熊本大学教育学部紀要，自然科学
第39号，101－113，1990

プログラミング教育の体系化をめざして

原田一孝・駄本浩介*

Development of Systematic Programming Education

Ikko HARADA and Kousuke DAMOTO*

(Received May 21, 1990)

The education of computer science has faced to a new stage by the popularization of microcomputers. Since students began to learn "programming" as a subject in high school curriculumn, about twenty years have passed. In these time we have had many trials and efforts have done, the situation has not changed for better. The spreading contents and requirement seem to have become more complicated. And any authorized textbooks are not prepared for the programming subject, also the contents and the methods are not established. These mean the difficulty of systematization in programming education, reversely the requirement of the systematization is eagered. The recursion and pointer techniques in Pascal and other programming language such as Prolog are useful and essential, but difficult to let beginners understand them effectively.

In this paper, we propose a new understandable concept of these techniques by using programming paradigm concept. The learners are expected to understand easily these techniques, and apply to other languages. Furthermore Pascal language, which is applied successfully in the Advanced Placement Course of Computer Science by College Board in U. S. A., is proposed to be suitable for the programming education.

Key words : programming education, high school education, programming paradigm, Pascal, Prolog

1．はじめに

新指導要領で情報に関する内容が大きく取り上げられ，学校教育で行われるコンピュータに関する教育が新しい段階を迎えた．日本では，工業高校などに情報関係の学科を設置し，コンピュータに関する教育を行って20年近くなる．しかし，その中で重要な位置をしめる「プログラミング」などいくつかの科目については，文部省の検定教科書がいまだに出版されていない．このことは，この分野の進歩がはげしく，学校教育としてのプログラミング教育の体系化ができにくいことを物語っている．

ここではまず，指導要領の内容とプログラミング・パラダイムの変遷を比較して考察する．プログラミング教育におけるプログラミング言語の選択は，プログラミング・パラダイムの選択であって重要な問題であることがわかる．

新しいプログラミング教育として，高校生へ Pascal による再帰的プログラミングとポインタ型の変数を使ったプログラミングの指導および Prolog を用いた指導をした．適切に指導すれば新しくプログラミングを学ぶものにとっては，新しいプログラミング・パラダイムの方が，かえって素

* 熊本県立八代工業高等学校

直に理解されやすい．

一方、アメリカでは，高校生が受験する Advanced Placement の Computer Science Cource があり意欲のある生徒のプログラミングの標準カリキュラムとなってプログラミング教育に役立っている．これらと比較して，日本におけるプログラミング教育の内容は大きく立ち遅れているといえる．日本では，通産省の情報処理技術者試験を高校教育の目標とする学校もあり，プログラミング教育に一種の弊害をもたらしているとも言える．プログラミング教育の体系化をめざすには，指導者が新しいプログラミング・パラダイムの発展を見通して，今後のプログラミング教育を行う必要がある．さらに，優れた生徒が学習意欲を増し，才能を伸ばすためにも，大学進学問題も含めて検討すべきである．

2．学習指導要領の変遷とプログラミングパラダイムの変遷

2．1　高校におけるプログラミング教育の変遷

日本における電子計算機の科目が正式に導入されるのは，1970年（昭和45年）10月告示された指導要領にもとづくもので，工業高校に「情報技術科」，商業高校に「情報処理科」ができたのはその2，3年後になる．これらの学科は，普通高校も含めた高校全般へ情報処理の教育を広める先導的な役割も担っていた．現行の教育課程は1978年（昭和53年）8月の改定によるもので，そして今回の改定が1989年（平成元年）3月である．最初も数えると，今回の改定が3回目の改定になる．工業高校のとくに「情報技術科」に関する科目は，ほぼ表1に示すように変わってきた．

表1の中の矢印は，指導要領の改定にともなって，科目の名称も変更しながら変遷したことを示している．たとえば，

●プログラミング　→　●情報技術Ⅰ　→　○プログラミング技術

という書き方は，当初の「プログラミング」が，科目の名称を「情報技術Ⅰ」，「プログラミング技術」と変えながら，内容的には引き継いで変遷していることを表している．この「プログラミング」は，プログラミング教育の先駆的な科目である．この科目の指導要領の内容は表2のように変わってきた．

この表2からわかるように，現行の課程の「情報技術Ⅰ」の最初に記載されている「プログラミングの基礎」が，新課程では「コンピュータにおける問題解決の処理手順」となっていることから分かるように，プログラミング教育の内容が，ただ単にプログラミング作成の科目から，プログラムの利用技術も含めた広い範囲に変わってきている．そのために，現在「情報技術Ⅰ」の内容の「オペレーティングシステム」は，新課程では「ソフトウェア技術」の内容に移っている．この「ソフトウェア技術」は，表1で紹介した科目よりもう少し高度な内容を取り扱う科目として，

