プログラミング教育の社会的意義
―設計から始めるプログラミング教育―

大岩 元[†1]

概要：小学校からのプログラミング教育が 2020 年から日本でも行なわれることになったが，プログラミング教育とは何であるかの認識が人によって違う．一般には何らかのプログラミング言語を習得することであると理解されている場合が多いが，これはプログラミング教育のための手段にすぎず，アルゴリズム構築教育が本質的な目的であるとするのが，国際的な共通理解である．中教審の「小学校段階におけるプログラミング教育の在り方について」では，「プログラミング的思考」を育くむことがプログラミング教育の目的であって，コーディングを覚えることが目的であないことを明言している．

プログラミング教育の歴史をたどり，現時点でのプログラミング教育の意義を，初等教育段階で行なう教育の視点から論じる．

キーワード：プログラミング教育，プログラミング的思考，初等教育，社会的意義

Social Significance of Programming Education
-- Design Process of Programming Education -
--

HAJIME OHIWA[†1]

Abstract: Although programming education is to be started from 2020. What is programing education is not properly understood. Many of the Japanese regards it as obtaining the skill for using programming language, which is different from internationally accepted understanding of the ability for constructing required algorithms. Obtaining the programming language skill is just a mean for algorithm construction.

We discuss the social significance of programming education from the viewpoint of elementary education by looking back the history of programing education in Japan.

Keywords: programing education, way of thinking used in programming, elementary education, social significance

1. はじめに

英，米，北欧諸国，バルト３国などで，小学校からのプログラミング教育が始まった．2015 年７月から８月にかけてリトアニアで開かれた IFIP TC3 Working Conference “A New Culture of Learning: Computing and next Generations”と，ACM の ITiCSE では，内容がプログラミング教育で殆どうめつくされていたが，参加者に聞いた所では，電子政府を作ったエストニアではプログラマーの不足が深刻なこと，大学生に教えたのでは育成が難しいこと，など直接的な理由で小学生からの教育を始める一方，ドイツなどでは日常生活の中にコンピュータが深く入りこんできたことから，その原理を知っておくべきであるという教養主義的な理由からである国もあるようだった．義務教育化に至ったのは，どこの国でも子を持つ親が，自分の子供の将来を心配して国に働きかけたことによるようである．

こうした先進国の動きを受けて，日本でも小学校からのプログラミング教育の義務化が始まった．従来の，「コンピュータは使い方を覚えれば十分」という方針からの 180 度転換であり，文部科学省や学校はその対応に苦慮している．その対応は欧米諸国への追随という側面が大きく，何が可能かという検討に追われている．一方民間では，子どものためのプログラミング教室がビジネスとして大きく発展しそうである．

日本の問題として大きいのは，プログラミング教育という概念の意味理解が欧米の理解と大きくずれていることである．これは，情報技術一般について人材育成を行ってこなかったという特殊事情の反映でもある．本論文では，プログラミングとは何かという議論を歴史的な観点を踏まえて行なう．

2. プログラミング教育の歴史

日本のプログラミング教育の歴史は森口繁一著の「FORTRAN 入門」で始まる[1]．これは，数式の計算を行なうための FORTRAN 言語の教科書であったが，大変よく出来ていたので，当時数値計算を必要とした研究者はこれを学んでから，1965 年に設置された東大大型計算機センタ

†1 お茶の水女子大学
Ochanomizu University

ーで計算を行った．

当時の計算は，数値計算がほとんどで，それ以外の計算を行なうのは情報技術者だけであった．その後，計算機で計算する人口はどんどん増えていき，1970 年に国立大学５校と私立大学 1 校に情報系学科が設置されてから，Computer Science の基礎教育としてのアルゴリズム教育が始まった．ここでも森口教授は優れた対話形式の教科書を書いている[2][3]．この本では数値計算だけでなく，コンピュータ自体のシミュレーション，言語処理系の基礎，テキスト処理まで，具体的な応用を例題として Pascal プログラムをどのように作り出していくかを説明している．

