Vol.2020-CE-154 No.16
2020/3/14

タンジブルな変数環境を活用した高校情報科のアルゴリズムプログラミング教育の実践

太田 剛[†1]

概要：新学習指導要領の情報Ⅰにおいてアルゴリズムのプログラミング教育が必須となる．このアルゴリズムの学習において，生徒・学生の困難さが指摘され，ビジュアル化やハンドトレースなどの手法が用いられている．筆者は，これらの従来の方法だけでは不十分であると考え，プログラミングの認知的過程を考慮し，実際に手で操作できるタンジブルな変数模型をフローチャートなどと用いる教材の開発を行い，実践を行っている．現在実践中であるが，普通の高校生でも並び替えなどのプログラミングを作ることのできる授業方法を提案した．

キーワード：プログラミング教育，高校情報科，タンジブル

Trial of Algorithm Programming with Utilizing Tangible Variable Environment in Informatics Education for High Schools

GO OTA[†1]

1. はじめに

新学習指導要領[1]において，高校情報科で必履修となった科目「情報Ⅰ」の内容に「コンピュータとプログラミング」が含まれることから，全高校生が 2022 年度よりプログラミングを学ぶことなる．この実施にあたっては，情報科の教師にプログラミングの高度な内容が教育できるか問題視され，現在先行していろいろなプログラミングの授業の実践が始まっている．ただし，それらは簡単なプログラミングを扱ったものが多く，新指導要領解説で「アプリケーションソフトウェアが持つ検索や置換及び並べ替えなどの機能の一部を実現したり」と明示され，高等学校情報科「情報Ⅰ」教員研修用教材[2]の中の「アルゴリズムの比較」でも取り上げられている検索や並べ替えを扱った授業は，中西[3]の「検索はともかく整列は二重ループが使われるので，生徒にとって簡単なものではない」の言葉にあるように，普通の高校では実践が自体十分に行われていないようである．

これに対して，筆者は 2018 年度よりアルゴリズムのプログラミング授業を実践していて，普通の高校生が情報科で検索や並び替えのプログラミングを学習する方法を模索している．本稿は，情報科の中でアルゴリズムのプログラミングのための教材や授業方法の実践結果を報告するものである．2 章ではプログラミング教育の現状を，3 章では，主にプログラミング学習の支援を考慮した 2018 年度の実践(以後，実践 2018 と略す)内容，結果とそこから得られた知見について説明する．そして，4 章では実践 2018 の反省からプログラミング学習の認知過程により注目し，タンジブルな手にとって操作できる変数模型を使用して大幅に改良した教材と，第五章では，その教材を使用した 2019 年度の実践(以後，実践 2019 と略す)状況を示す．最後に，今後の高校でのアルゴリズムのプログラミング学習の課題や授業への利用等に関して述べる．なお，タンジブルに関する，現在多くの研究は物理的な道具とコンピュータを融合したタンジブルインターフェースを扱っているが，本稿では変数などコンピュータ内部で見えないものを手に取って操作できるものという意味で，タンジブルという言葉を使用している．

2. プログラミング教育の現状

情報Ⅰのプログラミングに対応して現在進められている実践は，①Python/JavaScript 等の汎用テキスト言語を使用して，分岐や繰り返しなどを学習する[4]，②ビジュアル言語を用いて分岐や繰り返しなどを学習する[5]，③アンプラグドで並び替えなどのアルゴリズムを学習する[6]，④データサイエンス[7]や Web などのアプリケーションなどの実用プログラムを開発する[8]，⑥ロボットや他のフィジカルな装置を制御・計測する[9]，等に大きく分類されると考えられる．ただし，高校での検索や並び替えに関しては，文部科学省のプログラミング教育実践ガイドとして「基本的なアルゴリズムの学習」[10]の指導案は公開されているが，その中にある探索・並べ替えについての具体的な実践報告は公開されていないようである．

これに対して高等教育では情報関連の学部・学科においてアルゴリズムのプログラミング教育は長年行われ，一般

†1 千葉県立袖ケ浦高等学校/市川南高等学校
Chiba Prefectural Sodegaura High School/ IchikawaMinami High School

