2020 年度　修士学位請求論文

オンライン時代でのプログラミング
初年次教育を円滑にする手法と実践

明治大学大学院先端数理科学研究科

先端メディアサイエンス専攻

又吉　康綱

Master's Thesis

Methods and Practices to Facilitate First Year Programming Education in the Online Age

Frontier Media Science Program,

Graduate School of Advanced Mathematical Sciences,

Meiji University

Yasutsuna Matayoshi

概要

大学の情報系学部学科の初年次における必修のプログラミング教育では，数十名から数百名の学生を一度に指導する必要がある．さらに，初めてプログラミングを習う初学者から，プログラミングをやったことがある経験者までを対象とした講義を行う必要があるため，各講師は趣向を凝らしている．

しかし，COVID-19 の影響により大学のプログラミング教育がオンライン講義で行われるようになってきた．プログラミングのオンライン講義を行うことは容易ではなく，学生がタイピングを行う様子や顔色が対面講義に比べて見にくくなることで理解しているのかの確認ができないことや，オンライン通話特有の会話のしにくさによって指導がスムーズにできないことなど，大きな障壁がたくさんある．そこで，本研究では，大学初年次におけるプログラミングのオンライン講義の際に生じる最も重大な 2 つの問題点に着目した．

対面講義では，講師が学生の表情や手が動いているかどうかなどから苦手な学生を把握でき，個別の指導や対応を行うことが可能であったが，オンライン講義では，どの学生が苦手かを把握することが困難になった上に，オンライン上での指導には限度があるため適した指導が出来ず，大きな問題となる．ここで，初学者がプログラミング学習においてつまずく点は様々であるが，これまでの数年にわたるプログラミング教育の経験から，タイピングの技量と基本的命令の理解が不足していることが観察されていた．タイピングの技量不足については，タイピング速度の遅さや，プログラミングに利用される英単語，カッコやセミコロンなど特殊文字の入力に抵抗があることが妨げとなっていた．タイピング練習を行うためのタイピングゲームもあるが，多くは英単語を入力の対象としているため，プログラミング特有のタイピング練習には時間がかかってしまう．また，基本的命令を理解せずに学びを進めてしまうと，新しい命令や複雑な使い方を行うときに学習を止めてしまう．さらに，使いたい命令を思い出せないときに調べる手間が生じるため，学習の妨げになっていた．そこで，講師の指導が行き届かない問題を解決するためにプログラミングのタイピング練習を行うことでプログラミングが苦手な学生を減らせると考えた．そのために，学生のタイピングの技量と基本的な命令の理解が不足していることに着目し，プログラムの逐次的な実行などによってプログラムの動作を把握しつつ，自身や他者と競いながらタイピング速度を上げつつプログラムを学ぶ，プログラムタイピングシステム typing.run を提案および実装する．本システムを演習型プログラミング講義にて，講義前までの課題として 11 回ある講義ごとに複数回学生に取り組むことを課した運用の結果，学生 118 名によって 88,293 回分のタイピングが行われた．その分析より，タイピング速度にばらつきがなくなっていることや全体的なタイピング速度が向上していることから学生のタイピング技量の底上げが出来ていることを明らかにした．また，アンケート結果より，システムを繰り返し使用することで基本的命令の定着が可能であることが示唆された．

次に，プログラミング教育では学生の質問に対応する TA（ティーチングアシスタント）が必要不可欠である．しかし，学生の人数や質問数に対して TA の人数が十分でないことも多く，講義を円滑に進めることの妨げになっている．また，学生が手を上げた順番を考慮しながら TA が対応する必要があることに加えて，TA が学生の質問を解決できずに時間がかかったり精神的な負担がかかったりするといった問題もあった．さらに，学生が TA に対して些細な質問をして良いのかという不安や，質問している様子が恥ずかしいなど積極性を出せない問題があった．このような対面講義でもあった，質問したい学生と TA の間のコミュニケーションに生じる問題は，オンライン講義になったことで，より大きな問題となっているといえる．学生の積極性を考慮すること，TA の精神的な負担を軽減すること，そして質問順番の管理を適切に行うことが重要である．そこで本研究では，学生が TA に直接質問をするのではなくシステムに対して質問を行い，TA はシステムに投稿された質問を事前に確認をし，対応可能な場合に学生の呼び出しを行う，オンライン TA 質疑システム askTA を提案および実装する．本システムを実際の演習型プログラミングのオンライン講義で計 1600 分運用した結果，61 名の学生から 367 回もの質問を受け付けた．学生が質問を投稿してから TA が対応を開始するまでの時間と TA が指導をしていた時間を求めたところ，多くの場合，1 分以内に TA が対応を開始し，20 分以内には指導を終了していたことが分かった．また，学生と TA の双方のアンケート結果より，提案手法の効果が確認でき，肯定的な意見が多く集まった．オンライン講義において，質問したい学生と TA とのコミュニケーションを学生の積極性と TA の精神的な負担，質問順番の管理に着目して，円滑にすることができた．

最後に，2 つの提案手法の総合的な考察と， COVID-19 の状況下で学生にどのような影響を与えたかの考察を行い，本研究の運用からは明らかになっていない制限と今後のオンライン講義でのプログラミング教育や対面講義について，本手法がもたらす可能性やプログラミング以外の講義への応用について述べる．

Abstract

Compulsory programming education in the first year of a university's school of informatics requires teachers to provide lessons to dozens or hundreds of students at a time. Also, teachers need to give lectures to both beginners, who are learning programming for the first time, and experienced users. Therefore, teachers need to devise a good lecture plan. However, due to COVID-19, programming education in universities is now being conducted through online lectures. Online lectures on programming are too difficult. Online lectures pose many major barriers to learning compared with face-to-face lectures. For example, it is difficult to see the students' typing or even their faces, thereby making it difficult to check whether or not the students understand. Moreover, the difficulty of online communication makes it difficult to teach content smoothly. Therefore, in this study, I focused on the two most critical problems that occur during online programming lectures for first-year university courses.

Many barriers can cause beginners to stumble while learning programming. A teacher who has several years of experience in programming education may have observed a lack of typing skills and a lack of understanding of basic commands. Regarding the former, some students are slow at typing, and some resist typing English words in programming terms and using special characters such as parentheses and semicolons. Typing games to practice this skill are available, but many only target the input of English words. Thus, a long time is needed to learn programming specific typing exercises. If students proceeded with learning without understanding the basic commands, they may stop learning when they need to use new or complex commands. Also, when students cannot remember the commands they want to use, they have to look them up, and this hinders their learning. Therefore, this can be a major problem in online lectures because discerning which students are slow at typing is difficult and because online instruction has its limitations. In this paper, I describe the implementation of a proposed program typing system, typing.run, which facilitates programming learning. This system allows students to grasp the behavior of a program through its sequential execution and to compete with each other on typing speed to achieve good results. I operated this system during eleven programming lectures. In each lecture, students were required to type on the system multiple times. The results were that 88,293 typing attempts were collected by 118 students. The analysis revealed that the typing skills of the students could be improved. This is because the individual differences in typing speed were reduced and because the overall typing speed of the students improved. The results of a questionnaire suggested that the repeated use of the system helps students to memorize basic commands.

Next, teaching assistants (TAs) are indispensable in answering students' questions during programming classes. However, the number of TAs is not always sufficient for the number of students and questions, thereby hindering the smooth operation of the lectures. Other problems are that TAs have to answer questions in the order in which the students raise their hands and that TAs sometimes cannot solve their questions, which takes a long time and causes the TAs stress. Some students feel anxiety about the possibility of asking ignorant questions or embarrassment about asking questions at all, which prevents them from being proactive. This problem of communication between students who need to ask questions to TAs but are reluctant to do so, though it also occurs in face-to-face lectures, has become an even bigger problem in online lectures. Students' proactivity needs to be taken into account to reduce the TAs' stress and to manage the order of questions appropriately. In this study, I designed and implemented an online TAs questioning system, askTA. This system reduces the psychological burden on TAs while lowering the hurdle for students to ask questions. Students post questions to the system without directly asking the TA, and the TA checks the posted questions and lets the student know when an answer is available. As a result of running this system for a total of 1600 minutes in an actual online programming lecture, 367 questions were received by 61 students. I calculated the time it took for the TA to start responding and the time the TA spent teaching and found that the TA started responding within one minute and finished teaching within 20 minutes. The questionnaire results from both the students and TAs demonstrated the effectiveness of the method, and many positive comments were collected. In online lectures, I was able to facilitate communication between the students who wanted to ask questions and the TAs by focusing on students' proactiveness and TAs' mental stress and by managing the order of questions.

Finally, I provide a comprehensive discussion of the two proposed methods and a discussion of the impact on students during the COVID-19 situation. Furthermore, I discuss the unrevealed limitations and the possibilities and applications of our method for future programming education in online lectures and face-to-face lectures.

目次

第1章 はじめに

1.1. 大学の初年次プログラミング教育の現状

大学の情報系学部学科の初年次プログラミング教育では，数十名から数百名の学生を一度に指導する必要があると同時に，初学者から経験がある学生まで含めて広くカバーする必要があり，各大学，各学部で講師が趣向を凝らし，実施している．著者が所属していた明治大学総合数理学部先端メディアサイエンス学科では，学部 1 年生の必修講義としてプログラミング演習 I が開講されており，100 名以上の学生が同時に受講している．プログラミングは，学生によって得意不得意や理解度に大きなばらつきがあるうえ，そもそもコンピュータに対して不慣れであったり，タイピングが不慣れであったりするなど，苦手度合いが異なるため多角的なサポートが必要となる．したがって，限られた講義回数内で，プログラミングが苦手な学生の理解度をいかに向上させるかは，講師側にとって大きな課題である．

実際に，プログラミング演習 I の授業では，これまでに講義教材[1]の開発や課題の開発，予習のためのドリル[2]の開発，小テストの実施，発表会の実施など数年間に渡る様々な工夫を行っている．

1.2. COVID-19 によるオンライン時代の幕開け

世界中に猛威を振るっている COVID-19 の影響により，密を避けることやソーシャルディスタンスを始めとする新たな生活様式が提唱されている[1]．その結果，社会全体に変化が生じ，テレワークやオンライン飲み会など他者と直接接触しないことが求められるようになっている．本研究では，COVID-19 により社会全体に大きな変化が起きた事によって，あらゆることを他者と接触しないオンラインで行うようになった時代を「オンライン時代」と定義する．

オンライン時代の幕開けとなった 2020 年度は，大学を始めとする多くの教育機関でも学生間や教職員との接触を避けるためにオンライン講義が模索されながら開始された．講師が事前に録画した授業動画を学生に配布し視聴することによって行う収録動画配信型のオンデマンド講義や，ビデオ通話アプリを用いてリアルタイムに講師と学生が対面する同時双方向型のリアルタイム配信講義などが多く行われた．しかし，SNS 上では，学生が直接手を動かす必要のある演習型の講義をどのように行うのかについて講師側が試行錯誤している様子が見られる．さらに，国立情報学研究所がオンライン講義の運用に関するシンポジ

[1] https://nkmr.io/lecture/

[2] http://drill.nkmr.io/

ウム[2]を開催しており，オンライン講義に関する意見交換の場が設けられていることなどから，オンラインでの講義を行うための問題点は多く存在することがわかる．本研究は，オンライン時代における大学でのプログラミング演習で生じる 2 つの問題点に着目する．

1.3. 講師の指導が行き届かない問題

対面講義では，講師が学生の表情や手が動いているかどうかなどから苦手な学生を把握でき，個別の指導や対応を行うことが可能である．しかし，オンライン講義では，どの学生が苦手かを把握することが困難になった上に，オンライン上での指導には限度があるため適した指導が出来ず，大きな問題となる．

通常，プログラミング学習は，授業講義資料を読むことや実際にプログラムを書くことが多い．しかし，数年にわたる講義での知見から，プログラミングが苦手な学生の理解度が向上しない根本的な原因に，タイピングの技量と基本的な命令の理解が不足していることが観察されていた．

ここで，Python や C，Java や Processing [3]など，大学の初年次教育で採用されるプログラミング言語では，多くの日本人学生にとって日常生活では使用しないアルファベットを含む英単語や，カッコやセミコロンなどの特殊記号をタイプする必要があり，このことが学びの障壁となっている．タイピング速度を向上させるには，実際に手を動かして練習することが重要であり，実際に世の中には様々なタイピングに関するサービス[4]やゲーム[5]などが提供されている．しかし，大半のシステムは，日本語のローマ字入力や，英単語の入力を対象としていることが多い．そのため，こうしたタイピングシステムを利用してある程度タイピング速度を向上させたとしても，プログラミングにおいて多用される記号の入力などで時間がかかってしまい，結果としてプログラムのタイピングに苦戦する学生が多数存在していた．こうした学生は，プログラムを書き写すのにも多くの時間が掛かってしまい，学習や課題解決が困難になっていた．

また，プログラミング学習においては，プログラミングの際によく使う基本命令や関数への理解と，そうした関数へ慣れ親しむことも重要となる．学び始めのときは，使う関数が少ないため把握することは難しくないが，講義の回数を重ねるごとに新しい関数が登場するばかりか，条件分岐や繰り返し，配列，関数の定義，クラスなど，使い方を覚えて理解する対象が増えてくる．そのため，学習の過程で理解が不十分であると，次に進むことができなくなり，学習が止まってしまう．また，使いたい関数を思い出せない際，リファレンスや検索サイトなどで毎回調べる必要があり，わからないことが積み重なることも，学習をスムーズに進めるための妨げになっている．

以上より，講師の指導が行き届かない問題を解決するために事前にプログラミングのタイピング練習を行うことでプログラミングが苦手な学生を減らせると考えた．プログラミングのタイピングには，英単語や記号などのタイピング練習による入力速度の向上と，基本

命令や関数に繰り返し触れることでリファレンスを調べなくても自力で思い出し利用できるような反復による記憶が重要であり，学生自身が自学でこれらのスキルアップを図ることが可能な環境を構築する必要がある．

1.4. 質問をしたい学生と TA との間に生じる問題

講義時間内に学生のサポートを適切に行うには，TA（ティーチングアシスタント）の存在が必要不可欠である．一般的に，初年次教育において TA の数は不十分であり，1 名の TA が学生 10 名以上を担当することも珍しくない．

ここで対面講義では，質問したい学生が手を挙げて TA を呼び，学生は画面を TA と一緒に見ながら質問を解決し，TA は資料を学生に指しながら指導をすることになる．学生からの質問が多い場合には，手が上がった順番を考慮しながら対応していくことが可能である．しかし，順番を飛ばされることや，手を上げ続けることが大変などの問題もある．また，TA に対してこんなことを質問していいのかといった不安や，質問している様子を他者に見られることが恥ずかしいなど，積極性を出せないことによる問題もあった．さらに，TA も質問で呼ばれ対応したものの，解決できず困るという問題もあった．

COVID-19 の影響により，大学の講義がオンラインで行われるようになったことで，講義において TA がいかに支援するかという点はより大きな問題となっている．一般的に，学生と TA は多対多の関係であるものの，学部の必修講義などでは学生の数が圧倒的に多いため，オンラインコミュニケーション用のシステムにある挙手機能などを使って支援することは容易ではない．また，オンライン上にビデオ通話アプリなどで質問部屋を用意して TA に待機してもらい，その部屋に入室することで質問対応する方法も考えられるが，質問部屋の中に誰がいるのか，TA がどういう状況にあるのか，どんなことを喋っているのかなどを把握することができず，入室をためらってしまうという問題もある．さらに，オンライン上では学生間に物理的な距離に違いがないため，順番待ちを行ったり TA が順番を記憶したりするのは容易ではない．一方，TA も得手不得手があるため，質問を即座に解決することができなくて時間を浪費し，その結果精神的な負荷がかかるといった問題がある．さらに，質問の回答時間が伸びてしまうことで対応する効率が落ちてしまい質問待ちの学生を多く待たせてしまうことに繋がる．

以上より，講義時間内に質問をしたい学生と TA との間には，学生が質問を行うまでのためらいや TA の精神的な負荷，順番待ちの難しさなどの問題がオンライン時代において特に顕著に現れ山積しているため，講義を円滑に行うためにはこれらの問題を解消する必要がある．

1.5. プログラミング教育を円滑にするための 2 つの手法と目的

1.3 節で述べた，講師の指導が行き届かない問題を解決するために，苦手な学生を少なくする目的のもと，学生のタイピングの技量と基本的な命令の理解が不足していることに着目し，プログラミングを学び始める大学の学部 1 年生に対して，プログラミング特有のタイピングの練習に反復的に取り組んでもらうことと，よく使う基本命令の定着を目指した，プログラミング学習を円滑に進めるためのプログラムタイピングシステム typing.run を提案および実装する．ここで，単純にプログラムを写経する場合，今入力している文字列が一体何のためであり，どういう意味や働きがあるのかを把握できないと身につかないと考えられる．また，ひとりで黙々とタイピング練習するのは退屈であると考えられる．そこで，タイピング不要だが理解を促すコメントをタイピング対象のプログラム中に付与するとともに，1 行ごとにプログラムを逐次実行させることで書いた関数の挙動の理解を促進するための工夫や，タイピングスコアを計算することでハイスコアを目指すことや，ユーザ同士でタイピングスコアを競い合えることでやる気を高められるような仕組みを構築する．

