2P1-10in1

迷路プログラミングを用いた初学者向けプログラミング教育

Program Learning for Beginner by Maze

中田 豊久[*1]
Toyohisa NAKADA

[*1]新潟国際情報大学
Niigata University of International and Information Studies

Learning programming is notoriously difficult. Many students fail to learn programming in the earliest stages in which they usually learn variables, if-then statements, and simple loops. This research has proposed a learning method in which learners try to create a program in order to escape from a maze. The concepts are (1) divided small learning elements and learning them stepwise, (2) a limited frame which is needed to consider the problem, and (3) multi-directional learning. The results used the proposed tool in a classroom of an university were that many students could create programs using loop statements, however, it is not true that students could make loop statements in other normal computer language.

1. はじめに

迷路プログラミングとは，迷路を解くことをプログラムによって解答することを本論文ではいうとする．この迷路プログラミングを，初学者のためのプログラミング教育に利用することを本研究の目的とする．

プログラミング教育は，近年ではプログラマー養成のような専門性の高い学びの領域から，小学生の国語や算数のように誰でもが身に付けている必要のある「汎用性の高い」学びの領域へ変容しつつある．2013 年 12 月にはアメリカ大統領のオバマ氏によるプログラミングを学びましょう，という動画が公開された[*1]．また世界各国においてプログラミングを義務教育の中で行うという動きがある[*2]．このような状況の中，国内では 2012 年度からの新学習指導要領において中学校の技術・家庭でこれまで選択科目であったプログラムと計測・制御が必修化された[*3]．しかし教育の現場では，多くの学生がプログラミングの初歩の段階で挫折しているという状況である [1].

本論文では，この初学者向けのプログラミング教育に対しての一手法として迷路プログラミングを用いることを提案する．その中で特に，これまであまり議論されてこなかった汎用ではなく専用を重視したプログラムの作成，プログラムを作成するだけでなく予めあるプログラムを学習者が指示通りに動かすなどの別の視点から見たプログラムの学習などについて議論する．

2. プログラミング学習の課題

例えば次のような JavaScript のプログラムを考える．このプログラムは，変数 values に入っている 1 から 10 までの数値を変数 sum に足し合わせ，最終的に合計を表示するプログラムである．このプログラムは，プログラミングを不慣れとする初学者は，直ぐには理解することができない．

連絡先: 中田豊久, 新潟国際情報大学, 新潟県新潟市西区みずき野 3-1-1, 025-239-3723, nakada@nuis.ac.jp

∗1 http://techcrunch.com/2013/12/08/obama-celebrities-politicians-and-tech-cos-come-together-to-launch-computer-science-education-push/

∗2 http://techcrunch.com/2014/02/04/uk-government-backs-year-of-code-campaign-boosts-funds-to-teach-code-in-schools/

∗3 http://jouhouka.mext.go.jp/common/pdf/kyouiku_tebiki.pdf

合計を求める JavaScript プログラム

```
var values = [1,2,3,4,5,6,7,8,9,10];
var sum = 0;
for(var i=0;i<values.length;i++){
     sum += values[i];
}
console.log(sum);
```

このプログラムを見て，初学者が混乱する理由としては次のものが考えられる．

- 見たことのない記号が分からない
  var, for, length, console.log などの予約語，複数種類の括弧の使い分け，+=のような見たことのない演算子，行の最後にある;(セミコロン) など.
- 書き方の規則が分からない
  values などのプログラム作成者が決めて良い変数名と予約語の違い，;(セミコロン) で終わる行とそうではない行の違い，.(ピリオド) のあるところとないところの違いなど.
- すでに持っている知識と異なる使い方が分かりずらい
  =は比較ではなく，代入であること.
- 実行する順序が分からない
  for の実行順序，配列の代入は繰り返しとは違うこと.
- 変数の意味が分かりずらい
  配列変数と単体の変数の違い，それぞれの変数の利用目的など.
- 目的と手法の整合性が分からない
  合計を計算するために，なぜこの方法を用いるのか，Excel の sum 関数のようなものはないのか？

簡単に列挙しただけであっても，この小さなプログラムには数多くの分からないことがある．for を forEach に変えて変数 i を消去しても，分からないことの総数はあまり変わらない．これらの「分からないこと」は，1 つ 1 つは小さなことであるため，おそらくプログラミングが不慣れな人であっても，十分に時間をかければ理解できることである．しかし現状のプログ