に配列・添え字による配列の操作・二重ループなどが学生にとって難しいものであることは，多くの指導者の経験知として認識されていると考えられる．そして，これらの問題に対応する支援システムや教授法が開発・研究されてきて，伊藤ら[11]は，これらを，大きく「説明生成（可視化）型」システムと，「診断・指導型」システムとに分類している．「診断・指導型」システムは，デバッガやトレーサなどのプログラム作成支援ツール[12]，アルゴリズムアニメーション，変数の可視化ツール[13]等である．そして，「診断・指導型」はプログラムの自動判定のフィードバックや穴埋め問題提示などのプログラムの作成補助ツール[14]である．

3. プログラミング学習の支援を考慮した実践(実践 2018)

3.1 考慮点

先行実施されている高校・大学等のプログラミング授業を参考に，以下のような点を考慮して教材開発し，高校の情報専門科目「アルゴリズムとプログラム」の授業で実践した．

① 可視化とトレース

プログラミング学習支援で，変数などの情報を可視化することが有効であることから，ビジュアル言語である Scratch を使用することとした．Scratch は簡単な指定で，変数やリストの内容をプログラム実行画面に表示することができる．そして，授業時には，繰り返しにおいて，変数の値の変化を紙に書くようにトレースの指導を行った．ただし，Scratch 自体にはブレークポイント等を設定するデバッグ機能がないため，プログラムとしてキー入力があるまで，動作を停止する方法も指導した．

なお，Scratch のようなビジュアル言語は高校での使用は不適切というような意見も SNS 上にはあるが，新学習指導要領には使用するプログラム言語の要件として，ア)アルゴリズムをプログラムとして表現，イ)プログラムから呼び出して使うプログラミング言語やオペレーティングシステム及びサーバなどが提供するライブラリや API(Application Programming Interface)などの機能，ウ)プログラムの修正，エ)関数を用いてプログラムをいくつかのまとまりに分割してそれぞれの関係を明確にして構造化，オ)ツールやアプリケーションを開発，カ)カメラやセンサ及びアクチュエータを，キ)利用したり，画像認識や音声認識及び人工知能などの既存のライブラリを組み込み，と明記されていて，Scratch は，サブルーチン型のモジュール構造しか提供していないため，エ)については若干弱いが，他の要件は十分に見たいしているため問題無いと考える．

② アルゴリズムとプログラミング言語の学習の分離

江木ら[12]は，「プログラミング教育には，プログラミング言語，アルゴリズム，構築技術の 3 つの主要な側面がある，学習者にとっては，これら 3 つは不可分でありほぼ同時進行で学習が進む.」と述べているが，生徒にとって同時進行での学習は難しく，実践 2018 ではプログラミング言語・構築技術とアルゴリズムを分離して学習するようにした．このため，実践 2018 の前に，Scratch を使用した自由なプログラミング作成の授業を実施し，Scratch 自体を十分に使用できるようにしたあと，アルゴリズムに専念できるように考慮した．

③ 問題解決的要素を含む演習課題

多くのプログラミング授業では，高岡ら[8]が示したように「教授者が学習者に対して演習課題を提示し，学習者がその演習課題に取り組む形」であり，「学習者が参考書やその他の資料などを参照し，解決法などの本質的な理解をすることなく演習課題を完成させてしまうことがある.」という問題も指摘している．教材の作成にあたっては，「アルゴリズムとプログラム」の教科書[15]を参照するようにしているが，Web 上に置かれた pdf に記述された課題と指示に従って個人のペースで学習できるものを開発した．なお，課題は，後半部分は教科書に沿った探索，並び替えとしたが，前半部分は基本的な順次，選択，繰り返しをフローチャートと対比させながらプログラムを開発するものである．

具体的には，表 1 に示すチェックシートをもとに，内容の確認やプログラムの開発を行う．なお，教科書には C 言語および BASIC のサンプルプログラムが記載されているので，実践 2018 では Scratch を使用しているため，生徒は教科書の回答を単純にコピーして使用することはできない．

表 1 実践 2018 のチェックシート

内容	チェック		
	理解	解析	開発
変数と Scratch での利用		＼	＼
プログラムの構造/フローチャート		＼	＼
フローチャートと Scratch の対応		＼	＼
一番簡単な自動販売機			＼
チャレンジ: 正三角形の判断	＼		
単純な 1 から n の合計(配列/リストは使っていません)			＼
チャレンジ:単純な 2 から n までの偶数の合計	＼		
配列/リストの補足資料			＼
1 からの合計(リスト版)フローチャート:配列の合計			＼
課題 1(補足): 素数を求めるプログラム(割り算)			＼
課題 1: 素数を求めるプログラム(リスト利用)	＼	＼	
課題 2: 交換法を用いた整列プログラム	＼	＼	
課題 3: とにかく速い整列プグラム	＼	＼	

