ARTICLE

応専

プログラミング入門科目の指針と実践例(後編)

久野　靖
電気通信大学

前編のあらまし

前編[1]では大学のプログラミング科目について「単位を取得したのにプログラミングができない」という問題意識と，これを克服する指針「離陸ファースト」「多様な水準の演習問題」「演習の重視」「苦手な人の学びを促す」「プログラムが書けるという目標の明示」「プログラミングに唯一の正解はない」「プログラムは自分の頭で作り出す」を提案し，FP(電気通信大学初年次科目「基礎プログラミングおよび演習」)でそれをどう具体化したかを解説した．本稿ではこれらの指針に基づいたFPの具体的なカリキュラム，実施時の知見，結果について述べる．

使用言語とカリキュラム構成

前編で述べた通り，FPでは最初の回ですぐ離陸でき演習に取りかかれるようにRubyを採用しているが，後半はC言語を使用している．これは，本学の2年次以降の各科目がC言語を採用しているため，C言語も学んでおく必要があったことによる．

入門科目であるのに言語を2つ学ばなければならないことは学習者にとって大きな負担となるが，一方で複数の言語を学ぶことの有用性は一般に言われている．そこでFPでも，何とか工夫して「RubyからC」を実現することにした(詳細は後述)．

表-1にFPのカリキュラムを示す（2017年度と2018年度で同一，ただし細かい改良はある）．＃11～15がC言語の内容となる．理系の大学であることを活かし，なじみのある数学に関係する題材を積極的に取り入れた．構成で工夫したのは，「離陸ファースト」のためにその回の基本的な例題を学んだらすぐに演習に入れるように題材を組み合わせた点と，中間部分については個々の内容が「行き止まり」になっていて，そこが理解できなくてもその先の学習に支障がないようにした点である(詳細は後述)．

もう1つの特徴は，全体として入門科目にしては異例にコンピュータサイエンス（CS）の内容を多く盛り込んだ点である．これは東京大学が初年次プログラミング科目で取り入れた方針であり[2]，筆者も非常勤講師としてその内容を教えた際[3]，「CSのさまざまな題材こそ知的関心につながり学ぶ価値のある内容である」と思うようになった．

学内でシラバスの承認をもらうときに「この内容は2年次以降で扱うのでは」という意見もあったが，「全員が必須内容として身につけるのは2年次以降としても，興味の持てる題材として触れてもらうのは構わないはずで，複数回触れてもらう方が後の科目にとっても好ましい」と述べて了承を得た．

＃1 プログラミング入門；様々な誤差
＃2 分岐と反復；数値積分
＃3 制御構造(2)；配列とその利用
＃4 手続きと抽象化；再帰呼び出し
＃5 2次元配列；レコードと画像
＃6 画像の生成(総合実習)
＃7 整列アルゴリズム；計算量
＃8 計算量(2)；乱数とランダム性
＃9 オブジェクト指向
＃10 動的データ構造；情報隠蔽
＃11 C言語入門；$f(x)=0$の求解
＃12 様々な型と配列；動的計画法
＃13 文字列の操作；パターン探索
＃14 構造体；表と探索
＃15 チームによる開発（総合実習）

表-1　各回の内容

＃1～4：入門部分

入門部分の目標は，まずは普通にプログラムが書けるだけの材料を身につけることとした．中でも＃1の役割は重要である．この回は「三角形の面積」から始めて，直線的なコードだけで書けるさまざまなプログラムを課題として演習してもらう．初心者の学生でも4行のプログラムを打って動かすことはすぐにでき，また理解もできるので，それなりに熱心に演習に取り組んでいた．

後半の題材として実数計算の誤差を取り上げ，さまざまな誤差の出方を観察する課題を含めている．普通であれば初心者が初回から学ぶことはない内容であるが，一直線のコードで扱え，また「なぜそうなるのか」を考える題材になることから，このようにした．プログラミング経験者である学生にとっても，この内容は初めてのようで，興味を持って取り組んだ様子である．この内容を十分理解するまでに至らなくても，「コンピュータによる計算は100%正確などではない」という基本的な事項を記憶にとどめるだけで価値はあると考える．

