情報処理 (10)

プログラミング

増原 英彦

1 プログラミングの基本

プログラムとは計算機が行うべき処理の手順である。特に、計算機の CPU に対する直接的な命令を並べたものは**機械語プログラム**と呼ばれる。機械語プログラムは、メモリの読み書きや単純な演算、条件判断などから成っている。

人間がプログラムを作るときは、直接機械語プログラムを書かずに、**高級言語**と呼ばれることばを使うのが普通である。高級言語には様々なものがあり、用途によって使い分けられている。FORTRAN, COBOL, Lisp, BASIC, Pascal, C, Prolog, C++, Perl, Java などは、広く使われている (いた) 高級言語である。

人間が会話や読み書きに使っている言語 (区別のために「**自然言語**」と呼ばれる) と同様に、プログラム**言語**には**文法**と**意味解釈**がある。自然言語と違い、少しでも文法が間違っているプログラムは正しく動かない。

高級言語は人間が読み書きするためのことばなので、人間にとっては分かりやすいが、計算機の CPU が直接実行することはできない。そこで、高級言語で書かれたプログラム (**原始プログラム**とよぶ) から、計算機の CPU が実行できる命令に置きかえたプログラム (機械語プログラム) を作り (この作業を**コンパイル**という)、CPU には機械語プログラムを実行させる。

> 別のやり方として、高級言語で書かれたプログラムを直接実行できるような《賢い CPU》を考え、《賢い CPU》の真似をするプログラムを実行させる方法もある。この方法は**インタプリタ実行**といわれる。コンパイルしてから実行する方法と較べると、この方法は間接的なため実行が遅くなる。そのため、本格的なアプリケーションプログラムを作る際にはあまり用いられていない。

Java 言語は**命令型**の高級言語といわれる。この種のプログラムは、

- 命令が**順番に実行される**
- **変数**によって計算の結果を覚えておく
- 計算の結果によって、その実行する命令を変える (**条件分岐**)
- ひとまとまりの命令を**繰り返し**実行する

といった動作が基本であり、これらの組み合わせで大きなプログラムが作られている。これらの動作は CPU の動作を人間にとって書きやすい程度に抽象化したものと言える。別のプログラミング言語では、論理式が成り立つかどうかを自動的に調べるものや、数学的な関数の計算を基本とするようなものもある。

2 実習: プログラムの作成と実行

練習 10–1: (コンパイルと実行) HWB「17.2. プログラミングの基本」に従って、プログラムを入力・コンパイル・実行させよ。手順は:

1. エディタ (Emacs) でSeconds.javaというファイルを作り、HWB に示されているプログラムを入力し (HWB から切り貼りするとよい)、保存する。**ファイル名および内容の大文字・小文字を変えないように注意。**なお、プログラムの先頭は下のようになっているが、下線部 (Seconds) が**プログラム名**であり、ファイル名 (Seconds.java) と一致していなければいけない。

```
class Seconds {
```

2. ターミナルウィンドウから javac Seconds.java (Enter) と入力し、コンパイルする。(**コマンド名に注意。**javac の最後 “c” は「コンパイル (compile)」の “c” だと覚えておくとよい。ファイル名の大文字・小文字にも注意せよ。)

3. プログラムのどこかに誤りがある場合は、メッセージが出力される。その場合は、誤りがある場所を見つけ修正し、2 からやり直す。例えば、下は誤りがある場合のメッセージの例である:

```
Seconds.java:2: シンボルを解決できません。
シンボル: クラス string
場所     : Seconds の クラス
  public static void main(string[] args) {
                          ^
Seconds.java:3: パッケージ system は存在しません。
    system.out.println(60*60*24);
           ^
エラー 2 個
```

   この例のように、誤りはSeconds.java:2などのように「ファイル名:行番号」に続けて種類 (シンボルを解決できません)、その行の中での場所 (^) によって示される。この例はStringとSystemの語頭がそれぞれ小文字になっていたためであるが、必ずしも修正すべき場所を示しているわけではないことに注意。

4. 何もメッセージが出力されなければコンパイルは成功であり、「**プログラム名**.java」というファイルに機械語プログラムが作られている。ls -l Seconds.class (Enter) と入力して作られていることを確認する。

5. ターミナルウィンドウから java Seconds (Enter) と入力し、実行する。2 とは**コマンド名が違う (最後に “c” がない) こと・引数の大文字・小文字に注意。**

練習 10–2: (計算の練習)　以下の計算をするプログラムを作れ。

- 自分が生まれてからの秒数 (n 歳なら「n 年間の秒数」で近似してよい)
- 10,000,000 秒は何週間になるか
- 方程式 $x^2 - 22x - 408 = 0$ の解

