# プログラミング学習支援システムの提案

廣瀬研究室 4 年
C1172081 山口円馨

## 概要

2020 年度から小学校をはじめとしたプログラミング教育が始まる。東北公益文科大学ではプログラミングの講義を必修化しているが、受講する学生がプログラミングに抱いているイメージは難しいや学ぶ意味が分からないなどのものである。また、教員側もプログラミングの個人単位のスキルの評価において、学生の母数が多いほどそれに比例して採点する数が多くなるために時間を要するという点があり、動作確認まで行い正確な評価をするのはさらに困難を極める。本研究は、教員が個人の評価を行いやすく、かつプログラミング初学者が学びやすい環境を作るための、プログラミング学習支援システムの構築を目的とする。

本研究では、プログラミング学習における学生側と教員側のプロセスと問題点を発見し、そこから改善点と、プログラミング学習においてどういう機能があればより初学者がプログラミングを行いやすく、かつそのデータを使用して教員が採点の負担を減らすことができるかの方策を考えた。それをもとにシステムを作り実験と検証を行った。今後の課題については、システムをより広い範囲で運用できるように機能を増やしていくこと、教員側が採点をする際により負担を軽減するための機能を増やすこととした。(517 文字)

# 目 次

**第 1 章　はじめに** **7**
1.1 背景 . . . . . 7
1.2 基礎プログラミング . . . . . 9
1.2.1 画面の移動時間 . . . . . 9
1.2.2 教員の採点手順 . . . . . 10
1.3 目的 . . . . . 10

**第 2 章　既存のシステムや研究** **11**
2.1 既存のシステム . . . . . 11
2.1.1 paiza . . . . . 11
2.1.2 Progate . . . . . 11
2.2 関連研究 . . . . . 12
2.2.1 Wapen の改良 . . . . . 12
2.2.2 試験システム track . . . . . 13

**第 3 章　プログラミング学習の問題点** **15**
3.1 学生へのアンケート結果から見えた問題点 . . . . . 15
3.2 プログラミング学習での学生と教員のプロセスにおける問題点 . . . . . 17
3.2.1 学生側 . . . . . 17
3.2.2 教員側 . . . . . 17

**第 4 章　プログラミング支援システムの提案** **21**
4.1 プログラミング初学者がプログラミングをするにあたっての要件 . . . . . 21
4.2 プロセスの分割と方策の提案 . . . . . 22
4.2.1 学生側のプロセス . . . . . 22
4.2.2 教員側のプロセス . . . . . 23

**第 5 章　システムの設計** **25**
5.1 前提と要件 . . . . . 25
5.2 システムの設計 . . . . . 25
5.2.1 学生側の画面 . . . . . 26
5.2.2 教員側の画面 . . . . . 27
5.2.3 データベース画面 . . . . . 27

**第 6 章　システムの開発** **29**
6.1 開発環境 . . . . . 29
6.2 実装した画面 . . . . . 30
6.2.1 学生側の画面 . . . . . 30

6.2.2 教員側の画面 . . . . . 37
6.2.3 情報閲覧画面 . . . . . 38

**第 7 章 システムの実行速度における実証 41**
7.1 評価観点 . . . . . 41
7.2 事前準備 . . . . . 41
7.3 実験 . . . . . 41
7.4 実験結果 . . . . . 43
7.5 実験における考察 . . . . . 44

**第 8 章 考察 45**

**第 9 章 まとめ 49**
9.1 結論 . . . . . 49
9.2 課題と今後の展望 . . . . . 49
9.2.1 使用できる問題の範囲 . . . . . 49
9.2.2 教員の採点への補助 . . . . . 49

**付 録 A アンケート 55**

# 第1章　はじめに

本章では研究の背景と目的について説明する。

## 1.1　背景

新学習指導要領 [1, 2, 3, 4, 5, 6] が公示され、令和 2 年度から小学生中学生高校生を対象としてプログラミング的思考の育成やプログラミング学習が必修となる。小学生はアルゴリズムの理解が目的であり、実際にプログラミングが行われるのは 2021 年度の中学校からである。これにより、2020 年以降の義務教育及び高等学校ではプログラミングが行われる機会が増えることやプログラミング初学者が多くなることと、指導する教員が増えることが想定される。

プログラミング初学者がプログラミングするにあたって、野口ら [7] は以下のように提言している。

> 多様なレベルの学生たちが容易にプログラミングを行うためには、
>
> 1. 学習者のプログラム作成から実行までの操作（開発環境の各種設定やコンパイルなど）をできるだけ軽減する
> 2. プログラムに記載する（される）各種設定文（インクルードファイルの設定やライブラリのリンクなど）をできるだけ減らす
> 3. 入出力に関する設定を軽減する
> 4. 負担軽減を実現しながらも多彩なプログラムを作成することができるを可能にする必要がある

以上に対して、現在基礎プログラミング講義で初学者を対象にプログラミング教育を行っている東北公益文科大学（以下本学）の現状と照らし合わせる。

- 1 について：計算機内部の開発環境は構築されている。プログラム作成から実行までの学生側の操作は CUI[1]で行われており、コマンドを使用する
- 2 について：学内環境に講義内で使用するライブラリはダウンロードされており、 学生が手を加える必要はほとんどない
- 3 について：1 で述べた通り、画面は CUI である。テキストエディタでプログラムを作成、ターミナルエミュレータでプログラムを実行する
- 4 について：基本的にどんなプログラムでも作成できる自由度がある

また、プログラミング教育における採点手法としては、教員が学生側に問題を提示し、学生が解答となるプログラミングを作成し提出、提出されたプログラムを教員が採点、という形態が多く見られ

[1] Character User Interface。キーボード等からの文字列を入力とし、コマンドで操作するユーザインタフェースの様式。

る (図 1.1)。これは、学生数が多くなることに比例して教員の作業量が増えるため、プログラミング教育における問題の一つである。

そこで本研究では、プログラミング初学者と教員を支援するプログラミング学習の支援システムを提案し、構築する。システムは、プログラミング教育の学習の範囲のプログラムが実行できることと、プログラム作成の中での使いやすさや、教員が採点を行いやすいものを考慮するものとする。

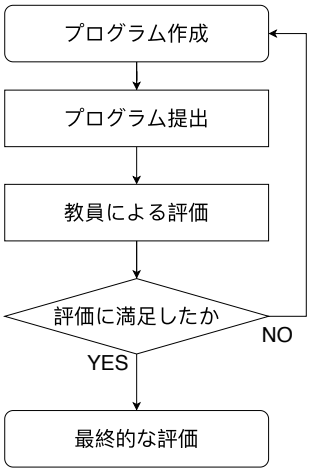



図 1.1: 採点手法のプロセス

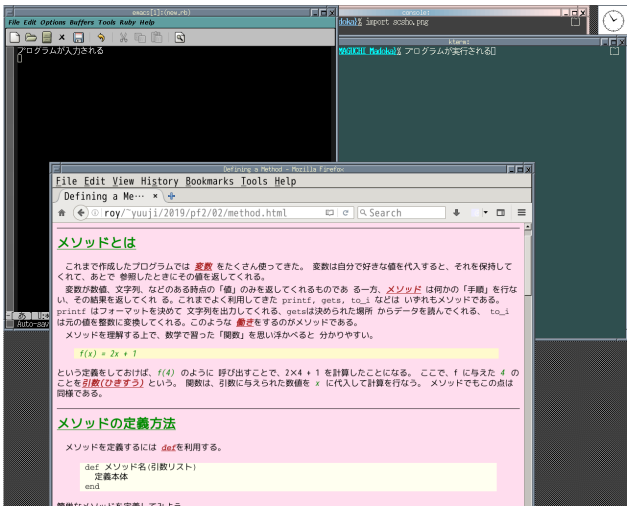



図 1.2: 受講者の標準作業画面

## 1.2 基礎プログラミング

本学ではプログラミングを学ぶ基礎プログラミング I・II が開講されている。本講義では、一つのクラスにつき約 40 人の学生が受講している。学生一人に対して計算機が一台用意される。画面上には、左上にプログラム文を入力する Emacs(テキストエディタ)、右側のプログラムの実行結果を見る Kterm と console(ターミナルエミュレータ)、教員が作成した講義ノートを見るための Firefox(Web ブラウザ) が表示されている (図 1.2)。実際に講義で使うのは 3 つである。これらの画面を標準作業画面と呼ぶ。講義の対象や流れは以下である。

