Processing Web IDEにおける無限ループの検出と分析

三浦 元喜[1,a)]

受付日 2017年5月25日, 採録日 2017年6月22日

概要：ブラウザのみで利用できる Web IDE は，利用者の環境構築にかかる手間を軽減でき，気軽に利用できるため，初学者に適している．しかし，Web IDE で無限ループを含むプログラムを実行してしまうと，復旧の手間が必要となるため，初学者は編集作業に不安を感じてしまう恐れがある．我々は，Processing を用いた Web IDE において，for 文による無限ループを検出し，初学者に通知する簡易的な仕組みを構築した．1 年間の運用を通じて，学習者の for 文の誤りパターンと修正記録を収集した．その結果，for 文の誤りの種類によって学習者の問題点への発見と解決のしやすさに差があることを確認した．提案手法によって，初学者の入力ミスによるブラウザのフリーズを軽減し，安心して演習できる環境が提供できることを確認した．

キーワード：プログラミング教育環境，Web，Processing, 誤り分析

Detection and Analysis of Infinite Loops on Processing Web IDE

Motoki Miura[1,a)]

Received: May 25, 2017, Accepted: June 22, 2017

Abstract: Web-based programming environment is beneficial for the novice programmers because it only requires a web browser to work. Thus the environment facilitates continuous and repetitive learning activities such as exercises and reports. However, some Web-based programming environment could not prevent execution of codes that contains infinite loop. Thus the novice programmer often frozen the web browser, and had to recover by restarting the browser. To prevent the execution of infinite loop, we have introduced a detector especially for the 'for' loop. The detector tells the students where the error exists. By adopting the detector for lectures for one year and analyzing the logs, we confirmed there are various errors, and the some type of error is difficult to solve by the novice programmer. The detector can reveal the frequency of browser freezing, and reduce the burdens and anxiety of novice programmers.

Keywords: Programming Environment, Web, Processing, Analysis of errors

1. はじめに

これまで，プログラミング初学者に対する様々な教授法や，プログラミング支援環境，適したプログラミング言語やツールなどの研究・開発がすすめられてきた [1]．とくに近年では，学習者が手軽に学習や演習に取り組めることを目的として，Web ブラウザのみあれば動作させることができるプログラミング言語や環境が，盛んに開発されている [2], [3], [4], [5]．近年では，HTML5 によるインタラクティブ性の高いプログラミング環境や，コンパイラについても，Javascript 言語で記述されることが多い．そのため，Web ブラウザが動作するクライアント環境でソースコードをコンパイルし，そのまま実行できる言語や環境が一般的となっている．

長島らは，Web ブラウザだけで学習が行えるプログラミング環境 Bit Arrow[5] を開発している．Bit Arrow は，Javascript プログラミングを行う初学者を支援するために，

[1] 九州工業大学　基礎科学研究系
1-1 Sensui, Tobata, Kitakyushu, Fukuoka, 804–8550, Japan
[a)] miuramo@mns.kyutech.ac.jp

エラーダイアログを画面上に表示し，エラーメッセージを伝えるとともに，ソースコードの誤り箇所をマークで強調する．また，自動インデント機能や，対応する括弧を自動的に挿入する機能を備えている．さらに，Bit Arrow 版の Javascript では，wait 命令や簡易記法によって，学習者の理解を促進する仕組みを備えている．また近年では，Cloud9[*1]のように，コンソールを含めた計算機環境やサーバ環境全体を，ブラウザを介して利用できるサービスもある．

我々は，C 言語に類似した文法を備えつつ，グラフィックスやアニメーションを簡潔に記述することができる Processing 言語に着目し，その Javascript 版の実装である Processing.js[*2] を利用して，大学生を対象とした講義を行ってきた [6]．Web ブラウザのみで完結するため，プログラミングの初学者にとっても敷居が低く，また復習や演習がしやすいといったメリットがある．

