Vol.2019-CE-151 No.5
2019/10/5

チューターによるプログラミング課題の確認を支援する入学前教育 e ラーニングにおける学習環境の構築

小山田圭吾[1]　市川尚[1]　高木正則[1]　富澤浩樹[1]　阿部昭博[1]

概要：本学部では e ラーニングによる入学前教育を実施しており，プログラミングの課題も含まれている．本研究では，受講対象者である高校生をチューターと学習者に分け，学習者が作成したプログラムに対して，チューターが確認を行うためのオンライン上の学習環境を構築した．そのシステムは，学習者がビジュアル型言語でプログラムを作成して提出するまでの履歴を記録し，チューターがプログラムや履歴を確認して学習者にフィードバックを行うことを支援する．構築した学習環境を用いて入学前教育で試行した結果，チューターの確認とフィードバックにより，学習者のプログラムの改善が見られた．一方で，履歴を提示することはチューターの役に立っていたが，プログラムの確認やフィードバックには不十分な点が見られるなどの課題が残った．その課題を踏まえ，システムの改善について検討した．

キーワード：プログラミング教育，入学前教育，チューター，ビジュアルプログラミング,e ラーニング

Development of e-learning environment for supporting tutors to check programming tasks in preparatory education

KEIGO OYAMADA[†1]　HISASHI ICHIKAWA[†1]　MASANORI TAKAGI[†1]
HIROKI TOMIZAWA[†1]　AKIHIRO ABE[†1]

Abstract: Preparatory education through e-learning including programming tasks has been conducted in our faculty. In this study, we divided the high school students who were the participants into tutors and learners, and built an online learning environment for the tutors to check the programs created by the learners. The system records the programming history data until the learner creates and submits the program by a visual programming language, and supports the tutor to check the program and history and provide feedback to the learner. As a result of trial in preparatory education using the constructed learning environment, the student's program was improved by the tutor's confirmation and feedback. Presenting the programming history was useful for tutors, but there were still problems such as insufficient confirmation and feedback to programs. Based on these issues, we examined the improvement of the system.

Keywords: Programming Education, Preparatory Education, Tutor, Visual Programming, e-learning

1. はじめに

入学前教育は全国の大学で行われており，e ラーニングを活用した事例も多数存在する[1].そこでは多様な学習課題に学習者が取り組んでいるが，情報の内容を扱っている事例も見られる[2].岩手県立大学ソフトウェア情報学部でも，AO 入試・推薦入試合格者に対して e ラーニングによる入学前教育を実施している.課題には情報の内容もあり,大学入学前にプログラミングを体験し，基本的なプログラムの構造を理解してもらうことを目的に，ビジュアルプログラミング言語を使用した課題を提示している．

プログラミング教育は，高等教育や初等中等教育，情報系企業の新人研修など幅広い世代で行われている．情報を専門的に扱っている学部や組織ではもちろんのこと，たとえば近年では，文系の学部でも導入教育として Scratch などのビジュアルプログラミング言語を使用した事例が多数見られる[3]．初等中等教育においては，現行の高等学校学習指導要領でも，中学校の技術家庭科や高等学校の情報科（情報の科学）でも触れられている．また，2017 年に告示された新学習指導要領[4]では,小学校からプログラミング教育が導入されるなど，近年プログラミング教育の社会的重要性が高まっていると言える．

本学部はソフトウェア情報学部として情報を専門とはしているが，現状では入学前教育参加者のプログラミング経験にはばらつきが見られ，入学前から十分な実力を持っている学生もいれば，ほとんど経験のない学生も存在することが分かっている[5]．そこで，本研究では，入学前教育の受講対象者である高校生をチューターと学習者に分けてグループを組ませ，学習者が作成したプログラムに対してチューターが確認を行うためのオンライン上のプログラミング学習環境を構築した．本稿では，2019 年度の入学前教育にプログラミング学習環境を導入した結果とその改善について述べる．

1 岩手県立大学大学院ソフトウェア情報学研究科
Graduate School of Software and Information Science， Iwate Prefectural University

2. 研究背景

2.1 入学前教育について

岩手県立大学ソフトウェア情報学部では，高校から大学で行われる授業への円滑な移行と学習習慣を定着させるために，2013 年度から AO・推薦入試合格者に対して e ラーニング形式で入学前教育を行っている．少人数の教員と大学生メンターにより運営され，学習管理システム（LMS）の Moodle 上にコースを設置し，学習者は 4〜5 人のグループを組んで，個人が主体的に学習を進めている．基本的には大学側はあまり手を出さず，合格者が自分たちで進めることを方針としている．入学前教育は合格者自身の高校生活を大切にしてもらう必要があるため,各自の状況に応じて，できるだけ取り組んでもらっている．合格者は全国各地に散らばっており，お互いは顔をあわせていない．コースへアクセスする環境もアクセスしている時間もさまざまであり，非同期的なやりとりになっている．

