Vol.2018-CE-143 No.6
2018/2/17

プログラミング初学者にむけた アルゴリズム的思考習得のための学習支援手法

鈴木 優実[1,a)]　松村 敦[2,b)]　宇陀 則彦[2,c)]

概要：2020 年から初等・中等教育でのプログラミング必修化が決定され，プログラミング初学者への教育の重要性が増している．プログラミング教育では，文法，アルゴリズムなど様々な要素を学習する必要があるが，初学者はプログラムの構成段階で行き詰まってしまう問題を抱えている．これに対処するため，プログラムの構成を考える手法のひとつであるアルゴリズム的思考法を用いた学習システムを提案した．大学生 10 名を対象とした評価実験を行い，一定の支援効果がみられた．

Learning support method for acquiring algorithmic thinking for programming beginners

Yumi Suzuki[1,a)]　Atsushi Matsumura[2,b)]　Norihiko Uda[2,c)]

1. 序論

2020 年度に改訂される小学校学習指導要領で初等教育にプログラミング教育を導入することが必修化され，プログラミング初学者への教育の重要性が高まってきている．高等教育機関である大学では，既にプログラミング教育が行われており，数多くの先行研究が存在するが，プログラミング初学者は多様な問題を抱えており，教育の難しさが課題となっている．

授業等で行われるプログラミングでは，プログラミング言語，アルゴリズム，プログラム構築技術など，異なる要素を並列に指導する [1] ．しかし，各要素の内容が理解できていても，それぞれをどのように紐付けてプログラムとして表現すれば良いか分からず，プログラムの構成段階で行き詰まってしまう問題を抱える学習者がいる [1].

久野は，実際に学校でプログラミング教育としてよく行われているが，うまくいっていない例として，「プログラミング言語の文法や機能を逐一，順番に説明していく」，「1 つの例題を選んで丁寧に説明し，試験もその例題から出す」，「例題をその通りに打ち込んで動かすことにだけ注力する」の 3 点を指摘している [2]．授業でのプログラミングでは，言語仕様や文法を覚えることから始まり，様々な例題プログラムを実行することで学習を進めるケースが多々ある．しかしながら，これでは暗記学習となってしまい，自分でプログラムを構成する力は身につかない．この問題に対処するため，プログラムの構成を考えさせるアルゴリズム的思考法や段階詳細化法を取り入れた教育が実践されている [1], [2], [3], [4], [5], [6], [7], [8].

飯田らは，機械的な実行ができると，実行に頼ってアルゴリズムの設計をしたり，その理解をしようとすることを指摘しており [3]，プログラムが機械的に実行できる環境では，その構成を考える力が不十分なまま育たない可能性が考えられる．そこで，本研究では実行環境に頼らず，プログラムの構成を考える段階に特化したシステムが必要であると考えた．特に「プログラミング時における思考の段階」に焦点を当て，アルゴリズム的思考法を用いて段階的に思考の整理を助けることで初学者の学習支援を目指す．具体的には，学習システムの開発を行い，システムを使用して学習することでプログラムの処理の流れや構成を考えることができると考えた．

1　筑波大学 情報学群 知識情報・図書館学類
2　筑波大学 図書館情報メディア系
a)　s1411539@klis.tsukuba.ac.jp
b)　matsumur@slis.tsukuba.ac.jp
c)　uda@slis.tsukuba.ac.jp

本稿の構成を以下に示す．2 節ではプログラミング教育の手法の先行研究述べ，初学者への学習支援としてプログラムの構成を考えさせることが重要であることを示す．これを踏まえ，3 節で本研究の手法を提示し，4 節で学習システムについて説明する．5 節では評価実験について説明し，6 節でその結果と考察を示し，最後に 7 節で本論文の結論を述べる．