ラミング教育では，これらを一度に教えようとしているところに問題がある．その原因としては，プログラムを教育する側の人間が初めてプログラムを学んだ時に，比較的にこの辺りはスムーズに理解してしまい，そのために他の人もそうなのであろうと思ってしまうからである．

また，分からないことの最後の「なぜ合計を計算するのにこの方法なのか？」は，理系科目を敬遠する人には多い疑問である．この点をこれまでのプログラミング教育ではあまり議論されてこなかったが，本論文では重要な事の一つとして考えていきたい．

3. 迷路を解くことを題材としたプログラミング学習

3.1 迷路プログラミングの目的

プログラミングの学習者が，とある現象からルールを抽出してプログラムという形で記述し，また，プログラムの形で記述されたルールを正確に適用できるようになることを目的とする．迷路を脱出する例えばダイクストラ法などのアルゴリズムを学ぶのではなく，迷路内の複数の条件を整理して論理的に考えてルールを作り出すことを目的としている．また，迷路を脱出するという題材にした理由は，次に示す設計方針を実現しやすいと判断したからである．

3.2 設計

2. プログラミング学習の課題 において述べた課題を踏まえ，次のようにプログラミング学習環境を設計した．

単一性 学ぶべきことを 1 つに絞る．

専用性 問題を解くために検討すべきことを，その問題の範囲内に限定する．

多面性 プログラムを作る問題だけでなく，プログラム，データ，実行のすべてを問う問題とする．

「単一性」は，学習においては王道の 1 つともいえる．初めて何かを学ぼうとしたときに，1 つずつ積み上げるように学習した方がすべてを一度に全て学ぼうとするよりも効率的である．この点について，本研究では，従来のプログラミング学習よりもより細かい粒度で学習項目を作ることを試みる．

「専用性」は，問題に暗示的な条件等を作らないことである．暗示的な条件とは，例えば数字の配列から合計を求めるプログラムを作れという問題において，その解答は合計だけでなくすべての数字を掛け合わせるプログラムにもすぐに変更可能な拡張性を求めるようなものである．または，学習者が学習の初期段階では不要と思いがちな関数などを，今後に必要となるために学習させるようなことも，学習者は「関数を使わなくても書けるのに，なぜ，関数を使用しなければいけないのか」と混乱する元になる．この専用性は，適切に問題に制約を与えることによって作り出すことができる．例えば迷路プログラミングの場合には，条件式は 2 つの異なる経路の迷路を 1 つのプログラムで解かなければいけないという制約を与える．また，プログラムに使える 1 つのプログラム要素を表すブロックの数を制限し，if 文だけでなく else の使用を促すこともできる．繰り返し文も同様に，ブロックの数を制限することにより繰り返し文を使わなければいけない理由になる．変数は，同じ迷路の分岐で違う行動をとる場合に変数を使用しなければいけないことになり，そこで初めて学習者は変数を学ぶ．

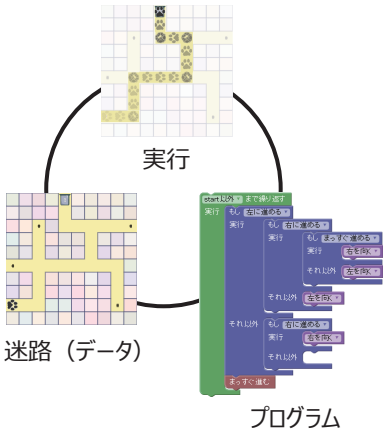

図 1: 多面的な学習

「多面性」は，プログラムを作るだけでなく，すでにあるプログラムを学習者がその通りに動かしたりすることも行うことである．プログラムは，この迷路プログラミングに限らず通常は，図 1 のように，プログラム，データ，そしてその実行結果によって構成されると考えられる．プログラムを作るという問題は，図 1 によって説明すれば，プログラムの部分を隠し，データと実行結果からそのプログラムを推定する問題とみることができる．そしてこの「多面性」は，隠すものがプログラムだけでなく，データを隠した問題，実行結果を隠した問題をも作るという意味である．例えばデータである迷路とプログラムが与えられて実行結果を求める場合には，学習者は自ら迷路の中を動くアイコンをコントローラーで動かし，プログラム通りに移動させることができればよい．

