人間システム工学科/情報科学科の学生のためのプログラミング入門

岡留　剛

1　はじめに

本稿は，超入門ということで細部にはいっさいこだわっておりません．そのことを念頭に置いて以下をお読みになっていただくとよいと思います．

計算を反省する

計算とメモリ一日の講義が終わって，バスに乗って家に帰ることを考えてみます．今から何分後に家に着くか，それが問題としましょう．バスに乗ってからは 35 分で家に着くとします．

現在時刻と次に発車するバスの時刻との差をとってそれに 35 分を足せばいいわけです．みなさんはこのような簡単な計算なら「無意識」に計算できるので，あまりひっからないと思います．

このような普段なにげなく頭の中でやる計算をちょっと反省しながら深く考えて記述してみますと以下のようになるかと思います．

まず，時計を見て現在時刻を知りそれを頭の片隅に入れておきます．続いてバスの時刻表を見て次のバスの発車時刻を同定します．そして，先ほど覚えておいた現在時刻と差をとり結果を覚えておき，それに 35 分を加えてあげる，ということになるかと思います．

計算機が同じ計算を行なう場合にも同様の手順を踏みます．つまり，まず，計算機がその内部に持っている「時計」から現在時刻を引っ張り出して覚えておきます．続いてやはり計算機の内部に保持しているバスの時刻表から先ほど覚えておいた現在時刻を使って，次のバスの発車時刻を同定します．そして，覚えておいた現在時刻と差をとりその結果を覚えておき，それに 35 分を加えます．

このように計算機では，途中の計算結果や，表から見つけた結果などを「覚えておく」ことが重要となります．計算機には，計算の結果を覚えておく場所があらかじめ用意されており，それをメモリと呼んでいます．

メモリは，非常に多くの情報を蓄えられるだけの容量があります．

2　変数と代入文

では実際に，計算機に上の計算を行なわせることを考えてみます．まず，計算機に計算をさせるためには，計算機が理解でき実行できるように，プログラミング言語という言葉で，計算の手順を書いてやります．その書かれた計算の手順をプログラムと呼びます．計算の結果を覚えておくべきメモリ上の場所はプログラム中では名前を付けてその名前を通して値をしまったり取り出したりできます．

例えば，現在時刻と次のバスの発車時刻との差が 5 分であれば，それをメモリの d という名前の場所に覚えておくためには

```
d = 5;
```

と書きます．等号 (=) は，数学の意味である等式の左辺と右辺が等しい，という意味ではなく，このプログラムを計算機が実行すると，数値 5 が d というメモリ上の名前の場所にしまわれることになります．同様に，この d に覚えておいた値と，35 を加えてその結果をメモリ上の新たな場所 t にしまうには，

```
t = d + 35;
```

と書きます．また，= は上記なような意味なので，結果を新たな場所ではなくもとの d に上書きすることもできます．

```
d = d + 35;
```

です．これを数学の式とみるとまったく奇妙ですね．

ここで出てきた

```
d = 5;
```

や

```
t = d + 35;
```

などは，プログラムの中の**文**と呼ばれ，一番最後についているセミコロン (;) は，文の終わりを表わしている重要な記号です．ちなみに，これがピリオド (.) やらコロン (:) などと間違えてしまうと，まったく違った計算結果となるか，単に計算機に,「そんな文はない」と怒られて計算してくれないかです．等号記号により，計算結果を変数にしまう文を**代入文**と言います．

では，上の説明で飛ばした「計算機内の時計の値を取り出す」ということと,「計算機内に蓄えてあるバスの時刻表を見て次のバスの時刻を同定する」といったことはどうやるのか，とお思いになった方も多いことと思

います．これは，少々進んだコース向けの話題でして，この超入門では残念ながら割愛します．

さて，名前を付けて数値を覚えておいたり読み出したりするメモリ上の場所の名前を**変数**と言います．これも，数学で出てくる変数とは意味がちょっと違います．プログラムではメモリのある特定の場所のことですからね．ただ，プログラムの変数はいろいろな値をそのつどしまうことができるので，その値が変化するという意味で変数と呼んでいるのでしょう．

C とか Java とかいったプログラミング言語では，プログラムを書く人は，数値を覚えておくために，これこれの変数を使うぞ，と使う前に高らかに宣言してやる必要があります．また，宣言する際に，その変数で特定されるメモリの場所には整数をしまうのか，小数をしまうのか，はたまた，アルファベットをしまうのか，などをあらかじめ指定しておく必要もあります．例えば，上の例では，

```
int d;
int t;
```

といった具合に変数の前にその変数が整数であることを示す int をつけてやります．英語で整数のことを integer といいます．この単語の始めの 3 文字をとっています．また，小数の例ですと，

```
float f;

f = 3.14;
```

