# 「プログラミング学習の指針」の有効範囲

久野　靖 *

2019.9.7

## 1　はじめに

□ 「プログラミングを学ぶ」ことの広まり

- ←「プログラム (アプリ)」の公開が一般的なものとなった?
- ←プログラミングの「習い事」化?
- ←中学 (技術科)・高校 (情報科) で学ぶから? まさかね?
- ←政府の後押し「AI・データサイエンス人材」?

□ 2020- 小学校プログラミング開始 (この年の 6 年生から必ず)。しかし…

- 学習の目標は「それぞれの教科の内容」
- 目的は「プログラミング的思考」

□ それはすべて置いておきまして…

□ 久野の主張: プログラミングを学ぶ際は「離陸」を目標とすべき

- 定義 1: 自分でやりたい動作をプログラムにより作り出せる
- 定義 2: 自力でコードを書いて動かし、結果を見て手直しできる

□ 久野の仕事: 電通大で初年次プログラミング科目 (必修 800 人) の統括

- 「研究などでプログラムが書けるように」→やはり離陸が目標
- 目標達成のための「7 つの指針」→有効だと考えている

□ 今回のお題: 「7 つの指針」を小学生向けに直して適用できるか?

## 2　「離陸」とそのレベル

□ 「離陸」の定義 (再掲) (1 のためには 2 がどのみち必要)

- 定義 1: 自分でやりたい動作をプログラムにより作り出せる
- 定義 2: 自力でコードを書いて動かし、結果を見て手直しできる

□ 見比べて行くうちに、離陸にも複数のレベルがあるかなと思った

□ L0(固定目標・固定経路) — 作りたい「目標」固定、行き方も固定

- 例: 理科教材 (?)、Hour of code(?)。想定正解がある

□ L1(固定目標・自由経路) — 「目標」は所与だが行き方が自由

- 例: アルゴロジック (旗は決まっているが取り方はとても自由)

□ L2(箱庭・砂場) — 「目標」のドメインが固定 (絵、動き、音等)、そのための汎用的な道具・材料

- 例: Viscuit, Dolittle, Scratch, Processing。ただし汎用言語同等なのでその「砂場」から出ることは可能ただしそれなりの手当必要

□ L3(汎用言語) — 目標ドメインを限定しない

- 目標に応じてライブラリ等を使う (ハードル)
- つまずきにくさとかの配慮はあまりない (ハードル)

□ L4(質的に異なる目標の実現) — 新たなドメインのための記法から作り出す (CS のプロになるために必要)

□ 学校での題材として考えた場合…

- L0 が最低ではない — マイナスのレベル (目標がない等)(実は普通)
- L0、L1 は授業者にとって安心 (目標がぶれない)、学習者には?(その目標が自分の目標になるかどうか)
- L2 は目標の達成という点では中くらい。ファシリテーションは必要
- L3 は「何でも目標にできる」が、扱いは大変

□ L3 は中学・高校以降で求められる→L3 に入るための方法論? ←大学でやっている方法を援用できないか (今回のテーマ)

## 3　プログラミング学習の指針

□ アンチテーゼとしての避けたいスタイル (2016 夏プロシン)

- テキストにある多数の知識をまずそのまま覚える

*電気通信大学

- テキストにある練習問題を解けるように練習
- 試験では「短時間に」「覚えたやり方で」解く

□ 避けたいスタイルを誘発するテキスト

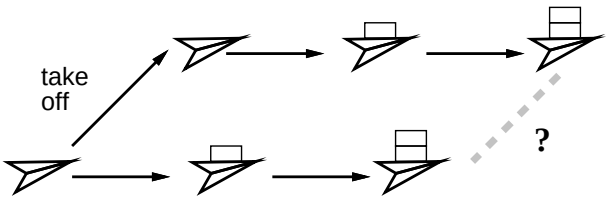
- プログラミング言語の規則を逐一説明
- 演習問題が「規則を覚えているか」にフォーカス
- 少しのプログラム、多数の説明

□ 「知識を覚える」では「離陸」にならない→打破する必要

- 「7 つの指針」(電通大の科目設計で採用)

### 3.1 P1:離陸ファースト

□ 最初に離陸し、それを維持しながら個々の内容を加えて行く



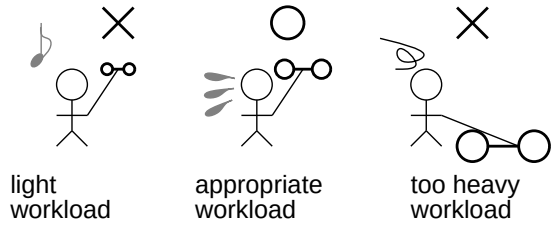

- 多くの授業では最初に沢山知識を詰め込むので離陸できなくなる

□ 離陸ファーストのために…

- 初回から学ぶ内容を「十分に精選し最小限に」「なおかつそれである程度プログラムが書ける範囲」
- 学ぶ内容については納得するまで説明し、自分の「道具」になるまで演習してもらうようにする

