プログラミング教育における小テストの実践報告

田中善雄[†] 三宅芳雄[††]

筆者らはプログラミング初等教育のクラスの中で，短い設問からなる小テストを行うことで，学生の学習状態を把握する試みを行い，教育の改善に役立てているので報告する．小テスト授業中にクラス全体で一斉に実施するペーパーテストだが，２，３問の短い設問から構成されているため，短時間で実施できる．小テストは的を絞って作ることによって，簡単に実施できてしかも有効なものになる．実際，多くの場面で，教授者が期待するほどには十分な学習が成立していないことが明らかになった．学習過程の実態の把握に基づいた柔軟で臨機王変な対応を取ることも可能になり教育の改善に直接，役立てることもできた．

Report on the shout tests for programming education

YOSHIO TANAKA[†] YOSHIO MIYAKE[††]

We regularly conduct short tests in our elementary class on programming. Although these short tests consist of a few questions and were short enough to carry out quickly, focusing some specific points makes them informative enough to show various aspects of complex learning processes. In this paper, we report several cases and show what we actually found on the learning processes on programming and how we used the results to improve our instruction.

1. はじめに *

より効果的なプログラミング教育の実現には，学習の実態の十分な把握が重要であり，筆者らは，そのためにさまざまな試みを行ってきた．その一つとして，授業中に短い設問からなる小テストを随時行うというものがあり，それらが，実際にどのような効果があったかをいくつかの具体的なケースに基づいて，報告する．

プログラミング教育の難しさは，学習過程の複雑な実態を十分に把握できないところにあると言ってもよいだろう．そのため，授業の進度が速くなりすぎ，学習者の側の十分な理解を伴わないままに授業が進行してしまうことがある．少し複雑な内容を教授する場合には，学習者が，一端，理解したように見えても，その理解は後の授業を進めるためには十分に安定したものではないことが少なくない．そのような時に，新しい内容を学習しようとしても，学習者はその内容を理解することができず，表面的な理解に留まったままに終わってしまうことが起きる．

筆者らは，ここで取り上げている小テストの他にも，いろいろな手段で学習者の状態を知る努力をしてきた．筆者らのプログラミング教育のクラスは，演習を中心にして運営されており，教育者や教育補助者と学習者とのやりとりの中で，学習者の学習状態に対して，一定の情報が得られる．また，学習者の提出する演習課題の解答内容も当然基本的な情報の一つである．さらに，毎回演習の最後に，学習者のコメントや質問を書く課題が設けられている．これらは，学習者の学習状態について，多くの情報を提供するが，教授者が知りたい学習者の学習状態についての情報が直接得られるとは限らない．これに対して，小テストは設問数も限られ時間も十分に取れないという限界はあるものの，，特定のポイントに的を絞ることにより，必要な情報を直接得られるという長所がある．また，その結果をその後の授業の展開に直接役立てることができる場合もある．例えば，プログラミング 学習を演習課題の解答内容だけから見ていると，一見課題ができおり，理解が成立しているように見えるが，基本的な理解や，習

† 中京大学大学院 情報科学研究科
Graduate school of computer and cognitive science, Chukyo University

†† 中京大学 情報理工学部
School of Information Science and Technology, Chukyo University

熟度が不十分だったり，理解がどれだけ安定しているかを見極めるのは難しいことが少なくない．小テストはそのような学習者の複雑な学習状態を把握し，早めに授業の方向を修正するのに役立つ場合がある．以下，具体的なテストの作成や実施方法について，具体例を挙げながら報告する．

2. 小テストの作成と実施の方法

小テストはプログラミングを初めて学ぶ初心者を対象にした，Rubyを教えるプログラミング入門コースにおいて実施している．この入門コースのクラスは2時限続きになっており，教授者による内容の解説と学習者が行う演習(Web上の解説に基づいて，提示された演習課題に沿ってプログラムを作成する)から構成されている．教授者による解説は新しい内容に入る時に，手短に行われるだけで，主な学習は学習者が教材を読みながら，自分のペースで演習課題を行っていくことで進行する．

小テストは授業の冒頭と，授業の切れ目のよい時に，1回または2回行われる．小テストは学習者の基本的な内容理解や習熟の程度を少数の設問で効率よく調べるために，的を絞って行う．従って，学習者の学習状態の不十分さに対するなんらかの仮説を基にして，設問が構成される．2問から4問の設問がA4版の用紙に解答欄も含まれた形で用意され，クラス全体で実施される．実施の時間は，全体で5分から10分程度になる．