この本の直後の 1983 年に出版された阿部圭一著の「ソフトウェア入門」の中には，ずばり「どのようにしてプログラムを作るか」の章が設けられており，日立製作所の新入社員教育に使われた[4]．またこの年には，高エネルギー物理学者である小野厚夫・川口正昭著の「情報科学入門」[5]が出版されている．

小野・川口の本は，本文 159 ページの小冊子ながら，論理回路からコンピュータの仕組，情報理論からチューリング機械，構造化プログラミング，アルゴリズムや計算量，数値計算における有限桁の影響，算術式の構文解析，OS の役割と処理方式，AI やソフトウェア危機，計算機の信頼性と安全性と，情報システムの本質的な問題点を具体的な例をあげて説明している．Computer Science の概説書として類を見ない教科書であるため，30 年以上たった現在でも，現役の教科書として売れ続けている．

これに対して，森口教授と阿部教授に続く標準的なプログラミング教科書が発行されていないように見うけられる．翻訳本には良書を見るが，日本人の著書に，どのようにプログラムを作るかということを論じた本はあまり見当たらない．その理由は，日本の大学におけるプログラミング教育が，「FORTRAN 入門」と変わらず，プログラミング言語の教育が行なわれているからであろう．これは，初期の自分でプログラムを作らなければならない状況が変わって，ソフトウェア商品の充実にともなって，自分でプログラムを書く必要性が徐々になくなってきたからかもしれない．もはや，プログラムを書く必要が一般ユーザーには必要なくなったので，ソフトの使い方を教えればよい時代になったのである．これにともなって，プログラミング入門の教科書は作り方を解説するのではなく，言語の説明に逆行してしまった．

情報処理学会では 1992-93 年度に，文部省の委託を受けて「一般情報教育」に関する研究を行った[8][9]．その中で，「プログラミング」について

特定の実用言語の習得だけを目的とするものではなく，問題を発見して，それを解決するシステムを創り出し，さらに出来上ったシステムの使用を通じて新たな問題を発見するという，いわばシステム進化の過程全体であると考えている．そこには，構造化や抽象化というような computer science の基本概念が必要とされる．

とする結論を出している．残念ながら，その後の日本におけるプログラミング教育は，プログラミング言語教育に後退してしまった．

大学生になってから「プログラミング」を教えても，効果があがる人は多くないということが世界中で認識されるようになった．また，どこの国でもプログラマーの数が足りないという状況が起こっている．その結果が小学生からの「プログラミング」教育ということになったのである．そこで行なわれるべき教育はプログラミング言語教育ではない．

ところが，驚くべきことに日本では今に至るまで「プログラミング」教育がプログラミング言語教育にとどまっている．それどころか，その方法が，サンプルプログラムをそのまま入力して実行し，その結果を確認するという「写経型学習過程」による教育が一般的になっているようだ[15]．

写経型学習の学習過程は，

（１） 作業を進める（自らプログラムを記述し，実行する）
（２） 実行結果（具象）を認知する
（３） 具象を抽象化し，理解する

という過程から成るという．どうやら，他人の書いたプログラムを打ちこんで実行してみて，その結果を「抽象化」して理解することがこの教育の目的であるようだ．

プログラムを実行して，その結果と書かれたプログラムの関係を考察してプログラムが書かれた言語の意味を理解させるということのようであるが，写経の前に言語に関する解説はあったはずであるから，いったい何が抽象する活動なのか理解に苦しむ．教育方法論として，森口教授の言語教育から大きく後退している．

先進国では情報教育の教師育成が確立していてそこでは computer science と教育学が教えられている．例えばオーストラリアではそのための学術誌 Australian Educational Computing が 1986 年以来, 年２回発行され続けていることから, 情報教育の専門家育成をすでに 30 年間行ってきたことが分る．

3. プログラムの設計

数値計算を行なう場合，プログラムの設計は問題を数式化するだけで殆ど済むので，あまり問題にならない．このため，「FORTRAN 入門」では言語の解説が中心となった．しかし，Computer Science の基礎としてのプログラミングは，複雑な構造のプログラムが作れる教育を行なう必要がある．その第１歩は，逐次処理，繰り返し処理，条件分岐

処理を理解し，使いこなせるようにすることである．そこで使われる設計法の一つとして，構造化プログラミングのための設計法として開発された HCP チャートがある[6].

