プログラミング基礎'00　＃1

久野　靖*

2000.6.3

はじめに

こんにちは、久野です。今週から「プログラミング基礎」を開講します。シラバスに書かれていたように、この科目は「まったくプログラミングが初めての人」を対象としています。皆様からのメールを拝見した限りではプログラミング経験者の方も多いようですが、全員というわけではないため、やはり「初心者対象」という線は維持します。経験者は退屈でも文句を言わないように…

言語もシラバスに書かれていたように Java を使用します。Java の処理系は Windows、Macintosh、Unix でまったく同じものが動きますので、自宅などで実習することも OK です。有償の開発環境も売られていますが、Java の開発元が出している無償の「JDK」と呼ばれている開発キットを前提としますので特段の理由がない限りそれをダウンロードすればよいでしょう。

```
http://java.sun.com/jdk/
```

ここから「Java 2 SDK」または「JDK 1.1.x」(1 つ前の版だが OS によってはこちらしかまだない) のなるべく新しいものを取り寄せてください。ただし大きいので雑誌の付録 CD-ROM などを狙う方がいいかも。なお、Microsoft 製の環境 (Visual J++など) は購入しないことを勧めます。

講義の進め方ですが、プログラミングはやはり手を動かしてプログラムを作らないと身につきません。なので、時間内もなるべく演習に時間を割くようにしますし、演習の一部を「宿題」とします。ハードだと思いますが翌週までにやってみてください。ただし演習課題から自分の腕前に応じて選べるようにしますので、自分の能力や掛けられる時間に応じて選んでください (易しすぎるのを選ぶと上達しませんからそのつもりで)。ではこれから 5 週間、よろしくお願いします。

1　プログラムとアルゴリズム

今までは皆様は、計算機である処理をしたければ、それを行うためのコマンドが既に存在していて、その名前を言うだけでいい、という形で計算機を使って来たはずである。しかし現実には、自分のやりたいことすべてが予めコマンドとして用意されている、などということはもちろん不可能である。

ではどうするか? それには、「どのような処理をしたいか」という情報を、計算機に与えればよい。この情報のことを (散々聞いたことがあると思うが)「プログラム」という。実は「既にあるコマンド」というのは、「あらかじめ計算機に入力されて蓄えられているプログラム」に他ならない。

言い替えれば、計算機というのは実はプログラムに従って情報を処理する装置なわけである。そして、どのような処理であっても (たとえその計算機が製造された時には夢想だにされなかったようなものでも)、その処理方法をプログラムとして与えさえすれば処理できるようになる、というところに計算機の特徴があるわけである。

なお、ここで「プログラム」と「アルゴリズム」ないし「手順」の違いを整理しておく。「アルゴリズム」ないし「手順」といった場合、それは加工の方法自体をいう。例えば駒場地区正門から渋谷駅まで歩いて行く行

*筑波大学大学院経営システム科学専攻

き方、といったものである。一方、「プログラム」といった場合はアルゴリズムを計算機に与えられるような形に表現したものをいう。例えば道筋を他人に教えるためには、行き方を記した地図やメモといった具体的な形に表現する必要がある。

演習 1 「アルゴリズム」と「プログラム」、「行き方」と「行き方を示した地図」のような関係にあるものの例を他にも挙げてみよ。

2 計算機のためのアルゴリズム

計算機で使う「アルゴリズム」ないし「手順」は、何かを求めるための具体的な計算 (ないし情報の加工) 方法でなければならない。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、計算機には四則演算のための機能がもともと備わっているから、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。

