情報処理学会論文誌　教育とコンピュータ　Vol.4　No.1　57–69 (Feb. 2018)

論　文

# Webブラウザを用いたプログラミング学習支援環境 Bit Arrowの設計と評価

長島 和平[1]　長 慎也[2,a)]　間辺 広樹[3]　兼宗 進[4]　並木 美太郎[1]

受付日 2017年5月10日，再受付日 2017年9月9日，
採録日 2017年11月11日

**概要**：高等学校の次期学習指導要領では，必履修科目の中でプログラミングが扱われることになった．プログラミングの学習には，インストールの手間や予算などの理由から，Web ブラウザや表計算ソフトなどの既存の汎用ツールが使われてきた．今後は教育用に適したプログラミング環境が求められている．そこで著者らは Web ブラウザ上でプログラミングを学習できる環境「Bit Arrow」を開発した．本環境を用いることで，インストールが不要になるだけでなく，教員が学習状況を確認する機能や，生徒が簡潔なプログラムを記述するためのライブラリと分かりやすいエラー支援の機能を提供する．本報告では，高校で利用した授業内容を紹介し，ログの解析から判明した「文法エラーの減少」と「エラーへの対応時間の短縮」などの効果について報告する．

キーワード：プログラミング学習，Web ブラウザ，JavaScript

# Design and Evaluation for Bit Arrow: Web-based Programming Learning Support Environment

Kazuhei Nagashima[1]　Shinya Cho[2,a)]　Hiroki Manabe[3]
Susumu Kanemune[4]　Mitarou Namiki[1]

Received: May 10, 2017, Revised: September 9, 2017,
Accepted: November 11, 2017

***Abstract:*** All high school students will study programming from next teaching guidelines. To use suitable environment or tool for programming education will be more important. Some current textbooks treat JavaScript with text editor and browser. But there are some problems such as complex file management, hard to find whether error happened and required to type long statement. We developed Bit Arrow which an online programming environment. The environment shows error messages and where it happened. Also The environment have API to make long statement of JavaScript short. In this report we describe design and evaluation of Bit Arrow from users' log data.

***Keywords:*** Programming learning, Web browser, JavaScript

## 1. はじめに

次期学習指導要領に関する答申のとりまとめ [1] では，共通必履修科目「情報 I」の中にプログラミングが含まれており，高等学校においてプログラミングが必修になる．現在の共通教科「情報」には「社会と情報」と「情報の科学」の 2 科目があるが，プログラミングを扱う「情報の科学」を履修している生徒は約 2 割にとどまっている．今後，す

[1] 東京農工大学
Tokyo University of Agriculture and Technology, Koganei, Tokyo 184–8588, Japan
[2] 明星大学
Meisei University, Hino, Tokyo 191–8506, Japan
[3] 神奈川県立柏陽高等学校
Hakuyo High School, Yokohama, Kanagawa 247–0004, Japan
[4] 大阪電気通信大学
Osaka Erectro-Communication University, Neyagawa, Osaka 572–8530, Japan
[a)] cho@eplang.jp

べての生徒がプログラミングを学習できるようにするためには，授業に対するさまざまな支援が必要になる．

現行の共通教科情報「情報の科学」の教科書を見ると，プログラミングの学習については「Web ブラウザでの JavaScript の実習」「表計算ソフトでの BASIC の実習」などが扱われている [2], [3]．Web ブラウザや表計算ソフトは汎用のツールであり必ずしもプログラミングの学習に適したものではないが，多くの PC 環境でプログラミングのために追加のインストールをせずに使えることから利用されていると考えられる．一方，教育用の学習ツールをインストールするためには，PC の環境を変えるための許可が必要なことが多く，管理者権限の必要性，作業の手間，費用などの面から容易ではない．そこで，特別なインストール作業が不要で教育に適した Web ベースのプログラミング学習環境の有効性が指摘されている [4]．

本研究では，Web ブラウザで利用可能なプログラミング学習環境 Bit Arrow を提案する．従来のツールと比較して，特別な学習環境のインストールが不要になり，教員が学習状況を把握でき，言語とメッセージの工夫により学習に適したプログラミングが可能になった．JavaScript を用いた授業での学習をログから分析した結果，Bit Arrow を用いることで文法エラーの割合が少なくなり，学習者のエラーへの対応も早くなったことから，プログラミングの学習を支援する効果を確認できた．

本論文は，まず 2 章でプログラミング学習環境に求められる機能ついて述べ，3 章で本研究の目的を述べる．4 章で本環境の設計と機能について述べる．5 章で本環境が利用されたログとアンケートからの評価を述べ，6 章で結論と今後の課題を述べる．

## 2. 学校教育で求められるプログラミング環境

公立の高等学校などの学校教育では，実習用の PC は自治体によって導入され，教員はコンピュータの管理者権限を持っていなかったり，ソフトウェアの導入が制限されたりしていることが多い．共通教科「情報」の教科書では JavaScript や BASIC などの言語が扱われているが，プログラミングの学習のために特別なインストール作業が不要であるという理由だけでこれらのソフトウェアが使われていることが考えられる．

しかし，表計算ソフトは，プログラミング教育用に開発されているものではなく，表計算の実務処理を補助するためのものである．内蔵された言語を用いて数値や文字を出力するようなプログラムの実行を行うことができるが，グラフィックスを用いた実習などにはあまり使われない．また，エラーメッセージも画面に表示されるが，メッセージが適切にエラーを指摘しないこともあり，初学者が学習するためのサポートが不十分である．

Web ブラウザも同様にプログラミング教育のための環境ではなく，Web ページを表示するための環境である．グラフィックスを用いた実習も行うことができるが，記述する命令が長くなり容易ではない．エラーメッセージは開発者ツールなどから見なければならず，エラーが発生しても自動的にエラーメッセージが表示されるわけではないため学習者がエラーの原因を探すことは容易ではない．

今後，次期学習指導要領において高等学校でプログラミングが必修になったとき，教育用に開発されていない環境は学習者にも教員にも負担が大きく，円滑に授業を進められなくなる危険がある．次に，教育に適したプログラミング環境に求められる特徴をあげる．

- インストール不要な環境を使用すること
- コードの編集とプログラムの実行が同時に行えること
- エラー修正を支援すること
- グラフィックス・GUI が利用できること
- 教員を支援すること

次節以降では，これらの特徴を，表計算ソフトと Web ブラウザをはじめとする既存の環境が対応している機能と比較しながら述べる．

### 2.1 インストール不要

表計算ソフトと Web ブラウザはどちらも PC にインストールされていることが多い．一方，プログラミングを支援するソフトウェアを新たにインストールすることは難しいことから，インストールが不要な学習環境が望まれている．

### 2.2 コードの編集とプログラムの実行

コードの編集とプログラムの実行を同一のアプリケーションで行うことで学習者はプログラミングに集中でき，結果もすぐに見ることができる．表計算ソフトでは，アプリケーション中にエディタが組み込まれており，プログラムを記述することができる．一方一般的な Web ブラウザでは，JavaScript を実行できるが，そのコードを作成・編集するためにテキストエディタを用いなければならないため，Web ブラウザだけで作業が完結しない．テキストエディタと Web ブラウザの両方でファイルを開くような煩雑な操作が必要になる．

