社団法人　電子情報通信学会
THE INSTITUTE OF ELECTRONICS,
INFORMATION AND COMMUNICATION ENGINEERS

信学技報
TECHNICAL REPORT OF IEICE.

日本とタイにおけるプログラミング初学者のプログラミング行動の比較

槇原絵里奈† 藤原 賢二† Putchong Uthayopas†† Chantana Chantrapornchai††
Jittat Fakcharoenphol††
井垣 宏††† 吉田 則裕†††† 飯田 元†

† 奈良先端科学技術大学院大学 情報科学研究科〒 630–0192 奈良県生駒市高山町 8916–5
†† Kasetsart University 50 Ngam Wong Wan Rd., Ladyao, Chatuchak, Bangkok 10900
††† 大阪大学 大学院情報科学研究科〒 568–0871 大阪府吹田市山田丘 1–5
†††† 名古屋大学 大学院情報科学研究科 〒 464-8601 名古屋市千種区不老町

あらまし 教育工学における研究の一環として，プログラミング演習において初学者がとる行動の分析が行われている．このような分析は，研究者が所属する大学において実施されることが多い．そのため，大学を跨いで初学者の行動を比較した研究は少ない．我々は，日本とタイの大学間において初学者がとる行動の差異について比較を行うことを計画している．本稿では，これら大学間における教育活動の比較を行い，その後初学者が行う探索的プログラミングについて説明する．そして，日本・タイの大学それぞれで初学者がとる行動を比較する方法について述べ，最後に現在得られている結果について報告と考察を行う．

キーワード プログラミング演習，探索的プログラミング，コーディング過程記録，国際比較，教育支援

A Comparison of the Programming Behavior by Novices Between the Japanese and Thai Universities

Erina MAKIHARA†, Kenji FUJIWARA†, Putchong UTHAYOPAS††, Chantana CHANTRAPORNCHAI††, Jittat FAKCHAROENPHOL††, Hiroshi IGAKI†††, Norihiro YOSHIDA††††, and Hajimu IIDA†

† Graduate School of Information Science, Nara Institute of Science and Technology
Takayama-cho 8916–5, Ikoma-shi, Nara, 630–0192 Japan
†† Department of Computer Engineering, Faculty of Engineering, Kasetsart University
50 Ngam Wong Wan Rd., Ladyao, Chatuchak, Bangkok 10900, Thailand
††† Graduate School of Information Science and Technology, Osaka University
Yamadaoka 1–5, Suita-shi, Osaka, 568–0871 Japan
†††† Graduate School of Information Science, Nagoya University
Furo-cho, Chikusa-ku, Nagoya, 464-8601, Japan

Abstract In the educational technology field, several studies have been done on the analysis of programming behaviors of novices. Typically, this analysis is performed in the university that researchers belong to. Therefore, only a few studies have been done on the comparison of programming behaviors of novices among different universities. We plan to do this comparison between the Japanese and Thai universities. In this paper, we compare the education activities between them, and explain exploratory programming by novices. Then, we propose an approach to comparing programming behaviors of novices in each university. Finally, we describe and discuss the current result of our study.

Key words Programming exercise, Exploratory programming, Records of programming behavior, Cross-national comparison, Education support

1. はじめに

近年，高度なソフトウェア技術者育成を目的として，プログラミングに関する教育が世界中の数多くの教育機関で実施されるようになっている [1], [2].

