Vol.2009-CE-102 No.19
2009/12/12

プログラムの動作を可視化する教育用プログラミング言語環境の提案

佐藤美奈[†]　安井浩之[†]　横山孝典[†]

本論文では，初中等向けの学校教育での利用を想定した教育用プログラミング言語環境を提案する．この言語環境は，正しい手順で処理しなければ正しい結果が得られないことを学ぶことを目的としており，実行の流れのアニメーション化や逆実行機能などにより，プログラムの流れの理解を助ける．また，教師の模範解答と生徒のプログラムを変数の変化に着目し比較することで，想定した動きにならない部分の発見を支援する．

Proposal of Educational Programming Language Environment which Visualizes Execution Process of Program

Mina Sato[†], Hiroyuki Yasui[†] and Takanori Yokoyama[†]

The paper proposes educational programming language environment that assumes to use in elementary and secondary education. This environment aims for users to learn that incorrect process makes only incorrect result and helps understanding execution process of program by a visualizing function and a reverse execution function of the execution process. Moreover, the discovery of the reason why the execution process differs from expected behavior is supported by focusing on the change of variables value of teacher's model answers and student's one.

1. はじめに

現在，コンピュータは生活に欠かせないものとなっている．しかし情報化社会が急に到来したため，情報技術の原理や仕組みをあまり理解しないまま利用しているという人が大多数を占めている．情報技術に関わる事件・事故が頻繁に起こり，ようやく2003年度より教科「情報」が高校で始まることとなった．その後も，構造計算書偽造事件や「1円61万株」誤発注事件，フィギュアスケート採点ミス事件など情報システムに関わる事件・事故が連続して起こり，混乱を引き起こした．これらのことから，情報処理学会情報処理教育委員会は日本の情報教育に対して危機感を抱き,「日本の情報教育・情報処理教育に関する提言2005」[1]や「2005年後半から2006年初頭にかけての事件と情報教育の関連に関するコメント」[2]など，多くの提言を発表している．事件・事故が起こった原因の一つとして，情報処理学会情報処理教育委員会では“情報処理と情報システムの原理に対する理解の欠如”が関わっているとしている．

だが実際の教育現場では，コンピュータや特定のアプリケーションの使い方や，インターネット絡みの事件・事故の多発から情報モラルなどに多くの時間が割かれている．その結果，情報技術の原理や仕組みについての理解は現在の初中等教育においてほとんど重要視されていない[3]．その理由の一つとして，情報技術は単なる道具であるから，原理や仕組みを理解する必要はなく使えればよいと考えられていることが挙げられる．しかし，科学技術の原理や仕組みの理解のために理科などの授業が行われているのに,情報技術にはそういった授業が必要ないというのはおかしいことである．

「2005年後半から2006年初頭にかけての事件と情報教育の関連に関するコメント」にあるように，“コンピュータは道具であり，人間が設計したことを，その能力の範囲内で，設計されたとおり正確に行うだけである．従って，設計した人間が間違えばコンピュータは正しくない答えを返すし，能力を超えたことをさせようとすれば動かなくなる．”ということを体験的に学ぶことが重要であり，学校教育で取り扱われるべきだと考える．そうすることで，情報技術と正しく向き合えるようになると考える．

コンピュータの原理を体験的に学ぶ方法として，プログラミングが注目を集めており，教育向けのプログラミング言語も多数提案されている．しかし，コンピュータの原理や仕組みをプログラミングと併せて学ぶことができるプログラミング言語環境は少ない．そこで本論文では，初・中等教育向けにプログラムとコンピュータ内の処理の原理や仕組みを学ぶことを目的とした,教育用プログラミング言語環境を提案する．プログラムやコンピュータの動作の流れをアニメーションにして見せる可視化機能や教師の模範プログラムと生徒のプログラムの動作比較を行う機能などを備えることで,初・中等学校の生徒でも十分理解できるものを目指す．