3.3 画面例

図 2 に，開発した迷路プログラミングの画面を示す．開発は JavaScript を基本とし，Scratch などで用いられるビジュアルプログラミングを実現する Google Blockly*4というライブラリなどを使用した．

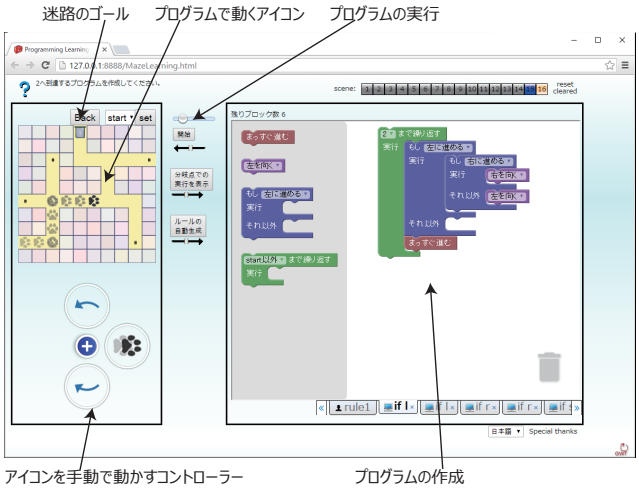

図 2: 迷路プログラミングの画面例

*4 https://developers.google.com/blockly/

3.4 問題の構成

16 問の問題が用意され，次の順番で学習者に提示される．学習者は，提示されいてる問題をクリアしないと次に進むことはできない．

1. ゴールまでの手順を 1 つずつ組み立てる　[問題 1,2,3,4]
2. 条件によって実行するプログラムを変える　[問題 5,6,7,8]
3. 繰り返しによって制限されたブロック数でプログラムを作る　[問題 9,10,11]
4. プログラムに合わせて迷路を作る　[問題 12]
5. 2 つのプログラムを統合する　[問題 13]
6. 複数の条件を統合したプログラムを作る　[問題 14,15]
7. 変数を使用したプログラムを作る　[問題 16]

3.5 問題例

図 3 に，迷路プログラミングの問題例を示す．問題 8 は，1 つのプログラムで左のアイコンも右のアイコンもゴールに到達するプログラムを作成する問題である．これは条件式を使わなければ解くことができない．また，学習者が使用できるプログラムのブロック数を制限しているため「もし」ブロックを 2 つ使うことはできない．そのために学習者は「もし」ブロックの「それ以外」をの部分を使わないといけないことになる．次に問題 9 は，繰り返し文を学ぶ問題である．学習者が使用できるブロック数は 2 つに制限されているため，繰り返し文を使用しなければ解くことができない．この問題 8 と問題 9 は，制約を与えることによって「専用性」を持たせた例である．次の問題 10 は，予めプログラムが与えられていて，それ通りに学習者がコントローラを使ってアイコンを動かす問題である．そして最後の問題 13 は，プログラムと迷路のゴールと経由地点だけが与えられて，アイコンがプログラム通りに動いたときに経由地点を通ってゴールに至る道を作成する問題である．学習者は壁をクリックして道を作り，アイコンをゴールに導く．この問題 10，問題 13 は「多面性」を具体化した例である．そしてこれらの問題は全て，「単一性」を保つために学ぶべきことを 1 つに絞っている．