HCP チャートは，構造化プログラミングが導入された時に，流れ図の代替として日本で提案された多くの設計図法の一つである．その基本的な考え方は，作りたいプログラムの目的を手段に展開した結果を図示することである．展開された手段は，それを実行する順に縦に並べて記述する．さらに，展開された手段はそれぞれ，次の展開の目的となる．このような展開を数回続けて行なうと，最後は実行されるコードとなる．HCP チャートを用いて具体的なプログラムの設計を行なうことで，抽象化という Computer Science の最も重要な概念を教育することができる．

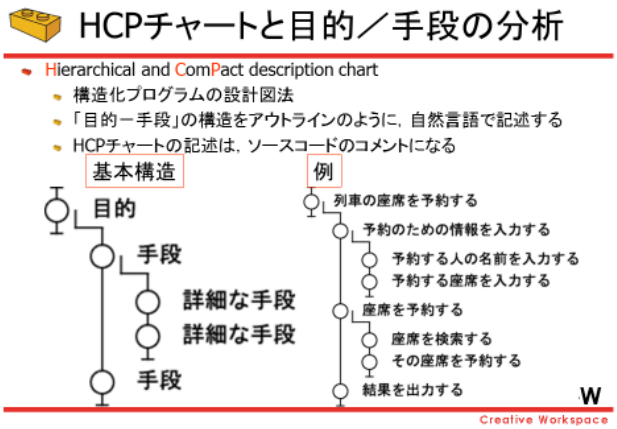

図 1　HCP チャートの説明

Figure 1　Hierarchical and Compact description chart.

このような展開を行なうと，右側に行く程具体的な記述となる．逆に左側は抽象的な記述ということになる．そして，同じレベルの手段の抽象度がそろっていなければならない．もう一つこの展開で問題となるのは，ある目的を実現するために展開された手段の，粒度がそろうことである．

例えば，タートルグラフィックスで三角形と四角形で家を描くプログラムを考えてみる．家を描くという目的が，屋根と本体を描くという手段に展開される．さらに，屋根は三角形で，本体は四角形で描くと，設計を展開できる．

このような抽象的な目的を具体的な手段に展開することは，構造化プログラミングが提唱された時には「段階的詳細化(stepwise refinement)」として広く知られていたが，今ではこの語は使われることがない．

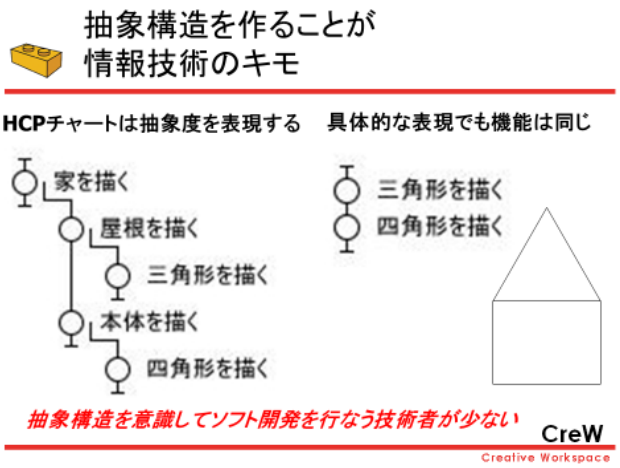

図 2　家を描く HCP チャート

Figure 2　HCP chart for drawing a house.

初心者はプログラムを作ろうとしても，何から始めてよいか分らない．今作ろうとしているプログラムが何をしようとしているか，それを実現するために何が必要かを確認した上で制作作業を始めれば，自分のやっている作業が全体の中でどのような位置づけになるかが分る．

うまくプログラムが作れない人はプログラムの全体像を把握せずに，自分が作れそうな所から作業を始めるので，途中で行き詰まってしまったり，それまでの作業が無駄になったりする．

HCP チャートでは，逐次処理全体にその目的を書かなければならない．また，その一つひとつの処理は繰り返し処理であったり，条件分岐処理であったりするが，それぞれに対して記号が用意されていて，その右にその処理の目的を書くようになっている．

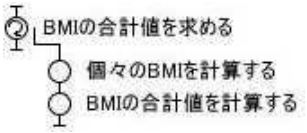

図　3.1　繰り返し処理の HCP 記述

Figure 3.1 HCP description for repetition

図 3．2　条件分岐処の HCP 記述

Figure 3.2 HCP description for conditional branch.