2. 先行研究

1 節で述べた通り，プログラムの構成を考えさせる研究は既にいくつか存在している．本節では，先行研究の特徴や改善点をふまえて本研究の手法を検討する．

飯田らは，問題を解決する際に指針とすべき考え方の一つで，「基本的な操作に分解しそれらの順序を意識する」ことをアルゴリズム的思考法と定義した [3]．また，アルゴリズム的思考法を学ぶ際にコンピュータを用いることが好ましくない理由として，文法への正確な適合が求められるためにプログラム言語自体の文法をある程度覚えてからでないと学習ができない環境があること，機械的な実行ができると実行に頼ってアルゴリズムの設計をしてしまうことを挙げている．以上の問題に対し，紙の上で書かれたものならば，多少の文法の揺れがあっても人間が理解できる範囲を限界として許容されること，フローチャートの場合は頭での理解がより求められることを理由に「アルゴリズム的思考法」を用いたカリキュラムの実践を行い，その有用性を示した．

飯田らは，暗記型学習と思考型学習を比較した [4]．ここでは暗記型学習でのアルゴリズム学習には限界があると示し，思考型学習へ移行させるサポートが必要であると述べている．授業では時間に制限があるという問題もあり，途中解答や誤答，無理やり辻褄を合わせた解答になることも多い．しかし，自ら思考し，試行錯誤を重ねて答えを導き出すことが重要であり，最初から簡潔には書けないことを知ってもらう必要があると述べている．また，初学者特有の問題として，問題文から過去に解いた問題を思い出し，それらの組み合わせで問題に対処しようとすることを挙げている．このような記憶に頼る学習は初期の段階では有効であるものの，徐々に対応できない問題が増えてしまう．飯田らは，学生の回答が正解でなくても，その考え方の延長線上に答えがあるという実感を持たせる必要がある，と述べている．

板戸らは，近年のプログラミング学習の問題点として，学習者が処理の流れを考えずにソースコードを作成してしまうことを挙げている [7]．プログラミングに必要な能力は，文法知識，コンピュータの知識，制御構造に落としこむ力の 3 つであるとし，制御構造に落としこむ力を，「処理の流れを考える力」（機能分割能力）と「ソースコードに落としこむ力」（表現力）に分割して評価する手法を提案している．これにより，学習者がどの段階で行き詰まったのかを把握できるため，指導を効率よく行えると述べている．また，擬似言語やフローチャートを用いる手法では，新たに表記法を習得する必要があり，新規学習者への負担が大きいことを指摘している．板戸らの作成したシステムでは，学習者はまず機能分割能力を判別する日本語の並び替え問題を解く．その後，正解した学習者は表現力を判別する穴埋め問題を解く．解答するにあたり，指導を必要とする間違いをしたかを判定して学習者を分類した．

以上から，初学者への学習支援として，処理の流れを考えさせることが重要であると示された．「考える」ことを経験させることが大切であるため，繰り返し実践することが必要である．そして，飯田らの指摘 [3] から，本研究では，実行環境に頼らず，プログラムの構成を考える段階に特化した教育が必要であると考えた．しかしながら，授業での実践では個人に合わせた教育や，出来ないところだけを繰り返すといったことが難しい．そこで，システム上での学習を提案する．学習をシステム上で行うことで，授業での学習と異なり，時間に縛られず，個人に合わせた学習ができるのではないかと考えた．システム化の試みはあるものの，板戸らは新たに表記を覚える必要がある手法では初学者への負担が大きいと指摘している．本研究では，新たな表記法を覚える必要がない学習の負担が少ないシステムを提案することでこの問題に対処する．

3. 提案手法

プログラミングを授業などで行う場合，問題文を読み解いてソースコードにするが，実際には「問題文」と「ソースコード」の間には「構成要素」を考えるプロセスがあるのではないかと考えられる．「構成要素」のプロセスは問題を解く際に処理の流れを確認し，その順序を考えるプロセスである．本研究では，このプロセスを学習させることで初学者支援ができると考えた．

