コンピュータと教育　49－7
（１９９８．　７．　３１）

高等学校「情報」科目の教科書案　－ 情報Ｂ（２）「プログラムとは何か」および情報Ｃ（２）

和田　勉
長野大学　産業情報学科

情報処理学会情報処理教育委員会・初等中等情報教育委員会での、高等学校に新設される「情報」教科のあるべき内容を示すことを目的としたモデル教科書執筆プロジェクトの一部として、執筆の一部：情報Ｂ（２）「プログラムとは何か」および情報Ｃ（２）「複雑なプログラムの実現」が私に割り当てられた。これを受けて執筆した初稿の要約を示す。これに対し今まで、タートルグラフィックス、ＬＯＧＯ、指導の手法等に関し批判・意見がよせられた。それらと、それに対し私が思うところ等を述べる。

Proposal for the subject "Information" in senior high schools - Information B(2) "What is a program" and Information C(2)

Ben Tsutom WADA
Division of Industry and Information Science
Nagano University

As a part of an IPSJ comittee's project for writing a textbook on the subject "Information" which will be established as a new subject in senior high schools, which is the project for the sake of showing its ideal content, I was alloted to write some parts: Information B(2) "What is a program" and Information C(2) "Realizing complicated programs." The abstracts of the first draft, which I have written in corresponding to it, are shown. Some arguments for it, especially on turtle graphicss, LOGO, and leading method, have been given. I will introduce them and describe some my arguments on them.

０．序論

情報処理学会情報処理教育委員会・初等中等情報教育委員会での、高等学校に新設される「情報」教科のあるべき内容を示すことを目的としたモデル教科書執筆プロジェクトの一部として、執筆の一部の担当が私にも割り当てられた。この試案の全体構成は［情報処理学会９８］に述べられており、私はそのうち情報Ｂ（２）「プログラムとは何か」および情報Ｃ（２）「複雑なプログラムの実現」の担当となった。

これを受けて、私自身が長年自分の大学での

初心者学生教育で積み上げてきた自作資料・ノウハウを元にして担当部分の初稿を執筆し、1998年5月27～29日のFJK98の際に行われたワーキンググループの会合に持参した。

以下に、まずこの初稿の要約を示し、続いて今までにこれに対し寄せられたいくつかの御批判・御意見と、それに対し私が思うところ等を述べる。

1．モデル教科書担当部分の初稿の要約

以下にモデル教科書担当部分の初稿の要約を示す。小さい字の部分は初稿のそのままの引用した部分、うち...はその箇所に省略した部分があるという意味であり、大きめの字の部分は今回書き下ろした要約である。

1.1. 情報B（2）プログラムとは何か

・アルゴリズムとプログラム

日常生活の中から、例として「電車に乗る」時のことを考える。

駅に行く
切符を買う
改札口を通る
電車が来たら乗る

このように、あることを行なう手順をアルゴリズムという。例えば上記のものは「電車に乗るためのアルゴリズム」と言える。

...

駅に行く
改札口を通る
電車が来たら乗る

これは「定期券を持っている人が電車に乗るアルゴリズム」といえる。この二つをまとめて以下のように書くこともできる。

駅に行く
もし 定期券を持っていない ならば
　　切符を買う
改札口を通る
電車が来たら乗る
...

上の例ではアルゴリズムは普通の言葉で書いたが、コンピュータに教える場合はコンピュータが理解できる言葉である「プログラミング言語」で書く。その書いたものを「プログラム」と言う。コンピュータが動いている時、必ずその中でには誰かが書いたそのためのプログラムが入っており、コンピュータは必ずそれに従って動いているのである。

・コンピュータのプログラム

...

プログラムは上記のように「プログラミング言語」で書かなければならない。
プログラミング言語にはいろいろな種類があるが、ここではLogo（ロゴ）言語を使う。

...

・プログラムの概念、連接

タートルグラフィックスの概念、forward 命令と right 命令を述べた後：

では、この亀に正方形を描かせるにはどのように指示したらいいか考えてみよう。たとえば一辺100の正方形を描かせるには

100歩前進する
90度右へ曲がる
100歩前進する
90度右へ曲がる
100歩前進する
90度右へ曲がる
100歩前進する
90度右へ曲がる

と繰り返せば良い。これで「亀が正方形を描くアルゴリズム」ができた。

...