演習 2 仮に、かけ算と割り算は「具体的でない」(つまり使えない) ものとする。足し算と引き算と数の大小比較は使ってよいとして、次のことをするアルゴリズムを考えよ (ただし x、y は正の整数とする)。

a. 数 x が奇数か偶数かを判定する。
b. 数 x と y の積を求める。
c. 数 x と y の最大公約数を求める。

3 計算機のためのアルゴリズム

前節まででは、計算されるべきデータはなぜか x だの f だのという変数に入っていたかのように書いていたが、ご存じの通り、計算機ではキーボードからデータを打ち込まなければプログラムに情報が伝わらないし、プログラムの中でいくら計算が終わっても画面に表示されなければ何が何だか分からない。すなわち、入力 (人間から計算機にデータを渡すこと) と出力 (計算機から人間にデータを渡すこと) も明確に示さなければならない。

しかも、何と何を入力する、というだけではだめで、まずこれを入力し、次にこれを入力し、というふうに順番まで明確に示さなければ何をどの順で入力してよいかも分からなくなってしまう。出力も同様である。

このため、計算機のためにアルゴリズムを記述するときは、(1) 各段階ごとに、(2) 入力、出力、計算の動作などを明確に、記述する必要がある。たとえば華氏摂氏変換を考えてみよう。

- 華氏の温度 f を入力する。
- $c = \frac{5}{9}(f - 32)$ により値 c を求める。
- 摂氏の温度 c を出力する。

このような段階になる。こういう書き方のことを「疑似コード」と呼ぶ (「コード」というのは「プログラムの断片」を意味しているが、これはあくまでも日本語だからプログラムではなく、「のようなもの」なので、「疑似コード」)。

華氏摂氏変換では処理が 1 直線だったが、箱詰め問題の場合には「繰り返し」が必要になる。

- ボールの個数 n を入力する。
- 箱の一辺 l を 0 とおく。
- $l^2 < n$ が成り立つ間繰り返し:
- $\quad l + 1$ をあらたに l とおく。
- 以上を繰り返す。
- 一辺の長さ l を出力する。

ここで注目して欲しいのは、「$l + 1$ をあらたに l とおく」という部分で、つまり l というのは「変数」とは言っても方程式のように 1 つの値を持っているというより、計算の途中で自由に値を書き換えられる「入れもの」みたいなものだという点である。

もう 1 つのポイントは変数と繰り返しの関わり方で、$l^2 < n$ が成り立つ間繰り返し l を増やして行くのだから、繰り返しが終わったときは $l^2 < n$ が成り立たなくなって、ということは $l^2 \geq n$ が成り立っていて、しかも l はこの条件が成り立つ最少の整数になっている、ということである (だって 1 つずつ増やして試して行くのだから)。この辺の感覚が分かることが、プログラミングのコツだと言ってよいだろう。

演習 3 演習 2 で考えたアルゴリズムを、入力や繰り返しなどを含んだ計算機用のアルゴリズム (疑似コード) に書き直してみよ。

4 アルゴリズムとプログラミング言語

「プログラム」とは、アルゴリズムを実際に計算機与えられる形で表現したものであり、その具体的な「書き表し方」がプログラミング言語である。「言語」という名前はついているが、プログラミング言語は「日本語」や「英語」などの「自然言語」とは違って、あくまで計算機に読み込ませて処理できることが前提の「人工言語」であり、そのため書き方も杓子定規なのがふつうである。

プログラミング言語も、すでにご存じと思うが、さまざまな特徴を持つさまざまなものが使われている。ここでは Java と呼ばれる言語を用いる。Java は、C や C++といった言語に比べると「型とか例外とかについてきっちり書かせることで、お行儀のよいプログラミングを行う」という特徴があるので、その意味では入門教育にも向いていると考え、ここで採用するものである。

では、先の華氏と摂氏の変換プログラムを Java で記述してみる。

```
import java.io.*;

class Sample1 {
  public static void main(String args[]) throws Exception {
    BufferedReader in = new BufferedReader(new InputStreamReader(System.in));
    System.out.print("degree F = "); System.out.flush();
    double f = (new Double(in.readLine())).doubleValue();
    double c = (5.0 * (f - 32.0)) / 9.0;
    System.out.println("degree C = " + c);
```

```
  }
}
```