逐次処理も，繰り返し処理も，条件分岐処理も，流れ図にすると全部，入口が１つ，出口が１つの構造をしている．

従ってそれぞれの処理の中に含まれる個々の処理に，これら３構造の処理を埋め込むことができる．この性質を利用することで複雑なアルゴリズムを表現することができる．結果としてできたプログラムの流れ図では，動線が交差することがないので，スパゲッティプログラムになることが自動的に避けられるのである[4].

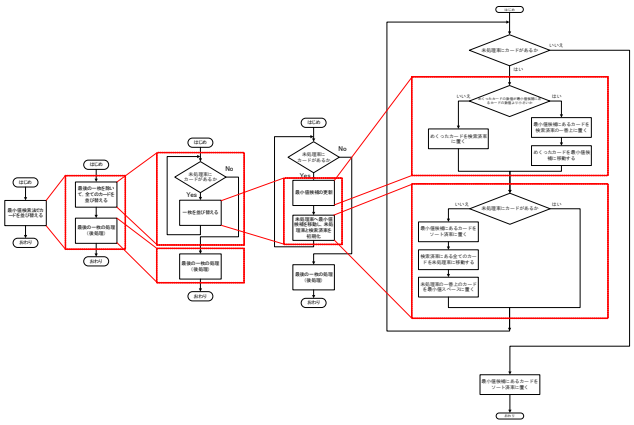

図４．制御構造の埋め込みによるアルゴリズム構築
Figure 4. Algorithm Construction by Nested Structure.

構造化プログラミングで複雑なプログラムを作るには，埋め込み構造の利用することになるが，これを系統的に行なうには，目的を手段に展開するという HCP チャートの考え方に従うのが自然である．

ここまでが，意味のあるプログラムを作るのにまず学ばなければならない第１歩であるが，さらに次に，一連の処理に名前をつけて手続きとして抽象化し，この手続きに対して入力と出力の情報を受け渡す方法を学ぶ必要がある．逐次処理内の複数の処理の，どこからどこまでを手続きとしてまとめるか，ということは難しい問題である．これがうまくできると，ソフトウェア部品が作れることになる．

現在はこのようなソフトウェア部品が供給されるようになったが，これを使ってソフトウェア開発が行なえる設計技術者が日本には非常に少ないようである．これは，大手 SI 企業が開発業務をスクラッチ開発と呼ばれる，プログラミング言語を直接使った開発を行ってきたために，自社外の部品を使った経験がないためであろう．また，技術者が系統的なプログラミング教育を受けてこなかったことの影響も大きいと推測される．

4. HCP チャートの効用

HCP チャートは，プログラミングができなくても読むことができる．このため，実務の場合には発注者と制作者の合意形成に役立つ．また，プログラムの全体構造が一望できるので，これを理解した上で細部についての理解が得られる．この結果，共同作業で制作する場合には，制作メンバー間で共通の理解を共有することが容易になる[7].

目的を手段に展開するという概念は小学生でも理解できるので，今後協調学習の場面で広く使われであろう．実際，この考え方はプロジェクト活動一般に適用可能であり，マネジメント教育に関心のある人々から関心が寄せられている．

しかし，概念が分り易いことは，作り易いことにはつながらない．ソフトウェア制作の新入社員教育を行なう中堅社員に HCP チャートの教育を行ったことがあるが，意味のあるチャートが作れる人達と，形は整っているが問題が多いチャートしか作れない，少なくとも指導には適さない人達の２つにはっきりと分れた．所属企業の知名度とは全く関係がなかった．

この設計法は電々公社の研究所で開発されたことから，NTT 関係のソフトウェア製品の納入には設計文書としてこのチャートの提出が義務づけられた．しかし，設計法の教育がしっかり行なわれていなかったために，ほとんどの納入企業は製品が出来上ってから納入のために形だけの HCP チャートを作ることが強制されることとなった．