筆者らは，本学部の入学前教育のなかで，2018 年度にオンライン上での相互チェックを取り入れたプログラム学習環境を構築した[5].入学前教育にはプログラミング経験者と初心者が混在している．その状況を活かして，学習前にテストを実施し入学前から十分な実力があると判断できた高校生にはチューターとしてプログラミング課題に参加してもらい，学習者の課題へのフィードバックに集中してもらった．進め方はまず，チューターと学習者でグループを作成する．学習者は課題を作成し提出する．その後，グループの学習者同士でお互いのプログラムを確認する．学習者同士の確認で合格となったらチューターが再度それらを確認するという流れとなっている．なお，本稿では，入学前教育全体の支援を行っている大学生を「メンター」，プログラミング課題で作品をチェックする役割を担う高校生を「チューター」として区別する．

このシステム（先行システム）は PC のブラウザ上での動作を想定しており,学習教材には Scratch を使用していた．全ての受講者が相互にプログラムの確認を行うため，特にプログラミング初心者が行うチェックを支援するために注目してほしいポイントをチェック項目として提示した．2018 年度の入学前教育に導入した結果，オンライン上での相互チェックは学習者が提出したプログラムの質向上にある程度寄与していたことが示唆された．しかし，問題点として特にソースコード面でのチェック漏れが見られたこと，チューターへの支援が不十分であったことが分かった．

2.2 関連研究

プログラミング教育において，プログラミング学習の履歴を活用して，プログラムミングの学習者の行動分析などを行っている研究も多数存在する.加藤ら[6]は作業時間やコンパイル結果から学習者の行動特徴を 4 つに分類している.田中ら[7]は間違いが発生した際に起こる学習者の行動パターンを確認している．また，指導者の支援のために，学習者の作業履歴を活用している研究は数多く存在する．井垣らの研究[8]では,講師が学習者のコーディング過程を分析・可視化することにより，助けを必要としている学習者の発見に役立ったことを報告している．加藤ら[9]は，学生の学習状況や作業履歴を把握することで，机間巡回では把握が困難な学習状況に対する指導ができ，学生から助けを求められる前に学習指導が行えたことを報告している．このように，先行研究では履歴を活用することが学習者の指導に役立つことを示唆している．一方で，遠隔かつ非同期で行われる高校生同士の教えあいに履歴を活用している例は見つけられなかった．

さらに，プログラミングを含めた情報教育において複数人で学習を行っている研究も多数存在する.生田目ら［10］は，プログラミングの授業においてグループ学習は学習目標を達成させるために有用であることを述べている．大矢ら ［11]は，情報基礎教育でのペアによる協調作業を行う際は学力差を大きくする編成にするのが望ましいことを述べている．これらのことから，プログラミングの経験者と初心者でグループを組ませて学習を進めることは有効であると考えられた．

2.3 研究課題

先行システムを入学前教育に導入した結果，プログラムの確認が不十分であることやチューターへの支援に関する問題が挙げられていた．また，先行システムでは，チューターの確認前に学習者同士の相互チェックも入れていたが,初心者同士ではチェックが難しく，かつ非同期のため待ち時間など余計な時間が増えてしまうという問題点も見つかった．そこで，本研究では，学習者複数名をチューター１名が担当して確認することにし，チューター役となった高校生が，学習者（こちらも高校生）が作成したプログラムに対して適切な指導を行えるようにすることを目的として,チューターが課題を確認して学習者にフィードバックを行うことを支援するシステムを設計・開発した．

3. 入学前教育におけるプログラミング課題

3.1 入学前教育の流れ

入学前教育の情報課題の流れは以下のとおりである．

①事前テストとアンケートに答える
②テスト結果を基にチューターを選びグループを組む
③学習者は課題に沿った作品を作り提出する
④チューターがプログラムのチェックを行う
※課題 4 終了まで③と④を繰り返す
⑤課題 4 終了後，事後テストとアンケートに答える

④でチューターが不合格と判断した場合，学習者はプロ

表 1　課題内容とチェック項目

Table1　Learning Tasks and check items.