編集と実行を同じアプリケーション内でできる環境は数多く存在する [5], [6]．しかし，これらの環境はグラフィックス関連の命令の利用が難しく，グラフィックスを用いた自由な作品制作に適していない．

編集と実行を同じアプリケーション内で行い，グラフィックスを扱える環境もある [7] が，この環境は実習用 PC へのインストールが必要である．

Web ブラウザ上でコードの編集，実行，エラーの確認を同時に行える環境も存在する [8] が，学習者が個別にユーザ登録を行う必要があり，2.5 節で後述するような教員がク

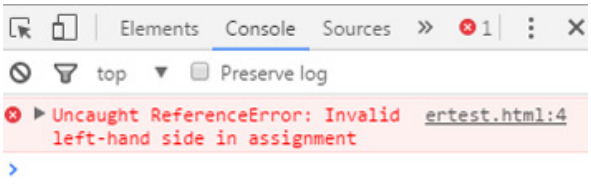


図 1　Web ブラウザ（Google Chrome）のエラーメッセージ
**Fig. 1**　An error message shown by Web browser.

ラスごとに学習者が作成したプログラムや学習者の編集・実行履歴を管理する仕組みは用意されていない．

### 2.3　エラー修正の支援

エラーが発生したとき，学習者がエラーを直すためには，エラーメッセージや，エラーがプログラム中のどこで起きているかを学習者に通知する必要がある．表計算ソフトでは，エラーの通知が行われるが，表示されるメッセージが十分ではなく，Web ブラウザでは特別な操作を行わなければエラーを確認できず，そのメッセージも分かりにくい（図 1）．Web ブラウザを用いて JavaScript のプログラミング授業を行った高校教員によると，エラーが発生したときに，エラーメッセージが表示されないという現象があったため，学習者は何をすればよいのかが分からずに作業が止まってしまい，それらのエラーへの対応に追われ授業を成立させることが難しかったという問題がある．

また，テキストエディタを用いた Web ブラウザによる学習では，括弧やダブルクォーテーションなど，開く記号と閉じる記号が対応して使われるような記号の閉じ忘れを防ぐために，開き記号が入力された時点で閉じ記号の入力補完を行うことで，エラーの発生そのものを減らすことができる．

文法エラーではないが，思いどおりの動きをしないプログラムもある．たとえばインデントが正しくないことで制御構造が分かりにくくなり，プログラムのバグを発見しにくくなる場合もある．このとき，メモ帳など OS に付属する編集環境を用いた Web ブラウザの学習では自動的にインデントがつかないため，プログラムの動きの把握が難しい．

エラーが発生することを抑えるために，テンプレートで学習者がまだ習っていないところや変更が不要な部分を編集できなくした環境や，画面にブロックを置くことによってコードを編集する環境もある [9], [10], [11]．しかし，テンプレートを用いることによって作品制作の自由度が低下してしまうこともある．また，高等学校では実用的なプログラムの作成過程になるべく近い体験ができるよう，ブロックによるプログラムの作成だけでなく，テキストを用いたプログラムの作成も必要であると考える．しかし，ブロックとテキストを両方学ぶ時数を確保できないことも考えられる．

表 1　表計算ソフトと Web ブラウザの機能
**Table 1**　Functions of spreadsheet and Web browser.

| | 表計算ソフト | Web ブラウザ |
|---|---|---|
| 用途 | 実務処理 | Web 表示 |
| インストール不要 | ○ | ○ |
| コードの編集 | ○ | × |
| エラー表示 | △ | × |
| グラフィックス・GUI | △ | ○ |
| 教員支援 | × | × |

### 2.4　グラフィックス・GUI の利用

プログラミング教育では，学習者に身近な題材を使用することが重要である．学習者が日頃慣れ親しんでいるゲーム，スマートフォンのアプリケーション，Web アプリケーションなどを題材にしてプログラムを作成させることで，身近なものがプログラムによって動作していることを体験させることができる．これらの題材には，グラフィックス，アニメーション（動きのあるグラフィックス），ボタンやテキストボックスなどの GUI が用いられており，学習者がそれらを簡単に扱えるようにする必要がある．表計算ソフトでは，グラフの描画などを利用することができ，JavaScript は HTML の要素を動かす命令を利用して，Web アプリケーションで使われるような GUI，グラフィックス，アニメーションなどを記述することができる．しかし，JavaScript で HTML の要素を動かして作品制作をするためには，`document.getElementById` のような長い命令が必要となる．このような長い命令を記述することはアニメーションを扱うときだけでなく，教科書に掲載されているサンプルにも含まれている．簡単に要素を操作できる短い命令を用意すれば，アニメーションを用いた作品プログラムの制作も，教科書に掲載されているような文字を出力するプログラムも簡単に書かせることができる．

グラフィックスを扱ったり，ゲームを作成したりすることができる環境も存在する [12], [13], [14], [15], [16], [17]．しかし，これらの環境はグラフィックス・アニメーションの描画に特化しており，GUI を扱うことが難しい．

### 2.5　教員支援

授業で利用するために，学習者を支援するだけでなく教員を支援することも重要である．具体的な機能として，クラスと学習者の管理，学習者の作成したファイルのクラスごとの管理，作品や課題の提出，課題の配布，実行のログと学習者の状況把握などがある．これらの機能は授業での利用を目的に開発された環境でないと実装されておらず，表計算ソフトや Web ブラウザにも搭載されていない．

クラス，学習者，課題を管理できる環境もある [18] が，開発環境は別途用意しなければならない．また，Web ブラウザだけで学習でき，教員が学習者の行動を詳細に把握で

きる環境もある [19] が，作成できるプログラムは文字の入出力に限定されていて，グラフィックスなどの題材は扱えない．

以上の特徴をまとめると**表 1** のようになる．

## 3. 本研究の概要

プログラミング教育には 2 章で述べたような教育用でない環境をやむなく使用するのではなく，教育用の学習環境を用意して使うべきである．本研究では 2 章で述べた要求される特徴を持ったプログラミング学習環境 Bit Arrow を提案する．Bit Arrow は Web ブラウザで実行を可能にすることで，現在用いられている表計算ソフトや Web ブラウザと同様にインストール不要で利用できる．また，教科書に掲載されているプログラムと，実際に社会で利用されている Web アプリケーションの仕組みを学習できることから，JavaScript を対象とする．初学者を助けるために，命令を短く記述でき，アニメーションを用いた作品制作も行えるライブラリを用意する．学習者自身がエラーを修正するのを手助けするために，エラーの場所の表示や，エラーメッセージの表示を行う．文法エラーへ対応する時間を減らすことで，分岐と反復やアルゴリズムの学習に時間を使わせる．

また，教員に学習者の情報を提供するために学習者のプログラムの実行ごとにプログラムと実行結果のログを収集する．5 章では，収集したログのデータをもとに学習者が起こしたエラーの割合やエラーに悩んでいた時間などから，プログラミング学習支援の効果を評価する．