しかし，Web ブラウザ上で動作するプログラミング環境のデメリットとして，無限ループを含むソースコードを作成し，Javascript として実行してしまうと，ページを表示しているブラウザのタブ，またはブラウザ全体が固まって (フリーズして) しまうことが挙げられる．一旦ブラウザが固まってしまうと，タブを閉じたり，タスクマネージャを起動してプロセスを停止させるなど，回復のための作業手順が必要となる．この回復のための作業手順を仮に覚えたり慣れたりしたとしても，ちょっとした編集ミスや入力操作ミスでブラウザが固まってしまうような学習環境では，初学者は安心して演習に取り組むことができないと考えられる．

そこで，Web ブラウザ上で動作するプログラミング環境の利便性を保ちつつ，初学者がブラウザのフリーズを心配せず，安心して演習に取り組めるようにするため，無限ループを発生させる可能性があるソースコードを実行前にチェックする仕組みを導入した．また，無限ループを発生させる可能性がある箇所を初学者に通知し，エラーやミスからの円滑な回復を支援する仕組みも導入した．

2. フリーズ防止と，フリーズ原因の検出

Web ブラウザのフリーズを単純に防止するだけであれば，Concurrent.Thread.js [*3] や，mtrepetition.js[*4] を導入すればよい．しかし，初学者が誤りを含むソースコードを修正しやすくするため，コード中のどの部分にどのような誤りや問題があるかを提示することも重要になってくる．そこで我々は，フリーズの防止とあわせて，フリーズの原因やループ記述の誤りを初学者に伝える仕組みを導入する．

[*1] http://c9.io/
[*2] http://processingjs.org/
[*3] https://www.infoq.com/jp/articles/js_multithread
[*4] http://aoihappa.seesaa.net/article/242400582.html

今回の研究におけるループ記述誤り分析の対象としては，while 文を除外し，for 文のみに着目した．for 文は，Processing プログラミングにおけるループ記述に一般的に使われる構文であり，初期化部・継続条件部・更新部がセットで記述されているため，誤りの検出と類別化がしやすい．また頻繁に用いられる構文であるため，フリーズ検知機構導入における優先順位が高いと考えた．

我々は，以下の手順でチェックを行う Javascript の関数を作成した．

(1) 引数で与えられたソースコードに含まれる，コメントをすべて削除する
(2) ソースコード中の改行を，すべて削除する
(3) '}' 文字を，'}'+'改行' におきかえる
(4) for 構文 (`for ( ... )`{) にマッチする文字列について，以下をチェックする
(4a) 括弧 () のなかの文字列に，',' (カンマ) が含まれている？ (ただし関数呼び出し部を除く)
(4b) 括弧 () のなかの文字列に，';' (セミコロン) が 2 つあるか？
(以降のための準備) 括弧 () のなかの文字列を，xxx ; yyy ; zzz (初期化部，継続条件部，更新部) に分ける
(4c) 'xxx' の部分が，`int k = 0` のように，「型」「変数名」「=」「初期値」のみで構成されている？ (「型」は任意．また「初期値」が変数の場合もありうる．その場合は後述 (4h) する「初期値と値の比較」は行わない)
(4d) 'yyy' の部分が，`k < 10` のように，「変数名」「比較演算子」「値」を備えている？ (ただし，`k < ary.length` のように，「値」が変数や配列を参照しているケースについては，チェックしない．また，後述 (4h) する「初期値と値の比較」も行わない)
(4e) 'zzz' の部分が，「変数名」と「増分演算子または減分演算子」または，変数を増加/減少する文になっているか？
(4f) (4d) における，比較演算子の種類は，(4e) における増加 (または減少) と整合している？ (ここでは「初期値」と「値」の比較は行わず，比較の向きと，増加/減少のみをチェックする)
(4g) (4c) における「変数名」は，(4d) や (4e) に出現する「変数名」と一致している？
(4h) (4c) における「初期値」は，(4d) における「値」や (4f) における「増加/減少」の種類と，整合している？ (たとえば，`int k = 10; k < 10; k++` のように，無限ループにはならない場合でも，誤りとして扱う)
(5) (4a)〜(4h) のすべてを満たさなかった場合は，「誤りなし」とみなして，プログラムの実行および，サーバへの保存を行う．