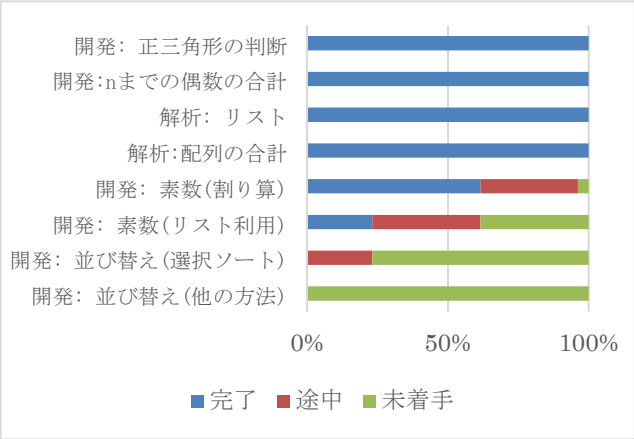

図 1 実践 2018 での各生徒の進捗状況

図 2 実践 2018 教材例：素数(割り算)

3.2 実践方法

公立高校の情報科 2 年生 41 名対して，2018 年 4～5 月にかけて 8 時間の授業を行った. 前述したように, その前に, Scratch を使用したプログラミング授業を 6 時間行っている．授業中は生徒間の助け合いを進めるため，自由な席として生徒同士が相談しやすい状態にした．そして，チーフである筆者とサブの教師 2 名で，質問などある生徒に対して個別に指導を行った．この指導については，同様な質問やつまずきのある生徒を集め，数名に対して PC を操作しながら画面を見せて説明を行った．

3.3 実践結果と考察

授業の最後に，各生徒がどこまで表 1 の課題が終了したか Web フォームで自己申告してもらい進捗を確認した．図 1 に示すように，全員が「配列の合計」のプログラムの意味を理解する解析まで終了したが，約 6 割が割り算方式で素数を求めるプログラムと開発が終了したが，リスト上で倍数にフラグを立てて素数を見つける消去法を使用したプログラムは 2 割しか完了しなかった．さらに，次の課題である並び替えを完了したものはいなかった．

生徒の学習状況を観察していた結果，教材自体に以下のような問題点があるように考えられる．

・[単純な 1 から n の合計]，[単純な 2 から n までの偶数の合計]，[配列の合計]についても生徒は理解や開発に困難を示し，教師が十分に指導解説する必要があった，繰り返し・分岐や配列の理解を促進する説明が教材に不足していた．

・[配列の合計]から[交換法をいた整列プログラム]の間を埋めるものとして, 素数を求めるプログラムを用意したが，このプログラム自体，分岐と繰り返し，配列の操作が入るもので，生徒にとって難易度の高いものであった(図 2 参照).

・プログラミングのアルゴリズムをフローチャートで示したが，フローチャートと例示するプログラムとの対応が不十分であった．

・開発するプログラムの一部をヒントとして提示していたが，このヒント自体が難しいようで，それを元にプログラムを完成させることが生徒にとって難しいようであった．

・アルゴリズムや配列の概念などは教科書を見ることを指示したが，課題と教科書の説明を関連付けて理解することが生徒にとって困難だったようである．

また，生徒がプログラミングする時，次のような困難さが観察された．

・変数の概念が確立されていないようであり，変数に入力することと，変数に定数をあらかじめ代入することの区別がついていないようである．また X = X + 1 等の代入が理解できないようである．

・配列が概念的にイメージできないようであり，添え字によって配列の要素を区別することは，さらに戸惑いを見せた．

・単純な繰り返しや分岐は理解できるが，それらが組み合わさった場合に混乱するようである．また，分岐による繰り返しの修了判断では，その判断に利用する変数と比較する定数の区別がつかないようである．