[†] 東京都市大学
Tokyo City University

Vol.2009-CE-102 No.19
2009/12/12

2. 既存の教育用プログラミング言語

2.1 教育用プログラミング言語とは

コンピュータの原理を体験的に学ぶ方法として，プログラミングが注目を集めている．その理由として，情報処理の仕組みを体験的に学ぶことができること，目的を達成するための考える力や試行錯誤しながら完成を目指す力を身につけられることなどが挙げられる．しかし，C や Java などの実用性を重視したプログラミング言語を扱うと，難しさが先行してしまい，生徒の意欲低下などの逆効果となることも危惧されている．その上，そういった言語を扱うと言語特有の命令や関数などを覚えることに時間を取られてしまい，プログラミングを学ぶのではなくプログラミング言語の書き方を学ぶことになってしまう．

そこで，そのような問題を避けるために教育用プログラミング言語が開発されている．教育用プログラミング言語は実用性を重視したプログラミング言語と比べ，機能や命令が少なく簡単にプログラムが組めるようになっている．具体的には，Squeak[4]やScratch[5]，LOGOなどがある．また日本でもドリトル[6]やPEN[7]，ビスケット[8]などの教育用プログラミング言語が開発され，ワークショップや学校の授業で利用されている[9].

2.2 日本で開発された既存のプログラミング言語の例

(1) ドリトル

ドリトルは LOGO のタートルグラフィックスをもとに，オブジェクト指向を簡単に学べるように開発された言語である．

```
ボール＝タートル！作る　ペンなし。
ボール：衝突＝タートル：跳ね返る。
ボール：飛ぶ＝「
　！（速さ）歩く。衝突＝０。！０（重力／２）移動する。衝突＝跳ね返る。
　速さ＝ｓｑｒｔ（速さ＊速さ＋重力＊重力＋２＊速さ＊重力＊ｓｉｎ（！向き？））。
　！（ａｓｉｎ（重力／速さ＊ｃｏｓ（！向き？））左回り。
」。
重力＝－０．２。ボール：速さ＝１０。
ボール！７０　向き　－３００　－５０　移動する　ペンあり（赤）線の色。

大工＝タートル！作る　消える　２０　線の太さ。
大工！ペンなし　－２００　４０　位置　０　向き　ペンあり　２５０　歩く　図形を作る。
大工！ペンなし　１７５　－１５０　位置　３５　向き　ペンあり　２５０　歩く　図形を作る。
大工！ペンなし　３５　－５０　位置　－９０　向き　ペンあり　１００　歩く　図形を作る。
タイマー！作る　０．０１　間隔　６００　回数「ボール！飛ぶ。」実行。
```

図 2　ドリトルによる放物衝突運動のソースコード[10]

GUI オブジェクトや音楽オブジェクトがあり，カメを操作して視覚的にプログラムを組むことができる．文法は記号が少ない構文とし，日本語の命令を使ってソースコードを記述する．Java によって実装されているので Java の環境さえ整っていれば Windows や Mac OS，Linux などで動かすことができる．図 2 にプログラムの例として放物衝突運動のソースコードを，図 3 にそのソースコードの実行結果の画面を示す．

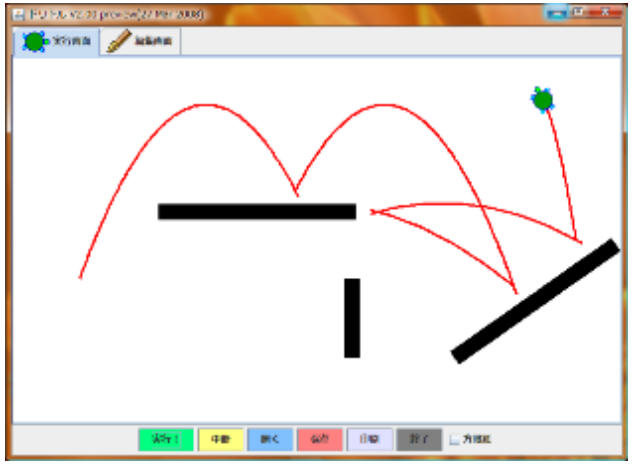

図 3　ドリトルによる放物衝突運動の実行結果の画面

(2) PEN

PEN の正式名称は Programming Environment for Novices である．PEN で用いているプログラミング言語は，大学入試センターの入試科目「情報関係基礎」で用いられている手順記述言語 DNCL，及び東京農工大学の入試用手順記述言語 TUATLE に準拠しており，これを PEN では xDNCL と呼んでいる．図 4 に PEN のプログラムと実行時の表示例を示す．