1.4 節で述べた，学生が TA に行う質問のハードルが上がることと TA が精神的なプレッシャーを感じてしまうこと，質問順番の管理が困難なことに着目して，学生の質問へのハードルを下げつつ，TA の精神的負荷を下げることや適切な質問順番の管理を可能にすることを目的とするオンライン質問応答システム askTA を提案および実装する．ここでは，学生が TA に質問するハードルを下げるため，TA に対して通話やチャットで直接質問をするのではなく，システムに対して具体的な質問を投げかけ，学生のタイミングではなく，TA のタイミングで学生を呼び，質問の対応を行う．また，TA には，質問をシステム上で事前にチェックできるようにし，自身が対応可能である場合にシステム経由で学生を呼び出し，質問対応することによって，精神的負荷も下げることを狙った仕組みにする．さらに，学生が TA の対応を待っている間に自己解決した場合は質問を取り下げることができるなど，オンラインでの質問待ちを円滑にする仕組みを構築する．

本研究は，この 2 つの手法をそれぞれシステムとして構築することによって，COVID-19 によって生じたオンライン時代のプログラミング演習型講義に生じる主な 2 つの問題点を解決することを目的とする．具体的には，講義前後の予習・復習として，プログラムを理解しつつ，プログラミング特有なタイピングの練習に慣れてもらうことで，初歩的なタイピングミスや講義中にされる質問を減らす．しかし，これだけでは，講義中の質問には対応出来ないため，学生の質問ハードルや TA の精神的負荷，質問の円滑化に焦点を当てたオンライン質問応答システムを用いる．こうすることで講義前後のタイピング技量の強化と講義中の質疑対応の容易化の双方を支援することができ，プログラミングを学ぶ学生のオンラインでのプログラミング教育の円滑化を図る．

本研究では，実際のオンラインでの演習型プログラミング講義で運用を行うことによって提案手法の有用性を明らかにするために，以下の項目について結果をもとに分析を行う．

- 学生がタイピングスキルを獲得できるか
- 学生が関数を覚えられるか
- 学生が TA に質問しやすいか
- TA の精神的負荷を下げられるか
- TA への質問の順番待ちが適切に行えるか

1.6. 本研究の構成

本研究は，本章を含む全 6 章から構成される．まず本章では，大学初年次の演習型プログラミング教育にとりまく問題について触れ，オンライン時代を迎えたことで顕著化する講師の指導が行き届かないことの問題点と質問したい学生と TA が抱えるハードルに関する問題点の 2 つについて述べ，それぞれの提案手法について語った．これ以降，2 章では 2 つの問題点と提案手法に関連したプログラミング教育の関連研究に触れることで本研究の立ち位置を明確化する．3 章では，タイピングスキルと基本命令の定着に関する手法をタイピングの理解を促す側面と内発的動機づけ・外発的動機づけの側面から提案し，Web システム typing.run としての実装とタイピングする問題の開発, その利用方法について説明する．さらに実際のプログラミング演習で運用を行い分析することによって手法の有用性を明らかにする．4 章では，質問したい学生と TA のハードルを下げる手法について，質問する学生のハードルと TA の質問対応のハードルの両面からの提案と順番待ちを円滑に行うための提案との 3 つの提案を行い，Web システム askTA としての実装と利用方法について説明する．さらに実際のオンラインで行われた演習型のプログラミング講義で運用し分析を行うことで，提案手法の有用性を確認する．5 章では，提案手法とその結果から明らかになったことについて総合的な考察と COVID-19 による本研究の運用について触れ，応用と今後の展望について説明を行う．最後の 6 章では，本研究のまとめを行う．

第2章 関連研究

本研究では，オンラインでのプログラミング演習型講義について 2 つの手法を提案しているため，それぞれの手法に関連する研究を 2.1 節と 2.2 節で述べる．これにより，本研究の立ち位置を明確化する．

2.1. 継続的なプログラムのタイピング練習に関する研究

2.1.1. 一般的なタイピング練習に関する研究

タイピングそのものの練習を支援する研究は多く行われている．Kollie ら[6]は，人間工学的に正しく早くタイピングするために追跡を行い，フィードバックを提供するシステムを構築している．Hasegawa ら[7]は，タイピング練習時間を短縮する目的で，JINS MEME に搭載されているセンサ情報からキーボードを見ずにタイピングできたかの判定が可能かを試みている．その結果，約 90%の精度で推定可能なことから，タイピング支援アプリケーションに応用可能であるとしている．

2015 年に文部科学省が行った情報活用能力調査[8]によると，日本の小学生のタイピング文字数は 1 分間に平均 5.9 文字，中学生は 1 分間に平均 17.4 文字と報告されている．それに伴い，キーボードに初めて触る小学生を対象とした研究も多くされている．堀田ら[9]は，検定試験の要領で 30 の級を作成し，キャラクタを用いたゲーム仕立てのタイピング練習ソフトを作成した．小学校で 4 年間の運用を行い，最終的に全国の 36.8%の小学校が登録して活用しており，短文入力の敷居が高いことや，学年が上がるにつれて上位級の分布が増えていることを報告している．さらに，中村ら[10]は，個人レベルにあったモデルを構築し指導することや，小学校の国語の教科書に出てくる単語や習った漢字のみ使用すること，進捗を教師が確認することができるシステムを実装し，広島県の小学校で運用を行った．その結果，正打率の向上や適切な教師の指導が行えることを確認している．また，Marjolijn ら[11]は，小学生を対象にタイピング教科書を用いながら 2 週間毎に 90 分のタイピングトレーニング教室を全 15 回実施し，家庭では毎日 20 分程度のタイピング練習を課した．トレーニングを行ったグループと行わなかったグループのタイピング，スペリング，物語の記述力を比較した結果，トレーニングを行ったグループの方が，タイピングだけでなく，スペリングや物語の記述力も向上したことを示している．

大学生を対象としたタイピング練習として，吉長ら[12]は，ひらがなでのローマ字入力のタイピングテストを保健福祉学部に通う大学生 78 名に対して 11 回実施した．その結果，主観的なタイピングの慣れ具合と，客観的なタイピング速度の変化から学生を 3 つの群に有意的に分けることができ，目標をそれぞれ定めることでタイピングを継続できるとして

いる．さらに松山ら[13]は，文系学部のコンピュータの講義において毎回 20 分程度のタイピング練習を行う 2 クラスと行わない 1 クラスを比較したところ，練習を行った 2 クラスではタイピング力が向上したことを確認している．タイピング練習の時間を確保するために前半の講義の進度が遅れるが，タイピング力の向上によって，進度を取り戻すことができたとしている．

これらの研究は，タイピングの行為そのものや練習を支援するものであり，本提案手法も併用やこれらと同様な運用方法を行うことで，効率的に習得できる可能性がある．本研究では，講義前の予習のノルマとして学生にタイピングを課した．

2.1.2. プログラミングとタイピングの関係に関する研究

Miura[14]は，大学の必修プログラミング講義において，学生のタイピングスキルと成績の相関が有意に高い年があったこと 4 年間分のデータから分析し，報告している．その上で，タイピングによる個人差が講義に影響しないように自動補完機能を実装したエディタを用いることで，カッコの入力数や記述ミスが低下したことを明らかにしている．Thomas ら[15]も，プログラミングのパフォーマンスとタイピング速度の間に相関があることを，複数の実験結果から示している．また，Leinonen ら[16]は，プログラミングの技量をタイピングのキーストロークの時間間隔から求める研究を行っており，精度は十分ではないものの技量を時間間隔から求めることができる可能性を示唆している．

これらより，プログラミングにとってタイピングが重要であることが言え，本研究で提案するタイピングシステムによって単純にタイピング速度を向上できるだけでも，プログラミングの上達につながることが期待される．

2.1.3. プログラミング講義内でのタイピング練習に関する研究

プログラミング言語のタイピングと，大学での演習授業に着目して行われた研究に中田[17]の研究が挙げられる．この研究では，C 言語のプログラムを用いて授業前に毎回タイピング練習することによって，成績やタイピング速度との関係性を複数年に渡って調査している．その結果，タイピングの実施自体に成績の向上は見られないものの，成績とタイピング速度に一定の相関があることを明らかにしている．また，喜多ら[18]は，写経型プログラミングを提唱しており，自著の本[19]を使った反転授業を行った報告をしている．ここでは，学生に本の内容に沿って授業時間外に C 言語のプログラムを写経させて，その進捗を提出させている．また授業時間内では，クラス全員の進捗をグラフで示して学生間の学習状況を比較させている．この授業スタイルを行った結果，最終課題の提出物から一定の成果を得ることができたとしている．ここで，TYPOS[20]は，学生のタイピングスキルを向上させる目的で実装され，基本的なプログラムを左のプログラムのスクリーン画像から右のプログ

ラミングエディタへ写経する Web システムである．実際の講義の中で，システムの使用を強制せず，学生の参加意思によって実験を行っている．システムに準備された全課題に取り組んだ学生とそうでない学生を比較した結果，全課題に取り組んでタイピングを行った学生の方が，演習中のビルドの失敗が少なく最終成績が良いことを報告している．また，Leinonen ら[21]は，学生がシンタックスエラーに苦しむことを軽減しつつプログラミングの基礎を学ぶために，プログラムの重要な構文だけ（例えば for 文だけ等）を穴あきのプログラムに追加する形で写経し，間違った場合は，その文字を赤色でハイライトする手法を提案している．この手法を講義の直前に行ったグループと手法を行ってないグループとを比較した結果，有意な差はなく，構文だけのタイピング練習を演習の直前に行うのは意味がなく，必要ない可能性を示している．

これらの研究の目的は本研究と類似しているが，関数などのプログラムの理解を促す仕組みの点で異なっており，さらに，プログラミングのタイピング練習と同時に，そのプログラム自体の意味や挙動を理解させることに重きをおいている．

2.1.4. プログラミングの動機づけに関する研究

プログラミングを楽しく学ぶために，動機づけに着目した研究も多くされている．岡崎ら[22]は，小学生のプログラミング教育において，講義型，2 人 1 組で教え合う協同型，個別型の 3 つの体験形式ごとの動機づけを調査したところ，講義型と協同型でプログラミングを学習すると動機づけは有意に上昇し，個別型では上昇しないことが示唆されたとしている．また，坂本ら[23]は，絵文字や日本語をベースとしたプログラミング言語を用いながらひよこのキャラクタを操作してダンスさせるスマートフォン上の教育ツールを開発し，中高生を中心に実験を行っている．その結果，プログラミング学習の動機づけ，プログラミングの印象改善，理解の促進の効果が見られたとしている．大学におけるプログラミング教育において，松本ら[24]は，アルゴリズムをゲーム感覚で体験可能なアルゴロジック[25]をプログラミング学習前に動機づけとして用いることで，アルゴリズム的思考に慣れさせることが重要であると考えており，ゲームの理解度と学習の到達度の分析から相関があることを明らかにしている．さらに，佐々木ら[26]は，学生が興味を持つ題材を演習に取り上げることが動機づけには効果的であるとして，学生自身が興味のあることを選択しながらプログラミングを学べる授業形式を提案し，認定試験を設けることで最低限の学力を担保しつつ学生の動機づけに成功したことを報告している．

以上より，プログラミングを楽しく学ぶ動機づけに着目して教え方や事前に学習を行う手法の研究が行われている．本研究も学生に予習として課しながら，ハイスコアやランキングで他人と競い合うことでプログラミングを学ぶ動機づけを図っている．

2.2. 質問したい学生と TA 間のコミュニケーションに関する研究

2.2.1. プログラミング指導支援に関する研究

大学におけるプログラミング演習型講義の支援に関する研究として著者[27]は，これまでに TA と学生の抽象的思考に着目した指導を楽にするための手法を提案し支援を行ってきた．しかし，これまでの研究は対面講義を想定した TA と学生を支援するものであり，オンライン講義についての検討は不十分である．また，井垣ら[28] は，課題に対する取り組み速度などを利用し，サポートを必要とする学生を早期に検出できるシステムを提案している．市村ら[29]も，学生の操作やエラーログから，講義中につまずいている学生を早期に発見するとともに，共通する問題点を明らかにするシステムを提案している．さらに加藤ら[30]は，模範解答との比較により，問題を抱える学生の学習状況を教員がリアルタイムに把握できるプログラミング演習向け授業システムを提案している．また，Yan ら[31]は，プログラムをどのように学生が編集したのかの過程を可視化することによって，講師が指導を行いやすくなる手法を提案している．

また，システムが自動的にアドバイスやフィードバックを行うことで講師や TA の負担軽減を目指す研究も存在している．Hattori ら[32]は，プログラムのコンパイル時にコンパイラが静的なエラーを吐くだけではなく，その先のユーザや状況も考慮した易しいアドバイスを享受する教育指向 Java コンパイラを提案しており，教師や TA の負担軽減に繋がるとしている．Marceau ら[33]も，初学者向けの丁寧なエラーメッセージを提示するために，エラーを修正するときの編集履歴を分析することで活用する手法を提案している．Petcha[34]は，e ラーニング上で，学生が提出したプログラムを自動で採点する機能に加えて，TA の代わりとなるようにプログラムのエラーバターンやフィードバックを提示する機能を提案している．この研究では一定の効果が示せたが，人間の TA には劣るため TA が対応できない状況に活用できるとしている．ArTEMiS[35]は，大規模な人数で行う演習講義においてオンラインエディタから提出された学生のプログラムを自動的に評価しフィードバックを返す目的で提案されている．その結果，初学者に適しており，徐々に学生のスキルを改善可能なことから TA の労力削減と学生の学習体験の向上ができるとしている．

プログラミングのコースの脱落を防ぐために学生が学び始めた初期のデータから最終成績や脱落を予測する研究も行われている．Munson ら[36]は，学び始めた 3 週間のプログラムにかける時間やコンパイル時のエラー割合などと最終成績において相関があることを示している．また，Pereira[37]は，コースを受講する学生の脱落を受講し始めた 2 週間のプログラミングのキー操作ログ等から，82.5% の精度で予測できたとしている．Vihavainen[38]も，プログラムの編集履歴からインデントのミスや提出にかかった時間などからコースに合格できるかを 78%の精度で予測している．さらに，Ahadi ら[39]は，学生が

プログラミングを学習し始めた 1 週間のソースコード編集履歴から高い精度で成績が良くなる学生とそうでない学生の予測を行っている．

これらは，学生の理解状況を把握したり，TA の指導を楽にしたりするものであり，本研究とは別のアプローチではある．しかし，システムを運用する方法やシステムが学生に通知するエラーメッセージの点などにおいて，これらの研究と組み合わせることにより講義をよりよくすることが可能になる．

2.2.2. プログラミングのリアルタイム共有と指導に関する研究

Goldman ら[40]は，2011 年にブラウザでリアルタイムにプログラムの共有と実行を備えた Collabode を提案し，アウトソーシング，ペアプログラミング，リモートクラスの 3 つの場面での運用について報告している．しかし，近年では，Google Colaboratory [41]や Visual Studio Live Share[42]などによって，リアルタイムにプログラムと実行環境を他者と共有することは容易になったが，プログラミング教育とその指導の観点からは不十分である．そのため，リアルタイムにプログラムを共有しながら指導に活かす研究が多くされている．Vandeventer[43]は，学生の進捗状況をリアルタイムで教師が観察できることやプログラム内の指定の箇所をリンクとしてメッセージを教師と共有できること，ログを再生できることなど，大学のプログラミング教育を支援する目的で作成されたシステムを提案している．また，Miller[44]らは，シンプルな操作で使用でき，かんたんに様々な OS にインストールが可能で，ステップ実行や変数のデータ構造を可視化可能なデバッガを備えた jGRASP を提案している．講義で使用した結果，デバッガが TA の支援になり，さらに学生もデバッガの使い方を習得したとしている．さらに Trong ら[45]は，プログラミングを学ぶ初学者を対象としたメタモデルを示した上で，CIDE システムを実装し提案している．

学生同士のチームでプログラムを共有しながら課題に取り組むことを目的とした研究もある．Park ら[46]は，ピアアセスメントと呼ばれる協調学習の評価手段においてコードの編集履歴を可視化するシステムを提案し，それを見ることによって自他ともにプログラム構造や意図の理解が深まることを明らかにしている．加えて，動画再生のようにプログラミング環境上でプログラミングの講義を受けることができ，再生，停止，シーク，書き換えが可能なシステム[47]も提案している．Čubranić ら[48]は，学生 2 名がそれぞれ作成しているプログラムの共有とチャット機能を設けた手法を提案し実装することによって，機能の有用性とコミュニケーションに必要な要件を明らかにしている．また，Codechella[49]は，プログラムをチュータや友達と一緒に書き，実行の変化について対話することが重要だとして，オンラインでプログラムをリアルタイムで共有しつつ変数の可視化が可能なシステムを提案している．9 ヶ月間の公開で 40 カ国 296 都市の人々が 299 回使用し，その 57%が複数都市間での共有であったと報告している．

以上のように，プログラミングをリアルタイムで共有しつつ指導を行う研究がされているが，本研究では，オンラインで TA に質問し，TA がリアルタイムで回答することを前提としたシステムを提案する．

2.2.3. 消極的なコミュニケーションに関する研究

栗原ら[50]は，消極的な人でも快適に使える情報技術のデザインを「消極性デザイン」と呼んでおり，研究会活動も行っている．また西田ら[51]は，実名と匿名の長所を併せ持つ傘連判状を用いたコミュニケーションプロトコルを提案しており，意見に圧力のかかる状況でも発言が可能であることなどを報告している．さらに，学会での交流を促進する目的で，近くになりたい人の希望を反映した席決めシステム[52]も提案し，実践的な運用を行っている．

