文系学部におけるプログラミング教育の意義：健全なユーザ育成のための情報教育の視点から

HARADA, Etsuko / 原田, 悦子

(出版者 / Publisher)
法政大学社会学部学会
(雑誌名 / Journal or Publication Title)
Society and labour / 社会労働研究
(巻 / Volume)
38
(号 / Number)
3・4
(開始ページ / Start Page)
119
(終了ページ / End Page)
134
(発行年 / Year)
1992-03
(URL)
https://doi.org/10.15002/00006660

文系学部におけるプログラミング教育の意義：健全なユーザ育成のための情報教育の視点から*1

原　田　悦　子

概要：

近年のコンピュータの普及と一般社会への浸透により，大学文系学部においても情報教育がなされることが当然であると考えられるようになった．しかしその教育の目的や方法，教材などについては，多様な主張がなされており，未だ意見の一致を見ていない．本論文では，四年制大学文系学部での情報教育を健全なユーザ育成を目標とする教育と位置づけ，その中でのプログラミング教育の意義とその方法を考察する．

1. 文系学部における情報教育の目的

現在，国公立・私立を問わず，ほとんどの大学・学部において基礎科目の一つとして情報処理教育科目を開講している．しかしそれらの科目を担当する教員の間では，何を目的として何をどのように教えるべきかという点について議論が絶えない．非専門の多人数の学生に教育することの難しさや大学・学部ごとの特殊事情といった背景も加わり，「ワープロ，表計算，データベースの三種の神器を教えればいい」という実用一点張りの主張から，「計算機科学・情報科学の本質を全般的に教授すべき」という正統的抽象派まで，様々な主張が聞かれる．特に議論が分

かれるのは，80年代までの情報処理教育の中で中心的な位置を占めていたプログラミングをどのように扱うか，という点である．プログラミングの扱いをめぐる主張を大きく2つに分けると次のようになる[*2]．

（1） 情報教育の目的は，コンピュータを自分の問題解決の道具として利用できるようになることである．したがって，いわゆるコンピュータ・リテラシーの育成と実践的なアプリケーションシステムの利用を目的とした教育を行なうべきである．このためプログラミングは基本的に不要であり，逆にプログラミングを強制することにより，コンピュータアレルギーを起こす危険性があるので，むしろプログラミングは有害といってもよい．

（2） 情報教育の目的はコンピュータの操作法を学ぶことではなく，情報科学の基礎原理を知ると同時に論理的思考を育成することにある．したがってアルゴリズム教育は必須であり，そのためにプログラミングこそが教育の中心となるべきである．

この2つの立場は，その目的から前者はコンピュータ道具主義，後者は情報科学教養主義と呼ぶことができるが，本来の目的自体は相互排反するものではなく，補完的な関係にあるといってよい．実際，両方の目的を達することができるようにカリキュラム体系を組むことも可能である．問題は，目的に対する教育方法であり，中でもプログラミングのとらえ方である．近年，特に文系学部での教育に関しては，前者の道具的立場を強調する声が強くなり，それに伴いプログラミング教育をほとんど行なわない実践例も聞かれるようになった．しかし，「アプリケーションシステムの利用が最終的な目的であるから，教育方法としてはそのシステムを使用すればよい」という結論はあまりに短絡的である．本論文では，道具主義的な情報教育の教育方法として，プログラミングが本当に不要か否かを考えてみたい．

道具主義的な立場は，いわゆる実用主義とは異なる．実用主義が卒業後，特に就職後に直接利用する（であろう）アプリケーションシステム

の操作の方法を実利的な立場から教えようとするのに対し，道具主義では問題解決場面でのアプリケーションシステムの利用の体験を通じて，主体的な情報活用能力の育成を強調する．いわば『健全なユーザ』の育成が教育の目標であると言ってよいであろう．武井（1990）は同様の主旨から道具主義の教育を「消費者型一般情報教育」と呼んでいる．