プログラム作成の際，その構成を十分に考えられる人は問題文の言い回しから必要な要素を発見し，それらを満たす機能の作成とデバッグを繰り返すことによって，プログラムに不足がないかを確かめていると考えられる．反対につまずいてしまう人は，「問題文をそのままコードにしようとする」「なぜそのように修正するのかを理解していない」といった問題により，細かいタスクを適切な順番で実行することにより大きい処理を実現するということに慣れておらず，その概念や技術を習得することが一つの壁となっている [9]．構成を考えられる人とつまづいてしまう人の違いである「構成要素」のプロセスを学習させ，構成を考えられる人に近づけることで，プログラム作成時の手詰まりを軽減できる可能性があるのではないかと考えた。そして，この構成要素プロセスへの支援にアルゴリズム的思考法を利用した．アルゴリズム的思考法の基本的な操作・順序を

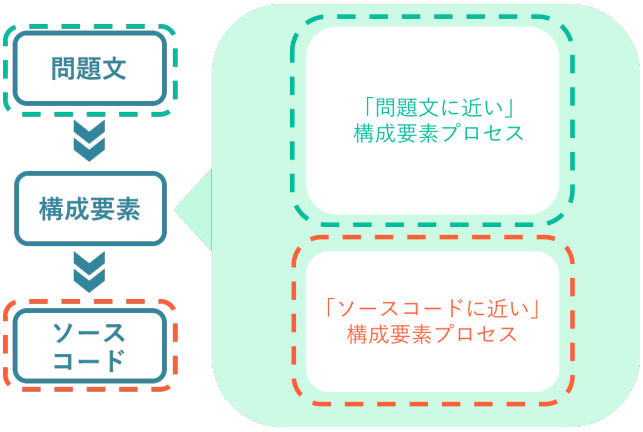

図 1 提案手法

意識する，という部分が構成要素を考えるプロセスに該当すると考えたためである．

1 に示す通り，問題文とソースコードの間には「構成要素」を考えるプロセスがあるとしたが，このプロセスにも段階があると考えられる．ひとつは「問題文に近い，構成要素プロセス」である．これは，短い文章の組みあわせによって処理手順をまとめた，自然文に近い表現でプログラム構成を表すプロセスである．もうひとつは「ソースコードに近い，構成要素プロセス」である．これは，より抽象化した処理手順でプログラム構成を表すプロセスである．

以上のように，まず大枠でプログラムの流れを捉え，次に具体的な手順を考える，ということを繰り返すことによって初学者が問題文からプログラムを構成する力をつけることができるのではないかと考えた．

4. 学習システムの開発

初学者のプログラミング時における思考の整理を助けるため，プログラムの構造や流れを考える学習システムを考案し，構成要素のプロセスを 2 段階で支援するための機能を実装した．システムを使用して学生が自主学習をすることを想定したため，環境に左右されにくい，ブラウザ上で動作する Web システムとして開発を行った．

学習システムは，プログラミング時における思考の整理を助ける機能として「ブロックの並び替え機能」(2)，「穴埋め問題機能」(4) の 2 つを有しており，ひとつの問題に対して両機能を使って学習支援を行う．「ブロックの並べ替え」では，問題文から抽出した要素がブロックとして表示され，並べ替えるだけで簡単に流れ図をつくることができる．「穴埋め問題」では，流れ図を見ながら選択肢形式の穴埋め問題を解く．以上の 2 つの機能を利用して問題を繰り返し解くことにより，問題文とブロックの関係や抽出パターンを見つけ，思考の整理をする事ができると考えた．