演習型講義の TA は，学生にとって同じ大学の先輩という目上の立場であることや，初歩的な質問をしたら恥ずかしいというような学生が消極的になりやすい環境であり，質問したくてもできない学生やわからないことを放置する学生がいると考えられる．また，TA 自身が元々内気な性格の場合に学生に適切で十分な指導ができないことも考えられる．さらにオンライン環境では，よりコミュニケーションが取りにくくなるため，大きな問題となりうる．本研究もこれらの研究のように消極的になってしまう学生や，もともと内気な性格の TA がオンライン環境でも快適に使え，質問をスムーズにやりとりできる手法を提案しシステムとして実装する．

第3章　タイピングスキルと基本的関数の定着手法

3.1. 提案手法

プログラミングを学び始めた初学者が，モチベーションを保ったままタイピング練習やプログラムへの理解を深めるようにするには，内発的動機づけと外発的動機づけ，そして理解を促すための各種の仕組みが重要になる．そこで，プログラムタイピングシステムの実現にあたり，その理解を促す仕組みと，動機づけを行う仕組みについて述べる．

3.1.1. プログラミングの理解を促す仕組み

プログラミングを理解するためには，ただ提示されたプログラムをタイピングするのではなく，命令を理解しながらタイピングする必要がある．ここで，タイピング対象となる問題のプログラムをただ提示するだけでは，理解にはつながらない．事前にその文字列が何を意味しており，その入力がどういった影響を及ぼすのかをユーザが学習してくれればよいが，そうしたことを多くのユーザ（講義においては学生）に要求するのは難しい．そのため，自身が入力している文字列が何を意味し，その結果がどのような影響を及ぼすかをそれとなく理解させる仕組みが重要になる．

そこで，ユーザがタイピングしている行の文字列（プログラム）の 1 行上に，タイピング対象とはしない説明コメントを提示し，知識として理解を促す．ここで，コメント行は色を変えるとともに，前のプログラムを入力した後にコメント行に来た場合には，自動的にコメント行の後の行にスキップすることでコメントを無視することを可能とする．

また，ユーザの入力に応じて徐々にプログラムを動作させることによって，ウィンドウ作成にまつわる行を入力するとウィンドウが現れ，背景指定にまつわる行を入力すると背景の色が変わり，円を描画する行を入力すると円が描画されるなどといったように，タイピング中の内容を視覚的に把握可能とする．なお，本来入力途中の場合は，閉じカッコがないなどによりプログラムは動作しないが，内部的に補完することによって入力途中でもプログラムを実行することを実現する．ここでは，1〜2 行の入力で視覚的情報が徐々に変化していくように，タイピング対象の問題にも工夫を行う．

一方，プログラミングにおいて利用される関数には，様々な引数をもつものが多い．ここで，タイピング練習では，その引数として与えられる値が固定されているため，最後まで入力したとしても，タイピング用に指定された引数の時の挙動しか見ることができない．このような試行錯誤ができないことは，プログラミング習得においてはデメリットである．そこで，タイピング練習後にそのタイピングしたプログラムを編集し，円の大きさや色を変更し

たり，表示順番を変更したりするといったように，関数の引数や記述の順番などを気軽に変更可能とすることにより，理解を促す．

3.1.2. 内発的動機づけと外発的動機づけ

プログラミングに限らず，タイピング練習は繰り返し行うことが重要である．そのためには，内発的動機づけと，外発的動機づけを考慮する必要がある．

まず，内発的動機づけについては，1 分間にタイプできた文字の数を算出する指標を Character per Minute（CPM）と定義し，それをタイピングのスコアにすることで，各問題における自分のタイピング速度を指標として見ることができるようにし，少しでも良い自己ベストの CPM を目指せるようにする．また，CPM ではタイピング速度を早めることに注目しているため，タイピングの正確性を高めてもらうために，タイピングの打ち間違えたミスの数に関する情報も提示する．

一方，外発的動機づけとして，多くのタイピングシステムでは他者のスコアを提示したり，ランキングを提示したりといった工夫をしている．そこで，本システムでも同級生の CPM と自分の CPM とを比べることができるようにすることで競争心を煽り，モチベーションを維持できるようにする．ここで，ランキングはある程度興味をもったユーザに対しては効果的ではあるものの，もともとタイピング速度が遅いようなユーザにとっては，負の効果になり，まったく取り組まない可能性もある．そこで，講義の予習のノルマとしてタイピング回数を取り入れることで，最低限のタイピングを実施してもらい，そこから興味をもったユーザにランキングなどへの挑戦を促す．

3.2. **typing.run**

多くの大学の初年次プログラミング講義で採用されている言語である Processing を対象とし，提案手法を用いたシステム typing.run[3] を実装した．

3.2.1. 実装

システムは，誰もがアクセスしやすいように Web アプリケーションとして実装した．実装には JavaScript, MySQL 等を用いた．サーバーサイドは Node.js と MySQL 等で構築し，タイピング問題の配信やユーザのタイピング結果の保存などを行った．クライアントサイドは Processing.js [53]を利用しつつ，JavaScript で実装し，取り組む問題を選択するページ

[3] https://typing.run/

図 1 タイピングする行の上に入力しないコメントを表示し，
カッコを内部補完しながらタイピングした行から実行される例

とタイピングを行うページを用意した．また，大学から発行されているメールアドレスの認証システムと紐づけることにより，ユーザの識別を行った．

また，3.1.1 項でプログラムの理解を促す仕組みにおいて説明した通り，各行端ごとにタイピングされたプログラムの入れ子構造が途中だった場合に，強制的に不足している個数分のカッコを内部的に挿入することで，各行の逐次実行を実現した．

図 1 は，実現したシステムを利用してタイピングを行い，そのタイピングした行に応じてプログラムが実行されている様子である．このプログラムでは，setup 関数内に書かれた size の行を入力後にウィンドウが作成され，次に background の行を入力後に背景が白色で描画され，ellipse の行を入力後に円が描画されるものとなっている．本来なら，setup 関数内のすべてをタイピングして setup のカッコを閉じるまで（「}」を入力するまで）描画されないが，内部処理的にカッコを補完することで逐次実行を行っている．

3.2.2. タイピング問題の開発

表 1 セクションごとのタイトルと，その問題数

セクション番号	セクションタイトル	問題数
第 1 回	イントロダクション	4 問
第 2 回	アニメーションと変数	8 問
第 3 回	計算と変数	10 問
第 4 回	条件分岐	10 問
第 5 回	多重の条件分岐	8 問
第 6 回	繰り返し	8 問
第 7 回	多重の繰り返し	7 問
第 8 回	配列	9 問
第 9 回	多重配列	7 問
第 10 回	関数	6 問
第 11 回	関数の復習	6 問

プログラミングについての知識がまったくない初学者に対して，いきなり繰り返しや配列といった知識が必要な問題を提示することは適切ではない．適切な学びを得るには，理解の度合いや講義の進度と合わせてタイピング課題に挑戦できるようにすることが重要であると考え，タイピング問題も講義に連動するようにセクションごとに分割した．実際に分割したセクションごとのタイトルと問題数を表 1 に示す．

各問題の作成においては，講義の予習となるように，事前に配布している予習資料[4]にあるプログラムを積極的に採用するとともに，問題数を 4〜10 問となるように調整した．ここでは，プログラムをできるだけ意味のあるものとするため，決め打ちの値を入力するのではなく，変数名をわかりやすい単語などに変更した．また，学生のモチベーションを考慮し，あまりに複雑なプログラムや，入力行数が多いプログラム，表示結果が面白くないプログラムなどは採用しなかった．一方，プログラムを 1〜2 行入力するだけで徐々に提示されるようにするため，プログラムの記述順番などを工夫した．

なお，システムの逐次実行の都合上，while を使用すると終了条件を満たすためのプログラムが書かれる前に実行されてしまい，無限ループとなり応答不能になるため書き方に制限があった．そのため，提示するプログラムでは while の部分入力を行った際に無限ループにならないように，適切な処理を行った．この無限ループに関する制限については，今後改善を図っていく予定である．

上記を考慮して，プログラミング演習 I の 11 回分の講義に対して，合計 83 個の問題を作成した．

3.2.3. 利用方法

本システムは，大きく分けてセクションごとの問題を選択する問題選択ページと，実際にタイピングを行うタイピングページに分かれている．

図 2 の問題選択ページでは，取り組む問題の選択や自己ベスト CPM の確認，セクションごとにランキングの確認，他者のハイスコア CPM とタイピングした回数の関係性を示した表などを確認することができる．また，取り組む問題のタイトルをクリックすることで，タイピングページに遷移される．

図 3 のタイピングページでは，左側にタイピングする問題の提示，右側に実行画面，下部にコンソールを設けている．ここで入力前の文字列はすべて灰色文字になっているが，タイピングしていくと予約語は青色になり，その他は白文字になるようにした．ユーザがタイピングしている文字を示すキャレットは，灰色のマーカに黒文字で表現した．なお，入力不要のコメントは緑色にした．下側に並んだボタンでは，プログラムの編集や実行が可能なモードやクリップボードにソースコードをコピーできる機能が利用できる．また，実行画面の下ではタイピングを初めてからの経過時間，現在の CPM やタイプミスの回数などを確認することができる．

[4] https://nkmr.io/lecture/

図 2 問題選択ページのスクリーンショット

図 3 タイピングページのスクリーンショット

3.3. 運用と分析

前節で述べた typing.run を 2020 年 5 月 6 日より公開し，プログラミング演習 1 で運用を行った．なお，運用期間は 2020 年 6 月 22 日から 2020 年 7 月 27 日までの毎週月曜日と水曜日であり，タイピング課題は講義開始前までのノルマとした．

学生には, 授業開始期間前の typing.run 公開直後から授業で使用することの周知を行い，練習やデバッグを兼ねての利用を呼びかけた．

3.3.1. タイピング結果と分析

運用で収集したデータをセクションごとの授業開始前までにタイピングされた 118 名分，タイピング 88,293 回分のデータを集計した．なお，タイピングにおいてマクロ等を使った不正行為が疑われるデータや，タイピング時間が 30 分を超えていて放置されたと思われるデータは分析から除外した．結果的に，88,055 回分のデータを分析対象とした．ここで，全 83 問のタイピング問題を横軸にとり，指標ごとに折れ線グラフで比較した結果を図 4〜7 に示す．

図 4 は，問題ごとのタイピング回数の変化を示しており，初めの方は 20 回以上タイピングに取り組んでいたが，25 問目を過ぎた辺りから 5 回程度になっていることが分かる．1 問目の回数がかなり多いのは，プログラムが短く，速度を競うために何度もタイピングした学生が多いためである．図 5 は，問題ごとの各学生の最速 CPM の平均を表したものである．全体を通して CPM の平均値が 200 付近になった問題が多かったことがわかるが，この結果だけを見ると全体として入力速度に変化はない．図 6 は，問題ごとに最速 CPM だった時のタイプミス平均を算出したものである．この結果から，後半の問題になればなるほどタイプミスが増えていることがわかる．これは，問題の文字数や，カッコやセミコロンなど複雑な表記が増え，結果的に難易度が上昇したことが原因であると考えられる．図 7 は，問題ごとの最速 CPM の標準偏差の推移を示したものである．この結果より，問題番号が上がっていくにつれて，学生のタイピングの速さに差がなくなっていることがわかる．図 8 は，各学生がタイピングしたすべての回数を 50 分割し，その分割された区間ごとの CPM の平均を求めたものである．図 5 ではタイピング速度の伸びは観測されなかったが，図 8 の結果より，後半になるにしたがってタイピング問題が難しくなったにも関わらず，全体として CPM の平均が上昇していることがわかる．

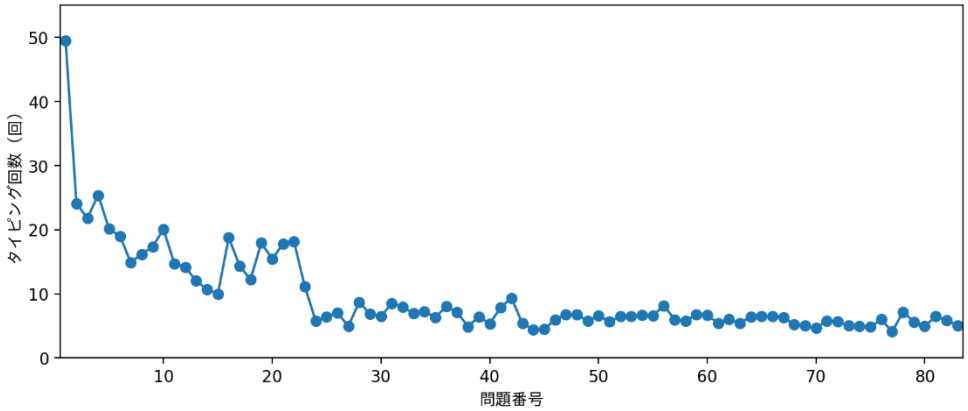

図 4 問題と取り組み回数の平均

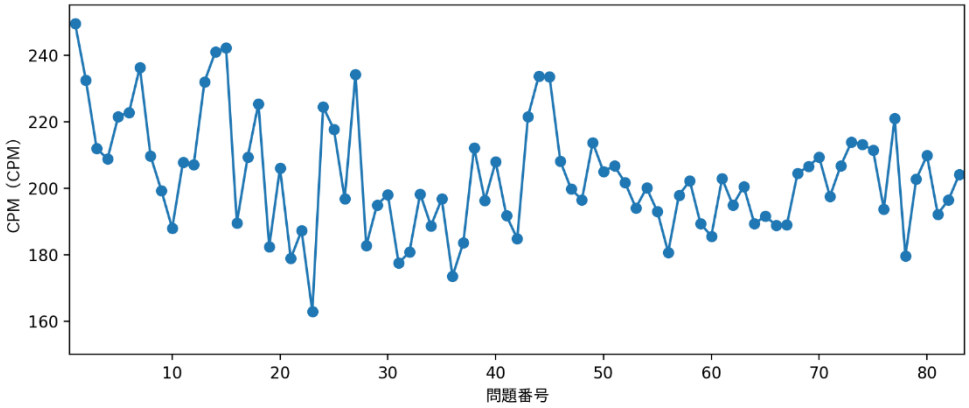

図 5 問題とハイスコア CPM の平均

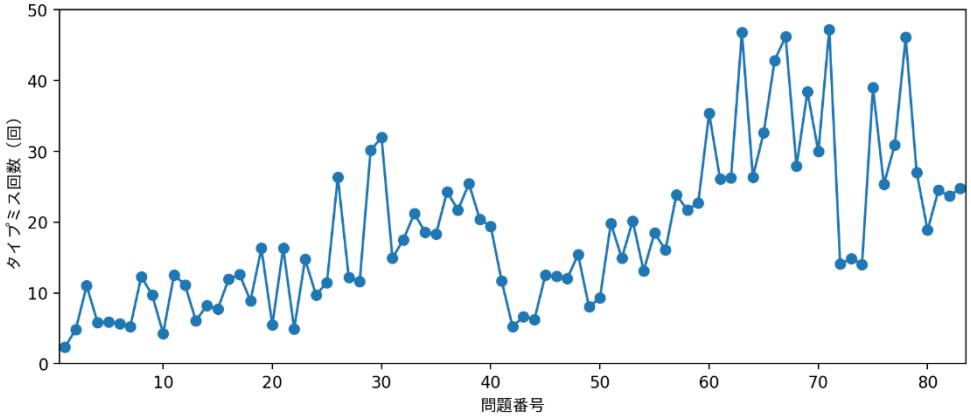

図 6 問題とタイプミス回数の平均

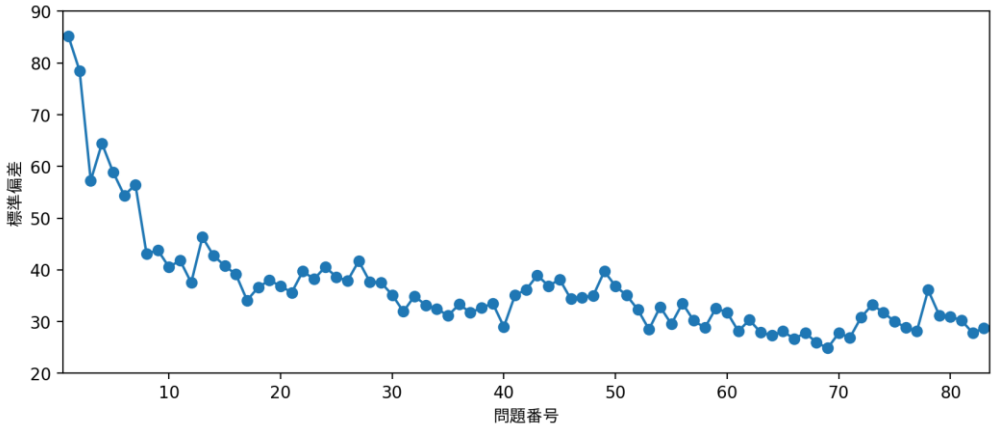

図 7 問題とハイスコア CPM の標準偏差

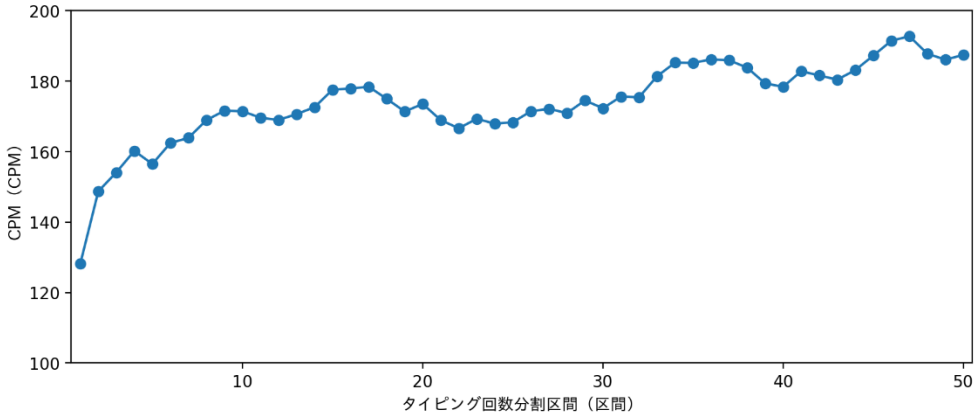

図 8 タイピング回数を系列順に 50 分割した区間と
それぞれの区間における全員の CPM の平均

3.3.2. アンケート結果と分析

講義の第 11 回目の開始時（タイピング課題に取り組んでもらった中では最終回）に，学生に対して typing.run に関するアンケートを実施し，回答してもらった．集まった 111 名分を集計したアンケート結果を表 1 に示す．なお，アンケートでは 5 段階（-2:まったく当てはまらない〜2:とても当てはまる）で評価してもらった．