  (x の平方根を計算するには Math.sqrt(x) という式を書けばよい。)
- ある打者は 1 試合に 5 回打席に立ち、一回あたり 0.385 の確率でヒットを打つとする。この打者が 1 試合にヒットを打たない確率。(x の n 乗を計算するには Math.pow(x,n) という式を書けばよい。)
- 上の打者が 25 試合連続でヒットを打つ確率。

(以降の*印付きの練習は、興味ある者向けである。)

練習 10–3: (*プログラムの引数) プログラムには引数を渡して実行することができる。図 1 のプログラムをコンパイルして、`java Input 15` Enter のように実行すると、「`15 years have 473040000 seconds.`」のように表示される。これは、プログラムを実行するときにプログラム名 (`Input`) に続けて書いた整数が、プログラムの 3 行目の`int age = Integer.parseInt(args[0]);` によって、`age` という**変数**にセットされるためである。また、このときの下線部 (`0`) は、最初 (0 番目) の引数をセットすることを意味するので、複数の引数を受け取るには、ここを変えたものを追加すればよい。

この例を参考に、練習 10–2 で計算に使う数を引数として指定できるように改造せよ。

なお、実数を引数として受け取るには`double a = new Double(args[0]).doubleValue();` のようにする。

3 計算機による問題の解決

計算機を使って問題を解決する方法は、人間のやり方とは異なることが多い。5 次方程式 $f(x) = 0$ を解くことを例に考えてみよう。

5 次以上の方程式には、解の公式がないことが知られている。人間が 5 次方程式を解く場合、式の対称性などに注意して因数分解を試みるのが 1 つの方法であろう。

計算機によって解く場合には、色々な x について $f(x)$ を計算してみてその値が 0 に近くなるかどうかを調べるというのが基本的な方法である。この方法は愚直に思えるが、数式の計算は非常に速くできるので現実的な方法である。この解き方で使われているのは、

繰返し: 色々な x について試す

条件判断: $f(x)$ が 0 に近いかどうかを判断する

というプログラミングの 2 つの基本的な要素である。

練習 10–4: (繰返しと条件判断) HWB「17.2.4 繰返し」に従って、繰返しをするプログラムを作ってみよ。

練習 10–5: (繰返し) 以下の値を繰返しによって求めるプログラムを作れ

- 1 から 100 までの和 (「まとめて繰返す」の `Powers` を参考に)
- 1 から 20 までの積
- $f(x) = \frac{x^2-3}{2x}$ のときの $f^{10}(1)$ の値 (つまり $f(f(f(f(f(f(f(f(f(f(1)))))))))))$
- $2^0 + 2^1 + \cdots + 2^{10}$ の値 (二重の繰返し)

```
class Input {
  public static void main(String[] args) {
    int age = Integer.parseInt(args[0]);
    System.out.print(age);
    System.out.print(" years have ");
    System.out.print(age*365*24*60*60);
    System.out.println(" seconds.");
  }
}
```

図 1: 引数を受け取るプログラム

練習 10–6: (条件判断) HWB「17.2.8. 二者択一」に従って、判断をするプログラムを作ってみよ。

練習 10–7: (条件判断と繰返し) 次のプログラムは、1 から 10 までの数の組を全て表示させるものである。

```
class XY {
  public static void main(String[] args) {
    int x;
    int y;
    for (x = 1; x <= 10; x=x+1)
      for (y = 1; y <= 10; y=y+1)
        System.out.println(x + "," + y);
  }
}
```

これをもとに $y^3 = x$ を満たしている 1000 以下の整数の組 (x, y) 全てを表示するプログラムを作れ。

4 実際のプログラム

4.1 計算方法の工夫

最初に述べたように、5 次方程式 $f(x) = 0$ を解いたとする。2 千万個の x について $f(x)$ の値を調べたところ、7 秒で計算が終了したとする。

では、同じやり方で 2 元の方程式 $f(x, y) = 0$ を解くとどうなるだろうか? この場合、2 千万個の x, y すべての組み合わせについて $f(x, y)$ の値を調べなければいけなくなるので、7 秒の 2 千万倍の時間がかかることになる。これは現実的でない。

そこで、問題の性質を使った工夫をして解くことが必要になる。工夫としては、

- ニュートン法のように、おおまかに推定した解の値を使って、実際の解により近い値を計算してゆく
- あらかじめ解が存在しない値の範囲が分かる場合、その範囲の値については調べない

など、問題によって色々なものがある。

4.2 大きなプログラムを作る

大きな問題を解決するためには、沢山の手順を埋まなければいけない。これをプログラムにすると、非常に沢山の手順が延々と続くものになってしまう。このようなプログラムを正しく読み書くことは人間にとっては簡単なことではない。