先の疑似コードと比べてみると、いろいろ細かい記述が増えているので、順次説明しよう。

1. Java のプログラムは「クラス」と呼ばれる単位の集まりである。もっとも単純なプログラムはクラス 1 つだけだから、次の形をしている。

   ```
   class クラス名 {
     中身...
   }
   ```

2. プログラムとして実行するクラスは、main というメソッド (手順ないしアルゴリズムを記述する単位) を持たなければいけない。そしてそれは次の形をしている。

   ```
   class クラス名 {
     public static void main(String args[]) throws Exception {
       手順の中身...
     }
   }
   ```

3. キーボードからの入力ではだいたい「1 行読む」という機能を使うことが多いのだが、なぜか Java の標準の入力機能ではこれがついていないので、以下の例題ではすべて BufferedReader というクラスの 1 行入力機能を使う。このクラスは java.io パッケージに入っているので、先頭に「import java.io.*;」という指定 (おまじない) が必要である。また、まず BufferedReader を in という変数に入れて置くことにするので、併せて次のようになる。

   ```
   import java.io.*;

   class クラス名 {
     public static void main(String args[]) throws Exception {
       BufferedReader in = new BufferedReader(new InputStreamReader(System.in));
       本来の中身...
     }
   }
   ```

   BufferedReader は 1 文字単位の入力を行うクラス InputStreamReader を利用し、InputStreamReader は System.in に入っている InputStream を利用することになる。

4. ここからいよいよ手順の中身になる。まず華氏の温度 F を入力してもらうのだが、何を入力するのか分からないと利用者に不親切なので「これこれを入力してね」というメッセージを最初に表示する。

   ```
   System.out.print("degree F = "); System.out.flush();
   ```

   System.out.print(...) というのは文字列を行かえなしで画面に表示する。ただしその後で System.out.flish() を実行してあげないと画面に現われない。なお、1 つの動作ごとに最後に ; を入れて区切る。動作は 1 つずつ順に実行されていく。

5. 次に 1 行ぶんの文字列をキーボードから読み込み、それを実数値に変換して変数 f に入れる (詳しい説明はちょっと後まわし)。

   ```
   double f = (new Double(in.readLine())).doubleValue();
   ```

6. f の値を元に摂氏の温度を計算し、実数変数 c に入れる。

```
double c = (5.0 * (f - 32.0)) / 9.0;
```

なお、Java など普通のプログラミング言語では数式は 1 行に書かないといけないため割り算には「/」をつかい、適宜かっこでくくる。また変数名も 1 文字とは限らないので掛け算も「*」で表す。

7. 結果を表示する。System.out.println(...) は前述の System.out.print(...) とほぼ同じだが、行替えをおこなう。こちらの場合は System.out.flush() は不要。

```
System.out.println("degree C = " + c);
```

なお、「+」は左辺が文字列 ("...") のときは足し算ではなく文字列の連結演算になり、右辺は適宜文字列に変換される。

以上で手順の中身はおしまいである。

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。

では実際にこれを計算機で動かそう。まず、エディタ (Emacs、Mule、vi など好きなものでよい) で上と同じ内容を「Sample1.java」というファイルに打ち込んでもらう。Java プログラムを入れるファイルはかならずクラスの名前と大文字小文字まで含めて同じでものの後ろに「.java」をつけた名前にしなければならない。

次にこのプログラムを実行に適した形に変換する。これをコンパイルする、という。なお、コンパイルする前の (人間が読める形の) プログラムを「ソースプログラム」と呼ぶ。Java のコンパイルには javac コマンドを使う (システムによってはちょっと時間が掛るかも)。

```
% javac Sample1.java
%
```

なお、「%」はシェルプロンプトのつもり。コンパイルする時に、プログラムの形に整っていない点やおかしい点があればメッセージがでる。何もメッセージがでなければ成功で、Sample1.class という出力ファイルができている。