下のようにしてひとまとめにして名前をつけてしまっておくと、いつでも呼び出して動かすことができる。通常はこの形のものをプログラムと呼ぶ。上の「亀が

正方形を描くアルゴリズム」をプログラムとして書いてみよう。

```
; 一辺100の正方形を描く
to seihoukei1
forward 100      ;最初の一辺を描く
right 90         ;右に曲がる
forward 100      ;次の一辺を描く
right 90         ;右に曲がる - 以下同様
forward 100
right 90
forward 100
right 90
end
```

このプログラムをコンピュータに入れておき

```
seihoukei1
```

と名前を呼び出してみよう。その中の命令が順に実行されて正方形が描かれる。このように、プログラム中に書いておいた命令は通常、上から順にひとつづつ実行される。

・入出力

（入出力 案1ートップレベルの引数による）

```
...
; 指定された大きさの正方形を描く
to seihoukei2 :a
forward :a       ;最初の一辺を描く
right 90         ;右に曲がる
forward :a       ;次の一辺を描く
right 90         ;右に曲がる - 以下同様
forward :a
right 90
forward :a
right 90
end
...
```

aというのは「数を覚えておく場所」の名前である。前の例の

```
forward 100
```

では、直接100と書いてあるので亀は必ず100歩進む。これに対し

```
forward :a
```

と書いておくと、コンピュータは実際に亀を前進させる前にまずaという場所を調べる。...

こう書いたプログラムを、例えば

```
seihoukei2 80
```

のように呼び出すとする。プログラムの1行目は

```
to seihoukei2 :a
```

となっている。この場合、プログラムの1行目に書いてある変数「a」に、呼び出す行の同じ位置に書いた「80」がまず入る。...

次に同じプログラムを

```
seihoukei1 150
```

と呼び出すとどうなるだろうか。

...

上の例を分かりやすく書き直したのが下のプログラムである。

```
; 指定された大きさの正方形を描く
;   ー読みやすい書き方の例
; :hen      正方形の辺の長さ
to seihoukei3 :hen
(print "辺の長さ :hen "の正方形を描く)
   ;「辺の長さ 80 の正方形を描く」などと表示する
forward :hen     ;最初の一辺を描く
right 90         ;右に曲がる
forward :hen     ;次の一辺を描く
right 90         ;右に曲がる - 以下同様
forward :hen
right 90
forward :hen
right 90
end
```

（入出力 案2ー通常の入出力による）

案1はコマンド行から手続きを呼び出す際の引数として数値を与えたのに対し、こちらでは手続実行中に入力の命令を実行して数値を読み込ませている。

（入出力　終り）

```
...
```

; があるとその右はコンピュータは読み飛ばすので、普通の言葉で説明を付けるなどの時に使う。説明（コ

メント）はコンピュータの動作に影響はないが、プログラムの見やすさ分かりやすさの点から適当な説明を付けておいた方が良い。

・繰り返し

（繰り返し 案１
　－repeat（回数指定繰り返し）による）

...プログラムでは「この命令を何回繰り返せ」という書き方を使うことができる。これを使って上のプログラムを書き直してみる。

```
; 指定された大きさの正方形を描く
;  －繰り返しを使った書き方
; :hen      正方形の辺の長さ
to seihoukei4 :hen
(print "辺の長さ :hen "の正方形を描く)
      ;「辺の長さ 80 の正方形を描く」などと表示
repeat 4 [
      forward :hen      ;一辺を描く
      right 90          ;右に曲がる
      ]
end
```

（繰り返し 案２－while による）

...プログラムでは「○○である限りこれを繰り返せ」という書き方ができる。これを使って上のプログラムを書き直してみる。

```
; 指定された大きさの正方形を描く
;  －繰り返しを使った書き方
; :hen      正方形の辺の長さ
to seihoukei4 :hen
(print "辺の長さ :hen "の正方形を描く)
      ;「辺の長さ 80 の正方形を描く」などと表示
make "i 1  ;変数ｉに最初１を入れる。
while [:i < 5] [
      forward :hen      ;一辺を描く
      right 90          ;右に曲がる
      make "i :i + 1    ;変数ｉを１増やす
      ]
end
...
```

このプログラムの動く様子を見てみよう。

```
...
make "i 1  ;変数ｉに最初１を入れる。
while [:i < 5]  ;変数ｉは５より小さいか
      ; 変数ｉは１なのでそのとおり。
      forward :hen      ;一辺を描く
      right 90          ;右に曲がる
      make "i :i + 1    ;変数ｉを１増やす
            ; 変数ｉが１から２に変わる
...
```

繰り返しを使った例題をもう一つ見てみよう。こんどは渦巻模様を描いてみよう。...