通常，プログラミング教育においては，特定のプログラミング言語の文法等に関する座学だけでなく，実際にプログラムを作成する演習を伴う．一方で，座学の実施タイミング，対象となるプログラミング言語，演習課題，演習実施形態，あるいはカリキュラム構成といった様々な要素については教育機関ごとに異なっている．そのため，授業スタイルの違いが学生のプログラミングスキルやプログラミング行動にどのような影響を与えるのかは分かっていないのが現状である．

そこで本研究では，日本の大学とタイの大学を例にプログラミング教育がどのように行われているのかを比較し，それぞれの学生がプログラミング課題にどのように取り組むかを調査する．

2. 準　　備

2.1 プログラミング演習

高等教育機関で行われるプログラミングに関する教育の多くが，座学による解説と，学生が実際に特定のプログラミング言語を用いてプログラミングを行う実践的な授業から構成されている．後者のような授業形態のことをプログラミング演習 [1] と呼ぶ．

2.2 プログラミング演習の比較

本節では日本とタイの情報教育を例に以下の観点から比較を行う．

観点 1：カリキュラム　日本とタイの情報系の学部・学科を有する高等教育機関において，プログラミング演習がどのような計画で開講されているかについて比較を行う．

観点 2：プログラミング演習の授業形態　日本とタイにおいて，プログラミング演習が学期を通してどのようなスケジュールで行われているか，また演習の進め方の違いについて比較を行う．以降では本稿で分析対象とする日本の東京工科大学の事例とタイのカセサート大学の事例にもとづき，プログラミング演習及び学生のプログラミング行動の比較を行う．

2.2.1 カリキュラムの違い

プログラミング演習では，普及率が高く構造化プログラミングに適している点から C 言語や Java 言語が扱われることが多い．演習は教員による解説を経て学生による実際のプログラミングが行われるため，解説とプログラミングで授業を分ける場合や，連続して 2 時限演習を行う場合もある．また，演習そのものをプログラミング演習 1，プログラミング演習 2 のように 2 つに分ける場合も存在する [3], [4]．この場合，学生はプログラミング演習 1 でそのプログラミング言語の基礎的な文法やアルゴリズムを学び，プログラミング演習 2 ではそのプログラミング言語独自の特徴にあたる複雑な機能を扱う事が多い．今回分析を行った東京工科大学 [5] では，この形式を採用しており，学部 1 年生前期で Java 言語の基礎的な部分を含むプログラミング演習 1 を，後期では継承等の概念を含む発展的な内容を伴うプログラミング演習 2 を実施している．

今回比較に用いたカセサート大学の Department of Computer Engineering では，学部 1 年時に C#のプログラミング演習が必修になっている．この C#のプログラミング演習は合計で 3 単位になるが，2 単位分の座学（Lecture）と 1 単位分の演習（Laboratory）で構成されており，それぞれが同一週の別の日に実施される．学部 2 年時以降は，プログラミング演習は選択科目になる．しかし，特定の言語のプログラミング演習が行われるのではなく，例えばアルゴリズムの授業を選択した場合はアルゴリズムの学習のために C 言語が扱われ，オブジェクト指向の授業を選択した場合はオブジェクト指向について学ぶため Java や Python を学ぶ．

両国共に学部 1 年時に最初のプログラミング演習が実施される．これは，なるべく早い段階からプログラミング演習を行うことで，グループ開発演習や PBL 演習といった，より実践的な教育を後に実施するためであると考えられる．

2.2.2 プログラミング演習の授業形態

プログラミング演習では，取扱うプログラミング言語を用いて，プログラムを作成するために必要な多くの要素を，条件分岐や繰り返しなど機能のまとまり毎に分割し段階的に演習を進めていく．これは日本の大学でもタイの大学でも同様である．

今回分析した東京工科大学だけでなく，NAIST [3] や大工大 [4] といった複数の大学においても同様に，教員は教科書やスライドなど補助資料を用いて学生にその日取り扱う単元について解説を行い，学生は教員による解説を受けた後，与えられた課題を解き進める．課題は解説に含まれる例題を少し改変した簡単なものから，これまでに学習してきた要素を組み合わせなくてはいけないような課題も提示される．これらの課題は最初の数問は教員による解説や解答例が提示されることもあるが，基本的には周りに相談せず独力で解くことが求められている．また，課題には提出期限が定められており，学生は決められた期限までに特定のフォルダや課題のソースコードを圧縮して教員に提出する．課題締切後，教員は提出されたソースコードを目視または採点プログラムなどを用いて評価する．プログラミング演習の評価方法は各授業ごとの出席や課題の提出状況だけでなく，定期的に行われる小テストや中間，期末テストなどでも評価される．