対象：基本的に本学の 2 年生。ほとんどが初学者である

流れ：

1. CUI で操作し、プログラムを作成する
2. 課題は作成した後にメールで教員に送信する
3. 教員はメールを確認して採点。評価の返信を行う

### 1.2.1 画面の移動時間

現状では Emacs、Kterm、Firefox の 3 つのウィンドウを行き来しながら講義を行っている。プログラム作成には Emacs を、実行には Kterm を用いる。また、プログラムを作成する際に問題文や記載方法を確認するために、Firefox を使用する。しかし、これらを逐一ウィンドウ移動によって実行することは時間がかかる。また、Click to Focus と Focus on Mouse というウィンドウシステムの設定により、カーソルがウィンドウに重なっていない場合は入力不可能となっている。これは予期しない場所への入力を防ぐためのものだが、逆に学生を混乱させ、カーソル外で打ち込んでいるために入力されないといった、本来の講義と関係ない部分で問題が起きているという実態がある。

### 1.2.2　教員の採点手順

この講義でのプログラム作成から採点までの手順は以下のとおりである。

学生は、講義中で説明を受けた上でプログラムを作成する。その際、講義専用の Web サイトにはプログラムを構成するために必要な構文と、それを説明するための簡易的なプログラムが記載されている。このプログラムを参考にして、Emacs にてプログラムを作成し、Kterm で実行する。最後にそれらを用いた課題が出され、学生は期限までに課題をメールで送信する。教員側はメールを確認し、記載されたプログラム文を目視した上で実行し、プログラムが成功しているかを判断する。

工数は多くないが、学生全員分に対して行う場合には作業量が人数に比例して増加するため、一つのクラスにつき約 40 名の学生が受講している本学では負担が大きい。

## 1.3　目的

現状の課題を解決するために、教員が個人の評価を行いやすく、かつプログラミング初学者が学びやすい環境を作るための、プログラミング学習支援システムの構築を目的とする。本研究の研究手順は以下のとおりである。

1. 問題点を見つける

   現状の基礎プログラミング講義、既存のシステムや研究を調査、比較することで、具体的な問題点を明らかにし対策を講ずる。

2. 基本の方策の提案

   対策をもとに、システムの提案を行う。

3. システムの設計

   前提と要件を決めた上で基本の方策に則り、システムを設計する。また、それぞれがどういった問題への取り組みになるかを明確にする。

4. システムの開発

   設計をもとにシステムを開発する。また、実際に運用した上でシステム上の問題を明確にする。

5. 全体的な考察

   以上の手順を踏まえた上で、さらに改善すべき点を述べ、具体的な手法がある場合は議論を重ねたのちに実装していく。

# 第2章　既存のシステムや研究

本章では、問題点を見つけるために既存のシステムや関連研究の内容を具体化する。

## 2.1　既存のシステム

本研究にて構築するシステムと類似する機能を搭載している既存のシステムを説明する。類似の機能の条件は以下の基準を満たすものとする。

- システム利用者は専用ウィンドウ内でプログラムを作成することができる
- システム利用者は専用ウィンドウ内で作成したプログラムの実行結果が確認できる

### 2.1.1　paiza

ギノ株式会社が開発した転職支援の Web サービスである [8]。目的別に名称が分かれており、プログラム学習コンテンツは Paiza ラーニングと呼ばれる。図 2.1 のようにウィンドウが各種あり、左画面で動画を見て、右画面で実際にプログラミングを行いながら学習する。各プログラミング言語の最初の動画では、プログラミング言語の説明や学習コンテンツの説明、採用事例等が紹介される。また、paiza.IO というオンライン実行環境によって実行環境が Web 上に構築されている。さらに、paiza の有料会員になることで実際のエンジニアに質問ができるようになるなど、疑問に対する取り組みも存在する。間違えやすいポイントなども Tips として扱われている。一つのチャプターの説明が終わると演習問題に取り組むことができる。演習問題では起こりうるエラーなどに対しての対策を問題として取り上げており、ここでは実際にコードを記入し、判定が行われる。

### 2.1.2　Progate

株式会社 Progate が開発したプログラミング学習サイトである [9]。スマートフォン向けアプリケーションも公開されており、スマートフォンなどの端末上からでもプログラミング学習が可能である。図 2.2 のように左画面で説明が書かれており、右画面で実際にプログラミングを行う。はじめにスライド式でプログラミング言語やプログラミング言語の構文、メソッドの説明を行い、実際にコードを入力する演習に取り組む。問題に正しく答えられた場合は次の演習へと進む仕組みとなっている。基礎的なレッスンは無料で受講できるが、高度な技術を要するレッスンは有料となる。

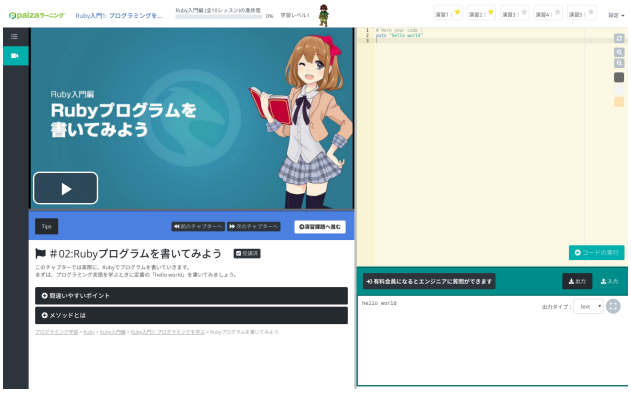



図 2.1: paiza ラーニングの実行画面

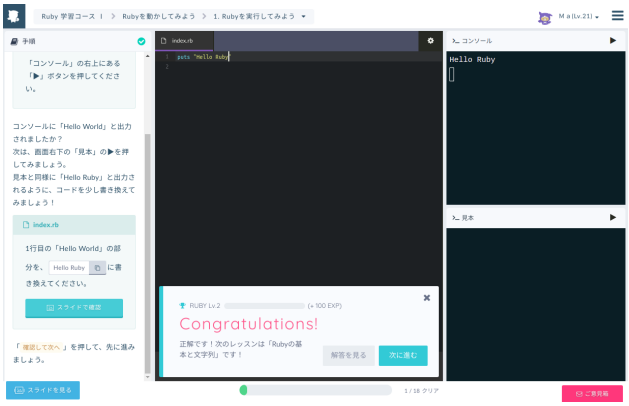



図 2.2: Progate の実行画面

## 2.2　関連研究

関連研究には、実際に大学で運用しているシステムや企業内でのプログラミング教育やテストに運用されているものがある。関連研究を以下に示す。

### 2.2.1　Wapen の改良

中西 [10] は、高等学校学習指導要領において高校生の殆どがプログラミングの学習を行うことを高校現場では大きな変化として捉えており、筆者が開発したプログラミング学習環境を Web ブラウザ上で実行できる WaPEN[11] について改良を行っている (図 3.3)。教員等がサンプルプログラムを簡単に差し替えられるようなプログラムも開発されており、授業を円滑に進めるためのシステムが多数存在する。プログラミング言語「ドリトル」は日本語で記載可能なため「print y」という文を「y を表示する」などに置き換えて分かりやすく作成できるようになっている。中西の研究の考察にあるとおり、これは初学者にとって有効であるがある程度上達した者にとっては足かせとなっている面もある。

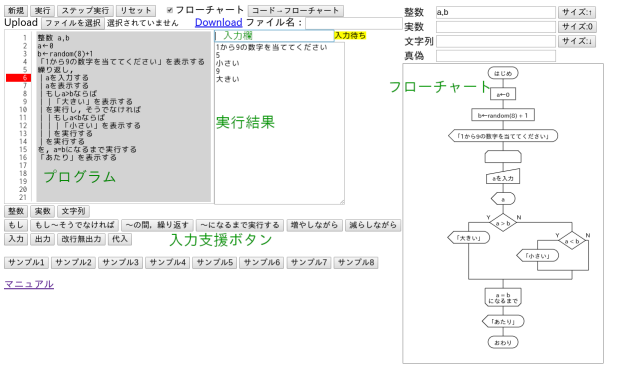



図 2.3: WaPEN の使い方

### 2.2.2 試験システム track

新田、小西、竹内らの研究 [12] では、プログラミング学習環境の構築や改善への期待が高まっていることを鑑みて、学校や企業におけるプログラミング教育に用いることのできるシステムとして track というオンラインプログラミング学習・試験配信プラットフォーム [13] を提案している。track の導入には料金が発生し、一般企業などで「採用選考で活用したい」「社員育成/評価で活用したい」といった用途に合わせて柔軟に対応できるようになっている。サービス使用料金が発生するため、本格的にプログラミングを使用する企業、または人を対象にしているが、初学者が学習の導入として使用することは困難である。導入している会社には株式会社バンダイナムコスタジオ、LINE 株式会社がある。