・条件分岐

...例えば下のプログラムは「辺の長さが100未満の場合は正方形を描く」というプログラムである。

```
; 指定された大きさが100未満の場合正方形を描く
; :hen      正方形の辺の長さ
to seihoukei100miman :hen
if :hen < 100 [
      (print "辺の長さ :hen "の正方形を描く)
      ;「辺の長さ 80 の正方形を描く」などと表示
      repeat 4 [
            forward :hen      ;一辺を描く
            right 90          ;右に曲がる
            ]
      ]
end
...
```

上の if 命令は「こういう場合はこれをする」という指示だけなので、そうではない場合は何もしない。これに対して「こういう場合はこれをする、そうではない場合は別のこれをする」という書き方もある。例えば下のプログラムは、「辺の長さが100未満の場合は正方形を描き、また100以上の場合は指定した場合は正三角形を描く」プログラムである。

```
; 指定された大きさが100未満の場合正方形を描く
; 100以上の場合は正三角形を描く
; :hen      正方形／正三角形の辺の長さ
to seihousankaku :hen
```

```
ifelse :hen < 100 [
      (print "辺の長さ :hen "の正方形を描く)
      ;「辺の長さ 80 の正方形を描く」などと表示
      repeat 4 [
            forward :hen ;一辺を描く
            right 90       ;右に曲がる
            ]
      ] [
      (print "辺の長さ :hen "の正三角形を描く)
      ;「辺の長さ 80 の正三角形を描く」など表示
      repeat 3 [
            forward :hen     ;一辺を描く
            right 120  ;右に120度曲がる
            ]
      ]
end
```

上の例で、繰り返しや条件判定の中の「命令」の部分は一段右に下げて書いてある。このよう字下げしてに書くことで、その「命令」の部分の範囲が分かりやすくなる。例えば下のように書いてあると...非常に分かりにくくなってしまう。

```
; 指定された大きさが100未満の場合正方形を描く
; 100以上の場合は正三角形を描く
; 字下げしていない良くない例
to seihousankaku :hen
ifelse :hen < 100 [
(print "辺の長さ :hen "の正方形を描く)
repeat 4 [
forward :hen     ;一辺を描く
right 90   ;右に曲がる
]
] [
(print "辺の長さ :hen "の正三角形を描く)
repeat 3 [
forward :hen     ;一辺を描く
right 120  ;右に120度曲がる
]
]
end
```

1.2. 情報C（2）複雑なプログラムの実現

頁数の関係で扱った主題を記すだけに留める。

＊制御構造の組み合わせ

・if の then 部に repeat が含まれている例
－前出の例が実はそうなっていたという話

・if の then 部、else 部それぞれに if がネストされている例
－数の偶数奇数と 80 以上未満により、違う図形を描き分ける

・制御構造のネストが深い場合はインデンテーションを正しく付けることが更に重要である

＊手続き

・前出の、数の偶数奇数と 80 以上未満により違う図形を描き分けるプログラムの、4つの部分を取り出して手続化し、主手続きはそれを呼び出すものに書き換えた例
－手続中からの手続き呼び出しは、引数に定数だけでなく変数や式も使えるということを例示

＊関数（戻り値）

・2つの数をうち小さい方を表示する例。演習として大きい方を表示するプログラムも作らせる－この段階では戻り値を使わない手続きの形にしておく

・これを呼び出して「2つの数 n1 と n2 を与えると正 n1 角形と正 n2 角形を続けて描く。ただしまず辺の数の少ない方を描き、次に多い方を描く」プログラムを作る
－「2値のうち小さい（大きい）ものを表示」の手続きを、結果を表示せず戻り値にする関数に書き換えて利用－関数と手続きの違いを説明。

2．今までにいただいた御意見とそれに対する私の考え

＊タートルグラフィックスについて

タートルグラフィックスと絶対座標グラフィックス（画面又は仮想平面上の座標を指定して図を描かせるもの）を比較してみてそれぞれの長所欠点を論じる御意見をいただいている。

まず最初に：

タートルグラフィックスは本質的に「タートルの位置、向き、ペンの状態」という、隠れた「状態」（プログラム中に陽に出てくる変数の値以外の「状態」）がある。例えば同じ forward 100 を実行しても、タートルが上を向いていれは垂直線が描かれ、右を向いていれば水平線が描かれる。プログラム中のすべての変数の値が同じでもこのように動作は異なる。このように隠れた状態があるのは手続型言語のわかりにくさのひとつであり、短所であろう。