表 2 の Q1 と Q2 の比較により，使用前と使用後のタイピングの得意度の変化を見ることができる．この結果より，使用後に評価が高くなっていることが分かる．また，提案手法に関する Q3，Q4，Q6，Q7，Q8 でも評価の平均値が正の値になった．しかし，Q5 のプログラムの変更ができる機能に関する評価の平均値は，負の値になった．

次に，学生ごとに Q2 で回答した評価値から Q1 で回答した評価値を引くことで，どのくらいの段階数でタイピングが得意になったかを計算した結果を表 3 に示す. この結果より，1 段階以上あがった学生が約 71%となっていたことがわかる．

また，アンケートでは，システム使用以前のプログラミング経験の有無も聞いていた．そこで，システム利用のアンケート結果がプログラミング経験の有無によって違いがあったかを，Q1 から Q8 の質問項目ごとに評価値の平均を求めたものが表 4 である．この結果より，Q1 や Q2，Q6 では両者の間に大きな差があったことがわかる．

次に，typing.run を利用して身についたことや身につかなかったことなどの，ポジティブな意見とネガティブな意見を自由記述によって収集した．

ポジティブな意見では，「タッチタイピングができるようになった」「記号のタイピングが早くなった」などのタイピングに関するものや，「予習資料の理解が深くなった」「関数の働きがわかりやすかった」「コメントがわかりやすかった」などのプログラムの理解に関するもの，「自然に手が関数を覚えた」「ランキング表示で意欲が湧いた」「実装のアイディアが吸収できた」「予習のハードルが下がった」などの意見があり，65 件集まっていた．

一方，ネガティブな意見では，「スペースや記号を自分の好きなタイミングで打てない」といったプログラミング上級者からの意見や，「タイプミスを気にしない癖がついた」「スピード重視になって内容理解をおろそかにしてしまった」「タイピングが速いとプログラムができる気になってしまった」などのタイピングシステムであるからこその問題に関する意見が 45 件集まっていた．

表 2 アンケート項目とその分布と平均

質問項目		評価値の分布					平均
		-2	-1	0	1	2	
Q1	授業開始前時点でタイピングが得意でしたか？	58	26	11	14	2	-1.12
Q2	授業終了後時点でタイピングが得意ですか？	11	24	35	35	6	0.01
Q3	同級生間のランキングや分布のグラフは意識しましたか？	12	36	11	43	9	0.01
Q4	1 行タイピングするごとに実行されることに意識しましたか？	8	24	15	42	22	0.41
Q5	プログラムを変更できる機能を使用しましたか？	43	36	6	21	5	-0.82
Q6	タイピング中にプログラムの内容をどのぐらい理解していましたか？	6	26	14	55	10	0.33
Q7	利用することで関数名を覚えることができましたか？	2	5	11	51	42	1.14
Q8	新しいことを学ぶ時に今後も利用したいですか？	4	2	13	44	48	1.17

表 3 タイピングの得意に関する評価値の差の件数（人）

	-4	-3	-2	-1	0	1	2	3	4
Q2-Q1	0	0	0	6	26	40	27	11	1

表 4 プログラミング経験有無による質問の評価値の平均

	経験なし(89 名)	経験あり(22 名)
Q1	-1.39	0.00
Q2	-0.12	0.55
Q3	0.01	0.00
Q4	0.39	0.50
Q5	-0.90	-0.50
Q6	0.12	1.18
Q7	1.12	1.18
Q8	1.20	1.05

3.4. 考察

3.4.1. タイピング結果からの考察

図 4 より，序盤の問題では，タイピングに取り組む回数が多かったが，次第に減っていることがわかる．これは，序盤の問題ではプログラムが短いため，短時間で終わることもあり，学生が何回も取り組んだからだと考えられる．また，後半に一定の取り組み回数に落ち着いたのは，予習のノルマ回数分だけ取り組んでいたためだと考えられる．ノルマ回数は，問題ごとに，高いハイスコア CPM を出した場合はノルマ回数を少なくし，低いハイスコア CPM の場合はノルマ回数を増やした．図 9 は第 5 回セクション時のハイスコア CPM とタイピング回数の関係を示したグラフである．150CPM 以下の丸が 10 回付近，150CPM から 175CPM までが 8 回付近，175CPM から 200CPM までが 6 回付近と階段状に分布が偏っていることからもノルマ回数分だけ取り組んでいた学生が多かったことが読み取れる．

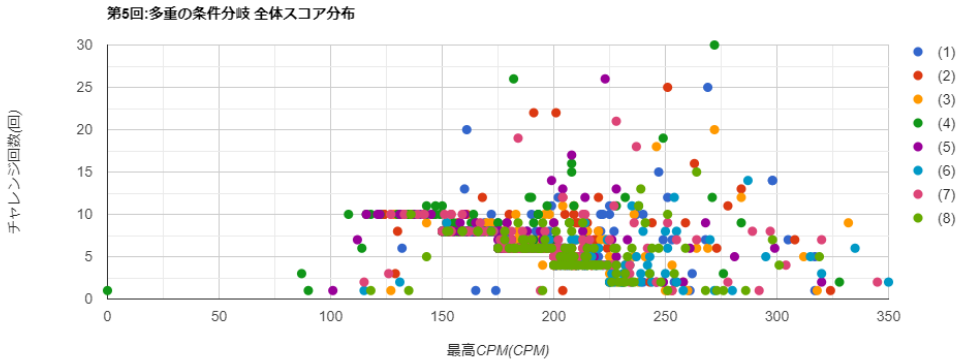

図 9 第 5 回セクションのハイスコア CPM とタイピング回数の関係を示すグラフ

(1 つの丸が 1 人，色の違いが問題番号を示す)

全 83 個の問題では，プログラムの長さが異なっていたり，記号などの複雑なタイピングの数が異なっていたりなど，問題自体の難易度が異なるため，図 5 や図 6 では，グラフの乱高下が激しくなっている．しかし，図 7 より学生内でのスコアの分布の広がりを示す標準偏差がしだいに下がっており，このことから，上位層と下位層のタイピング速さの差が縮まったと言え，下位層の底上げができていることが分かる．

また，図 8 より実施回数が増えるにつれ，CPM の平均値が伸びていることが確認できた．問題の難易度が上がっているのにも関わらず，タイピング速度が向上していることから，全学生のタイピング速度が次第に早くなり成長していると言える．

3.4.2. アンケート結果からの考察

表 2 のタイピングの得意度に関する Q1 と Q2 より，システムを使ったことで，得意になった学生が多くいたことが分かる．また，提案手法で述べた，外発的要因の意識が Q3 に，プログラムの理解促進が Q4 に対応し，それぞれ評価値の平均値が正の値になったことは，提案手法が有効である傾向が出ていると言える．一方，Q5 のプログラムの変更機能を使った学生は少なかった．これは，そもそもその機能に気づいていないことや，講義の予習で変更する時間的な余裕がなかったことなどが原因と考えられる．今後の継続利用に関する Q8 についても高い評価値になったことに加えて，自由記述でもポジティブな意見が多く集まったことから，システム全体的な満足度が高いことが分かる．なお，自由記述におけるネガティブな意見の大半は，写経することに重点を置いたことでの弊害であるといえる．これらの問題については，タイピング速度を競わないモードの導入や写経ではなくプログラムを 0 から書き上げる速度を競う上級者モードの導入というように実装を工夫することで解決することができると考えている．

表 3 の結果より，システムを使う前よりも後の方が，評価値が下がった学生が 6 名いた．この 6 名について，他の回答を分析したところ，評価値が極端に低いということはなかった．また，自由記述を見てもタイピング速度が低下したことに対して記述されてなかった．このことから，システムに対して不満を持っているとは考えられない．この点については，自身の成長より他者の成長の方が早く，相対的に自身が遅くなったというように，自身の絶対評価ではなく，他者との相対評価が影響している可能性がある．

表 4 の Q1 および Q2 の結果より，まずプログラミング経験あり群の方が，プログラミング経験なし群よりもタイピングが得意であることが分かる．また，Q3，Q7 については，プログラミング経験の有無によって評価が変わることはないと言える．しかし，プログラミング経験あり群では，Q4 の実行時の意識が経験なし群より高く，Q8 の継続利用については経験なし群より低くなっていた．プログラミング経験あり群は，プログラミングでは動作確認をひとつひとつすることが大事だと身をもって知っているため Q4 の評価が高くなったと考えられる．Q8 については，自由記述に「描画する関数から書き始めたい」や「開きカッコを書いた後に閉じカッコをすぐに書きたい」，「スペースの記法が違う」など既に自身のタイピングスタイルを確立していたり，癖があったりすることにより，システムでの写経により矯正されることが苦痛と感じているため，評価が低くなったと言える．なお，本研究はプログラミング初学者を対象としており，既にプログラミングを行ったことがある学生は本来対象ではないため，システムの本質的な問題ではなく，講義の運用上の問題であると言える．

一方，「タイピングが速いとプログラムができる気になってしまった」という記述がネガティブな意見であげられたのは，想定外であった．これは，プログラミングが苦手なのにも関わらず，タイピングが高速にでき，またプログラムが組みあがったことによって，出来て

いると勘違いさせてしまったことが原因と考えられる．この点については，講義中に説明を行うことで，導くことができると期待される．また，下位の学生までの全員のランキングを掲示していたため，「順位が上がらずやる気を無くす」や「萎えた」などの意見もあった．この問題については，スコアの計算の仕方や，他者に共有されるランキングを制限するなど，学生に提示する内容に工夫を加え実装することで，解決することが可能である．

3.5. オンライン講義での運用の知見

3.5.1. オンライン講義での運用の所見と分析

typing.run の運用を行ったプログラミング演習 I は，オンライン講義で実施されたが，プログラミング特有のタイピングや初歩的な命令，記号の用法に慣れてもらったことによって，講義中の課題を解いてもらう際に，初歩的なタイプミスによって生じる問題は例年に比べかなり少なかった．また，例年とは実施形態が異なるため単純に比較することは難しいが，例年と同レベルの課題について，その達成度は高かったことから，typing.run を取り組んだ効果が出ていると考えられる．

図 10 は，typing.run が 1 日でどのような時間帯に利用されたのかをグラフ化したものである．講義終了直後の 16 時ごろに最も多く行われており，講義が始まる直前の 11 時ごろや 23 時ごろにも多く取り組まれていた．また 4 時〜7 時を除いた時間帯でも幅広く利用されていることから，学生は暇を見つけては取り組んでおり，取り組みやすいものであったことがわかる．

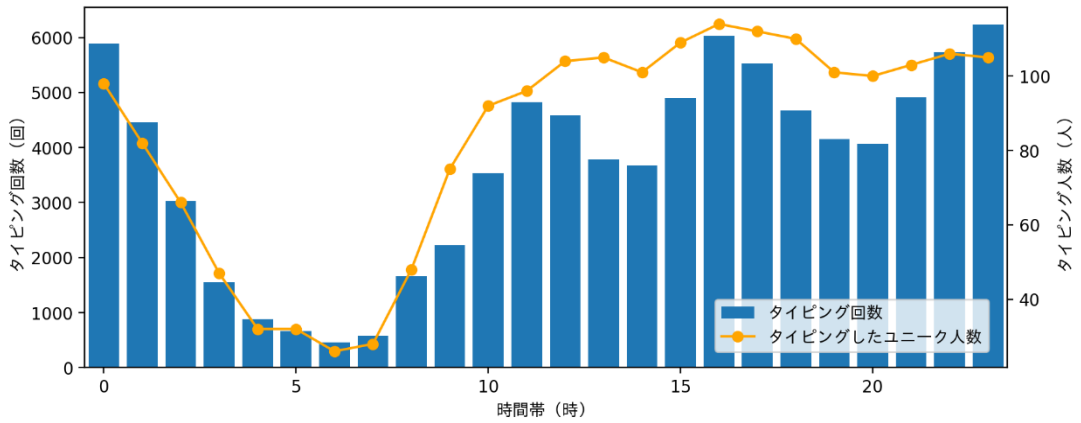

図 10 学生がタイピングを行った時間帯とユニーク人数

また，講義終了後にプログラミング演習 I に関するアンケートを実施したが，93 件の講義に関するフィードバックのうち，ほぼすべてが高評価であった．また，29 件のアンケートに typing.run により助けられたことに関する記述があり，特に，typing.run で楽しんだ，

タイピング速度が速くなった，プログラミングが楽しくなった，他のプログラミングの講義でも是非とも使いたい，他人と競うのが楽しかったなどの言及があった．

以上のことより，本システムは，オンライン講義において自由に好きな時間帯に学生が取り組むことでその効果を発揮することができたと考えられる．

3.5.2. オンライン講義でのシステム運用

本来なら運用前に，学生を対面で集めて使い方などのチュートリアルを行うのが一般的である．しかし，オンライン環境ではできないため，システム上で大事なことは赤色の文字にしたり，フォントを大きくしたり，広く使われている共通のマークを選んだりと，わかりやすいような UI の設計にこだわった．また，問い合わせが気軽に出来るチャットも設けたうえで，TA によるログイン方法のサポートや，バグの報告やその修正，学生からの要望を取り入れるなどの双方向なやり取りも多く行った．著者が想定していない問題が発生した時には，学生に動画や写真，文章などで細かく情報を伝えてもらった．

本システムは，システムの公開を講義開始 1 か月半以上前に行い，またシステムの動作テストを繰り返しつつ改良を行い，学生自体にも使用できるか確かめてもらっていたこともあり，大きな障害や問題が発生することなくスムーズに運用ができたと言える．

また，運用中は教師の監督下にないため，ツールを使った不正や JavaScript のプログラムを悪用した不正な行為が起こることを運用前から予想していた．実際に，1 名から 1000CPM 以上の人間には不可能なスコアが 5 回検出された．この学生のアカウントを一時凍結し，不正行為を行わないように指導したうえで，凍結を解除した．一方，3 時間で 398 回，つまり，1 回あたり約 30 秒かかるタイピングに 3 時間連続で取り組んだ学生がいた．不自然なため不正ツールの利用を疑いアカウントを一時凍結したが，実際には学生本人が入力していた（この学生は 1 つの問題に対して 1283 回取り組んでいた）．このように想像以上に真剣に取り組んでくれることもあるため，タイピングのスコアデータだけから不正行為かどうかを見極めるのは困難であった．

オンライン環境では，学生本人が実際に入力している様子が見えないうえ，直接の顔を合わせた対応ができないため，状況を把握することが容易ではない．そのため，導入に十分な時間をかけることやサポートできる環境を構築するとともに，文字入力の得手不得手などを考慮した不正判定手法が重要であると考えられる．

第4章　　質問したい学生と TA のハードルを下げる手法

4.1. 提案手法

オンラインでは，学生の質問ハードルはより上がると考えられる．また，TA も質問に対応する精神的負荷が上がると考えられる．さらに，こうした質問を適切にさばくための順番待ちなどの仕組みが重要になってくる．そこで，システムの実現にあたり，質問するハードルを下げる仕組みと，質問対応のハードルを下げる仕組み，質問の順番待ちを円滑に行うための仕組みについて述べる．

4.1.1. 質問するハードルを下げる仕組み

オンライン上で実施されている演習講義では，TA などによる質問対応のためにコミュニケーションスペース（質問部屋）を用意することがよく行われている．学生は，質問するためにはそうしたスペースに入室する必要がある．しかし，対面環境とは異なり，オンライン環境では部屋の中に誰が居るのか，どんな状況にあるのか，何を喋っているかなどを覗き見ることができないサービスが多い．そのため，入室のハードルが上がり，結果的に質問するハードルもより高くなってしまう．また，質問のハードルを下げる方法として，学生からの質問を Web フォームやチャットなどで受け付け，順次対応する方法も考えられるが，質問が多数ある場合に，誰がどの質問に対応するかの割り当てが難しく，また質問対応で別システムが必要となるなどの問題がある．

以上の質問するハードルが高い問題を解決するために，まず TA に直接質問するのではなく，一旦システムに対して質問を投稿し，その質問を見た TA に呼んでもらうことで，「TA に呼ばれたから質問できる」と立場を変え，気軽に質問できるようにする．

