情報科教育法II　2003＃10

久野　靖*

2004.1.14

はじめに

あけましておめでとうございます。50 分の模擬授業は大変でしたか? 授業時間内にやっている 20 分の模擬授業だと、多少不手際があってもそれなりに終わってしまいますが、50 分となるとかなり気合いを入れてやらないとうまく行かないことが分かったかと思いますが…

さて、今回は「プログラミング」を取り上げます。普通教科「情報」では「情報 B」についてのみ、アルゴリズム (NOT プログラミング) に関する内容が少し含まれているだけですが、プログラミングは取り上げ方によっては生徒の興味を引くよい題材になると思うので。

1　プログラミング

1.1　普通教科「情報」とプログラミング

基本的に、普通教科「情報」ではプログラミングは扱わない。ただし、「情報 B」では「コンピュータの仕組みと働き」の中でアルゴリズムを扱う (並べ替えや探索などのうち基本的なものにとどめること、となっている) ので、その範囲でプログラムを動かす実習をやることは考えられる。また、「問題のモデル化とコンピュータを活用した解決」の実習でもプログラミングを活用することは考えられる。

もっとも、これらの項目に限らず、コンピュータによる情報処理の特性が一番よく分かるのは実際にプログラムを作って試すことだと思うので、自分としてはもっと多くの場面でプログラミングを活用して欲しいと考えている (たとえば生徒に書かせないまでも、少し手直しさせて動かさせるといった方法で←この授業でもやったような方法)。

また、普通教科「情報」の一環として (指導要領的には「情報 B」のみだが)、コンピュータの仕組みや原理について取り上げるのであれば、単に「こういう原理になっています」というだけでなく、簡単でもプログラミングを取り上げると具体的な原理について納得しやすくなると思う。

1.2　専門教科「情報」とプログラミング

専門教科「情報」は、普通教科「情報」と違い、専門高校 (これまで工業高校、商業高校等があったが、これからは「情報高校」も加わる) における科目として主に採用される。ただし普通高校でも選択科目として開講するところはあると思うので、普通高校の教員になるとしても、必要に応じて教えられるように準備しておくのが望ましい。

これまでも工業高校、商業高校ではプログラミングを学ぶ科目があったわけだが、専門教科「情報」は内容的には、それをさらに広範囲にした「システム設計・管理分野」と、新たに追加された「マルチメディア分野」、および両者の共通部分の基礎・総合をになう「共通分野」の 3 つから成っている (指導要領解説 92p の表参照)。

*筑波大学大学院経営システム科学専攻

このうち、「共通分野」の基礎の一部で計算機の原理・プログラミングの基礎を扱い、「システム設計・管理分野」ではもっと系統的なソフトウェアの設計や開発などを扱う。そして「共通分野」の総合課題として実際にある程度本格的なソフトウェア開発を体験すること「も」選択できる。

そういうわけで、専門教科「情報」のプログラミング関連部分は大学の情報関連学科で学ぶようなプログラミング関連科目のスケールダウン版といえると思う。この部分については皆様もある程度身を持って学んでいるはずなので、それを援用して、自分が教わったように教えればいいとも言える (身についてないという言い訳は却下)。そういうわけでこの部分はこれ以上は扱わない。

1.3 アルゴリズムの取り扱い

プログラミングのためにはアルゴリズムをきちんと考える能力が不可欠だし、アルゴリズムについて理解を深めるにはそれをプログラムにして動かしてみるのが有効。しかし普通教科「情報」では上述のように「アルゴリズムは学ぶがプログラミングは扱わない」という立場なので困る。

自分としてはプログラムを動かさせてみるのがいいと思うが、そうでない代替案も持っておかないと困るかも知れない。アルゴリズムの題材としては指導要領では「基本的な探索や並べ替えなど」となっているが、それ以外でも実習しやすいものがあれば活用するとよい。

計算機を使わないでも、探索や並べ替えは「値を書いたカードを用意しておいて、それを並べる」といった形で実習できる。机の上にカードを並べて動かさせるというのもいいけど、「1 ステップずつ」を意識させるならメモリ役の複数人に 1 枚ずつカードを持たせ、CPU 役の人がそれを操作する、というのもいいかも。

