プログラミング初学者の課題解決方針の立案能力を高める選択肢タップ式学習支援システム

岡 大貴 * 大西鮎美 * 西田健志 * 寺田 努 * 塚本昌彦 *

概要. プログラミングが教養となりつつある近年では，さまざまな教育機関でプログラミングに関する授業が実施されている．しかし，初学者にとってプログラミング学習はつまずきの連続であり，特にプログラムを使った課題解決の方針を考えることや，その方針をプログラムコードを書いて実装するのが難しいことが学習障壁となっている．そこで，本研究ではプログラミングを「課題の解決方針を立案すること」と「解決方針をコーディングすること」に分離し，前者のみに専念させることで高速な試行錯誤を可能にし，プログラム課題の解決方針を立てる能力を高めるための学習システムを提案する．提案システムのプロトタイプを，スマートフォンからの利用を想定した Web アプリケーションとして実装し，ユーザが利用することでプログラム課題に取り組む際にどのような効果が得られるか実験を行った．

1 はじめに

プログラミング的思考 [1] や計算論的思考 [2] を育むことを目的とし，小学校においてもプログラミング学習が必修化された昨今では，プログラミングが誰もが身につけるべき教養となりつつある．プログラミングに関する技能を伸ばすことは重要である一方で，初学者にとってプログラミング学習はつまずきの連続であり，多くの学習障壁が伴う．

プログラミング初学者が学習時に直面する困難については多くの調査がなされており [3, 4, 5]，関数やポインタなど「プログラミングに関する概念の理解に関するもの」と，プログラムを設計することやその設計方法の理解，実装する機能を複数の手順に分割することなどの「プログラム設計の過程に関するもの」とがある．プログラミングに関する概念やプログラミング言語の理解を支援する学習教材やツールは多数存在するが [6, 7]，プログラムの設計の過程を支援するものは少ない．プログラムを設計する力を身につけるには，実際に課題に取り組んで試行錯誤し，課題を解決する経験が必要である．しかし，プログラミングにおける試行錯誤は難しい．

プログラム課題に対して試行錯誤することが難しい原因として,「プログラム課題を解くためにまず何から考えればよいのかがわからない」ことがあげられる．初学者はプログラミングでの課題解決の経験が少ないため，課題解決のために解決方針を立案することが難しい．そして「考えた解決方針を実践することが難しい」ことも原因である．初学者がある解決方針を考えついたとしても，それをプログラムにし実践するためには，プログラミングに関する概念や文法を理解したうえでプログラムを記述し，エラーを取り除いてプログラムを完成させなければならない．

そこで本研究では，プログラミングを「課題の解決方針を立案すること」と「解決方針をコーディングすること」に分離して前者のみに専念させ，課題解決のための解決方針を選択式で表示し，有限な解決方針を選択するのみで高速に試行錯誤を経験させることで，プログラム課題解決のためのアプローチを学ばせ，解決方針の立案能力を高めるシステムを提案する．

提案システムは視覚的な出力を用いたプログラミング入門講義の課題で用いられることを想定しており，スマートフォンからの利用を想定した Web アプリケーションとして実装した．ユーザは取り組む課題を選択し，課題の解決方針の選択肢をタップするのみで対応したコードが挿入され，手軽に課題解決のための方針を試すことができる．例えば「日本国旗をプログラムで描画する」という課題の場合，まずユーザは表示される 3 つの選択肢「円を描く」「図形を塗る色を赤に指定する」「四角形を描く」の中から,「円を描く」という解決方針を選択する．次に「円を描く関数を使い，描画領域の中心に円を描く」「円を描く関数を使い，原点に円を描く」といった方針の実装手順に関する選択肢が表示される．日本国旗のような円を描くためには描画領域の中心に円を配置する必要があるので,「円を描く関数を使い，描画領域の中心に円を描く」という選択肢を選ぶ．すると，この実装手順をプログラムで表現したものが表示され，ユーザはそれを確認し選択すると，エディタにプログラムが挿入される．ユーザはこのプロセスを繰り返すことで課題を解いていく．本論文では，プロトタイプシステムを実装し，これを用いて初学者に 3 問のプログラム課題を初学者に解かせ，プログラム課題の解決方針の立案にどのような