＃2と＃3ではif文と繰り返し（whileループ，計数ループ）を一通り学び，これらを組み合わせてコードを構成する課題（fizzbuzz[☆1]など）を演習してもらう．制御構造を工夫するとそれに応じて動作が作り出せることは多くの学生の関心を引いていた．配列についてはカリキュラムの都合上ここで一度取り上げているが，あまり重きは置いていない．

＃4は手続きが題材であるが，初回から「プログラム」と称して手続き（Rubyでは「メソッド」）を作成している．ここではそれを見直し，手続きにまたがったデータ（広域変数）や再帰の話題を取り上げている．RPN（逆ポーランド記法）電卓の例題（図-1）などは呼び出し時のかっこが省略できるというRubyの特性上それらしく見えるため，多くの学生の興味を引いていた．再帰は「数式の定義をそのまま再帰プログラムに対応づける」形で導入することで多くの学生が自然に受け止めていた．

```
$s = []
def e(x) $s.push(x); return $s end
def add
  y = $s.pop; x = $s.pop; $s.push(x+y)
  return $s
end
```

```
irb> e 1
=> [1]
irb> e 3
=> [1, 3]
irb> add
=> [4]
```

図-1　Rubyによる RPN 電卓と実行例

```
Pixel = Struct.new(:r, :g, :b)
$img = Array.new(200) do Array.new(300) do
         Struct.new(255, 255, 255) end end
```

```
def writeimage(filename)
  open(filename, 'wb') do |f|
    f.puts("P6\n300 200\n255")
    $img.each do |a| a.each do |p|
      f.write(p.to_a.pack("ccc")) end end
  end
end
```

図-2　Rubyによる画像の表現と出力

＃5～6：2次元配列と画像の生成

前編でも述べたように，画像の生成は「自分の頭で作り出す」体験を持てる題材としてできるだけ早い回に設定している．＃5で画像を「RGB値を表すレコード型の2次元配列」として導入し，200×300ピクセル24ビットカラーのPPM（Portable PixMap，Unixのツールで多く使われる簡潔なフォーマット）形式の画像を出力できるようにする（図-2）．すべて自前で書くことで，画像の原理や扱いを「謎」なしに理解してもらうことを目指した（実際にはwriteimageの動作は謎のままという学生も多かった）．

その上で基本的な線分，円，長方形，三角形などの塗りつぶしを例示し，＃6でそれらを利用して（または類似のものを自分で独自に作って）「美しい画像」を生成する総合課題を課している．

☆1　1, 2, 3……と数値を表示する．ただし，3の倍数ならfizz，5の倍数ならbuzz，両方ならfizzbuzzと（数値の代わりに）表示する．

レポートとともに提出された作品の一部を図 -3 に示す．技術的な難しさのレベルは学生によりさまざまだが，どの学生もそれなりに工夫して「自分が作ろうと思う絵」を作っていると感じる．

6 ～ 10：中間部分

中間部分は「整列（と時間計算量）」「疑似乱数」「クラス定義」「単連結リスト」とそれぞれ異なる題材を取り上げている．どれもやや高度だがその回で完結し，次回に影響しないため，できなかった場合でもあまりダメージはない．

整列は問題設定が「小さい順に並べる」と明快で，多様なアルゴリズムがあること，またそれらで大きく速度が違ってくることを体験する題材として適している．例題としてバブルソートを提示した後，選択ソート，挿入ソートを誘導つきで書くことを課題としたが，配列に十分慣れていない学生が多く苦戦していた．選択ソート用の「指定範囲での最小値の位置を返す」メソッドの作成が一番やさしい問題であり，これは配列の回に「最大値の位置を返す」演習が既習のため，苦手な学生でも大半がなんとかできていた（ヒントは与えている）．