また，日本のサイバー大学ではオンラインでプログラミング演習を行っている [6]．この場合，プログラミング演習の成績は，最終課題にあたるプログラムと一定以上の単元が終了した際に行うレポートや小テストで判定される．さらに，演習中にわからない問題があった場合，学生は Wink(注1) というスクリーンキャストツールを使う．このツールはコンピュータの画面を録画し，動画ファイルを作成する．学生は，そのファイルを教員に送ることで難解な箇所の提示を行うことができ，また教員もプログラム作成の考え方について学生に提示することができる．

一方，タイのカセサート大学ではプログラミング演習中に解

(注1)：http://www.debugmode.com/wink/

く課題と，プログラミング演習が終わってから次回の授業までの間に提出する課題（宿題）は別である．宿題はプログラミング演習の 1 回の授業が終わった後の復習として出されるものであり，プログラミング演習中に取り扱った課題よりも難易度が高いものが渡される．また，プログラミング演習中に分からない問題があった際は教員やティーチングアシスタント（TA）に聞くだけでなく，周りと相談することも推奨されている．さらに，受講生の多さからプログラミング演習の課題に対する評価はシステムによる自動採点のみであり，教員が学生の書いたソースコードを見ることはない．この採点システムは学生がソースコードを送信するとシステム中でコンパイルを行い，予め設定してある 5 つのテストケースを実行する．テストケースの結果は即座に帰ってくるため，学生はその結果を踏まえて全てのテストケースがエラーなく実行されるまで繰り返しソースコードを修正，送信する．

また，日本のプログラミング演習における初学者の特徴として，エラーメッセージを読まないことが指摘されており，コンパイルエラーメッセージを学生にわかりやすく伝えるための支援研究も行われている [7]．これはコンパイルエラーや実行時エラーのメッセージが英語で出力されることにも一因があると考えられる．一方，今回比較に用いたタイの大学ではプログラミング演習における座学の講義資料は英語で書かれており，授業も英語で進められている．そのため，日本の学生よりタイの学生の方が英語に対する抵抗が小さいと推測される．

2.3 プログラミング行動

本研究では，2.2 章で述べた日本とタイの大学間の差異が，実際のプログラミング演習における学生のプログラミング行動にどのように影響しているか調査を行う．

本稿では，日本とタイにおけるプログラミング行動の違いを調査することを目的として，探索的プログラミング [8] に着目する．

2.3.1 探索的プログラミング

実装するプログラムの要求や要求の実現方法が曖昧であるときに，開発者がプログラムの一部分を変更し，その実行結果を確認するというステップを繰り返すことで，プログラムを完成に近づけていくことができる．このようなプログラミング手法のことを探索的プログラミング [9] と呼ぶ．

探索的プログラミングの一例として，手戻りが発生する場合を図 1 に示す [10]．探索的プログラミング手法では，開発者はまずプロトタイプにあたるプログラムを作成する．ここで言うプロトタイプとは，最終成果物にあたるプログラムのうち，開発者が要求や要求の実装方法を理解することが出来た一部分のみをコーディングしたプログラムである．プロトタイプはコンパイルエラーや実行時エラーを引き起こすことがないように作られていることが望ましい．その後エラーが生じたとしても，エラーの原因はプロトタイプに行った変更であることが自明であるため，すぐに特定して修正を行うことができる．

開発者が探索的プログラミングを行う手順について説明する．開発者はプロトタイプにあたるプログラムを作成し，コンパイルエラーや実行時エラーを引き起こさないことを確認する．次に，開発者は要求や実現方法が曖昧であった一部のプログラムのみを変更し，コンパイル，実行を行う．この時エラーが生じなければ，開発者はまた次の要求や実現方法が曖昧である一部分を変更，コンパイル，実行を行う．仮にエラーが生じたとしても，エラーの原因となる箇所は先ほどプロトタイプから変更した一部分であると推測できるため，開発者は扱いに慣れていない言語や API を使う際でも，デバッグを容易に行うことができる．