* 神戸大学

効果があるかを調査した．

2 関連研究

2.1 プログラミング学習の学習障壁に関する研究

プログラミング学習時の学習障壁については広く調査がなされている．Derus らはプログラミングの基礎講義を受講している学生に，プログラミング学習の難しさに関するアンケート調査を行った．その結果，難しい学習内容は多次元配列や繰り返し，関数などの抽象概念や，プログラムの構造を理解したり，問題を解決するためのプログラムを設計したりすることであった [3]．Lahtinen らも 500 人以上の学生と教師を対象に，プログラミング学習の難しさに関するアンケート調査を行った．その結果，最も難しい概念は再帰やポインタと参照，抽象データ型のような抽象的な概念であり，最も難しいタスクは「ある課題を解決するためのプログラムをどのように設計するかを理解すること」「機能をステップに分けること」そして「自分のプログラムのバグを見つけること」であった [4]．また，Ismail らはプログラミング教育の問題点について，5 人のコンピュータサイエンスの教師にインタビュー調査を行った．その結果，学生がプログラミング学習で直面する問題の 4 つの主な原因は，(1) 問題を分析するスキルの不足，(2) 効果的でない問題解決のための表現，(3) 問題解決とプログラミングの効果的でない指導方法，(4) プログラムの文法と構造を理解し習得する能力の不足，であると述べた [5]．

これらの結果をみると，プログラミングの学習障壁には，関数やポインタなど「プログラミングに関する概念の理解に関するもの」と，プログラムを設計することやその設計方法の理解，実装する機能を複数の手順に分割することなどの「プログラムの設計過程に関するもの」とがある．本研究では，プログラムの設計過程に関する学習障壁に焦点をあて，なかでもプログラム課題の解決方針の立案能力を高めることを目指す．

2.2 初学者向けプログラミング学習のアプローチに関する研究

テキストベースのプログラミング言語についていきなり学ばせるのではなく，初学者のためにより学習しやすいアプローチを提案するシステムは数多く存在する．なかでも小中学校等のプログラミング学習には Scratch [8] を代表とするブロックベースのビジュアルプログラミング言語 (VPL: Visual Programming Language) が用いられることが多い．これらを用いた学習は構文エラーやタイプミスなどが発生しないため，考えた解決方針を実践することをある程度簡易化しているといえる．

また初学者向けプログラミング学習システムとして，テキストベースのプログラミング言語学習への足場かけを目的としたものがいくつか提案されている．中西らは，ビジュアルプログラミングからテキストを用いたプログラミングへの橋渡しとして，初学者向けプログラミング学習環境である PEN のソースコードをフローチャートとして生成する PenFlowchart を開発し，高等学校の授業で運用した．その結果，PenFlowchart の導入により，フローチャートやプログラムの構造を把握する力が向上する可能性が確認された [9]．末吉らは，VPL からテキストベースプログラミング言語への移行を支援するシステム BlockCode を開発した [10]．このシステムは Web から利用でき，ブロックベースのブロックを使ってプログラムすると，それが C 言語のプログラムに変換されたものが表示される．松澤らは，VPL からテキスト記述言語へのシームレスな移行を目的とした開発環境である BlockEditor を提案した．BlockEditor はブロックを組み合わせて作成したプログラムを Java プログラムに変換できる．これを文化系大学生向けのプログラミング入門教育で運用した結果，プログラミング学習が進行するにつれ，BlockEditor から Java へ徐々に移行していくことや，プログラミングに苦手意識をもつ学生ほど VPL の選択率が高いことがわかった [11]．

又吉らは，プログラミング時のタイピングに着目し，用意された課題のプログラムをタイプすることで，逐次入力した行が実行され出力が閲覧できる学習システムを提案した [12]．このシステムを演習型プログラミング授業に導入した結果，ユーザのタイピング速度のばらつきが減り，タイピング速度が向上していることがわかった．

しかし，プログラム課題に対しどう解決方針を立てるかや解決方針をどのように実装していくのか，といった支援を行う学習システムは筆者らの知る限り存在しない．本研究では，プログラム課題への取り組みを「課題の解決方針を立案すること」と「解決方針をコーディングすること」に分離し，前者のみに専念させることで高速な試行錯誤を可能にし，初学者のプログラム課題の解決方針を立てる能力を高める．

3 提案システムのプロトタイプ

本章では，プログラミングを「課題の解決方針を立案すること」と「解決方針をコーディングすること」に分離し，前者のみに専念させることで高速な試行錯誤を可能にし，プログラム課題の解決方針を立てる能力を高めるための学習システムのプロトタイプについて説明する．本論文における解決方針の立案とは，プログラム課題を解決するためにまず何をするかといった大まかな方針と，それを表現する手順・アルゴリズムを考えることである．