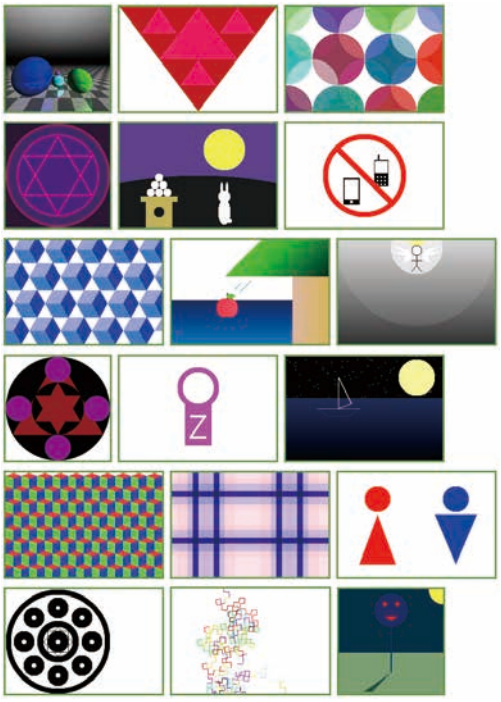


図 -3　提出された画像の例

疑似乱数は「サイコロやコインの利用をシミュレーションする」題材が直観的に分かりやすかったようで，多くの学生が楽しんで実施していた．

クラス定義はクラス中にメソッドを置くという構造を知ってもらうことに力点を置き，「値を put すると覚えていて get で取り出せる」などごく簡単な課題を含めた結果，学生からも「構造は目新しいがプログラムを作ることができた」と肯定的なコメントを多くもらうことができた．

単連結リストは Ruby の最終回であり，やや高度でもチャレンジと思って取り入れた．2017 年度はテキストがやや難しい内容から始まり，例題も再帰が分からず苦戦した学生が多かった．

そこで 2018 年度は，まず irb でリストのセルの連鎖を作り，途中に nil を入れて短くしたり 2 つのリストをつなげるなどの体験をしてもらった（図 -4）．その後「数値のリストで合計」「文字列のリストで全連結」など単純な課題を含む形で演習したが，学生の得手不得手に対応するため例題コードでループ版と再帰版を併記した上，両方でリストをたどって処理するコードのテンプレートもヒントとして提示した．この結果，多くの学生はこれら基本的な課題がこなせ，肯定的なコメントをもらうことができた．2 つの版の併記については「ループ版なら分かった」等のコメントもあったが，「再帰を理解する題材となり良かった」というものも多数あった．

11 ～ 15：C 言語

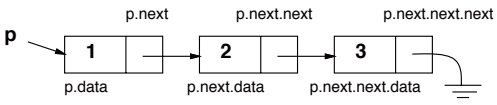
Ruby で入門後，途中で C 言語に移行することが，本カリキュラムの最大のチャレンジの 1 つである．特に，Ruby では irb のおかげで入出力を書かなくても済んだが，C ではそうはいかない．

2017 年度は main でコマンド引数を受け取り atoi や atof で数値に変換する形で例題を作ったが，これまでと大幅に構造が違うため混乱があった．また，コマンド引数の個数検査や，呼び出される関

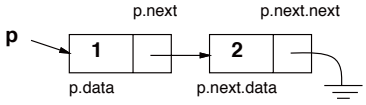
数を下に置くためのプロトタイプ宣言の記述を含めたことも混乱を増大させた．＃ 11 と＃ 12 の間に冬休みが 3 週間入るため，＃ 12 のときに＃ 11 の内容を覚えていないことも問題だった．

2018 年度はこの反省に立ち，図 -5 を提示して「いままでは本体の関数だけ書いたが，C 言語では main で入力と出力をする，ただし main の内容はどのプログラムでも類似」と説明し，scanf と printf で入出力する形とした．当面は main を最後に置かせることで，プロトタイプ宣言を＃ 14 の分割コンパイルの説明まで保留した．