# 第3章　プログラミング学習の問題点

本章では、アンケートを用いてプログラミング学習の問題点を分析する。また、学生と教員のプログラミング学習におけるプロセスにより問題点を明らかにする。

## 3.1　学生へのアンケート結果から見えた問題点

本節では、本学の基礎プログラミング受講者が授業を受ける際に、感じていることをアンケートを用いて集計した。(2019-11-19 時点)。アンケート用紙は付録 A に示す。アンケートの質問と回答形式を以下に示す。

**質問 1** プログラミングについてどう感じますか。

回答形式:自由記載

**質問 2** 大学入学前にプログラミングを体験した経験がありますか？(Progate など学習サイトを含む)

回答形式:2 択 (ある、なし)

**質問 3** 基礎プログラミングの難易度はどのぐらいですか。

回答形式:5 択 (簡単、少し簡単、普通、難しい、とても難しい)

**質問 4** プログラミングで特に難しいと感じた部分はどこでしたか。

回答形式:4 択 (動かない・エラー文が読めない、メソッドが覚えられない、そもそも組み方がわからない、その他)

**質問 5** 質問 5. プログラミングを楽しいと思う・理解したきっかけは何でしたか。

回答形式:4 択 (動いたとき、難易度が高いプログラムを作れたとき、緻密な作業、その他)

アンケートにより、プログラミングを難しいと考えている学生が 94%を占めている結果が出た (図 3.1)。この結果により、プログラムを学ぶ学生のほとんどがプログラミングに関して苦手意識を持っていることが分かった。また特に難しいと感じた部分については、メソッドが覚えられないが 27%、動かない・エラーが読めないが 34%、組み方がわからないが 36%というように 3 つの選択肢が均等に分かれており、難しいと感じることについて特定の問題があるとは一概に言うことができない結果となった。(図 3.2)。

第 2 章にて説明したプログラミング学習サイトを本学入学前に経験している学生の割合は 4%となった (図 3.3)。この結果により、経験者は少なく初学者が多いことが分かった。

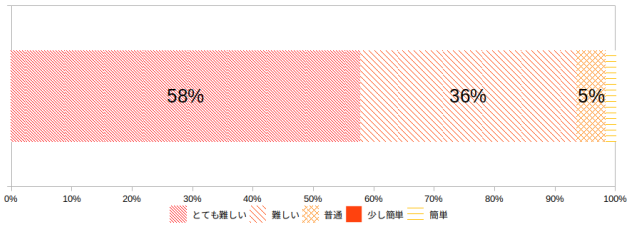



図 3.1: 基礎プログラミングの難易度

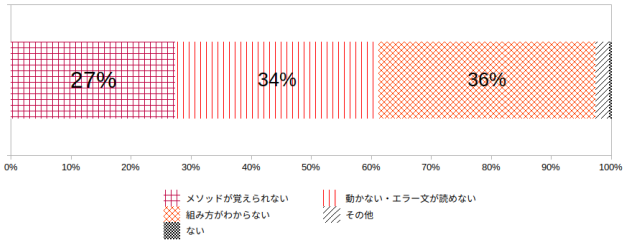



図 3.2: 特に難しいと感じた部分

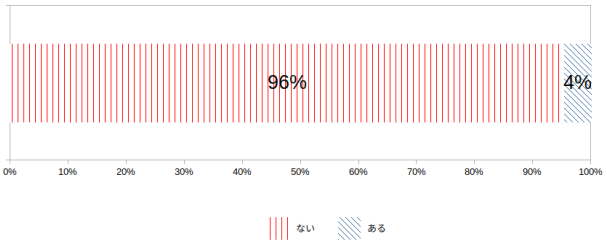



図 3.3: 入学前のプログラミング体験の有無

## 3.2 プログラミング学習での学生と教員のプロセスにおける問題点

プログラミング学習の流れでは一連のプロセスがある。学生側と教員側それぞれのプロセスをもとに説明する。

### 3.2.1 学生側

学生側のプロセスでは「説明を聞く」、「簡易なプログラムを実行する」、「課題に取り組む」、「結果を提出」の 4 つの工程に分けられる (図 3.4)。その際に、何度も繰り返されるのが簡易なプログラムを実行する動作である (図 3.5)。これを実行する際の手順もまた一連のプロセスとなっているが、3 つの画面内を入れ替えながら見る必要があり、効率的ではない。また、基礎プログラミング I・II の TA（ティーチングアシスタント）を務めるうえで感じたことの中に、プロセスの実行中に発生したエラー文に対して、学生が検索する習慣が見られなかったという問題がある。これによりプログラミングが難しいと感じている様子が見られた。

### 3.2.2 教員側

教員側のプロセスでは、「届いたメールを開く」、「プログラムを目視して実行可能かの判断」、「評価する」が大筋となる (図 3.6)。メールには基本的に学籍番号、名前、作成したプログラムの添付ファイル、プログラムについての説明、考察などが記載される。その際「プログラムを目視して実行可能かの判断」において、実行可能か分からないプログラムがあった際には実際にプログラムを実行するといった手順が 1 つ増える。また、メールをそのつど開く、プログラムを目視で確認するといった細かい動作を含めると評価には時間を有することが分かる。これも効率的とは言えない。

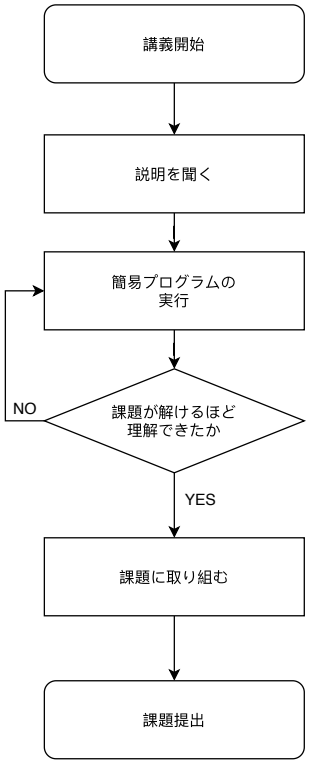



図 3.4: 学生側のプロセス

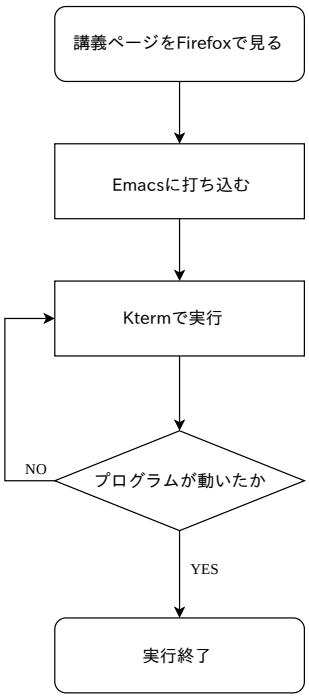



図 3.5: 簡易プログラムを実行するプロセス

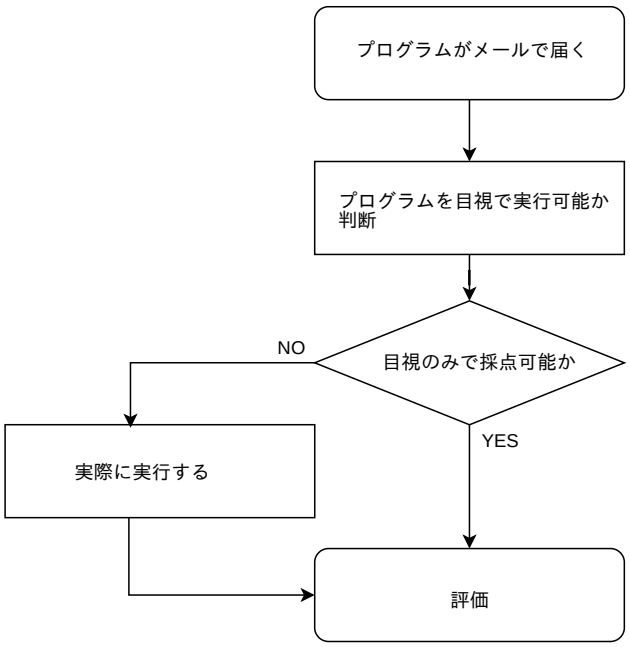



図 3.6: 教員側のプロセス

# 第4章　プログラミング支援システムの提案

第1章から第3章まで述べた現状や問題点を踏まえた上で、本章ではプログラミングに取り組む学生と教員の課題となる部分を解決するシステムを基本方策として提案する。

1.1節で述べた野口らの提言する4つの要件を満たしながらより使いやすさを求めたシステムの提案を行う。はじめに1から4を満たすための条件を挙げる。次に、使いやすさは実際に基礎プログラミングを受講した2年生を対象に行ったアンケートと、プログラミングを行う際の学生の行動プロセスと教員プロセスをもとに考える。