●プログラミング理論　→　●情報技術Ⅲ　→　ソフトウェア技術

と変遷してきた科目である．

2．2　プログラミングパラダイムの変遷

プログラミング教育について考える場合，プログラミング・パラダイムの変遷を抜きにしては語れない．そこで，過去15年ほどのプログラミング・パラダイムの発展の概略をのべる．

プログラミング・パラダイムとは，プログラミングを行う上での思考の様式・規範を表している．プログラミング・パラダイムの形成は，プログラミング言語の発展と深く結びついている．プ

表1　学習指導要領と教科の変遷

旧課程の科目		現行課程の科目		新課程での対応科目
年代　1970		1978		1989
工業高校情報技術科用の科目				
●プログラミング	→	●情報技術Ⅰ	→	○プログラミング技術
○電子計算機	→	●情報技術Ⅱ	→	○ハードウェア技術
●プログラミング理論	→	●情報技術Ⅲ	→	○ソフトウェア技術
●システム工学	→	●システム技術	→	○コンピュータ応用
○数値計算法		×		×
工業高校各科共通の科目				
		○工業数理（新設）		○情報技術基礎（新設）

●印の科目については，文部省の検定教科書は出版されていない．「数値計算法」は，現行の課程からは，独立した科目としてはなくなり，その内容は他の科目の内容に加えられた．

表2　プログラミング教育内容の変遷

【旧課程】　―― プログラミング ――

1．目　標

電子計算機のプログラミングについて，基礎的な知識と技術を習得させ，情報を合理的に処理する能力を高めるとともに，プログラミングの実習を通して，電子計算機各部の構成および機構の概要について理解させる．

2．内　容

(1) 基礎的なプログラム　(2) 大量データを扱うプログラム

(3) 複雑な処理のプログラム　(4) 言語文法のまとめ

3．指導計画の作成と内容の取り扱い

(1) プログラムの作成における思考過程を重視するものとする．

(2) 必要に応じて筆算と比較を行いながら思考させるようにする．

(3) プログラミングの具体的な内容は，各学科の専門科目に関係の深いものを選ぶものとする．

(4) 演習および実際にプログラムを働かせることなどを効果的に組み合わせて行うものとする．

【現行の課程】　―― 情報技術Ⅰ ――

1．目　標

電子計算機のプログラミングに関する技術の基礎を理解させ，実際に活用する能力を養う．

2．内　容

(1) プログラミングの基礎　(2) 複雑なプログラム

(3) ファイル処理　(4) オペレーティングシステム

【新課程】　―― プログラミング技術 ――

1．目　標

コンピュータのプログラミングに関する知識と技術を習得させ，実際に活用する能力と態度を育てる．

2．内　容

(1) コンピュータによる問題解決の処理手順　(2) 応用プログラム

(3) テーブルとファイルの利用

ログラミング言語の設計思想の中に，プログラミング・パラダイムが暗黙的に含まれるからである．プログラミング言語に何を選ぶかということは，プログラミング・パラダイムとして何を選ぶのかというのに同じといってよい．そこに，プログラミング言語選択の重要性がある．

コンピュータ・サイエンスの生まれたころは，コンピュータの動作にあわせた表現が自然に書けるプログラミング言語が必要であると考えた．計算は，コンピュータの動作にあわせて「記憶装置からデータを取ってきて演算装置で演算を行い，答えをもとに戻す，ということの繰り返し」としてとらえてきた．この「計算」において，似かよった処理のひとまとまりが手続き（procedure）と呼ぶ．これはプログラミングを構成する上での基本単位である．「計算を手続きの集まりの実行」としてプログラムを作るプログラミング言語を手続き型言語という．従来の Pascal や FORTRAN もこれにあたる．

しかし，計算の対象が大規模化してくると，プログラムを手続きの集まりとしてのみ捉えることが次第に困難になってくる．私たちにとって，記憶や手続きの管理が知的に耐えきれない重荷となってくるのである．

その困難の解決は，コンピュータの原理に合わせたものでなく，人間の思考過程を支援するようなプログラミング言語を考えることから始まる．新しいプログラミング・パラダイムの基本的発想は，自然科学の他の分野に豊富にある．数学（基礎論）や論理学は人間の思考過程の分析をその主な任務としてきた．また，工学では，複雑な対象を抽象的に把握し，モデル化し，モデルを実現する方法をとってきた．これらに，基本的発想を得て，コンピュータ科学者は，関数や論理の概念に基づくプログラミングの方法を見出した．さらに，人間の抽象化の能力を生かして，処理対象のモデル化を行う方法をも有効に取り入れた．これらの方法に基づいて，関数型，論理型，対象（オブジェクト）指向型プログラミング・パラダイムが形成された．また，これらのパラダイムを支援するプログラミング言語を，それぞれ，関数型，論理型，対象指向（オブジェクト・オリエンテッド）型言語という．

