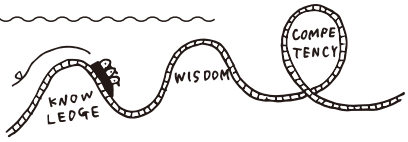
べた語義

Vol. 90

CONTENTS



COLUMN

「知識」「知恵」「コンピテンシー」―活躍につながる 3 つの基本能力―

基般 Jr.

ジュニア会員の皆さん，将来，技術者や研究者として活躍したいですよね．きっと，自身が活躍する姿を思い浮かべ，日々，学習に取り組み，自己の研鑽に励んでいると思います．

そんな皆さんに，当たり前で，でも，忘れがちな，あるいは，気づかずにいてしまう話をお伝えしましょう．

皆さんは，学生の間に，どのように成長しないといけないと思っていますか？

多くの人は，「知識」を増やすことが大事であると返事をしてくれると思います．

もちろん，「知識」は大切です．では，「知識」とは何でしょう？

知識は，『文章や図表，数式で表現可能で，人が認識，認知した内容』といえます．

自分が活躍する分野である情報工学，情報科学の知識を，より広く，より深く理解し，自分のものとすることは，活躍の糧になります．でも，「知識」の獲得にだけ意識を向けていてはいけません．「知恵」と「コンピテンシー」を育むことも大切です．

いくら「知識」を持っていても，それを使いこなす能力がなければ，Web 検索と同じです．それどころか，検索エンジンに，知識の量で負けてしまいます．人間が機械に優る点の 1 つは「知恵」です．「知恵」は，『道理を判断し，処理していく能力．適切なときに，適切な方法で，適切な知識を利用する能力』です．「工夫」する能力といってもよいでしょう．ちなみに，「工学」の「工」は，「工夫」の「工」です．人類の「知識」を，世の役に立つように活用する工夫を行うのが「工学」です．工夫する能力である「知恵」を鍛えることも忘れないでください．

「コンピテンシー」は，『高い業績を上げ，活動を成功に導く行動特性』です．「行動特性」は生まれつきのものではありません．歯磨きをする赤ん坊はいません．行動は，初めは意識しながらであっても，継続すると，「無意識の振る舞い」となります．「コンピテンシー」は，いろいろなところで論じられており，必要な知識を自ら獲得するという行動や，自分の考えを伝える行動，チームメンバとして振る舞える特性など，いくつかに分類されています．情報処理推進機構「実践的講座構築ガイド 評価基準編」（https://www.ipa.go.jp/files/000056678.pdf）の表 2-5 は，参考になると思います．

「少年老い易く学成り難し」，「光陰矢のごとし」．将来に向けて，学生の間に「知識」，「知恵」，「コンピテンシー」をしっかり伸ばしてください．

小林 真也（愛媛大学大学院理工学研究科）

LOGOTYPE DESIGN...Megumi Nakata, ILLUSTRATION&PAGE LAYOUT DESIGN...Miyu Kuno

ARTICLE

基般

プログラミング入門科目の指針と実践例(前編)

久野　靖
電気通信大学

大学におけるプログラミング入門科目の課題

情報社会の今日において，プログラミング教育は「コンピュータの基本原理にかかわってる」という理由からか，かなり多くの大学で教えられている．特に理工系の教育課程においては，「プログラムを組んで計算を行えることが必要な素養である」と課程設計者が考えるのか，初年次においてプログラミングを必修・準必修としている場合も多い．

しかし現実には，そのような科目はいろいろな問題を持つことがある．その最大のものはずばり，「プログラミング科目で単位を取得したのにプログラミングができない」ではないだろうか．筆者も，そのような現実をいくつか見聞したことがある．

それはなぜか，それを避けるためにはどうしたらよいか，というのが筆者にとっては長期にわたる問いだった．この問いに対する解を思いついたとして，それを実際にやってみるというのは簡単ではない……はずなのだが，好運にも「理工系大学の初年次情報教育の責任者」という職に就くこととなり，実際に「やってみる」ことができている(しかも大規模に)．本稿はその報告である（分量の都合から前編と後編に分けている)．

電気通信大学の初年次情報教育

電気通信大学は東京都調布市にある理工系の単科大学であり，初年次情報教育として前期に「コンピュータリテラシー」，後期に「基礎プログラミングおよび演習」(Fundamental Programming という英語名であり，以下 FP と記す）を，いずれも 2 単位の必修科目として開講している．