(4a) のカンマの有無のチェックは，セミコロンのかわり

表 1 誤りチェックによる分類結果
Table 1 Result of error check

Type	代表的な誤り	誤りの理由	総数	初回	2 回目以降	2 回目以降/初回
4a	`( int k=0; k < 10 , k++ )`	カンマ区切り	25	21	4	19.0%
4b	`( int h=-7h<7;h++)`	セミコロン不足	132	105	27	25.7%
4c	`( int j ; j < 10 ; j++ )`	初期化部の誤り	0	0	0	
4d	`( int k=0 ; k =< 10 ; k++ )`	継続条件の誤り	1	1	0	0%
4e	`( int k=0 ; k < 10 ; k+ )`	更新部の誤り	27	15	12	80.0%
4f	`( int k=0 ; k < 10 ; k-=2 )`	増減方向の矛盾	58	43	14	32.6%
4g	`( int v=0 ; v < 10 ; V++ )`	変数の大小文字誤り	113	63	50	79.4%
	`( int i=0 ; i < 5 ; j++ )`	変数名の不一致	67	39	28	71.8%
4h	`( int k=10 ; k < 10 ; k++ )`	初期値と終了値の矛盾	89	57	32	56.1%

に，カンマを使って区切ってしまう誤りを検出するため，(4b) のセミコロン不足よりも先にチェックを行っている．(4e) の更新部の誤りは，タイプミスによって正しく変数を更新していない場合をチェックする．一部の文法誤りについては，コンパイラによるチェックによって防ぐこともできるが，初学者に for 文の記述ミスであることを明確に通知できるようにするため，コンパイル前にチェックを行っている．(4f) は，ある程度 for 文を理解して，慣れた学習者でもケアレスミスで書いてしまう可能性が高い誤りである．(4h) は，厳密にはフリーズを引き起こさない場合もある．例にあげたケースであれば，初期値が，終了値と同じであるため，ブロックは 1 回も実行されない．ただ，論理的には正しくないプログラムであるため，このような記述も初学者に通知する必要があると考え，チェック項目に加えている．

3. 運用と分析

我々は，誤り検出機構を組み入れた Web 演習システムを構築し，2016 年 6 月 20 日から，およそ 1 年間のあいだ，実際の講義で利用してきた．Web 演習システムでは，ブラウザ上のエディタで入力した Processing ソースコードをコンパイル・実行し，ブラウザ内で動作確認を行うことができる．Web 演習システムにおいては，学習者がプログラムの実行ボタンを押すか，またはサーバへの保存操作を行ったときに，プログラムのコンパイルと実行を行う．そのタイミングにおいて，プログラムのコンパイル前に，前述のチェックを行い，誤りがある場合は**図 1** に示すように，モーダルダイアログによって for 文の記述誤りを通知している．**図 2** は，対応する括弧の挿入忘れを通知している例である．

なお，今回報告する分の運用においては，図 1 や図 2 で示したように，記述誤りの具体的な理由や解決法を含めずに，学習者に提示している．その理由は，記述誤りの種類による学習者自身による問題点への気付きやすさや，修正のしやすさを分析するうえで，詳細なメッセージによる影響を排除したかったからである．通知メッセージの具体化が，解決プロセスに与える効果については，今後検証していく予定である．

この Web 演習システムを，大学 2 年生を対象とした 6 つの講義および，大学院生を対象とした 1 つの講義で運用した．講義の履修者数や利用頻度，対象となる課題の数などは，講義によって異なるが，いずれも Web 演習システムで実際にプログラムを編集して回答するタイプの演習課題や試験である．**表 2** に，講義の受講者総数と，Processing 課題の総数を示す．また学習者の誤りを分析するため，誤り通知機構によって誤りが発見されたソースコードについては，チェック時刻とともにサーバに自動的に記録した．

3.1 分析と結果

2016 年 6 月 20 日から 2017 年 5 月 25 日までに，サーバに記録された誤りログは，7 つの講義の合計で，511 個であった．また，誤りログにおいて，重複を含めないでカウントした学生の数は，107 名であった．実際に検出された記述誤りの一部を，付録 A.1 の**図 A·1** から**図 A·7** に示している．