しかし，デンソーでは今に至るまで HCP チャートを使って設計を行なう努力を続けているようである．

5. 情報産業におけるプログラミング教育

プログラミング教育がプログラミング言語教育にすぎないという状況は日本の特殊事情である．1970 年代に，IBM がソフトウェアの価格分離を行った結果，日本 IBM の技術者はプログラムを書く仕事が高くつくので出来なくなり，プロジェクト管理だけを行なうようになった．その他の企業もこれにならって，プログラミング作業は外注することが一般化した．

外注先の企業は体力がないので，言語教育でない「プログラミング」教育を行なうことはできなかった．本来，「プログラミング」教育は大学などの教育機関が行なうべきことであるが，日本の高度成長の時期の増大する需要に応えることは全くできず，情報産業は自前で教育せざるを得なかったのである．それが可能であったのは，日本の教育の一般的な質が非常に高かったからであろう．

ソフトウェア産業におけるプログラミング教育がこうしたものであることから，これによって育った IT 技術者は与えられた仕様に従ってプログラムコードを作るだけのコーディングしか出来ず，部品が豊富に提供される時代になった現在，IT に職を奪われることになりつつある．

金融機関の基幹ソフトウェアは COBOL で作られるが，日本では COBOL のコードを銀行員が日本語で記述し，それを”プログラマー”が COBOL に翻訳するという作業形態が取られてきた．

このことを知った時に唖然としたが，アプリケーションプログラムの本質を知る人が，自分の「ことば」でこれを記述し，それを“プログラマー“がコンピュータの実行できるプログラミング言語に翻訳するという作業形態は，悪くないとも考えられる．本質を知った人が自分のことばで

それを記述するのは，これからの「プログラミング」を先取りした開発方法とも考えられる．

現在の技術をもってすれば，日本語を COBOL に翻訳することはコンピュータがする仕事であり，コーダーにすぎない金融”プログラマー”が失業するのは当然のことである．

日本の情報産業は設計できる技術者が１人でできる仕事を，”プログラマー”が 10 人がかりでやってきた．この結果，日本のソフトウェア価格は国際価格の．10 倍になっている．

このような状況が起こるのは，日本人は見栄えには大変神経質で気にするという特質があり，それが日本語の壁に守られて，外国の情報産業の進出が困難であったという事情により，これまで温存されてきたからである．かつて世界第２位の経済力がこれを可能にした．

しかし，経済力の低下でこの高コストを支えきれなくなったこと，米国製のクラウド環境が 1/10 の低コストで提供されるようになったことが日本の情報環境を大きく変えつつある．

設計ができる技術者が必要なのであるが，その数は業界の識者に聞くと IT ゼネコンと呼ばれる大手企業では１％以下であると言う．

設計ができる技術者がいれば，コストの問題は解決できるが，それがもし１％しかいなければ，99%の情報技術者が失業して，日本のソフトウェア開発力が 1/10 に削減されることになる．

大手企業には１％しかいないとしても，外注先の中堅企業にはいる可能性がある．外国人技術者を活用することと合わせて 10%の設計できる情報技術者が確保できたとしても，90%の情報技術者が失業することは避けられない．この問題の解決は困難であるが，その解決策の一つはＳＥと呼ばれる日本の情報技術者に「プログラミング」教育を実施することであろう．

日本の情報産業は壊滅の危機にあるようだ．大事なことは，その実態を調べることだが，その手段も確立しておらず，調べることもできない．

6. 日本語プログラミング

日本語の語順は目的語の後に動詞が来るため，複雑な処理を記述する場合に，記述者は目的語のスタックだけを脳内に用意しておけば自分が実行したい動作をイメージできる，逆の語順の英語の場合は，動詞のスタックも必要となり，脳の負荷が大きくなる．

このことから，米国人が APL, FORTH, Post Script など，日本語の語順の言語を作ってきたが，母語の英語と語順が異なるため普及しなかった．日本語はコンピュータとのインターフェースがすぐれた言語なのである．

例えば，（２＋３）×４という計算を日本語では「２と３を足して，４を掛ける」と簡単に言えて，その計算を言いながら脳内でイメージできるが，英語では数式全体を思い浮かべて，全体像を思い浮かべながら multiply added sum of 2 and 3 by 4 と言うのであろうか．

FORTH を輸入販売していた片桐 明が, FORTH を日本語化するだけで日本語プログラミング言語になったことから，これを日本語プログラミング言語 Mind として商品化した[10]．この言語は，まず教育用として使われた所，Logo に較べて格段に学習効果が上がることが確認された(西之園晴夫：私信).