ソースコードの入力支援機能「プログラム入力支援ボタン」があり，容易にプログラムを入力できる．グラフィックの命令以外は日本語の命令で記述する．また，プログラムを実行している最中の変数の移り変わりを観察できる機能や，プログラムの実行速度を変えることや一時停止機能なども備わっている．Java によって記述されているので Java の環境さえ整っていれば Windows や Mac OS，Unix，Linux の上で動かすことができる．

Vol.2009-CE-102 No.19
2009/12/12

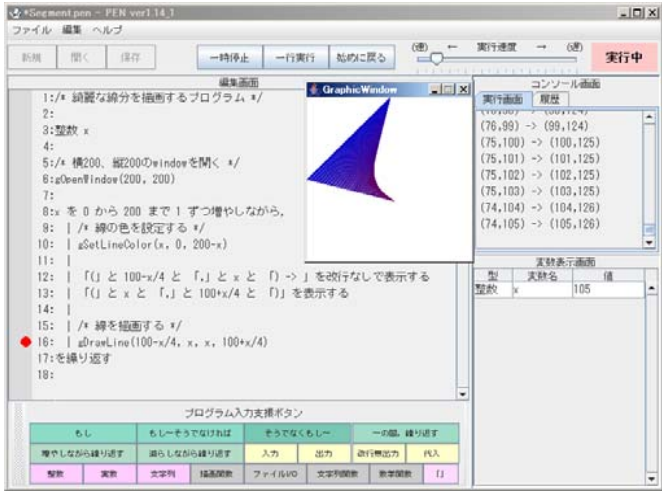

図 4　PENのプログラムと実行時の表示画面[11]

(3) **ビスケット**

ビスケットはルールベースで絵を動かすビジュアルプログラミングである．絵の変化する様子を描くことでプログラムを作り，実行すると絵が動く．命令を文字で記述することなしにプログラムを動かすことができる．これによりアニメーションやゲーム，動く絵本などが絵を描くだけで簡単に作ることができる．図 5 にビスケットのプログラムとその実行画面の例を示す．

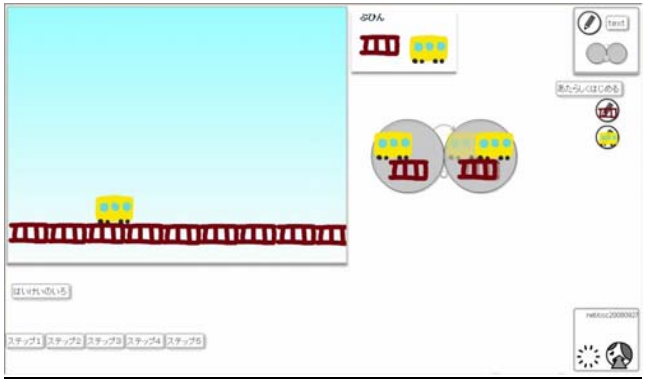

図 5　ビスケットのプログラムと実行時の表示画面[12]

2.3　問題点

前節では，日本で開発された教育用プログラミング言語のいくつかを大まかに説明した．このような言語が利用されることで，プログラミングの敷居が下がり，楽しく体験的に学ぶことができる．しかし，ドリトルやビスケットでは，プログラミングを楽しく体験させることに重点を置いているため，プログラムの流れの理解を支援する機能は十分といえない．PEN では，先に挙げた以外に実行している行にマーカーをつけることや変数の表示画面が用意されているなどの工夫がなされているが，どのように変数が計算され処理されるか，コンピュータが内部でどういう処理をしているか，どういう動きをしているかというところまではわからない．大まかにでもそういったことがプログラミングを通して学ぶことができれば，プログラムがどういうものかがわかるだけでなく，プログラムとコンピュータの繋がりや原理，仕組みがわかると考える．