### 3.2 P2:多様な水準の演習問題

□ 演習は不可欠だがその際の「負荷」は適切である必要




- 多くの授業では「1 つの問題を皆がやる」→大半には負荷が重すぎ/軽すぎ

□ 適切な負荷を実現するために…

- 演習問題を易から難まで多数用意し、「1 つ以上提出」とする
- 学生は「易しい問題はつまらない」ので適切な問題を選ぶと期待
- 最初から順にやったとしても、適切な負荷の問題で時間を使うことに

### 3.3 P3:演習の重視

□ 離陸ファーストだと最初の数回はすごく易しく見える

```
時点 X までは「易しい」と馬鹿にして演習しない
時点 X で突然「難しすぎ、自分は落ちこぼれた」
```

- 上記は多くの授業で導入時に起こること

□ 演習を必ずやってもらうために…

- 反転授業による授業時間中の演習時間確保
- 毎回プログラムを含むレポートを提出してもらい評価

### 3.4 P4:苦手な人の学びを促す

□ 大学初年次で必修のときよく起こること…

- 経験者 (少数): 労せずしてよい点数を取る
- 初心者 (多数): それを見て無力感、あきらめ、嫌いになる

□ 初心者にやる気を出してもらうために…

- レポートは「プログラムの高度さ」でなく「レポートの質」で評価
- レポートの評価を「ほぼ全員 A」(S を極少数) とする
- 経験者は必死で S を取ろうとするが大体とれない (気の毒だけれど試験のときは普通に絶対評価するのでそれでがんばってもらう)

### 3.5 P5:「プログラムが書ける」目標の明示

□ プログラミング入門科目でよく起こること…

- 沢山の規則や知識→どうしても知識暗記科目化する傾向
- プログラミング習得に失敗→多数落第を避けるため知識的な試験に

□ プログラムが書けることを目標として維持するために…

- レポートも小テストも試験もすべて「プログラムを書く」が 100 %
- 期末試験ではテキストの PDF を参照可能 (暗記は不要)
- 800 人の期末試験の全問プログラミングで手採点は不可能→

### 3.6 P6:書き方の自由度の明示

□ プログラミング入門でよく起こること

- 教科書に載っているプログラムが正解→ひたすら暗記
- 学生は「正解との一致」に非常に強くこだわる

□ プログラムは何通りにも書けて当り前であることを納得させるため…

- テキストの例題で最初から複数の方法を提示

```
def abs1(x)
  if x < 0
    result = -x
  else
    result = x
  end
  return x
end

def abs2(x)
  if x < 0
    return -x
  else
    return x
  end
end

def abs3(x)
  result = x
  if x < 0 then result = -x end
  return x
end
```

- 「複数の可能性から自分で書き方を選ぶ」ように指導
- 短冊テストでも複数の正解が作れるように選択肢を提供

### 3.7 P7:プログラムは自分の頭で作り出す

□ プログラミング入門でよく起こること

- 学生が「正解プログラムの再現」にこだわる
- 題材が学生の興味を惹かない

□ 学生が「自分の問題として」「自分で考える」ために…

- 題材として絵や動画の生成を取り上げる
- 自分が作りたい絵・動画の作成を促す
- (しかしどうしても既存の絵を再現しようとする学生だらけ。ド○えもん、カー○ィ、パッ○マン、… どうしたらいいでしょうね?)

## 4 小学校向けの指針の検討

□ 原田の知見 (開発室ブログ) から重要と思うものを検討

- 19-5-28: 「分岐とか反復を教えるといっているが、変数をきちんと教えなければ意味がない」→ とても賛成。で、きちんと変数を教える方向で。(もう 1 つ「うまく教える人が教えてしまうからだめ」というのもあるが…)
- 18-11-21: 「仕事や研究で使う言語で自由に作れるのが教育、専門家しかできなかったプログラミングを誰でもできるのが大衆化、大衆化をめざしたい」→ もちろん大衆化はいいことだけど、教育はすることになってるので。
- 18-10-06: 「教育は抽象化の習得をめざしているが、やりたいと思ったことをやってみてできるのが理想の言語」→ 久野と原田の完全な対立 (笑)。反論としては「算数も国語も山のように時間を使っているので、そこをすこし削って抽象化を習得する方が幸せ」