## 4. Bit Arrow の設計

**図 2** に，本環境のシステムの構成を示す．本環境では，プログラミングはすべて Web ブラウザ上で行う．記述したプログラムは Web ブラウザのローカルストレージに保存され，実行か保存のボタンが押されたタイミングでサーバと同期される．学習者がプログラムの実行を行うと，実行結果か，エラーがあればエラーメッセージやエラーの場所が表示される．この情報で，学習者のエラー修正を支援する．また，学生のプログラムは，クラス情報とユーザ情報によって管理されている．

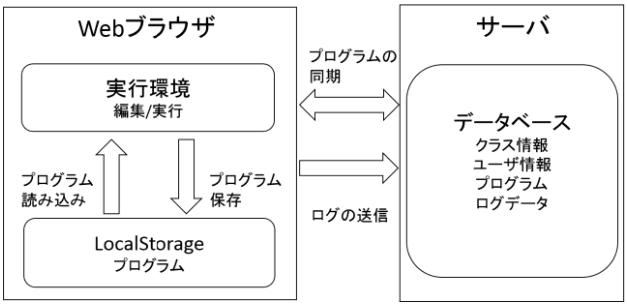


**図 2**　システムの構成図
**Fig. 2**　System structure.

既存の Web ブラウザ上の学習環境では，Web ブラウザの編集領域で書かれたコードを，サーバに転送してサーバ上でコンパイルや実行を行うことがほとんどである．その実装では実行ごとに通信が行われるが，本環境は Web ブラウザにダウンロードされ，コンパイルや実行を Web ブラウザで行うことで，サーバの役割をファイルの保存，ユーザの管理やログの蓄積などに限定することができ，サーバとの通信頻度を減らしネットワークの接続性や安定性に左右されずに実習ができる．現在稼働しているサーバは開発者が提供しており，同時アクセスユーザ 100 程度の場合は CPU3 コア，メモリ 4 GB 程度で安定して提供できている．また，Web ブラウザで動作するトランスレータを実装し，エラーの発生場所やエラーメッセージの表示などを可能にした．

教科書に掲載されているプログラム例などでは，HTML ファイルに HTML と JavaScript が記述されていたが，このようなプログラムを，HTML と JavaScript を別のセクションに分けて書けるようにした．分けて書くことにより，HTML と JavaScript を分離して考えることができる．

JavaScript では短い命令でプログラムを書くためのライブラリを用意した．このライブラリを用いることで，命令文が短くなりエラーを減らすことができる．また，このライブラリでは HTML の操作も簡潔に記述することができるため，アニメーションを用いた作品制作も助けることができる．

本環境では JavaScript のほかに C，ドリトルから言語を選択できる．Web ブラウザで実行するために，各言語を JavaScript に変換するトランスレータを独自に用意した．そのため，Bit Arrow で書かれたプログラムはすべてクライアント側で JavaScript への変換と実行を行う．なお，本論文では JavaScript を利用した学習について報告する．

本研究では，Web ブラウザ上でコードの編集とプログラムの実行を行う環境を提供し，入力の補助を行う．文法エラーに対しては，学習者が自力でエラーを直せるようなサポートと，エラーの発生を抑える工夫を行う．プログラミング学習支援環境に実装した機能を次に説明する．

### 4.1 Web ブラウザの編集・実行環境

Web ブラウザはコンピュータに標準で装備されているため，特別な準備をすることなく授業に導入することができる．授業以外においても，自宅での学習や課題の制作などを容易に行うことができるようになる．

#### 4.1.1 操作の簡略化と自動セーブ

従来の環境では，テキストエディタでプログラムを記述し，Web ブラウザで実行結果を見る，というウィンドウの移動が必要であった．Web ブラウザでコードの編集と実行を行うことで，エディタと実行画面のアクティブウィンド


図 3　Bit Arrow のコード編集画面
**Fig. 3**　Code editor of Bit Arrow.

ウを遷移する必要がなくなる．同時に，テキストエディタと Web ブラウザの両方でファイルを開く必要がなくなるため，煩雑な操作も不要になる．**図 3** に，コードの編集画面を示す．左には，ファイルリストが並んでおり，ファイルを作成すると HTML と JavaScript のファイルがそれぞれ 1 つずつ作成される．ファイルを開いた状態で実行を押すと，画面上に実行画面ダイアログを表示し，すぐに実行結果を確認することができる．

#### 4.1.2　Web ブラウザでの実行

プログラムを実行するとき，一般的なオンラインプログラミング学習環境ではソースコードをサーバに送信し，サーバ側でコンパイルと実行を行ってから結果を受け取りクライアント側に表示する．しかし，高等学校などではネットワークの速度が不足したり，一度に多くの接続を受付けることが難しかったりする場合がある．Bit Arrow では，プログラムをサーバへ送信してのコンパイルや実行を行わず，Bit Arrow が提供するすべての言語で JavaScript への変換を行うことで，クライアント側で実行を行い，サーバとの通信も削減できる．授業中に実行が集中することなどでネットワークのトラブルが発生したとしても実行は行うことができる．

### 4.2　文法エラーへの対策

#### 4.2.1　文法エラー発生時の対応

Bit Arrow ではエラーが発生した際には**図 4** のようなエラーダイアログを画面上に表示することで学習者へ通知を行う（表示される内容は言語によって異なる）．このダイアログは，文法エラーが発生するファイルを実行したときすぐに表示される．このことで，エラーの有無や原因の確認のために開発者ツールやコンソールを開く必要がなくなる．また，ダイアログによる通知の内容は，原因を示すエラーメッセージを表示させるだけではなく，エラーが発生した場所周辺にマークを同時に表示させた．これを学習者に提示することで，エラーが起こった原因やその場所の特定を支援することができる．これにより，従来の環境よりも簡単にエラーを確認することができ，エラーの修正を支援する．

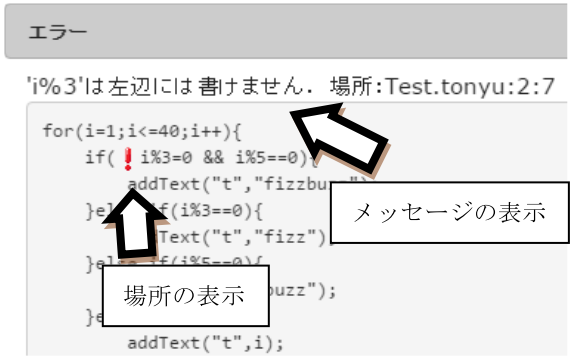


図 4　Bit Arrow のエラーダイアログ
**Fig. 4**　Error dialog box of Bit Arrow.

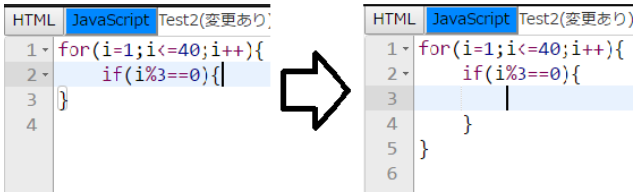


図 5　対応する括弧の自動入力
**Fig. 5**　Automatic input of corresponding symbol (a curly bracket is inserted in this figure).