一方で実際のソフトウェア開発においては，バグが残ったままのプロトタイプを対象として探索的プログラミングを実施してしまうことも多く，支援ツールや環境の開発が必要であるとされている [11]．既存研究として Sandberg ら [8] は Smalltalk が探索的プログラミングに適しているとし，Smalltalk の開発者らにインタビューを行うことで探索的プログラミングの有効性を立証した．Yoon ら [10] は探索的プログラミングの中で行われる手戻りに注目し，どのようなときに手戻りが発生しているか分析を行った．また，彼らは開発者の手戻りを手助けする eclipse のプラグインである AZURITE(注2) や FLUORITE(注3) を作成している．しかしながら，プログラミング初学者が探索的プログラミングをどのように実施するか，その実施内容がカリキュラムや実施形態によって異なるのかを分析した研究は存在しない．そこで本研究では，複数の大学におけるプログラミング演習において，初学者による探索的プログラミングがどのように行われているかを記録し，調査する．

初学者は対象言語の文法や Library の操作方法を理解していない可能性があることから，探索的プログラミング開始時に正しくプロトタイプが作成できるとは限らない．そのため，本研究では初学者における探索的プログラミングを**特定の変数や引数，条件文を含む行あるいはブロックを連続して変更し，コンパイル，実行していること**とし，その実態を調査する．

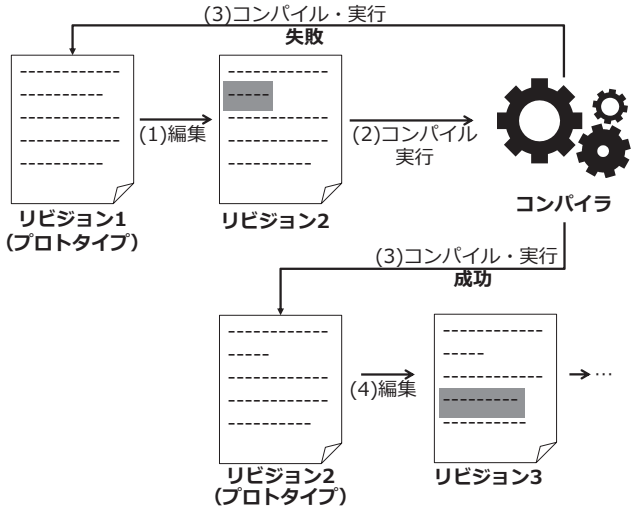

図 1　探索的プログラミングの手順の例

(注2)：http://www.cs.cmu.edu/∼azurite/
(注3)：http://www.cs.cmu.edu/∼fluorite/

表 1　C3PV ログの例

ID	学生番号	操作種別	記録日時	出力結果
1	001	Compile	12-19 15:2e3:00	3:';' expected
2	002	Save	12-19 15:23:00	（出力なし）
3	003	Run	12-19 15:23:15	OK
4	001	Compile	12-19 15:25:00	123
5	004	Run	12-19 15:25:30	20km 走行しました。

3.　探索的プログラミングの分析

3.1　コーディング過程可視化システム C3PV

探索的プログラミングの分析のために，本研究ではコーディング過程可視化システム C3PV [12] を利用した．C3PV はブラウザ上でプログラミングにおいて必要なソースコードの編集，保存，コンパイル，実行といった操作を行うことができる Web アプリケーションである．C3PV は実際のプログラミング演習で教員が学生の進捗を把握することを目的として作られたため，学生を対象とした機能と教員を対象とした機能が存在する．