という具合に変数名，この場合は f が小数 (float) であることを表わしています．

条件分岐文とループ文

これまでの例でお分かりのように，プログラムは，変数をいくつか用意しておき，何かを計算しては用意した変数のどれかにしまい，また，計算しては変数にしまい，といったことの列となります．では「計算して」の計算は何が許されるのでしょうか？四則演算はできて当然と言えば当然ですね．先ほど飛ばした計算機内部の時計の値をとるというような「計算」やら，関数の微分や積分をできる，と言えばできるのですが，ベースとしては，四則演算に毛が生えた程度の計算と，今から述べる「条件分岐文」と「ループ文」だけでこつこつとプログラムを書くのだと思ってください．

関数の微積分や方程式の解を求めるといった計算はプログラムの長い歴史の中で誰かが四則演算と条件分岐文・ループ文でもって作ってくれたのでそれを利用すれば計算できるということなのです．

ではまず，**条件分岐文**とはなんでしょうか．簡単に言えば場合分けを伴う計算です．例を挙げましょう．上で述べたバスに乗って帰宅する例を拡張して，例えば，今の時刻から計算して直接家に帰り着くのが 40 分未満なら 10 分だけ本屋に寄ってから帰る，また，40 分以上なら 5 分だけ寄って帰る，とした場合の帰宅にかかる時間を計算するとしましょう．

条件分岐のための if 文というものを使って，愚直に書けば次のようになります.．今からバスが発車する時間が変数 d に入っているとして，

```
t = d + 35;

if (t < 40)
  { t = t + 10; }
else
  { t = t + 5; }
```

if 文というのは，

```
if （条件式）{実行する文１ } else {実行する文２ }
```

という形をしております．その意味ですが，まず (条件式) の条件式を実行して，それが成り立つ場合には実行する文 1 が実行され文 2 は実行されません．逆に条件式が成り立たなければ文 1 は実行されずに文 2 が実行されます．これだけです．

次に，**ループ文**とはどういうものかを説明しましょう．今，1 から 10 までの整数を足し合わせたいとします．

```
int i;

i = 1*2*3*4*5*6*7*8*9*10;
```

と書いてももちろんオーケーです．しかし，10 までででなく，もっと大きい数，例えば，10000 までだと上のような記述では大変な労力です．こんなときループ文を使います．1 から 10 までの数を順にとる変数 i と，足し合わせた途中の結果を保持する変数 r とを用意します．

```
int i; int r = 0;
```

次いで，ループ文です．以下は ループ文の一つを実現する while 文と呼ばれるもので，

```
int r; int i;

r = 0;
i = 1;

while (i <= 10) {
   r = r + i;
   i = i + 1;
}
```

と書き，i の値をチェックしてそれが 10 以下であれば，

```
r + i;
```

を計算して r にしまい，i の値を 1 だけ増やす，ということを i の値が 10 より大きくなるまで繰り返すのです．変数 i は繰り返しの回数を保持しており，また，r は結果を途中結果を含めて保持する変数です．while 文は，

```
while (条件式){文１文２ ． ． ． 文ｎ }
```

の形をしており，(条件式) の条件式を実行してそれが成立するば文 1 から順に文 n までを実行します．そして再度 (条件式) を実行してそれが成立するか否かをチェックし，成立すれば再度同じことを行ないます．これを条件式が成立しなくなるまで繰り返します．

四則演算と条件分岐文とループ文，これだけですべての計算ができるって !? そうなんです，本当なんです．パソコンに乗っている Microsoft Office やら，iPhone や Android 携帯のアプリなどなど，すべて基本的にはこれらから構成された文の集まりなのです．実行したい計算をそれらの列でどう表現するかがプログラミングのだいご味であり，難しいところです．って，そんなの多くの人にとって絶望的ななこともわかりすぎるほどわかります．幸い先人が多くの便利な「関数」を既に作ってくれていますので，われわれはそれを利用すればいいわけです．しかし，先人の構築物を利用して新たな構築物を自ら作る場合には，やはり変数を用意して，ループ文などでプログラムを作る必要があります．その意味で，ループ文などの書き方を学ぶ必要があるのです．

3 プログラミングの片鱗

さてでは，ここで四則演算と条件分岐文・ループ文でどんな計算もできることの片鱗をお見せして話を終わりたいと思います．

ここでは，横軸 0 から 1 までと $y = x^2$ の放物線で囲まれる面積，つまり積分を計算するプログラムをお見せします．数値計算しますのでもちろん近似です．