#### 4.2.2　入力補助による文法エラー発生の防止

初学者の打ち間違いや打ち忘れなどがもととなることが多い文法エラーの中には，その発生をあらかじめ防ぐことができるものもある．たとえば，if 文や for 文のブロックを表す波括弧を開いた後に閉じ忘れるものである．波括弧の数が合わないとき，その原因がプログラムの途中にあったとしても，エラーメッセージではプログラムの最後で閉じ波括弧が足りないかのような指摘をされることもある．そこで，**図 5** に示すように「{」を開いた後に改行するタイミングで対応する「}」を自動入力することで，打ち忘れを防ぐ．波括弧のほかにも，丸括弧や HTML の閉じタグ，ダブルクォーテーションのように対応して使われる記号などを自動で入力するなどで，エラー発生の減少を支援する．

#### 4.2.3　自動インデントによるプログラムの読みやすさ向上

反復や分岐を多く使うプログラムを扱うと，ブロックの数が増える．同時に入れ子構造も使われるようになる．このとき，適切なインデントを付けることでプログラムが見やすくなる．しかし，メモ帳などの標準的なテキストエディタではインデントが自動で付かない．ブロックの範囲を分かりやすくするため，Bit Arrow では**図 6** に示すように実行または保存を行うタイミングで自動的にインデントを付ける．波括弧の数が合わないときにも，インデントを見ることでどこを閉じ忘れているかなどに気付きやすくする．また，実行はできたが想定した動きをしないときにも，ブロックの範囲を見やすくすることで対応しやすくなる．これにより，エラー発生数の減少と，エラー発生時の修正時間の短縮を支援する．

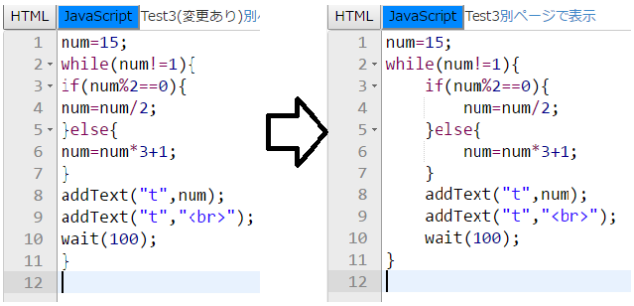


図 6　自動インデント
**Fig. 6**　Automatic indentation.

```
<html>
<body>
<script>
jcode=Math.random()*3;
if(jcode<1){
    document.write("Gu-");
}else if(jcode<2){
    document.write("Choki");
}else{
    document.write("Pa-");
}
</script>
</body>
</html>
```

図 7　HTML と JavaScript を分けないで書いたプログラム
**Fig. 7**　A JavaScript program embedded by HTML.

### 4.3　初学者用 JavaScript

#### 4.3.1　HTML と JavaScript の分離

教科書に掲載されているプログラムでは，HTML の script タグの中に JavaScript のプログラムが書かれている．しかし，HTML の中に JavaScript を書くことで，それぞれ異なる文法が一緒になってしまい初学者が理解することは容易ではない．HTML は静的な文書構造を設定し，JavaScript は HTML で設置した要素を操作するという別の働きをするものである．同一のファイル内に書くことによる混乱を防ぐために，Bit Arrow では HTML と JavaScript を分けて記述できる．じゃんけんプログラムを通常の JavaScript で記述すると，**図 7** のようなプログラムになる．ランダムで 0 以上 3 未満の数値を生成し，その値によって画面上にグー，チョキ，パーを示す文字列が表示されるものである．

このプログラムを，初学者用 JavaScript で実装すると**図 8** になる．図 8 には，Bit Arrow で書かれる HTML と JavaScript を示しており，HTML では JavaScript の結果を書き込む領域を name 属性付きで設定している．そして，初学者用 JavaScript のプログラムで，HTML で設定した領域に文字列を書き込む処理を行っている．HTML と JavaScript を分けて記述させることで，同一ファイルに HTML と JavaScript の異なる文法が混在することによる混乱を防ぐ．なお，図 8 では分かりやすいように乱数の値

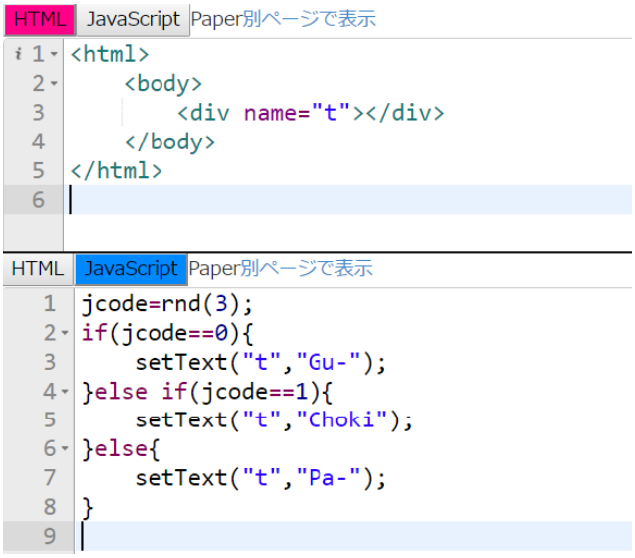


図 8　HTML と JavaScript を分けて書いたプログラム
**Fig. 8**　A JavaScript program separated from HTML.

を整数値にした例を扱っている．各命令の詳細は 4.3.2 項で後述する．

#### 4.3.2　初学者用 JavaScript ライブラリ

「情報の科学」の教科書に掲載されている JavaScript のプログラムには，結果の表示に `document.write` が用いられていることがある [2], [3]．これにより記述しなければならない命令が長くなり，初学者に書かせることで，打ち間違いによるエラーが増加するおそれがある．

図 7 のプログラムでは，乱数を取得するためにまず `Math.random` で 0 から 1 未満の小数を生成し，その値に 3 をかけて乱数を 0 から 3 未満にするというステップをふまなければならず，この計算式についても説明しなければならないため手間が多い．また，もし整数の乱数値を求めたいときには，これに `Math.floor` 命令を付け足す必要がある．Bit Arrow を用いた図 8 のプログラムでは，乱数の取得には `rnd` 命令が用いられている．この命令は，0 から引数未満の整数を返すものである．そのほかにも，`document.write` 命令を用いていた画面への表示は，`setText` 命令が使われる．これらのプログラムを用いることで，JavaScript で使用されていた長い命令を短くして文法エラーを減らすことができる．

**図 9** には，実際に使われていたプログラムの例を示す．ここでは `wait` 命令が使われており，ゆっくり処理させることができる．初学者用 JavaScript では，プログラムに 1 行命令を書き足すだけで，プログラムをゆっくり動かして動作を確認することができる．

また，アニメーションなどを用いた作品制作を補助する命令も用意している．たとえば，テキストエディタで書いた画像を右に動かし続けるプログラムを**図 10** に示す．画像の要素を取得するために，`document.getElementById` を使用しているほか，移動させるために `position`，`left`，

表 2　初学者用 JavaScript で使用できる命令の一部
**Table 2**　Typical functions available in JavaScript for novice.