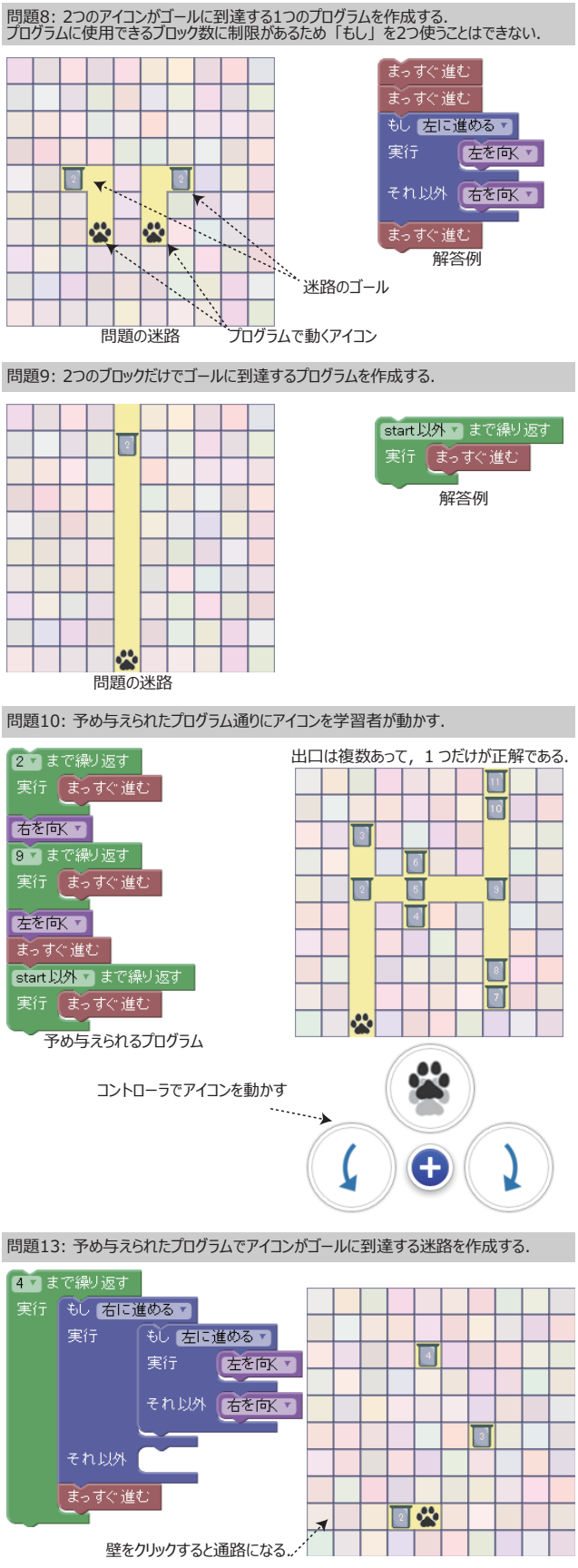

図 3: 迷路プログラミングの問題例.

4. 評価

4.1 大学 1 年生に対する実施

大学 1 年生の 24 名に対し，授業の中で迷路プログラミングを実施した．実施した授業は，HTML，CSS，JavaScript を学ぶ授業である．その授業の開始時点で迷路プログラミングを実施し，その後，JavaScript の講義を通して被験者のプログラミングに対する理解度を把握していった．

まず迷路プログラミングを実施した時の被験者のプログラムに対する理解度は，後で実施した JavaScript の学習から，授業開始時点では 2. プログラミング学習の課題 で示した合計を求めるプログラムはほぼ全員が理解できない状態であったと推測される．その状態で実施した迷路プログラミングの結果は，図 4 である．このとき被験者は，約 90 分間の時間を与えられ，迷路プログラミングのサイトの URL を指示されただけであとは何も説明を受けていない．

図 4 の結果から，すべての学生が繰り返し文の問題の少なくとも 1 つはクリアしていることが分かる．そして多くの学生は，2 つのプログラムを統合する問題で躓き，時間切れになっている．