	課題内容	チェック項目
課題 1 （順次処理）	数字を 2 回入力して変数に格納し，2 つの数字で四則演算を行った結果を，和・差・積・商の順で表示するプログラムを作りなさい	・和，差，積，商の順番で答えが表示されているか ・数値は入力できるようになっているか ・出力された答えは正しいか ・変数は正しく使えているか ・必要のない処理は行っていないか
課題 2 （分岐処理）	数字を 2 回入力して変数に格納し，分岐処理を使って大きい方の数字を出力するプログラムを作りなさい．また，同じ数字が入力された場合，そのことを表示するようにしなさい	・分岐処理を使っているか ・数値は入力できるようになっているか ・変数は正しく使えているか ・正しく動作しているか ・必要のない処理は行っていないか
課題 3 （反復処理）	反復処理を使って星型を複数描くプログラムを作りなさい（一部が重なるのは良いが，1 箇所に重なってはいけない）	・反復処理が使われているか ・複数の星が描けているか ・1 か所に重複していないか ・星はすべて同じ形であるか ・必要のない処理は行っていないか
課題 4 （総合問題）	数字を 10 回入力し，奇数が入力された回数を出力するプログラムを作りなさい	・数値は入力できるようになっているか ・分岐処理を使っているか ・反復処理を使っているか ・変数は正しく使えているか ・作品は正常に動いているか

グラムを再提出する（つまり③に戻る）ことになる．事前テストと 2 度のアンケートはすべての入学前教育受講者に回答してもらうが，事後テストは学習者として課題に取り組んだ生徒のみが回答することを想定している．

3.2 課題内容

今回の課題は基本的な制御構造とされている順次･分岐･反復の処理に加えて，入出力と変数の扱いについても理解させることを目標として用意した．また，課題 4 として順次・分岐・反復を組み合わせた総合問題を提示した．提示した課題内容を表 1 に示す．

3.3 チェック項目について

チューターは高校生であるためプログラムの確認は不慣れだと考えた．そのため，何を確認すればよいか明確になるようにチェック項目（表 1）を提示した．チェック項目は学習者にも提示し，課題作成時に注意してもらった．今回は，これらのチェック項目すべてを満たしている課題を合格とした．

3.4 学習環境の構成

学習環境のシステム構成を図 1 に示す．学習者には課題内容や必要に応じてヘルプを参照しながら，自分で学習を進めてもらい，メンターや教員はシステムの運用や状況に応じた支援を行う．入学前教育は Moodle 上で行っているため，学習者は Moodle から別のシステム上に用意したプログラム作成機能と課題評価機能にアクセスする．システムは Moodle からユーザーの情報を取得しており，誰が課題を作成したか，または誰が課題の確認をしたかが分かるようになっている．プログラム作成機能で行われた作業は履歴取得機能によってデータベースに記録される．課題評価機能はチューター用と学習者用の 2 種類があり，チューターはシステムで確認を行い，学習者は確認結果を受け取る．また，課題の内容と学習者のプログラム作成またはチューターの確認を支援するためのヘルプを Moodle 上に用意した．

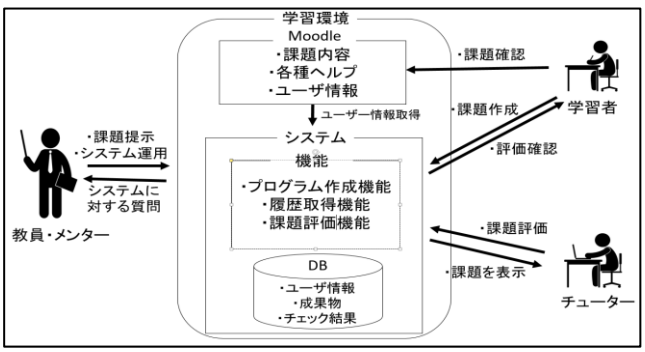

図 1　システム構成

Figure1　System Component.