| 命令 | 説明 | 通常の JavaScript での命令 |
|---|---|---|
| setText(要素名,文字) | 要素に文字を設定 | document.write(文字); |
| addText(要素名,文字) | 要素に文字を書き足す | document.write(文字); |
| onClick(要素名,命令) | 要素が押されたときの動作を設定 | element.onclick=命令; |
| onTouch(命令) | 画面が押されたときの動作を設定 | element.addEventListener("touchmove",命令); |
| move(要素名,x,y) | 要素を x,y 座標に移動 | element.style.left=x; element.style.top=y; |
| rotate(要素名,角度) | 要素を指定の角度回転 | element.style.transform="rotate("+角度+"deg)"; |
| resize(要素名,横,縦) | 要素を指定の比率に拡大/縮小 | element.style.transform="scale("+横+","+縦+")"; |
| rnd(値) | 0〜値未満の整数を返す | Math.random(); |
| wait(時間) | 指定された時間(ms)処理を中断 | setInterval(繰り返す処理,時間) |

HTML　JavaScript　FizzBuzz別ページで表示

```
for(i=1;i<=50;i++){
    if(i%15==0){
        addText("t","FizzBuzz");
    }else if(i%3==0){
        addText("t","Fizz");
    }else if(i%5==0){
        addText("t","Buzz");
    }else{
        addText("t",i);
    }
    addText("t","<br>");
    wait(100);
}
```

図 9　初学者用 JavaScript で記述した FizzBuzz プログラム
**Fig. 9**　A FizzBuzz program written in JavaScript for novice.

```
<html>
<body>
<img id="n" src="images/neko.png">
<script>
x=10;
s=document.getElementById("n").style;
s.position="absolute";
s.top=50;
setInterval(function(){
  s.left=x;
  x+=5;
},100);
</script>
</body>
</html>
```

図 10　一般的な JavaScript で記述したアニメーション
**Fig. 10**　An animation program with common JavaScript.

top を設定する．また，一定間隔で右に動かし続けるために setInterval を用いなければならない．Bit Arrow では，同じ動きをするプログラムを図 11 のように記述できる．このほかに用意したライブラリの一部と，それを通常

HTML　JavaScript　Anim別ページで表示

```
x=10;
while(true){
    move("n",x,50);
    x+=5;
    wait(50);
}
```

図 11　初学者用 JavaScript で記述したアニメーション
**Fig. 11**　An animation program written in JavaScript for novice.

の JavaScript で記述した場合の命令を表 2 に示す．

### 4.4　ログの収集

本環境では，プログラムが実行されるたびに，実行された時間や実行結果，プログラムのログを収集する．このデータは，授業支援のために教員に提供することを目的として収集している．収集したデータを次にあげる．

(1) ユーザ ID，クラス ID
(2) 実行を行った時刻
(3) 実行結果（実行成功，エラー）
(4) エラーメッセージ
(5) 実行時のプログラム

ログは実行ごとに収集することで，学習者の学習過程を追跡できるようにした．本論文執筆時においては提供を予定している段階であるが，このログデータからエラーが続いているユーザや，そのエラーの原因，作業が止まっているユーザといった授業の進行を補助する情報を教員へ提供することができる．

5.1 節の評価は，収集されたこれらのログデータから，学習者の利用状況の分析を行う．

## 5.　Bit Arrow の利用と評価

本論文では，Bit Arrow を 2 種類の授業で利用した．1

HTML JavaScript Kadai1別ページで表示

```
1  for(i=1;i<=50;i++){
2      if(i%2==0){
3          addText("t",
4          "<img src=images/neko1.png>");
5      }else{
6          addText("t",
7          "<img src=images/bluefish.png>");
8      }
9  }
```

図 12　交互に画像表示するプログラム

**Fig. 12**　A program which displays images in alternate shifts.

HTML JavaScript Kadai2別ページで表示

```
1  for(i=1;i<=50;i++){
2      if(i%2==0){
3          addText("t",
4          "<img src=images/neko1.png>");
5      }else{
6          addText("t",
7          "<img src=images/bluefish.png>");
8      }
9      if(i%10==0){
10         addText("t","<br>");
11     }
12 }
```

図 13　交互に画像表示し，一定間隔で改行するプログラム

**Fig. 13**　A program which displays images in alternate shifts and inserts breaks at regular intervals.

つは，分岐や反復といったプログラミングの基本的な概念を学習するための授業で，もう 1 つはアニメーションを用いたゲーム制作を行う授業である．基本的な概念を学習するための授業の評価は，4.4 節で述べたログデータをもとに行う．ゲーム制作を行った授業は，学習者へのアンケートをもとに評価する．

### 5.1　分岐や反復の学習による実践

#### 5.1.1　分析の対象

Bit Arrow で初学者用 JavaScript を利用していた高校 1 年生 170 人の実行時のログを高校教員と協力して調査した．ここでの利用者は，プログラミング経験がない初学者がほとんどである．なお，4.3 節で例示したプログラムは，この 170 人の利用者が実際に記述していたものである．はじめに「奇偶判定」,「じゃんけん」のプログラム（図 8）が作成され，次に「FizzBuzz」のプログラム（図 9）が作成されている．最後に，**図 12**，**図 13** のプログラムが作成されて，これを通じて分岐と反復を学習している．なお，初学者用 JavaScript を用いた授業を行った生徒には，一般的な JavaScript とは異なる言語を用いていることを実習開始前に説明した．

比較対象として，同じようにプログラミング経験のない同じ高校の高校 1 年生 40 人が同様の内容をテキストエディタで行った際のデータを用いる．テキストエディタで

表 3　1 人あたりの文法エラーの平均回数と修正時間

**Table 3**　Average of syntax error occurrences and error correction time.

| 環境 | 文法エラーなし | 文法エラーあり | 修正時間 |
|---|---|---|---|
| テキストエディタ | 40.9 (69.6%) | 17.9 (30.4%) | 190.6 秒 |
| Bit Arrow | 85.2 (79.2%) | 22.3 (20.8%) | 105.0 秒 |

は，Bit Arrow のようにログを収集することができないため，ファイルを保存するタイミングでログを収集することができるアプリケーションを開発・使用した．ここで開発したテキストエディタでは，以下のログが収集できる．

(1) ユーザ ID
(2) 実行を行った時刻
(3) 実行時のプログラム

テキストエディタを用いた生徒の実行結果とエラーメッセージはログに残らないため，後からログに保存されているプログラムを担当教員が実行し直し確認した．

#### 5.1.2　文法エラーの修正支援の検証

はじめに，1 画面上での編集と実行を実装したことにより学習者がプログラムを実行した平均回数が増加しているかを測るためすべての実行を集計した．さらに，短い命令を使えることで文法エラーの発生を減らすことができているかを確認するため，すべての実行を文法エラーが出たものと出なかったものに分けて集計した．そして，エラー発生時に学習者が自力でエラーを直すことができるよう表示させるエラーダイアログによってエラーの修正にかかる時間が短くなっているかを確認するため，エラーが発生してからそれを修正するまでの時間を算出した．これは，エラーが連続で発生したかどうかなどは考慮せず，前回実行時はエラーがなかった状態のときに，処理系がエラーを出した時間から，次にエラーがなく実行されるまでの時間を計測している．