実際に検出された記述誤りの数について，種類別に示したものを**図 1** に示す．「初回」は，ある学習者が最初に入力した記述誤りの数を示し，「2 回目以降」は，その初回の誤りの直後に，学習者がソースコードをまったく修正せずに再度実行しようとして，サーバに記録された記述誤りの数を示している．また，「2 回目以降/初回」は，記述誤りを入力した回数に対する，直後の修正なし再実行の割合を示す．この数字が大きいほど，学習者にとっては自己修正がうまく行えなかったり，気づきにくかったりした誤りであったといえる．

この割合から導かれる，気づきにくい記述誤りの上位 3 つは「更新部の誤り (80.0%)」「変数の大小文字誤り (79.4%)」「変数名の不一致 (71.8%)」であった．更新部の誤りは，変数を増減させるための記法に慣れていないことや，知識不足から生じていると考えらえる．変数の大小文字誤りは，知識不足に起因する単純な写し間違いや打ち間違いに加えて，不等号記号や等号記号，プラス記号などのシフトキー

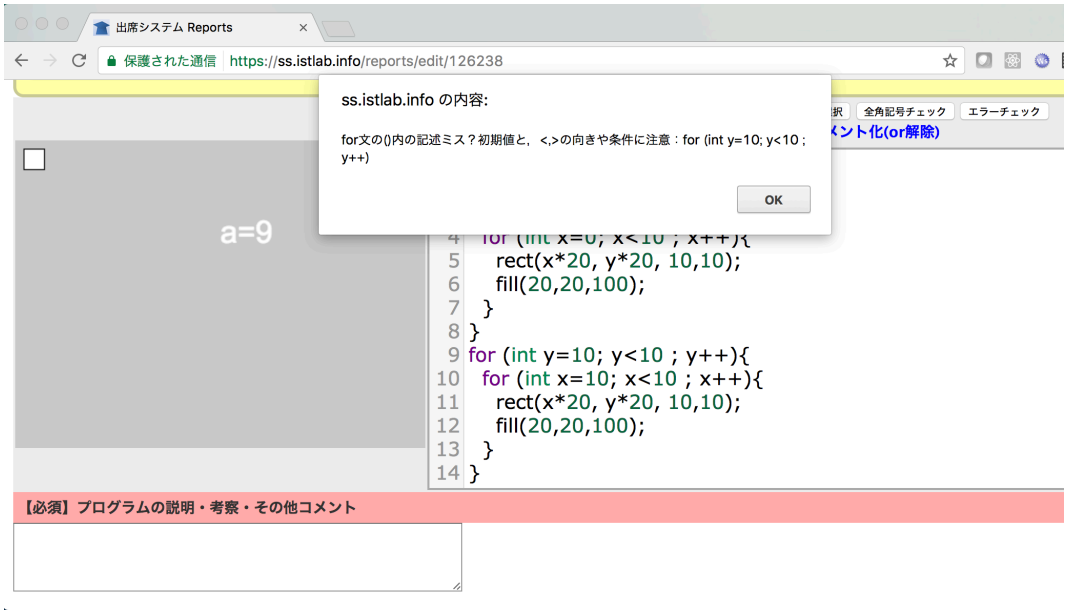

図 1　for 文の記述誤り通知

図 2　括弧の対応忘れ

押下を必要とする文字と近接していることから生じる入力ミスの可能性も考えられる．変数名の不一致は，ソースコードを修正している際に，直すべき箇所を忘れてしまったり，他のソースコードから写しているときに入力を誤ることなどが考えられる．いずれの誤りについても，システムのメッセージによる指摘から，自力で問題点を発見し，修正することが比較的難しい部類であることがわかった．また，総数および初回の数が多い，「セミコロン不足」については，for 文の構文に対する知識不足や，入力ミスが影響している可能性がある．