3. 小テストで明らかになった事

小テストを取り入れたことで，学習状態の理解が深まり，後の授業展開にも役立ったいくつかの事例を以下に述べる．

事例1

以下の例は，提出された演習課題の解答内容からはすぐには分からない学習者の習熟度について，小テストを実施することで，より詳しく知ることができた事例である．

学習者Aの学習状態は,演習課題を真面目に行っていくタイプだが，分からないところなどは，友達や教授者の助力を得ながら解決していくので，提出された演習課題の内容は正解が多い．例えば，配列について初めて学習した時の配列の要素を合計するという演習課題の解答内容については以下のように正解している．

課題．次のプログラムは配列の要素の数を合計して結果を出力するものにしようとして、作ったものですが、完全なものではありません。実際に実行しながら誤りを見つけて正しいプログラムを完成させなさい。まず、このまま実行してみて何が表示されるのか調べて、それから考えてみてください。
ヒント：while の使い方の一つの「数の足しこみのパターン」が参考になる。i＝i＋1も数の足しこみのパターンの一つ。

```
data = [2, 4, 7, 3, 9, 6, 4, 4, 10, 23, 1, 1]
i = 0
sum = 0
while(i < data.size)
  sum = data[i]
  p sum
  i = i + 1
end
print "配列の合計は", sum, "です。¥n"
```

sum＝data[i]の部分をsum＝sum＋data[i]と書き変えてp sumを消すというのが正解である．この課題のポイントはsum＝sum＋data[i]の「数の足しこみのパターン」が，うまく書けるか？という部分で，学習者Aは素直に正答していた．

しかし，次の週のクラスの冒頭で行った小テストでは，一週間前に演習課題で正解している内容に正しく答えられていない．

＜演習課題から一週間後の学習者Aの解答＞

問1．解答欄のプログラムは配列aryの要素の数を合計するプログラムです。実行結果以下の通りになるようにプログラムを完成させなさい。

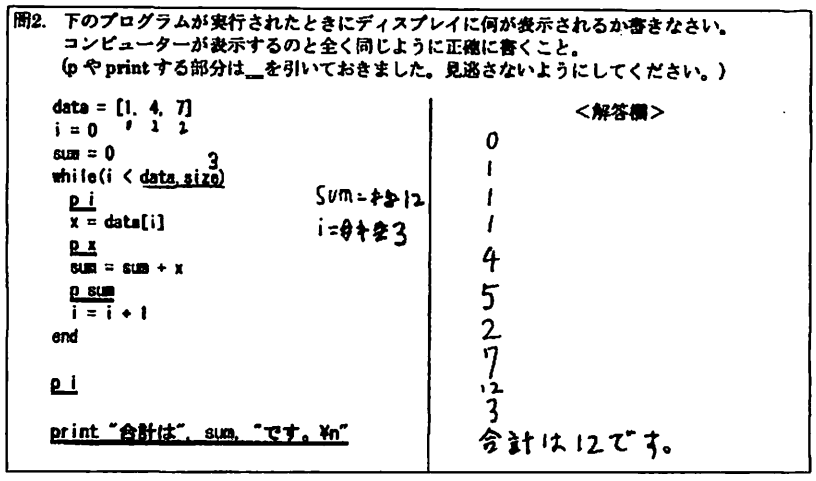
＜実行結果＞
配列の合計は15です。

＜解答欄＞

```
ary = [1, 2, 3, 4, 5]

i = 0
sum = 0
while(i < ary.size )
  sum = sum + 1                ←ココ
  i = i + 1
end
print "配列の合計は", sum, "です。¥n"
```

要素の足しこみのパターンのため，ary[i]を足す式にしなければならないが，数の1を足すという誤解答をし



図：問1の後で行ったテストの学習者Aの解答

ている．学習者Aと同じように，演習課題ではできているのに小テストでは解答できない学習者がこの他にも何人かいた．演習課題を正しく答えていても，次の週までその理解を保持できない浅い学習しか成立していなかった可能性がある．

一方で，同じ学習者Aの同時期に行われた同様の配列についての小テストの解答を上の図に示す．この問題は配列の要素を足しこむプログラムだが，この問題については，学習者Aは正答している．問題のプログラムの部分にメモの書き込みを見ても，変数の値をしっかり把握して解いており，プログラムの進行を正しく追いかけることができている．

これらの結果から，学習者Aは配列に関して，プログラムの進行を追うことができるレベルの理解には達していても，自分から配列に関する「足しこみのパターン」を生成できるだけの習熟には達していないことが分かる．

事例2

以下では，小テストを取り入れた事より，より基本的な事項に関して，学習者の理解の状態を把握することができた事例である．

プログラムが一歩一歩，上から実行されるという制御の流れの原則を理解し，また，変数への代入や取り出しを理解するのはそれほど難しくないように見える．そのことを確かめるために，変数について初めて学んだクラスで，演習課題を行った後の，切れ目の良いところで，学習者の理解を確かめるために次のような小テストを行った．