両科目とも全学必修のため受講人数は 800 名前後であり（過年度生を含み，既習得単位認定者は除く)，13 クラス(うち夜間課程 1）に分かれて実施している．初年次科目にしては珍しく，担当者すべてが本学の専任教職員となっているが，これは本学が情報系の比率の高い大学であることによっている．

このうち前期の科目については，その名称にもかかわらず，一部（15 回中 2 回）プログラミングを体験してもらう内容を盛り込み，好評を得ていることを別稿において報告した[1), 2)]．

しかし本題は後期の科目である．この科目は「簡単なプログラムの作成と読解ができる」ことを目標にしているが，筆者が 2016 年度に着任した時点では期末試験時においても「プログラミングができない」学生がかなり存在していた☆1．そこで 2017 年度から，教材や授業運営を大幅に見直し，「できない」学生をできるだけ減らすべく努力中である．その具体的な設計・実装は，筆者が上記の「問い」に対して「こうしたらよいはず／こうするべき」と考えてきた指針群に基づいている．

そこで以下では，指針を 1 つずつ挙げ，その内容，およびそれをこの科目でどう実現しているかについて説明させていただく(前編の内容)．後編では科目の具体的内容と個別の内容に即した配慮・工夫と結果について紹介する．

☆1 試験もそのことに配慮して，持ち込み可・資料閲覧可で，文法などの知識を問う設問なども交え，なんとか再履修が多すぎないようにしていた．

❑ 指針 1：離陸ファースト

離陸[3]とは「自分の考えたことをプログラムにして動かせる」ことを指す．それは最終目標であり指針ではないのでは，と思われるかもしれないがそうではない．

多くの教科書や授業では，プログラムの書き方の規則や対応する動作などを延々と説明してから「では書いて見ましょう」という段階に到達する．しかし「書いてみる」時点までに学んだ（というより単に提示された）内容が多ければ多いほど，学生はその中で何が大切で何はさほど重要でないか分からず，途方に暮れてしまう．そのような学生ができることは「例題をそのまま写して動かす」くらいであり，到底離陸はできない（図 -1 の下側の経路）．

そこで FP では，できる限り簡単な（しかし意味はある）例題をまず説明し，すぐにそれを動かしてもらい，それを土台に演習を行う．このために使用言語も C 言語から Ruby に変更した（科目の前半 2/3）．最初の例題を図 -2 に示す☆2．

初回はこの例題を一通り説明してから各自にも動かしてもらい，その程度でできる演習をすぐ実施する．演習の課題は「和・差・商・積」「剰余」「逆数」「円錐の体積」「8 乗」などである．たとえば最後のものは「乗算演算のみでその回数を少なく」と注記してあり，図 -3 が想定解となる．つまり，この程度のボキャブラリでも十分考えさせる演習は行える．

☆2 実行は irb（Ruby 用の read-eval-print loop）を用い，コードをファイルに書き load して実行する操作だけ教える．

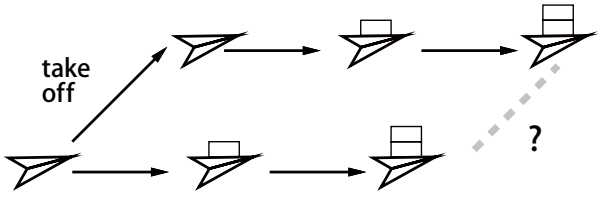


図 -1　離陸の概念

```
def triarea(w, h) #3 角形の面積
  s = (w * h) / 2.0
  return s
end
```

図 -2　最初の例題：3 角形の面積

すなわち「離陸ファースト」では，まず簡単な内容で離陸してもらい，離陸の状態を維持したままで徐々に内容を学び増やしていく（図 -1 の上側の経路）．こうすれば，学んだことはすぐ試せ，演習問題で考える機会が持てるため，身につきやすい．

❑ 指針 2：多様な水準の演習問題

プログラミングに限らず何事も，演習するときにはその難しさ（負荷）が適切でなければ成長（学び）は起こらない（図 -4）．しかし，プログラミングの授業では，全員に同じ演習問題に取り組ませるものが多い．もちろん，プログラミングの腕前（や素養）は人により大きく違っている．そのため，その 1 つの問題は多くの（下手をするとすべての）学生にとってやさしすぎか難しすぎであり，効果的な演習とならない．