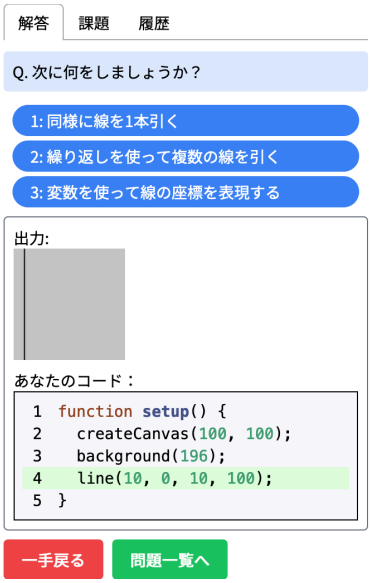


図 1: 提案システムの画面

3.1 設計指針

プログラム課題の解決方針の立案能力を高めるシステムの実装のため，4 つの設計指針を設けた．

解決方針を気軽に試せる

プログラム課題は「解決方針を思いついてもそれをコーディングできないので正誤がわからない」「実装プログラムのエラーが解決できずつまずいてしまう」などの障壁により，初学者が思いついた解決方法を試行錯誤することが難しい．よって選択式で解決方針を選択するのみでプログラムが自動で実装されるよう設計し，容易かつ高速に試行錯誤を繰り返すことができるように設計した．

また選択肢が多すぎたり，学習システムの利用ハードルが高いと手軽に試行錯誤ができず，継続的な使用も見込めない．よって表示する選択肢は常に 3 個以下とし，スマートフォンでのタップ操作のみで課題に取り組めるよう設計した．

解決方針を段階的に実装することを意識させる

プログラムを書く際，例えば for 文で同じ処理を繰り返す場合でも，まずループ変数や条件式などを記述し，次にループ変数を利用して処理内容を記述する，といったように複数の小さなステップに分解できる．ある解決方針を考えた際に，それが複数のステップによって構成されており，その 1 つ 1 つを実装していくことで解決方針を実現できる，ということを体験させるために，表示する選択肢は複数の小さなステップに分かれるよう設計した．

過去の選択の振り返りを促す

プログラム課題を実際にプログラムを記述して解いた際，プログラムの流れと課題解決のための思考過程は必ずしも同じとは限らないため，どのような思考過程で解答に至ったかを振り返ることは初学者にとって難しいと考えられる．よって，課題に取り組む過程や課題解決時に思考過程を振り返ることを促すよう設計した．

実際のコーディングと段階的に接続する

提案システムの目的は解決方針を立てる能力を高めることであるが，これができるだけでは実際にコーディングして課題を解くことに繋がらない可能性がある．よって，ユーザが解決方針のみを選ぶだけでなく，どのようなプログラムの実装になるかを確認させるフェーズを設け，提案システムを使用せずにコーディングするための足場かけとなるように設計した．

3.2 対象とするプログラミング分野

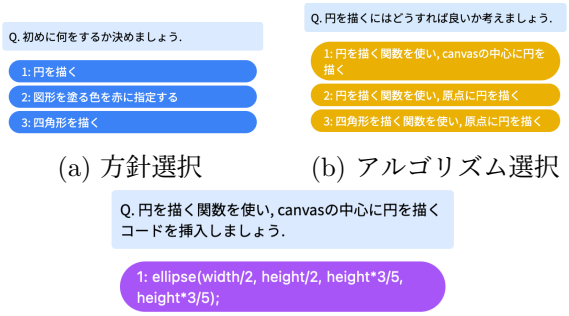
提案システムは，初学者向けのプログラミング学習での利用を想定している．昨今では，初学者向けに Processing [13] や p5.js [14] といった視覚的な出力を用いたテキストベースのプログラミング環境を用いることが増えている [15]．提案システムは，プログラミング言語 JavaScript とその描画ライブラリである p5.js を利用したプログラミング講義での課題を解く際に，スマートフォンを利用して使用されることを想定している．例えば p5.js を使って日本国旗を描くプログラムは，プログラム 1 のようになる．

プログラム 1: p5.js のプログラム例

```
1 createCanvas(180, 120);
2 background(255);
3 fill(255, 0, 0);
4 noStroke();
5 ellipse(width/2, height/2, height*3/5,
      height*3/5);
```