現在広く使われているプログラミング言語で言えば，関数型の例として Lisp が，論理型の例として Prolog が，オブジェクト指向プログラミングの例として smalltalk が上げられる．しかし，この例は，分かりやすくするためのものであって，オブジェクト指向プログラミング＝smalltalk といったものではない．

最近は，いくつかのパラダイムを融合させることが行われている．

論理型言語と対象指向型言語とを融合させた言語としてよく知られた例に ESP (Extended Self-contain Prolog) がある．これは，文部省の中央研修会，専門コース（工業88年）でも使われている．

以上のようなパラダイムの特徴を明確にし，おのおののパラダイムの長短得失を明らかにすることは，教育にたずさわる者としても課題となっている．

2．3　構造化プログラミングとその後

70年代は，コンピュータ・サイエンスでは，プログラミング・ディシプリン（dicipline 作法あるいは規律）が問題とされた時代であった．プログラムのしっかりとした書き方が文章を書くときの作法と同じように問題にされた．本来なら，プログラム開発にあたって，大局的なプログラム・パラダイムに関する議論があって，その上でマナー（作法）が問題になる，というのが順序である．しかし，70年代には，コンピュータの実行原理をそのままパラダイムとする手続き型のプログラミング・パラダイムが大前提としてあったので，作法のみが議論された．この時代が，旧課程の指導

要領の時代であり，FORTRAN で指導していた時代である．

理解しやすい整然とした構造を持つプログラムをつくるための「構造化プログラミング」が論議された．その結果，最良の方法は，整った構造を持つプログラムが作れるように，プログラミング言語の構文の改良をすることであった．Algol を源流とする，PascalやCのようなプログラミング言語がその結果として生まれた．

このような背景から，Pascal の使用が普及し，大学はもとより高校などでも Pascal でプログラミングの教育を行い，プログラミング作法について指導するようになった．手続き型の言語に関しては，今後ますます Pascal，Cでの指導が増えるであろう．70年代の半ばに，日本では，「新しい数学」（Finite Mathematics）で有名な，ケメニーが開発した BASIC は，このころ，True BASIC として生まれかわる．現在少なくともプログラミング教育的観点からは，Pascal で指導することについて，議論の余地はないといえる．はじめに覚えなければならない最小限の知識が，他の言語に比べて少なくてすむこと，教育的に書かれたよい教科書，コンピュータ・サイエンスの香り高い参考書がたくさん書かれていること，パソコンで動くすぐれた処理系があるなど，Pascal の特長はいくらでも枚挙できる．

そして，80年代は新しいプログラミング・パラダイムの時代といえる．構造化プログラミングだけでは，抽象化の過程の重要性の認識が不十分であった．抽象化の能力を培うことを抜きにしたプログラミングの訓練は，算数の問題における文章題を抜きにした計算問題だけを解かせるようなものである．鶴亀算とか植木算など，複雑で一見別の問題のように見える問題から，論理を抽象化し既成の問題のパターンに形式化する方法である．この抽象化の過程をプログラミングの場合にも経なければならない．しかし，構造化プログラミングにおいては，「文章題を解く」ことが忘れさられ，ほとんど抽象化の済んだ問題に対して抽象化を説くといったことになってしまった．意味がわすれられ，構文が強調されたといえる．

2．4　プログラミング・パラダイムの変遷からみたプログラミング教育

このようにプログラミング・パラダイムと学習指導要領の変化を見比べるとその変化のようすを知ることができる．日本の高校に「プログラミング」が科目として導入された70年の改定当時は，まだ，プログラム作法の時代以前であった．「プログラミング」に関する，指導要領の解説では，指導されるべき内容として記述されているなかに FORTRAN固有のキーワード，FORMAT，SUBROUTINE などが取り上げられている．この時期には，工業高校に設置された主記憶装置 8 kW（16KB，現在はパソコンでも 1 MB）のコンピュータでは，FORTRAN しか利用できなかったという時代の制約を受けていた．

一方，現時点において，プログラミングの指導を BASIC のみで行うことを事実上指定するような，BASIC に固有な PRINT とか GOSUB といった指導項目をあげるのは20年前の FORTRAN の時代と同じ発想と思われる．昨年（1989年），中高校生を対象としたあるパソコンプログラミング・コンテスト事務局は，応募する学校へプログラムの書き方の冊子を配って goto 文の多用をいましめている．応募されるプログラムが goto 文の多用（冊子のなかでは，「goto 文の嵐」と表現している）によりあまりにも読みづらいためである．このことは，プログラミングが得意な生徒でも，いまだに構造化プログラミング以前という日本のコンピュータ教育の状況を表している．プログラミングを指導している教師は大いに反省しなければならない．

現行指導要領では，構造化プログラミングがさけばれたあとのことであるので，いろいろな言語を使うことを想定している．それまでは FORTRAN が中心であったが，現行では，Pascal なども