## 4.1　プログラミング初学者がプログラミングをするにあたっての要件

以下に野口らが提唱する要件を満たすための提案を示す。

1). **学習者のプログラム作成から実行までの操作（開発環境の各種設定やコンパイルなど）をできるだけ軽減する**

COVID-19の影響でオンライン授業の形態が多く見られたが、その際に学生自身の計算機の環境により講義が停滞することが多く見られた。課題を家で行う学生もいるが、その際に計算機に入っているライブラリの影響で実行できないことがある。プログラミング教育を行う上では、個々に環境が構築されていることが望ましい。しかし、環境構築には複雑な手順が含まれるため構築する学生は少ない。そのため、あらかじめシステムにはプログラムを実行できる環境が備わっており、学生が手を加える必要性がないものが好ましい。これは自身の計算機に影響されないブラウザ上でのアプリケーションにする。後述する基礎プログラミングに則り、環境はRubyを実行できるものがよい。

2). **プログラムに記載する（される）各種設定文（インクルードファイルの設定やライブラリのリンクなど）をできるだけ減らす**

プログラミングを行うに当たって頻出するライブラリなどはあらかじめシステムに入れておくことが好ましい。例としてJSON[1]やCSV[2]など。

3). **入出力に関する設定を軽減する**

入出力については、テキストエディタとターミナルエミュレータのみを用意し、簡潔に学生が何を行えばよいかを分かりやすい形態にすることが好ましい。ひと目見てここにプログラムを入力、ここで実行、を分かりやすく提示する。そのため、画面に表示する情報は最低限にする必要がある。

---

[1] JavaScript Object Notation。JavaScriptのオブジェクトの書き方をもとにしたデータ定義方法。講義ではデータの整理などに使用される。

[2] comma-separated values。テキストデータをカンマ「, 」で区切ったデータ形式。講義ではこの形式のファイルを利用してデータ処理を行う。

4). **負担軽減を実現しながらも多彩なプログラムを作成することができる**

多彩の範囲をどこまで広げるかは詳しく論じる必要がある。現時点では print メソッドを利用した出力や while などの制御構造を使った繰り返し処理など、外部プログラムを使用しない簡易的なプログラムのみに対応することを考えている。

## 4.2　プロセスの分割と方策の提案

3.2 節の学生と教員のプロセスを分割した上で、改善できる部分を見つけ使いやすさとして考えていく。

### 4.2.1　学生側のプロセス

学生側のプロセスを分割して考える。また、それを改善するための手法を示す。

**方策 1** プログラム実行の際の表示画面を 1 つにまとめる

学生側のプロセスにある簡易なプログラムの実行に関しては、標準作業画面のウィンドウを逐一切り替えるといった動作は、一つの画面内に展開することで解決できる。基礎プログラミングでの標準作業画面のうち、Firefox は問題の情報などを確かめるために閲覧されるものである。これは、あらかじめプログラムの実行画面上に問題文やそれらに関わる情報を表示しておくことで、よりプログラムの作成にのみ集中できる。そのためシステムの画面上にはプログラムを実行する場所以外にも、プログラムに関わる情報をあらかじめ表示するといった場所を作る。

**方策 2** 課題の送信にメールを用いずにシステム内で提出する

課題の送信はメールリーダを起動せず、システム上にプログラムの情報が保存される場所を用意する。システム上にプログラムの実行結果を始めとしたデータを格納するデータベースを作成し、学生が作ったプログラムの情報を別ページに保存する。保存はプログラムの実行結果を確認して、送信の是非を確認したうえで行う。これにより学生側の動作を省き単純化できる。

**方策 3** エラー文を解読するための補助を行う

画面上に表示されたエラー文には URL リンクを貼ることでワンクリックで検索ができるようにする。これによりエラー文が出ることによってプログラミングへの意欲が減少することを減らす。

基本的にプログラムを作成する上で、プログラムは何度でも実行して修正するというプロセスが可能である必要がある。そのため本システムでは、プログラム文は何度も入力、修正可能であることと、最終的に完成したプログラムのみを送信できるようにする。


図 4.1: ワンクリックのイメージ図

### 4.2.2 教員側のプロセス

教員側のプロセスを分割して考える。また、それを改善するための手法を示す。

**方策 1** プログラムが実行可能であることをひと目で判断できるようにする

教員側の評価のプロセスでは、「目視で実行可能か判断する」という工程が作業が増える要因になる。そのため、教員側に情報が届いた時点でそれが成功しているかの判断が可能であれば、採点にかかる時間が大幅に削減できる。また、届いたプログラムがそもそも動かない場合はエラーの原因を突き止めることに時間を割くことができるため、できていないプログラムへのフォローアップが可能になる。

**方策 2** 個人の評価をより簡単にする表示

教員側の方策 1 を踏まえた上で、学生ごとに評価を行う際の表示に工夫を加える必要がある。教員に届くプログラムの情報に個人を判別するものを付与することで、その学生のみを評価の対象として扱うことが容易になる。これによって逐一名簿などと照らし合わせる必要もなくなる。

# 第5章　システムの設計

本章では第4章で論じた方策を実現するためのシステムを設計する。

## 5.1　前提と要件

このシステムの前提と要件を以下に示す。

**前提1** 使用する学生と教員のみがアクセスでき、外部からのアクセスが不可であるローカル環境を想定する。

**前提2** 運用は基礎プログラミングの講義中のみとする。

**前提3** プログラミング初学者を対象とする。

**要件1** 学生がプログラムを入力し、入力の実行結果を学生側に表示する。

**要件2** 実行したプログラムがエラーの場合は、エラーの情報を学生側に表示する。

**要件3** 学生は入力したプログラムを何度でも入力できる。

**要件4** 学生がプログラムの実行に成功した場合は、学生のユーザ名と共にプログラムの情報が保存する。

**要件5** 学生がユーザ名が入力していない場合はプログラムの情報を保存しないようにする。

**要件6** 教員は保存されたプログラムの情報を利用して、採点することががができる。

## 5.2　システムの設計

5.1節をもとにシステムを設計する。本システムは図5.1のフローを行うものである。また、学生や教員ごとに専用の画面を設ける。以下にそれぞれのページの説明を示す。

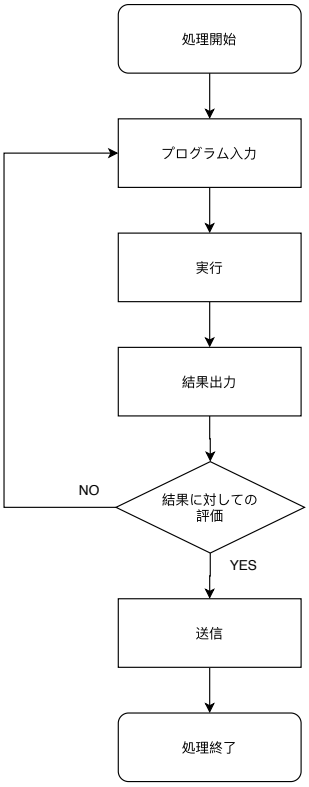



図 5.1: システムの流れ

### 5.2.1　学生側の画面

学生側の画面では以下の機能が含まれる。表示される内容は教員が入力できる仕組みになる。

- 問題文の表示

  問題文が表示される。基礎プログラミングでは教員が作成したページを Firefox で確認しながらプログラムをテキストエディタに書き込むという作業があるが、一つのページに問題文を表示することで手間の軽減を図る。

