計算機プログラミング I (3)

代入・繰り返し・分岐

2000 年 10 月 24 日 増原 英彦

前回作った「10 年後の預金残高を求める」プログラムでは、年数は固定されていた。この年数を後から簡単に変えられるようにしてみよう。そのためには、(1) 同じ処理を年数分繰り返す方法と、(2) 今年の残高から来年の残高を計算する方法があればよい。注意すべきは、処理が進むと「来年の残高」が「今年の残高」にならなければいけない点である。

1 代入

これまで変数は「値に名前を付けたもの」と説明していたが、実はプログラムの進行とともに値を変化させることができる。これを行うのが代入文である。これによって、「10 年後の預金残高を求める」プログラムは以下のようになる:

```
1  double balance = 100000;          // 最初の預金額
2  double interest = 1.0045;         // 利率
3  System.out.println("現在の残高は" + balance + "円");
4  balance = balance * interest;     // 1 年後の残高を計算する
5  balance = balance * interest;     // 2 年後の残高を計算する
6  balance = balance * interest;     // 3 年後の残高を計算する
7  ...(中略)...
8  balance = balance * interest;     // 10 年後の残高を計算する
9  System.out.println("10 年後の残高は" + balance + "円");
```

ここでは、4 ~ 8 行目が代入文であり、「現在のbalance とinterest の値を掛けたものを、以降のbalance の値とせよ。」という意味である。4 行目を実行する前は、**balance** には最初の残高があり、実行した後は 1 年後の残高が入っている。さらに、5 行目を実行した後は 2 年後の残高が……のように、1 行実行するごとに一年分の計算が進む。

2 一定回数の繰り返し

2.1 for 文

上のプログラムでは、各年の計算が全く同じである。回数を指定して計算を繰り返すことができれば、望みの年数後の残高を計算するプログラムを容易に作ることができるだろう。

Java 言語には、繰り返しを行うための構文としてfor 文がある。それを使って書くと、図 1 のようになる。(⇒ for 文の文法と、詳しい意味はテキストを参照せよ。) このようにすれば、years の値を変えるだけで、望みの年数後の残高を求めることができる。

練習 3-1: (2 進数の変換) 2 進数を文字列で与えられているとき、その各桁の値は以下のようにして扱う[1]ことができる。(上位の桁から表示されることに注意。)

[1] binaryNumber.length() は、文字列binaryNumber に含まれる文字数を調べる。また、binaryNumber.charAt(index) は文字列binaryNumber の左からindex 番目の文字を調べる。(先頭は「0 番目」になることに注意。) そこから '0' を引くと、文字コードどうしの引き算が行われ、「1」は 1 に、「0」は 0 になる。

```
1  double interest = 1.0045;        // 利率
2  int years = 10;                  // 計算する期間
3  int currentYear;                 // 何年後かを数える変数
4  double balance = 100000;         // currentYear 年目の残高
5  // 1 年後から years 年後までを順に実行
6  for (currentYear = 1; currentYear <= years; currentYear = currentYear + 1)
7      balance = balance * interest; // これが years 回実行される
8  System.out.println(balance);     // years 年後の残高
```

図 1: 繰り返しによる預金の計算

```
1  String binaryNumber = "1101"; // 文字列で表わした 2 進数
2  int numberOfDigits = binaryNumber.length(); // ビット数
3  int index; // 現在注目している桁数をしまう変数
4  for (index = 0; index < numberOfDigits; index = index + 1) {
5    int bit = binaryNumber.charAt(index) - '0';// index 桁目のビット
6    System.out.println("第"+ index + "桁目の値は" + bit + "です。");
7  }
```

これを応用して、文字列として与えられた 2 進数から 10 進数を計算するプログラムを作れ。($\Rightarrow$ 4,7 行目の波かっこ ({ }) の意味については 2.2 節を参照。)