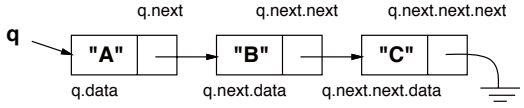
```
irb> Cell = Struct.new(:data, :next)
irb> p = Cell.new(1, Cell.new(2,
           Cell.new(3, nil)))
irb> p
```



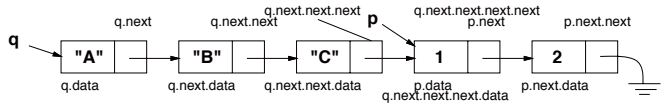

```
irb> p.next.next = nil
irb> p
```




```
irb> q = Cell.new("A", Cell.new("B",
           Cell.new("C", nil)))
irb> q
```




```
irb> q.next.next = p
irb> q
```



Quiz: 次のようにできますか？

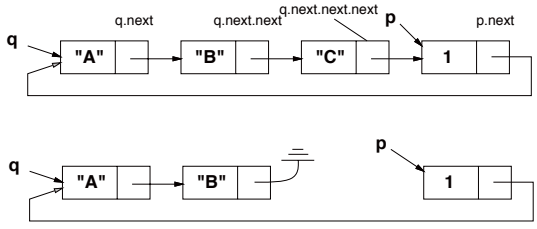


図 -4　irb を用いた単連結リストの練習

もう 1 つの課題として，C 言語はメジャーなので既習者も混在している点がある．そこで総合課題を除く＃ 11 ～ 14 では内容を「基礎的な前半」「高度な後半」に明確に分け，後半は試験範囲外で経験者のみでよいと明示することで，初級者が挫折を感じないようにしつつ，経験者にもチャレンジしがいのある演習を提供した（予習は全範囲で義務づけているので，初級者でも関心を持ち後半の演習に取り組んでもらえることは多くあった）．

＃ 11 は C 言語の初回なので図 -5 を提示した後「三角形の面積」を最初の例題とし，Ruby で学んだ題材を C でも書いてみるという演習を中心とした．ループや枝分かれ等の制御構造は C と Ruby の対比を示し，機能は同じで書き方だけ変えれば済むという形でまとめて説明した．そして「冬休みに忘れる」問題に対処するため，Ruby ですでに取り上げたやさしい課題を多く用意し，通常は「1 個以上提出」であるレポートをこの回のみ「10 個以上提出」とし，冬休み中に練習するよう呼びかけた．

＃ 12 は配列が中心だが，C 言語では配列アクセスがポインタ演算に基づくため，ポインタも併せて扱う．この内容は一般に「難しい，分かりにくい」とされており，学生にもそのような意見が多かった．配列の扱い自体は Ruby と同様でできるが，Ruby 部分での配列の練習が不足で身についていない学生が多いという問題があった．

＃ 13 は文字列の内容で，文字列がナル文字で終端された文字の配列であることを説明した後，「文字列の長さを求める」「文字列中の文字 c_1 を c_2 に置き換える」など基本的な演習を中心に行った．文字列はイメージしやすく分かるとの感想もあったが，課題は（ヒントを活用して）できても，配列が不得手で自信がないという感想も多かった．

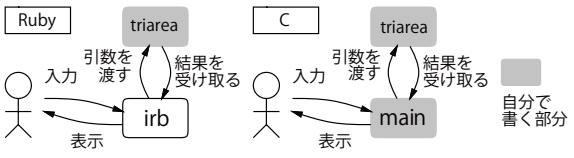


図 -5　Ruby と C の対比の図

＃14 は構造体で，Ruby の画像でも使用した RGB 値を表す構造体を定義し，複数の色を混ぜる関数を例題として示した後，1 つの色を受け取り明るく／暗くした色を返す，補色（実際には RGB それぞれを 255 から引く）を返すなどの演習問題を課した．16 進 6 桁の RGB 値を入力するとその色が見られる実習ページを用意して実際の色を確認することで，演習の内容が分かりやすいようにした．学生の感想は，よく分かったという意見と（おそらく C 言語の構造体の書き方になじんでいないため）難しいという意見とに二分されていた．