そこで FP では毎回，テキストの演習問題の数を多くし（1 回あたり 10 個程度），好きなものを 1 つ以上選んで実施し，結果をレポートとして提出するよう求めている．問題はおおむねやさしいものから難しいものの順で並べてあるので，実力のある学生はさっさと後の問題まで進み（明らかに簡単だと思えばパスしてよい），そうでない学生はやさしい問題に取り組む．これにより，多くの学生が自分に合った難しさの問題で演習をできるようにしている．

手抜きでやさしい問題ばかり選ばれると危惧されるかもしれないが，実際には実力ある学生は簡単な問題は「つまらない」ため，多くの学生が能力一杯く

```
def pow8(x)       #x の 8 乗
    x2 = x*x; x4 = x2*x2; return x4*x4
end
```

図 -3　x の 8 乗のコード例

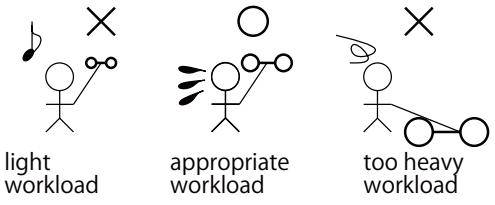


図 -4　適切な負荷

らいの課題にチャレンジしている（体調その他の都合によりやさしい問題を選ぶことは認めている）．

取り組む問題が異なるのは不公平という意見もありそうだが，科目の目的は有効な学びであり公平ではないし，それぞれの実力に応じた練習をすることが公平だともいえる．

❑ 指針３：演習の重視

この科目は週１回 90 分の授業であり，授業時に講義をしていたら演習の時間はほとんどない．筆者が着任した 2016 年時点の授業はほぼそういう状況で，演習ゼロで授業終了となることも多くあった．そこで起こるのは次のことである☆3．

> 学生は時点 X までは「やさしい内容だと馬鹿にして」演習せず，時点 X で突然「内容が難しすぎて落ちこぼれた」と気づく．

これを防ぐにはとにかく「全員が毎回演習する」科目設計が必要である．そこで，テキストと講義ビデオを用意してあらかじめ公開し，予習を義務づけた上で，担当教員には「最低限の説明にとどめる」よう依頼することで演習時間を確保した．

また，各回とも演習問題を「1 つ以上」選んでプログラム・解説・考察を含むレポートとして提出することとした．その点数合計は 50 点あり（試験も 50 点満点），レポートが出ていなければ単位は取れない．演習してプログラムが作れていなければレポートは出せないので，学生は毎回演習に取り組むようになった．

レポートは「説明や考察を重視」するとしたので，プログラムだけ写して出しても 0 点であり，自分で書いた（少なくとも説明してもらって理解した）ものでなければ意味がない．元々学生がプログラムを写すのは「難しすぎて書けない」からであり，やさしい問題（初回の例では 2 つの数値の和など）が選べるようにすればまず写すことはないというのが筆者の経験である（そもそもテキストには毎回「前回演習問題の解説」を掲載しており，写すとすればそれを写している．しかしそれでも説明や考察は自力で書かないとレポートにならない）．

☆3 もちろんこの「時点 X」は学生によってまちまちである．

❑ 指針４：苦手な人の学びを促す

多くのプログラミングの科目では，前述の「同一の問題」のせいもあり，できる学生が労せずしてよい成績を取り，苦手な学生は否定的な感情を持つ．初年次の必修科目としては，苦手な学生が最大限成長できることを目標にするべきではないだろうか．

このため FP では毎回のレポートを ABCD の 4 段階で評価し，D は不受理 C は明らかな不備，A は特に優秀，それ意外は B（通常）とし，すべて B のとき 50 点（満点）とした．苦手な学生も普通にレポートを出していれば満点となるため，前向きに取り組んでもらえる効果があったと考える．

A は 50 点から上積みされる（上限 59 点，単位は 60 点以上）ため，できる学生は必死に A を取ろうとするが，A は 2 ～ 3% にとどめるように担当教員に依頼している．この採点方針では基本的に B をつければよいので，大量のレポートを毎週評価する担当教員の負担軽減にもなっている☆4．