とくに試験や，短時間での小テストのように，回答時間が限られており，受講者が急いでタイピングする場面において，フリーズ防止機構を組み込む以前は，(1) ブラウザプロセスの停止，(2) 保存された無限ループを含むソースコードをコメント化，(3) コメントを解除 という回復手順をふむ必要があった．そのため，回答時間が限られているということもあり，受講者の不利益となってしまっていた．今回のフリーズ防止機構導入により，試験における回復手順の回数は格段に減少した．また受講者にとっても，フリーズに怯えることなく，安心してプログラムを入力できるようになった．

Processing.js は，Javascript でコンパイルを行うため，プログラムをサーバに送ったり，結果を受信したりする必要がない．今回のフリーズ防止機構も同様に，Javascript

によって短時間でチェックが終わるため，学習者の心理的負担は少ない．

4. 関連研究

McIver らは，学生の文法エラーを最小にするため，GRAIL[7] という初学者教育に特化したプログラミング言語を開発し，初学者教育に導入した [8]．LOGO 言語との比較を行い，GRAIL のほうがエラーを軽減することを確認した．横田らは，大学 1 年生を対象とした概念モデリング教育における，モデル駆動開発方法論 (MDD) 導入前後のモデル品質の差と，学習者の誤りパタンの差について考察している [9]．UML のモデル図編集ツールを導入することで，モデル図の品質は向上し，学習者自身が誤りを修正できるという結果が得られた．このように，言語そのものを改良したり，学習内容を初学者向けに修正していくアプローチは，文法ミスを軽減したり，学習時のしきいを下げる点で有効である．本研究では，C 言語に近い構文を用いた Processing に対して，学習者のエラー傾向を分析した．

Rodrigo らは，初級プログラミング講義における，学生のプログラム編集履歴やコンパイルエラー履歴から，学生の欲求不満 (フラストレーション) の検出を行っている [10]．Lee らは，初学者がプログラミングのエラーを修正する際の混乱や不安の度合いと，成績との相関を分析している [11]．これらの研究から，プログラミングエラーは学習者の感情や成績に影響を及ぼすことが確認できる．

エラーの提示方法や，エラーメッセージのわかりやすさを向上させるアプローチとして，Kelleher らは，1960 年代からのプログラミング言語と環境について，分類を行っている [12]．また，Marcheau らは，初学者がエラーメッセージをどう捉えているかについての系統的な調査を通じて，初学者向けの IDE が備えるべきインタフェースデザインや，プログラミング言語のあり方について考察を行っている [13]．HMMMML [14] は，学生のプログラミング学習に対するモチベーション維持および向上を目的として開発された言語および実行環境である．単純かつ直感的な文法を導入しており，セミコロン抜けや宣言忘れなどのミスも許容される．ループは`<for 3> ... </for>` のように HTML のタグに似た命令を用いる．音声合成による文字列の読み上げや，MIDI 音の出力，インターネットを利用した検索や翻訳など，学習者の意欲を高めるための命令も組み込まれている．HMMMML2 [15] は，HMMMML における「好意的な解釈によって，とりあえず実行する」スタイルを踏襲しつつ，コンパイラがどこを修正したのかを，ソースコードへのコメント挿入によって，プログラマに伝える仕組みを導入している．またスペルミスを許容したり，複数の解釈が可能なときはすべて実行したうえで，あとでプログラマに選択させるといった柔軟なインタフェースを提供している．先進的で突出した発想であり，高速プロトタイピング用途としても有効性が高いが，学習効果を高めるという観点からは，適切なサポートレベルを実験によって検証していく必要があると考えている．

5. 議論

プログラムの停止性の判別は原理的に不可能であることが証明されている [16]．そのため，プログラムの停止性を判別したり，保証したりことはできない．本研究で示した検出方法は，初学者の誤りに対応することを想定した，簡易的でアドホックなものである．定型的な記述を強制しているため，初期化部で変数を記入しないケースや，継続条件で `&&` や `||` を用いた複合条件を許容できていない．また，継続条件や更新部分に関数を用いた場合の停止性はチェックしていない．加えて，ログ取得では，本手法で検出できたエラーのみを収集しているため，false negative(本当は検出すべきだがスルーした) を取得できていない．また，再帰呼び出しなど，for ループによらない記述によるプログラムの不具合についても，検出できない．本手法で検出できなかったエラーがどの程度あったかについては，別の活動ログを参照したり，また実際にブラウザで発生したフリーズを検知する仕組みを導入するなどの工夫が必要である．