念頭において，細かく言語の文法や作法を教科内容としてしめすとは避けている．

今回の新指導要領では，さらに時代を反映して，内容はよりおおまかに簡略化しており，ますます現場の教師の裁量が大きくなってきた．今回改定される指導要領からいかなる内容を読みとり，それをどのように実施に移すのか，それが，私たちに課せられた重要な課題である．私たちは，プログラミング・パラダイムの問題をはじめ，現在のコンピュータ・サイエンスの発展を見きわめ将来を予想して教育する必要がある．

3．高校生に対するプログラミング指導の実践

3．1　再帰的なプログラミングを指導して

再帰的プログラミングは，非常に大切な概念であるが，FORTRAN や BASIC では，そのままの形では取り扱えない．したがって，必要がないと思われたり，難しいと思われたりする．情報技術科３年43名に，Pascal で再帰的プログラミングを10時間指導した．すでに，１年で Pascal，２年で FORTRAN の基礎を学んでいるが，再帰的プログラミングについては初めての生徒である．指導後，アンケート形式で再帰的なアルゴリズムによるプログラミングと繰り返しによるプログラミングのどちらがわかりやすいと思ったかを，なるべくどちらかに決めるように，といって聞いた結果が表３に示されている．主観的で感覚的な解答で，教師の指導法や今までのプログラミングの経験，数学など他の教科の履修状況も関係しているであろうが，再帰的プログラミングの方がわかりやすいと答えたものが多いことがわかる．

指導内容の(1)数列の一般項で，再帰的プログラミングを支持する生徒がとくに多いのは，数列を漸化式で表現したものを多く取り扱ったことによると思われる．また，(4)の２進10進変換では，繰り返しによるプログラミングの支持が多いのは，１年のときから，手計算で２進10進変換を繰り返し処理により行っているからである．

もう少し詳しく指導内容を紹介する．

つぎの数列

2　5　8　11　14　17　20 ……

の一般項 f(n)を求める関数をつくる場合を考えてみよう．

表３　指導したプログラムの例と生徒の反応

指導内容	再帰的処理	繰り返し処理	不明
(1) 数列の一般項	36	6	1
(2) 自然数の和	25	16	2
(3) 階乗	34	8	1
(4) ２進10進の変換	19	24	2
(5) 最大公約数	19	16	8
(6) 順列の生成	×	×	×
(7) 組み合わせの数	×	×	×
(8) ハノイの塔	×	×	×
(9) クイックソート	×	×	×
(10) ８クィーン	×	×	×

注意　①　指導項目の(6)から(10)については，再帰的プログラミングでないと困難である．したがって繰り返しによるプログラミングの指導は行っていない．

②　「不明」は，どちらとも決めかねるものと無記入のものである．

① 数列の一般公式

$$f(n)=2+(n-1)*3$$

② 繰り返しのプログラムで考える（初項2に，公差3をn－1回加える）

③ 漸化式

$$\begin{cases} f(1)=2 \\ f(n)=f(n-1)+3 \end{cases}$$

でも表現できる．

Pascalでは，このように漸化式で表された関数を直接定義できる．高校の数学でも漸化式で表される数列が数多く現れるが，その関数を直接記述できる点で，再帰的なプログラムは必須であると思う．

再帰的な関数を手で計算してコンピュータの動作を確認してみる．n＝4の場合を図2に示す．番号のように，左側の矢印に従って下りて行き，右の矢印に添って上がる．これを，プログラミングの指導のときに手計算させると理解が深まる．

八代工業高校では，小学生と先生を対象にしたLOGOによるプログラミング講座をした経験がある．LOGOを学んだ子供は，再帰的プログラミングを意識せず，繰り返しと同じ感覚でとらえているようである．繰り返しは，「おしまいのほうに，もう一度同じものを書くとよい」といった形で理解されている．たとえば，ヘリコプターが大きく飛び上がる手順はつぎのように書ける．「ヘリコプターを飛ばす」が，自分自身の中の一番終わりに呼びだされている．（記述は，ロゴジャパンのロゴライターによる）

```
てじゅんは　ヘリコプターを飛ばす
    ヘリコプター                    { 注　ヘリコプターを1回分動かす手順 }
    ヘリコプターを飛ばす
おわり
```

i）一般項の公式を使うと

```
function f(n:integer):integer;
  begin
    f:= 2+(n-1)*3 ;
  end;
```

ii）繰り返しのプログラミングで書けば、

```
function f(n:integer):integer;
  var i,a:integer;
  begin
    a:=2;
    for i:=1 to n-1 do
      a:=a+ 3;
    f:=a;
  end;
```

iii）再帰的定義に従って書けば、

```
function f(n:integer):integer;
  begin
    if n=1 then f:=2
           else f:=f(n-1)+3;
  end;
```

図1　数列の一般項を求める関数

```
f(4)=f(4-1)+3          = 11
      ①↓               ⑥↑
   f(3)=f(3-1)+3       = 8
      ②↓               ⑤↑
    f(2)= f(2-1)+3= 5
      ③↓       ④↑
        f(1)= 2 再帰の終了
```