次に、このプログラムを実行するには java というコマンドを使う。なお、java コマンドではクラス名の部分だけを指定すること。実行が始まるとすぐ最初のメッセージがでて来るので、華氏の温度を入力してあげよう。

```
% java Sample1
degree F = 100    ← 「100[RET]」と入力した
degree C = 37.777777777778
%
```

苦労のわりにはあんまり大したことはない感じだが、まあ初心者の第 1 歩ということで、そうがっかりしないで戴きたい。

演習 4 例題の華氏摂氏変換プログラムをそのまま、打ち込んで実行させてみよ。入力に数字でないものを入れるとどうなるかも試せ。

演習 5 動いたら、"..."の中身を別のものに変更したり、System.out.println を 2 回行うなど、プログラムを直して再度実行し、変更の結果が反映されることを確認せよ。

演習 6 次のような動作をするプログラムを書いて動かせ。

a. 2 つの実数を入力し、その和を出力する。

b. 2 つの実数を入力し、その和、差、積、商を出力する。

c. 円錐の底面の半径と高さを入力し、体積と表面積を出力する。

5 うんちく: 値とオブジェクト

Java が扱うデータには大きく分けると「値」と「オブジェクト」の 2 種類があり、その区別は次のようになっている。

- 値 — 数値 (四則演算の対象になるもの) と、文字。四則演算程度の機能だけを持つ。単純なデータ。
- オブジェクト — さまざまな「もの」を表すデータ。込み入った構造や機能を持つことができる。クラスによって定義され (てい) る。言い替えればクラスとはオブジェクトの種類である。

値の種類とオブジェクトの種類 (クラス) を合わせたものを「型」といい、Java ではすべての変数 (値やオブジェクトを入れておくいれもの) は

```
型 変数名;
型 変数名 = 初期値;
```

の形で宣言することになっている。

値としては `int`(整数)、`char`(文字)、`float`(実数)、`double`(倍精度 — 精度の高い — 実数)、などがある。一方、文字列などは複雑な構造を持つ (中に文字がたくさん詰まっている) のでオブジェクトで表す。ところが困ったことに、整数や文字などを表すオブジェクトも別途ある。これは「値」の方はデータの変換などの込み入った機能が入れられないため。そのような値とクラスを次に示しておく。クラス名は大文字で始まることに注意。

種別	値	クラス
真偽値	`boolean`	`Boolean`
整数	`int`	`Integer`
文字	`char`	`Character`
実数	`float`	`Float`
倍精度実数	`double`	`Double`

あと、先のプログラムでてて来たクラスについても説明しておく。

クラス	説明
`String`	文字列 (文字のならび)
`BufferedReader`	1 行入力機能を持った入力ストリーム
`StreamInputReader`	1 文字入力機能を持った入力ストリーム
`InputStream`	バイト単位入力ストリーム
`PrintStream`	画面表示用ストリーム

最後の 2 つはプログラムの上でその名前が出てこなかったが、実は `System.in` は `InputStream` オブジェクト、`System.out` は `PrintStream` オブジェクトを表している。

オブジェクトを作るには特別な演算 `new` を使って

```
new クラス名 (...)
```

とする。かっこの中は空っぽのこともあるし、作るに当って必要なデータを渡すこともある。

オブジェクトは値にない特徴として、「メソッド」を持てる。たとえば `InputStream` オブジェクトの `read()` メソッドを使うと入力元から 1 文字読み込んでその文字を返してもらうことができる。また `StringBuffer` オブジェクトの `append()` メソッドを使うと領域に文字を追加することができる。

これらの解説の上で、ようやく残っていた説明ができる。

```
double f = (new Double(in.readLine())).doubleValue();
```