6. おわりに

Web IDE は，ブラウザのみで演習環境が利用できるため，初学者が気軽に演習に取り組めるという点で優れている．Processing 言語を対象とした Web IDE の利便性を高めるため，for 文の基礎的な記述誤りを検出し，ブラウザのフリーズを軽減する機構を導入した．講義における 1 年間の運用において，計 511 回のフリーズを防止できたことを確認した．また，記述誤りの種類によって，学習者による問題点の発見や，修正のしやすさが異なる可能性について検討した．

提案手法は非常に簡易的な検出機構しか用いていないが，それでも初学者のプログラム入力・編集ミスに対する心理的な不安の軽減に寄与できたと考えられる．今後は，学習者に提示するメッセージが，学習者の問題点発見・修正行動にどのような効果をもたらすかを検証していく．

謝辞　本研究の一部は JSPS 科研費 (課題番号 15K00485) および公益財団法人電気通信普及財団の支援によるものです．

参考文献

[1] Arnold Pears, Stephen Seidman, Lauri Malmi, Linda Mannila, Elizabeth Adams, Jens Bennedsen, Marie Devlin, and James Paterson. A survey of literature on the teaching of introductory programming. *ACM SIGCSE Bulletin*, Vol. 39, No. 4, pp. 204–223, 2007.

[2] Max Goldman, Greg Little, and Robert C Miller. Real-

表 2　講義別の受講者総数と，誤りログにおけるユニーク受講者数

Table 2　Stat of students on lectures

講義名	受講者総数	Processing 課題の総数	誤り記録の受講者数	誤り記録の数
情報処理基礎 2016	69	20	39	177
実践プログラミング PBL2016	53	6	10	58
情報の表現法 (A)2016	43	5	11	50
情報の表現法 (B)2016	57	5	19	90
インタラクティブシステム 2016	5	1	1	1
情報処理基礎 2017	53	5	24	125
実践プログラミング PBL2017	51	1	3	10

time collaborative coding in a web IDE. In *Proceedings of the 24th annual ACM symposium on User interface software and technology*, pp. 155–164. ACM, 2011.

[3] Vu Nguyen, Hai H Dang, Kha N Do, and Thu D Tran. Learning and Practicing Object-Oriented Programming Using a Collaborative Web-based IDE. In *Frontiers in Education Conference (FIE), 2014 IEEE*, pp. 1–9. IEEE, 2014.

[4] 和平長島, 慎也長. Tonyu system 2 ゲーム制作を通じたプログラミング学習に適したフレームワーク. 情報処理学会研究報告. コンピュータと教育研究会報告, Vol. 2015, No. 2, pp. 1–8, mar 2015.

[5] 長島和平, 長慎也, 間辺広樹, 兼宗進, 並木美太郎. 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. 研究報告コンピュータと教育 (CE), Vol. 2017, No. 2, pp. 1–8, February 2017.

[6] 三浦元喜. Processing Web IDE を用いたプログラミング基礎教育の試み. 情報処理学会情報教育シンポジウム (SSS2013) 予稿集, pp. 225–231. 情報処理学会, August 2013.

[7] Linda McIver, M Linda, and Damian Conway. GRAIL: A Zeroth Programming Language. 1999.

[8] Linda McIver. The effect of programming language on error rates of novice programmers. In *12th Annual Workshop of the Psychology of Programming Interest Group*, pp. 181–192. Citeseer, 2000.

[9] 横田寛明, 香山瑞恵, 小形真平, 橋本昌巳, 大谷真. ロボット動作設計を対象にした状態遷移図による概念モデリング教育へのモデル駆動開発方法論導入の効果. 研究報告コンピュータと教育 (CE), Vol. 2014, No. 9, pp. 1–6, 2014.

[10] Ma Mercedes T Rodrigo and Ryan SJd Baker. Coarse-grained detection of student frustration in an introductory programming course. In *Proceedings of the fifth international workshop on Computing education research workshop*, pp. 75–80. ACM, 2009.

