解説

基般

なぜプログラミングは難しいのか?

繰り返しの理解構造とCの教科書分析からのアプローチ

保福やよい
神奈川県立相模向陽館高等学校

スモールステップと教育

高校で数学と情報を教えている．高校の情報Bという教科では，プログラミングを扱うことがある．高校では，プログラミングの楽しさや，うまくいったときの達成感を大切にしたいという考えから，教育用の言語が用いられることが多い．高校生などの初学者がプログラミングを学習するときに躓きやすい個所の1つに繰り返しがある．

繰り返しは分かってみれば難しい概念ではないが，初学者はなぜ難しさを感じるのだろう．そこで，初学者がプログラミングをどのように理解していくかという過程に興味を持ち，大学の先生方に相談しながら研究してみることにした．

数学を教えてきた経験から，初学者が新しい概念を学ぶときには，新しいことを少しずつ取り入れていくスモールステップの考え方が有効であることが知られている．そこでこの考え方をプログラミング教育に活かせないかと考えた．そして，プログラミング教育におけるステップを考える上で，構文のステップに着目し，ツールを用いて教科書等の分析を行ってみたことを報告したい．

教育用言語での繰り返し

繰り返しの概念は，教育用言語を用いたときは難易度が低いように感じている．ここでは教育用言語Scratchとドリトル[1)]の繰り返しを紹介する．

❑ Scratch

Scratchは MITメディアラボで開発された子ども向けのプログラミング言語である．

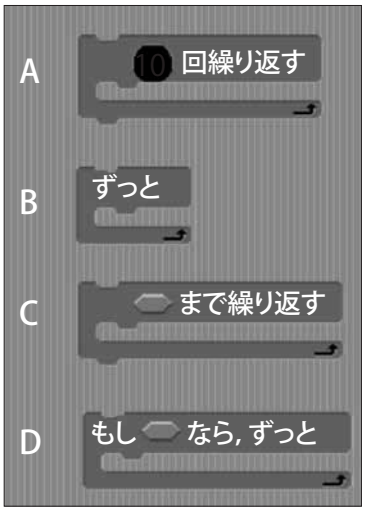

図-1　Scratchの繰り返し

Scratchには繰り返しを表現する4種類のブロックが用意されている．

高校の授業でScratchを扱ったところ，生徒は自由課題で繰り返しを表現するために，**図-1**の4種類のブロックのうちAとBしか使わなかった．生徒にとって固定回数を繰り返すAや無限に繰り返すBは理解しやすい繰り返しと考えることができる．それに比べて，変数の概念を伴うCとDは，より高度な繰り返しと考えることができる．

❑ ドリトル

ドリトルは初学者でも学びやすいように設計された教育用言語であり，大阪電気通信大学の兼宗進先生が公開している．プログラムは一般の言語と同様にテキストで記述する．ドリトルには繰り返しを実行する構文が3種類用意されている．

```
カメ太＝タートル！作る.
「カメ太！10歩　歩く」！5回　繰り返す.
```

図 -2　ドリトルの固定回数の繰り返し

```
カメ太＝タートル！作る.
時計＝タイマー！作る.
時計！「カメ太！3歩　歩く」実行.
```

図 -3　ドリトルの固定回数の間欠繰り返し

```
ｘ＝1.
「ｘ <=10」！の間「カメ太！10歩　歩く．x=x+1」実行.
```

図 -4　ドリトルの条件判定繰り返し

高校の授業でドリトルを扱ったところ，生徒は**図 -2**，**図 -3** の固定回数の繰り返しを容易に理解し，アニメーションなどの自由作品で多く使ったが，**図 -4** の条件判定を伴う繰り返しは変数を使うために難易度が高く，生徒の自由作品で使われることは少なかった．

これらのことより，変数を伴わない繰り返しは理解が容易だが，変数を用いる繰り返しは理解が難しいことが分かった．

C 言語での繰り返し

❑ C 言語の繰り返し構文

C 言語には繰り返しの構文として while 文, for 文, do 〜 while 文の 3 種類が用意されている．C 言語の繰り返しはいずれも制御変数を伴い，変数の理解なくしては繰り返しを理解することは困難である．

for 文は，図 -1 の A や図 -2 と同様に固定回数だけを繰り返すことが多いため簡単に見える．固定回数の繰り返しを強調するために，構文を丸暗記して使うように指導されることも多いようだ．一方，while 文や do 〜 while 文などは繰り返しの手順をトレースしやすいが，やや難しく感じられるかもしれない．

C 言語の繰り返しを教える順番の検討

次に，C 言語の 3 種類の繰り返しがどのような順番で教えられているかに興味を持ち，市販の入門テキストを調べてみた．

テキスト	A[2]	B[3]	C[4]	D[5]	E[6]
出版国	日本	日本	日本	米国	米国
if	1	1	1	4	1
while	3	3	3	2	2
do 〜 while	4	2	4	3	4
for	2	4	2	1	3