・配列については，その添え字に変数を使用することが非常に困難である．特に，繰り返しで添え字に利用した変数の内容が変化することは理解しにくいようである．

・上記に関連して複数の配列や変数の値を繰り返しの中で変更することは，どの変数が変化するか理解しにくく混乱するようである．

・フローチャートとプログラムの対応の理解が不十分であり，フローチャートが提示されても，それをどのようにプログラミングするか思いつかないようである．

4. 認知過程を考慮した，改良方法の検討

前述した生徒の困難さの中で，筆者が一番印象に残ったのは下駄箱やロッカーを例えに配列を説明した時に,「下駄箱には数字を入れないから」という生徒の反応である．「60%の人間はプログラミングの素質がない[16](2014 年に撤回)[17]」として話題になったが，プログラミング教育に

たずさわっている人間であれば，生徒・学生に何らかのプログラミング適性を感じることもあり，例に出した生徒は，プログラムの配列を自分が操作する対象として使うには認知的ギャップを持っていたとも考えられる．

この学習者がプログラミングを学ぶ時の認知過程について，三宅[18]が「プログラミング学習において，何らかの認能力を伸ばそうとするのなら，そもそもプログラミングがどういう認知過程なのか，そこから調べていかなくてはならない」と指摘したように，すべての生徒がプログラミングを習得できる教育を実施するためには，この認知過程の分析とその対応が必要であると考える．但し，国内でのプログラミング時の詳細の認知過程に関する研究は少なく，前田ら[19]のプログラミング時の操作プロトコルの分析があるが，プログラミングの認知的ギャップについて詳細に解説するものではない．そして，小松[20]はプログラミング教育での認知的側面の重要性を示すとともに，「変数の壁」として変数を箱で例えて説明することを論議している．

実践 2019 では，なぜ学習者がプログラミングできないか，なぜ認知的なギャップが生じるかの根本的な原因を追究するものでは無いが，実践 2018 で観察された認知的ギャップを処方的であっても，例えば単なる画面上に可視化するよりは，より適した方法で教材を改善することを目標とした．

プログラミングでのアルゴリズム教育以外に，学習者がコンピュータ科学を理解できるよう，図 3 に示すような，アンプラグドコンピュータサイエンス，フローチャート等によるアルゴリズム教育が実施されている．特に並べ替えはプログラミングでの実装が難しいため，間宮ら[6]は天秤や天稟ソフトを使用してアルゴリズムの実践を行っている．ただし，アンプラクドで並べ替え，プログラムの動作を学習した後に，それを汎用的なプログラミング言語での実装に融合するような実践は少ないようである．そして，中西ら[21]はフローチャートとプログラミングを融合させた PenFlowChart を開発し，プログラミング導入に効果があることを示しているが，配列や並び替えなどのフローチャートで表現が難しい課題についての実践については不明である．このようなプログラミング以外の教授法と融合することにより，プログラミング作成時の認知的ギャップを解消できるようになると考えた．

そして以下の 3 点を教材の改良の中核とした(表 2 参照)．

① タンジブルな変数環境の利用

従来はトレースなどの紙に書きだすことや，画面上に可視化することで支援することが行われた変数や配列について，具体的に手に取って操作でき，変数のイメージに近い箱型の模型をタンジブルな変数環境として用意することとした．そして，繰り返しのカウンターとして使用する変数については，その操作がよりイメージしやすいように計数機を使うこととした(図 4 参照)．