以下，それぞれの機能について説明する．

図 2 ブロックの並び替え機能

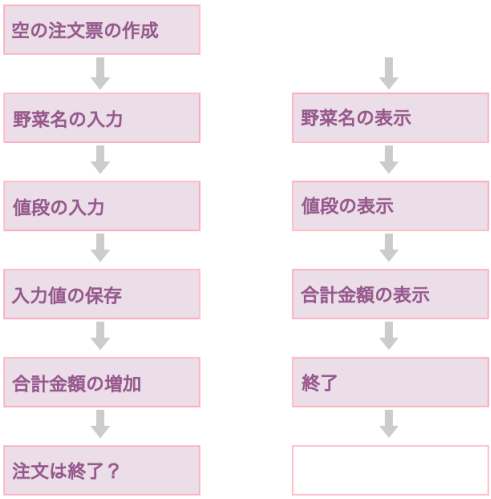

図 3 流れ図の作成

4.1 ブロックの並び替え機能

2 に示すように，ブロックの並び替え機能は，学習システム上部のメニューにある「ブロックの並び替え▼」をクリックすることで切り替えることができる．その際，プルダウンメニューで問題番号が示され，解く問題を選択する必要がある．機能切り替え後は画面左側上部に「問題文」を，画面左側下部に問題文から抽出した要素を「ブロック」として表示している．画面右半分は流れ図作成のためのテンプレートであり，枠線と矢印を表示している．ユーザーはブロックをマウスでドラッグし，テンプレートの枠の上に自由に動かすことで簡単に流れ図をつくることができる．

3 はブロックを並び替えて流れ図にしたものである．

4.2 穴埋め問題機能

4 に示すように，穴埋め問題機能は，学習システム上部のメニューにある「穴埋め問題▼」をクリックすることで切り替えることができる．その際，プルダウンメニューで問題番号が示され，解く問題を選択する必要がある．穴埋め問題機能では，画面左側上部に「問題文」を，画面左側下部に「ブロックの並び替え機能」で作成した流れ図を表

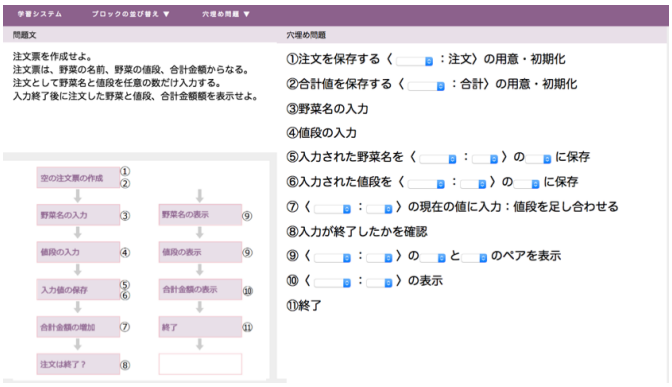

図 4 穴埋め問題機能

示している．穴埋め問題の番号は流れ図の番号と一致している．穴埋め問題の空白部分はプルダウンメニューになっており，解答は記入式ではなく選択式となっている．

以上の 2 つの機能を利用して問題を繰り返し解くことにより，「問題文」と「ブロック」の関係や抽出パターンを見つけ，思考の整理をする事ができると考えた．

5. 評価実験

5.1 実験概要

本システムの効果を検証するため，大学生 10 名を対象に評価実験を行った．10 名の内訳は，筑波大学 知識情報・図書館学類の 1 年生 9 名，3 年次編入生 1 名である．1 年生は必修科目にプログラミングの授業があるが，大学で初めてプログラミングに触れる人が多い．また，短大等からの 3 年次編入生もプログラミング経験が無い場合，プログラミングの授業を受ける必要がある．以上のようなことから，今回対象とした 1 年生，3 年次編入生はプログラミング初学者であり，支援効果が高いと考えられる．