表 -1　市販テキストの順番

❑ 市販テキストの分析

日本や海外の市販テキストを分析し，C 言語の繰り返しがどのような順番で教えられているのか調査をした．市販テキストは大学の授業などでよく使われているものを取り上げた．**表 -1** に構文を扱う順番を番号で示す．

繰り返しを教える順序はさまざまだが，表 -1 を見ると，if を最初に教え，do 〜 while を最後に教える傾向が高いことが分かる．条件分岐を教えることで，繰り返しの条件判断の理解につなげ，do 〜 while は使用頻度が低いために最後に教えるのだろう．

一方，while と for については，while より for を先に記述するテキストが比較的多いことが分かった．実際の授業でもこのような順番で教えている例が多いのではないかと考えられる．

実際 2012 年 8 月に本会の情報教育シンポジウム（SSS2012）のポスター発表で，参加者に質問したところ，for を先に教える教員が多かった．

❑ 理解のために必要なこと

固定回数を繰り返す概念は，小学生でも理解できるほど簡単であると感じているが，汎用言語の場合には変数を伴うためそれほど単純ではないことが分かった．さらに for 文の場合，初学者は括弧内に式が 3 種類もあることで複雑さを感じ繰り返しの仕組みまで理解できないことも考えられる．

今回は構文を中心に考えてみたが，実際には，変数の理解，終了条件の判定位置，繰り返しの入れ子など，繰り返しを実際のプログラムの中で活用するために理解しなければならない概念は多い．実際，

2 重ループになっている繰り返しは普通の高校生には到底理解できるものではない.

では，普通の高校生が，無理なく繰り返しの概念を理解するにあたって，自分の専門である数学や算数を理解させるための工夫を利用できないかと考えた.

たとえば，小学校に入学したばかりの児童は引き算ができないが，最初は「3－1」など繰り下がりのない計算から始め，次に「5－2」など大きな数の引き算ができるようになり，続いて「13－1」のように 2 桁の数を扱った後，「10－3」のように繰り下がりのある計算に進み，「98－21」のような 2 桁の大きな数を扱うといったように，小さなステップで段階的に細かく教えていくことで，結果として教室全員の子どもたちが引き算をできるようになっていく.

このような小さなステップで学習を進めることを目的として，繰り返しの構文を理解するための**図 -5** のような理解構造の例を考えてみた．小さなステップを実現するためには教える順序も大切である.

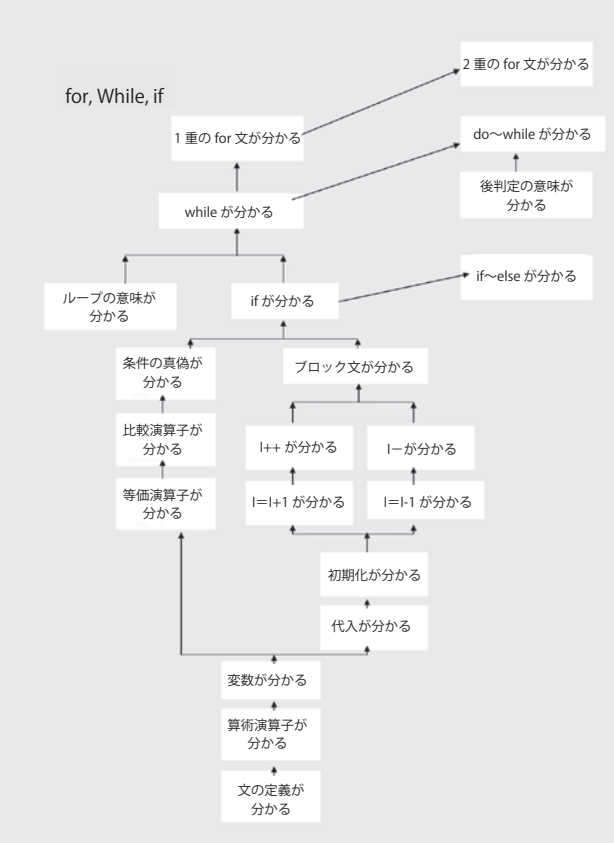

図 -5　理解構造

図 -5 では，それぞれの項目について，その概念を理解するために必要な前提条件を下位に示した.

たとえば，while を理解するためには，ループの意味，条件式，変数の理解が必要である．条件式を理解するために if 文で比較演算などを事前に学習する．また，while 文を学習する前に，変数の代入，インクリメントなどを学習しておく必要がある．テキストもこのような順番で記述してあるとスモールステップが実現でき理解しやすい.

差分分析ツール

繰り返しで理解が難しくなるもう 1 つの原因として，たくさんの新しい内容が繰り返しの部分で教えられているのではないかと考えた.