Bit Arrow のユーザ 170 人が利用していた 10/6 から 11/8 までのログを調べると，実行の総数は，18,282 回となり，約 1 カ月間で 1 人あたり 85.2 回の実行を行っていた．テキストエディタでは，同様の内容を行った期間で 2,351 回の保存がされており，1 人あたり 40.9 回となった．また，この実行のうち，文法エラーがなかったものとあったものの回数の割合を調べたところ，**表 3** の結果となった．文法エラーがなかったものの割合が約 79%となり，文法エラーの割合が少なかった．このことから，Bit Arrow では煩雑な操作を減らすことで多くのプログラムを記述しての実行や試行錯誤をさせることができていたといえる．また，処理手順を簡潔に書けるよう用意したライブラリにより，文法エラーの割合を減らすことができたと考えられる．文法エラー発生時，その発生場所を表示してエラーの修正を支援できているか，エラーの修正にかかった時間を調べた．この時間は，テキストエディタを用いた授業では，約

190.6 秒かかっていたが，Bit Arrow では，文法エラーが発生してから修正するまでに約 105.0 秒と減少していた．Bit Arrow を用いることでエラーの修正時間を約 85 秒短縮させることができており，授業を担当した教員からも，これまでの環境ではエラーが起こるとエラーメッセージが表示されず何をすればいいか分からず固まっていた生徒が多かったが，ダイアログでエラーが分かることで前向きにそれを直そうとしている生徒が増えていた様子が見て取れたことから，生徒のエラー修正に向かう意欲を維持できていることを確認できた．

エラーの修正にかかる時間を短くすることができたことから，Bit Arrow でエラーの発生場所を表示することで，学習者が自力でエラーから抜け出すことを支援できたといえる．素早いエラーの修正を支援することで，プログラムの構造などの本質的な部分の理解に費やす時間を増やすことができる．また，生徒が自力でエラーから抜け出すことを支援することで，授業などにおいては生徒のエラーへの対処に関する教員の負担も軽減させることができているといえる．

一方，一部には JavaScript ファイルに HTML を記述したことによるエラーもあった．HTML と JavaScript のファイルを分けることでプログラムを分かりやすくする狙いがあったが，このログで用いられていたじゃんけんや FizzBuzz のような短いプログラムにおいては，1 つのファイルに書くことのほうが簡単になると考えられる．

### 5.1.3　課題作成中の実行の検証

文法エラーの対応にはあまり時間を使わせずに，実行はできるが思いどおりに動かない原因を考えさせることが，アルゴリズムやプログラムの構造を考え，理解させることにつながると考え，課題のプログラムを作成している間の実行の成否を調査した．処理系はエラーを起こさないが期待した動作をしない場合，実行できたかどうかを見るだけでは検出することができない．そこで，利用者のログから，図 12，図 13 のプログラムを作る過程のプログラムを実行して観察した．

**図 14** に，テキストエディタと Bit Arrow での実行の割合を示す．ここでは図 12 などのプログラム作成に取り組んでいた間の結果のみを対象としたため，5.1.2 項で述べた全体の文法エラーを出した割合とは異なる．文法エラーを出した割合は，分析期間全体の割合に比べ増加したが，テキストエディタでは 49.1%あったのが Bit Arrow では 36.1%と減少した．文法エラーはないが思いどおりに動かなかった実行は 24.6%から 32.6%，思いどおりの動きをするプログラムが 12.2%から 18.8%へ増加した．その他の実行の中身については，図 12 のプログラム作成前に作られた FizzBuzz のプログラムを実行していることが多く，これを実行しながらアルゴリズムを考えていたと推察できる．文法エラーの割合は，Bit Arrow で約 13%減らすことができた．また，文法エラーはないが思いどおりの動きをしない実行は約 8%増加した．思いどおりの動きをしない実行は，どうすれば思いどおりに動かすことができるかというアルゴリズムを構築する必要がある．Bit Arrow を利用することで文法エラーの割合が減り，思いどおりの動きをしない実行と思いどおりの動きをする実行が増えたことから，プログラムの構造の学習に時間を使わせることができたといえる．

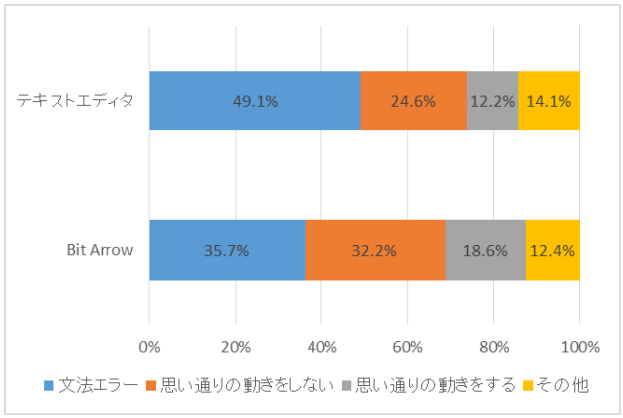


**図 14**　課題作成中の実行成否の割合
**Fig. 14**　Rate of execution success/failure.

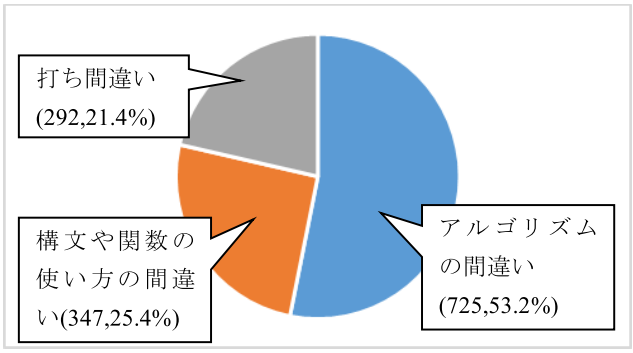


**図 15**　思いどおりの動きをしない原因の内訳
**Fig. 15**　Breakdown of causes of execution failures except syntax errors.

### 5.1.4　思いどおりに動かない実行の原因の検証

実行はできるが思いどおりに動かなかった原因が，実際にアルゴリズムやプログラムの構造に関わることかを調べるため，その原因を調べ分類した．**図 15** に，Bit Arrow での実行時，文法エラーはないが思いどおりにいかなかった原因の内訳を示す．図 12，図 13 のプログラムを作成中の実行ログから，間違えた原因をアルゴリズム，構文や関数の使い方，打ち間違いに分け，それぞれの割合を示す．最も多いのはアルゴリズムの間違いであった．これは以前に作った FizzBuzz のプログラムを改造しながら作成している利用者が多く，FizzBuzz のプログラムが残っていたことが原因の多くを占め，53.2%となった．この間違いに対応するために，表示がどのタイミングで行われるか，いつ行うことが正しいかといったアルゴリズムを考えさせるこ

表 4　構文や関数の使い方の間違いの主な原因と出現回数
**Table 4**　Breakdown of misuse of syntax or function.

| 原因 | 回数 |
|---|---|
| 二重ループに同じ変数を使用 | 88 |
| ループ条件の誤り | 53 |
| ループ変数加算忘れ | 43 |
| ループ変数初期化なし | 15 |
| 分岐判定とループ変数が違う | 14 |
| その他 | 134 |