図 2 に学生が使用する C3PV の画面を示す．学生はコントローラ領域にあるボタンを押すことで課題の保存，コンパイル，実行，提出を行い，エディタ領域にあたるオンラインエディタで実装を行い，出力領域でコンパイルや実行をした時のプログラムの挙動を確認する．

また，教員向けの機能として，各使用者（学生）のプログラミング行動を記録する機能がついており，Save，Compile，Run といった操作を使用者が行った時刻と，そのときのソースコードそのものがログとしてデータベースに保存される．また，編集中にも 1 分ごとにソースコードそのものが保存されるようになっている（autoSave 機能）．なお，Run 実行時には Compile も自動的に行われている．

表 1 に C3PV を用いて保存されたログの一部を示す．左から順にログ ID，学生番号，学生が行った操作種別，記録日時，その時のソースコードを操作したことによる出力結果（Compile あるいは Run が実行されたときのみ）が保存されている．

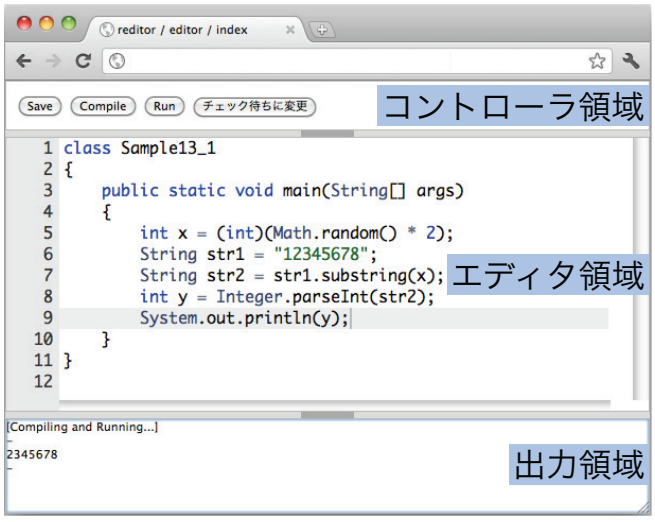

図 2　C3PV のオンラインエディタを用いた Java プログラムのコーディング（文献 [12] より引用）

3.2　初学者における探索的プログラミング抽出基準

初学者は演習で扱っているプログラミング言語や，授業で扱っているエディタ，またプログラミング環境そのものにもまだ不慣れであることが多い．そのため，先述したように従来の探索的プログラミングのようなコンパイルの通るプロトタイプの作成や，エラーが生じた時にプロトタイプのソースコードに戻ることは困難であると考えられる．以上のことから，今回の分析では同一行を 4 回以上連続して編集している際に，探索的プログラミングであると判断して抽出を行った．

3.3　日本の初学者を対象とした実験手順

日本の大学における初学者の探索的プログラミングの実態を調査するために，以下の手順で分析を行った．

手順 1： 実際のプログラミング演習におけるプログラミング初学者のプログラミング行動を取得する

手順 2： 手順 1 で取得したプログラミング行動のうち，3.2 節に示した初学者における探索的プログラミング定義にあたるプログラミング行動を抽出

手順 3： 手順 2 で抽出した探索的プログラミングを著者らが分析する

まず手順 1 では初学者における探索的プログラミングの分析のため，41 名の学部 1 年生を対象とした Java の初学者向けプログラミング演習（90 分の授業が 2 コマ）において，学生のプログラミング行動を C3PV を用いて収集した．

次に手順 2 では収集したプログラミング初学者のプログラミング行動を 3.2 節で示した初学者における探索的プログラミングの定義に従い抽出した．

更に，手順 3 では手順 2 において抽出した初学者における探索的プログラミングのプログラミング行動を人手により分析した．分析は主に以下の 2 つの視点より行った．

定量的な分析

定量的な分析では以下に示すデータの数および時間を求めた．

- 確認された探索的プログラミングの件数及び探索的プログラミングを行っていた初学者の人数
- 収集した結果の Save，Compile，Run，Autosave の回数
- 探索を行っている時間