というものがあった。これに関し考えることをいくつか書いてみる：

・タートルの状態を返すライブラリ関数はあり、これを説明に活用するという手もある。

・絶対座標グラフィックスでも、すでに画面に描かれている図形、という隠れた状態はある。場合によってはパレットの設定などという隠れ状態がある場合もある。グラフィッククス以外でも、例えば端末画面に文字列を次々に表示してゆく場合には「既に表示した文字列」という隠れ状態がある。タートルグラフィックスだけが隠れ状態を持っているわけではない。

・絶対座標グラフィックスを用いて教育を行った場合に、実際にどのような、よりよい点があるのかは、私にはわからない。そのようなコースを設計し実際の教育現場で実験実証してみたことがないから。ただタートルグラフィックスにより初心者学生を教える、試行錯誤で作り上げた自作のコースだけはあり、きちんと比較評価はできないものの一定の効果は確かにあげている、と言えるだけである。

次に：

そもそもグラフィックスは不適当ではないか、高校では問題解決の道具としてのプログラミングを考えさせるのが適当で、お絵かきなどのお遊びは小学校ならいいが高校では不適当だ、生徒への動機づけの点からもよくない。

というものがあった。これに対して考えることは：

・対象者により確かにそのような場合も存在するだろう。この点は「受講者（高校生）はどのようなことに興味を示す人達か、何を目指している（または何も目指していない）のか、また何を目標とさせるべきなのか」という大きな前提にかかわることである。最後の「何を目標とさせるべきか」はともかく、前の二つに関しては、高校生だけに限っても実にいろいろな場合がある。（これについては後で再度論じる）

＊ＬＯＧＯは不適当ではないか

よくタートルグラフィックスの意味でＬＯＧＯと言われることもあるが、ここではその意味でなく、たとえタートルグラフィックスでやるにしても他の言語にタートルグラフィックスライブラリを組み込んだもののほうが適当ではないか、という意味である。

これに関しては、私も特にＬＯＧＯを支持するつもりはない。ただ入手容易なタートルグラフィックス処理系がＬＯＧＯしかないから使っていたにすぎないからだ。

ただ、ＬＯＧＯの持つＬＩＳＰ系のユーザインターフェイス、すなわち手続中からの別の手続きの呼出しとコマンド行から手続呼出しが同じ形ででき、かつ同一の手続きがどちらからでも呼び出せることや、プリミティブ（ライブ

ラリ関数に相当）も手続中から呼び出せるだけでなく直接コマンド行からも呼び出せる環境が提供されていることは、とくに第一歩の段階で導入がしやすいという大きな長所がある。

ただこれも両刃の剣であり、逆に、手続きを書かなくてもプリミティブをコマンド行からいくつも入れることで一応図が描けてしまうことから、手続きすなわちきちんとした形のプログラムを書かせたい段階に入っても、その意義や違いがわからず、あいかわらずコマンド行からプリミティブをいくつも入れて、図が一応描けるのでそれで事足れりとしている場合が多く見られ、その指導に苦労する、という場合がかなりある。ただ引数付きの手続きを書かせると一転してちゃんと書くという場合も多いが。

＊この案はスモールステップすぎる書き方であり、生徒の発想を押さえこんでしまう。

例えばこの初稿では、まず forward と right を4回ずつ実行させて正方形が描けることを説明し、その後で逆回りで正方形を描く方法と正三角形を描く方法を演習問題として考えさせている。これは説明のしすぎであり、基本動作だけ紹介して「正方形を描くためにはどうすればよいか」を問いかけるべきだ、という御意見があった。

場合によってそうかもしれない。そのような導入方法は試したことがないので、次の機会に試みに行って結果を見てみようとは思う。ただこの、「どこまでを説明して与えどこからは考えさせるのがよいか」は微妙な程度問題であり、その最適値は対象となる受講生の層により全く異なる。またその設定は、より高い層の場合は許容範囲が広く楽であり、より低い層の場合は許容範囲が狭く、より気を付けて腫れ物に触るように進めないと習得自体を放り出してしまう。（ここで高い／低いとは、論理的に考える能力／訓練程度や授業への意欲が高い／低いという意味である。）

また初稿の中で、くり返しを使って正方形を描く命令

```
repeat 4 [forward 100 right 90]
```

を説明し、正n角形に一般化した

```
repeat :n [forward 100 right 360 / :n]
```

については演習として出題しているが、これに関し「相手は高校生であり小学生ではない、90と4と360の関係くらいすぐわかるはずだ」という御意見をいただいた。

しかし私は自分が大学で教えてきた現場でこの問題を何度も出したが、かなり苦労したりあるいは結局自力では解けなかった大学生が今まで何百人もいた。もちろん簡単にとけた者もいたが、それほど多くはない。この状況は大学により、学部により、全く異なるだろう。同じ問題に対し「小学生と間違えているんじゃないのか、あほらしくてやっていられない」と騒ぎ出す所も確かにあるだろう。