4.1.2. 質問対応のハードルを下げる仕組み

TA の業務は，どんな学生の質問に対しても回答することであるが，TA 個人にもレベル差や分野の得手不得手があり，質問の内容を聞くまで TA 自身が答えられる質問かどうかがわからない．もし不得意な分野の質問を学生から受けた場合，回答に時間がかかることによるプレッシャーが焦りにつながり，精神的負荷が高まるという問題が発生する．

そこで，TA が質問対応の前に，質問の内容をある程度把握出来るようにし，TA 自身が回答できない内容であれば，その質問を別の TA に頼むことが出来るようにする．こうすることで，質問が来たら必ず対応しないといけないという TA の義務感や，質問に回答できないかもしれないという不安を軽減することができる．

4.1.3. 順番待ちを円滑に行うための仕組み

学生からの質問が増加すると，質問の順番待ち行列が発生し，その行列を適切に管理する必要が生じる．また，学生によっては，TA が対応するのを待っている間に自己解決できる可能性もある．ここで，自己解決したら順番を抜けられるようにすることは二度手間をなくすためにも重要である．

そこで，システムで質問された順番を管理し，順番が回ってきた学生に対して，通知を行うことで質問の場への誘導を促す．また学生の意思で順番から抜けられるようにする．これにより，学生が自分の質問順番に気付かない問題や TA が学生に連絡をする手間や通知が飛ばない問題，自己解決しているのに順番を抜けられない問題を解決することができる．

4.2. **askTA**

以上の手法を取り入れた，オンライン上の質問応答システム askTA を提案する．提案手法を講義において実運用することを目指し，Web システムとして実装を行った．なお，学生や TA を識別するため，大学発行のメールアドレスによる認証システムを使い，ログインを行ってもらうようにした．

4.2.1. 実装

学生の質問ハードルを下げつつ，TA が事前にどのような質問なのかを把握可能とするため，質問投稿フォーム（図 11）では，質問する課題の番号，質問の種類，質問の概要，できたところまでのプログラムという入力項目を設け，システムへの質問の投稿を可能にした．なお，質問の種類は，「ヒント」「ヘルプ」「問題文に関する質問」「採点に関する質問」「その他」の 5 つを用意し，選択式にした．質問内容によっては，出来たところまでのプログラムの内容とは関係がない質問がされる場合も想定されるため，ソースコードの欄は省略可能とした．なお，運用の都合上，学生が所属する組で指導する TA を分ける必要があったため，図 11 では組も選択するようになっている．

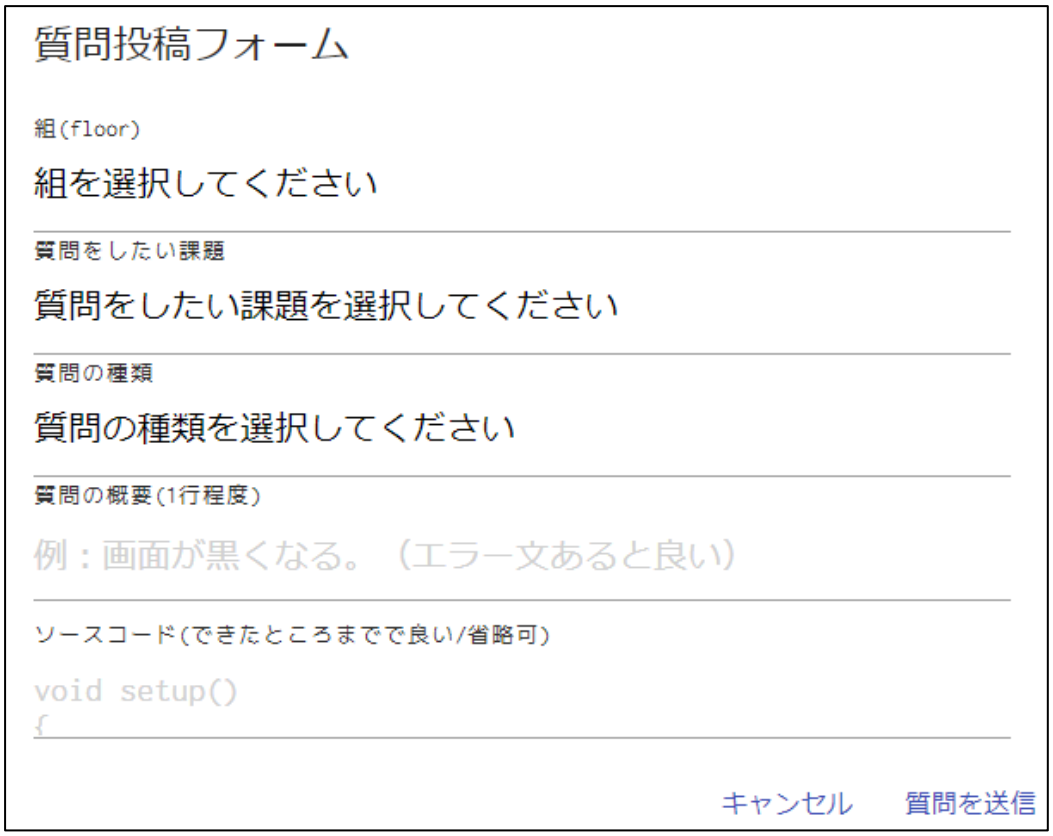

図 11 学生が課題番号や質問の種類，概要，
プログラムを入力して質問を投稿するフォーム

ダッシュボード（図 12）では，学生から投稿された質問の状況を一覧で見ることができようにした．また，TA のみならず学生も他の学生の質問の概要が見られるようにした．1 つの質問には，質問の種類，質問している学生の名前，対応した TA の名前，質問の概要という情報をもたせ，質問のステータスは，「TA 待ち」「TA 対応中」「TA 終了」「TA 保留」「学生キャンセル」の 5 つを設けることで，質問対応状況を可視化した．質問のひとつひとつには，英数 10 桁の質問 ID が割り当てられており，質問 ID の表示は TA のみに行った．

TA が学生の質問に対応する際には，学生が書きかけているプログラムと口頭での会話による質問の詳細を把握することが重要である．そこで質問ごとにシェアエディタ（図 13）を生成し，TA と学生が同じ URL（質問 ID）にアクセスすることで，プログラムの共有と共同編集，実行，音声通話を可能とした．また，この画面上で質問の対応開始や対応終了などの TA による質問ステータスの更新や，学生による質問の取り下げ（図 14）を設定可能とした．なお，対応中の TA の名前や学生の名前，互いのマウスのカーソル位置も表示することで，質問対応を容易化している．また，学生による不正行為防止のため，学生は他者の質問のシェアエディタにアクセスしてプログラムなどの中身を見ることができないようにした．

図 15 は，学生に質問の順番が回ってきたことを知らせる通知である．しかし，この単純な通知では気づかない可能性があるため，5 秒程度の効果音も同時に鳴らすことで順番が回ってきたことの気づきを促した．

本システムでは，著者の学科のプログラミング教育に Processing が採用されていることから，Web 上で実行を行える Processing.js を Processing と同じような挙動を行うように改

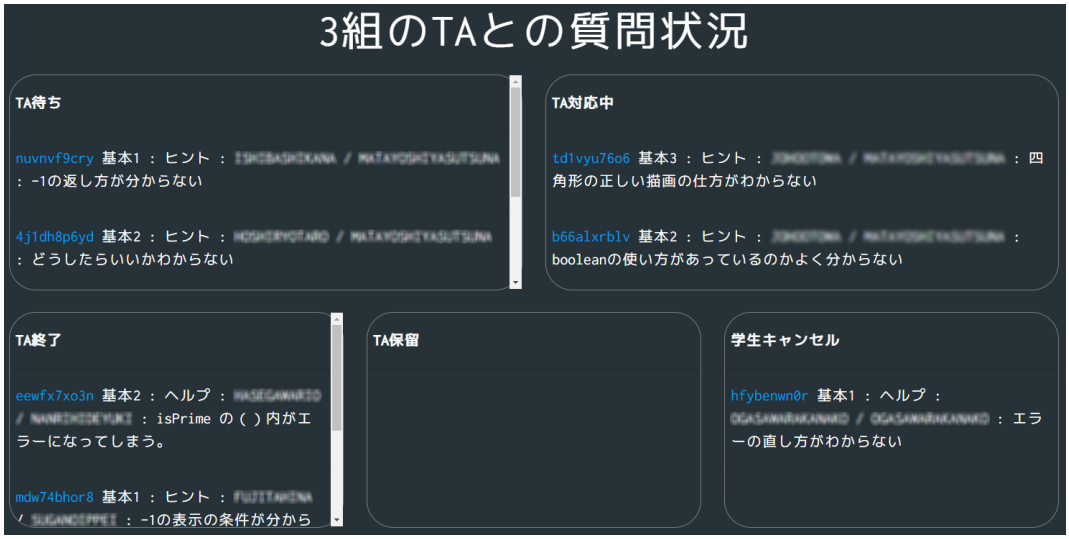

図 12 投稿された質問の状況や内容が一目で分かる
ダッシュボード（個人氏名はモザイク加工）

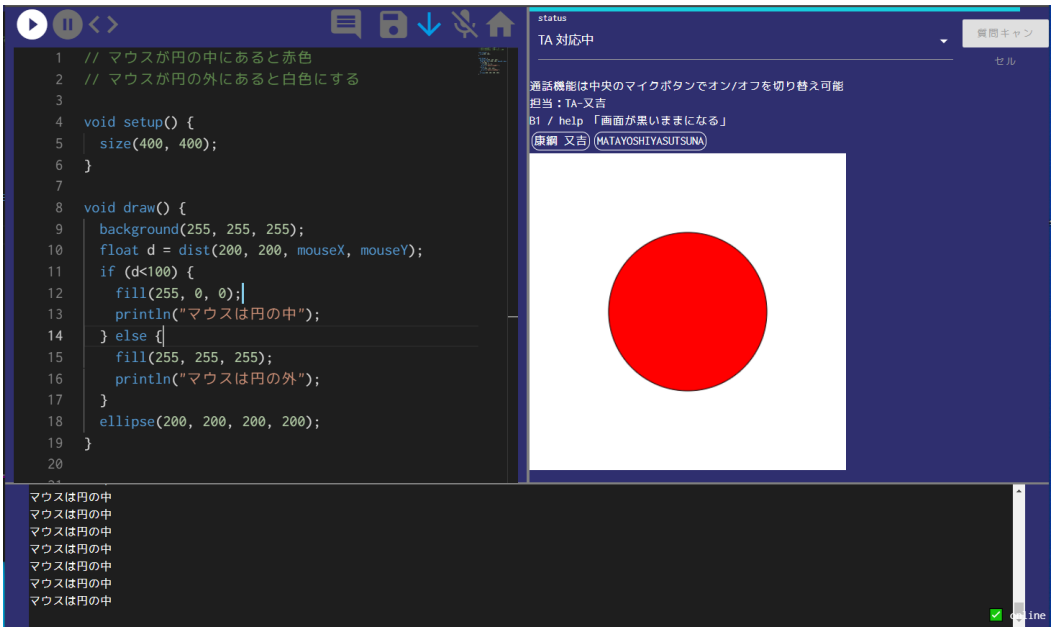

図 13 学生と TA がプログラムの共有や実行を行い
音声通話しながら質問を解決するシェアエディタ

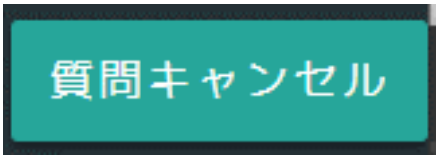

図 14 学生が質問を取り下げることが出来る
質問キャンセルボタン

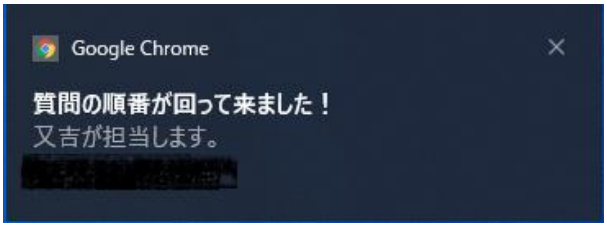

図 15 学生に順番が来たことを伝えるトースト通知

良して実現した．なお，セキュリティの観点から Web 上で実行する前にサーバ側の Processing でコンパイルし，実行可能かどうかについて確認を行った．

4.2.2. 利用方法

学生は，システムにログインしたうえで，質問投稿フォームより解決したい質問を投稿する．質問を投稿した後は，自動的にシェアエディタに遷移し，TA が対応可能となるまで，ブラウザのタブとしてシステムを開いておいて順番待ちをすることになる．

TA はシステムにログインすると，ダッシュボードを確認でき，質問対応待ちや対応済みなどといった質問の状況を把握することができる．また，リンクになっている質問 ID をクリックすることでシェアエディタ画面に移り，質問の概要とプログラムを確認できる．ここで，質問やプログラムの内容から対応可能であると判断した場合は，ステータスを「TA 対応中」に変更することで，質問対応を始めることできる．対応不可だと判断した場合は，他の質問をチェックしていくことになる．

TA が「TA 対応中」にステータスを変更すると，学生には通知が飛び，呼び出しがされる．その後，シェアエディタ上でやり取りを行うことにより質問の解決を行う．なお，学生が質問の順番までに自己解決できた場合は，シェアエディタ右上の「質問キャンセル」ボタンをクリックすることで質問を取り下げることができる．

4.3. 運用と分析

4.3.1. 運用

プログラミング演習 I の講義において，2020 年 7 月 1 日から 27 日までの毎週月曜日と水曜日に全 8 回の運用を行った．プログラミング演習 I の履修者は学部 1 年生と再履修の学生による 111 名で，TA は著者を含む，大学院生 9 名だった．オンライン講義のためのツールとしては，学生が複数あるテーブルを自由に移動ができる Remo Conference[54]を使用し講義を実施していた（図 16）．Remo Conference 上では，テーブル内での音声通話やビデオ通話，画面共有，チャットを使うことができる．また，テーブル内での通話を止め，講師が全員に Web カメラ映像や画面共有などをブロードキャスト配信することができる．

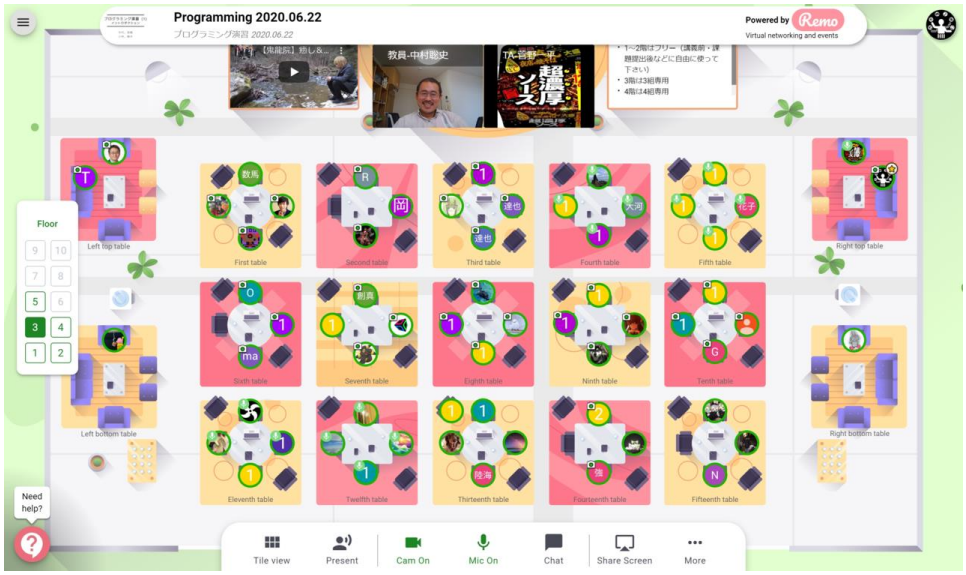

図 16 中央のテーブルに学生が座り，周りに質問待機中の TA が着席している講義中の Remo Conference の様子

講義（100 分×2 コマ）では，前章の typing.run や予習資料などをもとに予習してきたことを前提に，スライドを用いて座学形式で説明を行い，講義時間終了までに提出する必要がある課題を複数課した．学生は，課題解答中は Remo Conference 上で指定されたテーブルで作業を行い，簡易的な質問は同じテーブルの学生同士で助け合い，どうしても解決できない質問については askTA で質問してもらった．

運用の 4 回目までを終えた時点で，TA が学生の質問に対応している時間が長くなりすぎる傾向が見られ，質問対応が滞ってしまう問題が生じた．そのため，5 回目以降の運用では，1 回の質問当たり 10 分を目処とするように TA に依頼するとともに，シェアエディタの右上部に 10 分間を測る水色のバーを設けた．なお，この措置に強制力はなく，対応を行う時間は TA に任せた．

4.3.2. 利用状況の結果と分析

運用期間中にされた質問の回数は 367 回だった．そのうち 15 回は，TA が対応する前に学生によってキャンセルされていた．質問した人と質問回数の関係のグラフを図 17 に示す．1 回以上システムを使って質問した人は 61 名おり，そのうち 5 回以上質問した人は，39.3% にあたる 26 名だった．一番質問回数が多い人は，28 回質問していた．