定性的な分析

定性的な分析では著者らが抽出されたプログラミング行動及び定量的な分析で得られたデータを元に，初学者における探索的プログラミングの分類，特徴付けを行った．

3.4　タイの初学者を対象とした実験手順

タイでは実際のプログラミング演習で C3PV を使用することが困難であったため，演習とは別に課題を提示し，日本にあるサーバーに導入された C3PV 上で課題を解いてもらうことで初学者のプログラミング行動を収集する．タイの初学者のプログラミング行動のデータ収集後（手順 1'）は，3.3 節の日本の初学者を対象とした実験手順の手順 2，手順 3 を同様に行う．

次節でタイの初学者におけるプログラミング行動の収集方法について詳しく述べる．

表 2　日本の初学者における実験の
スナップショットの内訳

人数（名）	Save	Compile	Run	Autosave	Other	Total
41	1024	284	1161	3485	98	6052

表 3　探索的プログラミングの傾向が見られた日本の初学者の
スナップショットの内訳

人数（名）	Save	Compile	Run	Autosave	Other	Total
16	568	169	518	1445	3	2703

3.4.1　タイの初学者におけるプログラミング行動の収集

タイの大学の初学者におけるプログラミング行動の収集のため，C#を半年間習った学部 2 年生 17 名を対象に C3PV を用いて実験を行う．対象の学生には予め C3PV の扱い方と実験の流れを書いたドキュメントをメールで送信する．実験は，3 時間で日本の学生と同じ課題を 5 問解いてもらう．実験開始時刻になると，対象の初学者は支持された URL に記載されている課題を，1 時間で可能な限り回答する．また実験終了後，対象の初学者のプログラミング経験を知るために簡単なアンケートに答えてもらう．

なお，タイの実験で用いる課題は全て英語で表記されているが，カセサート大学ではプログラミング演習の座学の授業で用いるスライドは全て英語で表記されている．そのため，課題が英語で表記されていることによる大幅な時間の遅延は生じないと考える．また，タイの実験で用いる課題は，日本の初学者に探索的に解かれる傾向が多かった問題の上位 5 つを用いる．

4.　日本の大学における初学者の探索的プログラミング実態

本章ではまず手順 1 で得たプログラミング初学者のプログラミング行動の内訳を示す．その後，手順 3 で得たプログラミング初学者における探索的プログラミングの定量的分析の結果および著者らが行った定性的分析の結果を，実際に得られた探索的プログラミングの例とともに示す．

合計 180 分の Java プログラミング演習で C3PV を使用した結果，6052 個に及ぶスナップショットを得た．収集したスナップショットの内訳を表 2 に示す．

Other には C3PV の機能である学生による課題の提出（Submit）と課題を閉じた時の最終スナップショットを自動保存する機能（Close）の 2 つが含まれている．また，C3PV の AutoSave 機能により 1 分おきに自動でソースコードのスナップショットがデータベースには保存されているが，前回に保存したスナップショットが Autosave かつ差分がなかった場合はカウントしていない．Compile の数が Run の数より著しく小さい理由は，Run 実行時には Compile も自動的に行われているためであると考えられる．

4.1　定量的分析による結果

本実験の定義にもとづく探索的プログラミングの傾向が見られた初学者の人数は 41 名中 16 名であり，探索的プログラミングの事例数は 20 件であった．探索的プログラミングの傾向が見られた初学者のスナップショットの内訳を表 3 に示す．探索を行っていた時間（探索的な変更と判断され抽出された連続したスナップショットのうち，最初のスナップショットの保存から最後のスナップショットの保存までかかった時間）については，最短で 3 分，最長で 38 分，平均 12.1 分に渡って特定の行に対する変更が連続して行われていた．