4. システム開発

4.1 開発環境

学習ツールには Google が無償で提供している Blockly というビジュアルプログラミング言語を使用した．システムの開発環境にはPHP と Javascript,データベースはMySQL を使用している．また，本システムはPC のブラウザ上での動作を想定している．

4.2 機能概要

(1) プログラム作成機能

学習者が課題に取り組む際のエディタである．ブロックによるプログラミングを行うスペースの他に，実行結果を出力するスペースと各種ボタンがある．ボタンは，プログラムを実行するボタン，画面上のブロックをすべて削除するボタン，課題を提出するボタンがある．このエディタ上で行われた動作が履歴取得機能によって記録される．

(2) 履歴取得機能

学習者の行動を記録する機能である．エディタの状態は XML 形式でデータベースに記録する．データベースに記録されるデータは以下の通りである．履歴はブロックを配置または削除する度に記録される．プログラムを実行したかどうかはエディタ上の実行ボタンを押した際に記録される．

- 作業した学習者
- 記録時のエディタの状態
- 作業した時刻
- プログラムを実行したかどうか

(3) 課題評価機能

チューターが確認を担当した学習者のプログラムを確認する，または学習者が確認結果を確認する機能である．画面を図 2 に示す．チューターには学習者が取り組んでいる課題内容とそのプログラミング履歴，チェック項目，アドバイスを記入する欄が提示される．チューターはそれらを見ながらプログラムの確認を行う．プログラミング履歴は画面下にあるシークバーとボタンを使いながら確認する．シークバーを動かすことで，プログラムを作成したプロセスを閲覧できる．プログラムの正解例も提示し，チェック項目と同様に参考にしてもらう．一方で，学習者にはチューターの確認結果が表示される．表示されるのは，どのチェック項目が満たされていたのかと自由記述によるチューターからのコメントである．

(4) 各種ヘルプ

入学前教育が行われている Moodle のコース上に各課題で注目するべきポイントをチェック項目として提示した．また，Blockly の使い方やプログラムの考え方，各チェック項目の説明などを記載したページを用意した．

具体的には，基本的な制御構造の説明とそれを活用したサンプルプログラム，ブロックの使い方，チェック項目の説明，システムの使い方を記述したものである．

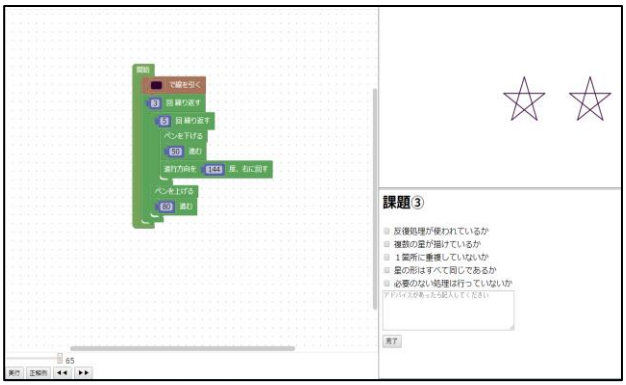

図 2　チューター用のプログラムの確認画面
Figure2　Screen to check program by tutor.

5. 評価

5.1 入学前教育における実践

開発したシステムを 2018 年度の入学前教育のプログラミング課題に導入した[12]．まず，入学前教育参加者 73 名に事前テストとアンケートを実施し，それを基にチューターを選抜した．学習者 51 名，チューター22 名となった．事前テストは課題内容に対応した Blockly のプログラムを提示し，そのプログラムがどんな動作をするのか答える問題を 10 点満点で出題した．チューターは事前テストで満点だった受講者に担当してもらった．課題は 2018 年 11 月 22 日～2019 年 2 月 10 日の期間を設けて取り組ませた．チューターには，学習者には直接答えを教えないようにするなどの注意点をあらかじめ伝えた．課題終了後，事後テストとアンケートを実施した．事後テストは事前テストと同レベルの異なる問題を出題した．アンケートは，システムの利用に関する質問を，7 件法と自由記述で構成して提示した．本稿では，2019 年 3 月 5 日までに集まったデータを分析対象とした．

5.2 学習結果

(1) テスト結果

事前テストは受講者 73 名，事後テストは学習者 51 名全員が受験していた．学習者 51 名の事前テストと事後テストの結果（平均点）は，10 点満点中，事前テスト 2.5 点に対して事後テスト 6.0 点であった．個人の点数を見ると，結果が変わらない学習者が 8 名見られたが，他 43 名は得点が向上していた．

(2) 作業時間

学習者が各課題のプログラム作成にかけた時間として，作業に着手をした時刻から最後に作業をした時刻までどれほど時間がかかったかを計測した．途中で 30 分以上作業を行っていない場合は学習を一時的に終了したと判断し，作業時間には含めずに計測した．計測結果を表 2 に示す．総合的に見ると，最短で作られたのは課題 2 で最も時間がかけられたのは課題 4 であることが分かった．作業時間が

200 分を超えている学習者に共通して見られたのは作業を断続的に行っていたことであった.

表 2　課題にかけた時間

Table2　Time spent on each programming task.