- プログラムに対する教員のコメントの表示

  教員からのアドバイスやヒント、どういった場面で使うかなどの情報を表示する。

- 出力回答例の表示

  あらかじめ成功した場合の出力例を出すことで、作成するプログラムのイメージを浮かべてもらう。

- プログラムを入力するテキストエディタ

  プログラムを入力するテキストエディタである。また実行のためのボタンを設置し、ボタンを押すことで実行してもらう。これによりコマンド操作を不要とする。

- プログラムの実行結果が表示されるエディタ

  プログラムが動いても動かなくても結果を表示する。後述するが、Kterm と違いプログラムの実行結果とエラー文の表示は別とする。

- エラー文の表示

  エラー文には URL リンクを貼ることで検索を容易にする。またエラー文はプログラムの実行結果と別の場所に表示する。

- データベースへの格納

  教員の採点のため、実行し成功したプログラム文はデータベースに格納する。検索を行えるように学籍番号またはユーザ名での登録を行う。

### 5.2.2 教員側の画面

教員側の画面では以下の機能が含まれる。アクセスは教員のみ可能とする。

- 問題文の登録

  解いてもらうプログラムの問題を設定する。問題文は学生用の画面の上部に表示される。

- コメントの登録

  出題される問題に対しての補足や説明を入力する。

- 実行例の入力

  学生側に表示される出力例（模範解答）を出力するためのプログラムを打ち込む。プログラムと出力結果の両方がデータベースに登録される。

- セクションの設定

  データベースでの索引のために設定する。ひと目見てどのメソッドについての出題かをわかりやすくするものである。

### 5.2.3 データベース画面

データベースに格納された情報を閲覧できるページである。アクセスは教員のみ可能とする。データベースのテーブルは 3 つに分けられる。以下にテーブルの説明を示す。

- ユーザテーブル (users)

  使用する学生、教員の情報が入ったテーブルである。ログイン認証などにも使用する。

| id_imm | id | password | flag |
|---|---|---|---|
| 主キー | 判別番号 | パスワード | 使用者特定用番号 |

- 教員テーブル (mixes)

  教員が入力した情報を始めとした、学生に表示されるデータが入るテーブルである。

| section_id | question | program | comment | answer | section |
|---|---|---|---|---|---|
| 主キー | 問題文 | プログラム | コメント | 出力結果 | セクション名 |

- 学生テーブル (students)

  学生の入力した情報などが入ったテーブルである。

| id | user_id | code | result | comment | success | section | time |
|---|---|---|---|---|---|---|---|
| 主キー | 学生番号 | プログラム | 出力結果 | コメント | 成功判定 | セクション名 | 提出時間 |

画面の機能は以下である。

- データベースに格納された情報の表示

  データベースに格納された情報を確認できる。個人判別のためのユーザ名とプログラム文を表示する。最低限の内容として学籍番号、プログラム文、実行結果、成功したかの判定結果が入る。

# 第6章　システムの開発

本章では、以上の機能を含めたシステムの開発を行う。

## 6.1　開発環境

開発環境を以下に示す。

- 開発言語
  - サーバサイド
    - Ruby(version 2.7.1)
      1995 年にまつもとゆきひろによって開発されたプログラミング言語である [14]。
  - クライアントサイド
    - HTML(HyperText Markup Language)
      ハイパーテキストを記載するためのマークアップ言語である [15]。
    - CSS(Cascading Style Sheets)
      ウェブページのスタイルを指定するためのスタイルシート言語[1]である [16]。
    - JavaScript(version 1.5)
      JavaScript はウェブページに複雑な機能を搭載できるようにするプログラミング言語である [17]。
- プラットフォーム[2]
  - Docker(version 19.03.13)
    コンテナ仮想化を用いてアプリケーションを開発・配置・実行するためのオープンソースソフトウェアである [18]。
- Web フレームワーク
  - Sinatra(version 2.0.8.1)
    Ruby で作成されたオープンソースの Web アプリケーションフレームワークである [19]。
- データベース
  - SQLite3(version 3.31.1)
    パブリックドメインの軽量な Relational Database Management System(関係データベース管理システム) である [20]。

[1]構造化文書の見た目を記述するコンピュータ言語
[2]ソフトウェアが動作するための土台として動作環境のこと。

- ブラウザ
  - Mozilla Firefox for Ubuntu(version 83.0 64bit)
    オープンソースの Web ブラウザである [21]。

## 6.2　実装した画面

本節では、作成した画面とそれに含まれる機能を説明する。

### 6.2.1　学生側の画面

学生側がプログラムを作成する際に見る画面である (図 6.1)。この画面では問題文の表示や作成の際の手助けとなる表示を行うことで、視覚的にプログラムを作成しやすくなることを目的とした。学生が打ち込んだプログラム文を始めとする入力した情報は全てデータベースへ格納される。プログラムを行う際には計算機で行うこと、Firefox 上でウィンドウサイズを最大にして見ることが前提である。機能は以下にあげたものを実装している。表示される画面の上部から説明を示す。

#### 入力エディタと出力ターミナルエミュレータ

画面の一番左側にある黒いエディタがプログラム文を入力する場所である。プログラムは複数行に渡るため、textarea タグを使用している。プログラム文は自作メソッドによって Docker と接続したのちコンテナ内で実行される。そして標準出力やエラー文が JSON 形式で返ってくるため、標準出力は右側の黒いターミナルエミュレータに、エラー文は右側の白いエディタに表示される。

プログラムが実行されるたびに逐一ページが遷移するのは、使用者の使い勝手を考えると不適切である。プログラムを打ち直す度に前のページに戻る手間が発生するからである。本システムではそれを避けるために、ページ遷移を用いずにデータを取得することができる Fetch API を使用し非同期のネットワーク通信を行っている。これにより 1 つの画面で何度でもプログラム文を実行することを可能にした。

プログラムの実行は Docker 内で行われている (図 6.2)。コンテナ内部でプログラム文を実行することにより、悪意のあるコードを入力された際にその計算機自体の情報を表示させないといった対応を可能にした。

実行結果が表示された際でも入力したプログラム文はそのままエディタ内に残る。これは長いプログラムを入力した際に失敗したことでまた始めから打ち直すことの手間を考え実装した。

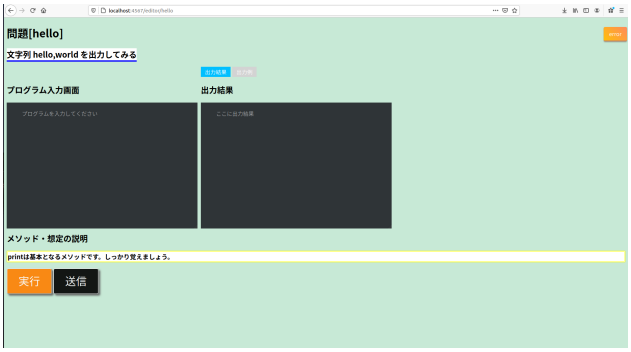



図 6.1: 学生画面

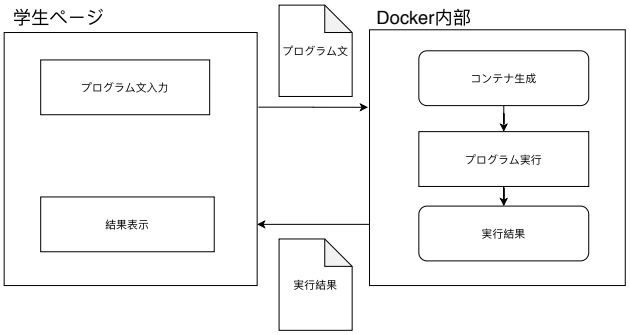



図 6.2: Docker 接続の流れ

### システムのアルゴリズムについて

自作したメソッドのソースコードを記載する。このメソッドは多数の使用者が本システムを使用した際、負荷がかかりすぎないように一定の時間以上が経過してもプログラムが実行されない場合は強制的に実行を停止する流れにした。これが起こる例としては while や for、再帰処理といったコードがあげられる。

docker_exec_ruby

```
def docker_exec_ruby(ruby_script)
  data = {
      stdin: ruby_script,
      stdout: nil,
      stderr: nil,
      errmsg:  nil
  }
  Open3.popen3("docker run ruby ruby -e '#{ruby_script}'") do
  |stdin, stdout, stderr, wait|
      #標準入力、出力、エラー出力、タイム処理
      # stdin に文字列を入力し、stdout、stderr から文字列を受け取り、
      # 必要に応じて wait スレッドを用いてタイムアウト処理を実装する
      begin
          Timeout.timeout(50) do
              stdin.close
              # data ハッシュの対応するキーの値に入れる
              data[:stdout] = stdout.read
              data[:stderr] = stderr.read
          end
      rescue Timeout::Error
          data[:errmsg] = "タイムアウト"
          begin
              p Process.kill(:INT, wait.pid)
              p wait.join
          rescue Errno::ESRCH
          end
      end
      return data
  end
end
```

