを集計して自由自在に分析するなど) プログラムがすらすら書けるようになるなどとは間違っても期待しないように。なにしろ、情報専門学科の学生は、2年生と3年生の間2年間にわたって、プログラミング関係の科目を常時3コマくらい取らされ、少なくとも週のうち丸2〜3日は計算機とにらめっこで、夏休みなどもプログラミングの課題に追いかけられたりする。それが5日間でできてしまってはみんな困るでしょう?

ではこの講義は何のためにあるのか? まさかあなたはこれから修行してプログラマに転身しようという気はないでしょう? しかし、既にできているプログラムを使ったり、または部下や後輩や下請けにプログラムを開発させたりすることはあるのではないですか? だから、

- プログラムを作る、というのは要はどういうことなのか、その感覚を持ってもらう。
- プログラムと密接な関連にある、「アルゴリズム」ないし「手順」について、知ってもらう。
- なぜプログラムを作ると嬉しいのか、なぜプログラムを作るのは大変なのか、について実感を持って理解してもらう。

というのが本科目の目的である。ついでに (!?) もう1つ、

- プログラムが書けるようになる。

これもやるからには目指さないと。上で言ったことと矛盾しているって? 実は「役に立つプログラムが書ける」のと、単に「プログラムが書ける」のと間には大きな違いがあるのでした。

あと、単に「書ける」レベルのプログラムでも、「役に立つ」度合いの方を十分譲歩すればちゃんと「役に立つ」こともある。だから例えばちょっと電卓や手持ちのソフトではやりにくいが、プログラムすれば意外と簡単、という問題には十分役に立つだろうと思う。ということで、頑張りましょう! [1]

*筑波大学大学院経営システム科学専攻

[1]あと、「役に立つ」ばかりが能ではなく、計算機について造詣が深まるとか、頭の体操かつ知的な楽しみになるという利点もあります。

1　アルゴリズム

「アルゴリズム」ないし「手順」とは、何かを求めるための具体的な計算方法をいう。たとえば、海外のニュースなどを見ていると気温が華氏で表示されているので、それは摂氏では何度かな、と知りたくなる。それを求めるには、華氏の温度を f として、

$$c = \frac{5}{9}(f - 32)$$

により値 c を求めればよい。これも立派なアルゴリズムである。

ところで、何をもって「具体的」というかは実は簡単ではない。たとえば、n 個のボールを (すき間があってもいいから) 正方形の箱に平らに入れたければ、その箱の 1 辺の長さはボールの直径の

$$l = \lfloor \sqrt{n} \rfloor$$

倍であればいい。ただしこれは、「切り上げ」とか「平方根」とかの計算手順が具体的にわかっていれば、である。もし加減乗除だけしか使えないのなら、例えば

$$l^2 \geq n$$

なる最初の l が見つかるまで $l = 0, 1, 2, 3, 4, \ldots$ を順に試していく、という手順を使うことになろう。

実は、四則演算も「具体的」かどうかは議論があってよい。たとえば小学生はどうやって四則演算をやるかの手順を一生懸命憶えさせられるわけである。しかし幸いなことに、計算機はには四則演算のためのハードウェアが備わっているから、それにお願いすることにして、まずは四則演算は「具体的」だということにする。

2　アルゴリズムの表記方法と PAD

アルゴリズムを計算機での処理に使おうと思うと、「具体的」の他にも色々問題がある。まず、上の温度の例では「華氏の温度を f として」などと書いてあったが、人間が手で計算するならそれでいいとしても、計算機で処理する場合にはそこで「華氏の温度を計算機に読み込ませる (入力する)」という動作が必要である。その次に式にしたがって四則演算を行ない、最後に結果を「人間に見えるように表示する (出力する)」動作が必要になる。これらのことを順番におこなって始めて、計算機で仕事がこなせるのである。

そこで、計算機の分野ではもっとこの「入力」「出力」「順番の動作」がはっきりするような書き方を工夫することが古くから行なわれている。その古典かつ代表が「フローチャート (流れ図)」だが、現在ではこれは様々な弱点があってよくないとされている。ここでは PAD 図と呼ばれるものを使用する。上の温度変換の手順を PAD で記したものを図 1 に示す。

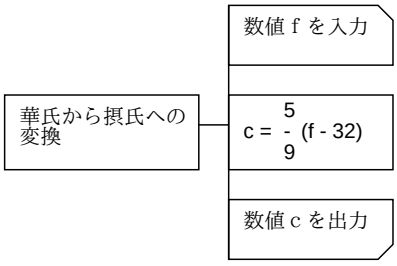

図 1: 華氏摂氏変換の PAD

ここで、箱は何らかの計算機による処理を表している。特にそれが入力、出力の場合には、それぞれ右上、右下が斜めに欠けた箱を使う。箱の連なりが縦線で結んである場合には、それらを上から順にやっていくことを示す。そして、ある箱から右に引き出して記述がある場合には、その箱に書いてあるのは概略で、より具体的には右側の記述によること (詳細化) を示す。