問3．下のプログラムが実行されたときにディスプレイに何が表示されるか書きなさい．
a = 10 b = 20 a = b b = a print a, "¥n" print b, "¥n"

大方の学習者が正当することを予想し，確認のつもりで出題したものだが，結果は，予想に反して，クラスの半数しか正しく答えられなかった．プログラミングの逐次実行や変数へ代入，変数からの値の取り出しなどの基本については，解説や演習課題を一度行うだけでは十分な理解が得られないことがわかった．この小テストによって学習者の理解の程度が明らかになったので，その後すぐに，小テストの正解を示し，プログラムの基本的な流れや変数への代入について再度説明を行った．その結果，一週間後の1限目の頭に行った小テストでは，同様の問題に対して，ほとんどの学習者が正解していた．

事例3

制御の流れを追えることはプログラム理解の基本だが，簡単なプログラムで制御の流れを追うことができても，習熟が十分でない場合，プログラムが少し複雑になると，安定してプログラムの制御を追えなくなる場合が少なくない．このことを確認するために，whileの繰り返しについて，小テストを行った．

問4．下のプログラムが実行されたときにディスプレイに何が表示されるか書きなさい。

```
i = 0
while(i < 2)
  j = 0
  while(j < 3)
    print "i→", i, "  "
    print "j→", j, "¥n"
    j = j + 1
  end
  i = i + 1
end
```

この問題は制御の流れを機械的に追うことができれば良いだけなので，一重のwhileループについての基本的な理解があれば二重のwhileループでも，同じように制御を追えそうだが，必ずしもそうはならない．実際，小テストの結果は，クラスの4分の3が二重ループになると制御が追えなくなり正答することができなかった．

制御の流れを追うというようなプログラムの基本についても，どんな場合にでも安定してできるためには，十分な習熟が必要になり，十分な学習時間が必要になる．例えば，whileの場合endまでいったらwhileの頭まで戻るということを理解していても，二重ループのようなプログラムが複雑な構造を持つ場合では，プログラムの制御を追って内側のwhileのendに辿りついた時点で，外側のwhileの先頭に戻ってしまうなどということが起こる．

このテストを行った後，数回のクラスの後で2重ループについての小テストを行ったところ，正解者の数は着実に増えていることがわかった．

事例4

授業が進むに従って，プログラミングの新しい内容，例えば，「条件分岐」「繰り返し」「配列」などを順に学んでいく．これらを個々に理解していることと，それを自在に組み合わせて現実のプログラムを生成することには大きな隔たりがある．

そのことを確認するために，プログラムの実行過程を追うのではなく，実際にプログラムを作成する問題からなる小テストを作成した．

問5．解答欄のプログラムは、a,b,cが定義されている計算プログラムです。実行結果が以下の通りになるようにプログラムを完成させなさい。ただし、途中の計算式にa,b,cを使うこと。

<実行結果>

```
x の値は 10 です。
y の値は 53 です。
z の値は 8 です。
```

<解答欄>

```
a = 3
b = 7
c = 56

x = a + b
y =                          ←ココ
z =                          ←ココ

print "x の値は", x, "です。¥n"
print "y の値は", y, "です。¥n"
print "z の値は", z, "です。¥n"
```

このテストでは，プログラムを生成することの難しさを考慮して，変数やprint文の形を指定し，解答する場所の指定をした．(「←ココ」と書いて指定)　しかし，それでも，クラスの3分の1は正答することができなかった．これまでのクラスで+,-,*,/の4種類の演算の表現自体については学んでいる。上記の結果から，簡単な組み合わせであっても、学習者にとって新しい形式でプログラムを作り出すという問題は，学習初期の学習者にとっては意外と難しい問題になることがわかった．

4. まとめ

小テストを効果的に実施することにより，学習者の学習状態の複雑な実態を知ることができた．その結果，必要な説明を詳しく行うことなどの，小テストの結果を直接，授業の改善に役立たせることができる場合もあった．小テストは小回りが利いて実施しやすいため，的を絞り，的確な設問を用意することで，学習状態を的確に知るための有効な手段になる場合が少なくない．小テストの性格上，多くの設問を用意することはできないが，直接知りたいことを設問にすることができる点は有利である．

学習状態を的確に把握するという困難な課題を達成するには，学習者の状態を捉える有効な理論の発展も必須になる．その理論の発展のための基本的な事実の収集のためにも，小テストは有効に活用できる可能性がある．学習者の習熟度に応じて，有効な設問を用意する小テストの自動化も今後の重要な課題である．