この目標の下で取られている教育方法の中心は，専門領域での実データ（例えば，社会学部での社会調査データ）を材料とし，専門教育の中での問題場面を擬似的に作り上げ，その解決の過程でいかに主体的にコンピュータ（アプリケーションシステム）を位置づけ，利用していくかを実体験するものである．この方法は，学生の動機づけを高め，「手ではできないことを簡単に実現してくれる役に立つ道具」としてコンピュータを認識するには有効な方法である．また計算機を利用すること自体が最終目的ではないという認識を与える上でもよい方法であると考えられる．

しかし，この問題解決法だけで最終的な教育目的である「健全なユーザ」が育つであろうか．筆者の個人的な経験では難しいと思われる．例えば，日本語ワープロと表計算ソフト（スプレッドシート）を使って，簡単な調査用紙を作成し，数名分のデータをとった後に表計算で処理する，という実習授業を行なったとしよう．問題の一つは，受講する学生にとって日本語ワープロは「パソコンで動くワープロ」，表計算は「便利な電卓」として，別個のものとして受けとめられることにある．両者の共通点や相互関連性など，教える側が意図している有機的な関連性が理解されることは少ない．したがって表計算で得た集計結果をプリントアウトし，その後で印刷出力をみながら日本語ワープロに表を手入力する，といった場面に遭遇するはめになる．また「自分が習ったのはワープロと表計算」であって，パソコンは知らない，習ってない他のソフトは教えてもらわないと使えない，と述べて教員を落胆させることもある．

また最近のアプリケーションソフトの傾向として，すべてメニュー駆

動で操作できることから，「利用すること」だけを目的として利用していると，まさにブラックボックスとして対象システムをとらえてしまい，ウィンドウの裏で何が起こっているのか，自分は今何をしているのかを理解しようとしない傾向が見られる．中には，教師が教えたメニュー選択を棒暗記することによって作業をこなす受講生もあり，「知識が利用される現場を作って，その場での問題解決を通しての学習」がさほど容易ではないことを感じさせられる．

これらの逸話は，問題解決過程で道具として与えられた場合に，その経験だけからコンピュータを基盤とするシステムやそこでのデータ処理の一般的特性などについての洞察を得ることは思いのほか難しいことを示唆している．これは受講生の母集団の資質や興味に依存し，また教える側の技量や工夫にも依存する問題であるが，特に機械的システムやメカニズムへの興味が一般的に低い文系学部の学生にとってはかなりの頻度で生じる問題と考えられる．

こういった経験から，コンピュータを道具として使うことを学ぶ，すなわち健全なユーザを育てることを目的とした場合にも，アプリケーションシステムを利用する実習場面だけでは不足ではないかと考えられる．

2. 健全なアプリケーション・ユーザとは何か

では健全なユーザを育てるためには何が必要であろうか．その点を考察するために，まず「健全なユーザ」とは何かを考えてみたい．直観的に，特定のシステムの仕様や制約に振り回されるユーザ，あるシステムしか利用できず，そのシステムから他のシステムへの応用ができないユーザ，システムを利用すること自体が目的となり，本来の問題解決における合理的判断ができないユーザ，などは不健全なユーザである．またシステム利用が苦痛である場合にも不健全であると感じられる．

健全なユーザには少なくとも 2 つの条件が備わっていることが必要と

思われる．条件の一つは，利用するシステムを客観的に自分の問題解決過程の中に位置づけ，主体的に利用しようとする態度であり，もう一つは，道具としてシステムを使いこなす能力である．

前者は，実際の問題状況の中で，解決の過程を実体験することによって育成可能であると考えられ，上記のいわゆる道具主義的な教育方法が意図している教育効果はここにあるといえよう．特に実際の問題解決場面では，複数の解決手段の中から特定のシステム利用を選択する過程を経なければならないため，ユーザにはシステム（およびシステム利用の結果）を客観的に評価する能力が必要とされる．現状の道具主義的カリキュラムにはその点が欠如しがちであり，今後はその点についての工夫が必要となると思われる．