3.　提案する言語環境

本論文で提案する言語環境は，プログラムとコンピュータ内の処理の原理や仕組みを大まかな流れで学ぶことを目的とし，プログラムの流れを可視化する教育用プログラミング言語環境を提案する．この言語環境が利用される場面としては，初・中等学校の授業を想定しているが，プログラミングの初学者向けの学習環境としても利用できるように想定している．図 6 は提案する言語環境のイメージ図である．この言語環境は大きく 3 つの部分にわかれている．左上の部分がソースコード表示記述部分でその下が実行結果を表示する部分である．そして右の大きな部分がプログラムの流れを可視化する部分となっている．

3.1　対象言語

提案する言語環境で対象とする言語は，Basic を基にしている．2.2 節に挙げた言語のように日本語の記述を採用しない理由は，プログラムは特別な言語で書くということをより印象づけるためである．日本語は普段使っている言語であるため，記述すればなんでもコンピュータで実行できると誤解を与えてしまう恐れもあると考えた．但し，記述が英語であることが利用者の苦痛にならないよう，提案の目的を達成する上で必要となる最低限の命令だけを扱うこととした．

具体的には，入・出力文，代入文，制御文の if-else 文と for 文・while 文，変数，配列，演算子，注釈文などの基本的なものだけとする．

Vol.2009-CE-102 No.19
2009/12/12

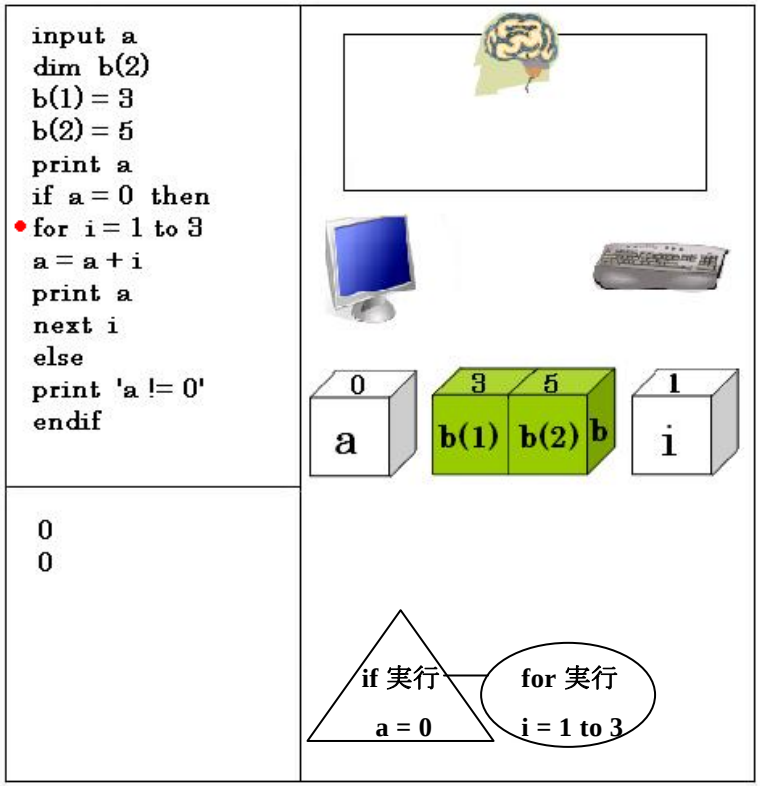

図 6　提案する言語環境のイメージ図

3.2　ソースコード表示記述部分

図 6 左上の部分がソースコードを記述する部分である．プログラム実行時に，現在実行している行がマーカー（左に出る赤い丸）で示され，一行ずつ逐次実行や，実行の一時停止，また後述する逆実行機能で実行状態を表示する役割も果たす．

3.3　プログラムの流れの可視化表示部分

図 6 右側部分はプログラムの流れを可視化する部分である．ソースコード表示記述部分のマーカーとプログラムの流れの可視化を同期させることによって，プログラムとコンピュータの動作の流れをわかりやすく追跡できるようにする．