図2　再帰的プログラミングの動作を手で確認させる順序

高校生，小学生の感想から，再帰的プログラムは決して難しい考えではなく，むしろやさしくなることが多いことがわかる．再帰的プログラミングの指導は，早期に指導すべきである．高校の場合，数学との関連もあって指導するには数列の漸化式から指導しはじめるとよい．通産省の情報処理技術者試験においても，再帰的プログラミングを書きやすい言語を採用すべきである．

3.2 ポインタ型を使ったプログラムの指導

3.2.1 指導対象と内容

情報技術科3年に対して，Pascalでポインタ型変数をつかったプログラミングの指導を行った．時間は10時間，内容としては線型リストと2分木を行った．

具体的な指導内容の一部を少し紹介する．

つぎの例題を，配列を使った場合とポインタを使った場合で考えさせた．

例

アルファベット文字列データを繰り返し入力して入力のたびにアルファベット順に並べ変え，入力が終わったら出力する．

配列型を使った手順を図3に，プログラムを図7に示す．配列では，途中に挿入すると，あとのデータは1つづつ後にずらす，データが多いと大変な作業になる．

ポインタ変数を使う場合はデータ構造も考えて，図4のようにカードにデータだけでなく，つぎのデータを指す部分をつけ加える．

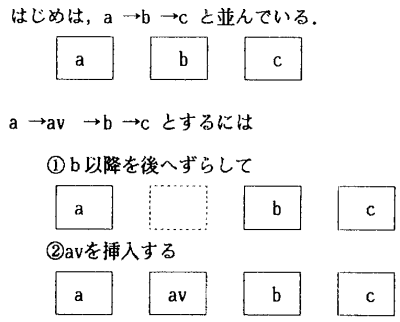

図3 配列を使ったデータ挿入の手順

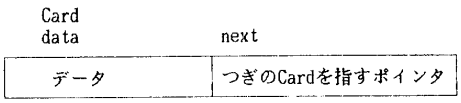

図4 ポインタ型変数を使う場合のデータ構造

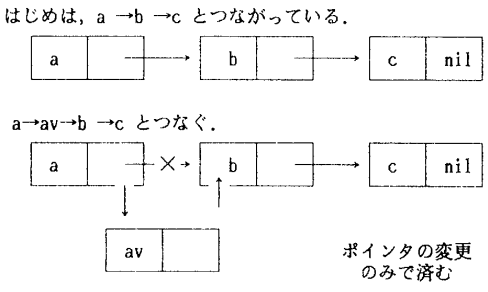

図5 ポインタ型変数を使った場合の挿入の手順1

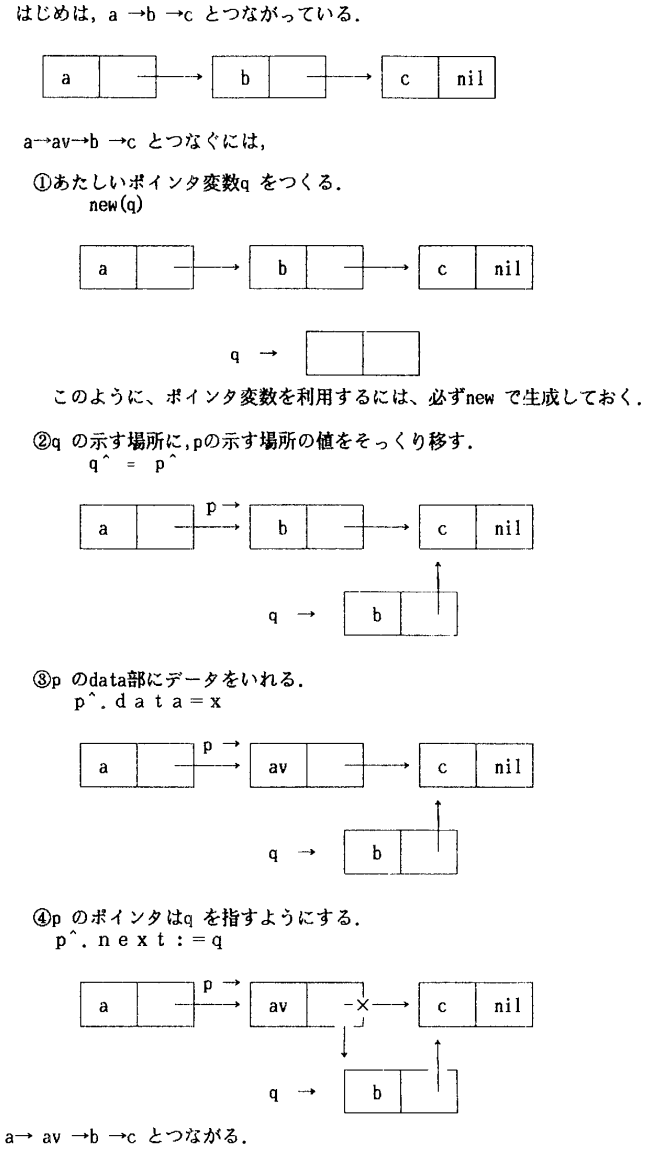

図6 ポインタ型変数を使った場合の挿入の手順2

このカードをつないでデータを並べる．nil は，ポインタの指すものがないという意味の定数で，あとにつながるものがないことを表している．(図５)