② タンジブルな変数，フローチャート，プログラムの対応の明確化

生徒がプログラムのイメージをつかみやすいようにするため，タンジブルな変数，フローチャートとプログラムを明確に関連付けるようにした(図 5 参照)．

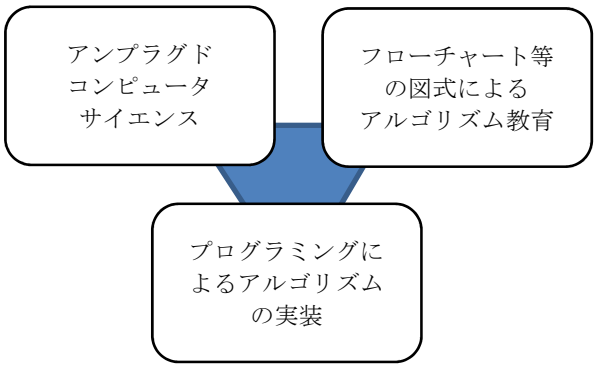

図 3 アルゴリズムの教育方法

表 2 **実践 2019 のチェックシート**

内容	チェック		
	理解	打込み	開発
変数と Scratch での利用			
Scratch での変数への入力			
変数(X=X+1)			
プログラムの構造/フローチャート			
フローチャートと Scratch の対応			
合格の判断			
一番簡単な自動販売機			
チャレンジ: 正三角形の判断			
1 から 10 を言う			
単純な 1 から 10 の合計			
チャレンジ:単純な 2 から A までの偶数の合計			
5 回数字を入力してその合計を求めます.			
配列(リスト)を作る			
おくみくじを作る			
チャレンジ: 5 回数字を入力してその合計(配列利用)			
配列に数をセットする方法			
チャレンジ: 配列の中から数を探す			
数の並び替えの作業の説明			
配列の中の一番小さい数を見つける			
配列の中の一番小さい数を配列の先頭に入れ替える			
二重繰り返しに挑戦			
最期のチャレンジ:数の並び替え			
発展課題 1: 沢山の数の並び替え			
発展課題 3: 並び替えの無駄を省く			
発展課題 2: バブルソート			