もう一つのシステムを道具として使用するという能力は，どの様な教育により育成が可能であろうか．

Norman (1986)[1] は「使って楽しいシステム」がシステムデザインの最も重要な目的であるとして，そのためにはシステムが道具（tool）となることが必要だと述べている．この場合の道具とは，そのシステム自体の内部を概念モデルとして明示することにより，ユーザによる相互交渉が可能になり，快適で容易に使えるような道具である．その場合に最も重要なことは「ユーザが，自分が実行した操作に対して持つ統制感 (feeling of control)」であるという（Norman, 1986, p. 49）．この Norman の見解を逆の側，すなわちユーザの側の要件として見ると，システムを道具として使うためには，内部の概念モデルを獲得していると同時に，自己の統制の下にシステムを動かしているという感覚が必要であるということができよう．この感覚をもつために獲得されるべき概念モデルとして，佐伯（1988）[2] は各種の道具に共通なユーザ／システム（道具）／タスクの3者間の関係を図1のようにモデル化している．この図から，道具を利用するスキルの学習には，第一接面（ユーザとシステムとのインタフェース）と第二接面（システムとタスク＝物理世界と

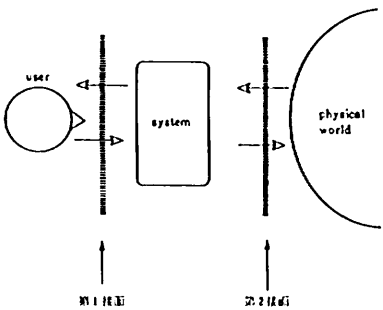

図1　佐伯（1988）のユーザ・インタフェースの二つの界面

のインタフェース）の両者の理解が必要であることがわかる．すなわちユーザが目にしているのは介在者としてのシステムであって，そのシステムが内的な処理を経て，ユーザが要求する課題，すなわちデータの処理を行なっているのだ，というシステム利用のモデルが理解されない限り，アプリケーションシステムを統制感を持って利用することは難しい．特に計算機システムの場合には，アプリケーションシステムが一種のプログラムであり，あらかじめ決められた手続きに沿って動作するものである（ユーザの要求を言葉として理解しているのではない）というシステムのメンタルモデルと，自分がシステムに要求しているタスクが各種のデータ処理であり，システムを通じて「データに触っている」という感覚（認識）を持つことが必要であると考えられる．

これら2つ，すなわちシステムのプログラムとしてのメンタルモデル，およびデータ処理感覚を獲得させる教育方法についての一つの考え方は，実際にアプリケーションシステムを動かして見ることによって，両者は自然に獲得されるであろうというものである．極端な道具主義的カリキュラムがこの立場である．しかし前述のように，現実の状況を見る限りでは，特定のアプリケーションソフトの利用経験だけにより，統制感を持って利用できるだけの計算機システムの概念モデルおよびデータ処理感覚を得ることは難しい．その理由の一つは，問題場面が強調されればされるほど受講生の注意は「何をどうすればよいか」という問題解決に

集中し，道具であるシステムの動き方について洞察を行なう余裕がなくなること，もう一つは，知的でよく工夫された「よい」システムであればあるほど，内部の概念モデルがそのシステムのターゲットとする課題に特化されており，一般的な計算機システムの概念モデルを覆い隠してしまうためである（たとえばスプレッドシートを学習した学生にとって，セルはデータが書かれる「場所」であって，変数としての意味は容易には理解できない）．

このため，計算機システムの持つ共通のシステムモデルを得るためには，システムを動かすこと自体を焦点とする課題状況を設定すること，またタスクとしてのデータ処理を実感するためには，処理の過程を逐次自分の目で確認しながら課題が実行できる環境を用意することが必要と考えられる．この両条件をみたす課題状況として，プログラミングは健全なユーザ育成のための教育の手段と成るのではないかと考えられる．以下に，プログラミングによるシステムモデルの獲得の可能性について検討する．

3. システムのメンタルモデルを持つために

プログラミング教育を行なったことにより，より主体的にアプリケーションシステムを利用できるようになるか否かについての直接的な実証データは，現在のところ存在しない．そこで，プログラミング教育を通じて獲得されるシステムモデル（概念モデル）についての研究を紹介し，その傍証としたい．