4.2　定性的分析による結果

本実験で得られた 20 件の初学者における探索的なプログラミング行動のうち 1 件について図 3 に示す．この例では，学生は subString と parseInt という API を利用して，文字列を数値に変換した後に指定された計算を行う課題を解いている．R1 が今回特定した学生の探索的なプログラミング行動の最初の状態である．この学生は R1，R2，R1，R3，R1，R4 という順でソースコードの編集を行った（実際には各状態の間に複数回の異なる変更が含まれている）．変更された箇所は 5 行目から 7 行目の範囲である．

この学生の場合，substring の返り値の方が String 型であることや，それによって得られた結果をどのようにして int 型の変数に代入してよいか分からず試行錯誤する過程が見られた．この学生は最終的に R4 に辿り着くまでに 12 分かけて 16 回の Run コマンドを実行していた．また，そのうち 5 回は R1 のソースコードへ手戻りを行っていた．

この過程からは学生は R1 のソースコードをプロトタイプとして探索的プログラミングを行っていることが分かる．しかし，経験のある開発者とは違い初学者はエラーが生じないプロトタイプを作成できない状態で探索を行うことや，また手戻りを行

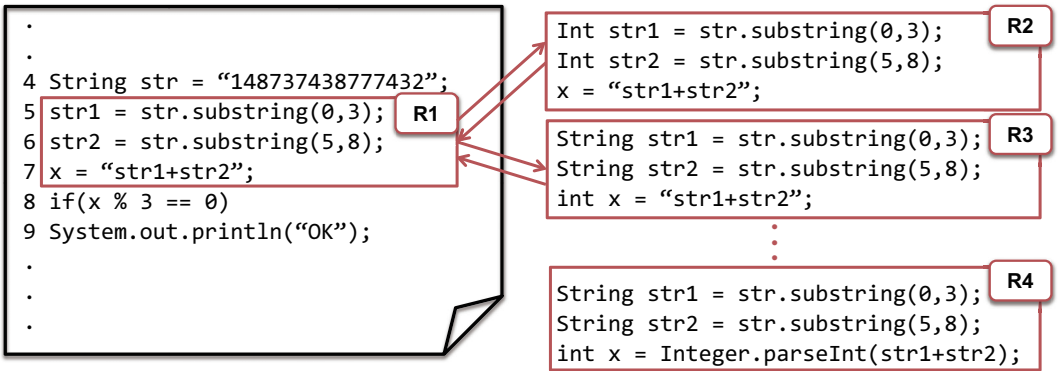

図 3　日本の初学者における探索的なプログラミング行動の例

うことなく同一変数や同一行を変更し続けるような探索の例も確認された．今回の日本における分析では同一行や同一変数を4回以上連続して編集していた場合探索的と判断したが実際にはより多くの探索的プログラミングが実施されている可能性が高い．実際，既存研究で述べられている探索的プログラミングの基準の1つである，編集を行った後すぐにコンパイル・実行を複数回繰り返している事例を調べたところ，5分以内にコンパイル・実行をやり直しているログが1128件確認された．すなわち，探索的プログラミングは初学者においても非常に多く行われている可能性が高いといえる．

5. 考　　察

2.2節の比較および4.章の日本における探索的プログラミングの結果を踏まえ，タイの初学者における探索的プログラミングの考察を述べる．

まず，タイの初学者において探索的プログラミングが行われる頻度について考察する．今回比較に用いるカセサート大学では，2.2.2節で述べたようにソースコードの提出を促していない．教員は学生の書いたソースコードを読まない代わりに，学内の採点システムによって実行された5つのテストケースの結果を持って，学生の演習の成績評価を行う．そのため，カセサート大学の学生は出力結果に応じたソースコードの書き換えに慣れているのではないかと推測される．従って，出力結果が自分の想定するものと違っていた場合，熟考するよりも先に，自身が扱いなれていないAPIや再帰関数，条件分岐関数の条件にあたる部分を，自分の想定する結果が出力されるまで修正と実行を繰り返す可能性が考えられる．よって，探索的プログラミングが行われる頻度は増加すると考えられる．