練習 **3-2:** (数値誤差) for 文を使って一定回数の繰り返しを行う場合、図図 1 の 6 行目のcurrentYear <= years のように、条件を大小関係で書くのが普通である。これを例えば下のように、「等しくない」ことを条件にして実行するとどうなるか? またその理由は?

```
double year; // 現在の年数
for (year = 1; year != 10; year = year + 0.1) // 10 分の 1 年刻みごとの繰返し
  System.out.println("現在の年数は"+year+"です。");
```

2.2 文とブロック

もし「一年に 0.1%ずつ金利が増えてゆく預金」があった場合、10 年後の預金高はどうなっているだろうか? この場合、上のプログラムを少しだけ変えて、毎年 balance とinterest の両方を代入文を使って増やせばよいだろう。しかし、for 文で繰り返されるのは一つの文だけである。そのため、

```
1  // 1 年後から 10 年後までを順に実行
2  for (currentYear = 1; currentYear <= years; currentYear = currentYear + 1)
3      balance = balance * interest; // これが years 回実行される
4      interest = interest + 0.001;  // 毎年金利を増やす? (間違い)
```

と書いても、繰り返されるのは 3 行目だけである[2]。

Java 言語には、{ }を使って、複数の文を一つの文として扱わせる機能がある。囲まれたものをブロックという。これを使って下のように書くと、めでたく 3–6 行目がfor 文で繰り返される部分になる。

```
1  // 1 年後から years 年後までを順に実行
2  for (currentYear = 1; currentYear <= years; currentYear = currentYear + 1)
3  {                                // ブロックの開始
4      balance = balance * interest; // 残高を増やす
5      interest = interest * 0.001;  // 金利を増やす
6  }                                // ブロックの終了
```

練習 **3-3:** (二重の繰り返し) 金利が 0.1%, 0.2%, ..., 1.0% の場合について、それぞれ 10 万円を 10 年間預けたときの残高を計算して表示させよ。(ヒント: for 文の中にfor 文を書く。)

[2]Java 言語では、行の先頭に空白を入れるのは見易さのためだけであり、プログラムの実行には全く関係がないことに注意。

3　不定回の繰り返し

3.1　while 文

「10 万円を何年間預けたら 20 万円以上になるか?」という問題を解くことを考える。いままでとの違いは、繰り返しの回数が予め分かっていない点である。この場合、条件を満たしている間繰り返しを続けるwhile 文を使って図 2 のように書くことができる。(⇒ while 文の文法と意味についてはテキスト参照。)

```
1  double interest = 1.0045;                 // 利率
2  double principal = 100000;                // 元金
3  double targetBalance = 2 * principal;     // 目標残高
4  int year = 0;                             // 年数を数える変数
5  double balance = principal;               // year 年後の残高
6  while (balance < targetBalance) {         // 目標に達するまで繰り返す
7      balance = balance * interest;         // 次年度の残高を計算
8      year = year + 1;
9  }
10 System.out.println(year + "年後に残高が" + balance + "円になります。");
```

図 2: while 文による繰り返し

練習 3-4: (for, while)　for 文は、while 文に「繰り返しの最初に行う計算」と「繰り返しを続ける際に行う計算」が書けるようにしたものと見なせる。いま、

$$\texttt{for}\ (\boxed{\texttt{A}};\ \boxed{\texttt{B}};\ \boxed{\texttt{C}})\ \{\ \boxed{\texttt{D}};\ \boxed{\texttt{E}};\ \cdots\ \boxed{\texttt{Z}};\ \}$$

というfor 文があったとき、これと同じ計算をするwhile 文はどうなるかを示せ。また、図 1 をwhile 文を使って書き直せ。

練習 3-5: (do ~ while 文)　テキストのdo-while 文の説明を読み、上の練習にならってwhile 文への書き換えを考えよ。

練習 3-6: (二重の繰り返し 2)　10 年後に元金が 2 倍になる最低の金利を求めたい。金利を 0.1%, 0.2%, ... と 0.1% 刻みで変化させ、10 年後の残高が 2 倍未満の間繰り返すことで求めよ。(ヒント: do ~ while 文を使った方がすっきりと書けるかも知れない。)