3.3 ユーザインタフェース

提案システムの利用時の画面を図 1 に示す．はじめに，ユーザは Web ブラウザから指定の URL にアクセスし，取り組む課題を選ぶ．そして表示される選択肢をタップ操作で選択することで課題を解いていく．図 1 の画面は，現状のユーザのコードとその出力画像，選択肢からなる「解答タブ」，課題が表示される「課題タブ」，ユーザがこれまでに選択した過去の選択肢を閲覧できる「履歴タブ」が表示されており，それぞれのタブをタップすることで対応した内容を閲覧できる．ユーザが取り組む課題を選ぶとその解決方針が選択式で表示され，タップで選



(a) 方針選択　(b) アルゴリズム選択

(c) コード確認

図 2: 解答タブのフェーズ

択肢を選んでいくことで，それを実現したコードがテキストエディタに追記される．これを繰り返すことで，ユーザはプログラムの実装をせずとも課題の解決方針のみを考え，選択するだけで手軽にさまざまな解決方針を試すことができる．また，誤りの選択肢を選んでしまったり，自分の解決方針に違和感を感じたりした場合は「一手戻る」ボタンをタップし，前の選択肢に戻り選択をやり直すことができる．

解答タブ

このタブに各設問の選択肢が表示され，ユーザはそれらをタップすることで課題を解いていく．各設問は 3 つのフェーズ「方針選択」「アルゴリズム選択」「コード確認」に分かれており，これらのフェーズを繰り返すことでユーザは最終的な解答を完成させる．それぞれのフェーズの画像を図 2 に示す．

1 つ目の「方針選択フェーズ」 (図 2(a)) では，ユーザは課題の解決のためにどのようなアプローチを取るかを選択する．例えば，「円を描く」，「繰り返しを使って複数の線を引く」などである．選択肢は青色で表示される．

2 つ目の「アルゴリズム選択フェーズ」 (図 2(b)) では，選択した課題解決のためのアプローチをプログラムするための手順やアルゴリズムを選択する．例えば，「円を描く関数を使い，canvas の中心に円を描く」「ループ変数を使って 9 本の線を引く」などである．選択肢は黄色で表示される．

3 つ目の「コード確認フェーズ」 (図 2(c)) では，アルゴリズム選択フェーズで選択したコードを確認し，それを選択するとテキストエディタにコードが挿入され，ハイライトされる．このフェーズにより，立案した解決方針を実際にコーディングするとどのようなコードになるのかをユーザに確認させる．選択肢は紫色で表示される．

方針選択フェーズとアルゴリズム選択フェーズは，選択肢をタップして高速に試行錯誤することで，解決方針を立てる能力を高めるためのフェーズであり，コード確認フェーズは，実際のコーディングへの接続を意図したフェーズである．

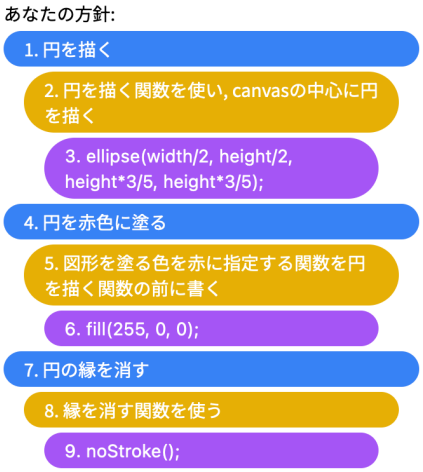


図 3: 解決方針の例

課題タブ

このタブには取り組んでいる課題の説明文と再現する対象の画像が表示される．また，現状のユーザの方針選択により作成されたプログラムの出力も表示される．

履歴タブ

これまでにユーザが選択した選択肢の一覧が図 3 のように表示される．それぞれのフェーズの選択肢が色とインデントで階層的に表示されている．

3.4 実装

提案システムは，Web アプリケーションフレームワークである Next.js [16] を使い，Web アプリケーションとして実装した．選択肢は JSON 形式で表現されており，プロトタイプの実装では第一著者が手作業で用意した．