2 模擬授業

今回の模擬授業は「情報 C」p118〜119 だそうです。担当の方はよろしく。

3 プログラミングの実習体験

3.1 言語と処理系の選択

授業でプログラミングを取り上げる場合、どの言語を使い、どのような実行環境を使うかが問題になる。たとえば、C 言語を使う場合、次のような利点がある。

○ 多くのソフトウェア開発に現に使われている言語なので、「学んでおくと役に立つ」という気持ちになる。

○ 言語の実行モデルが裸の CPU に近く、計算機の仕組みを理解する助けになる。

○ コンパイラによる実行なので実行が高速である。

一方、C 言語には次のような弱点もある。

× 実際に役に立つようなプログラムが書けるほど修行するのはかなり大変である。

× ポインタとアドレス計算など、初心者には理解が難しい概念があり、しかも簡単なプログラムもでそれを理解しないと作れない。

× 環境によるがコンパイラが有償で高価だったりして、ちょっと実習してもらうためだけに購入しにくい。

× 型検査などが緩いので、「死んでしまって動かない」プログラムを簡単に作ってしまい、なぜ動かないのか初心者には分からない。

たとえば最後の点だが、次のプログラムのどこが間違っているか、このまま動かすと何が起こるか分かるだろうか?

```
main() {
  int x, y, z;
  printf("input x y : ");
  scanf("%d %d", x, y);
  z = x + y;
  printf("%d + %d = %d\n", x, y, z);
}
```

こうして整理して見ると、C 言語は普通教科「情報」の中で少しだけプログラミングを体験するといった目的にはあまり向いていないことが分かる。

ここで、今回取り上げる JavaScript について同様に検討してみる。まず利点は次の通り。

- ○ 代表的なブラウザに処理系が組み込まれているので購入はもちろんインストールなども不要ですぐ体験できる。
- ○ 型宣言などが不要で動作部分だけを書けばよい。
- ○ HTML と組み合わせることで GUI 部品などが直ちに使える。
- ○ オブジェクト指向言語であり、オブジェクトの機能を活用すればかなり高度な/見栄えのする動作が実現できる。
- ○ メモリ管理などの詳細部分は処理系がカバーしてくれるので気にしなくてもよい。

もちろん弱点もある。

- × インタプリタ実行で速度はやや遅い。
- × ブラウザに組み込まれていてブラウザと一緒でないと使えない (そうでない処理系もあるがあまり一般的でない)。
- × 実行モデルの抽象度が高いので、CPU などの仕組みと対応づけて考えるのには適していない。
- × オブジェクト指向のメカニズムなどをすべて理解するのは大変である。

でもとりあえず、普通教科「情報」などで少しだけプログラミングを経験してもらうのには C 言語よりずっと適していることが、上の比較だけ見てもお分かり頂けることと思う。

一般に、自分が教師として生徒に教える言語を選ぶときは、単に「自分が知っているから」だけでなく (それは知っていないと教えられないから必要なことだけど)、その言語が授業の目的にどれくらい合っているかどうかを上のような形で検討してから選ぶようにして頂きたい。

3.2 HTML のフォームと JavaScript

これまでに HMTL の基本的な機能はひととおり学んだが、フォーム機能については取り上げていなかった。フォーム (書式) とは、記入欄やチェックボックス、ボタンなどが並んだもので、もともとは掲示板など、データやテキストを記入してサーバに送るために使うためのもの。

この授業ではサーバ側のプログラミングは扱わないのでここまでフォームも扱って来なかったが、別の用途として JavaScript でフォームのデータを処理するという使い方もあるので、ここで説明しておく。とりあえず、次の 3 つのタグだけ知っていれば十分。

- `<form>...</form>` — 1 つのフォームの範囲を示す。
- `<input type="text" name="名前" size="文字数">` — 文字の表示/入力欄を表す。

- `<input type="button" value="ラベル" onclick="コード"` — 「ラベル」を表示し、押すと「コード」の部分が実行される押しボタンを表す。