次に，探索的プログラミングの特徴について考察する．日本でもタイでも，出力されるエラーメッセージは英語であるため，日本の初学者に比べカセサート大学の初学者はエラー箇所の特定に慣れているのではないかと推測される．したがって，正しい出力は行えていない場合もあるが，コンパイルエラーや実行時エラーの回数は日本よりも少ないと考えられる．日本の初学者において，コンパイルや実行が可能なプロトタイプを作成せず，コンパイルエラーや実行時エラーを繰り返しながら探索を繰り返す様子が確認されたが，このような事例は減少すると考えられる．

さらに，探索的プログラミング以外のプログラミング行動について考察する．カセサート大学では普段の演習で統合開発環境であるMono developer(注4)を使っている．したがって，C3PVのような自動でコンパイルや実行結果を確認できるような環境には慣れていると考えられる．一方，日本の実験を行った大学では，普段はテキストエディタを用いてコーディングを行い，コマンドプロンプトでコンパイルを行った後に手動でプログラムを実行していた．つまり，日本の被験者はコーディングを行ってから実行結果を確認するまで，普段の演習とは違う過程で演習を進めていた．そのため，タイの学生の方が演習にとりかかる時間や課題を進めていく時間が日本の学生より早くなるのではないかと推測される．

(注4)：http://monodevelop.com/

6. ま と め

本研究では日本とタイにおけるプログラミング教育の比較を行い，これらの違いが初学者における探索的プログラミングのプログラミング行動について，どのような影響を及ぼすか考察を試みた．そのために，まずプログラミング演習において日本の学生を対象とした探索的プログラミングの実態調査および分析を行った．今後，日本の調査結果をふまえ，タイの初学者はどのように探索的プログラミングを行っているのか実験を行い，日本とタイにおけるプログラミング教育の違いが学生のプログラミング行動にどのような違いを生み出しているのか分析する予定である．

謝辞　本研究は日本学術振興会 科学研究費補助金 24700030, 26540182, 26730036の助成を受けている．

文　　献

[1] 独立行政法人情報処理推進機構，IT人材白書 2012，独立行政法人情報処理推進機構，2012.
[2] “Coding for kids: schoolchildren learn computer programming”. http://www.telegraph.co.uk/technology/10468460/Coding-for-kids-schoolchildren-learn-computer-programming.html.
[3] “NAIST シラバス”. http://is-education.naist.jp/.
[4] “大阪工業大学シラバス”. http://www.oit.ac.jp/japanese/syllabus/.
[5] “東京工科大学 プログラミング基礎 I”. https://kyo-web.teu.ac.jp/syllabus/CS_CK03801_ja_JP.html.
[6] “第7章　サイバー大学における演習講義の取り組み：java プログラミング演習事例報告”. http://www.cyber-u.ac.jp/about/e-learning_research/0002/pdf/0002_08.pdf.
[7] 玉田春昭，荻野晃大，上田博唯，“アシスタントロボットを用いたプログラミング教育支援システムの構築,” 信学技報 マルチメディア・仮想環境基礎研究会，no.MVE2010-48，pp.143–148，June 2010.
[8] D.W. Sandberg, “Smalltalk and exploratory programming,” ACM SIGPLAN Notices, vol.23, pp.85–92, Oct. 1988.
[9] Beau. Sheil, “Environments for exploratory programming,” Datamation, vol.29, no.7, pp.131–144, July 1983.
[10] Y. Yoon and B.A. Myers, “An exploratory study of backtracking strategies used by developers,” Proceedings of the 5th International Workshop on Cooperative and Human Aspects of Software Engineering, pp.138–144, June 2012.
[11] “Variations to support exploratory programming”. http://www.exploratoryprogramming.org/.
[12] 井垣　宏，齊藤　俊，井上亮文，中村亮太，楠本真二，“プログラミング演習における進捗状況把握のためのコーディング過程可視化システム C3PV の提案,” 情報処理学会論文誌，vol.54, no.1，pp.330–339，Jan. 2013.