原田（1990）[3]は，1年間のプログラミングの授業を受講することにより，学習者がプログラミングのどの様な概念を獲得しているのかを検討するために，プログラムの説明プロトコルの分析を行なった．被験者（社会学部学生）38名は実験室で与えられたプログラムを説明するよう求められ，その言語プロトコルデータが記録された．説明のプロトコル

データは53項目のチェックリストにより分析され，プログラミングの能力による高／中／低の三群の被験者グループの特徴が明らかにされた(表1参照)．その結果，プログラミング能力の高い被験者は，プログラムの流れを順次追っていき，逐次『この文を実行することにより何が起こるか』を明示的に表現しており，いわゆるメンタル・シミュレーションを行なっていることが示された（例．表2a)．これに対し低能力群ではプログラム上の流れを無視した大雑把な記述，あるいは表面的な文(statements) の読み上げだけで説明を終える場合が多く，プログラムを実行した際のダイナミックな動きにはほとんど言及しなかった（例．表2b)．

中程度のプログラミング能力を示した群は，プログラムによる制御の流れに沿った説明，プログラムの意味的な分節化，入出力に関する実行結果の言及などは見られたが，変数に対する操作の実行結果や変数による前後の対照関係への言及は見られなかった（例．表2c)．中能力群のこういった特徴は，その後の変数概念の分析でも同様に見いだされている．原田 (1991)[4] はプログラムの中の変数部分をマークする課題を行い，正しくマークされた個数およびその特性をプログラミング能力群別に比較したところ，中能力群では外部変数は正しく認識しているのに対し，内部変数，特にカウンターなど特殊な機能を持つ変数については，変数であるとの認識をしない傾向が得られている．

以上の結果から，大学でのプログラミング教育を通じて，2/3以上の学生はプログラムを「実際に動いて何かを作業するもの」として理解しており，そのダイナミックな特性をメンタルシミュレーションすることも可能であるということが出来る．これらの被験者が受けたプログラミング教育が特にシステムモデルの獲得を目標として行なわれた訳ではなく，また複数の教員により多様な教育を受けていたことを考慮すると，システム理解のためのプログラミング教育として明確な目標設定をした上で授業を行なうならば，さらに高い成果が期待できる．この結果から，

表1　プログラム能力群によるプログラム説明プロトコルの特徴分析の結果（一部）

●説明の仕方の全般的特徴

項目	H			M			L			χ^2
	Y	N	?	Y	N	?	Y	N	?	
概略のみ説明	2	10	0	3	10	0	5	6	0	2.59 ns
単純な読み上げ	0	12	0	3	10	0	5	6	0	6.87 (*)
行をスキップ	1	11	0	4	8	1	10	1	0	16.66 **
実行順に説明	11	1	0	5	8	0	0	11	0	19.83 **
ループによる分節化	10	2	0	9	4	0	5	6	0	3.77 ns
内容による分節化	3	8	1	0	13	0	1	10	0	4.46 ns
アルゴリズムの言及	12	0	0	6	5	2	3	8	0	13.21 **
現実世界との対応	10	0	2	5	8	0	3	8	0	12.89 **

●ステートメントの実行結果への言及の有無（例）

項目	Y	N	?	Y	N	?	Y	N	?	χ^2
◇制御の流れ										
FORループ反復 [130-150]	12	0	0	10	2	1	5	4	2	6.86 *
条件分岐 (GOTO) [280]	12	0	0	12	1	0	5	6	0	12.70 **
GOTOによる反復 [410]	12	0	0	13	0	0	7	4	0	10.23 *
◇入出力										
PRINT画面表示 [230]	12	0	0	11	0	2	6	5	0	12.26 **
空PRINT画面表示 [260]	11	1	0	3	9	1	2	9	0	15.65 **
INPUTキー入力 [300]	12	0	0	10	1	2	5	4	2	8.25 *
◇変数の値の入力										
READ変数入力 [140]	12	0	0	12	0	1	3	5	3	17.78 **
INPUT変数入力 [270]	10	0	2	7	2	1	3	6	2	10.58 *
代入 [210]	12	0	0	5	8	0	1	10	0	20.6 **
計算結果代入 [350]	10	1	1	3	6	4	0	9	2	17.24 **
◇変数の値の利用										
変数比較 [280]	11	0	1	11	1	1	6	5	0	8.93 *
変数表示 [400]	9	2	1	10	3	0	3	6	2	6.25 (*)
配列変数・比較 [310]	7	1	4	1	12	0	2	8	1	15.15 **
配列変数・計算 [350]	7	1	4	3	10	0	2	9	0	11.44 **
配列変数・表示 [400]	8	2	2	3	10	0	0	10	1	15.42 **