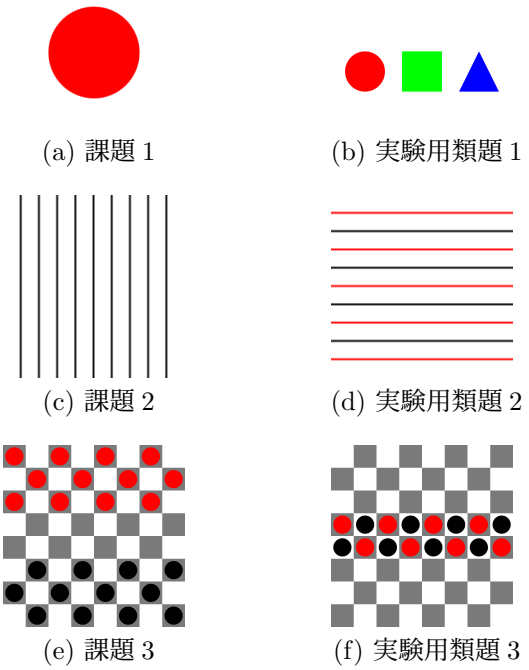
4 予備実験

提案システムのプロトタイプの利用により，学習者のプログラム課題の解決方針の立案能力にどのような効果があるかを調査するための実験を行った．

4.1 実験内容

実験の被験者は 10 から 20 代の学生 8 名 (女性 1 名, 男性 7 名) である．被験者のプログラミング経験に関するアンケートの結果，「全くない」が 1 名，「独学や授業での学習経験あり」が 3 名，「変数や繰り返しなどの基礎的な概念を使ったプログラムが実装できる」が 3 名，「ソフトウェアを開発したことがある」が 1 名であった．

実験手順は，まずプログラミングに関する基礎概念の理解と確認のため，プログラミング初学者向け



(a) 課題 1　(b) 実験用類題 1

(c) 課題 2　(d) 実験用類題 2

(e) 課題 3　(f) 実験用類題 3

図 4: プロトタイプで用意した課題と実験用類題で用いた画像

講義のテキストを読ませ，プログラム課題を解くのに必要な知識を学習させる．そして，提案システムのプロトタイプを使用させ，用意しているプログラム課題 3 問に取り組ませる．取り組むプログラム課題の内容は，図 4(a), 4(c), 4(e) の画像をプログラムで再現するというものである．その後に提案システムの所感についてアンケート調査を行ったのち，プログラム課題の立案方針を問うテスト課題に取り組ませた．この課題は提案システムで取り組んだ課題の類題 3 問である．類題の画像を図 4(b), 4(d), 4(f) に示す．また，提案システムはコーディングそのものの支援は行っていないため，被験者には課題をコーディングして解かせるのではなく，日本語で解答方針のみを箇条書きで記述させた．なお本実験は，神戸大学大学院工学研究科人を直接の対象とする研究倫理審査委員会の承認を受けて行ったものである (審査番号 05-18).

4.2　実験結果と議論

テストの解決方針

類題のテストで得られた解決方針の例を以下に示す．解決方針例 1, 2, 3 はそれぞれ別の被験者によるものである．このように粒度に違いはあるものの，多くの被験者が解決方針を日本語で段階的に記述できていることがわかった．解決方針例 1, 2 のように日本語のみで完結している解答もあれば，解決方針例 3 のように for 文などのプログラムの構文を用いた解答もみられた．

解決方針例 1（類題 1）
・塗る色を赤にする
・円を描く関数を座標と共に使って左側に描く
・塗る色を緑にする
・四角を描く関数を座標と共に使って中央に描く
・塗る色を青にする
・△を描く関数を座標と共に使って右側に描く

解決方針例 2（類題 2）
1. 1 本の線を描画する
2. 繰り返し文を利用して線を 9 本に増やす
3. 条件分岐を利用して線の本目が奇数ならば赤色，偶数ならば黒色に線を塗るように指定する

解決方針例 3（類題 3）
・`for` 文でループを作成（`i=0` とし，`i` が 8 より小さい間，`i` を 1 ずつ増やす）
・作成した `for` 文の中にもう一つ `for` 文を作成（`j=0` とし，`j` が 8 より小さい間，`j` を 1 ずつ増やす）
・`i+j` を 2 で割ったあまりが 1 のとき，図形の色を指定（グレー），正方形を作成（x 軸が `i`，y 軸が `j`），正方形の枠線を削除
・`i+j` を 2 で割ったあまりが 1 かつ，`j` が 2 より大きく 5 より小さい時，図形の色を指定（赤），円を作成（x 軸が `i`，y 軸が `j`），図形の枠線を消す
・`i+j` を 2 で割ったあまりが 0 かつ，`j` が 2 より大きく 5 より小さい時，図形の色を指定（黒），円を作成（x 軸が `i`，y 軸が `j`），図形の枠線を消す