実験内容は，実験参加者がプログラミングの問題をシステムを用いて 3 問解答するものである．

実験は 2017 年 11 月 17 日から 12 月 7 日の間に実施した。

5.1.1 実験手順

1). 事前アンケート

実験を行う前にプログラミングの苦手度や，普段どのようにプログラムの構成を考えているかについて尋ねた．アンケートは Web 上で回答してもらった．

事前アンケートでは以下の項目を尋ねた。

Q1 プログラミング演習の受講以前にプログラミング経験があったか

Q2 プログラミングが好きか

5 段階で解答する設問である。 1 に近いほどプログラミングが好きであり，5 に近いほど嫌いである。

Q3 プログラミングの得意度はどのくらいか

5 段階で解答する設問である。 1 に近いほどプログラミングが得意であり，5 に近いほど苦手である。

Q4 プログラミングの問題を解く際に難しいと感じる点

以下（a）～（e）の項目から選択して解答する設問である。（複数回答可能）

（a）問題から処理を組み立てること
（b）プログラミング言語での記述方法
（c）プログラミングに関連する概念を理解すること
（d）特にない
（e）その他（自由記述）

Q5 プログラミングの問題を普段どのように解いているか

以下（a）～（d）の項目から選択して解答する設問である（複数選択可）

（a）プログラミング言語でプログラムを記述しながら処理の流れを決める
（b）フローチャート等を用いて最初に処理の流れを決め，プログラミング言語で記述する
（c）要件ごとにプログラムを作り，後でそれぞれを合わせてひとつのプログラムにする
（d）その他（自由記述）

Q6 プログラミングの問題を解く際，処理の順番や流れを考える事を意識しているか

5 段階で解答する設問である。1 に近いほど処理の流れを意識しており，5 に近いほど意識していない。

Q7 5 で答えた理由

以下（a）～（e）の項目から選択して解答する設問である（複数回答可）

（a）プログラミングの問題を解きやすいから
（b）プログラムを書きやすいから
（c）自然とそのように解くようになった
（d）考え方が分からないから
（e）その他（自由記述）

2). 実験説明

実験参加者が使用する学習システムの操作方法や各機能についての説明を行った．それぞれの問題につき「ブロックの並び替え機能」と「穴埋め問題機能」の両機能を使うこと，「ブロックの並び替え機能」で問題を解いてから「穴埋め問題」機能を使うことを確認した．

3). 実験タスク

実験タスクはプログラミングの問題をシステムを用いて 3 問解答するもので，それぞれの問題につき「ブロックの並び替え機能」と「穴埋め問題機能」の両機能を使う．機能別に各問題が解けた時点で「解けた」と合図してもらい，解答時間を計測した．「ブロックの並び替え」は 10 分，「穴埋め問題」は 5 分の時間制限を設けているが，それより早く終わった場合には次の問題へ移って良いこととした．

実験タスクには，実際に過去のプログラミング演習で取り扱った問題を使用した．(5)（6）(7) 難易度別に

注文票を作成せよ。注文票は、野菜の名前、野菜の値段、合計金額からなる。注文として野菜名と値段を任意の数だけ入力する。入力終了後に注文した野菜と値段、合計金額を表示せよ。

図 5 問題 1

石取りゲームを作成せよ。石取りゲームとは、石の山から順番に石をとっていって最後の石を取った人が負けとなるゲームである。一度に取れる石の最大数は 5 である。また、石の山にある石の数、一度に取れる石の数は適宜決定せよ

図 6 問題 2

歩きながらモンスターを捕まえるゲームを作る。モンスターは名前と強さを値として持っている。モンスターが出現した時、投げられるボールが無ければプログラムを終了し、あれば捕獲を試みる。「捕獲を試みる」部分はメソッドとして以下のように定義する。(緑文字部分) モンスターの強さと 0〜2000 までの範囲の乱数を比べ、乱数の値の方が大きい場合、捕獲できる。捕獲できた場合はつかまえた旨のメッセージを表示しできなかった場合は逃げられた旨のメッセージを表示する。ボールを投げた場合はボールの数を減らし、残りボール数を表示する

図 7 問題 3

3 問用意し，問題 1 は授業の総まとめとして授業の最終回に取り扱った問題，問題 2 は授業の最終回後にレポート課題として課された問題，問題 3 は問題 2 と同様のレポート課題のうち，メソッドを使ってプログラムを考えよと指定されているものとした．様々な問題を用意することにより，どのような問題に対しても有効であるかを考えたためである．また，違いが現れやすくなると考えた．