Mind は単に教育用言語であるだけでなく, ソフトウェア開発言語として発展し，例えば「ぐるなび」の全文検索エンジンは，Mind の発展形の MindSearchII を使って開発が行なわれ, そのシステムは 2004 年５月から６年間使われ続けた[10].

Mind の特長として, 修得がＣ言語の約３分の１の時間で実務アプリケーションが組めるようになるという．また，可読性が高く，再利用がし易いので，開発期間が短縮できるようだ[10].

日本語でプログラムを作ることを提案すると，国際化時代に逆行すると言われる．しかし，この問題は，必要なら作ったプログラムを読者が読める言語に翻訳することにすれば解決できる．人工知能研究として機械翻訳が研究されたが，人間は膨大な常識を持って言語を使っているため，この扱いが難しくて実用化に至らなかった．

しかし，形式的に意味が規定されているプログラミング言語の場合にはこの問題が起こらない．従って機械翻訳の研究成果を利用すれば，プログラミング言語間の翻訳は十分可能であることが予想される．実用言語で書かれたプログラムを自然言語に翻訳することは，プログラミング教育にとどまらず，使われる可能性がある．

「日本語は論理的でない」ということから，プログラムのような論理的な事象の記述には適さないと考える人も多い．しかし，これも思いこみに過ぎない．どんな言語を使っても，使用者が論理的でなければ，表現された文章は論理的でない．高等教育に至るまで日本語だけで国を維持している日本語が論理性に欠けることは考えられない．

科学ジャーナリストの松尾義之は，ノーベル賞講演を行なうまで外国に出たことのない益川俊英教授が日本語で物理学を研究してきたことを例にあげて，日本語が日本における科学研究の進歩に大きな影響を与えていることの主張している[11].

最近米国人のロジャー・パルパーズが，日本語は少数の語彙と単純な文法で豊かな表現ができることから，英語より世界語としての可能性が高いという指摘をしている[12].

少数の語彙と単純な文法はプログラミング言語の特徴であり，日本語は，プログラミング言語と同じような特性を持っていることになる．

7. 母語によるプログラミング

慶応大学湘南藤沢キャンパス（ＳＦＣ）では，1990 年の開学以来，外国語教育，保健体育とともに３つだけの必習科目の１つとして情報科目が８単位，全学生に義務づけられてきた．情報科目の内容はプログラミングを中心とするものであるが，全員が自分の目的を実現できるプログラミング能力を獲得出来たわけではなかった．

そうした状況の中で，プログラミング言語を日本語にすることで，クラス全員が挿入ソートを，フローチャート作成による設計から始めて作れるようになったことが杉浦らによって報告されている[13]．彼らの授業では，週９０分２コマの授業を１３週にわたって行っていたが，その前半に「ことだま on Squeak」[14]による日本語プログラミング教育を行ったことによって，目的とするプログラムを作れたものと推測される．授業の後半は,「ことだま」で教えられた内容を Java で繰り返すことによって，実用言語においても，前半の授業で獲得された論理構築能力が,生かされたことが報告されている．

総合政策学部２年の男子学生は次のレポートを残した．

＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝

プログラミングには（中略）論理に基づき筋道だったプログラミングをする以外の方法はない．その意味でプログラミング自体に関する知識や表現の使用を多く求めない「ことだま on Squeak」は，思考訓練の面では大いに有効であったと思うのである．（中略）プログラミングに関する文法や表現を詳しく知らない段階では，「ことだま on Squeak」を使うことで，すでに用意された表現を使用しながら，より本質的な論理思考の訓練に専念できたと感じている．（中略）読解が出来ない状態で作文など無理であるように，表現や文法を知らない段階で Java によるプログラミングは大変なハードワークであろう．私は初期段階において「ことだま on Squeak」を通して「順次」「分岐」「繰り返し」「変数」などの基礎的な思考方法を重点的に学ぶことができた．そこで得た思考能力が，他者が作成したプログラムを読解し理解する際に活用されたと思う．

＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝

従来ＳＦＣでは，授業全体を実用言語の Java（やC）で行ってきていたのであるが，９０年代後半以後は，いわゆる学力低下のせいか，自分の目的を実現できるプログラミング能力を獲得する学生が減少して，プログラミングが習得出来るのは入学前からその能力を持っている学生だけに，ほぼなってしまっていたようである．

実用言語を使って初心者にプログラミングを教えると，

（１）「新しい言語を理解すること」

（２）「それを使って仕組を作ること」

という２つの初めての作業を同時に行なわなければならない．殆どの受講者は，新しく学ぶ言語の文法通り，プログラムを書く段階で挫折してしまう．最近の日本の教育は正しいことを覚える教育に終始して，試行錯誤して何かを作り出すような教育が行なわれなくなってきたため，文法エラーが頻繁に起こることだけで挫折してしまい，授業は文法通り書くことの訓練で終ってしまう場合が多い．

実用言語は実用目的を達成することを目的とした，専門家を対象として言語であるので，実用プログラム作成を効率的に行なう工夫が多く成されている．このため，初心者にとって意味の理解できない作業を多く強いられることになってしまう．

もう一つの問題は，実用言語を将来使う可能性である．一般教育の受講者で，将来プログラマーになる人は少数であろう．今後はさらに，高度な技術者だけが生き残る時代になってきたために，プログラマー自体の人数が減っていくことが予想される．従って，現在の実用言語を学習者が使う可能性は少ない．

一方で，論理思考が必要とされる人数はこれから増大していくことが予想される．情報技術の応用分野がますます広がるからである．コンピュータの導入で，社会生活で使う情報システムが複雑な作業を行なうようになり，使い方を丸暗記するのでは対応できなくなって，論理的に考えて使う必要性が高まることが予想される．こうした論理思考能力を育成する手段として，プログラムを作ってみる体験は大変有効である．

論理思考教育を行なおうとすると，プログラミング活動の中で，アルゴリズム構築が一番役に立つ部分である．従来のプログラミング授業では，ここに到達する前に，文法理解で終ってしまっていたために，社会に出て役に立つ教育が行なわれてこなかったのである．

アルゴリズム構築の教育においては，従来は疑似コードが使われ，それは母語としての日本語が用いられてきた．疑似コードは実行できないので，それを実用言語に翻訳しなければならない．すると，文法エラーを退治できたとしても，続いて実行時エラーを退治する必要がある．それが終ると，予期したようには動作するとは限らない．これを行なおうとすると，慣れない実用言語を正確に解読しなければならない．これが初心者には難しいため，あてずっぽうでプログラムをいじり出して，試行錯誤で動かそうとする．これでは論理思考の教育にはならない．

8. 「プログラミング」教育の社会的意義

文科省が推進する「21 世紀型能力」の育成の中心課題は問題解決力，発見力，想像力，論理的・批判的実考力，メタ認知・適応的学習力である．これらの能力の育成には HCP チャートが方法論として役立つが，そこでチャートの表記に使われる言語は母語としての日本語であり，表記に至る思考活動も日本語で行なわれるはずである.「言語能力

の向上」は次期指導要領改定の重要課題であるが，設計活動を中心とする「プログラミング」教育はこの課題解決に大きく貢献できるはずである．

「プログラミング」教育の実施も，現行の指導要領で決められた科目の中で行なわなければならない．日本の教育は日本語で行なわれるのである．例えば，算数の教科書を見てみると，数の計算を訓練することがその中心となる．しかし，その方法は図解による説明は行なわれているが，アルゴリスミックな日本語による記述は行なわれていない．

教師が直接口頭で生徒に計算を実行させる部分を日本語プログラミング言語で置き代えることで，従来の教育の枠内でプログラミング教育を行なうことができるはずである．このプログラムをコンピュータ上で実行し，さらにプログラムを書きかえることで，実行がどのように変わるかを観察すれば，算数の理解が深まると同時に，プログラムに関する理解も深まるはずである．大学生で可能であった日本語「プログラミング」教育は，柔軟な小学生なら十分可能なはずである．