タップ操作のログ

被験者が提案システムの利用時に行った操作のログを分析した結果について述べる．被験者全体のタップ操作をみると「一手戻る」ボタンが平均 14.6 回使用されており，被験者が複数の選択肢を選んで試行錯誤してたことがわかる．履歴タブの参照回数は平均 3.6 回であり，どの被験者も解答の過程で何度が方針を振り返っていることが確認できた．また課題タブと解法タブの参照回数がそれぞれ平均 20.0 回，15.1 回であり，2 つのタブを往復して課題を確認しながら解いていく様子がみられた．

表 1: 実験のアンケート結果 (1: 全くそう思わない, 5: とてもそう思う)

コード	質問	AVE	SD
Q1	プログラム課題の解決方針を立てる力が向上したと感じるか	3.5	0.93
Q2	各問題の選択肢を気軽に試すことができたか	4.4	0.92
Q3	授業等のプログラム課題と比べて気軽に取り組めたか	4.6	0.52
Q4	各問題を解く方針を段階的に考えることができたか	4.5	0.53
Q5	各問題について，選択肢の数や粒度は適切だと感じたか	3.8	1.3
Q6	アプリは使いやすかったか	3.0	0.53
TQ1	テスト課題を解く際に段階的に考えることができたか	4.3	0.46
TQ2	テスト課題を解く際に提案システムの影響があったと感じるか	4.9	0.35

アンケート結果

実験で得られた提案システムのプロトタイプに関するアンケートの結果を表 1 に示す．Q1 の結果より，提案システムを利用することでプログラム課題の解決方針を立てる能力を高められる可能性がある．自由記述からは「日本語で次に何をすべきかを考えることが『解決方針を立てる』という面でみれば，プログラミングスキルの優劣にかかわらず役に立つのではないかと感じた」といったコメントが得られた．一方で「このレベルの問題は C 言語で何度もやってきたので，特に能力が向上したとは感じなかった」というコメントも得られたため，他のプログラミング言語の経験があり，解決方針の立案能力がすでに高い場合は効果が得られない可能性がある．Q2 と Q3 の結果より，高速な試行錯誤を促進できる可能性がある．さらに Q4 の結果より，スモールステップでの問題解決を促進できる可能性がある．

また Q5, Q6 の結果より，ある程度プロトタイプは使いやすく，手軽に利用できる可能性がある．しかし，UI に関しては「最初は問題がどこにあるのかわからず困ったので，課題と解答のタブが左右逆になっていた方がやりやすいと思う」などのコメントが得られたため，画面に表示する情報を整理し，使いやすさを向上させる必要がある．

なお，テスト課題に関するアンケートである TQ1 と TQ2 の結果から，提案システムを利用することで，プログラム課題を解く際に段階的に解決方針を立てられるようになる可能性がある．また「目標物が段階的に作り上げられている過程を見ることができるのは良いと思った．コードを選択肢で選びつつ文字でも確認できるのは初学者には嬉しいのではないかと思った」「Web アプリで学んだプログラムの構成の手順に関する知識を活かして解答出来たと感じた」「プログラムを始める前に，いったん頭を整理しようという段階が明確に意識された気がした」「実際にあったら使ってみたい．普段無意識でやりがちな，完成に向けての各工程の言語化の大切さをわからせてくれるアプリだなと思った」などのコメントが得られ，課題の解決方針を設計・分割してから取り組むことを促進できる可能性がある．しかし，今回の実験は 1 時間程度の短期間の実験であり，取り組んだ課題もプロトタイプで取り組んだ課題の類題であるため，提案システムの効果を検証するためにはより被験者数を増やし，長期的な実験を行う必要がある．

5 今後

実験結果を踏まえた提案システムの今後の方向性について述べる．まず，課題の作成方法についてである．今回の提案システムのプロトタイプでは，3 問の課題の選択肢を第一著者が手作業で用意したが，実際のプログラミング演習講義では多くの課題が出題され，すべての課題の選択肢を教授者が手作業で用意するのは現実的ではない．よって，今後は課題の選択肢を半自動的に作成できるエディタの実装や，ChatGPT [17] などの大規模言語モデルを利用した選択肢の自動生成を検討する．