❑ 指針５：プログラムが書けるという目標の明示

世の中の多くのプログラミング科目では（そして 2016 年度の FP も），プログラムが書けなくても単位が取れる．これは試験で，知識問題（構文規則の知識などを問う），穴埋め問題（プログラムが書けなくてもパターンで正解し得ることが多い）を採用するためである．このような「逃げ道」があると，学生が何がなんでもプログラミングできるようになろうとは考えない可能性がある．

FP では試験を CBT（Computer Based Test）で実施するため，問題形式として「短冊問題」を採用している．これはプログラミングの設問に対して，正解プログラムを 1 行ずつばらばらにして誤答と混ぜて選択肢とし，これらの選択肢から拾って画面上で

☆4 しかし多くの教員がまじめにレポートを見てくれて，一定量のコメントを返している．フィードバックがあることは学生にとって大きなモチベーションとなっている．

並べることで正解を構成させるやり方である．

この出題形式は，プログラムが実際に書けない人が正解できる可能性は非常に小さい．毎回の授業において，この形式の「確認問題」を 2 問ずつ提示し，試験時にはその類題を出題することとしている．これにより，多くの学生は「プログラムが書けなければ単位は取れない」と（正しく）認識し，プログラミングの技能の向上に努めるようになっている．

❑ 指針 6：プログラミングに唯一の正解はない

今日の日本の教育は「問題には唯一の正解がありそこに早く到達すると勝ち」という強い刷り込みを作っている．これは社会に出てからさまざまな問題を解決する上でマイナスである．そしてこの刷り込みは，プログラミングの科目においても「テキストの例題の丸暗記」「誰かの正解のコピー」のような意味のない行為につながっている．

FP ではこの刷り込みを打破するため「プログラムの書き方は一通りではない」「自分でよいと思う書き方を各自で見つける」ことを繰り返し述べており，またそれを体現する演習問題を題材としている．

たとえば図 -5 は，「2 数のうち大きいもの」を返すコードである．どちらも動作は同じだが，好みを尋ねると max2a の方が好みという学生が多い．しかし，「2 数」を「3 数」に拡張すると（演習問題になっている），max2a を土台にしたものは if の入れ子になり複雑になるが，max2b を土台にしたものは同じ形の if を並べるだけで簡潔になる．このような例をなるべく多く取り上げることで，書き方の多様性に触れ，それらに対する自分の考え方を持たせるようにしている．

```
def max2a(x, y)
  if x > y then return x else return y end
end
def max2b(x, y)
  max = x
  if y > max then max = y end
  return max
end
```

図 -5　2 数の最大を実現する 2 種類のコード

❑ 指針 7：プログラムは自分の頭で作り出す

プログラミングの科目では，テキストにある例題プログラムがそのまま再現できれば単位がとれるようなものもある．しかしそれでは，実際にプログラミングが必要になったときに「まだない（必要な）プログラムを作り出す」ことができず，役に立たない．

そこで FP では，例題は最小限にとどめ，その例題を元にして「新たな」プログラムを「自分の頭で」考え作り出すことを重視した．各回ともこの方針は共通しているが，その中でも特にこの点を重視した内容が「画像の作成」「動画の作成」である．

これらはそれぞれ「2 次元配列」「プログラム構造の整理」の単元になるが，課題としては自分（たち）の考えた画像や動画を生成するプログラムを作成してもらう．

画像や動画は自分で「このようなもの」と思いつくのが容易であり，かつ個性が出しやすいので，これを課題とすることで自然に「自分の作品を自分で設計し開発する」形となる．実際にこれらの課題は学生が特に熱心に取り組んだという印象である．

本稿では初年次プログラミング教育に関する問題意識と科目の設計指針について説明した．後編では具体的なカリキュラムと内容ごとの工夫，および実施結果について述べる．

参考文献

1）久野　靖：電気通信大学における「コンピュータリテラシー」科目，情報処理，Vol.59, No.10, pp.934-938 (Oct. 2018).
2）久野　靖，江木啓訓，赤澤紀子，竹内純人，笹倉理子，木本真紀子：コンピュータサイエンス入門教育の題材としてのアセンブリ言語プログラミング，情報処理学会誌教育とコンピュータ，Vol.4, No.2, pp.23-36 (June 2018).
3）久野　靖：プログラミング教育／学習の理念・特質・目標，情報処理 , Vol.57, No.4, pp.340-343 (Apr. 2016).