□ 上記の議論に基づき、7 つの指針の改訂を検討

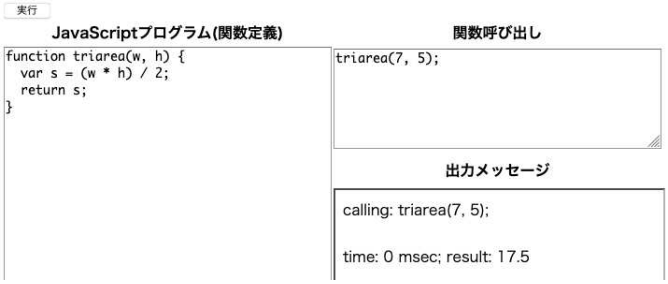
- P1: 初回で「手続き」「変数」を習得して離陸がよい。
- P2: 多数の問題を用意するのは同じでよい。教え合いが望まれる。
- P3: 演習の重視は小学校では自然。機材は 1 人 1 台必要。
- P4: (小学校ではそう格差は無いのでは)
- P5: (小学校では知識偏重はまだ無いのでは)
- P6: 書き方の自由度を強調することは維持。
- P7: 絵や動画を作るところまでの長さが課題→解決していない

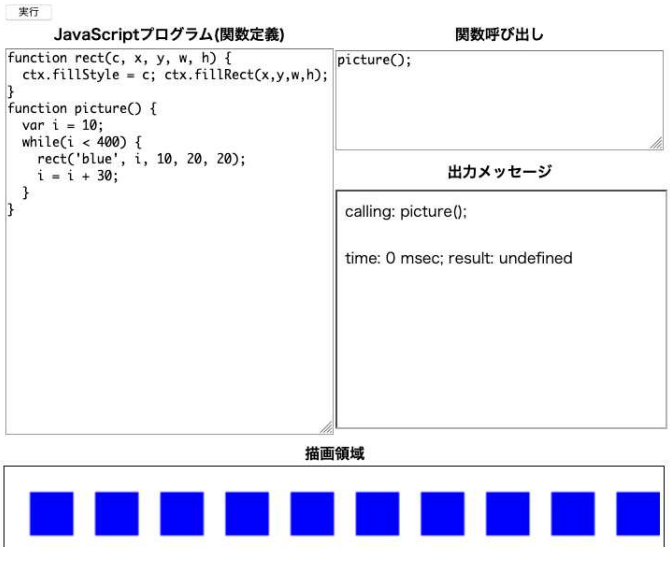
□ L0〜L2 の様々なツールでは「すぐ動かせる」ための手当てが充実。L3(汎用言語) ではそれがないのでどう対応するか

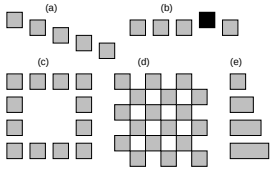
## 5 小学生対象テキストの試作

□ JavaScript を使用し、3 回ぶんのコースを試作 (絵を描くまで)

- ハードルを下げるために「JavaScript 練習ページ」を用意
- P1 のため、内容を精選 (ループは while のみ、絵は fillRect のみ等)。
- P2 のため、演習問題は難しいものまで用意 (x**8、3 数最大、fib、市松模様)。
- P3 のため、説明は短くしてすぐ演習に。
- P6 のため、複数の書き方を多く示す。
- P7 のため、canvas で絵を描くところまで作成。






□ # 1 変数による値の計算

- 例題: 三角形の面積の計算
- 題材: 関数・パラメタ・変数・代入・式
- 演習: 様々な計算を行う関数を書く

□ # 2: 制御構造 (if, while)

- 例題: 2 数の差 (の絶対値)
- 題材: if 文、条件、場合分け、再代入
- 演習: 2(3) 数の最大、正負零の場合分け
- 例題: カウントアップ
- 題材: while 文、反復、変数の加算
- 演習: 様々な規則による計数ループ

□ # 3: canvas グラフィクス

- 例題: 正方形を並べる
- 題材: canvas、矩形の fill、下請け関数
- 演習: 矩形の様々な配置
- 例題: RGB 色指定 (前の例の拡張)
- 題材: 色指定、連続的な色変化
- 演習: 自由制作

## 6 議論とまとめ

□ 筆者が大学初年次向け科目で取り入れて来た指針→小学校でも有効か検討

- 離陸の概念のレベル分け
- 指針 P1〜P7
- 原田の知見の検討
- 小学校高学年向けカリキュラム

□ 全体として…

- 「誰でも」はまだ難しい気がする
- 中学校ではテキスト言語でコーディングする必要→小学校高学年でこのような内容をやるのは必要かも