3.2　不変量

繰返しのあるプログラムは、状態が刻々と変わってゆくため間違いやすい。プログラムが問題を正しく解いていることを確認するためには、不変量(invariant) という考え方が有効である。

不変量とは「繰返しの間、常に一定である値や関係」のことを指す。例えば、図 2 の金利計算であれば、残高 (balance) は現在計算中の年数 (year) 後の残高なので、元金 (principal) に金利 (interest) の年数乗を掛けたものという関係が常に成立しているので、

$$\texttt{balance} = \texttt{principal} \cdot \texttt{interest}^{\texttt{year}}$$

という関係が不変量である。図 2 のプログラムの繰り返しが、この関係を常に満たしていることは、以下のようにして確められる。まず、プログラムの 4,5 行目から、繰り返しを始める前にはこの関係が成立している。

次に、何回繰返しをまわってもこの関係が満たされ続けていることは次のようにして示す。まず、上の関係が成り立っているときに 7,8 行目を実行した後のbalance, year の値を $\mathtt{year}'$, $\mathtt{balance}'$ とする。これらの値は、それぞれ $\mathtt{year}' = \mathtt{year} + 1, \quad \mathtt{balance}' = \mathtt{balance} \cdot \mathtt{interest}$ になる。このとき

$$\mathtt{balance}' = \mathtt{principal} \cdot \mathtt{interest}^{\mathtt{year}'}$$

であるので (簡単に示せる)、「繰り返しを一回まわっても不変量が変わっていない」ことが言える。あとは帰納的に繰り返しを何回まわっても満たされている。

練習 **3-7:** (不変量)　練習 3-1 で作ったプログラム中の繰り返しの不変量は何かを考え、正しいことを確認せよ。

4　条件分岐

4.1　if 文

定期預金では、預ける期間によって金利が変わるのが普通である。例えば、10 年以上預ける場合の金利は 0.45% でそれ未満の場合は 0.3% だったとする。このような場合に、預ける年数から金利を自動的に判断して、その年数が経過した後の残高を表示させるようにするにはどうすればよいだろうか。

これには、ある条件が成立した場合と成立しなかった場合で違った動作をさせるif 文を使う。すると、以下のようなプログラムになる:

```
1  double longInterest = 1.0045;    // 10 年以上の金利
2  double shortInterest = 1.003;    // 10 年未満の金利
3  int years = 10;                  // 預ける期間
4  double interest;                 // 預ける期間に対応した金利
5  if (years >= 10)
6      interest = longInterest;     // 10 年以上の場合
7  else
8      interest = shortInterest;    // 10 年未満の場合
9  // 以下、図 1 の 3 行目以降と同じ
```

(⇒ if 文の文法についてはテキスト参照。)

練習 **3-8:** (複雑な条件)　上のプログラムを拡張して、金利が、10 年以上のとき 0.45%, 7 年以上のとき 0.4%, 5 年以上のとき 0.3%, 4 年以上のとき 0.25%, 3 年以上のとき 0.23%, それ未満のとき 0.2% で計算するようにプログラムを直し、1 年間 ~ 10 年間預けた後の残高を計算してみよ。

4.2　条件の組み合わせ

何らかの判断をして動作を変えるプログラムがあったとしても、値が等しい/大きい/小さいという比較の組み合わせだけで全ての判断を行っている。これらの比較を組み合わせるには、(1) 条件式を && (かつ) / || (または) / ! (否定) で組み合わせる方法と、(2) if 文の中にif 文を書く方法がある。

練習 **3-9:** (条件の組み合わせ)　ある年year が閏年かどうかのメッセージを表示するプログラムを (1)1 つのif 文の条件式を&& (かつ) / || (または) / ! (否定) の組み合わせによって、(2) 値の比較だけをする条件式を持つif 文の組み合わせでそれぞれ作れ。ただし、西暦年が 4 の倍数でない年、または西暦年が 100 の倍数かつ 400 の倍数でない年は平年であり、それ以外の年は閏年である。