＃15 は 2 回目の総合課題で，2 ～ 3 名のグループで分担してアニメーションを製作する内容である．例題として，図 -2 と同様の機能を C 言語の API として実装したものを提供し，ただしファイルは書き出すごとに連番のファイル名で作成されるようにし，多数のファイルを生成した後，ツールで動画表示させた．説明時はアニメーションという高度そうな内容に不安を示す学生もいたが，授業時に例題をコピーして動かすことで個々の絵は＃6 と変わらないことに納得し，グループで作品内容を話し合って取り組む様子が見られ，多くの力作が提出されている．

表 -2　学習内容は今後自分の役に立つと思うか？

とても そう思う	そう思う	どちらで もない	思わない	まったく 思わない	未回答
186 26.2%	331 46.7%	145 20.5%	20 2.8%	12 1.7%	15 2.1%

表 -3　どれくらい多く新たなことを学んだか？

多数学んだ	まあ学んだ	どちらで もない	少しだけ	まったく ない	未回答
279 39.4%	283 39.9%	92 13.0%	26 3.7%	15 2.1%	14 2.0%

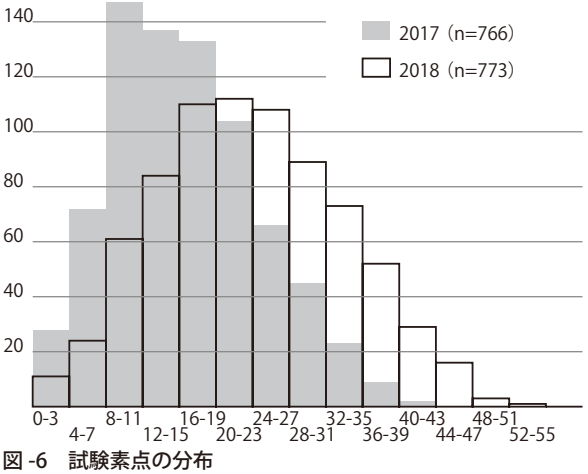


図 -6　試験素点の分布

結果とまとめ

表 -2，表 -3 に，2018 年度授業最終回のアンケート（n=709）結果抜粋を示す．多くの学生がこの科目を有用と考え，（プログラミングの科目なのでプログラミングについて）多く学んだと回答している．

期末試験はすべて CBT（Computer Based Test）による短冊問題で，80 分の試験で 28 問を課した．採点は全問，正解（複数の場合もある）と一致で 2 点，1 カ所の欠落，余分，入れ替わりで部分点 1 点とした（素点満点 56 点）．素点合計を算出後，ある程度できた学生が 50 点になるように換算を行い，レポート点と合算している．図 -6 に 2017・2018 年度の試験素点合計の分布を示す．前述した授業内容の手直しと，2017 年度にやや難しい問題があったものをやさしい問題に変更したことから，全体に成績は向上している．たとえば，95% の学生が 8 点（4 問相当）以上，77% の学生が 16 点（8 問相当）以上の得点を得ている．短冊型の試験はプログラムが書けなければ得点は難しいため，本科目は多くの学生について「基礎的なプログラムが書ける」という目標を達成したものと考えている．

一方で，2018 年度の内容についても「配列の練習が不足」など，改良が必要な点は複数見つかっているため，科目内容は今後とも手直ししていきたい．

参考文献

1）久野　靖：プログラミング入門科目の指針と実践例（前編），情報処理，Vol.60, No.3, pp.244-247 (2019).
2）森畑明昌：東京大学における全学プログラミング教育，情報処理，Vol.57, No.4, pp.362-365 (Apr. 2016).
3）久野　靖：Ruby による情報科学入門，近代科学社 (June 2018).

（2019 年 2 月 5 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．