また、JavaScript をページ内に入れる場合、上記の `onclick` 等には短いものしか書けないのでまとまったものを書くために次のタグを使う。

- `<script type="text/javascript">...</script>` — JavaScript コードの範囲を表す。

では具体例をさっそく見てみよう。次の HTML+JavaScript は、2 つの数の足し算を行うもの。

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN">
<html>
<head>
<title>JavaScript Sample</title>
<script type="text/javascript">
function calc() {
  var x = parseFloat(document.forms.f0.elements.x.value);
  var y = parseFloat(document.forms.f0.elements.y.value);
  document.forms.f0.elements.z.value = x + y;
}
</script>
</head>
<body>
<form name="f0" action="#"><p>
<input type="text" name="x" size="5"> +
<input type="text" name="y" size="5"> =
<input type="text" name="z" size="5"><br>
<input type="button" value="計算" onclick="calc()">
</p></form>
</body>
</html>
```

まず後半の方から見てもらうと、3 つの欄 x、y、z と計算させるためのボタンを用意している。部品以外の文字 (「+」とか「=」とか改行タグとか) は通常通り HTML として書けばよいことに注意。ボタンが押されると `calc()` という関数が呼び出されるようにしている。この関数は上の方で`<script>...</script>`の中に定義しているが、その内容は次の通り。

- フォーム部品の値は「`document.forms.` フォーム名`.elements.` 部品名`.value`」でアクセス (読み書きとも) できる。
- これを利用して、部品 x の値 (value) を取り出し、それを実数に変換 (`parseFloat()`) したものを変数 `x` に入れる。
- 同様に、部品 y の値 (value) を取り出し、それを実数に変換 (`parseFloat()`) したものを変数 `y` に入れる。
- `x + y` の計算結果を部品 z の値としてセットする。

この程度のプログラムでも、入力欄に記入してボタンを押すと計算されるというだけで、さっきの C プログラムよりもずっとそれっぽい感じがするでしょう?

演習 1 このプログラム (`/home/guests/kuno/work/js1.html`) をコピーしてきてそのまま動かせ。動いたら次のように変更してみよ。

a. 和のかわりに差、積、商を表示するようにさせよ (どれか 1 つでもいいし、全部表示してもいい)。
b. 華氏の温度を入力すると摂氏に換算してくれるプログラムを作れ。なお換算式は $C = \frac{5}{9}(F - 32)$。
c. その他、何か便利な計算をするプログラムを作れ。

せっかくだからもう 1 つ例を出そう。今度は「入力された数が素数かどうか判定する」もので、計算の中でループを使っている。皆様はどうってことないと思うが、始めての生徒さんにループを教えるのはいろいろと工夫が必要なところなので教え方を考えて見て欲しい。

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN">
<html>
<head>
<title>JavaScript Sample</title>
<script type="text/javascript">
function sosu() {
  var n = parseInt(document.forms.f0.elements.n.value);
  var f = "";
  for(var i = 2; i < n; ++i)
    if(n % i == 0) f += (i + " ");   //☆
  if(f == "") f = "素数です";
  document.forms.f0.elements.f.value = f;
}
</script>
</head>
<body>
<form name="f0" action="#"><p>
<input type="text" name="n" size="5">
<input type="button" value="判定" onclick="sosu()"><br>
<input type="text" name="f" size="25">
</p></form>
</body>
</html>
```

さっきの parseFloat() に対し、ここでは整数を入力するので parseInt() を使っている。

このプログラムで面白いのは、欄 f に見つかった因数をまとめて表示するところ (☆の行)。ここで「i + " "」は数値の足し算ではなく、文字列の連結演算で、つまり数値 i を文字列に変換して、うしろに空白を連結している。また「f += ...」というのも連結演算で、既に入っている値の後ろに今作った文字列を連結する、という意味になる。そして i を 2 から $N - 1$ まで変化させながら順に調べるので全ての因数が見つかる。もし 1 つも因数がなければこの数は素数なのでループが終った後で欄 f が空かどうか見て、空なら「素数です」という表示を入れる。なかなかそれっぽいでしょう?