4). 事後アンケート

タスク終了後は実験全体についてのアンケートと，システムのそれぞれの機能的についての 2 種類の事後アンケートを実施し，主観的な評価を行ってもらった．アンケートには，当てはまる項目を選ぶ設問と自由記述の設問を用意した．

システム全体について

思考プロセスの理解と整理ができていたかどうかをアンケートによって調査した。システム全体についての事後アンケートでは以下の項目を尋ねた。

Q1 問題を解くことは難しかったか

以下（1）〜（3）の項目から選択して解答する設問である。

（1）難しかった
（2）丁度良かった
（3）簡単だった

Q2 どのような点が難しかった（または簡単）だったか？

自由記述の設問である。

Q3 処理の流れを考えることを意識すると問題が解きやすくなったか？

以下（1）〜（4）の項目から選択して解答する設問である。

（1）解きやすくなった
（2）解きにくくなった
（3）特に変わらなかった
（4）どちらともいえない・わからない

Q4 問題を解く際，ヒントがあった方が良いか？

以下（1）〜（2）の項目から選択して解答する設問である。

（1）ヒントがあった方が良い
（2）ヒント無しで良い

Q5 選択肢を選んだ理由と，どのようなヒントが欲しいか？

自由記述で解答する設問である。

Q6 問題を解く時間について

以下（1）〜（3）の項目から選択して解答する設問である。

（1）短い
（2）適当
（3）長い

Q7 感想

自由記述で解答する設問である。

機能に関する事後アンケート結果

機能に関する事後アンケートでは以下の項目を尋ねた。

Q1 今回はブロックの内容が決まっていたが，自分でブロックの内容も決めたいと思ったか

以下（1）〜（3）の項目から選択して解答する設問である。

（1）はい
（2）いいえ
（3）わからない

Q2 それはどんな理由からか

自由記述の設問である。

Q3 ブロックの内容が空白の場合，自分で内容を考えることができたと思うか」以下（1）〜（3）の項目から選択して解答する設問である。

（1）はい
（2）いいえ
（3）わからない

Q4 それはどんな理由からか

自由記述で解答する設問である。

Q5 今回は選択肢のある穴埋め問題だったが，記入式にしたいと思ったか

以下（1）〜（3）の項目から選択して解答する設問である。

（1）はい

(2) いいえ
(3) わからない

Q6 それはどんな理由からか
以下 (1) 〜 (3) の項目から選択して解答する設問である。(複数回答可能)
1. システムのブロックに思うような内容のものがなかったため
2. 並び替えるだけではあまり思考の整理の助けにならないと感じるため
3. 他の解き方もしてみたいと思ったため
4. そのた

Q7 穴埋めが空白の場合、あなたは自分で内容を考えることができたと思いますか？
以下 (1) 〜 (3) の項目から選択して解答する設問である。
(1) はい
(2) いいえ
(3) わからない

Q8 よろしければ理由をお答えください
自由記述で解答する設問である。

Q9 システム機能のうち、役に立ったと思うものをお答えください (複数回答可能)
以下 (1) 〜 (2) の項目から選択して解答する設問である。
(1) 並び変え機能
(2) 穴埋め問題

Q10 システム機能のうち、役に立たなかったと思うものをお答えください (複数回答可能) 以下 (1) 〜 (2) の項目から選択して解答する設問である。
(1) 並び変え機能
(2) 穴埋め問題

Q11 よろしければ理由をお答えください
自由記述で解答する設問である。

5). インタビュー
アンケートの回答結果をみて，補足的な半構造化インタビューを行った．インタビューでは「システムでは予めブロックや選択肢を用意したが，自分で内容を考えられそうか？」「普段もこのような書き方をしているか？」等気になった点について質問を行った．