●ステートメント間の関係性への言及の有無（例）

項目	Y	N	?	Y	N	?	Y	N	?	χ^2
READ-DATA対応 [140-1160]	11	1	0	5	8	0	0	10	1	18.91 **
前方照応・連続 [280-270]	9	0	3	9	3	1	2	6	3	11.48 **
前方照応・遠隔 [310-180]	8	0	4	1	8	4	2	9	0	17.41 **
後方照応 [330-350]	3	9	0	0	13	0	0	11	0	6.55 (*)

注：χ^2 係数は，判断不能（表中？欄）のデータは除去して算出された．
**は1%水準で有意，*は5%水準で有意，(*) は10%水準の有意差傾向を示す．

表2　プログラム説明プロトコルの例（310行目についての説明部分）

（付表・プログラムを参照のこと）

a.【プログラミング能力高群の例】Sub. KK

310行目，えーと，3，前の300行目（に）入力された値がHより小さいか，あるいは，えーと，N (T) より大きい場合だから，N (T) っていうのは，1番最初のこの，Nの配列っていうのは，その月の，えーと，……1つの月はなん，何日あるかっていう，そういう変数ですね．その，えーと，その月の……日数をこえた場合は，もう1度入力やりなおしの300行目にもどれという，条件分岐の文がここに，えー，310行目です．

b.【プログラミング能力低群の例】Sub. RT

えーと，310行目で，1より，小さいか……N……わかんない……NよりHが大きかったら，300行目にいって……．

c.【プログラミング能力中群の例】Sub. SK

で，次310行目，またIF文がきます．で，IF文で，……えーと，Hが1より小さかったら，OR，もしくはですね，で，HがN (T)，なんだこれは，……Tというのは下の曜日の計算にでてくるやつですね……んー……N (T) より大きかったら，THEN，300行目．300行目というのはさっきのもう1回INPUT PROMPTをだす，あのー，「日を入力してください」てことです．

プログラミング教育を通じて計算機システムのメンタルモデルを獲得させることは可能であるといえよう．

しかし変数概念を正しく獲得したのはおよそ1/3の高能力群だけであり，「言葉で説明されると理解できるのに，自分ではプログラミングが思うように出来ない」中能力群の被験者にとって，変数概念とそれに対する操作の明確なモデル化が出来ていない点が問題となっていることが示唆された．

健全なユーザ育成のためのプログラミング教育としては，システムの概念モデルをより明確に獲得するための工夫をしていくことが可能であり，また必要である．一つの工夫として，プログラムをシステムへの命令であることを強調した「エイジェント・モデル」に基づく説明が挙げ

られる．たとえばLOGO言語における亀のように，プログラムの実行者を擬人化し，その実行者へのコミュニケーションとしてプログラムをとらえ直すならば，システムを「心の中で動かせる」メンタルモデル(三宅，1984)[5]として理解が容易となると考えられる．また，変数に対するシステム操作の不明確性は，変数がシステム内部のものであり，明示的にイメージすることが困難であることに起因すると考えられる．このため，プログラム実行画面と同時にシステム内部の各変数の状態を視覚的に追跡 (trace) できるような環境（言語処理系）を用意するなど，変数操作を観察可能にする工夫が必要であると思われる．