しかし，図５のように，すべてをポインタの移動だけで行うには，ｂのカードの前のａのカードをたどらねばならないので，自分の前のカードを指すポインタも必要になる．簡単のために，後ろのカードだけを指す１方向のリストを使う場合の手順を詳しく書くと図６のようになる．この方法だとデータがいくつあっても困らないし，処理が速い．プログラムを図７および図８に示す．

３．２．２　生徒の感想と考察

ポインタを使ったプログラムの学習を終えた生徒は次のような感想を述べている．

○カードの中に，新しいカードを１枚を挿入したり，カードの中から１枚のカードを抜き取ったりすることを考えるとき，挿入したカードのあとの番号はいちいち気にしない，ポインタ型を使うとそんな日常の考えでそのままプログラムできる．○挿入する部分，削除する部分の小さい範囲について考えればよいのでプログラムがらくである．○矢印を，point to と読むと説明があってからよく理解できた．

ポインタ型変数を使ったプログラムを難しいと感じるとすれば，ポインタ変数が難しいのではな

```
program p7;
  var a    :array  [1..100]  of string [20] ;
      n,i,j:integer;
      x    :string [20] ;
begin
   n:=0;a [n+1] :='';
   write('data=');readln(x);
   while x>'' do
      begin
         i:=1;
         while x<a [i]  do i:=i+1;
         if x<>a [i]  then
            begin
              n:=n+1;
              for j:=n downto i do
                  a [j+1] :=a [j] ;
              a [i] :=x;
            end;
         write('data=');readln(x);
      end;
    for i:=1 to n do writeln(a [i] );
 end.
```

図７　配列をつかったデータ挿入のプログラム

```
program p8;
  type ptr =  ^card;
       card = record
                data : string [20] ;
                next : ptr;
              end;
  var  root : ptr;
       x    : string [20] ;
       p,q  : ptr;
begin
   new(root);
   root^.data:=' {';
   root^.next:=nil;
   write('data=');readln(x);
   while x>'' do
     begin
       p:=root;
       while x>p^.data do p:=p^.next;
       if x<>p^.data then
          begin
             new(q);
             q^:=p^;
             p^.data:=x;
             p^.next:=q;
          end;
       write('data=');readln(x);
     end;
   p:=root;
   while p^.next<>nil do
     begin
       writeln(p^.data);
       p:=p^.next;
     end;
end.
```

図８　ポインタ型変数をつかったデータ挿入のプログラム

くリスト処理に慣れていないからである．われわれは，数値を取り扱うことには小さいときから慣れていているが，リスト処理には慣らされていない．リスト処理の数学の基礎，いわば「リスト処理の算数」を学んでいないことが主な理由である．リスト処理の途中の経過を詳しく図示することで理解は容易である．

コンピュータの利用の中で，記号処理が増えている．ポインタ変数を使うと，データのつながりを容易に表現でき，また，変更も容易で，記号処理も楽になる．

3．3 Prolog を指導して

3．3．1 指導対象と内容

情報技術科3年に対して6時間実習の形態で実際にコンピュータの前でコンピュータを操作しながら指導した．内容の例を図9に示す．

3．3．2 生徒の感想と考察

Prolog によるプログラミングの学習を終えた生徒たちの感想を紹介する．

○いままでと，まったく違った感じのする言語であった．いままでのプログラム言語では，プログラムを作って入力して走らせていたが，Prolog では，教えて，たずねるといったように言い換えることができる．事実を教えて質問すると言ったようにするのには驚いた．(H)
○Prolog が面白かった点は，規則を入力し，それに事実を1つ1つ加えてやると，規則が正確に入力されていれば，最終的に結果が必ず yes になることだった．規則を自分で考えると結構楽しめた．(I) ○S君がProlog で悩んでいたので，それほど難しいのかな，と思った．
○プログラムを組むというよりは,文章を書くといった感じで意味がつかみやすかった.(S)
○Prolog と言う言語は，Pascal や FORTRAN とは違う．言葉あそびといった感じで，プログラムの苦手な私にも，馴染めました．考え方が，簡単のようで複雑だったり，発想に苦しむことも，しばしばありましたが，楽しめました．(U) ○Prolog は，関係や定義などを分析して，AND やOR にしたり，2つの事実に分けたり，国語力のない私は理解するのに難しいところがありました．(M) ○Prolog は，もっと，日本語のあいまいな意味を捕らえて，処理できるようになれば使いやすい言語になると思った．(T)

これらの感想を見て考えられることは，いままでプログラミングがよくできていた生徒も，柔軟な発想にとまどったり，また，基本的に，国語の力，論理的に考える力のある生徒はかえって早く理解できてゆくということである．新しいパラダイムによる言語は，早いうちからの指導が重要であることがわかる．

(1) スキ(タケシ，ジロウ).
スキ(タケシ，ヒロミ).
スキ(ヒロミ，タケシ).
スキ(カズヨシ，レイコ).
スキ(カズヨシ，ヒロミ).
と入力されているとき，つぎの質問にどう答えるか。
① |?－スキ (カズヨシ，ヒロミ) .
② |?－スキ (ヒロミ，_ダレ) .
③ |?－スキ (X，レイコ) .