ここでは、まず

```
in.readLine()
```

で、BufferedReader オブジェクトのメソッド readLine() を呼び出す。このメソッドは入力先 (この場合はキーボード) から 1 行ぶんの文字を読み、それらをまとめた String オブジェクト (文字列) を返す。次に

```
new Double(...)
```

で、読み込んだ String オブジェクトが表現しているのと同じ値を持つ Double オブジェクトを作る。次に

```
(...).doubleValue()
```

を呼ぶことで Double オブジェクトが表している倍精度実数の「値」を double(小文字!) 型の値として取り出す。そして

```
double f = ...
```

でその値を変数 f に格納するわけである。このように、プログラミング言語では計算機に対するさまざまな/多様な指令をけっこう「ぎゅっと」詰まった形で指定することができる。

6 枝分かれ

さて、ここまでに説明した機能だけでは、上から順に動作を実行していくようなプログラムしか作れない。それでは不便なので、枝分かれを学ぶ。たとえば、「入力 x の絶対値を計算する」例題を考えよう。しばらくの間、条件としては数値の大小比較 (> — より大、>= — 以上、< — より小、<= — 以下、== — 等しい、!= — 等しくない) だけを考える。

- 数値 x を入力する。
- もし $x < 0$ ならば、
- 　　$abs \leftarrow -x$。
- そうでなければ、
- 　　$abs \leftarrow x$。
- 枝分かれ終わり。
- 数値 x とその絶対値 abs を出力する。

なお、「←」は変数に値を入れることを意味する (「〜と置く」と書くのが面倒だから)。ではこれを Java にしてみよう。

```
import java.io.*;

class Sample2 {
  public static void main(String args[]) throws Exception {
    BufferedReader in = new BufferedReader(new InputStreamReader(System.in));
    System.out.print("value X = "); System.out.flush();
    double x = (new Double(in.readLine())).doubleValue();
    double abs;
    if(x < 0.0) {
      abs = -x;
```

```
    } else {
      abs = x;
    }
    System.out.println("value: " + x + " absolute value: " + abs);
  }
}
```

変数 abs を宣言だけしているのが目立つ。これは、変数の宣言を { ... } の中に書くとその範囲内でしか有効にならないので、if の外側に書かなければいけないのでしかたがない。

ところで、同じプログラムだがこう書いたらどうだろう?

- 数値 x を入力する。
- $abs \leftarrow x$。
- もし $x < 0$ ならば、
- $abs \leftarrow -x$。
- 枝分かれ終わり。
- 数値 x とその絶対値 abs を出力する。

このように、「そうでなければ」の部分で何もすることがなければ「そうでなければ」以下を書かないでよい。Java プログラムでも同様。

```
import java.io.*;

class Sample2b {
  public static void main(String args[]) throws Exception {
    BufferedReader in =
      new BufferedReader(new InputStreamReader(System.in));
    System.out.print("value X = "); System.out.flush();
    double x = (new Double(in.readLine())).doubleValue();
    double abs = x;
    if(x < 0.0) {
      abs = -x;
    }
    System.out.println("value: " + x + " absolute value: " + abs);
  }
}
```

先のプログラム例と、どちらが好みですか?

ところで、「わざわざ変数 abs を用意しなくても、変数 x の符号を反転させたら?」と思ったかも知れない。確かに、このプログラムの場合はそれでもよいけれど、場合によっては後で「元の値」と「絶対値」の両方が必要になるかも知れない。一般に、1 つの変数を複数の意味 (「x」と「x の絶対値」) で使うのは混乱する (書く人も、読む人も) からやめた方がよい。ただし、x はすぐに絶対値に直して、あとは絶対値だけしか使わない、ということなら x を直接書き換えてもよい。

要するに、プログラムの書き方は「どれが絶対正解」ということはなく、場面ごとに何がよいかが違ってくるし、人によっても基準が違うところがある。早く自分の基準を見つけよう!