6. 結果と考察

実験結果として，実験参加者の事前アンケート結果（表 1），事後アンケート結果（表 2）（表 3），を以下に示す．10 名の実験参加者の結果は，被験者 ID によって管理した．

表 1 の Q1 の結果よりプログラミング経験があった人は 3 人、なかった人は 7 人だった．Q3 の結果より，3 が 2 人、4 が 5 人，5 が 3 人だった．以上から大学で初めてプログラミングに触れ、苦手意識がある人が多いことがわかる．

表 1 事前アンケート結果

ID	Q1	Q2	Q3	Q4	Q5	Q6	Q7
01	ある	2	3	(a)	(c)	3	(c)
02	ない	3	4	(a) , (c)	(a)	5	(c)
03	ない	3	4	(a) , (b) , (c)	(a)	3	(d)
04	ない	4	5	(a) , (b) , (c)	(b)	4	(d)
05	ある	2	4	(b) , (c)	(a) , (c)	1	(b) , (e)
06	ない	2	4	(b) , (c)	(b)	3	(b)
07	ない	2	3	(a) , (b)	(a)	2	(c)
08	ない	4	5	(a) , (b) , (c)	(a) , (c)	4	(d)
09	ある	3	4	(b) , (c)	(a)	3	(b)
10	ない	5	5	(a) , (b) , (c) , (e)	(a)	5	(d)
平均		3	4.1			3.3	

表 2 事後アンケート結果（実験全体について）

ID	Q1	Q3	Q4	Q6
01	(3)	(2)	(2)	(2)
02	(1)	(2)	(1)	(2)
03	(1)	(1)	(1)	(2)
04	(1)	(1)	(2)	(3)
05	(1)	(1)	(1)	(3)
06	(2)	(1)	(1)	(2)
07	(1)	(1)	(2)	(3)
08	(2)	(1)	(2)	(2)
09	(2)	(4)	(1)	(2)
10	(1)	(1)	(1)	(2)
平均	1.5			

表 3 事後アンケート結果（機能）

ID	Q1	Q2	Q3	Q5	Q6	Q7	Q9	Q10
01	(1)	(2)	(1)	(1)	(2)	(1)	無	(1) , (2)
02	(3)	(3)	(2)	(2)	無	(2)	(2)	無
03	(2)	(1)	(3)	(3)	(1)	(3)	(1) , (2)	無
04	(2)	(4)	(2)	(1)	(2)	(1)	(1)	(2)
05	(3)	(1)	(2)	(2)	無	(1)	(2)	無
06	(1)	(4)	(3)	(3)	記述	(2)	(1) , (2)	無
07	(1)	(1)	(1)	(1)	無	(1)	(1) , (2)	無
08	(2)	(4)	(2)	(2)	記述	(2)	(1) , (2)	無
09	(2)	(4)	(2)	(2)	記述	(1)	(1) , (2)	無
10	(2)	(4)	(3)	(2)	記述	(3)	(1)	(2)
平均		3	4.1		3.3			

実験終了後のアンケート（表 2）では，Q3 の処理の流れを考えると問題が解きやすくなったかという設問において，10 名中 7 名が解きやすくなったと回答し，2 名が解きにくくなったと回答した．システムの支援によって問題が解きやすくなったという回答が多く，学習システムに一定の支援効果があると考えられる．

さらに，解きやすくなったと答えた 7 人のうち，全員が役に立った機能が 1 つ以上あると回答しており，そのうち 4 名は両方の機能が役に立ったと回答している（表 3 の Q9）．しかし，役に立たなかった機能として「穴埋め問題」機能をあげている実験参加者が 2 名いた（表 3 の Q10）．自由記述やインタビューで理由を聞いたところ，「流れを考える」機能としての役割を果たせていないと感じていたユーザーであった．