(2)Aの親，兄弟，おじが,Bであることを，それぞれ，
oya(A,B).
otouto(A,B).
ani(A,B).
oji(A,B).
で表すとき，「親の弟や親の兄のことを叔父という」
は，Ｐｒｏｌｏｇの節でどう表されるか。

図9 Prolog プログラミングの例

4．アメリカの Advanced Placement Cource

4．1　アメリカのＡＰコースにおけるコンピュータ教育

アメリカでは，高等学校在学中に勉強したものを大学の入学時の試験で単位として認める制度がある．その Advanced Placement Course (AP) の中に，Computer Science (CS) のコースがある．この内容は高校生が学ぶべき内容として大変興味深く参考となるものを含んでいる．アメリカで使われている高校生向けの Pascal の教科書では，意欲ある生徒がＡＰコースの受験をめざすよう勧めている．

A)Programming Methodology
　1.Specification　2.Design　3.Coding　4.Problem correctness
　5.Documentation
B)Feature of Programming Languages
C) Data Structures
　1.Linear　2.Tree Structures
　3.Other linked structures
D) Algorithms　E) Computer Systems
F) Responsible Use of Computer Systems

図10　ＡＰのＣＳ（コンピュータ・サイエンス）コースのテキストの項目

4．2　ＡＰのＣＳ（コンピュータ・サイエンス）コースの概要および比較

ＡＰのＣＳコースのテキストの項目を図10に示す．

日本との違いは，まず高校生に対する学習の指針としてコンピュータ・サイエンスという立場にたって書かれている点である．さらに，つぎの点が上げられる．

(1) プログラミングの制御構造からデータ構造までかたよりなく基礎的で系統的・発展的に取り上げられている．

(2) 従来コーディングのみに重きがおかれてきた日本と違いがある．

(3) データ構造などが豊富である．

(4) 内容をカバーできるプログラミング言語として唯一 Pascal が使われている．

教師用のテキスト，過去の問題集がある．さらに，問題の解答例に対する評価を加えて解答集が用意されている．

5．まとめ と 今後のプログラミング教育

5．1　プログラミング教育を実務教育から，コンピュータ・サイエンスへ

日本での「プログラミング」教育は，これまで工業・商業系の高等学校が中心であった．そのために，「実務」「即戦力」が要求されてきた．工業の FORTRAN，商業の COBOL という図式が生まれた．

日本におけるプログラミング教育の指針とでもいうべきものとして，通産省の情報処理技術者試験をあげることができる．商業高校，工業高校の中には，この試験の受験をすすめ学校教育の目標の１つにしているところもある．しかし，通産省の委託調査[12]にも現れているが工業高校では，その内容を不満とする声も聞かれる．その内容は，ＡＰコースの内容からみても遅れていて，このままでは，マイナスの面が大きく現れるであろう．高校生が通産省の検定試験に振り回されたりするのは長い目でみると得策ではない．

5.2 プログラミング言語の選択はプログラミング・パラダイムの選択

プログラミング言語の問題は，プログラミング・パラダイムの問題であり，考え方，発想の問題である．「プログラミング言語は何でもよい，アルゴリズムをしっかり教えれば同じだ」というのは，プログラミング・パラダイムとして，ただひとつ手続き型のパラダイムしかなかった時代の発想である．言語が違えば，アルゴリズムもまた違ってくるのである．「アルゴリズムが大切だから，言語を重視する」でなければならない．

5.3 Pascal などの構造化言語について

これからの言語教育については，しばらくは，ぜひ Pascal を使って教えてみてほしい．ＡＰコースでも，いまは Pascal 唯一であるが，「将来適当な言語が採用されることもあろう．しかし，その時になっても Pascal は残るであろう」といっている．私たちの経験でも，Pascal で入門した生徒は，FORTRAN で書いても分かりやすいプログラムを書く．筆者の一人が勤務する八代工業高校では，当面は Pascal と機械語（アセンブラ）は必修で，あとせめて一つくらい別のプログラミング・パラダイムによる言語を学ばせたいと考えている．

機械に密着しすぎるプログラムなどを自由に書きやすい BASIC やＣ言語では，マニアチックになる恐れがある．マニア的な教師にまかせておいてはコンピュータ・サイエンスの本当の教育はできそうにない．