補足: 対応する教材は
http://beyondbb.jp/HI1materails/Unit05__Algo.pdf
で公開済み

図 4 タンジブルな変数

図 5 教材例

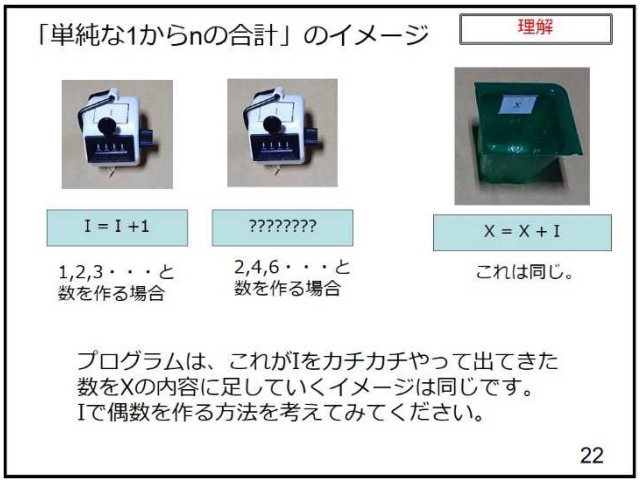

図 6 1～n の偶数の合計の補足教材

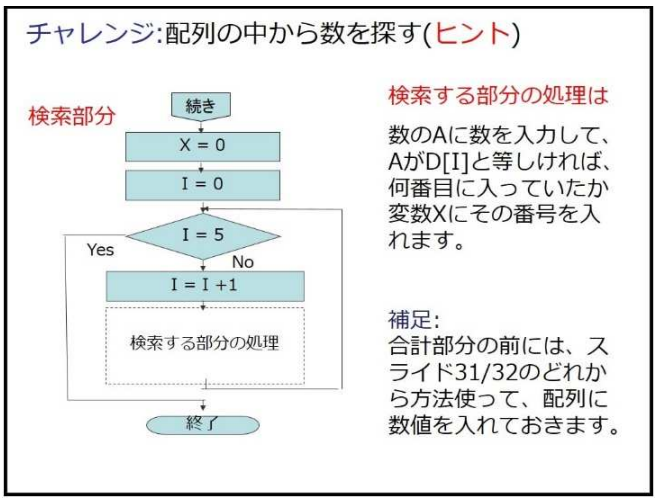

図 7 フローチャートの穴あき例

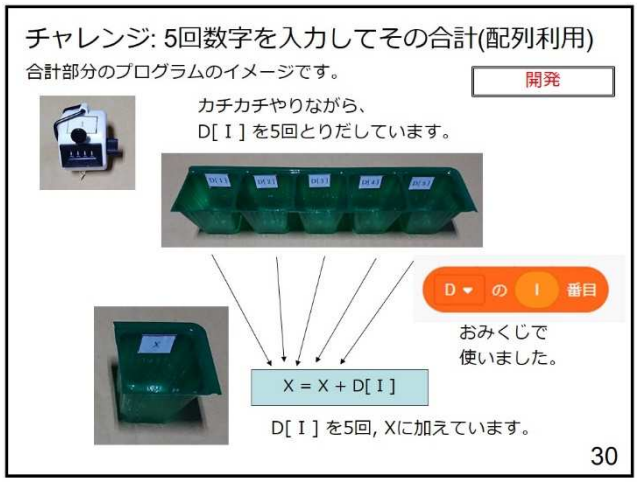

図 8 配列要素の取り出しのヒント例

③ スモールステップであるが考える内容

実践 2018 から，生徒がプログラミングのどのような点でつまずくか判ってきたため，生徒がスモールステップで確実にプログラミングできるように教材の粒度を細かくした．ただし，すぐに生徒が回答を見つけたり，単に例を写経のようにコピーするのではなく，ある程度考えて作ることも考慮した．例えば，1～n の合計を作るプログラムを完成後，1～n の偶数の合計を求めるプログラムでも生徒にとってハードルが高いため，偶数をプログラムで作るヒントなど追加した．(図 6 参照)．また，②に関連して，プログラムを作るための手助けにするため，前の課題で作ったプログラムとの違いを明確にするため，一部穴抜けにしたフローチャートをヒントとして提示するようにした(図 7 参照)．そして，2018 年度は教科書を使用することを前提として教材であったが，今回はスライドだけで学習できるようにした(図 8 参照)．さらに情報 I での全体のカリキュラムの中に入れることとから授業の想定時間も 6 時間とした．

改良した教材による実践 2019 は公立高校の情報科 2 年

生 19 名対して，2020 年 2 月より 6 時間の授業を行っているところである．この授業の前に，Scratch を使用したアプリ開発の授業 8 時間を行っているため，Scratch 自体の操作は十分に修得している状態である．さらに，アンケート調査の分析で Excel を使用して並び替えがどのようなものかも学習している. まだ, 最終的な実施結果はでていないが，本稿執筆時までの実践の様子や現在まで得られた知見を示す．

・4 時間の授業が終了した段階であるが，多くの生徒が順調に学習を進めていて，半数以上の生徒が検索である「配列の中から数を探す」を終了している．

・タンジブルな変数については，生徒自身が主体的に利用することは少ないが，生徒がつまずいているときに配列やカウンターを筆者が説明することに使っていて，その説明によって生徒の理解も進んでいるようである．また，スライドにもタンジブルな変数の説明があるが，実際に紙に書いて，変数の箱に出し入れすることが生徒にとって解り易い説明になるようである．

・フローチャートについては，多くの生徒が初めはあまり利用して無いようだったが，問題が難しくなると，すでに作ったプログラムとの比較のヒントとして有効であることに生徒も気づいたようで，フローチャートからプログラムを考えているようである．

・実践 2018 の知見から生徒のつまずきを手助ける細かいヒントを教材の中に入れた．例えば，配列を添え字に変数を使って要素を取り出すプログラムは生徒にとってすぐに思いつかないものであるが，図 8 の Scratch でのリスト要素のアクセスするブロックを提示するだけで，生徒のプログラミンクは楽になっているようである．

・スモールステップの教材ではあるが，生徒自身が苦労して考える部分も多く，プログラムが正しく動くようになった時の喜びや達成感を感じているようである．

5. まとめと今後の課題

本実践はまだ，授業の途中であるか，学習指導要領に示された「検索や置換及び並び替え」を普通の高校生がプログラミングするという授業のスタイルを示せたと思う．ただし，このようなスモールステップで実施することにより，生徒がアルゴリズムや，そのプログラミングを本質的に理解したかという疑問は残る．

前述したようにプログラミングが得意な生徒・不得意な生徒もいて，科学的な根拠は一般には無いが進学校などにいる学力の高い生徒(論理的思考能力や推論力が高い生徒)は得意と考えられるようである．本来，高校のプログラミングの内容は，中学でプログラミング教育が技術科の中で実施されていることを前提にしていると考えられるため，，もし中学でのプログラミング教育が十分に担保されないようであれば，普通の学力レベルの高校で「検索や置換及び並び替え」を学習しないという選択肢も考えるべきだと思う．ただし，文部科学省の情報科教員用の研修資料において，本稿で示した教材を使うことは有効だと考えられる．

そして，本稿ではプログラミング教育の研究・実践において，もっと何がわからないのか，なぜわからないのか，どうしたらわかるようになるかの，もっと詳細な認知的な面の研究が必要であることも提案している．現在のすべての人にプログラミング教育が必要とされる時代にあって，すべての高校の生徒がプログラミングを理解・習得するためには，多くの研究者や教師が，より個々の生徒の学習の状況を詳細に分析することが必要だと考える．

