1 計算

コンピュータは長い間「計算機」と訳されてきました。現在でこそコンピュータは計算以外の非常に幅広い目的で用いられていますが、コンピュータの仕組みが「計算をする機械」であることには現在も変わりありません。本章ではプログラミングの最も基礎的な技術として、計算に関するプログラミング方法について説明します。

1.1 【サンプルプログラム】BMI の算出

早速ですが簡単なサンプルプログラムとして、BMI を算出するプログラムを紹介します。BMI とは人間の体型を評価する値[1]で、

$$BMI = (\text{体重 } [kg])/(\text{身長 } [m])^2$$

によって算出されます[2]。
以下に示すプログラムでは、

- double 型と呼ばれる型（後述）の変数 weight に体重（67.0kg）
- 同じく、double 型の変数 height に身長（1.78m）

が、それぞれ代入されています。そして

- double 型の変数 bmi に、weight / (height * height) の算出結果

が代入されます。そして最後、「BMI=」の後に、変数 bmi に代入された値が表示されます。

```
public class Bmi {
   public static void main(String[] args) {
     double weight = 67.0;
     double height = 1.78;
     double bmi = weight / (height * height);
     System.out.println("BMI=" + bmi);
   }
}
```

それでは、上記のプログラムをテキストエディタで打ち込み、Bmi.java というファイル名で保存して下さい。そして、ターミナル上で以下の 2 個のコマンドを実行してみて下さい。

```
javac Bmi.java
java Bmi
```

以下のような表示結果が表示されれば、正常に動作しているといえましょう。

```
BMI=21.146319909102385
```

[1] 22 が標準値で、大きく下回ると体重過小、大きく上回ると体重過多、と言われています。
[2] 身長の単位はセンチメートルではなくメートルであることに注意。

また、変数 weight および height に別の値を代入して実行したときに、表示結果である BMI の値も変化することを、あわせて確認してください。

このような計算を実施するプログラムを、本章にて解説いたします。

1.2 文字列と数式

さて、本章では Java 言語を用いた計算について解説します。まずは以下のプログラムをつくり、Calculate.java というファイル名で保存して下さい。

```
public class Calculate {
   public static void main(String[] args) {
     System.out.println("3+5");
   }
}
```

まず 1 行目の public class Calculate { について説明します。w Java では **class(クラス)** をプログラムの最小単位として考えます。この 1 行目は、他のプログラムにも公開される（=public な）クラスとして、その名前を Calculate と定義しています。

続いて 2 行目の public static void main(String[] args) { ですが、細かい定義は後述するとして、Java では main と書かれたところからプログラムの実行を開始することになっています。つまり、この行はプログラム実行開始の目印を示しています。

続いて 3 行目の System.out.println(”3＋5”); ですが、これも細かい定義は後述するとして、System.out.println の次の () の中の文字列を画面に表示せよ、という処理命令が記述されています。

続いて 4 行目の } ですが、これは 2 行目の行末の { を閉じる役割をしています。5 行目の } も同様に、1 行目の行末の { を閉じる役割をしています。Java 言語や C 言語などのプログラム言語では一般的に、{ と } で囲まれたカッコ の中を、処理の一ブロックとして扱います。そしてカッコが二重以上の入れ子を構成する場合には、数式と同様に、内側のカッコどうしが対応関係をもち、外側のカッコどうしが対応関係をもちます。

では、ターミナル上で以下の 2 個のコマンドを実行してみて下さい。

```
javac Calculate.java
java Calculate
```

1 行目の javac Calculate.java という命令は、Calculate というクラスをコンパイルしなさい、という意味を持ちます。2 行目の java Calculate という命令は、Calculate というクラスの中にある main から出発してプログラムを実行しなさい、という意味を持ちます。

ここで注意すべき点は、以下の通りです。最初は紛らわしいかもしれませんので、よく理解して下さい。

- **プログラムを修正したときには javac を実行する必要があるが、修正していないときには java だけを実行すればプログラムを繰り返し実行できる。**
- **javac のあとには、クラス名ではなくファイル名を書く。よって上記の例の場合には、javac のあとは Calculate.java となる。**
- **java のあとには、ファイル名ではなくクラス名を書く。よって上記の例の場合には、java のあとは Calculate となる。**

さて、以上を実行して何が表示されましたでしょうか？ おそらく皆さんの画面では、

```
3+5
```

と表示されたことと思います。これはJava言語では、2個の¨(**引用符** または **ダブルクオーテーション**という）で囲まれた文字群を、一種の文として扱う、というように規定するからです。

このように文として扱われた文字群を、**文字列** といいます。上記のプログラムでは、「3+5」を文字列として扱います。このとき計算機は、この数式を計算させる、というようには解釈しません。

では、この引用符を消して、以下のようにプログラムを書き直して下さい。

```
public class Calculate {
   public static void main(String[] args) {
     System.out.println(3+5);
   }
}
```