（2018 年 12 月 19 日受付）

久野　靖（正会員） y-kuno@uec.ac.jp

1984 年東京工業大学理工学研究科情報科学専攻博士後期課程単位取得退学．同大学助手，筑波大学講師，助教授，教授を経て現在，電気通信大学情報理工学研究科教授．筑波大学名誉教授．理学博士．プログラミング言語，プログラミング教育，情報教育に関心を持つ．

ARTICLE

基般

小中学校における〈普通教育としてのプログラミング教育〉の展開と課題

紅林秀治
静岡大学

〈普通教育としてのプログラミング教育〉とは

現在，日本では，中学校技術・家庭科（技術分野）と高等学校情報科においてプログラミング教育が行われている[1),2)]．しかし，小学校においては，一部の先進的に取り組んでいる学校以外は，ほとんど行われていない[3)]．一方，諸外国においては，すでにプログラミング教育を含む情報教育の低年齢化が進んでおり，英国では小中学校（5 ～ 16 歳）でのプログラミング教育を 2014 年から必修としている[4)]．そのため，日本においても，小学校 1 年生からプログラミング教育を実践できる可能性は十分にある．しかし，プログラミングは，単なる知識ではなく，国民に必要な教養であるとの見方もあり，小中学校のプログラミングの授業に対して，普通教育として行う意義を明確にしなくてはならない．

普通教育を簡潔に理解するためには，その対極にある専門教育の理解が欠かせない．たとえば，小中学校のプログラミングの授業に対して，専門教育的な見方で教育的な価値を見いだそうとすると，作成しているプログラムの内容やロボット教材の動作については，陳腐でまったく役に立たないもののように見えてしまう．なぜなら，専門教育は，学習者が将来その道の専門家になることを想定した教育だからである．そのため，専門教育を受ける学習者は自分が就きたい職種が明確であり，教員はその職に就くために必要な知識や技能，およびそれらの評価規準を把握できている．

それに対して，普通教育は，将来何になり，どのような職業に就くのか決めていない学習者を対象に教育をする．そのため，学習後に獲得してほしい知識や技能，およびそれらの評価規準は，職に就くためとか，想定される仕事に役立つためなどという観点からは設定できない．つまり，考え方や問題解決能力，あるいは学習対象の価値に気づくといった汎用的な能力や態度の育成を教育目標として掲げることになる．しかし，プログラミングの考え方や問題解決能力，またその価値に気づくことを目標とするとなると，普通教育としてのプログラミングの授業ではいったい何を教えるべきなのか検討する必要がある．

私が見てきた〈普通教育としてのプログラミング教育〉の展開

ここで〈普通教育としてのプログラミング教育〉の在り方を考えるために，私が見てきた小中学校での授業を紹介する．

❑ 小学校 1 年生にスクラッチを用いたプログラミングの授業

授業の目標は，「プログラミング学習を通して，『プログラミング的思考』を育成する」である．具体的には，複数のコマンドを使用しないと解決できない課題を児童に与え解決させる流れである．プログラミング経験がまったくないという児童の実態や，1 年生という発達段階を踏まえ，まずは，マウス操作やキーボード入力（かな）の練習から授業を始めていた．PC の基本操作を学習した後，スクラッチを用いたプロ

グラミングの学習を行っていた．「スクラッチキャットをステージの左上から右下に動かす」という課題を教師側から提示し，その動きを実現するために，動きの組合せを考えさせていた(図 -1)．

❑ 小学校 3 年生に教材用のロボットを用いたプログラミングの授業

イモムシ型ロボット「コード・A・ピラー」を用いて，1 時間の授業を行っていた．「コード･A･ピラー」とは，赤ちゃん向けのおもちゃを販売するメーカが，幼児向けの新世代おもちゃとして開発したプログラミング学習ロボットである．イモムシを作る胴パーツには，直進や右折などの命令コードが内蔵されている．ロボットはそれぞれの命令コードが内蔵された胴パーツをつなぐことにより，イモムシが動作する仕組みになっている．ロボットを教室に 1 台準備し，教員に与えられた課題(障害物をよけてゴールする)を達成するために，命令コードが書かれた胴パーツをどのように接続するか，グループ (4 人から 5 人で構成)ごとに話し合う．各グループには，矢印が書かれたカード (直進・右折・左折) が数枚ずつ分けられている．児童は，そのカードを使ってロボットにさせたい動作を話し合い，グループごとに意見をまとめる．教員は，グループごとに決めた矢印の並びをロボットで順番に実証していった(図 -2)．