可視化には，できるだけ直感的で，初・中等教育の生徒でも理解できるようなメタファーを使ったアニメーションが適していると考えた．そこで，演算処理を「脳」，入力を「キーボード」，出力を「ディスプレイ」，変数を「箱」で表すこととした．これらのアイコンは動作するときにハイライトしたり，アイコン間を値が移動したりすることで，動作内容を可視化する．また表示の下部には，制御構造を表現するためのメタファーを配し，制御文実行の様子とその構造を知らせる．

3.4　動作イメージ

ここで例として，図 7 のプログラムを用いて，可視化表示部分の動作イメージを示す．

```
dim   a(4)
a(1) = 3
a(2) = 1
a(3) = 4
a(4) = 2
for   i = 1 to 3
 for   j = i+1 to 4
  if   a(i) > a(j)   then
    b = a(i)
    a(i) = a(j)
    a(j) = b
  endif
 next   j
next   i
print   a(1), a(2), a(3), a(4)
```

図 7　プログラムの例（配列の並び替え）

dim　a(4)で，a という配列を表す 4 つに仕切られた立方体が表示され，続く 4 行で配列要素にそれぞれ値が代入される．続く **for　i = 1 to 3** の行で，for 実行と書かれた楕円が表示され，反復処理が始まったことを示す．また同時に，変数 i を表す立方体に初期値 1 が入った状態で表示される．

次の **for　j = i + 1 to 4** の行では，前の for 文の実行中に，さらに for 文の処理が行われる．そのため，for 実行と書かれた楕円の隣に，さらに for 実行と書かれた楕円が線で繋がって表示され，制御文の入れ子構造を表現する．また同時に変数 j を表す立

方体に初期値 2 が入った状態で表示される．

if a(i) > a(j) then の行では，if 文の条件式が成立するので，次に **b = a(i)**が実行される（図 8 参照）．この if 文は 2 つの for 文の中にあるため，2 つ目の for 実行と書かれた楕円の右に if 実行と中に書かれた三角形が表示される．