普通、大きな問題を解決するためには、問題を小さな部分問題に分割し、それぞれを解く。プログラムも同様で、沢山の手順から成る処理を分割し、小さな処理に分けて書くことが行われている。

このような分割された処理の単位は、**関数・手続き・メソッド**などと呼ばれる。(呼び方はプログラム言語によって異なる。) 同様に、データもひとまめにして扱う方法もある。

これらの発展として、こんにち使われている高級言語では、**クラス・オブジェクト**や、**モジュール**を単位としてプログラムを分割して書くことが一般的に行われている。

5 プログラムの誤りを見つける

5.1 コンパイルしたときの誤り

ターミナルウィンドウで javac プログラム名.java Enter としてコンパイルしたときに、何かメッセージが出た場合は、プログラムに誤りがある場合である。(逆にメッセージが出ない場合は無事にコンパイルができたときである。)

典型的なものには次のようなものがあるが、メッセージは状況によって違うので、一般的な対処法は「メッセージを読んで間違った場所を見つけ、その周辺について考える」というものになる。

- ファイル名を間違えた— 「ls プログラム名.java」というファイルが保存されているか?
- ファイルの保存を忘れた — 誤りを直した後で、保存をせずにもう一度コンパイルしたら、誤りを直す前のファイルが使われる
- プログラム中の大文字・小文字の間違い
- 「;」の書き忘れ
- つづりの間違い

5.2 実行したときの誤り

コンパイルに成功しても期待通りにプログラム動かないこともある。

ターミナルウィンドウでjava プログラム名Enter と入力したときに「クラスが見つからない」旨のメッセージが表示されたら、プログラム名.class というファイルができない場合である。プログラム名を間違えていないか、プログラム名.class ができているかを確認する。

誤ったプログラムには、いつまでも終了しないものもある。そのようなプログラムは Control c をタイプすると強制的に停止させることができる。

また、「プログラムは動くのだが計算結果がおかしい」ということもある。そのような場合、途中経過を表示させ、どこでおかしくなったかを調べる。例えば、1 から 10 までの積を計算するのに下のようなプログラムを作ったとしよう。

```
class Fact {
  public static void main(String[] args) {
    int p = 1;
    int i;
    for (i = 0; i <= 10; i=i+1)
      p = p * i;
    System.out.println("1*2*...*10=" + p);
  }
}
```

このプログラムを実行すると1*2*...*10=0 のように間違った答え表示される。そこで、下のように各所に計算の途中経過を示すように変更して実行する。

```
class Fact {
  public static void main(String[] args) {
    int p = 1;
    int i;
    for (i = 0; i <= 10; i=i+1) {
      System.out.println("i=" + i + ", p=" + p);
      p = p * i;
    }
    System.out.println("i=" + i  + ", p=" + p);
    System.out.println("1*2*...*10=" + p);
  }
}
```

すると、下のよう表示され、i が 1 のときからp の値がおかしくなっていることが分かる。

```
1*2*...*10=0
i=0, p=1
i=1, p=0
...
```

練習 10–8: (実行結果をファイルに保存する) 誤りがある場合に限らず、プログラムの実行結果をファイルに保存することは有用である。以下のようにして実行結果をファイルに保存せよ。

1. (例えば) ABC という名前のプログラムを ABC.java というファイル作成し、コンパイルする。
2. ターミナルウィンドウで「`java ABC` Enter」と入力し、実行できることを確認する。
3. ターミナルウィンドウで「`java ABC | tee ABC.txt` Enter」と入力する。実行したときのメッセージが画面に表示されるとともに ABC.txt という名前のファイルに保存される。
4. エディタ (Emacs) で 3 で保存したファイル ABC.txt を読み込み内容を確認する。

6 おまけ: 練習 10–2 の解答例

```
class Calculation {
  public static void main(String args[]) {
    System.out.print  ("Seconds of 20 years = ");
    System.out.println(20*365*24*60*60);

    System.out.print  ("Weeks of 10000000 seconds = ");
    System.out.println(10000000/60/60/24/7);

    System.out.print  ("Solution of x^2-22x-408=0 is x=");
    System.out.print  ((22 - Math.sqrt(22*22-4*(-408)))/2);
    System.out.print  (", ");
    System.out.println((22 + Math.sqrt(22*22-4*(-408)))/2);

    System.out.print  ("Probability of no-hitting game=");
    System.out.println( Math.pow(1-0.385,5) );

    System.out.print  ("Probability of 25 hitting games in a row=");
    System.out.println( Math.pow(1-Math.pow(1-0.385,5), 25) );
  }
}
```