#### 問題文の表示・出力結果例・メソッドや想定の説明の表示

視覚的に表示されている機能の説明である。表示される内容は全て教員が入力したものである。

- 問題文の表示

  画面の上部には問題文を表示している。これにより教員の講義資料などの問題文が書かれている画面を見に行く手間を省いている。文字数の制限はないため長い文章にすることも可能だが、長くするほど画面が下に伸びていくため推奨されない。

- 出力結果例の表示

  出力ターミナルエミュレータで切り替えが可能である。デフォルトではプログラムの出力結果が表示されるが、出力結果と出力例のタブを切り替えることで出題した教員が設定した出力例を見ることが可能である (図 6.3)。

  この機能は、学生が実際にプログラムの実行結果を見た際に「本当にこれであっているのか」と不安に感じることへのフォローである。基礎プログラミングでは、プログラムを実行した際に、合っているにも関わらず確認を行うケースが多くみられた。そのためあらかじめ任意で出力例を見せることにより、これを表示させれば目的達成という指標として実装した。

- メソッドや想定の説明の表示

  学生がどうしてプログラムに取り組むのか、こういったプログラムはどこで使われるのかといった想定をはじめ、メソッドの説明などが表示される。そもそも何を使えば良いのかわからないといった問題文から読み取れない補足の説明などがここに表示される。

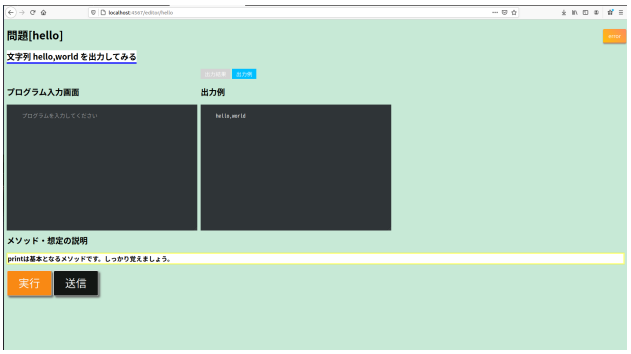



図 6.3: 出力例の表示

### ユーザ名入力欄とコメント入力欄

それぞれプログラムの実行結果を送信する際に要求される。現時点では値を入れなくても送信が可能である。

- ユーザ名入力欄

  プログラムを入力する前にユーザ名を入力する (図 6.4)。本学では学生 1 人につき学籍番号が割り振られているためユーザ名は学籍番号と同じになる。ユーザ名があることにより、どの学生がどのプログラムを作成したのかというのを分かりやすく表示することができる。プログラムの実行結果の送信時に入力用ダイアログを表示し、入力を促す仕様になっている。

- コメント入力欄

  プログラムを作った際に感じた疑問や感想などを自由に入力するための欄である (図 6.5)。ユーザ名入力欄と同様に、送信ボタンを押した際に入力ダイアログが表示される。


図 6.4: ユーザ名入力欄


図 6.5: コメント入力欄

### エラー文の表示

実行に失敗していた場合は右画面にエラー文が表示される (図 6.6)。エラー文をそのまま検索エンジンで検索できるように検索エンジンを直接開く動作を付与した。

また、エラー文のみでは他の言語を含んだ検索結果が表示される可能性があるため、Ruby のエラー文として検索できるように文字列 “Ruby” をエラー文の文字列に追加した。表示されるエラーによっては注釈を加えている。現時点では NameError[3] と end[4] に対しての注釈を行っている。

[3] 変数が定義されていない場合に起こりえるエラー文の一種
[4] while などの構文において end を付け忘れた際に起こりえるエラー文の一種


図 6.6: プログラムの実行に失敗した場合の表示

注釈を出すためのループ

```
// もしエラーの中に特定の文字があったら
var error = result.stderr

var hash = {
        NameError: '変数の設定おかしいよ！',
        end: 'end 付け忘れてるかも！'
};

if (error != "") {
  for (let key in hash) {
    //console.log(key);
    if (error.indexOf(key) > -1) {
      var a = document.getElementById("tuketashi").innerHTML =
              key + hash[key];
    }
  }
} else {
    var a = document.getElementById("tuketashi").innerHTML = "";
};
```

### 6.2.2 教員側の画面

教員が打ち込む画面である (図 6.7)。ここで入力された情報が学生側の画面に表示される。入力できる内容を以下に説明していく。

#### 問題文・コメントの設定

学生側の画面に表示される部分である。入力する内容は使用する教員によって自由に決まる。文字数の制限はない。ここで入力されたデータもデータベースに格納される。

#### 実行例の設定

学生ページの出力例に表示される結果を実行するためのエディタを用意した。ここにプログラム文を打ち込むと実行された結果が学生ページに反映される。プログラム文自体は反映されない。学生ページの入力エディタと同様に Docker 内部でコードが実行されている。

#### セクション名

どんなメソッドを使うかといった目次となるための入力部分である。例えば入出力に関しては print 文を使用する場合と puts 文を使用する場合がある。その際にどちらのメソッドを使わせるのを目的としているかを明確にするものである。


図 6.7: 教員ページ

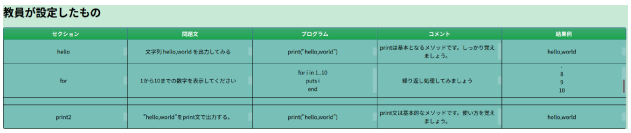
### 6.2.3　情報閲覧画面

本システムには閲覧できるデータベーステーブルが 2 つある。それに伴って閲覧ページも 2 つ作成した。1 つが学生が入力した情報が格納されるテーブル (student) が表示される学生テーブル参照ページ (図 6.8) である。2 つ目が教員が入力した情報が格納されるテーブル (mixes) が表示される教員テーブル参照ページ (図 6.9) である。


学生データ

| 学籍番号 | プログラム | 実行結果 | コメント | 成功 |
|---|---|---|---|---|
| C1172081 | print("hello,world") | hello,world | print文マスターした | ○ |
| C1303333 | puts"hello,world" | hello,world | putsも何じゃん！ | ○ |
| C1304555 | puts"hello,world | | エラーでちゃった | △ |

図 6.8: 学生テーブル参照ページ


教員が設定したもの

| セクション | 問題文 | プログラム | コメント | 結果例 |
|---|---|---|---|---|
| hello | 文字列 hello,world を出力してみる | print("hello,world") | printは基本となるメソッドです。しっかり覚えましょう。 | hello,world |
| for | 1から10までの数字を表示してください | for i in 1..10<br>puts i<br>end | 繰り返し処理してみましょう | 1<br>2<br>3<br>10 |
| print2 | "hello,world"をprint文で出力する。 | print("hello,world") | print文は基本的なメソッドです。使い方を覚えましょう。 | hello,world |

図 6.9: 教員テーブル参照ページ

# 第7章　システムの実行速度における実証

本章では第6章で実装したシステムを実際に使用し、要件と照らし合わせて評価を行う。実験では本システムを使用した際にプログラムの実行から出力結果までの時間を計測し、使用者の使い勝手の向上に向けた考察を行う。

## 7.1　評価観点

本システムにおいて評価を行うべき点は以下である。

- システムを使うことで学習効果に影響が出るか
- システムを使うことで使用者に影響が出るか
- システムの機能のひとつである「プログラムの実行、出力結果の表示」のパフォーマンス

本研究では、プログラミング初学者が学びやすい環境を作るためのシステムの構築を行っている。学びやすい環境の構築には実行速度などのパフォーマンスの最大化を行うことが最低限必要である。そのため、今回の実験では3つ目の「プログラムの実行、出力結果の表示」についてのパフォーマンスを計測した。

## 7.2　事前準備

事前に本システムを起動し、実験用の設問を行った。設問の内容は以下のとおりである。

問題文: 1から100までの数字を出力して下さい。

コメント: while でも for でも何でも使って下さい。

セクション名: 1..100

また、今回の実験は、システムの応答速度を分析するためにネットワーク速度の違う場所に人員を配置した。その際、被験者は全員 Jitsi Meet[1]で会話することを前提とている。

## 7.3　実験

本学の Ruby の基礎知識を持った被験者4名で同時に本システムを稼働させ、動作を確認した。システムを利用する上で人数が増えるとプログラムが実行されるまでの時間にどう影響が出るかを検証した。被験者はそれぞれ A〜D とする。
実験の手順は以下のとおりである。

[1]フリーかつオープンソースのビデオカンファレンスアプリケーション

1. 無線 LAN 接続速度の計測

   あらかじめ各自の無線 LAN 接続環境における通信速度を計測する。ネットワークの疎通を確認する際に使用する ping コマンドにより、10 個分のパケットの送信時間から平均値を確認した。この数値が高いほど動作が遅くなる。