演習 7 枝分かれを用いて、次の動作をする Java プログラムを作成せよ。

a. 2 つの異なる実数 a、b を入力し、より大きい方を表示する。
b. 3 つの異なる実数 a、b、c を入力し、最も大きいものを表示する。
c. 実数を 1 つ入力し、それが正なら「positive」、負なら「negative」、零なら「zero」と表示する。

7　繰り返し

枝分かれができても、まだプログラムで大したことができる気がしない。計算機の特徴は、どんなつまらない単純作業でもいくらでも繰り返してくれることにあるので、繰り返しが使えるとプログラムもぐっと役に立つ。先の箱詰め問題をプログラムにしてみよう。まず疑似コードを再掲する。

- ボールの個数 n を入力する。
- 箱の一辺 l を 0 とおく。
- $l^2 < n$ が成り立つ間繰り返し:
- 　$l+1$ をあらたに l とおく。
- 以上を繰り返す。
- 一辺の長さ l を出力する。

これを Java にしてみよう。

```
import java.io.*;

class Sample3 {
  public static void main(String args[]) throws Exception {
    BufferedReader in = new BufferedReader(new InputStreamReader(System.in));
    System.out.print("value N = "); System.out.flush();
    int n = (new Integer(in.readLine())).intValue();
    int l = 0;
    while(l*l < n) {
      l = l + 1;
    }
    System.out.println("# of balls: " + n + " edge length: " + l);
  }
}
```

書き方そのものは繰り返しの方が枝分かれより簡単なくらいだが、「繰り返す」という考え方がマスターできないとよく分からないだろう。なお、ここでは n も l も整数なので `int` 型と指定し、入力の変換にも `Integer` クラスと `intValue()` を使っている。

演習 8 次の動作を行う Java プログラムを作成せよ。

a. 正の整数 x を入力し、x が奇数か偶数かを表示する。ただし割り算演算は使わないこと。
b. 正の整数 x と y を入力し、$x \times y$ を表示する。ただし掛け算演算は使わないこと。
c. 正の整数 n を入力し、2^n を表示する。

8　制御構造の組み合わせ

少し込み入ったプログラムになると、ある制御構造 (枝分かれ、繰り返し) の内側にさらに制御構造を入れることになる。たとえば、

- もし～であれば、
- 　条件～が成り立つ間繰り返し:
- 　　○○をする

- 　以上を繰り返し。
- 枝分かれ終わり。

だと次のようになるわけである。

```
if( ... ) {
    while( ... ) {
        ...
    }
}
```

このように規則に従って要素を組み合わせて行くことで (単に並べるのも組み合わせ方のうち)、いくらでも複雑なプログラムが作成できる。これはちょうど、簡単な規則と単語からいくらでも複雑な文章が (日本語や英語で) 作れるのと同じである。

演習 9 次の疑似コードは 2 つの数の最大公約数を求めるものである。これを Java に直して動かせ。またなぜこれで最大公約数が求まるのかを説明せよ。

- 整数 x、y を入力する。
- $x! = y$ の間繰り返し、
- 　もし $x > y$ であれば、
- 　　$x \leftarrow x - y$。
- 　そうでなければ、
- 　　$y \leftarrow y - x$。
- 　枝分かれ終わり。
- ここまで繰り返し。
- x を最大公約数として出力する。

演習 10 次の疑似コードは入力 x の平方根を求めるものである。これを Java に直して動かせ。またなぜこれで平方根が求まるのかを説明せよ。

- 実数 x を入力。
- $a \leftarrow 0.0$、$b \leftarrow x$。
- $(b - a) > 0.0000001$ である間繰り返し、
- 　$c \leftarrow \frac{a+b}{2}$
- 　もし $c^2 > x$ ならば、
- 　　$b \leftarrow c$
- 　そうでなければ、
- 　　$a \leftarrow c$
- 　枝分かれ終わり。
- ここまで繰り返し。
- a の値を平方根として出力。