演習 2 このプログラム (/home/guests/kuno/work/js2.html) をコピーしてきてそのまま動かせ。動いたら次のように変更してみよ。

a. 単に因数を並べるのではなく、素因数分解を示すようにせよ (ちょっと大変なのでよほど気が向いたらでよい)。
b. 2 つの数を入力して、最大公約数を求めるプログラムを作れ。
c. その他、ループを使って何か便利な計算をするプログラムを作れ。

3.3 JavaScript によるアニメーション

フォームとボタンを使って計算をさせる以外に、JavaScript ではブラウザの機能と連携していろいろなことができる。特に面白いのは、HTML で記述された要素が「独立位置指定」になっている場合に、その位置を JavaScript 側で制御できることで、これと「定期実行」(一定時間間隔で JavaScript コードを実行する機能) とを組み合わせるとアニメーションが作れる。ただし、このあたりはブラウザによって互換性がないのでこのままではブラウザもある。とりあえず見ていただこう。

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN">
<html>
<head>
<title>JavaScript Sample</title>
<script type="text/javascript">
var x = 0, y = 0, vx = 8, vy = 7;
function step() {
  var e = document.getElementById("d0");
  x += vx; y += vy; e.style.left = x + "px"; e.style.top = y + "px";
  if(x > 400 && vx > 0 || x < 0 && vx < 0) vx = -vx;
  if(y > 400 && vy > 0 || y < 0 && vy < 0) vy = -vy;
}
</script>
</head>
<body onload="setInterval(step,50)">
<div id="d0" style="position: absolute"><h1>亜</h1></div>
</body>
</html>
```

まず本体部分だが、`<div>...</div>`に対して絶対位置指定のスタイル指定と `id` 名 (固有識別名) をつけている。中には HTML だから何を入れてもいいけど「亜」という文字を入れてみた。次に、`<body onload="...">` でページ表示時に実行させるコードを指定しているが、その内容は「`setInterval(step, 50)`」つまり関数 `step()` を 50 ミリ秒間隔で定期的に実行させる、となっている。

関数 `step()` はさらに前にあるが、`document.getElementById()` という関数を使って先に定義した `d0` の div 要素のオブジェクトを取得し、x 座標と y 座標に値 `vx` と `vy` を足し込んでから、先の div 要素の座標として設定することで、一定時間ごとに位置がずれていくようにしている。ただしそれだけだと直線運動して画面から外に出てしまうので、x 座標、y 座標とも「画面の外に出てしまいそうなときは `vx` や `vy` の符号を反転する」ようにしている。これだけですが、見ていて結構面白いでしょう?

演習 3 このプログラム (`/home/guests/kuno/work/js3.html`) をコピーしてきてそのまま動かせ。動いたら次のように変更してみよ。

a. 速度が一定でなく、はね返る時に段々速くなったり遅くなったりするように直してみよ。
b. 動くものを 2 つとか 3 つにしてみよ。(ヒント: div を増やして、変数 x/y/vx/vy の組みも増やす必要がある。)
c. 乱数を使って一定時間ごとに動き方が突然変わるようにしてみよ。なお、`Math.random()` という式は区間 $[0, 1)$ の一様乱数を返してくれる。
d. その他、何か面白い動きをさせてみよ。

なお、先に出て来たフォーム (特にボタン) とアニメーションを組み合わせることももちろんできる。たとえば先の例題を直してボタンを押すと動きが反転するようにしてみた。

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN">
<html>
<head>
<title>JavaScript Sample</title>
<script type="text/javascript">
var x = 0, y = 0, vx = 8, vy = 7;
function step() {
  var e = document.getElementById("d0");
  x += vx; y += vy; e.style.left = x + "px"; e.style.top = y + "px";
  if(x > 400 && vx > 0 || x < 0 && vx < 0) vx = -vx;
  if(y > 400 && vy > 0 || y < 0 && vy < 0) vy = -vy;
}
function chg() { vx = -vx; vy = -vy; }
</script>
</head>
<body onload="setInterval(step,50)">
<div id="d0" style="position: absolute"><h1>亜</h1></div>
<form name="f0" action="#"><p>
<input type="button" value="反転" onclick="chg()">
</p></form>
</body>
</html>
```

演習 4 このプログラム (/home/guests/kuno/work/js4.html) をコピーしてきてそのまま動かせ。動いたら次のように変更してみよ。

a. X 方向の反転と Y 方向の反転を別々のボタンに分けてみよ。

b. 動くものをボタンで操って的に当てる等のゲームにしてみよ。

c. その他、何か面白いゲームを作ってみよ。