2. 本システムへのアクセス

   本システムを使用するために事前に VPN 接続を行ってもらった。URL リンクは `http://IP アドレス/editor/セクション名` である。今回の実験では `http://172.19.5.56/editor/1..100` である。

3. 実際にプログラムを打ち込み、実行する

   あらかじめ想定されたソースコードをコピーペーストで貼り付けてもらった。コードは以下のとおりである。

   実験用ソースコード

   ```
   for i in 1..10
   puts i
   end
   ```

   このコードを実行し、結果が被験者の画面に表示されるまでに何秒間かかるかを計測した。計測は被験者自身がストップウォッチを用いて行った。プログラムの送信と同時にストップウォッチを起動し、結果が表示されると同時にストップウォッチを止める。手動のため誤差が出る可能性もあるが今回は考慮していない。平均値を取るために 5 回、同じソースコードを用いて実行した。

4. プログラムの情報を登録する

   打ち込んだ情報と実行結果をシステムのデータベースに保存した。

実験の様子を図 7.1 に示す。また、図の説明を以下に示す。

1. A は SSH 経由で本システムサーバに接続し、システムを起動している。
2. システムには各自計算機の Firefox からアクセスしている。
3. 本システムはセキュリティ対策として閉じられたネットワーク上で動いているため、VPN サーバへの接続が必要である。そのため、被験者 A〜D はあらかじめ VPN サーバに接続している。
4. A〜D のパケットは全て VPN を経由している。
5. 各ユーザはブラウザを起動して本システムに接続する。同じくブラウザでビデオ会議サーバに接続し、音声会話による指示 (画面はオフ) を聞きながら操作している。

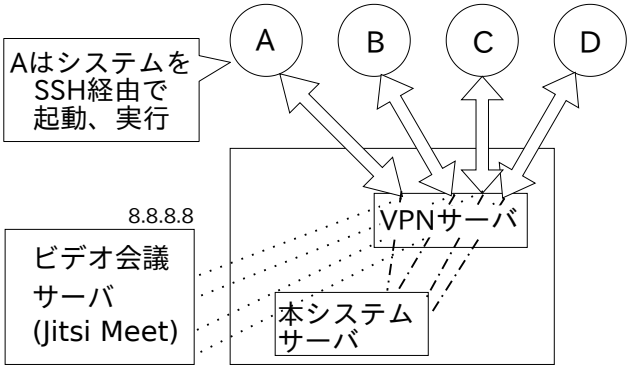



図 7.1: 本システムと各クライアント接続の概要

## 7.4 実験結果

実験結果は以下のとおりである。

無線 LAN 接続の結果より B、C が特に速度が安定していることが分かった (表 7.1)。

プログラムを実行し、それぞれ表示までの時間を計測した (表 7.2)。

5 回計測したそれぞれの平均値は昇順で B の 3.39 秒、C の 4.41 秒、D の 4.67 秒、A の 12.39 秒であった。全体の平均値は 6.21 秒 (小数点 3 位以下四捨五入) である。また、A 単体でプログラムを実行した場合は 2.31 秒であった。

表 7.1: 無線 LAN 接続速度の計測結果 (単位:ms)

| | 最大 | 最小 | 平均 | 標準偏差 |
|---|---|---|---|---|
| A | 88.777 | 103.818 | 162.170 | 20.602 |
| B | 51.009 | 56.170 | 60.469 | 3.117 |
| C | 59.405 | 66.553 | 72.996 | 3.945 |
| D | 79.822 | 111.325 | 177.965 | 29.830 |

表 7.2: プログラムを実行したときの出力速度 (単位:s)

| | 1 回目 | 2 回目 | 3 回目 | 4 回目 | 5 回目 | 平均 | 標準偏差 |
|---|---|---|---|---|---|---|---|
| A | 15.13 | 7.95 | 13.58 | 19.71 | 5.58 | 12.39 | 20.602 |
| B | 4.15 | 3.84 | 4.62 | 2.90 | 4.13 | 3.93 | 3.117 |
| C | 3.86 | 7.48 | 4.06 | 2.68 | 3.95 | 4.41 | 3.945 |
| D | 6.41 | 6.69 | 4.33 | 2.61 | 3.79 | 4.67 | 29.830 |

## 7.5　実験における考察

今回行った実験から考察を述べる。

無線 LAN 接続の速度に関して、安定した B と C に比べ D は若干の不安定さが見られたが、実験結果による数値では B、C と遜色ない時間だった。これにより、無線 LAN 接続環境に多少の速度の違いがあっても平均 6 秒程度でプログラムが実行されて使用者に結果が渡されることを確認できた。また、4 名で行った場合の平均値が 6 秒に対し、A が 1 名で行った際には 2 秒程度で結果が返って来たため、同時に接続、実行すると遅れが見られる傾向があることが判明した。しかし、ユーザがウェブページを閲覧または使用する際のページの読み込みは、およそ 2 秒以内で行われない場合にストレスを感じるという結果がある [22]。そのため、今回の実験から見えた本システムの平均値である 6.21 秒は非常に遅い結果となった。また、今回の実験では 4 名での実行だったが、基礎プログラミングではおよそ 40 名が一斉に実行することを考えるとさらに平均値は伸びる恐れがある。

# 第8章　考察

本章では、第6章、第7章の結果を踏まえて考察する。システムは動作確認を行い、方策を実現したシステムを実装できたことを確認した。また、7章で得られた結果により、実際にプログラムを実行して結果を得ることができることから、講義でも利用可能なことが分かった。実証の際に起こった問題について深く考察していく。

本システムを使用する上で発生する問題を上げる。また、それに対する考察を述べる。

- エラー文が発生しない場合

  プログラムによっては、プログラムが失敗していなくてもエラー文が表示されないプログラムが存在する。制御構造の while や for、upto を用いて問題に答えるものは、演算子が間違えている際にはプログラムの実行結果が表示されず、さらに解決の手法となるエラー文も表示されていなかった (図 8.1)。例のプログラム文は、i が 10 になるまで i に代入された数字を出力する、といったプログラムだが、演算子が `while i>= 10` により、i が 1 だった場合、10 以上になることはないため出力は行われない。この結果は、エラー文を分かりやすく表示することで苦手意識をなくす、といった目的の大きな妨げになる。しかし、こういった場合はプログラムが実行に失敗しているのではなく、エラーが出ず、結果があっていない、というものであるため、具体的な解決案を示すのが困難である。実行するプログラムの中で予想外の結果が起こった場合に対する表示方法を考えていく必要がある。

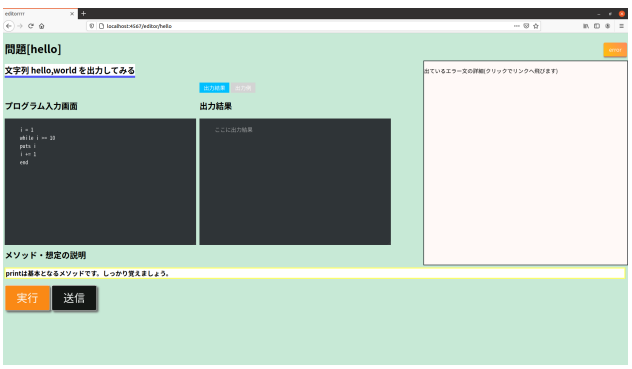



図 8.1: エラー文がでない場合

- 演習形式の講義でも利用できるか

  本システムが実際に演習形式の講義でも使用できるか可能かどうかを論点にする。実験的に本システムを運用した際には print メソッドを使った標準出力や while、for などの制御構造を扱うことができた。逆にターミナルエミュレータから入力を受け付けるメソッドである gets メソッドや外部ファイルを必要とする open メソッドなどは使用不可である。そのため本システムを使うことで作ることができるプログラムは以下の二点である。

  – 単純な標準出力
  – 制御構造

  制御構造に関しては極端に時間がかかるものは実行不可能である。

  配列や正規表現といったものもターミナルエミュレータから出力を得ないものであれば解くことができる。以下のコードは実際に講義で使用されており、また本システムでも実行を確認した例である。

配列

```
box=["a", "i", "u", "e", "o"]

puts box[0] #1 番めの値を取り出す
```

メソッド

```
def tax(price, ratio)   # price に値段、ratio に税率 をもらう
  charge = price*ratio
  charge.to_i
end

puts tax(250, 1.10)
```