[11] Diane Lee, Ma Rodrigo, Ryan Baker, Jessica Sugay, and Andrei Coronel. Exploring the relationship between novice programmer confusion and achievement. *Affective computing and intelligent interaction*, pp. 175–184, 2011.

[12] Caitlin Kelleher and Randy Pausch. Lowering the Barriers to Programming: A Taxonomy of Programming Environments and Languages for Novice Programmers. *ACM Computing Surveys (CSUR)*, Vol. 37, No. 2, pp. 83–137, 2005.

[13] Guillaume Marceau, Kathi Fisler, and Shriram Krishnamurthi. Mind your language: on novices' interactions with error messages. In *Proceedings of the 10th SIGPLAN symposium on New ideas, new paradigms, and reflections on programming and software*, pp. 3–18. ACM, 2011.

[14] 宮下芳明. プログラミングに対するモチベーションを向上させる新言語 HMMMML の開発. 第 51 回プログラミング・シンポジウム予稿集, pp. 57–64. 情報処理学会, 2010.

[15] 中橋雅弘, 宮下芳明. HMMMML2：モチベーション向上の為のコンパイラ. 夏のプログラミング・シンポジウム報告集, pp. 107–110. 情報処理学会, 2011.

[16] Alan Mathison Turing. On computable numbers, with an application to the entscheidungsproblem. *Proceedings of the London mathematical society*, Vol. 2, No. 1, pp. 230–265, 1937.

付　　録

A.1　実際に記録された記述誤り

```
void setup(){
  size(700,300);
  fill(0);
  int x = 10;
  for(int h=1 ; h<=19 ; h++){
    rect(x, 20,30,30);
    x += 35;
  }
  for(int h=1 ; h<=19 ; h+40){
    rect(h, 55,30,30);
    x += 35
  }



}
```

図 A·1　更新部：h+40

```
void setup(){
  size(400,300);

  for(int k=0;k=10;k++){
    rect(rect(k *10, k * 10, 8, 8);
  }
}
```

図 A·2　終了条件

```
void setup(){
  size(290,290);
  textSize(20); //文字のサイズ
  textAlign(CENTER); //文字の中央揃え

  // 以下のサンプルを消して，for文で書いてください．
  // for文を使ったほうが，最終的な位置の微調整がしやすいです．
  int k, v;
  for(k = 1; i<9; i++){
    for(v = 1; v<9; v++){
  fill(255);//塗り色を白に設定
  rect(k*30,v*30,30,30);

  fill(0);//文字の色を黒に設定
  text(k*v, k*30+15, v*30+24);
    }
}
```

図 A·3　変数名の不一致

```
void setup(){
  size(800,200);
  int x = 10;
  for(int k=1 ; k<=3 ; h++){
    rect(x,15,50,20);
    x +=100;
  }
  for(int h=1 ; h<=3 ; h++){
    rect(x,35,70,30);
    x += 100;
  }
  for(int h=1 ; h<7 ; h++){
    ellipse(h*50-30,80,30,30);
  }
}
```

図 A·4　変数名の不一致 (2)

```
void setup(){
  size(400,300);
  int x = 10;

  for(int k=0;K<0;k++){
  rect( 10,20,30,30);
    text(k,x,63)
  x = x+40
  }
}
```

図 A·5　大文字小文字 (k/K)

```
void setup(){
  size(400,300);
  int x = 0;

// HSBで．最大値を２５５から１０に変更
  colorMode(HSB,10);

  for(int v=1 ; v<=9 ; V++){
    for(int h=1 ; h<=9 ; h++){
     fill(v,h,10);//色彩.彩度.明度
     rect(h*40-30,v*30-20,30-20);
    }
  }
}
```

図 A·6　大文字小文字 (v/V)

```
float x;
for(x=0;x<100;x=++){
size(300,200);
background(250);
fill(50,170,50);
ellipse(width/2+x,height/2,100,100)};


textSize(30);
fill(100,40,255); //textの色をかえるときもfill()関数
text("Suppon",20,40);
```

図 A·7　更新部：=++