4. データ処理感覚を持つために

上記のシステムのメンタルモデルが獲得されるならば，ユーザはシステムを介して何らかのデータの処理（もしくはやりとり）を行なっているという大まかな見取図は得られる．しかしここで問題となるのは，実際のシステム利用に際してどのようなデータにはどのような処理ができるのか，またそこにはどのような限界があるのか，というタスク自体のより深い理解である．この理解が深まるほど，アプリケーションシステムが行なっている作業の理解やより効率的な問題解決方法の模索に役立つことは容易に想像できる．

これに関連して，水島 (1990)[6]は複数のアプリケーション間でのデータのやりとりを経験させることにより，一つのアプリケーションソフトの世界での閉鎖性を打ち破ることを提唱している．本論での「データ処理感覚の獲得」は基本的には水島の提案と同主旨であるが，教育方法としては，複数のアプリケーションシステム間の橋渡しの体験よりも，プログラミングを通じてデータを「見る」学習を行なう方が容易であり，また学習成果の一般性も高まると考える．

しかし従来のプログラミング教育はアルゴリズムの教育が主眼であっ

表3　データに対する操作とその結果の意識化を促すための教材項目案

○ファイル操作
　　sequential/random access file とのやりとり
　　OS によるファイル管理
○数値処理
　　桁溢れや誤差はなぜ起こるか
○文字列操作
　　文字とコードの関係
　　1文字ずつの処理（検索と置換）
　　固定長と不定長
○データ構造
　　レコード構造
○画面制御
●データ変換：アプリケーションシステムとのやりとり（水島，1990）
●グラフィック・データの操作
●サウンド・データの操作

註：○は必須と思われる項目，●は必ずしも必須ではないと思われる項目を示す．

たため，数学的な題材をプログラム教材として用いる場合が多かった．またプログラミング教育の主たる担い手が理工系を専門とする教員であったために，操作の対象としてデータをとらえ，その属性に焦点をあてる教材はほとんど用いられてこなかった．したがって今後，プログラミングを通じてのデータ処理感覚の獲得のために有効な教材として，どの領域のどのような教材を扱っていくことが，可能であり必要であるかを検討していく必要があると考えられる．表3にその試案として教材項目案を示したが，今後，情報科学・計算機科学の現状を考慮の上で，その内容を精緻化し確定していくことが必要であろう．

5. まとめ

以上，コンピュータ道具主義の立場から健全なユーザ育成を目的とする情報教育を行なう場合でも，2つの側面からプログラミング教育が有効であり，有益であることを示した．第一節で述べたことから明らかなように，ここで述べるプログラミング教育は従来の「言語教育」とは異なる．すなわち，特定のプログラム言語の仕様を理解・マスターし，プログラムが書けるようになることを目的としているのではない．言語の文法などは必要な部分だけを必要なときに教えることとし，あくまで2つの目標すなわち「システムのメンタルモデルの獲得」および「データ処理感覚の獲得」を達成するために，プログラミング体験を教材として使うことを意図している．

しかし，これに加えてアルゴリズムの意味と概念を有効に教授できるならば，その内容は情報科学教養主義の目標をも満足させるカリキュラムとなる．その意味においても，一般情報教育における教養主義と道具主義は排反するものでなく，両者を有機的に関連づけていくことがカリキュラムをより豊かなものとしていく契機となるといえよう．

また本論の主張は，プログラミング教育だけが必要であるというものではない．健全なユーザ育成のためには，プログラミング教育による基礎教育の他に，いわゆる道具主義的な教育方法，すなわち問題解決場面での情報処理実習も不可欠である．その際には各学部の専門科目との関連づけが一つの重要な要因であり，この点については私立大学等情報処理教育連絡協議会の報告書（1990）[7]の枠組みが役立つ．さらに，情報の持つ意味を単なるhow to的知識に矮小化することなく，より広い文脈から「情報学」として教育すべきだとの主張もあり（土田，1991）[8]，今後，より多角的に文系学部での情報教育をとらえ直していくことが望まれる．