この問題は、前に論じたことと合わせ、次節で再度意見を書く。

最も言いたいこと：

＊良いものは一種類ではない　適材適所がある

＊世の中の高校生は一流大学に進むような層ばかりではない、実にいろいろな層がある

前出のように大学生に限ってもそのばらつきは激しく、そして恐らく高校生ではさらに想像を絶するほど激しいバラつきがあるだろう。sigps（このプロジェクトのメイリングリスト）などを見ていると、多くの（一流・準一流程度の）大学の先生方、一流企業の研究員の方々にはそれが見えていないように思われる。これは無理ないことではある。その方々自身も日常接している人達も、「一流大学へ進む層」

だった人達ばかりであろうから。しかし世の中でも高校でも、それは全体から見れば実はわずかな割合である。高校では（大学でも同じ）学校内にいろいろな層がいるというよりむしろ高校による差が大きく、前記のような層だけを想定した指導方法では全く実行不可能、というところも多いことは忘れてはならないと思う。

3．余談

以下は最近の私の実話である。

今日の授業もまた憂鬱だった。1年生の必修授業でのこと、いつものように早めに実習室へ行き教卓で準備をしていると、あまり見覚えのない学生がやってきた。「あのう、○○といいますけど、どこに座ったらいいでしょうか？」

私のこの実習授業は座る席（ブース）を指定してある。今は7月始め、すでに4月の初回から数えて10回目の授業、今回と次の回そして前期試験を終われば夏休みに入るという時期である。席の指定も4回前に席替えして以来変えておらず、学生はみな自分の席はわかっているはずである。

「自分の席を知らないってどういうこと？学籍番号は？」「えーと…」

入学後3ヵ月以上過ぎているのに自分の学籍番号をおぼえていない様子だ。記録を調べてみたところ…

「もしかして、この授業に今日はじめてきたの？」「はい…」

年度最初に事務から回ってきた名簿には名前があったが、一度も現れないので名簿からも座席割当もとっくに削除した数人のうちの一人だった。

「今までどうしたの？何か特別な事情があったの？」「いえべつに…」「事情が無いのになぜ今まで来なかったの？」「最初の回をサボってしまって…」「最初だけでなく今まで一度もきていないでしょう？どういうこと？」「…」「他の授業は出ているの？出てないの？」「時々…」「とにかくとりあえずそこに座っていなさい、あとでもう少し話をしよう」

授業を始めた。今日は計算機上の演習はせず、講義と紙上の小演習を行うのだが、学生の何人かはそれぞれの画面に勝手なWebページを表示させたまま話を聞いていた。（聞いていない者も多いが。）　彼の目の前にも計算機があるが、ログインすらしたことがない彼はうつろにそこに座っていた。後で話をしようといっておいたにも後かたずけをしているあいだに彼の姿は消えていた。

彼はこの春にこの大学に手続き上は入学したが、必ず受けなければならないはずのこの授業に全く足を向けておらず、当然何一つ学んでいないのである。また十中八九、他の授業も出ていないのであろう。

これは決して珍しいことではない。同様の者は彼を含めて４０人中３人いた。あとの二人はいまだに一度も現れないのである。最初の数回来ただけという者も含めるともっと多い。今回、名簿上の人数４０人に対し、実際に出席していた学生はちょうど３０人だった。その前の回は２５人だった。毎年似たような状況である。

彼はこの３月まで高校生だったはずである。（浪人していれば別だが）　どのような高校生活を送っていたのだろうか。きちんと授業を聞いて積極的に学んでいたとは思われない。

彼のような層は今の日本に確実に存在する。もちろん学校によりほとんどいないところもあるだろうが、そのしわ寄せを受け極めて多いところもあるはずだ。どちらであろうと全て高校として扱われている以上、「情報科」が必修（選択必修）になると、その担当の先生は彼のような者にも必ず「情報」を学ばせなければならないわけである。我々はその先生のために、どのような指針を示し助力することができるだろうか？

参考文献　［情報処理学会９８］　「初等・中等教育における情報教育の提案（文部省初等中等局長あて）」情報処理学会情報処理教育カリキュラム調査委員会（委員長　都倉信樹）平成１０年２月２５日