	課題 1	課題 2	課題 3	課題 4
最長	6 分 55 秒	4 分 5 秒	6 分 58 秒	4 分 36 秒
最短	235 分 5 秒	111 分 27 秒	98 分 48 秒	340 分 36 秒

(3) 作業回数

学習者 51 名の作業回数を集計した．作業回数はブロックの配置または削除が行われた時を 1 回として集計した．各課題の作業回数の中央値･最大値･最小値を表 3 に示す．集計の結果,課題 4 の作業回数が多くなる傾向にあること，課題4の次は課題1が多くなる傾向があることが分かった．また，最大値を見ると作業回数が 500 を超える学習者が見られた．このことから，チェックを担当する学習者によってはチューターの確認が長時間で行われている可能性があると考えられる．

表 3　プログラミングの作業回数

Table3 Number of programming operations.

	課題 1	課題 2	課題 3	課題 4
中央値	179	81	101	241
最大値	1263	732	885	2898
最小値	47	25	25	25

(4) 課題ごとのプログラミングの特徴

学習者 51 名が提出した課題のプログラミングの履歴から得られた課題ごとの特徴を述べる．筆者が閲覧して調べた結果である.

課題 1：Blockly の使い方を確かめるような作業をしている学習者が 23 名見られた．その内，課題に関係のないブロック（分岐処理のブロックなど）を使っている学習者が 9 名みられ内 1 名はそのブロックをどのように使うかを長く考えていた．変数の中身を確認する作業を行っている学習者が 23 名見られた.

課題 2：51 名中 41 名が，2 つの変数を最初に用意してから分岐処理について考えていた．合格基準に達した学習者 34 名の課題のうち，16 名が出力結果を 3 種類作成する点に躓いていた．9 名の学習者は，ヘルプで提示してあるサンプルを最初または途中で再現し動きを確認してから課題に取り組んでいた.

課題 3：星を 1 個描いてからその数を増やすという順序で考えた学習者が 44 名見られた.星を描くための直線を 5 回引く処理は 45 名が躓くことなく取り組んでいた．線を引く処理や星を描く処理に躓いたと思われる学習者が 29 名見られた.

課題 4：反復処理と分岐処理を同時に使用する点に躓いた学習者が 7 名見られた．また，分岐処理の中に反復処理を入れるという点に躓いた学習者が 4 名見られた．躓きの原因のほとんどは変数の扱いであり，奇数をカウントするという処理に躓いた学習者が 40 名見られた.

5.3 チューターによる確認の結果

学習者の各課題の提出率とチューターによるプログラム確認の妥当性について調査した．チューターが行ったプログラムの確認結果を表 4 に示す．また，チューターの確認とは別に，メンターが，チューターが妥当なチェックを行っているかの確認を入学前教育終了後に行った．そこで合格であると判断した数を表 5 に示す.

(1) 提出率

課題 3 と 4 を提出していない学習者がそれぞれ 2 名見られたが，それ以外の学習者は最低 1 回以上課題を提出していた．学習者はほとんどが合格するまで再提出を続けていた．しかし，1 回目の確認で不合格にも関わらず 2 回目以降の確認を受けていない学習者が，特に課題 2 で 6 名，課題 4 で 5 名と目立った．課題 2 の 6 名の内，2 名は課題 3 と 4 が未提出の学習者である．課題 2 と課題 4 のどちらも不合格という学習者は 2 名であった．2 回目の確認を受けていない学習者の内 7 名の特徴として，合格不合格にかかわらず，その学習者が提出した他のすべての課題の確認を一度しか受けていないという共通点が見られた．理由として，提出期限の直前または期限後に提出していたことが考えられる．また，未提出の原因も調査したところ，課題 3 以降が未提出の 2 名の内 1 名は提出期限が過ぎてから取り組んでいた．もう 1 名は課題が理解できてないまま課題 2 に進んでおり，かつ解決しなかったために学習意欲が削がれてしまったものと考えられる．また，課題 4 の 3 回目以降のチェックで不合格となっている 2 名は最後の作業時刻から提出期限に間に合わなかったことが原因であると考えられる.

(2) 確認の効果

学習者の合格者数の変移は表 4 の通りである．すべての課題について，1 回目の合格が不合格であっても再提出後の 2 回目に合格するなど，確認を重ねるごとに合格者数が増えていることから，チューターによる確認は学習者のプログラム改善につながったことが示唆される．特に効果が見られたのは課題 1 で，逆に見られなかったのは課題 4 であった．課題 1 に効果が見られたのは，最初の課題なので学習者にも修正点と修正方法が分かりやすかったものと考えた．課題 4 は最後の課題で，修正方法に苦労したものと考えられる.