しかし，算数の演算教育を日本語で行なうことは，検討してみると簡単ではないことが分った．まず用語が整理されていない．帯分数に対応する帯小数があることを初めて知ったが，真分数に対応するのは純小数であった．２桁以上の整数の計算を行なうには，各桁の数を入れる変数が必要になるが，これを示す用語をどうしたらよいかも，適当な用語がない．

もちろん，適当に英文字を使った配列を使えばよいが，それを行なうのは中学校の代数を学んでからになるであろう．演算の過程を，プログラミング言語を使って書いてみることは，この段階の方がよいかもしれない．

9. おわりに

情報技術者の教育を日本は怠ったことから，日本の情報産業は壊滅の危機にあるが，その事は殆ど認識されていない．小学生からのプログラミング教育が始まると，一般人の情報リテラシーが上がり，情報産業の在り方も大きく変わるであろう．

プログラミング入門教育で一番困難であることは，学習者が自分の書いたプログラムの意味を読みとることである．これが，日本語でプログラムを書くことで解決される．これは，日本語の正確な読解を要求するので，言語能力向上の手段としても重要な教育方法なる可能性がある．

日本語でプログラムを記述することは，金融機関で何十年も行われてきたことから，実用的なプログラミングの方法としても十分な実績がある．

日本語の語順は，コードを発想する時に，動詞を記憶する必要がないので，脳への負担が少ない．声を出して読める日本語でアルゴリズムを記述する方法を研究する必要がある．

プログラム部品が潤沢に用意される時代が来ている．これを IT 技術者に頼らず, 自分で使いこなせる方が知的活動の生産性が上がる時代が来た．

「プログラミング」教育の意義を再認識し，自分が情報システム上で行ないたい仕事を利用可能なソフトウェア部品の利用で実現できるようになることが求められるようになるであろう．目的を手段に展開する設計教育を中心とした，日本語によるプログラミング教育を確立することが必要になるであろう．

参考文献

[1]森口繁一. FORTRAN 入門（第２版）. 東京大学出版会, 1973.
[2]森口繁一，小林光夫，武市正人. Pascal プログラミング対話. 共立出版, 1980.
[3]森口繁一，小林光夫，武市正人. Pascal プログラミング講義. 共立出版,1982.
[4]阿部圭一. ソフトウェア入門. 共立出版, 1983.
[5]小野厚夫， 川口正昭. 情報科学概論. 培風館, 1983.
[6]花田収悦. プログラム設計図法. 企画センター, 1983.
[7]松澤芳昭，杉浦学，大岩元. 学習者同志の相互レビューを通したプログラミング設計教育，情報教育シンポジウム 2004 論文集, pp.189-193, 2004.
[8]大岩元. 一般情報教育. 情報処理, 1991, Vol.32, No.11, pp.1184-1188,
[9]大岩元. 一般情報教育のカリキュラム. 情報処理学会研究報告コンピュータと教育 21-4, pp.29-38.
[10] 片桐 明. 日本語プログラミング言語 Mind. http://www.scripts-lab.co.jp/mind/whatsmind.html
[11]松尾義之. 日本語の科学が世界を変える. 筑摩書房， 2015
[12]ロジャー・パルパーズ著早川敦子訳. 驚くべき日本語. 集英社インターナショナル， 2014
[13]杉浦 学，松澤芳昭，岡田 健，大岩 元. アルゴリズ構築能力育成の導入教育：実作業による概念理解に基づくアルゴリズム構築体験とその効果. 情報処理学会論文誌, Vol.49, No.10, 2008
[14]大岩 元(監修), 松澤芳昭・杉浦 学(編著). ことだま on Squeak で学ぶ論理思考とプログラミング. イーテキスト研究所, 2008, http://crew-lab.sfc.keio.ac.jp/lectures/2011s_ronpro/data/Squeak/Text/SqueakText.zip
[15]岡本雅子，はじめてのプログラミングとつまずき，情報処理, Vol.56, No.6, pp.580-583, 2015.