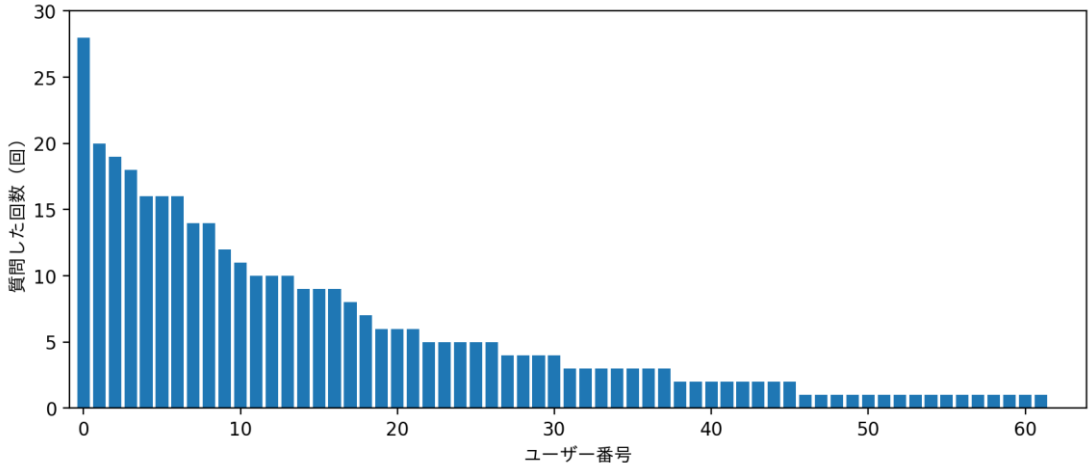

図 17 質問した人とその回数の関係

図 18 は，学生が質問を投稿した時間と TA が「対応中」にステータスを変更した時間の差を取ることによって，学生の質問に TA が反応するまでの待ち時間（分）の分布を求めたものである．この結果より，51.6%の質問は，1 分以内に TA が対応していたことが分かる．

図 19 は，TA が「対応中」と「対応終了」にステータスを変更した時間の差を取り，TA が学生の質問に対応した時間（分）の分布を求めたものである．この結果より，対応時間が 4 分台の質問がもっとも多かったことが分かる．また，ほとんどの質問は 20 分以内には対応が終わっていることも分かる．

なお，講義中の休憩時間（10 分程度）や講師による補足解説で TA 業務が一時中断中もシステムを運用しており，図 18 では待ち時間，図 19 では対応時間にこれらの TA 業務が一時中断していた時間を含む質問があった．

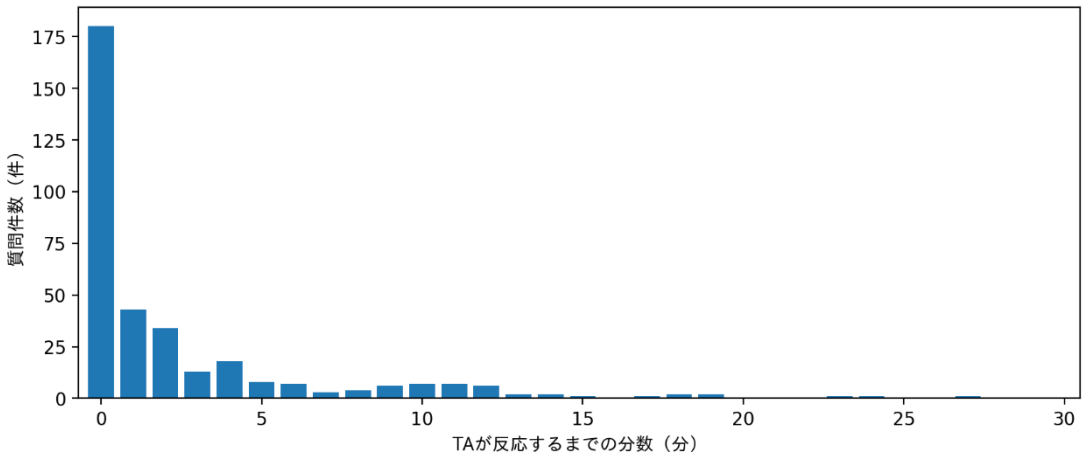

図 18 学生の質問に TA が反応するまでの
待ち時間（分）の分布

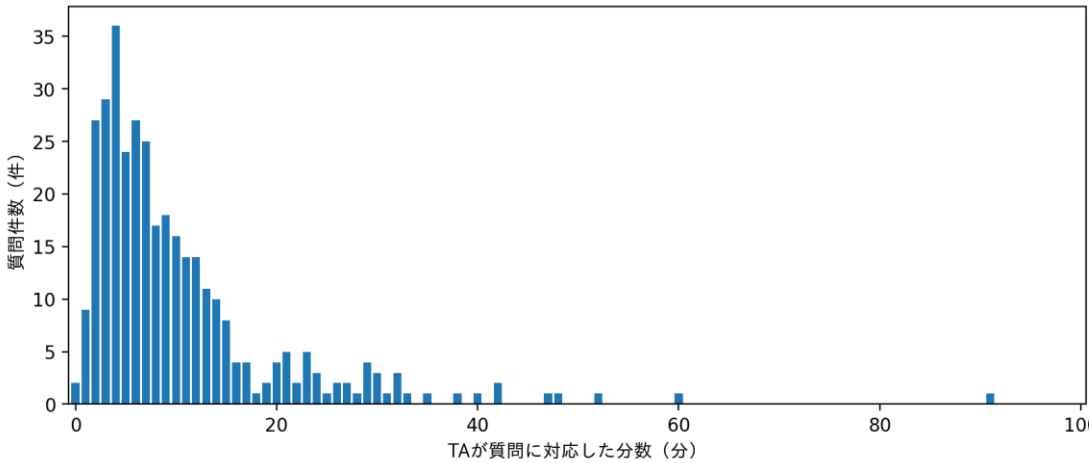

図 19 TA が学生の質問に対応した時間（分）の分布

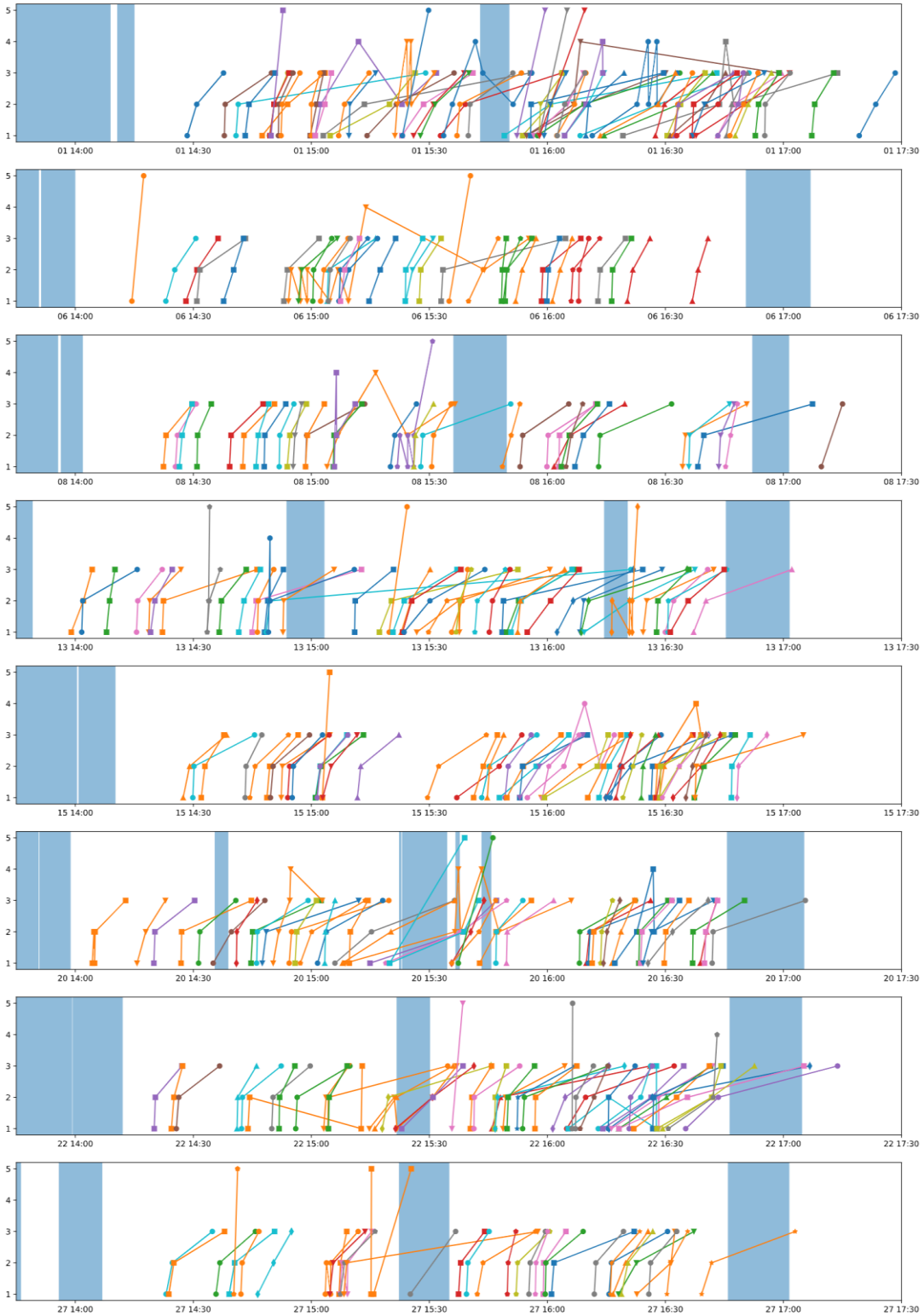

図 20 学生ごとに一意な色とマークで示された質問に対するステータス変化を縦軸に時間を横軸にとった可視化(青塗りの領域は TA 業務中断時間)

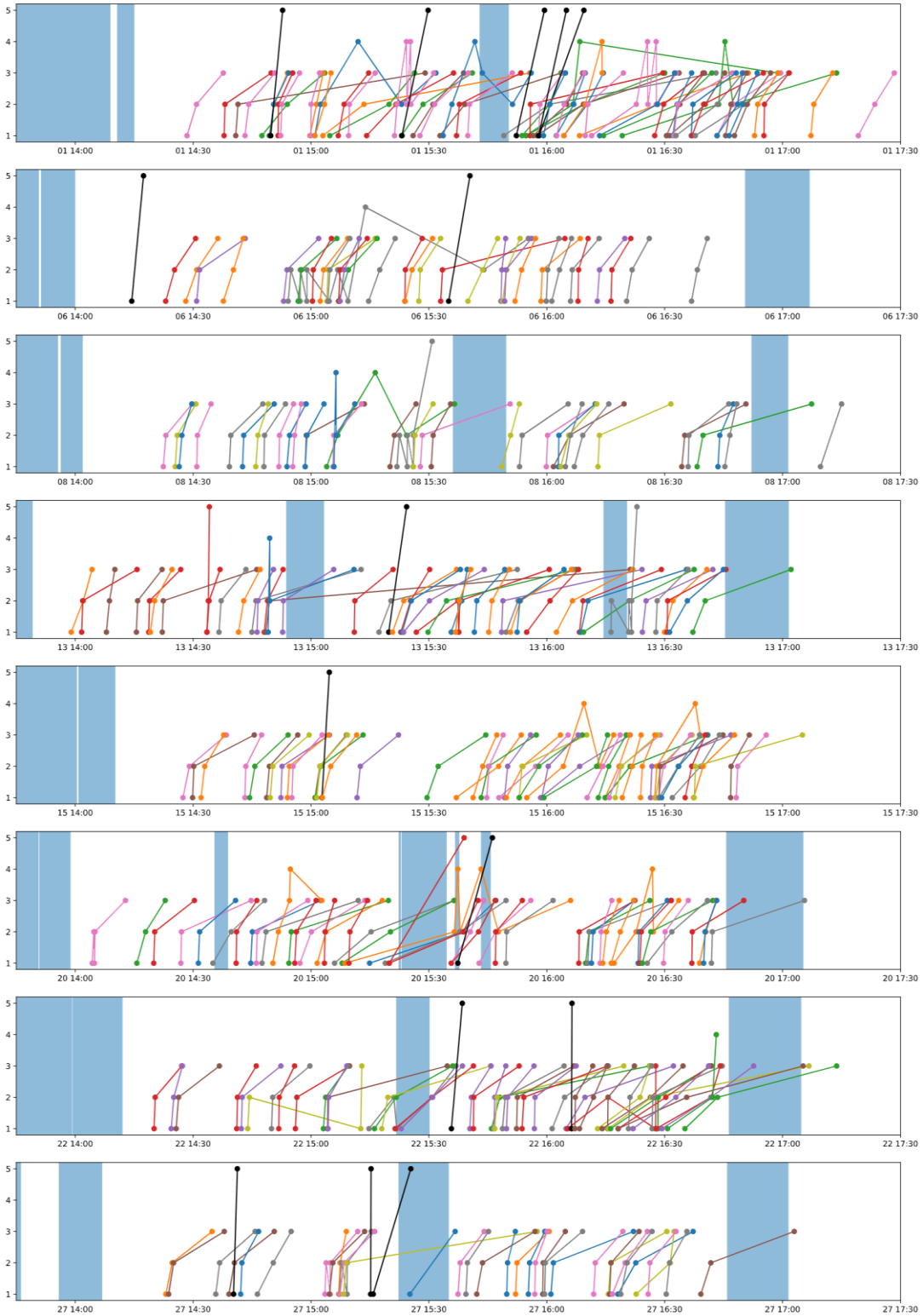

図 21 TA ごとに色分けされた質問に対するステータス変化を縦軸に
時間を横軸にとった可視化(青塗りの領域は TA 業務中断時間)

図 20 と図 21 は，横軸に講義時間，縦軸に質問のステータス（1:TA 待ち，2:TA 対応，3:TA 終了，4:TA 保留，5:学生キャンセル）を取ることで，学生の質問ごとにどのようなステータスの遷移を行ったかが時系列でわかるように全 8 講義分を可視化した結果である．横軸の時間は 7 月の日付と時間を表し，1 つの折れ線は学生 1 名がした質問を表している．また，青色の時間帯は TA 業務が一時中断した時間帯である．なお，TA によって休憩をとる時間が異なっているため，TA の休憩時間は TA 業務の一時中断した時間には含まれておらず，講師の補足解説による中断した時間だけを示している．

図 20 は質問を 1 回以上した学生 61 名ごとに折れ線を色付けとマークのパターンで示したものである．色とマークの組み合わせが同じ折れ線は同一学生である．これより，時間によっては多くの質問が異なる学生から同時に送信されたり，比較的長時間 TA の指導を受けていたりしている様子が観察できる．

図 21 は TA9 名ごとに色を決めて TA ごとに色付けしたものである．学生自身がキャンセルを行った場合（縦軸が 5）は黒色で表している．これより，TA が順番よく対応していることがわかる．なお，TA は講義ごとに学生からの質問に対応する班（6 名）と提出されたプログラム課題の採点を行う班（3 名）に分かれていたため，講義によって色合いが異なる．しかし，第 5 回目の運用に当たる 15 日の 16 時 40 分付近に 1 本だけ赤色の TA が混ざっていることがわかる．これは，質問を待っている学生が多くなったため，採点班だった TA が応援として質問対応をしたことが読み取れる．

4.3.3. 学生アンケートの結果と分析

運用終了後にシステムを使った学生にアンケートを行った．アンケートでは，askTA の利用者 61 名のうち 54 名から回答を得ることができた．表 5 はアンケートの結果である．なお，アンケートでは質問に対する 5 段階評価（-2:全く当てはまらない〜2:とても当てはまる）での回答と，自由記述で感想を求めた．

表 5 より，システムを使って満足度が下がった人や今後の利用はしたくない人はいないことが分かるうえ，回答者の半数以上について，質問への抵抗感を下げることができていることが分かる．

表 5 学生のアンケート項目とその分布と平均

質問項目	評価値の分布					平均
	-2	-1	0	1	2	
使った満足度は高いですか？	0	0	3	14	32	1.58
TA に質問する抵抗感が低くなりましたか？	0	3	17	14	16	0.91
プログラミング講義で今後も使いたいですか？	0	0	1	13	35	1.66

自由記述のポジティブな意見として，「質問を端的に伝えられた」「オンライン講義での質問のしにくさの不安が解消された」「質問することのハードルがかなり下がった」などのように，我々が意図した意見が得られた．また，「プログラムの共有が役に立った」「プログラムを実行できるのが助かった」などの機能に関することも多かった．一方，ネガティブな意見では，「質問を投稿する時に書く概要がうまくまとめられなかった」「プログラムの共有だけでいいと思った」と言った意見も得られた．また，「TA のカーソルが目立つようにしてほしかった」「ボタンを押し間違えた」「実行結果がシステム上では違った」など，実装に関することも多かった．

4.3.4. TA アンケートの結果と分析

運用終了後に，システムを使った TA のうち，著者を除く 8 名にアンケートを行った．アンケートでは 5 段階（-2:全く当てはまらない〜2:とても当てはまる）で評価してもらうとともに自由記述での感想を求めた．

アンケートの結果を表 6 に示す．この結果から，システムを使うことの満足度が高く，今後も使いたいと TA が感じていると言える．

表 6 TA のアンケート項目とその分布と平均

質問項目	評価値の分布					平均
	-2	-1	0	1	2	
使った満足度は高いですか？	0	0	1	4	3	1.25
プログラミング講義で今後も使いたいですか？	0	0	1	1	6	1.63