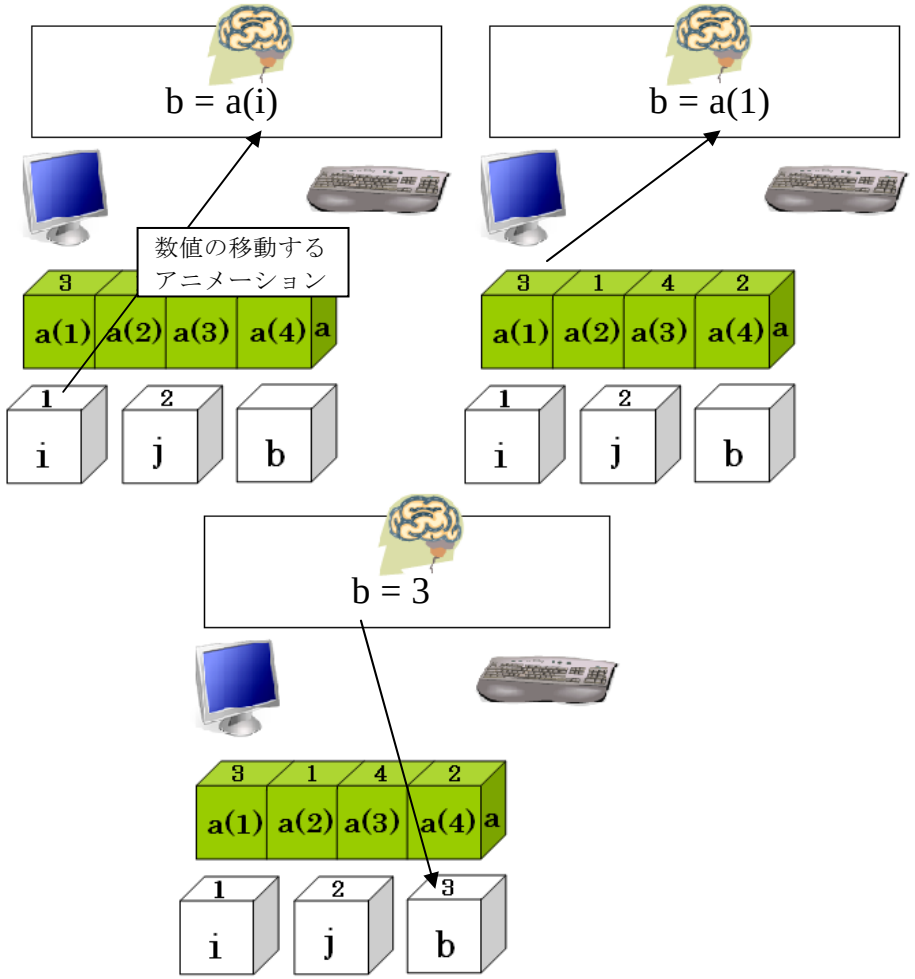

図 8 **b = a(i)** のときの可視化部分の状態の変化

a(i) = a(j)と **a(j) = b** の行も同様の動作となる．

endif の行で，if 実行の表示が消え，**next j** で j の値が 3 に変化，再び **for j = i+1 to 4** の行に戻る．

その後，for 文の処理が終わると，構造文の実行を示す楕円は順次消え，最後に **print a(1), a(2), a(3), a(4)**の行でディスプレイがハイライトして，実行結果部分に a(1)から a(4)までの値が出力される．

3.5 その他の理解を助ける機能

提案する言語環境では，動作の逆実行機能や変数の変化に着目した模範解答との比較をする機能により，プログラムとコンピュータの動作理解を助ける．

動作の逆実行機能は，プログラムを逆送りに実行するもので，通常の順送りと組み合わせて，動作を繰り返し確認できるようにする．また，誤りの発生した部分の特定にも使用できる．一方，模範解答との比較機能は，変数の変化に着目して，模範解答との違いを明らかにする．変数の値の変化が違う場合，どの変数のどの部分が違うかを特定でき，ソースコード表示記述部と可視化表示部にそれぞれ同時に表示することで両者の動作比較も可能とする．

3.6 システム構成

提案言語環境のシステムの構成と処理の流れを図 9 に示す．

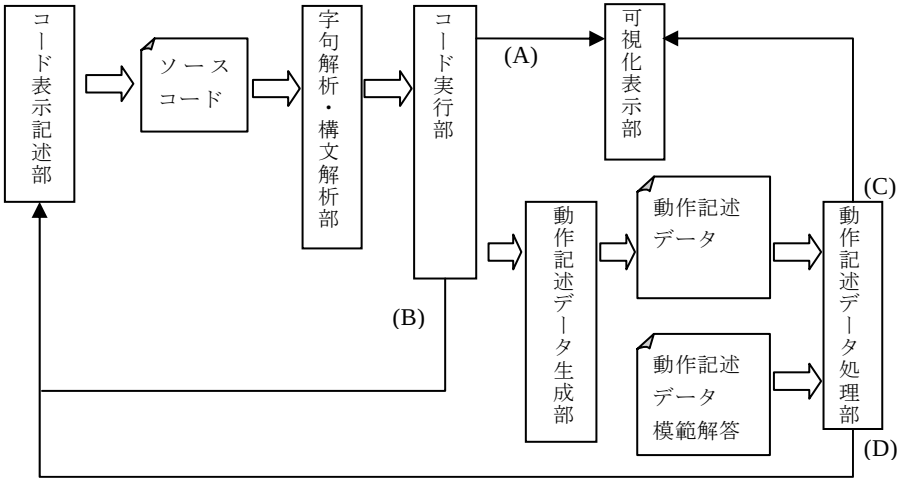

図 9 システムの構成と処理の流れ

図 9 で示すように，システムはコード表示記述部，字句解析・構文解析部，コード実行部，可視化表示部，動作記述データ生成部，動作記述データ処理部の 6 つの部分で構成されている．

コード表示記述部で記述されたソースコードは，字句解析・構文解析部で解析され，コード実行部分により実行される．可視化表示部はプログラムの動作をアニメーションで表示する部分であり，コード実行部での実行結果を表示する（図 9 の(A)）．それと同時に，コード実行部がソースコード表示記述部に現在実行している行を表示させる（図 9 の(B)）．その際，動作記述データ生成部はコード実行部での実行結果を記録し，動作記述データを作成する．