(3) チューターによる確認の妥当性

チューターが行なった確認（表 4）とメンターによる最終確認（表 5）を比べると，チューターの確認のうちの約

表 4　学習者による課題の提出率とチューターによる確認結果
Table4　Submission rate and results of checking programs by tutor.

	1 回目				2 回目			3 回目以降		
課題番号	提出	未確認	確認		再提出	確認		再提出	確認	
			合格	不合格		合格	不合格		合格	不合格
1	51	5	25	21	18	8	10	9	9	0
2	51	6	26	19	13	7	6	5	5	0
3	49	7	34	8	7	6	1	1	1	0
4	49	9	28	12	7	4	3	3	1	2

表 5　メンターの最終確認による合格者数
Table5　Results of final confirmation by mentor.

	1 回目	2 回目	3 回目以降
課題 1	21	8	9
課題 2	22	7	5
課題 3	28	4	1
課題 4	23	4	1

86.4%（約 9 割）が妥当に行われていた．1 回目の確認ではそれぞれの課題で 4〜6 件のチェック漏れが見られた．しかし，2 回目以降の確認は課題 3 の 2 件を除いたすべての確認が適切であった．不十分な確認を行っていたのは，チューター22 名中 7 名（31.8%）であった．また，課題の不備は全体で 21 件見られた．そのうち 1 名は確認を担当した 12 件の課題の内 7 件（33.3%）に問題があった．他には 4 件（全体の 19.0%）の確認に問題が見られたチューターが 2 名見られた．これらのことから，確認に問題のあったチューターは少ないが，一部のチューターに集中していたと考えられる．

(4) 確認が不十分であった原因

チューターとメンターがチェックして違いが見られた項目について調査した．チェック項目についての説明はあらかじめ行っていた．不十分の原因としては，本来チェックをしてほしい項目にチェックがされていないかった場合（チェック漏れ）と，チェックをしてほしくないのにチェックがされていた場合（余計なチェック）がある． チェック漏れについては，課題 2 と課題 3 の項目 1 は互いに制御構造のブロックを使用しているかどうかという内容だが，ブロックを使用しているにも関わらずチェックされていない場合が合計 6 件見られた．課題 1〜3 までの項目 5 は不必要な処理を行っていないかという内容だが，無関係な処理が特にみられなかった合計 9 件にチェックがされていなかった．余計なチェックについては，課題 1 と課題 2 の項目 2 は同じ内容であるが，合計で 5 件の不備が見られた．課題 3 の項目 1 は，不備の原因として，2 つ目以降の星を描く際に反復処理を使っていない点を見逃しているチューターが多くみられた．課題 4 の項目 4 では，最後のカウントの結果を出力する際に「奇数が入力された回数」という文章を出力している学習者をそのまま合格としている例が見られた．

(5) 確認時の付与されたコメント

チューターが学習者に送ったコメントを調査した．チェックの際，どこが間違っていたのかなどのアドバイスをコメントで伝えてほしいということをチューターにはあらかじめ伝えていた．チューター22 名が行ったプログラムの確認は全体で 267 回であり，そのうち 178 回（66.7%）にコメントがつけられていた．コメントの集計結果を表 6，実際に見られたコメントの例を表 7 に示す．今回は，プログラムの間違っている点や不足している点に対して，その解決法を間接的に教えているような内容が含まれていると判断したアドバイスを適切なコメントとして捉えた．表 6 のその他は，相手を労ったり工夫を褒めたりするような，アドバイス以外の内容が含まれているコメントの数を示している．集計の結果から，課題 1 が最もコメントが多く見られ，課題 3 が最も少なかったことが分かった．これは，チューターの確認で不合格となった人数が関係しているものと考えられる．

次にコメントの内容を確認した．確認の結果，アドバイスとしてコメントの内容が不十分な例としては，同じ間違いを繰り返している学習者に対して同じアドバイスを繰り返しているものや，課題内容を伝えているだけでどこが間違っているのか伝わらないようなコメントが見られた．また，適切な例の 2 つ目は「作成のはじめのあたりで」という文が見られた．このことから，作業履歴を確認してアドバイスを行ったものと思われる．一方で，作業履歴に着目してアドバイスを行っていた例は全体を通して 5 件にとどまった．

表 6　コメント数の集計結果
Table6　Comment count results.

	適切	不十分	その他	計
課題 1	32	12	12	56
課題 2	18	12	18	48
課題 3	16	3	14	33
課題 4	21	3	17	41

表 7　アドバイス例
Table7　Examples of advices.