5.4 プログラミングを大学の入学試験に

今年，日本からはじめて国際数学オリンピックに代表を送る．プログラミングについても[15-17]，従来の数学の幾何がはたしてきた役割をはたせるのではないか，大学の入学試験にもとりいれたらという意見もある．マイコン少年たちを集めて大学をつくるという動きもある[18]．中学・高校の時代にプログラミングに熱中できる環境を，カリキュラムも含めてつくる必要がある．

アメリカのＡＰコースのような試験が日本で行われたり，プログラミング・オリンピックなどが企画がなされるようになるであろう．

5.5 新しいパラダイムによるプログラミングの経験

新しいプログラミング・パラダイムは，我々には難しく写っても，初めて学ぶ高校生には，われわれが思うよりずっとやさしいのではないだろうか．むしろ，多く人々によって従来のものを改良して生まれた概念であるから，新しい考えの方がわかりやすいのは当たり前なのかもしれない．0を知らないで計算するより，0を知ってから計算する方がはるかに計算が楽であるように．

どんなに新しく高級な概念であっても，適切な指導をすれば，その年令に応じて，必ず理解できるというブルーナの仮説を励みにして新しいパラダイムのプログラミング教育を行って見た．手続き型の言語しか教育できなくても，新しいプログラミング・パラダイムによる知識があるなしでは，教育のしかたが違ってくる．「オブジェクト指向プログラミングの経験を持つと，目からうろこが取れたような感じ」（竹内郁夫ＮＴＴ基礎研究所）がして，他の言語でプログラムを書くときにも今までと全然違った感覚で取り組むことができる．

6．おわりに

コンピュータ・サイエンス自体が，常に新しい概念を取り込んで発展しているわけであるから，プログラミング教育の体系的な指導は，完成され固定されたものとしては，存在し得ない．しかし，プログラミングに役立つ力というのは何か．また，プログラミングの教育でどんな力がつくのかを整理するところから，プログラミング教育の体系的な指導法が生みだされて来ると考えている．これを基礎として，今後さらに，新しいパラダイムの本質をとり入れたプログラミング教育の体系化に取り組みたい．

参考文献

(1) The College Board : "Advaced Placement Course Description Computer Science", (May 1988).

(2) Bruce Presley and Tim Corica : "A Guide of Programming in TURBO PASCAL", Lowrenceville.
（プログラミング教育に関する25年以上の経験に基づいて作成されたテキストであり，Advaced Placement Examination in Computer Science に備えることも考慮してある．高校・大学の半期または通年講義用テキストであり，再帰的プログラミングの重要性をグラフィックとデータ構造の上から強調されている．教師用のガイドブックもあり，ディスケットも利用できる．）

(3) 井田哲雄（編）："あたらしいプログラム・パラダイム"，共立出版（1989）．

(4) 河合　慧："プログラミングの方法"，ソフトウェア科学講座 2，岩波書店（1988）．

(5) 土居範久，筧捷彦："プログラミングの考えかた"，コンピュータ入門 1，岩波書店（1987）．

(6) 小谷善行："知識指向言語 Prolog"，技術評論社（1988）．

(7) 大岩　元："初等・中等教育におけるコンピュータ"，情報処理学会研究報告，88－CE－1, No. 48 (1988).

(8) Robert J. Bent and George C. Sethares : "Instruction Manual for 'BASIC', An Introduction to Computer Programming", 4th Edition, Brookes／Cole Publishing Co. (1990).

(9) David L. Myers, Valarie A. Elswick, Patrick W. Hopfensperger and Joseph P. Pavlovich : "Houghton mifflin computer education program——Computer Programming in BASIC", Houghton Mifflin Co. (1989).

(10) 宮下弘文："高校数学への再帰的方法の適用とその効果"，信学技報，ET87－11 (1987).

(11) 佐伯，坂村，赤木："コンピュータと子供の未来"，岩波ブックレット，No. 109 (1988).

(12) 日本情報処理開発協会・中央情報処理教育研究所："情報処理教育実体調査報告書アンケート調査集計結果大要"，平成元年度　通産省委託調査（1990.2）．

(13) 正田実他（編）："教育用コンピュータハンドブック '90"，日本評論社 (1990).

(14) 和田日出夫："問題解決の思考と構造"，東洋館出版社（1989）．

(15) 吉村　啓："計算ができることと数学ができること"，数学セミナー（1989.9）．

(16) 磯道義典："ソフトウェアと少年野球"，蟻塔，共立出版（1990.5）．

(17) 戸川隼人："コンピュータを受験科目に"，プログラムの設計論 36, bit, Vol. 22, No. 5 (1990).

(18) --："大人になった新人類企業アスキー"，ASAHI パソコン，(1989.11.15)．

(19) --："パソコンクラブ指導・運営用　プログラミングマニュアル"，旺文社（1989）．