作成された動作記述データは，動作記述データ処理部で実現する逆実行や模範解答との比較を行うために使用される．動作記述データ処理部では，動作記述データを用いて，ソースコード表示記述部と可視化表示部に記録した動作結果を表示させる（図 9 の(C)，(D)）．これにより，逆実行を実現する．また，2 つの動作記述データを比較し，それぞれで用いられている変数の値の変化を対応付けることで同期させ，可視化表示部とソースコード表示記述部で両者を比較できるように表示する．

変数の変化の比較や逆実行に必要な動作記述データには，可視化のための動作の流れ，変数の変化などが記述されている．そのデータ一つ分が可視化動作（アニメーション）の 1 ステップに相当しており，変数の代入や変数の参照，配列の宣言，右辺の数値演算，入力文，出力文，制御文の開始，制御文の終了などである．

3.7 提案言語環境の授業での活用の例

提案言語環境は少ない命令しか利用できないが，次に挙げるようなプログラムを組むことが可能であり，プログラムとコンピュータ内の処理の原理や仕組みを大まかな流れで学ぶという目的を，十分達成できると考えている．

- 代入文，入・出力文，演算子を利用しての数値計算（図形の面積や平均値の計算）
- 制御文を利用しての分岐，反復処理（数当てゲームや九九表の作成）
- 配列を利用しての基本アルゴリズム学習（並べ替えや探索アルゴリズム）

4. まとめと今後の課題

本論文では，初・中等教育での利用を考え，処理の流れの可視化を行うことでプログラムやコンピュータの動作の理解を支援する言語環境を提案した．

情報技術を理解するには原理や仕組みの理解は必要なものであり，そのために提案した教育用プログラミング言語環境は大いに役に立つと考える．特に，逆実行機能や教師の模範解答との違いを比較しながら確認できる機能は，初・中等教育での利用に役立つと考える．

提案言語環境は現在構築中であり，今後，言語環境が完成した後で，実際に想定している教育現場や対象者に対しての試用を行う予定である．そして利用者の意見や感想を参考にして，現段階ではわからなかった課題の改善や改良を行っていく予定である．また，この言語環境を授業などで利用する場合に，どのような学習順序や学習方法が適しているか，どのように教えると生徒にわかりやすいかなどの提案も行っていきたい．

参考文献

[1] 情報処理学会情報処理教育委員会「日本の情報教育・情報処理教育に関する提言 2005」http://www.ipsj.or.jp/12kyoiku/proposal-20051029.pdf
[2] 情報処理学会情報処理教育委員会「2005 年後半から 2006 年初頭にかけての事件と情報教育の関連に関するコメント」, http://www.ipsj.or.jp/12kyoiku/statement2006.html
[3] 筧捷彦, 久野靖: 情報科学教育の重要性と情報処理学会の活動, 情報処理, Vol.50, No.10, pp.959-963 (2009).
[4] Squeak, http://www.squeak.org/
[5] Scratch, http://day.scratch.mit.edu/
[6] プログラミング言語「ドリトル」, http://dolittle.eplang.jp/
[7] PEN 公式サイト, http://www.media.osaka-cu.ac.jp/PEN/
[8] ビスケット, http://www.viscuit.com/
[9] 兼宗進, 阿部和広, 原田康徳: プログラミングが好きになる言語環境, 情報処理, Vol.50, No.10, pp.986-995 (2009).
[10] サンプル(V2)/放物衝突運動, http://dolittle.eplang.jp/index.php?%A5%B5%A5%F3%A5%D7%A5%EB%28V2%29%2F%CA%FC%CA%AA%BE%D7%C6%CD%B1%BF%C6%B0
[11] PEN クイックリファレンス, http://pen.moe.hm/files/Manual/PEN-QuickReference.pdf
[12] ビスケット指導者マニュアル, http://www.kodomo-abc.org/pdf/viscuit_manual.pdf