自由記述のポジティブな意見としては「どれぐらいの人がどこで困っているのかが一覧でわかった」「質問状況から溜まっている質問の量とその内容を確認できた」「挙手よりも質問に気づきやすい」という提案手法の質問をシステムに投稿することや共有することによるものと思われる意見や，「通知機能ですぐに学生が来た」という通知機能によるもの，「学生が修正する様子がカーソルで見える」「プログラムがリアルタイムに共有されるのでカーソルで指をさせる」「行番号が助かった」「通知機能でも来ない人に対して通話機能で話しかけれた」というシェアエディタの実装に関することも多いことが分かった．また，「askTA 単体で TA 業務が成り立つ」という意見もあった．一方，ネガティブな意見として，「TA の引き継ぎができないため，他の TA が指導した続きを教えるのがキツかった」「TA も自由に楽に編集できるので教えすぎてしまうことがあった」というオンライン特有の意見や，「TA 側にも通知機能が欲しかった」「図を使って説明したかった」「実行環境にバグがあった」という実装に関する意見もあった．また，「askTA に頼り切りな人がいた」という対面講義時と同様に TA に頼り切る学生がいたこともわかった．

ダッシュボードに掲載される質問の回答方法について詳しく記述を求めたところ，「基本的には上から順に対応した」「1 度対応し，再び質問する学生には積極的に対応していた」といった一方で，「自分が理解していない課題の質問は他の TA が対応したほうが良いと思ったため対応しなかった」「概要の文章が乱雑な質問には対応したくなかった」「対応に躊躇してただ眺めてるときもあった」「学生の質問対応の担当ではないときにサポートで入ったが，難しそうな質問が来たら対応せずにサポートをやめた」といった感想があった．

4.4. 考察

4.4.1. 利用状況の考察

図 17 より，システムを利用した人は学生の約半分であり，広く利用されていたことが分かる．同じ人が何回もシステムを使って質問していることに関しては，対面講義でもよく見られる現象であるが，TA がヒントを一度与えて，もう一度来るように指導したことも原因として考えられる．なお，質問回数が最も多かった学生は再履修生であり，単位を取るために質問を繰り返していた可能性がある．

図 18 より，ほとんどの質問は 1 分以内に TA が対応するが，数分や数十分待つことになった質問もあったため，学生にとって本手法の通知機能が活躍したと考えられる．なお，TA が質問対応を行うかの選択が可能であるが TA 全員が対応しなかった質問はなかった．

図 19 より，質問の対応に長く時間がかかることもあり，TA が指導の順番を把握するのは困難であったと言え，システムで順番を管理することで TA の負担を下げることができた可能性がある．

図 20，21 より，多くの折れ線が 3 点（1:TA 待ち，2:TA 対応，3:TA 終了）で表され，1 から 2 の長さは短く，傾きが急であり，2 から 3 の長さは長く，傾きが緩やかであることで，お辞儀しているような形をしていることから，学生は少ない待ち時間で TA に対応してもらえつつ，それ以上の長い時間をかけて質問対応している傾向が見える．また，TA 業務が一時中断したことを示す青色の時間帯を横切るかたちで折れ線が伸びている質問は，青色の右端である TA 業務の再開と同時に対応終了ステータス（縦軸が 3）になっている質問が多くある．これは，TA 業務の中断中に行った講師の補足説明の中で学生が TA にしていた質問が解決されたことによる現れであると考えられる．全体的に，講義によって質問数や密度の違いがあるものの，TA が忙しく対応していたことが読み取れる．これらから，本システムが学生と TA 双方にとって円滑に質疑対応ができ，快適であったことが考えられる．

askTA を運用した 8 回のうち 1 回の講義（7 月 15 日）で，最初の 30 分以降，Remo Conference に接続できなくなり，最後まで復帰しないトラブルが発生した．しかし，askTA の機能である質問の投稿とステータス，及びシェアエディタのプログラム共有，音声通話よ

り，特に大きな支障なく学生は課題に取り組むことができ，TA もその時間に質問された 51 件について十分に質問対応を行うことができていた．

4.4.2. 学生アンケートの考察

表 5 の結果より，満足度と今後も使用したいの質問に対して，負の評価をした学生はおらず，学生からの評価は高いものであった．また TA に質問することの抵抗に関しては 0 を選択した学生が最も多かったが，正の評価をした学生が倍に近い人数であったことから，TA に質問する抵抗感も低くすることが可能であったと言える．

自由記述で得られたポジティブな意見から，提案手法の質問するハードルを下げる効果が出ていると言える．一方，ネガティブな意見から，学生が質問投稿時の概要の記述が難しいと感じていることがわかった．概要は，TA が対応可能であるかを判断するために重要な項目であるため，適切な記述が求められる．しかし，質問の概要には，「よくわからない」「だめなところがわからない」「2 行目がおかしい」「エラーが起こる」などといった，具体的な質問内容を記述しないものが一定数あった．これは，学生に概要の書き方について指導しなかったことで起こったと考えられる．これについては，指導で「エラー文を含んでください」や「悩んでいることの主語（何が動かないのか）を可能な限り記述してください」などと運用前に丁寧に説明することで改善可能である．なお，適切に記述できていた学生の質問の概要は，講義実施者である講師が，課題解決に際し，どういった問題が発生しているかをダッシュボードから容易に把握できるため，適切なタイミングで全体向けのヒントを提示することにつながっていた．

4.4.3. TA アンケートの考察

表 6 の結果より，満足度と今後の使用に関して，負の評価をした TA はおらず，各質問項目の平均も 1 を超える高いものとなっていることから，TA はシステムを好意的に捉えていたことが分かる．

自由記述からは，提案手法の順番待ちをしっかり行えることや質問が事前に分かること，通知機能が評価されていることが分かった．さらに，カーソル位置の共有や通話機能などのシステムの細かい実装や些細な工夫が評価されていた．しかし，同じ学生に対して同じ TA が対応できないという問題が指摘された．対面講義では，TA が継続して同じ学生を見ることで，「ココまで出来たらまた呼んで」など，解法の流れを区切って教えることや質問意図の疎通が楽になるメリットがあるため，手を上げた学生を TA が誰か判断し，継続で見たほうがいいのであれば，TA 間で融通を利かせて指導を行うといった工夫がみられていた．一方，本システムは学生が TA を指定して呼ぶことは出来ない上に，TA 間コミュニケーションを促して，融通を利かせられたりするような設計を行っていなかった．これらから，質問

したい学生と TA をマッチングするだけではなく，マッチングまでの TA 間でのコミュニケーションや，学生と TA の指導中のハードルが下がったことによって生じる新たな指導方法に関する問題が発生していると考えられる．また，実行環境にバグがあったことや，TA も質問が来たことの通知が欲しいなどの機能に関する意見も得られていたが，これについては実装に反映し改善を行った．

ダッシュボードに掲載される質問の回答方法についての記述より，TA は業務のため質問が来たら順序よく対応していた一方で，他の TA に質問を回していたことや，躊躇して対応しなかったことがあることが明らかになった．これらから，本提案手法により TA の質問対応のハードルを下げることができたと言える．

第5章 総合的な考察と今後の展望

5.1. 総合的な考察

2 つの提案手法に基づいたシステムの運用結果より，タイピング及び基本的関数の定着と，質問したい学生と TA のハードルを下げることでオンラインでの演習型プログラミング講義を円滑に行うことができた．ここでは，1.5 節で述べた項目をもとに，本研究全体を通した総合的な考察を行う．

まず，3 章の typing.run に関して，3.3.1 項より，学生ごとにタイピングした回数を 50 分割し，その区間ごとの CPM 平均を求めたところ，右肩上がりに CPM が上昇していることが確認出来たため，学生のタイピング速度が全体平均で見たときに成長しており，多くの学生のタイピングスキルが向上したといえる．さらに，3.3.2 項のアンケート結果において，タイピングの得意度について 71%の学生が上昇していることもこれを裏付けている．また，自由記述より，「予習資料の理解が深くなった」や「自然に手が関数を覚えた」などの意見が得られたことから，システムを繰り返し使用することで基本関数の定着が可能であることが示唆された．

次に，4 章の askTA に関して，4.3.3 項より，学生のアンケートで TA に質問する抵抗感について尋ねた平均値が正の値になったことから，学生が質問しやすい状況であったといえる．また，4.3.4 項の TA のダッシュボードの質問対応方法に関する記述より，躊躇し対応しなかった質問があることで TA の質問対応の精神的負荷を下げることができていたといえる．さらに，4.3.2 項より学生がキャンセルした質問が 15 回あったことで TA を待っている間に自己解決した質問は TA 待機列から抜けており，また，質問のおよそ半分に対して 1 分以内に対応していたことから，システムによって円滑な順番待ちが可能であったといえる．

5.2. **COVID-19** による大学初年次オンライン運用の考察

本研究は，COVID-19 によって生じた演習型プログラミング教育のオンライン化に起因する問題を低減させようと試行錯誤しながら生まれた 2 つの手法について述べたものである．2 つの手法とも，著者が TA を担当していて，かつ指導教員が Q2（1 年を 4 つの期間のクオーターに分けた，その 2 つ目）で開講しているプログラミング演習 I を運用の場として利用した．著者は 2019 年度のプログラミング演習 I も TA を担当しており，運用を行った 2020 年度は 2 回目の担当であった．その経験から 2019 年度の対面講義と本提案手法が運用された 2020 年度のオンライン講義を比較も交えながら考察を行う．

2019 年度のプログラミング演習 I は対面講義で実施され，毎講義開始直後に，前回の講義までの内容を加味した小テストが実施されていた．小テストの問題文は印刷し TA によって学生に配布され，パソコン上でプログラムを制限時間内に完成させて提出をさせていた．小テスト終了後は，単元ごとの座学を行い，演習問題の提示と実施を行っていた．TA 業務は，演習問題の採点を行いながら，手を上げた学生の元に TA が出向き，指導を行っていた．

2020 年度は COVID-19 の影響により学年暦が 1 ヶ月遅れ，5 月講義開始となった．4 月末の時点で講義内容に関しては，内容の復習が主だった小テストは，コピーアンドペースト等による不正行為を防ぐことが出来ないとの理由で廃止が決まり，TA の学生への指導に関しては，TA と学生のビデオ通話を実施すると予定されていた．しかし，結果的に本手法の運用を行ったため，小テストの代わりに typing.run の毎授業前までの実施と askTA を用いた TA 業務となった．

typing.run は，4 月 29 日に開発に取りかかり，5 月 5 日から最小限の基本的な機能だけでの運用を開始し学生に周知を行った．オンライン講義 1 回目となる 6 月 22 日までに typing.run でタイピングされた回数は，44,181 回にも上り，全体の 50%に値する．最初のプログラムが短いため何度も挑戦した可能性もあるが, 学生が講義開始前から typing.run のタイピング練習を繰り返すことで，タイピング力や基本的関数の定着を図っていたと考えられる．なお，例年であれば，プログラミング演習 I の前に，Q1 で開講されるエンタテイメントプログラミング演習といったプログラミングそのものについての理解やプログラミングの楽しさを学ぶ講義が設定されていたが，オンラインで実施するための体制整備により，夏休み中の集中講義での実施に変更された．つまり，専門的なプログラミング講義が入学してから 2 ヶ月半の間，実施されない状況になっていた．その間を埋めるように typing.run の運用が開始されたことは, 学生の大学生活における精神的な不安を埋めることにも繋がっていたと考えられる．

askTA は，著者が以前行っていたプログラミングの抽象的思考の可視化を行う研究[27]のシステムをベースに，tyiping.run の開発の目処が立った 6 月 15 日に開発を始め，講義 7 回目となる 7 月 1 日から運用を開始した．そのため，講義 1 回目から 6 回目は，TA が待機している部屋に他の学生が居なければ学生が自由に質問できるという運用を行っていた．しかし，学生が TA の部屋に居続けたり，特定の TA に質問が集中し TA によって忙しさが異なったり，学生のプログラムを画面共有しながらミスを探すのが TA にとっては辛かったりするといった問題があった．またこの時から，学生は 4 人 1 組のオンライン上の仮想的なテーブルで受講しているため，そのテーブルのメンバーに相談して解決しなかったら TA に質問するような手順にしていた．しかし，askTA 運用中も含め，この手順を踏んだ学生は少なかったように感じた．これは，グループ内のコミュニケーションの活発度合いが影響していると考えられる．出席番号順に区切った 4 人組グループであったため，講義の最初にメンバー間での自己紹介などの時間を設けたが，オンライン上での初対面のコミュニ

ケーションは難しく，コミュニケーションの活発なグループとそうでないグループがあった．活発なグループでは，グループ内で解決を行えるため TA への質問は減り，活発ではないグループでは，グループ内で相談出来ずにそのまま TA に質問していたと考えられる．どのぐらいの数のグループが活発な状態であったかは不明だが，少なかったように感じた．このことから，グループ内で相談したいが出来ない状態の学生も多かったと考えられる．

本研究では，2 つのシステムをオンラインで運用したが，以上のことから，COVID-19 による大学初年次の学生の精神的不安を和らげていたことや，活発な交流が出来ないことによってグループ内で相談したいができなかった質問を TA が対応していたと推測される．本研究は，ただ今まで対面で実施していたことをオンラインでできるように運用したのではなく，COVID-19 やオンライン時代特有の問題に対応した運用を行っていた．

5.3. 応用と今後の展望

3 章の typing.run に関して，実施状況が実際の成績に及ぼす影響が明らかになっていない．今後，学生のタイピング実施状況と成績の蓄積を行い，相関関係を出すことによって学生が取り組むほど成績が向上するのか，一定のタイピングスキルを身につけることができるとそれ以上の効果はないのか, などを明らかにする必要がある. また, 本研究で typing.run は Processing のプログラムをタイピングし逐次実行を行うものであったが，他言語への対応も始めており，既に JavaScript の構文を元にしつつ Processing を記述できる p5.js[55]版と Python 版を作成している．p5.js 版は本研究と同じような問題とプログラムで実施することが可能だが，Python 版は描画システムが標準では存在しない言語仕様のため，コンソールベースで逐次実行を行う．そのため，問題のプログラムも専用に開発した．Python は AI ブームなどにより学ぶ需要が高まっているため，Python 版を多くの初学者に使ってもらうことで支援することが可能だと考えている．また，本研究では，typing.run をオンライン講義で利用していたが, 対面講義での利用も可能であり，学生自身が予習や復習として取り組みながらタイピング力をつけることができると思われる．typing.run は，2021 年度に日本大学へ導入を行うことが確定している．本手法が多くの学生を支援できることと同時に，多様な学生に使ってもらいフィードバッグを得ることでより洗練された手法や側面を明らかにすることができると予見される．

4 章の askTA に関して，プログラミング演習 I での運用しかしておらず，様々な講義形式が実施されているなかで，手法が有用である講義形式の条件や TA と学生の比率などが明らかになっていない. 講義を行う講師によって重視する点が異なることや，大学によって学生からの質問の傾向が異なることも考えられるため，複数の条件において，手法がどのような状況で有効であるかを明らかにする必要がある．さらに 1 人の TA が複数人の学生を一斉に指導する状況や，TA 間での学生の引き継ぎや継続指導が askTA では考慮されていないため，今後改善を行う必要がある．一方，応用的なこととして askTA では，質問対応

がすべてオンラインで行われるため，学生が記述した質問の概要やプログラム，さらには学生と TA の音声通話のやりとりをすべて記録，録音することが可能となる．通常の TA 業務を行いつつデータを収集することが可能であるため，講義中に起こる質問をさまざまな角度から分析ができ，1 講義内で行われた学生と TA の質問のやり取りの内容を俯瞰的に示す機能の実現や支援が必要な学生の早期発見，TA の得意不得意を考慮したマッチングなども可能になる．さらに，蓄積することで TA に学生への指導方法の提案や学生のプログラムから質問内容を予想・回答する人工知能など，より良い TA 業務につなげることができるようになる．また，オンライン講義だけでなく，対面講義でも askTA を導入することで，順番待ちの効率化や共同編集と実行などによって，ハードル低下による学生の質問数増加や TA の負担軽減，学生との意思疎通の手助けが可能になる．さらに，askTA は演習型のプログラミング講義を対象としているが，他のプログラミング以外の講義にも応用することができ，質問投稿の仕組みと TA との音声通話だけでも TA の負担や学生の質問を快適できる可能性がある．

第6章 おわりに

オンライン時代における，大学初年次のオンライン演習型プログラミング講義において，講師の指導が行き届かない問題と質問したい学生と TA のハードルに関する問題の 2 つの問題に着目し，それぞれ提案手法を元にした実装と実際のオンラインプログラミング講義での運用を行った．

講師の指導が行き届かない問題に対して，プログラミング特有のタイピング練習と基本命令や関数に関する理解と定着を行うことでプログラミングを苦手とする学生を少なくすることができると考え，プログラミングの理解を促すための逐次実行やタイピングスキルを学生間で競争することで内発的・外発的動機づけが可能な手法を提案し，実装および運用を行った．その結果，タイピング速度の学生間のばらつきが小さくなっていることや，期間の後半でタイピングのスコアが前半よりも成長していることが確認できたことから，タイピングが苦手なひとの底上げが可能である．また，アンケート結果より関数の理解と定着が可能なことも示唆された．

オンライン講義には，学生が TA へ質問するハードルが上がってしまうことや TA が精神的負荷を覚えること，質問の順番待ちを円滑に行う必要があることといった問題があげられる．これらの問題を解決するため，質問をシステムに投稿し呼んでもらうことで，質問する心理的ハードルを下げるとともに，TA 側も事前に質問を把握することで精神的負荷を下げることができる手法を提案および実装した．その結果，ほとんどの質問は学生が質問を投稿して 5 分以内に TA に対応され，対応時間については 20 分以内のものが多かった．これらから，学生の質問ハードルを下げていたことや TA の精神的負荷を低減できていたこと，質問の順番待ちが円滑に行われていたことを明らかにした．

本研究によってオンライン演習型プログラミング講義を円滑にすることが可能になるが，対面講義やプログラミング以外の TA 業務でも部分的に運用を行うことで，さらなる効率化や TA の負担軽減につながり，大学での教育全般に貢献できる手法であることが期待できる．

謝辞

本研究で 2 つのシステムを講義前と講義中に活用してくれた明治大学総合数理学部先端メディアサイエンス学科の 8 期生及び, システムの導入や質問対応をしてくださった TA の皆様に感謝します.