図 -1　小学 1 年生のスクラッチでの授業の様子

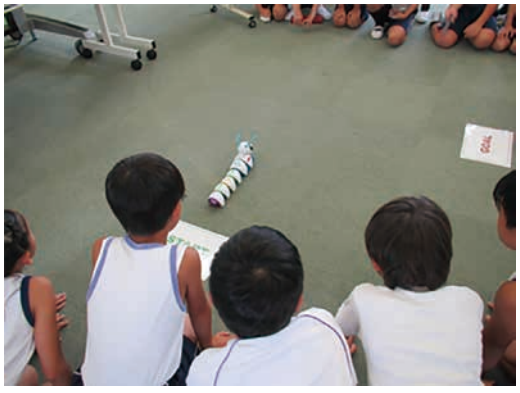


図 -2　イモムシ型ロボットで学習している児童の様子

❑ 小学校 6 年生にドリトルでロボットを制御するプログラミングの授業

クローラ型の移動ロボットの動作をプログラムで作成する．定められたコース通りにロボットを動かすという課題で授業が行われた．プログラムは，ドリトルで作成し，ロボットに転送して実行する．小学校 6 年生くらいになると，キーボード操作にもローマ字入力にも慣れていて，無理なくプログラムを作ることができていた．

❑ 中学 2 年生の Pepper を用いたプログラミングの授業

この授業は，Pepper という人型ロボットを動作させるプログラムを Choregraphe（コレグラフ）というソフトウェアを用いて作成するものである．Choregraphe（コレグラフ)では，必要なロボットの動作やセンサを用いるためのプログラムがすでに揃っており，生徒はそれらを選択してつなげる作業を行う．授業では，生徒からの質問を Pepper に答えさせるためのプログラムを作ることが課題として与えられていた(図 -3)．

図 -3　Pepper を用いた授業の様子

❑ 中学３年生の制御基板を用いたプログラミングの授業

授業で使用した制御基板は，計測値や出力値等をLCDで確認できるものである．また，その制御基板はブレッドボードを利用して自由に入出力回路を作成できる．さらに，制御プログラムにはドリトルを用いているため，制御プログラムをドリトルの編集画面上で作成した後，USB接続した基板に転送する．基板にはアナログセンサを4個，ディジタル出力を6個まで接続できる．計測はPCから独立して行うことができるものであった．LEDの点灯回路やCdsセルを利用したアナログ計測を学習した後，オリジナルなプログラムを作成し発表するという課題を与えられていた(図-4)．

〈普通教育としてのプログラミング教育〉が定着する上での課題

これらの授業では，教員が児童・生徒にサンプルプログラムを示し，そのサンプルプログラムから基本的な知識や技能を獲得する．その後，サンプルプログラムを基に，応用できる課題が教員から出されて取り組むという内容であった．

授業を見学した教員から感想を聞いてみた．さまざまな感想の中から，あえてマイナスな評価の感想の代表例を紹介する．

(1) 結局これは何を学んだことになるの？（小学校校長）

(2) プログラミングって手続きを学ぶことなの？（小学校教員）

図-4　制御基板を用いた計測している生徒の様子

(3) 教材を準備するお金と時間がない．（中学校教員）

(4) これを学んで何かの役に立つの？（小学校と中学校教員）

(5) 試行錯誤をくりかえしていたら，何かできてしまうということが本当に教育としてよいのか？（小学校と中学校教員）

このような教育としての有用性を疑問視する感想は，小学校の英語学習の授業では私が見た限りでは聞いたことはない．私が見た小学校の英語の授業では，基本的に会話を中心に展開されている．ある授業では，児童は英語の言い回しや簡単な単語を覚えてから，教室内の仲間と自己紹介したり，自分の伝えたいと思っていることを教員から紹介された言い回しを用いて話したりしている．授業の中の活動を通じて，児童は新しく覚えた英語を使ってみることに喜びを感じている様子がうかがえた．