ここにあるように、どんなプログラムでもまずそのプログラムが全体として何をするかを書き、その詳細化として手順を書いていくのがよい習慣である。

練習 1-1 次のような計算の手順を PAD で書いてみよ。

a. 二つの数 a、b を入力し、その合計を出力する。
b. あるものの長さが X メートル Y センチ Z ミリの形で表されていたとして、それをセンチ単位に換算する。(例えば 1 メートル 10 センチ 5 ミリ→ 110.5 のように。)
c. 3 日前、2 日前、1 日前、今日の円ドルレートを与えて、この 3 日間のレート変動 (+:円高、- :円安) を打ち出す。

3 プログラミング言語 Pascal

「プログラム」とは、計算機に具体的な計算を行なわせるための指示をいう。「アルゴリズム」との違いは判りますか? 「アルゴリズム」は考え方の道筋だから、それを書き表す方法はいろいろある。上では日本語とか数式を使って書き表していた。それに対して、計算機で計算させるためには日本語や数式ではやりにくいから、もっと杓子定規な書き方の規則を設けて、それに従って書く。これがプログラムである。

その「書き方の規則」も実は様々な流儀がある。つまり各種のプログラミング言語に相当する。どの流儀も、それなりの長所と弱点がある。計算機科学基礎では C とかシェルとか awk とか tcl/tk などいろんな言語が出てきたが、ここでは Pascal と呼ばれる言語を用いる。Pascal はチューリヒ工科大学の Writh 教授が設計した言語であり、教育用によく使われている。また、強い型検査機能のためプログラムの間違いが発見しやすい。反面、大きなプログラムや、システムプログラミング (各種システム機能を駆使するようなもの) にはあまり向いていないとされている。

では、先の華氏と摂氏の変換プログラムを Pascal で記述してみる。

```
program sam1(input, output);
var c, f: real;
begin
  write('degree F> ');
  readln(f);
  c := (5.0 * (f - 32.0)) / 9.0;
  writeln('degree C = ', c:5:2)
end.
```

先の PAD と比べてみると、いろいろ細かい記述が増えているなあと思う。

1. プログラムは「`program` プログラム名 `(input, output);`」という書き出しで始まらなければならない。
2. 値を表す名前 (変数) は「`var` 変数名, `...:  real;`」によって予め使用を予告しておかなければならない。
3. `begin` から `end.` までの間に手順を順次; で区切って書く。なお、この区切られた各 1 手順を Pascal では「文」と呼ぶ。

4. 入力は `readln`、出力は `writeln` で表す。なお `write` は改行なしになること以外は `writeln` と同じ。
5. 加減乗除は`+-*/`でそれぞれ表す。1 行に書かないといけないので、適宜 () でくくる必要がある。
6. 変数に値を設定するには代入 (「`:=`」と呼ばれる記号を使う。代入は「等しい」とは違う意味 (cf. `x := x + 1.0`)。

さらに `writeln` では文字列を書く時には'' で囲む、数値を書く時には「値:幅:桁数」の指定を行なうと、全体として指定した幅以上の文字数を用いて、少数点以下を指定した桁数ぶん表示する、といったこともおぼえる必要がある (ああ面倒だ!) [2]

要は、プログラミング言語というのは計算機に対して実際にアルゴリズムを実行する際のありとあらゆる細かい所まで指示できるように決めた形式であり、だからプログラムのどこか少しでも変更すると計算機の動作もそれに相応して変わるか、(もっとよくあることだが) そういう風には変えられないよ、と怒られることになっている。いくら怒られても偉いのは人間であって計算機ではないのだから、そういうものだと思って許してやって頂きたい。

さて、ソースプログラムをエディタを使ってファイルに入れるとかコンパイルすると機械語になるとかいう話は計算機科学基礎でやったのでもういいことにする。ただ、今回は言語が Pascal なので、ファイルの名前の最後は「`.p`」にすることと、コンパイルするコマンドも「`pc` ファイル名」に変えることだけおぼえて頂ければよい。では実行してみよう。

```
% pc sam1.p
sam1
Translation completed.
% a.out
degree F> 60
degree C = 15.56
%
```