適切な例	大小を判定する処理は出来ていますが、この課題は、大小の判定に加え、入力された数値が同じだったとき、そのことがわかるような出力をしなければなりません。 したがって、判定の処理を少し変え、同じ数値が入力された場合の処理を追加する必要があります。
	数値を入力でき、正しい値を表示できています。しかし、変数 answer を使わなくとも表示することができ、その方がよりすっきりとしたプログラムになります。作成のはじめのあたりで〞を表示〞というブロックに計算するブロックを入れていたのを思い出して作成してみてください。
不十分な例	変数に数値を入力できるようにしてください。(←何度も同じ間違いをしているのにこのコメントを何度も繰り返す)
	変数 x が大きい場合は x の値を出力されるようにしましょう。 同じ数字の場合にそのことが分かるようにしましょう。(←課題内容をそのまま入れているだけ)

5.4 アンケート結果

チューター22 名中 21 名からアンケートの回答を得た．プログラミング履歴を確認したかどうかを質問したところ，21 名中 19 名が確認したと回答していた．残り 2 名は履歴が見られることに気づいていなかったと記述しており，内 1 名は確認に不備が多く見られたチューターであった．同時に，どの課題の履歴を確認したのか調査したところ，課題 1 は 17 名，課題 2 と課題 3 は 16 名，課題 4 は 14 名が確認したと回答していた．プログラムを確認するために履歴が役に立ったかを質問したところ,履歴を確認した 19 名全員が役立ったと肯定的な回答をしていた．役立った場面について質問したところ，課題 1 は必要のない処理を行っておりどこで間違えたのかを見つけるため，課題 2 は条件分岐がしっかり行えているかの確認や正しい値が表示されない原因の調査，課題 3 と 4 では課題の試行錯誤を確認したというような意見が見られた．

履歴を見るうえで注目した点はあるかという質問をしたところ，課題 1 は初めてのプログラミングということもあり入力や表示でてこずっていたから簡単に説明した，課題 2 は if 文に関しては else を使う人もつかわない人もいるためそこに注目した，課題 3 は星が 1 つ描ければスムーズに進められるかもしれないと思って見ていた，課題 4 は奇数をどんな方法でカウントするのかを注目したなどの記述がみられた．また，全体の課題を通して無駄な処理を行わずに反復処理や条件分岐などを用いて簡潔なプログラムに仕上げられているかということに着目したという記述も見られた．学習者の履歴を見て気づいたことがあるかという質問をしたところ，作業回数から苦労しているのが伝わったというような学習者に対する気づきや，答えの表示ができてないといったプログラムに対する気づきが見られた．

6. 考察

6.1 チューターによるプログラム確認の支援

本システムにより，学習者のプログラミング履歴を取得することができ，学習者のプログラミングの状況をより具体的に把握することができた．また，チューターの確認とフィードバックにより，学習者のプログラムの改善が見られた．課題内容では，作業回数や作業時間から，課題 4 の基本的な処理を組み合わせるということが初心者には難しいと考えられる．次に多いのが課題 1 であるが，これは，初めて Blockly を扱うので使い方を確認する作業が発生していることが原因と思われる．また，初心者にとって最後まで変数の扱いに躓いている学習者が見られたことから，変数の扱いが難しいことも原因の 1 つとして考えられる．提示した課題と違う処理をしている学習者が見られたことから，学習者に課題内容がうまく伝わっていない可能性が考えられる．

チューターによるプログラムの確認に関しては，不十分な内容が一部のチューターに見られた．原因を確認した結果，各チェック項目でチェック漏れと余計なチェックが見られたことから，チェック項目の合否の基準が曖昧であるなど，チェック項目によって課題提示者の意図が伝えられていないことが考えられる．そのため，チェック項目の改善や説明の追加が必要だと思われる．また，コメント例から，指導には不十分なアドバイスも見られた．そのため，チューターに選ばれた人にはどんなアドバイスをしてほしいかを明確にさせる必要がある．システムに関しては，履歴を取得することはチューターからも履歴を取得して提示することは，チューターからも肯定的な意見が得られ，効果的な指導を行うための支援にある程度寄与していたと考えられる.しかし,学習者の作業回数が 500 を超えており，チューターがそれらを確認するのに時間がかかっていることが考えられる．また，プログラム確認の妥当性について確認したところ，チェックを忘れているチューターも見られたため，防止する策を検討する必要がある．

6.2 システム改善

評価結果と考察を踏まえ，システムの改善について検討した．今後必要と考えられる改善について示す.