プログラミングの授業では新しく覚えたコマンドやプログラムを使って画面上のキャラクタの動作やロボット等の動きを試している．小学校英語の授業も小・中学校プログラミングの授業も基本的に学習の流れはよく似ている．小学生も中学生も新しく覚えた知識を使ってみたい，試したいと思う気持ちは同じである．

しかし，プログラミングの授業では，なぜか有用性を疑問視する声が聞こえてくる．その原因の一端は，学習しているプログラミング言語や学習用ツールが世の中の職業に使われているのか，あるいは社会生活に役立つものなのかという価値基準で，児童・生徒が作ったプログラムを見ていることに原因があると思われる．つまり，普通教育ではなく，専門教育としての規準で評価しているのである．これは，普通教育としての授業が他教科では成立しているが，導入されたばかりのプログラミングの授業では成立していないことを意味しているのではないだろうか．

加えて，普通教育としてプログラミングを行う場合，学習対象（ここでは，教育用言語や学習用ツールを使用して行うプログラミングそのものを指す）

に学習者が価値を見いだすことが重要である．それは単に「プログラミングって面白い」という感想を学習者が述べるものでなく，「もっと勉強してみたい」，「もっと作ってみたい」といった，学習対象に向き合っていこうとする能動的な態度や，学習対象に向き合うことで生じてくる問題を自ら考えたり，解決したりする能力に裏付けられた意欲や態度に表れるものである．そのような意欲や態度が，専門的に学んでみたいと思ったり，プログラミングにかかわる仕事に敬意を持ったりすることに繋がる．

その意味で，私が見てきたプログラミングの授業において，先に示したような教員の感想が出てきたもう１つの背景には，サンプルプログラムからの学びから，自ら課題や問題に気づき，能動的にプログラミングに向き合うといった児童・生徒の学習場面がほとんど見られなかったことがあるのではないかと考えている．

〈普通教育としてのプログラミング教育〉の価値と可能性

児童・生徒は，好奇心が旺盛であるため，プログラミングの授業で覚えたコマンドやプログラムを試してみたいと感じる．そのため，プログラミングの授業では，サンプルプログラムを動かしているだけでも，喜んで授業に取り組む．これは，小学生の場合特に顕著である．しかし，サンプルプログラムを学習するのみで授業を終えるとなると，それは専門教育でいうところのトレーニング的な指導とさほど変わらない．このような学習を繰り返していたら，児童・生徒はプログラミングの授業に価値を見いださないであろう．

プログラミングの授業がトレーニング的な指導の形態にならないためには，授業で覚えたコマンドやプログラムを使って，児童・生徒が作りたいと思うものをプログラムで実現する授業が必要である．なぜなら，プログラミングは，作成者の思いや構想を実現する手段であり，何かの役に立つことを学ぶというような目的ではないからである．また，自らが作りたいと考えるものを，プログラミングで実現したときこそ，プログラミングの面白さや，学ぶ意義や価値を実感できるからである．加えて，その作ったものが他人に評価されたり，喜ばれたりしたならば，その思いはさらに強くなるであろう．このようなことは，クライアントの要求に悩まされたことのあるシステムエンジニア，または実験や研究のためにプログラムを作成した経験のある人なら誰でも分かることだろう．したがって，サンプルを通じて習ったコマンドやプログラムを学習した後，自分が作りたいものを考える機会を与えることが，プログラミングの授業では重要になる．

とはいえ，自分が作りたいものを考えるという課題を児童・生徒に与えることは簡単なことではない．ところが，ある中学校での実践事例の中に，児童・生徒が自ら作りたいものを考え出すヒントがあった．その実践では以下の流れで授業を展開していた．

(1) 技術科の授業で制御基板を使った回路とプログラムを作成した．
(2) 多くの生徒は，サンプルを基に LED の点滅や音が鳴る回路を作成した．
(3) 教員は，生徒に作品を家に持って帰り，家の人に見せ，家の人に感想を用紙に書かせる宿題を出した．
(4) 生徒の多くは，家の人に作品を見せ，感想を求めた．家の人からは「よくがんばった」と感心されたが，同時に作品に対する修正要求も出された．
(5) 生徒たちは，教員にプログラムを修正したいと願い出た．
(6) 教員は，授業で生徒に修正案を書かせ，プログラムを修正させ，その成果を発表させた．