表 5　打ち間違いの主な原因と出現回数
**Table 5**　Breakdown of typo.

| 原因 | 回数 |
|---|---|
| URL 間違い | 79 |
| src を scr と打ち間違い | 60 |
| >がない | 23 |
| ++ と書くところを tt と書いている | 23 |
| メソッド名間違い | 17 |
| その他 | 90 |

とができる．

構文や関数の使い方が原因となったものは 25.4%となった．この間違いの主な原因とそれが起きた回数を**表 4** に示す．最も多かった原因は，二重ループに同じ変数を使用していたことであった．これは図 13 のように if 文で改行を判断せずに二重ループで改行を判断しようとした学習者によるものであった．ほかにも，ループの条件やループ変数の初期化と加算が多く表れており，反復の使い方が原因となることが多かった．こうした反復や分岐の使い方に起因して思いどおりに動かないプログラムを直そうとすることで，分岐や反復の概念を学習させることができていると考えられる．

一方，打ち間違いが原因で思いどおりに動かないプログラムも 21.4%あった．この原因を**表 5** に示す．上位を占めたのは URL 間違いや img タグ内の src を scr と打ち間違えていたことなどであった．これは，図 12 のプログラムで画像を表示させようとしている部分で，文字列の中の HTML の記述を間違えていたものであった．JavaScript のメソッド名を間違えたり，セミコロンを忘れたりといった打ち間違いは，Bit Arrow で文法エラーとして通知することができるが，文字列の中身を間違えても，エラーメッセージは表示されないため，間違えた場所を探す必要がある．打ち間違いに起因するような多くの文法エラーを削減することはできたが，それでもなお，このような打ち間違いによりうまく実行ができないことがあることが分かった．これを修正する作業もプログラムの本質的な概念を学習できているとはいえない．

処理系がエラーを出さないが思いどおりに動かない原因

HTML　JavaScript　Game別ページで表示

```
1  x=100;y=300;
2  bx=[0,100,200,300,400];
3  by=[0,10,20,30,40];
4  score=0;
5  onClick("right",right);
6  while(true){
7      move("neko",x,y);
8      moveBall(0);
9      moveBall(1);
10     moveBall(2);
11     moveBall(3);
12     moveBall(4);
13     wait(50);
14 }
15 function moveBall(i){
16     move("ball"+i,bx[i],by[i]);
17     by[i]+=10;
18     if(by[i]>400){
19         by[i]=0;
20         bx[i]=rnd(400);
21         score+=10;
22         setText("score",score);
23     }
24     if(bx[i]>x-30 && bx[i]<x+30 &&
25     by[i]>y-30 && by[i]<y+30){
26         y=-100;
27     }
28 }
29 function left(){
30     x-=10;
31 }
32 function right(){
33     x+=10;
34 }
```

図 16　教材で完成するゲームのプログラム
**Fig. 16**　Complete game program by teaching material.

は，アルゴリズムの間違いと構文や関数の使い方の間違いが合わせて 3/4 以上になり，Bit Arrow を利用することで，分岐や反復の使い方やアルゴリズムの理解に取り組ませることができたといえる．一方，文法エラーを出さない打ち間違いもあり，これを減らすことで，よりプログラミングの構造の理解に時間を使うことができると考えられる．

### 5.2　アニメーションを用いた作品制作における実践

#### 5.2.1　実践の対象

5.1 節の分析対象とは別の，プログラミング経験のない高校 1 年生を対象に初学者用 JavaScript を用いてゲームを作成させた．ゲームなどの身近な題材を扱う授業では，学習者の意欲を向上させることができると考えられる．この実践では，表 2 に示したライブラリを用いて落ちてくるボールをよけるゲームを作成するチュートリアル教材を用意した．教材どおりにプログラムを作成すると，**図 16** のようになる．はじめはキャラクタを表示させるだけのプログラムから，段階的に反復や分岐，最後は配列の概念を用いてゲームを作成することになる．実践は 1 回あたり 65

表 6　Bit Arrow の使いやすかった点のアンケート結果
**Table 6**　Result of questionnaire “What is the convenient point of Bit Arrow?”.

| 使いやすかった点 | 人数 |
|---|---|
| エラー表示(ポップアップ，!マーク) | 16 |
| 対応する記号の自動入力 | 15 |
| 実行画面をすぐに確認できる | 8 |
| HTML と JavaScript の切り替え | 8 |
| コメントや文字列の色付け | 7 |

分の授業 2 回（チュートリアルと作品制作にそれぞれ 1 回ずつ）を用いて行った．

#### 5.2.2　学習者へのアンケート調査

用意したライブラリがゲーム制作の助けになっているかを確認するため，サンプル教材にならってゲームを制作できたか学習者に聞いた．ゲームを作成した学習者のうち 76 人にアンケート調査を行った．ゲームを教材のサンプルにならって制作できたか，という問いに対して，67 人ができたと回答した．配列を用いる難易度の高いプログラムであったが，多くの学習者が教材どおりにゲーム制作をすることができた．図 16 のプログラムで，通常の JavaScript では難しいゲーム制作のような題材が，move() のようなアニメーション制作を補助するライブラリを用意したことで行えるようになったことが分かる．

次に，実際に初学者がプログラミングを学習するうえで効果的であったサポート機能を調べるために，自由記述で Bit Arrow の使いやすかった点を聞いた．多く得られた回答を**表 6** に示す．最も多かった回答は，エラーの表示がされることであった．実行すると，エラーの有無がすぐに分かり，エラーが発生していればその場所を通知する機能が，学習者の助けになっていたことが分かる．次に多かった回答は対応する記号の自動入力であった．この機能により，エラーを未然に防ぐことができた．実行画面をすぐに確認できること，HTML と JavaScript の切替えができることが次に多かったことから，操作の煩雑さや，HTML と JavaScript の混同といった問題を解決できている．

最後に，提案した支援のほかに初学者が必要としている機能を調査するため，自由記述で Bit Arrow の使いにくかった点のアンケートをとった結果を**表 7** に示す．自由記述であるため，回答の中には Bit Arrow という環境への指摘だけではなく，プログラミング全般に対する感想も見られた．最も多かったのは使いにくい点は特になかったという回答であり，Bit Arrow がプログラミング教育の助けになったと感じている学習者が多かったことが分かる．また，次にプログラミングが難しいことや，打ち込む量が多かったこと，ミスがあると動かないことなどがあげられた．打ち込む量が多いという意見は，初学者用 JavaScript を用いても多くあげられたが，教材として用意したプログラムが長かったことも要因として考えられる．ミスがあると動かない点は，プログラミング全般にいえることであり，これを学習することも必要な経験である．

表 7　Bit Arrow の使いにくかった点のアンケート結果
**Table 7**　Result of questionnaire “What is the inconvenient point of Bit Arrow?”.

| 使いにくかった点 | 人数 |
|---|---|
| 特になし | 21 |
| プログラミングが難しい・打ち込む量が多い | 10 |
| ミスがあると動かない | 9 |
| 画面の切り替え | 8 |
| エラーの場所が分かりにくい | 5 |