練習 1-2 上の例題と同じものを打ち込んで動かしてみよ。出力の所の幅指定をいろいろ変えたり、桁指定をなくしたりして出力形式がどう変わるか観察せよ。

練習 1-3 練習 1-1 で作った PAD に対応するプログラムを書いて動かせ。

4 条件判断

今度は「数値 x の絶対値を求める」という問題を考えてみる。絶対値というのは 0 以上の数についてはその数自身、負の数についてはその符号を反転したものだから、0 以上かどうかで処理を分けなければならない。このように条件によって処理を選択する場合、PAD では図 2 のように書く。

Pascal ではこのような条件による選択は if 文と呼ばれる形で記す。その一般的な形は、次の通り。

```
if 条件 then begin
  条件が成り立った場合の処理 (文の並び)
end
else begin
  条件が成り立たなかった場合の処理 (文の並び)
end;
```

[2] 桁数を指定しない場合には、指数形式での表示になる。

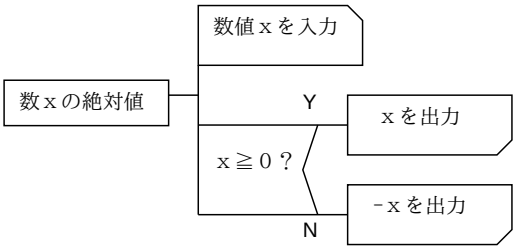

図 2: 絶対値計算の PAD

なお、2 番目の end の後のみ; が書いてあるのに注意。これは、if 文も全体として 1 つの文だから、それが終わった後次の文との区切りが必要になることを意味している。もしすぐ次が (例えばプログラム全体などの end ならこの; はいらない。どちらにせよ、else の前には; は書いてはいけない。(面倒だ!) というわけで、この構文を使って先の PAD をプログラムにしたものを示す。

```
program sam2(input, output);
var x: real;
begin
  write('X> ');
  readln(x);
  if x >= 0 then begin
    writeln('absolute value = ', x:6:2)
  end
  else begin
    writeln('absolute value = ', -x:6:2)
  end
end.
```

なお、writeln の中に任意の式が書ける (それを利用して $-x$ を計算している) のに注意。なお、条件としてはとりあえず大小関係を表す>、>=、=、<>、<、<=の 6 種類が使えるものと思って頂ければよい。

ところで、値を表す名前すなわち変数は、一度値を決めたらそれ以上変えられないというものではなく、何回でも新しい値を代入できる。それを利用して、同じ問題を図 3 のようにも設計できる。

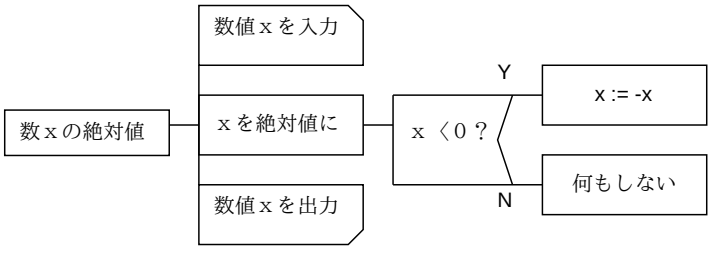

図 3: 絶対値計算の PAD(変数書き換え)

つまり、「x を絶対値にする」のは、x が負の場合に符号を反転することでできるわけである。このように、いきなり具体的な処理を書いてしまうより、まず処理の概要を判りやすく書き、徐々に詳細化していく方が読みやすく間違いの少ない設計になる。

これを Pascal にする場合には「何もしない」に相当する部分の `else` 以下はそっくりなくてよい。従ってプログラムは次のようになる。

```
program sam3(input, output);
var x: real;
begin
  write('X> ');
  readln(x);
  if x < 0 then begin x := -x end;
  writeln('absolute value = ', x:6:2)
end.
```

どっちが好みですか? どちらが正しい、ということはない。このように、同じ答えでもそれを計算するプログラムは様々に設計できる。その自由度を活かしていかに「美しく」表現するかはプログラマのセンス次第である。

あともう 1 つ。Pascal プログラムは、どこで行かえをするか、どこに空白を入れるかについては、(名前や文字列の途中を除けば) 自由である。このプログラムはそれを利用してコンパクトに収めてある。プログラムが短いのは、ぱっと見てわかりやすい限りにおいていいことである。やたら短くしようとばかり詰め込むのは感心できない。

練習 1-4 次のような手順を設計し PAD で記述せよ。

a. 二つの数値 X、Y を読み込み、大きい方 (等しい場合にはその等しい値) を出力する。
b. 三つの数値 X、Y、Z を読み込み、最大の値を出力する。
c. ある日の X 時 Y 分 Z 秒から X' 時 Y' 分 Z' 秒までは何時間何分何秒あるかを計算する。

練習 1-5 練習 1-4 で作った設計をプログラムにして動かせ。