そして、これを実行すると、何が表示されますでしょうか？ おそらく

```
8
```

と表示されたことと思います。これは言うまでもなく、3+5という数式を計算した結果です。

つまりJava言語では計算機は、引用符で囲まれていない数式は、「計算せよ」と命令されたものと解釈し、その計算結果を扱います。

このようにJava言語では、引用符を使うか否かによって、数式を文字列として扱うか、計算命令として扱うか、を判別します。

1.3 変数と代入

では今度は、Calculate.javaを以下のように書き換えてみましょう。

```
public class Calculate {
   public static void main(String[] args) {
     int i;
     i=3+5;
     System.out.println(i);
   }
}
```

そして、これを実行すると、何が表示されますでしょうか？ おそらく

```
8
```

と表示されたことと思います。つまり前節と同様に、3+5 という加算を実行した結果が表示されることでしょう。

まず 3 行目の「int i;」について説明します。この行は、i という変数を使うことを意味します。そして int は、この i という変数が整数であることを意味します。このように Java 言語（に限らず多くのプログラム言語）では、この行以降に用いられる変数の名前と性質を、予め明記する必要があります。この明記を **宣言** と呼びます。

続いて 4 行目ですが、これは右辺である 3+5 を算出した結果を i に代入しています。このように Java 言語（に限らず多くのプログラム言語）では、**右辺の算出結果を左辺の変数に代入する** という形式で、多くの算術演算を記述します。この習慣は、見慣れるまで違和感があるかもしれませんが、早く慣れて下さい。

続いて 5 行目ですが、System.out.println(i); は i の値を画面表示せよ、という意味を持ちます。4 行目の演算にて i には 3+5 の計算結果が代入されていますので、ここで 8 が画面表示される、というわけです。

ここで 5 行目を

```
System.out.println("i");
```

と書き換えて実行したら、何が表示されますでしょうか？もうおわかりですよね。ここでは

```
i
```

と表示されることと思います。i を引用符で括った場合には、この i は文として扱われ、変数としては扱われません。

1.4 変数の型

前節では、i という変数を用いること、そして i は整数であることを宣言する必要がある、という旨を説明しました。Java 言語（に限らず多くのプログラム言語）では、「整数」「小数」といった変数の性質を **型** と呼びます。

Java 言語での計算に必要な最小限の型の種類を、表 1 に紹介します。見ておわかりのように、同じ整数の型にも、byte, short, int, long と実に 4 種類の型があります。一般的には、計算機の規模や、計算機に処理させたい計算の規模や精度によって、型を使い分けることになります。本書では大半の場合において、整数に int を利用します。同様に、同じ小数の型にも float, double という 2 種類の型がありますが、本書では大半の場合において、小数に double を利用します。

Java 言語における変数の型について、もっと広く習得したい人は、検索エンジンで調べるか、専門書を購入して調べて下さい。

ところで、表 1 の小数の説明にて、見慣れない数値表記があるのを気がつきましたでしょうか。ここで一例として、3.40282347E+38 という表記に注目して下さい。この表記は、(3.40282347) 掛ける (10 の 38 乗) という意味を持ちます。

この表記において、3.40282347（つまり E の前まで）を**実数部** といい、+38（つまり E の後から）を**指数部** といいます。計算機の内部構造においても、一般的には小数を、実数部と指数部に分けて処理しています。この結果、計算機の内部では、小数点や有効桁数を変動させながら小数を扱うことになります。この仕組みによって計算機内部で扱われる小数を、**浮動小数点数**といいます。

表 1: Java 言語における変数の型。

型の名前	型の説明
boolean	false(偽) または true(真) のいずれか 2 値しかとらない変数。
char	Unicode という規格で定められた文字を表す変数。1 変数が 1 文字に対応する。
byte	8 ビットという単位で表現できる整数をとる変数。 範囲は-128〜127
short	16 ビットという単位で表現できる整数をとる変数。 範囲は-32768〜32767
int	32 ビットという単位で表現できる整数をとる変数。 範囲は-2147483648〜2147483647
long	64 ビットという単位で表現できる整数をとる変数。 範囲は-9223372036854775808〜9223372036854775807
float	32 ビットという単位で表現できる小数をとる変数。 範囲は-3.40282347E+38 〜 -1.40239846E-45 および+1.40239846E-45 〜 +3.40282347E+38
double	64 ビットという単位で表現できる小数をとる変数。 範囲は-1.7976931348623157E+308 〜 -4.94065645841246544E-324 および+4.94065645841246544E-324 〜 +1.7976931348623157E+308

表 2: Java 言語における四則演算の記号。

記号の種類	記号の説明
a + b	a と b を加算する
a - b	a から b を減算する
a * b	a と b を乗算する
a / b	a を b で除算する
a % b	a を b で除算した剰余を求める

1.5 四則演算

前節では加算（3+5）を例にして計算のプログラムを示しましたが、本節では四則演算について説明します。四則演算といえば加算（＋）、減算（−）、乗算（×）、除算（÷）を指しますが、Java 言語（に限らず多くのプログラム言語）では表 2 に示すように、四則演算の一部の記号が算数の教科書などと異なる点に注意して下さい。