参考文献

[1] “文部科学省. 高等学校学習指導要領(平成 30 年告示), https://www.mext.go.jp/a_menu/shotou/new-cs/1384661.htm, (参照 2020-2-19).
[2] 文部科学省. 高等学校情報科「情報 I」教員研修用教材, https://www.mext.go.jp/a_menu/shotou/zyouhou/detail/1416756.htm, (参照 2020-2-19).
[3] 中西渉. 高校におけるプログラミング教育. 情報処理, 2016, Vol.57, no.4, p.358-361.
[4] 間辺広樹, 長島和平, 並木美太郎, 長慎也, 兼宗進. 高等学校における複数言語によるプログラミング教育の提案. 情報処理学会論文誌教育とコンピュータ(TCE), 2017, Vol.3, no.3, p.29-41.
[5] 伊藤一成. クトグラミング―人型ピクトグラムを用いたプログラミング学習環境. 情報処理学会論文誌教育とコンピュータ(TCE), 2018, Vol.4, no.2, p.47-61.
[6] 間辺広樹, 神藤健朗, 並木美太郎, 兼宗進. コンピュータ・アルゴリズムの「発見・記述・伝達」を導く授業の実践と評価. 情報処理学会論文誌教育とコンピュータ(TCE), 2016, Vol.2, no.1, p.10-24.
[7] 兼宗進, 白井詩沙香, 竹中一平, 長瀧寛之, 小林史弥, 島袋舞子, 田邊則彦. データベースと情報システムを学習する授業の提案と実践. 情報処理学会論文誌教育とコンピュータ(TCE), 2017,
前田恵三，中野靖夫. プログラム作成過程の分析，日本教育工学雑誌，1995, vol.19，no.3，p.171-180.Vol.3, no.3, p.18-28.
[8] 高岡詠子, 山内崇裕, 滑川敬. 章高等学校における実用的プログラミングの教育実践. 情報処理学会論文誌教育とコンピュータ(TCE), 2016, Vol.2, no.2, p.37-52.
[9] 富永浩之, 中井智己, 辻健人, 劉世博, 花川直己. 高大連携の導入講座としての LEGO プログラミング演習の実践. 情報処理学会論文誌教育とコンピュータ(TCE), 2018, Vol.4, no.2, p.1-13.
[10] 文部科学省. プログラミング教育実践ガイド, https://www.mext.go.jp/a_menu/shotou/zyouhou/detail/1408013.htm, (参照 2020-2-19).
[11] 伊藤良二, 小西達裕, 伊東幸宏. プログラムの問題領域上での動作説明を行うプログラミング学習支援システムの構築. 人工知能学会誌, 2000, Vol.15, no.2, p.362-375.
[12] 江木鶴子, 竹内章. プログラミング初心者にトレースを指導するデバッグ支援システムの開発と評価. 日本教育工学会論文誌, 2009, Vol.32, no.4, p.369-381.
[13] 大城正典, 永井保夫. 初学者向けプログラミング学習のための初等アルゴリズム視覚化システム. 「情報教育シンポジウム(SSS2018)」予稿集, 2018, p.104-111.
[14] 掛下哲郎, 柳田崚, 太田康介. 穴埋め問題を用いたプログラミング教育支援ツール pgtracer の開発と評価. 情報処理学会論文誌教育とコンピュータ(TCE), 2016, Vol.2, no.2, p.20-36.
[15] 実教出版. アルゴリズムとプログラム. 2014.

[16] Saeed D., Richard B.. The camel has two humps ,2006, Middlesex University, Working Paper.
[17] Richard B.. Camels and humps: a retraction ,2014, Middlesex University, Working Paper.
[18] 三宅なほみ. コンピュータを教える；岩波講座 教育の方法 教育と機械, 1987, 岩波書店, p120－159.
[19] 前田恵三, 中野靖夫. プログラム作成過程の分析, 日本教育工学雑誌, 1995, vol.19, no.3, p.171-180.
[20] 小松香爾. プログラミング教育の問題と対策. 2015, 経営論集, Vol.25, no.1, p.83-104.
[21] 中西渉, 辰己丈夫, 西田知博. PenFlowchart を用いた, フローチャートによるプログラミング学習の効果に対する評価. 情報処理学会論文誌教育とコンピュータ(TCE), 2015, Vol.1, no.4, p.75-82.