(1) プログラム履歴閲覧の効率化

作業回数の集計結果から，学習者によってはかなりの回数の作業をしており，チューターがそれを確認するのは負担であると考えられる．そのため，作業履歴を短縮し，そ

れをチューターに提示する必要がある．

(2) チェック項目の再検討

チェック項目の不備を確認した結果から，項目ごとの説明が不足していることやチューターの合否の基準が曖昧となっていることが考えられる．そこで，項目そのものを改善するかヘルプに説明を追加する必要がある．

(3) アドバイスの質向上

アンケートの結果から，プログラム確認の際に履歴を提示することはチューターの役に立っていると思われるが，コメントの集計結果から，作業履歴をうまく活用したアドバイスができていないものや，不十分なアドバイスも見られていた．そこで，適切なアドバイスの例をいくつか提示し，それを参考にチェックを行ってもらう．

(4) チューター履歴の取得

現状のシステムではチューターがどのように確認を行ったかが把握できていない．そのため，チューターの作業履歴を取得できるようにして，チューターの行動も分析できるようにする．未確認の課題も見られたので，履歴をもとに,システム上で通知するなどして作業を促すようにする．

(5) 課題１でのブロックの厳選

課題１ではブロックの使い方を確認する作業の一環として，課題作成に無関係のブロックを使っている学習者が見られた．その作業は問題ないが，無関係のブロックの使い方に躓いて作業回数がかかったと思われる学習者も見られた．そのため，課題１の段階では，課題に関係のないブロックは提示しないようにして，初心者の認知負荷を減らすことを検討する．

(6) 課題の内容に関する検討

課題の内容については，提出された課題を確認したところ，最後まで変数の扱いに躓いている学習者が見られた．そのため，変数の扱いに関しては別途課題を用意するか，課題のヘルプを詳細にすることで支援を手厚くする必要がある． また，学習者に課題の内容が正確に伝わっていない可能性も考えられたので，出力結果の画面を提示するなどしてゴールを明確にする必要がある．

7. 終わりに

本研究では，入学前教育において，学習者が作成したプログラムに対してチューターが確認を行うためのオンライン上の学習環境を構築し，入学前教育に導入した結果と改善案を述べた．評価の結果，チューターのプログラム確認の際にプログラミングの履歴を提示することは，チューターから好評であった．しかし，プログラムの確認に不十分な点が見られたことなどの課題も残った．今後は検討した改善案をもとにシステムを改善し，次の入学前教育に導入して評価を行う予定である．

参考文献

[1]川西雪也ほか：e-Learning を活用した入学前教育に関する実証研究,メディア教育研究 第５巻 第１号(2008)
[2]新ケ江登美夫ほか:大学におけるコンピュータリテラシー教育，中村学園大学・中村学園短期大学部研究紀要第 48 号，pp247-253（2016）
[3]松村寿枝ほか:ビジュアルプログラミングツールを用いた効果的なプログラミング導入教育の実践，教育システム情報学会研究報告，Vol30，No7，pp54-64（2015）
[4]文部科学省：学習指導要領「生きる力」(2018)
[5]小山田圭吾ほか:オンライン上での相互チェックを取り入れた入学前教育におけるプログラミング学習環境の開発，情報処理学会第 80 回全国大会講演論文集，pp617-618（2018）
[6]加藤利康ほか：プログラミング行動の履歴に対する Deep Learning 分析，JASLA Research Report Vol.2017,No.1(2017)
[7]田中良樹ほか:ビジュアルプログラミング対応版プロセス観察システムの開発と学習効果分析,情報教育シンポジウム(2015)
[8]井垣宏ほか: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 vol54,No.1,pp330-339(2013)
[9]加藤利康ほか:プログラミング演習のための授業支援システムにおける学習状況把握機能の実現，情報処理学会論文誌 Vol55，No.8，1918-1930（2014）
[10]生田目康子ほか：ピア・レビューをともなうグループ学習の評価--一斉型プログラミング授業への適用，情報処理学会論文誌 vol45（2004）
[11]大矢芳彦ほか：情報基礎教育におけるペア学習の試みとその組み合わせ指標に関する基礎研究，名古屋外国語大学外国語学部研究紀要，Vol36，pp223-241（2009）
[12]小山田圭吾ほか：入学前教育におけるプログラミング課題の履歴を活用した学習環境の試行，情報処理学会第 81 回全国大会講演論文集，pp545-546（2019）