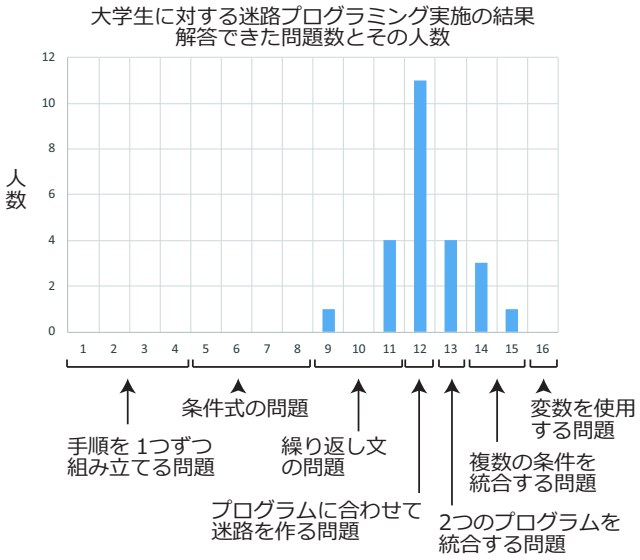

図 4: 24 名の大学生に対する 90 分間の試行結果．解いた問題の番号とその人数を表す．

4.2 考察

この実施結果は，4.1 大学 1 年生に対する実例 で既に述べたように，授業期間の開始時点で実施し，その後に JavaScript の通常の授業を行ったというものである．JavaScript の授業では，変数，条件式，繰り返し文と従来のプログラミング学習でよく用いられる順序で行っていった．その時の学生の理解度をみると，学生は少なくとも迷路プログラミングを実施した時には JavaScript の合計のプログラムを全く理解できない状態であったということである．ここから分かることは，迷路プログラミングで繰り返し文まで解答できたといっても JavaScript の合計を求めるプログラムが分かるというわけでは無いということである．言い換えると，迷路プログラミングを実施しても JavaScript の繰り返し文は学習していないということになる．または，迷路プログラミングのプログラムを理解していなくても学生がガチャガチャと適当にプログラムを作っているうちに解答にたどり着いてしまったとも考えられる．

この点から次の 2 点が今後の課題として考えられる．1 つは迷路プログラミングから JavaScript などのエディタで書くプログラムへの移行をどのように行うのか，を検討することである．しかし 1. はじめに で示したように，本研究の目的はプログラマー養成のプログラミング研究ではなく，一般教養としてのプログラミング教育である．よってエディタで書くプログラムへ移行する必要はそもそも無いという考え方もある．そのように考えると 2 つ目の今後の課題が浮かび上がる．それは，迷路は解けてもそれ以外はできない，という問題をどのように解決するかである．それは，迷路プログラミングのプログラムを他のジャンルの問題へ適用することを数多く学習することで解決できると考えている．迷路プログラミングの設計方針の中の 1 つにあった「専用性」を，いくつもの形で学習者に提示して学習者自身が徐々に「汎用性」の高い知識を身につけていくことを考えている．

学習効果は，学習者が学習に従事した時間の関数であるといわれることがある [2, 3]．近年に流行っている Active Learning[3]，PBL[5]，そしてゲーミフィケーションによる教育 [6] もすべてこの学習者の勉強時間の増加をするために行っていることであると見ることもできる．この迷路プログラミングについても，学習する項目を細分化したり，作る動機を与えたりとしたりしてきたが，これらはすべて学習者が学習に前向きに挑む準備をしているに過ぎないともいえる．そしてその結果として，迷路プログラミングによると，簡略化したものではあるが従来は初学者には難しいとされていた繰り返し文を学習者は扱うことができる結果を得た．よって少なくとも，きめ細かい学習支援は学習者にとって重要であるといえる．

5. おわりに

本論文では，プログラマー養成ではなく一般教養としてのプログラミング学習とその課題について示し，その 1 つの回答の提案として迷路プログラミングを示した．実際に大学生にその迷路プログラミングを実施したところ，JavaScript では理解できない繰り返し文が迷路プログラミングでは形が簡単になっているとはいえ解答できていた．しかし迷路プログラミングを行ったからといって被験者が JavaScript の繰り返し文を学習したとはいえない，という結果となった．

参考文献

[1] Richard Bornat, Saeed Dehnadi, and Simon: Mental models, consistency and programming aptitude, In Proceedings of the tenth conference on Australasian computing education - Volume 78 (ACE '08), Vol. 78. Australian Computer Society, Inc., 53-61, 2008.

[2] Carroll, John: A model of school learning, The Teachers College Record, Vol.64, No.8, pp.723–723, 1963.

[3] Gettinger, Maribeth and Seibert, Jill K: Best practices in increasing academic learning time, Best practices in school psychology IV, Vol.1, pp.773–787, 2002.

[4] Johnson, David W and Johnson, Roger T and Smith, Karl A: Active learning: Cooperation in the college classroom, Interaction Book Company Edina, MN, 1991.

[5] Hmelo-Silver, Cindy E: Problem-based learning: What and how do students learn?, Educational psychology review, Vol.16, No.3, pp.235–266, 2004.

[6] Salen, Katie: Quest to learn: Developing the school for digital kids, MIT Press, 2011.