授業は，大いに盛り上がった．授業が盛り上がった理由は，改善そのものを生徒が習った知識と技能を使って実現し，親の要求に応えたオリジナルな作品を作ることができたからである．そのため，プログラミングの授業に達成感が持てたと考えられる．このように，作りたいものや実現したいものを自覚できたと同時に，プログラミングで実現できるとい

う見通しを持てたとき，あるいは実際に作成できたとき，生徒はプログラミング学習の面白さに気づき，プログラミング学習の意義や価値を見いだすことをこの授業は示していた．

この授業の中で，生徒に作りたいという思いを抱かせたのは，親から修正要求が出た場面である．それは，PDCA サイクルでいうところの A（アクション：改善）の場面である．生徒にいきなり P（プラン）を求めるのではなく，サンプルプログラムを用いた授業から始め，できた作品の C（チェック：評価）を行い，A（アクション：改善）の場面を設定する．それにより，生徒は改善すべき問題を見いだし，習ったプログラミングの知識や技能を使ってどうやって解決するか考え始める．生徒は，ここで学習対象に能動的に向き合ったことになる．そして，このとき生徒の P（プラン）の場面が始まる．これは，D（Do）から始め，P（プラン）に続く授業形態である．このような流れの授業は，プログラミングを「目的を達成する手段」として捉え，自分の P（プラン）によりプログラムを作成する学習となる．このときの生徒は，まさに問題解決に向けた取り組みを始め，改善できた暁には，プログラミングの面白さや学習する価値を感じ始めることであろう．このような学習では，扱うプログラミング言語は何でもよい．児童・生徒の発達段階に合わせた学びやすさが保証された言語や学習ツールならば問題なく実施できるであろう．

〈普通教育としてのプログラミング教育〉の今後に向けて

私は，プログラミングを「目的を達成させる手段」と捉えさせることが普通教育としてのプログラミングの授業の中では重要ではないかと考えた．ところが，初めてプログラミングを習う児童・生徒に対して，最初から「プログラミングは手段である」と説明しても授業は成立しない．そこで，教育界でよく使われる PDCA のサイクルを応用した D（Do）から始める授業を提案する．初めてプログラミングを学ぶ場合，サンプルプログラム通りに打ち込むことから始める学習は大切である．しかし，サンプルプログラムを教える学習にとどまるならば，多くの児童・生徒はプログラミング学習に価値を見いだすことはないだろう．しかし，サンプルプログラムを基に解決するプログラミングの経験を与えることができるならば，児童・生徒が自ら学習対象に向き合う（P（プラン）を立てる）授業が生まれてくる．サンプルプログラムを使い，まずは教員の指示通りに作業する授業から始めても構わない．しかし，その後に児童・生徒がサンプルを通して学んだプログラミングを手段として，達成したい目的に向って取り組む授業を構成する工夫が大切である．

最近の児童・生徒は自ら問題を発見できないと嘆く教員をよく見かけるが，私が提案する，D（Do）から始めて P（プラン）に至る授業を構成すれば，自ら考え問題を発見し始める児童・生徒は増えるはずである．また，この授業の流れでは，教員の工夫次第でさまざまな応用もできる．さらに，児童・生徒だけでなく，教える教員もプログラミング教育の価値を見いだす取り組みになるのではと期待している．

参考文献

1) 文部科学省：中学校学習指導要領解説　技術・家庭編，教育図書，pp.32-37 (2008).
2) 文部科学省：高等学校学習指導要領解説　情報編，開隆堂 (2010).
3) 森　秀樹，杉澤　学，張　　海，前迫孝憲：Scratch を用いた小学校プログラミング授業の実践，日本教育工学会論文誌 34（4），pp.387-394 (2011).
4) 上松恵理子：小学校にプログラミングがやってきた！，三省堂，pp.91-92 (2016).

（2018 年 11 月 15 日受付）

紅林秀治（正会員） kurebayashi.shuji@shizuoka.ac.jp

静岡県内公立中学校，国立大学附属学校，工業高校教諭を経て，2005 年より静岡大学教育学部助教授．2009 年より教授．専門は技術教育．日本産業技術教育学会理事．博士（学校教育学）．2017 年より静岡大学教育学部附属浜松中学校校長を兼務．