## 6.　まとめ

本論文では，プログラミング学習支援環境の設計と評価について述べた．実現した機能は，Web ブラウザでのコードの編集・実行，エラーの通知，入力補助，短い命令文のライブラリ，ログの収集である．これらの機能により，特別な準備を必要としないため，以前までの環境と同様導入が容易であるが，エラーの通知やインデントなどの入力補助機能から，これまでの環境と違い初学者が処理系の出すエラーに対応することを支援できる．本環境の利用者のログから，処理系がエラーを出す割合が少なく，そのエラーの修正時間が短いことが分かった．このことから，エラーの発生を減らし，学習者が自力でエラーを修正できることで，アルゴリズムや分岐・反復といったプログラミングの基本的な概念の学習を支援することができていると考えられる．また，学習者の興味を惹くような DOM を操作するアニメーションを題材として扱うこともでき，本環境の学習支援はプログラミング教育に有用であるといえる．

今後の課題は，実行中に発生したエラーの場所の表示と，打ち間違いなどによりプログラミングの本質でないことが原因で処理系はエラーを出さないが思いどおりに動かない状態の修正の支援があげられる．

### 参考文献


[1] 幼稚園，小学校，中学校，高等学校及び特別支援学校の学習指導要領等の改善及び必要な方策等について（答申）（中教審第 197 号），入手先 ⟨http://www.mext.go.jp/b_menu/shingi/chukyo/chukyo0/toushin/1380731.htm⟩（参照 2017-08-25）．

[2] 水越敏行，村井　純，生田孝至ほか：情報の科学，日本文教出版 (2012).

[3] 赤堀侃司，永野和男，東原義訓ほか：情報の科学，東京書籍 (2012).

[4] Solin, J.: TECHNOLOGY THAT EDUCATORS OF COMPUTING HAIL (TECH): Using Cloud9, a powerful cloud-based IDE in the classroom, *ACM Inroads*, Vol.8, pp.29–30 (2017).

[5] Papancea, A., Spacco, J. and HoveMeyer, D.: An open platform for managing short programming exer-

cises, *ICER '13*, *Proc. 9th Annual International ACM Conference on International Computing Education Research*, pp.47–52 (2013).
[6] Denny, P. et al.: CodeWrite: Supporting student-driven practice of Java, *SIGCSE '11*, *Proc. 42nd ACM Technical Symposium on Computer Science Education*, pp.471–476 (2011).
[7] 西田知博ほか：初学者用プログラミング学習環境 PEN の実装と評価，情報処理学会論文誌，Vol.48, No.8, pp.2736–2747 (2007).
[8] Runstant, available from ⟨http://runstant.com/⟩ (accessed 2017-08-25).
[9] Quinson, M. and Oster, G.: A Teaching System to Learn Programming: The Programmer's Learning Machine, *ITiCSE '15*, *Proc. 2015 ACM Conference on Innovation and Technology in Computer Science Education*, pp.260–265 (2015).
[10] Truong, N., Bancroft, P. and Roe, P.: A web based environment for learning to program, *ACSC '03*, *Proc. 26th Australasian Computer Science Conference*, Vol.16, pp.255–264 (2003).
[11] 松澤芳昭ほか：ビジュアル—Java 相互変換によるシームレスな言語移行を指向したプログラミング学習環境の提案と評価，情報処理学会論文誌，Vol.55, No.1, pp.57–71 (2014).
[12] Cooper, S., Dann, W. and Pausch, R.: Alice: A 3-D tool for introductory programming concepts, *Journal of Computing Sciences in Colleges*, Vol.15, pp.107–116 (2000).
[13] Gomes, A. and Mendes, A.J.: An environment to improve programming education, *CompSysTech '07*, *Proc. 2007 International Conference on Computer Systems and Technologies*, No.88 (2007).
[14] 荻野哲男，藤岡健史，柳瀬大輔：教育現場での実践に向けたプログラミング実行環境「ますめ」の試作，研究報告教育学習支援情報システム（CLE），Vol.2011-CLE-5, No.5, pp.1–6 (2011).
[15] enchant.js, available from ⟨http://enchantjs.com/⟩ (accessed 2017-08-25).
[16] CreateJS, available from ⟨http://createjs.com/⟩ (accessed 2017-08-25).
[17] phina.js, available from ⟨http://phinajs.com/⟩ (accessed 2017-08-25).
[18] Elamir, A.M., Jailani, N. and Bakar, M.A.: Framework and Architecture for Programming Education Environment as a Cloud Computing Service, *Procedia Technology*, Vol.11, pp.1299–1308 (2013).
[19] Efopoulos, V. et al.: WIPE: A programming environment for novices, *ITiCSE '05*, *Proc. 10th Annual SIGCSE Conference on Innovation and Technology in Computer Science Education*, pp.113–117 (2005).

**長島 和平**　（学生会員）

2014 年明星大学情報学部卒業．2016 年同大学大学院情報学研究科修士課程修了．現在，東京農工大学工学府電子情報工学専攻在学中．プログラミング教育に関する研究に従事．日本情報科教育学会会員．

**長 慎也**　（正会員）

2005 年早稲田大学大学院理工学研究科にて博士（情報科学）の学位を取得．2006 年より一橋大学総合情報処理センター助手．2010 年より明星大学情報学部准教授．2017 年同学部教授．プログラミング教育，プログラミング言語の開発に関する研究に従事．2005 年情報処理学会山下記念研究賞受賞．教育情報システム学会，ACM，IEEE 各会員．

**間辺 広樹**　（正会員）

1986 年東京理科大学理工学部数学科卒業．同年から神奈川県立高校にて数学科教諭として勤務．2013 年大阪電気通信大学医療福祉工学研究科博士課程修了．博士（工学）．2013 年から神奈川県立柏陽高等学校にて情報科・数学科教諭として勤務．情報科学教育とオンライン学習教材の研究に従事．2010 年情報処理学会山下記念研究賞受賞．日本情報科教育学会会員．

**兼宗 進**　（正会員）

1987 年千葉大学工学部電子工学科卒業．1989 年筑波大学大学院理工学研究科修士課程修了．2004 年筑波大学大学院ビジネス科学研究科博士課程修了．博士（システムズマネジメント）．企業勤務後，2004 年から一橋大学総合情報処理センター准教授．2009 年から大阪電気通信大学医療福祉工学部/総合情報学部を経て，工学部電子機械工学科教授．プログラミング言語，情報科学教育に興味を持つ．ACM，IEEE Computer Society 各会員．

**並木 美太郎**　（正会員）

1984 年東京農工大学工学部数理情報工学科卒業．1986 年同大学大学院修士課程修了．同年 4 月（株）日立製作所基礎研究所入社．1988 年東京農工大学工学部数理情報工学科助手．1993 年 11 月電子情報工学科助教授．1998 年 4 月情報コミュニケーション工学科助教授．現在，東京農工大学大学院工学研究院教授．博士（工学）．オペレーティングシステム，言語処理系等のシステムソフトウェア，並列処理，コンピュータネットワーク，計算機アーキテクチャ等の研究・開発，計算機科学の教育に従事．ACM，IEEE，電子情報通信学会，ソフトウェア科学会各会員．