以上から，初学者への一定の支援効果が見られたが，機能の見直しも必要であると考えられる．

また，問題が解きにくくなったと答えた 2 人は，その理

由としてインタビューで「普段の解き方と異なる」ことをあげている．本実験が想定した初学者よりもプログラムの構成能力があり，物足りなかった実験参加者や普段と異なる解き方をすることで戸惑いがあった実験参加者が居た．

以上から，提案手法でうまくいかない人たちへどう対応するかが課題となった．

7. 結論

本研究では，アルゴリズム的思考法をシステム化し，プログラミング初学者の学習支援を実現することを目的とした．開発した学習システムは，プログラミング時における思考の整理を助ける機能として「ブロックの並び替え」機能と「穴埋め問題」機能の 2 つを有しており，両機能を使って支援を行った．

システムの評価実験では，大学生 10 名を対象にプログラミングの問題を学習システムを用いて 3 問解答する実験を実施した．実験参加者には，それぞれの問題を「ブロックの並び替え」機能と「穴埋め問題」機能の両方を使って解答するよう指示し，タスク終了後はアンケートとインタビューを実施した．

実験終了後のアンケートでは，10 名中 7 名が理の流れを考えると問題が解きやすくなったと回答した．また，この 7 人のうち，全員が役に立った機能が 1 つ以上あると回答し，7 人中 4 人は両方の機能が役に立ったと回答している．以上から学習システムに一定の支援効果があると考えられる．しかし，解きやすくなったと回答した 7 人中 2 人が「穴埋め問題」機能を役に立たなかった機能としてあげた．また，そもそも問題が解きにくくなったと回答した実験参加者が 2 人いた．以上から，提案手法でうまくいかない人たちへどう対応するかが課題となった．

本手法では構成要素を 2 つの段階に分けたが，実際には更に詳細な段階があり，今回取り扱った 2 つの段階で考えることに馴染みのない人には，支援効果がうまく機能しなかったのではないかと考えられる．学習者の負担を減らすため，新たな表現形式を覚える必要がない，日本語のブロックを並び替えさせる機能を作成したが，解きにくくなったと回答した 2 人に対しては結果的に新しい形式を覚える負担をかけてしまったと考えられる．

今後は提案手法でうまくいかない学習者に対してのアプローチを考える必要がある．

参考文献

[1] 江木鶴子. プログラミング初心者を対象にしたデバッグ戦術の学習支援に関する研究. 九州工業大学, 2009, 87p. 博士論文. 入手先, 九州工業大学学術機関リポジトリ.
[2] 久野靖. プログラミング教育／学習の理念・特質・目標. 情報処理学会誌: IPSJ magazine. 2016, 57(4), p.340-343.
[3] 飯田周作ほか. アルゴリズム的思考法の教育. 情報処理学会研究報告コンピュータと教育（CE）. 2008, 2008(13(2008-CE-093)), p.57-64.
[4] 飯田千代ほか. アルゴリズム的思考法の学習―記憶から思考へ―. 研究報告コンピュータと教育（CE）. 2009, 99(1), p.1-7.
[5] 木村優那, 矢入郁子. 段階的詳細化能力とプログラミング学習の関係に関する研究. 人工知能学会全国大会論文集. 2015, 29, p.1-4.
[6] 稲葉大祐, 原田史子, 島川博光. プログラミング教育のための語句選択を用いたプログラミング設計, 第 8 回情報科学技術フォーラム (2009).
[7] 板戸陽子, 小山昂紘, 原田史子, 島川博光. 問題文の切り分けによる学習者の機能分割能力の評価, 第 11 回情報科学技術フォーラム (2012).
[8] 新開純子, 宮地功. プログラミング学習支援システムを用いた入門教育の実践, 日本教育工学会論文誌, 2009, 33(Suppl.), p.5-8.
[9] 湯瀬裕昭ほか. 学士潜在能力涵養のためのプログラミング教育カリキュラムの開発. 第 75 回全国大会講演論文集. 2013, 2013(1), p.383-384.