参考文献

[1] “新型コロナウイルスを想定した「新しい生活様式」の実践例を公表しました｜厚生労働省”. https://www.mhlw.go.jp/stf/seisakunitsuite/bunya/0000121431_newlifestyle.html, (参照 2020-12-18).

[2] “4 月からの大学等遠隔授業に関する取組状況共有サイバーシンポジウム - イベント - 国立情報学研究所 / National Institute of Informatics”. https://www.nii.ac.jp/event/other/decs/, (参照 2020-12-18).

[3] "Processing.org". https://processing.org/, (参照 2020-12-18).

[4] "腕試しレベルチェック - インターネットでタイピング練習 イータイピング｜e-typing ローマ字タイピング". https://www.e-typing.ne.jp/, (参照 2020-12-18).

[5] "【寿司打】WebGL 版". http://typingx0.net/sushida/, (参照 2020-12-18).

[6] Kollie, E., Loizides, F., Hartley, T. and Worrallo, A.. TTracker: Using Finger Detection to Improve Touch Typing Training. Human-Computer Interaction – INTERACT 2017. 2017, vol. 10516, p. 469-472.

[7] Hasegawa, T. and Hatakenaka, T.. Touch-Typing Skills Estimation Using Eyewear. 2018 IEEE International Conference on Systems, Man, and Cybernetics (SMC). 2018, p. 3833-3837.

[8] " 情報活用能力調査の結果について：文部科学省 ". https://www.mext.go.jp/a_menu/shotou/zyouhou/1356188.htm, (参照 2020-12-18).

[9] 堀田龍也, 高橋純. キーボー島アドベンチャー：検定機能を実装した小学生向け日本語キーボード入力学習システムの開発と評価. 日本教育工学会論文誌. 2006, vol. 29, no. 3, p. 329-338.

[10] 中村学, 田熊知幸, 菅野英弘, 玉井基宏, 岩根典之, 大槻説乎, 松原行宏. 小学校中・高学年を対象としたタイピング技術習得支援システム. 日本教育工学会論文誌. 2006, vol. 29, no. 3, p. 339-347.

[11] Marjolijn, V. W., Mariëtte, T. and Henny, V. D. M.. Touch-typing for better spelling and narrative-writing skills on the computer. Journal of Computer Assisted Learning. 2018, vol. 35, no. 1, p. 143-152.

[12] 吉長裕司, 金川明弘, 川畑洋昭. 打鍵技術の習熟過程における学習者の初期熟達感と打鍵能力の関係. 情報処理学会論文誌. 2003, vol. 44, no. 12, p. 3252-3255.

[13] 松山智恵子, 中島豊四郎, 石井直宏. 演習でのタッチタイピング練習の効果. 電気学会論文誌. C, 電子・情報・システム部門誌. 2002, vol. 122, no. 12, p. 2189-2190.

[14] Miura, M.. Effect of Auto-complete Function on Processing Web IDE for Novice Programmers. 13th International Conference on Knowledge, Information and Creativity

Support System. 2018, p. 166-171.
[15] Thomas, R. C., Karahasanovic, A. and Kennedy, G. E.. An Investigation into Keystroke Latency Metrics as an Indicator of Programming Performance. Proceedings of the 7th Australasian conference on Computing education (ACE '05). 2005, vol. 42, p. 127-134.
[16] Leinonen, J., Longi, K., Klami, A. and Vihavainen, A.. Automatic Inference of Programming Performance and Experience from Typing Patterns. Proceedings of the 47th ACM Technical Symposium on Computing Science Education (SIGCSE '16). 2016, p. 132-137.
[17] 中田豊久. プログラミング学習の理解度とソースコードタイピングに関する考察. 研究報告コンピュータと教育（CE）, 2013, vol. 2013-CE-121, no. 7, p.1-8.
[18] 喜多一, 岡本雅子. 写経型プログラミング学習と反転授業. 第 60 回システム制御情報学会研究発表講演論文集, 2016.
[19] 喜多一, 岡本雅子, 藤岡健史, 吉川直人. 写経型学習によるC 言語プログラミングワークブック. 共立出版, 2012.
[20] Gaweda, A. M., Lynch, C. F., Seamon, N., Silva, D. O. G. and Deliwa, A.. Typing Exercises as Interactive Worked Examples for Deliberate Practice in CS Courses. Proceedings of the Twenty-Second Australasian Computing Education Conference (ACE'20). 2020, p. 105-113.
[21] Leinonen, A., Nygren, H., Pirttinen, N., Hellas, A. and Leinonen, J.. Exploring the Applicability of Simple Syntax Writing Practice for Learning Programming. Proceedings of the 50th ACM Technical Symposium on Computer Science Education (SIGCSE '19). 2019, p. 84-90.
[22] 岡崎善弘, 大角茂之, 倉住友恵, 三島知剛, 阿部和広. プログラミングの体験形式がプログラミング学習の動機づけに与える効果. 日本教育工学会論文誌. 2017, vol. 41, no.2, p. 169-175.
[23] 坂本一憲, 本田澄, 音森一輝, 山崎頌平, 服部真智子, 松浦由真, 高野孝一, 鷲崎弘宜, 深澤良彰. まねっこダンス：真似て覚えるプログラミング学習ツール. コンピュータ ソフトウェア. 2015, vol. 32, no. 4, p. 74-92.
[24] 松本慎平, 加島智子, 山岸秀一. プログラミング学習前に行われたプログラミングゲームの理解度とその学習後の到達度との関係分析. 情報教育, 2020, vol. 2, p. 31-40.
[25] 野口孝文, 千田和範, 稲守栄. 初心者から上級者までシームレスにプログラミングを学ぶことができる持続可能な学習環境の構築. 教育システム情報学会誌. 2015, vol. 32, no. 1, p. 59-70.
[26] 佐々木晃, 伊藤克亘, 荒川傑. 動機づけと達成度を保証するためのプログラミング入門科目の設計. 情報処理学会論文誌. 2014, vol. 55, no. 1, p. 16-24.
[27] Matayoshi, Y. and Nakamura, S.. Abstract thinking description system for programming education facilitation. 22th International Conference on Human-Computer Interaction (HCII 2020). 2020, vol. 12206, p. 76-92.

[28] 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. 2013, vol. 54, no. 1, p. 330-339.

[29] 市村哲，梶並知記，平野洋行．プログラミング演習授業における学習状況把握支援の試み．情報処理学会論文誌．2013, vol. 54, no. 12, p. 2518-2527.

[30] 加藤利康，石川孝．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．2014, vol. 55, no. 8, p. 1819-1930.

[31] Yan, L., McKeown, N. and Piech, C.. The PyramidSnapshot Challenge: Understanding Student Process from Visual Output of Programs. Proceedings of the 50th ACM Technical Symposium on Computer Science Education (SIGCSE'19). 2019, p. 119–125.

[32] Hattori, S. and Kameda, H.. Knowledge-based Compiler with e-TA for Software Engineering Education. Proceedings of the 9th Joint Conference on Knowledge-Based Software Engineering (JCKBSE'10). 2010, p. 265-278.

[33] Marceau, G., Fisler, K. and Krishnamurthi, S.. Measuring the Effectiveness of Error Messages Designed for Novice Programmers. Proceedings of the 42nd ACM Technical Symposium on Computer Science Education (SIGCSE'11). 2011, p. 499-504.

[34] Queirós, R. A. P. and Leal, J. P.. PETCHA: A Programming Exercises Teaching Assistant. Proceedings of the 17th ACM Annual Conference on Innovation and Technology in Computer Science Education (ITiCSE '12). 2012, p. 192-197.

[35] Krusche, S. and Seitz, A.. ArTEMiS: An Automatic Assessment Management System for Interactive Learning. Proceedings of the 49th ACM Technical Symposium on Computer Science Education (SIGCSE '18). 2018, p. 284-289.

[36] Munson, J. P. and Zitovsky, J. P.. Models for Early Identification of Struggling Novice Programmers. Proceedings of the 49th ACM Technical Symposium on Computer Science Education (SIGCSE '18). 2018, p. 699-704.

[37] Pereira, F. D., Fonseca, S. C., Oliveira, E. H. T., Oliveira, D. B. F., Cristea, A. I. and Carvalho, L. S. G.. Deep learning for early performance prediction of introductory programming students: a comparative and explanatory study. Brazilian Journal of Computers in Education (Revista Brasileira de Informática na Educação - RBIE). 2020, vol. 28, p. 723-749.

[38] Vihavainen, A.. Predicting Students' Performance in an Introductory Programming Course Using Data from Students' Own Programming Process. 2013 IEEE 13th International Conference on Advanced Learning Technologies. 2013, p. 498-499.

[39] Ahadi, A., Lister, R., Haapala, H. and Vihavainen, A.. Exploring Machine Learning Methods to Automatically Identify Students in Need of Assistance. Proceedings of the Eleventh Annual International Conference on International Computing Education Research (ICER '15). 2015, p. 121-130.

[40] Goldman, M., Little, G. and Miller, R. C.. Collabode: Collaborative Coding in the Browser. Proceedings of the 4th international workshop on Cooperative and human aspects of software engineering (CHASE '11). 2011, p. 65-68.

[41] ”Colaboratory - Google Colab”. https://colab.research.google.com/, (参照 2020-12-18).

[42] “Visual Studio Live Share | Visual Studio - Visual Studio”. https://visualstudio.microsoft.com/ja/services/live-share/, (参照 2020-12-18).

[43] Vandeventer, J. and Barbour, B.. CodeWave: A Real-Time, Collaborative IDE for Enhanced Learning in Computer Science, Proceedings of the 43rd ACM Technical Symposium on Computer Science Education (SIGCSE '12). 2012, p. 75-80.

[44] Miller, A., Reges, S. and Obourn, A.. jGRASP: A Simple, Visual, Intuitive Programming Environment for CS1 and CS2. ACM Inroads. 2017, p. 53-58.

[45] Trong, K. N. and Nguyen, N. D.. Towards a Collaborative Integrated Development Environment for Novice Programmers. International Journal of Innovative Technology and Exploring Engineering (IJITEE). 2016, vol. 6, no. 5, p. 21-26.

[46] Park, J., Park, Y. H., Kim, S. and Oh, A.. Eliph: Effective Visualization of Code History for Peer Assessment in Programming Education. Proceedings of the 2017 ACM Conference on Computer Supported Cooperative Work and Social Computing (CSCW '17). 2017. p. 458-467.

[47] Park, J., Park, Y. H., Kim, J., Cha, J., Kim, S. and Oh, A.. Elicast: Embedding Interactive Exercises in Instructional Programming Screencasts. Proceedings of the Fifth Annual ACM Conference on Learning at Scale (L@S '18). 2018. no. 58, p. 1-10.

[48] Čubranić, D. and Storey, M. A. D.. Collaboration Support for Novice Team Programming. Proceedings of the 2005 International ACM SIGGROUP Conference on Supporting Group Work (GROUP '05). 2005, p. 136-139.

[49] Guo, P. J., White, J. and Zanelatto, R.. Codechella: Multi-User Program Visualizations for Real-Time Tutoring and Collaborative Learning. 2015 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC). 2015, p. 79-87.

[50] 栗原一貴, 西田健志, 濱崎雅弘, 簗瀬洋平, 渡邊恵太. 消極性デザイン宣言 – 消極的な人よ、声を上げよ。……いや、上げなくてよい。. ビー・エヌ・エヌ新社, 2016.

[51] 西田健志, 五十嵐健夫. 傘連判状を採り入れたコミュニケーションプロトコル. 情報処理学会論文誌. 2010, vol. 51, no. 1, p. 45-53.

[52] Nishida, T.. Designing Social Interaction Support System with Shyness in Mind. Proceedings of the 2018 ACM Conference on Supporting Groupwork (GROUP '18). 2018, p. 140-144.

[53] "GitHub - processing-js/processing-js: A port of the Processing visualization language to JavaScript.". https://github.com/processing-js/processing-js, (参照 2020-12-18).

[54] "Remo: Live Video Conversations Now Simplified - Remote collaboration has never been so

simple.". https://remo.co, (参照 2021-01-19).

[55] "GitHub - processing/p5.js: p5.js is a client-side JS platform that empowers artists, designers, students, and anyone to learn to code and express themselves creatively on the web. It is based on the core principles of Processing. http://twitter.com/p5xjs — ". https://github.com/processing/p5.js, (参照 2020-12-18).

研究業績

[1] 福地翼，又吉康綱，松井啓司，中村聡史. IllumiFrame：錯視図形を利用した額縁型視聴体験拡張システム. 第 24 回インタラクティブシステムとソフトウェアに関するワークショップ（WISS2016）. 2016.

[2] 又吉康綱，久保田夏美，斉藤絢基，大島遼，中村聡史，鈴木正明. 自他との平均化により手書きをきれいにするシステムの提案. ARG 第 10 回 Web インテリジェンスとインタラクション研究会. 2017.

[3] 又吉康綱，久保田夏美，斉藤絢基，大島遼，中村聡史，鈴木正明. 平均手書きノート. 第 25 回インタラクティブシステムとソフトウェアに関するワークショップ（WISS2017）. 2017.

[4] 簗瀬洋平，又吉康綱，坂井俊介，稲見昌彦. D-Ball 縮減現実を用いたスポーツ競技のデザイン. 情報処理学会 研究報告エンタテインメントコンピューティング（EC）. 2017, vol. 2017-EC-46, no. 11, p. 1-4.

[5] 佐々木美香子，斉藤絢基，新納真次郎，又吉康綱，中村聡史，鈴木正明. 手書きとフォントの融合による視認性向上と書き手の抵抗軽減に関する調査. 情報処理学会 研究報告ヒューマンコンピュータインタラクション（HCI）. 2018, vol. 2018-HCI-176, no. 19, p. 1-7.

[6] Matayoshi, Y., Nakamura, S. and Oshima R.. Mojirage: average handwritten note. Proceedings of the 2018 International Conference on Advanced Visual Interfaces (AVI). 2018, no. 74, p. 1-3.

[7] Sakai, S., Yanase, Y., Matayoshi, Y. and Inami, M.. D ball: virtualized sports in diminished reality. Proceedings of the First Superhuman Sports Design Challenge: First International Symposium on Amplifying Capabilities and Competing in Mixed Realities (SHS). 2018.

[8] 坂井俊介，又吉康綱，簗瀬洋平，檜山敦，稲見昌彦. D-Ball: リアルタイム色抽出動画処理技術を用いた縮減現実における球技体験設計. エンタテインメントコンピューティングシンポジウム 2018 論文集. 2018, vol. 2018, p. 267-270.

[9] 又吉康綱，中村聡史. プログラミング教育円滑化のための抽象的思考記述システム. 情報処理学会 研究報告ヒューマンコンピュータインタラクション（HCI）. 2019, vol. 2019-HCI-184, no. 14, p. 1-8.

[10] 二宮洸太，又吉康綱，中村聡史. 崩れた手書き文字データセット構築と平均化による可読性向上の検証. 情報処理学会 研究報告ヒューマンコンピュータインタラクション（HCI）. 2019, vol. 2019-HCI-184, no. 15, p. 1-8.

[11] 又吉康綱，小山裕己，深山覚，後藤真孝，中村聡史. 画像の類似度を用いたダンス動画

モーション訂正手法. 第 27 回インタラクティブシステムとソフトウェアに関するワークショップ（WISS 2019）. 2019.

[12] 二宮洸太, 又吉康綱, 中村聡史, 鈴木正明, 掛晃幸, 石丸築. 平均化による崩れ文字可読化のための補正文字選定手法. 第 12 回データ工学と情報マネジメントに関するフォーラム（DEIM2020）. 2020, vol. A2-3, no. 1-8.

[13] 松山直人, 又吉康綱, 中村聡史. メッセージ画面のスクリーンショットを用いた横断型返信忘れ防止手法の検討. 情報処理学会 研究報告グループウェアとネットワークサービス（GN）. 2020, vol. 2020-GN-110, no. 18, p. 1-8.

[14] Matayoshi, Y. and Nakamura, S.. Abstract thinking description system for programming education facilitation. 22th International Conference on Human-Computer Interaction (HCII 2020). 2020, vol. 12206, p. 76-92.

[15] 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 研究報告ヒューマンコンピュータインタラクション（HCI）. 2020, vol. 2020-HCI-189, no. 1, p. 1-8.

[16] 青木由樹乃, 古市冴佳, 又吉康綱, 中村聡史, 掛晃幸, 石丸築. 多人数での手書き環境において文字の綺麗さが与える影響の調査. 情報処理学会 研究報告ヒューマンコンピュータインタラクション（HCI）. 2020, vol. 2020-HCI-190, no. 24, p. 1-8.

[17] 又吉康綱, 中村聡史. askTA： 消極的な受講生でも質問可能なオンライン演習講義支援システム. 第 28 回インタラクティブシステムとソフトウェアに関するワークショップ（WISS 2020）. 2020.