そもそも曲線と x 軸との間の領域の面積を求める積分の定義を思いだせばそれは，領域を細い短冊（長方形）に分けて，すべての短冊の面積を足し合わせればよい，と思い当たります．これをそのまま，いくつかの変数と，順番に実行されるいくつかの文に直してやります．まず，変数 x として 0 から 1 まで変っていくものを用意します．また，短冊の面積を一つずつ足し合わせた結果を保持する変数 res，さらに，短冊の幅である delta を用意します．x 軸状の x にある短冊の面積は，x*x*delta と書けますね．縦の長さが x*x で，幅が delta ですから．これで後は，x が 0 のところか

ら 1 まで順に，細い短冊の面積を while 文で足し合わせていけばいいのです．以下のようになることがお分かりになるかと思います．

```
float x;
float res;
float delta;

x = 0.0;
res = 0.0;
delta = 0.000001;

while (x < 1.0) {
  res = res + x*x;
  x = x + delta;
}
```

これだけです．条件分岐文すら使っていません．申し上げましたように，愚直にもとめる図形を幅が delta (= 0.000001) の細い短冊に分解して, 0 のほうから 1 まで順に細い短冊の面積を足しています．それが積分の意味ですからね．先ほどの 1 から 10 までの和とたいして変わりませんね．

高校の数学を勉強した方なら求める面積は，x^2 の 0 から 1 までの定積分なので，x^2 の不定積分は $(1/3)x^3$ であり，求める面積は，$(1/3)(1^2-0^2) = 1/3$ なので，なんでわざわざ上のように面倒なことを，と思うかもしれません．しかし，上のプログラムなら，積分が分からない，あるいは積分ができないような関数と x 軸がはさむ部分の面積を求めることができるのです．

より形式的にプログラミングの核心部分は上で述べられたことですべてだと言っても過言ではありません．

ただし，有意味なプログラムを書くためには，例えば計算の結果をユーザーに示すとか，プログラムにキーボードやらマウスを使って数値を与えたい，積分の例で言えば積分の範囲をその都度与えられるようにしたいとか，ですね，あるいは，膨大な過去の遺産を自分のプログラムに組み込みたい，などさまざまなことを学ぶ必要があります．

その中でもまずはプログラミング言語ごとに基本中の基本のお作法がありまして，例えばプログラミング言語 C ですと，main プログラムというのがなければならず，先ほどの面積の例では，main() の後に，{ と } ではさまれた部分に先ほどの文の並びを書きます．

```
#include <stdio.h>

main(){
  float x;
  float res;
  float delta;

  x = 0.0;
```

```
  res = 0.0;
  delta = 0.000001;

  while (x < 1.0) {
    res = res + x*x;
    x = x + delta;
  }
  printf("%f\n", x);
}
```

というようになります．各行の先頭が 2 マスほど空いているのは計算機にはとっては意味がなく，単に人間にとって，その「意味するところ」をわかりやすくするためです．また，下から 2 行目の

```
printf("%f\n", x);
```

は，画面に結果，この場合は x の値を出すための文です．この文に書かれた printf という関数を使うためには，一番最初の行の

```
#include <stdio.h>
```

というおまじないが必要です．この一文は，stdio.h というファイルに，予め定義された普段使う重要な定数の値やら関数やらが定義されています．

関数を呼び出す直前で「関数」という言葉が出てきました．また，先ほど「先人の構築物」を使って新たなものを作り上げていく，と書きました．プログラミング言語 C では，先人の構築物は通常，関数あるいはいくつもの関数からなる関数群といった形で，利用するのが普通です．プログラムにおける関数は，数学の意味での関数と類似しています．ただし，数学の関数のように，何かの入力に対して，一意に決まる出力を出すものももちろん関数ですが，何も入力がないのに何かを出力するもの，あるいは printf のように画面に文字を表示する，といったものも関数として実現されることがあります．

数学的な意味の関数との類似例としては，例えば，数学でよくお目にかかる三角関数 sin や cos などは，プログラミング言語 C でも，やはり，sin, cos として自分のプログラムの中で使うことができます．$\pi/6$ に対する sin の値を求めたいときは，

```
sin(3.141592/6);
```

としてやればその値である 0.5 が返ってきます．この「値が返る」という意味は，例えば，double 型の変数 d を用意しておいて，

```
d = sin(3.141592/6);
```

とすると，d に 0.5 が代入される，ということです．

sin や cos といったあらかじめ C に組み込まれて利用できるもの以外にも，先人が C で作った関数や，後で述べる自分で作った関数も利用することができます．例えば，二つの整数をもらって一つの整数を返す関数 func が利用できるとします．入力が (2, 3) のときの func の値を求めて利用したい場合には，

```
int h;
h = func(2, 3);
```

とプログラム中でしてやると h に func(2, 3) の値が入ることになります．

Web が当たり前の今の世の中では，自分がやりたいことに対して web 検索でもって誰かが既に作成してくれたプログラム（関数）がないかを調べて，それがありかつ利用可能であるなら，それをダウンロードして，自分のプログラムで上のように利用してやるというのが普通です．