ここまでの議論では情報教育としてカリキュラム，教育方法および教

材を中心に考えてきたが，実際の教育効果を考えるとき，情報教育を授業にのみ限定して考えるべきではなかろう．1週1時間1年のみのコンピュータ利用体験で，あとは全く触れる機会がないという環境で，「情報利用能力の向上」をうたうのはむしろ滑稽である．特に文系学部で現在早急に求められているのは，学生のおかれている環境そのものを，一般社会と同程度にまで情報化することである．これは施設設備に代表される物的環境に限らず，人的環境（すなわち大学教員による研究教育活動の情報化，およびシステム利用をサポートする人的資源配置）および文化環境（情報資源を尊重するとともに，無駄を恐れずに豊かな情報環境を作っていこうとする全体的姿勢）の充実が強く求められる．世の中に本があり，それを読む権利を保障するためにリテラシー教育が当然の権利として認められたことを考えるならば，コンピュータリテラシー云々の前にそれを必要とする環境を整備することが必要である．大学における情報教育の向上は，情報教育担当者だけが負うべき問題ではないことをもう一度強調して，今後の多方面での議論を期待するものである．

引用文献

（1） Norman, D. A. 1986 Cognitive engineering. In D. A. Norman & S. W. Draper (eds.) *User Centered System Design: New Perspectives on Human-Computer Interacion*. Hillsdale, NJ: Lawrence Erlbaum Associates.

（2） 佐伯胖　1988　機械と人間の情報処理—認知工学序説　竹内啓編「意味と情報」東京大学出版会．

（3） 原田悦子　1990　プログラム理解の個人差にみるプログラミングの認知モデル　**日本認知科学会第7回大会予稿集．**

（4） 原田悦子　1991　プログラミング初級者に見られる変数概念　**日本教育心理学会第33回大会予稿集．**

（5） 三宅なほみ　1984　メンタルモデル　**児童心理学の進歩 vol. 20.**

（6） 水島賢太郎　1990　アプリケーションソフトは万能か？データ処理プログラム教育の必要性　**平成二年度文部省情報教育担当教員研究集**

会報告書.

（7） 私立大学等情報処理教育連絡協議会 1990 「私立大学における情報教育の目指すべき方向」(情報処理教育研究委員会最終報告).

（8） 土田昭治 1991 文系初等情報教育はどうあるべきか 明治大学情報科学センター「文系初等情報教育に関するパネルディスカッション」発表資料.

註：

＊1 この研究の一部は平成元年度法政大学特別研究助成金による助成を受けた．また，本稿は平成3年度情報処理学会「夏のシンポジウム」(第31回) での発表を元に構成したものである.

＊2 この他に，いわゆる職業教育の一環として，特定のプログラミング言語を習得することを目的とする教育もあり，またプログラミング教育すなわち言語教育であるとの誤解も（依然として）存在する．しかしこういった考え方が，文系学部での情報教育の目的とは相入れないことは自明であるため，ここでは取り上げていない.

付表：原田 (1990, 1991) で実験材料としたプログラムリスト

【APCS-III BASIC】

```
100  REM  ＊＊＊  曜日の計算  ＊＊＊
110  DIM Y$(6), N(12)
120  !――初期設定――――
130  FOR  I=0  TO  6
140  READ Y$(I)
150  NEXT  I
170  FOR  I=1  TO  12
180  READ  N(I)
190  NEXT  I
210  Y=0
220  !
230  PRINT TAB(7);"<<<  曜日の計算 (1989年)>>>"
240  !
250  !==月／日の入力==
260  PRINT
270  INPUT PROMPT "月を入力（終了の時は，0を入力）:": T
280  IF T <0 OR T> 12 THEN 270
```

```
290     IF T=0 THEN STOP
300     INPUT PROMPT"日を入力      :": H
310     IF H <1 OR H> N(T) THEN 300
320     !==曜日の計算==
330     W=0
340     FOR  I=1 TO T-1
350       W=W+N(I)
360     NEXT  I
370     S=MOD(W+H-1, 7)
380     Y=MOD(Y+S, 7)
390     !==結果の表示==
400     PRINT T;"月";H;"日は";YS(Y);"曜日です。"
410     GOTO 240
1160 DATA 日, 月, 火, 水, 木, 金, 土
1200 DATA 31,28,31,30,31,30,31,31,30,31,30,31
1420 END
```