そして，今回の実験は数名の学習者を対象に提案システムのプロトタイプを利用させることにとどまったため，プロトタイプを改善し，より多くの初学者を対象に長期的な実験を行う．具体的には，神戸大学国際人間科学部の講義「プログラミング基礎演習 1」の予習課題に提案システムを導入し，受講者たちの課題への取り組み方の変化を調査する．

6 まとめ

本研究ではプログラミングを「課題の解決方針を立案すること」と「解決方針をコーディングすること」に分離し，前者のみに専念させることで高速な試行錯誤を可能にし，初学者の課題解決方針を立てる能力を高める選択肢タップ式学習システムを提案した．提案システムのプロトタイプを，プログラム課題の解決方針を選択肢タップにより手軽に試すことができる Web アプリケーションとして実装した．プロトタイプの予備実験の結果，提案システムを利用することで課題解決方針を立てる能力を高められる可能性があるとわかった．今後はプロトタイプを改善し，より多くの初学者に対し長期的な実験を行う．

謝辞

本研究の一部は，JST CREST(JPMJCR18A3) および JST 科学技術イノベーション創出に向けた学フェローシップ創設事業 JPMJFS2126 の支援によるものである．ここに記して謝意を表す．

参考文献

[1] 文部科学省: 小学校段階におけるプログラミング教育の在り方について（議論の取りまとめ), https://www.mext.go.jp/b_menu/shingi/chousa/shotou/122/attach/1372525.htm (accessed on 30 October 2023).

[2] J. M. Wing: Computational Thinking, *Communications of the ACM*, Vol. 49, No. 1, pp. 33–35 (2008).

[3] S. R. M. Derus and A. Z. M. Ali: Difficulties in Learning Programming: Views of Students, *Proc. of the 1st International Conference on Current Issues in Education (ICCIE 2012)*, pp. 74–78 (Sept. 2012).

[4] E. Lahtinen, K. Ala-Mutka, and H. Jarvinen: A Study of the Difficulties of Novice Programmers, *Proc. of the 10th Annual SIGCSE Conference on Innovation and Technology in Computer Science Education (ITiCSE 2005)*, pp. 14–18 (June 2005).

[5] M. N. Ismail, N. A. Ngah, and I. N. Umar: Instructional Strategy in the Teaching of Computer Programming: A Need Assessment Analyses, *Turkish Online Journal of Educational Technology (Turk. Online J. Educ. Technol.)*, Vol. 9, No. 1, pp. 125–131 (2010).

[6] Progate: https://prog-8.com/ (accessed on 30 October 2023).

[7] ドットインストール: https://dotinstall.com/ (accessed on 30 October 2023).

[8] M. Resnick, J. Maloney, A. Monroy-Hernández, N. Rusk, E. Eastmond, K. Brennan, A. Millner, E. Rosenbaum, J. Silver, B. Silverman, and Y. Kafai: Scratch: Programming for all, *Communications of the ACM*, Vol. 52, No. 1, pp. 60–67 (2000).

[9] 中西 渉, 辰己丈夫, 西田知博: PenFlowchart によるプログラミング導入教育の評価, 研究報告コンピュータと教育 (CE), Vol. 2013, No. 9, pp. 1–7 (Oct. 2013).

[10] 末吉春一, 佐藤 喬: ビジュアルプログラミングを用いたテキストベースプログラミング学習支援システム, 第 78 回全国大会講演論文集, pp. 897–898 (2016).

[11] 松澤芳昭, 酒井三四郎: ビジュアル型言語とテキスト記述型言語の併用によるプログラミング入門教育の試みと成果, 情報処理学会研究報告コンピュータと教育 (CE), Vol. 2013, No. 2, pp. 1–11 (Mar. 2013).

[12] 又吉康綱, 中村聡史: typing. run: 初学者のプログラミング学習を支援するプログラムタイピングシステムの提案と実践, 情報処理学会, Vol. 2020, No. 1, pp. 1–8 (Aug. 2020).

[13] Processing, https://processing.org/ (accessed on 30 October 2023).

[14] p5.js, https://p5js.org/ (accessed on 30 October 2023).

[15] 三好きよみ: Processing による初学者向けプログラミング教育の実践, 情報教育シンポジウム論文集, Vol. 2020, pp. 89–95 (Dec. 2020).

[16] Vercel: Next.js, https://nextjs.org/ (accessed on 30 October 2023).

[17] OpenAI: ChatGPT, https://chat.openai.com (accessed on 30 October 2023).