そこで明星大学の長慎也先生の開発した差分分析ツール“De-gapper”を用いて，前述の市販テキストの例題には新しい内容がどれだけ扱われているかを調べた．“De-gapper”は，プログラミング言語のテキストに出てくる例題について「初出の構文要素」の出現を検出する．現在 C と Java に対応しており，拡張が進められている．このツールでは，テキストや授業で使う例題に対して，「それまでの課題で出現した構文」と「新しく出現した構文」との差分を検出して表示する．表示には 2 種類あり，差分の程度を 2 種類の数値で示す集計表示と，具体的な初出個所をソースコード中に表示するコード表示である.

ここでは，例として，表 -1 のテキスト B を分析した結果を紹介する．例題 301 の“De-gapper”による分析結果を**表 -2** に，例題 301 のプログラムを**図 -6** に示す．今回取り上げた例題は，if 文が初めて使われる例題である．「if 文による条件判定」と「条件が成り立ったときの処理」の 2 個が初出のため，大きな初出（major）が 2 個と表示される.

さらに，「条件」の中にある「変数」と「数」，「条件」後の「文」はそれまでの例題で扱われたことはあるが，if 文の要素として扱われたことがないので小さな初出（minor）と考える．major と minor の個数を合わせたものを合計とする.

問題	合計	major	minor
301	8	2	6

表 -2　差分分析ツール De-gapper による分析結果

```
#include <stdio.h>
int main (void)
{
    int  vx;
    printf ("整数を入力してください:");
    scanf ("%d", &vx);
    if (vx % 5)
        puts("その数は5で割り切れません. ");
    return (0);
}
```

図 -6　例題 301

この分析結果を章ごとにまとめ平均を算出したものが**表 -3** である．これによると「分岐」では多くの major な初出内容が扱われていることが分かる．「繰り返し」では，major な初出内容も多いが，それよりも既出事項の活用となる minor の内容が多く扱われている．

一方，**表 -4** のテキスト D では「繰り返し」の後で「分岐」を扱っている．このテキストでは major な初出が均等になるよう工夫されている．

しかし，「分岐」では既出事項の活用である minor の数が際立って多い．

以上のことにより「分岐」後の「繰り返し」または「繰り返し」後の「分岐」で既出事項の活用が増え，両者を組み合わせた例題が多く扱われている．このように，分析を行うことで，テキストごとに著者の考える説明順の工夫があり，それを分析により明らかにすることが可能であることが分かった．また，同じ説明順であっても，例題ごとのステップ数を調査することで，無理のない難易度で説明が行われているかを検証できることが分かった．

まとめ

普通高校の生徒にプログラミングを体験させた経験から，プログラミングで躓きやすい個所の 1 つである「繰り返し」に着目し，基本的な概念から小さなステップで理解させることの可能性を検討した．小さなステップで学習させるためには，例題が一度に 1 つずつ (多くても数個ずつ) の概念が出現するように作られている必要がある．そこで，例題ごとに新規に出現した概念を分析して表示するツールを使用したところ，市販の教科書の例題について，著者がどのような考えで例題を作っているかをある程度分析できることを確認した．

今回は「構文」を手がかりに例題プログラムを分析する手法を用いたが，実際にはプログラミングを理解するためにはさまざまな概念の理解が必要である．今後は教科書等の分析を進めつつ，それを用いて学習したときの学生の理解度を評価するなど，検討を進めていきたいと考えている．

章	項目	合計	major	minor
1	変数，表示	2.33	2.17	0.17
2	演算，型	1.25	0.58	0.67
3	分岐	6.53	2.58	3.95
4	繰り返し	8.63	2.05	6.58
5	配列	5.33	2.60	3.80

表 -3　テキスト B 集計結果

章	項目	合計	major	minor
1	基本	2.83	2.83	0.00
2	式，演算，型	4.60	3.80	0.80
3	繰り返し	2.83	2.83	0.00
4	分岐	13.08	3.25	9.83

表 -4　テキスト D 集計結果

参考文献

1) 兼宗　進，久野　靖：ドリトルで学ぶプログラミング，第 2 版，イーテキスト研究所 (2011)．
2) 林晴比古：改訂新 C 言語入門 ビギナー編，ソフトバンククリエイティブ (1998)．
3) 柴田望洋：新版明快 C 言語 入門編，ソフトバンククリエイティブ (2004)．
4) 高橋麻奈：やさしい C 第 3 版，ソフトバンククリエイティブ (2007)．
5) Kochan, S. G. : Programming in C Third Edition, Sams Publishing (2005).
6) B. W. カーニハン／D. M. リッチー：プログラミング言語 C 第 2 版，共立出版 (1989)．

(2012 年 11 月 27 日受付)

保福やよい（正会員）　hohuku@amy.hi-ho.ne.jp

神奈川県立相模向陽館高等学校教諭．教科は数学，情報を担当．3 学年年次リーダをしながら，情報教育，校務の情報化を担当している．2012 年から大阪電気通信大学大学院に在籍．