Hash

```
hashbox=[
         {"alp"=>"a", "ex"=>"apple"},
         {"alp"=>"b", "ex"=>"blue"},
         {"alp"=>"c", "ex"=>"crick"},
         {"alp"=>"d", "ex"=>"delete"},
         {"alp"=>"e", "ex"=>"error"}
        ]

puts hashbox[0] #1 番めの値を取り出す
puts hashbox[0].keys #1 番めの key を取り出す
puts hashbox[0].values
```

再帰処理

```
def hanoi(n, from, to, via)
  if n == 1
    puts "#{from} から #{to} へ移す"
  else
    hanoi(n - 1, from, via, to)
    puts "#{from} から #{to} へ移す"
    hanoi(n - 1, via, to, from)
  end
end

hanoi(3, :A, :B, :C)
```

なお再帰処理における hanoi(3, :A, :B, :C) の 3 を 21 にした時点でタイムアウトに達したほか、コンソールで Process:Status の値が 1 で返ってきたことから、thread 処理によってプロセスが強制終了されたことが分かった。

- 実行する際の時間と外部からの影響について

  実証では 4 人の時点で約 6 秒のレスポンスが確認された。実際の講義で運用する際に 1 人につき 6 秒間かけて結果が送られてくるのは非常にストレスになることが考えられる。また、タイミングによって 6 秒は 2 秒になることもある。この時間のばらつき具合が実際の講義でどう影響するかを確認する必要がある。また動作が遅い点について考えられる原因を以下に示す。

  – 無線 LAN 接続など個人の環境による影響
  – 本システムのプログラムを実行する機能の不備
  – 同計算機上で他の作業を行っているか

  無線 LAN 接続の速度は実証の際に影響があまりないことが確認されたが、顕著に不安定な場合は影響が出る可能性がある。

  次に考えられるのが本システムにおけるプログラムを実行、表示までの動作である。本システムではシステム利用者が一つのプログラムを実行するたびに Docker のコンテナを起動している。これにより、他の人が続けてコンテナを起動し続けることで遅れが出る結果になっていると考えられる。

  最後に同一の計算機上で他の作業を行っているかである。実証では A のみ本システムを起動しているという特殊な状況であり、実験結果も B〜D に比べて顕著な時間の遅れが見られた。このことから、同一計算機で何か他の動作をしている場合には速度に影響が出ることが考えられる。しかし、実証では Jitsi Meet を使用し、実質他の作業を行っている状況であったため、一概に他の作業が本システムの稼働に影響するとは言えない。

# 第9章　まとめ

本章では、結論と課題、今後の展望について述べる。

## 9.1　結論

本研究ではプログラミング学習支援システムを提案しシステムの構築を行った。本システムを利用することで、基本的な入出力と制御構造を使用するプログラムなどを作成し実行することができた。提案をもとに設計を行って実装し、運用した上で評価を行った。実証では本システムを使う際の問題点が明らかになった。システムの例外や問題を明らかにし、さらに運用できる範囲を広げていく必要がある。

## 9.2　課題と今後の展望

第 8 章と 9.1 節を踏まえて、今後の課題を明らかにする。またこれをもとに今後の展望を述べる。

### 9.2.1　使用できる問題の範囲

本システムで行えることはプログラムの実行と結果をデータベースに登録することである。しかし、本システムで取り組むことができるプログラムには限りがある。現時点で実装できていない標準入力の受け付けや外部プログラムを組み込むためのシステムを実装することで、更に使用できる範囲や実行できるプログラムが増えていくと考えられる。

### 9.2.2　教員の採点への補助

データベースの閲覧は可能だが検索機能が実装されていないため、学生個人の情報を見つけることが難しい。早急に検索システムを実装する必要がある。また、実証では教員側の採点における手間が本システムによってどう軽減されたかの検証を行っていない。そのため、教員側にも本システムを運用してもらい実際に採点の手間の軽減に繋がるかを考える必要がある。

上記に上げた課題を解決しつつ、本システムを以下の Git リポジトリで公開し、誰でも使えるようなオープンソースのソフトウェアにする。現時点の完成版は以下のとおりである。

https://www.yatex.org/gitbucket/Madoka/editor_Sinatra

# 謝辞

本研究を進めるにあたり、指導教官の廣瀬雄二准教授からは多大な助言を賜りました。三浦彰人特任助教には研究の指導を通じて多くの知識や示唆を頂きました。また、東北公益文科大学広瀬ゼミの皆さまには、実験において被験者を務めていただきました。ここに感謝の意を表します。

# 参考文献


[1] 文部科学省. “「高等学校学習指導要領」(平成 30 年告示)(2019)”. (参照 2019-12-3).

[2] 文部科学省. “教育の情報化の推進”. http://www.mext.go.jp/a_menu/shotou/zyouhou/detail/__icsFiles/afieldfile/2019/10/03/1421730_001.pdf. ( 参照 2019-11-18).

[3] 文部科学省. “小学校プログラミング必修化に向けて”. http://www.mext.go.jp/b_menu/shingi/chukyo/chukyo3/004/siryo/__icsFiles/afieldfile/2018/10/05/1409851_6.pdf. ( 参照 2019-11-15).

[4] 文部科学省. “小学校プログラミング教育の手引の改定 (第二版)”. http://www.mext.go.jp/component/a_menu/education/micro_detail/__icsFiles/afieldfile/2018/11/06/1403162_02_1.pdf. (参照 2019-11-18).

[5] 文部科学省. “小学校プログラミング教育の手引の改定 (第二版) について”. http://www.mext.go.jp/component/a_menu/education/micro_detail/__icsFiles/afieldfile/2018/11/06/1403162_01_1.pdf. ( 参照 2019-11-18).

[6] 文部科学省. “プログラミング教育に関連する研究教材”. http://www.mext.go.jp/component/a_menu/education/micro_detail/__icsFiles/afieldfile/2019/05/21/1417094_004.pdf. ( 参照 2019-11-18).

[7] 野口孝文, 千田和範, 稲守栄. “初学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 vol 32,No.1 2015 pp.59-70. (参照 2020-12-10).

[8] paiza ラーニング. “環境構築不要！初心者でも楽しく学習できるプログラミング入門サービス【paiza ラーニング】”. https://paiza.jp/works. (参照 2019-12-4).

[9] Progate. “Progate | プログラミングの入門なら基礎から学べる Progate[プロゲート]”. https://prog-8.com/. (参照 2019-12-4).

[10] 中西渉. “Web ブラウザ上のプログラミング学習環境 WaPEN の改良”. 情報教育シンポジウム論文集 .2019,130-135,(参照 2019-11-12).

[11] WaPEN. “自作プログラム”. https://watayan.net/prog/. (参照 2019-12-4).

[12] 新田章太, 小西俊司, 竹内郁雄. “複数言語に対応しやすいオンラインプログラミング学習・試験システム track”. 情報教育シンポジウム論文集.2019,114-121,(参照 2019-11-12).

[13] track. “エンジニアの採用と育成を支援するプログラミング「学習・試験」プラットフォーム”. https://tracks.run/,(参照 2019-12-4).

[14] Ruby. “オブジェクト指向スクリプト言語Ruby”. https://www.ruby-lang.org/ja/,(参照 2020-12-3).

[15] HTML. “HTML Living Standard — Last Updated 3 December 2020”. https://html.spec.whatwg.org/multipage/,(参照 2020-12-7).

[16] CSS. “Cascading Style Sheets home page”. https://www.w3.org/Style/CSS/,(参照 2020-12-7).

[17] JavaScript. “Welcome to Ecma International”. https://ecma-international.org/,(参照 2020-12-7).

[18] Docker.“Get Started with Docker”.https://www.docker.com/,(参照 2020-12-7).

[19] Sinatra. “Sinatra”. http://sinatrarb.com/,(参照 2020-12-7).

[20] SQLite. “SQLite Home Page”. https://www.sqlite.org/index.html,(参照 2019-12-5).

[21] Firefox. “moz://a firefox Browsers”. https://www.mozilla.org/ja/firefox/,(参照 2020-12-5).

[22] Akamai. “State of the Internet Connectivity Reports - 2014”.https://www.akamai.com/us/en/resources/our-thinking/state-of-the-internet-report/archives/state-of-the-internet-connectivity-reports-2014.jsp,(参照 2021-1-8).

# 付 録A　アンケート

質問1. プログラミングについてどう感じますか。

質問2. 大学入学前にプログラミングを体験した経験がありますか？
(Progate など学習サイトを含む)

□ ある
使用した学習サイトなどあれば記入をお願いします。
(　　)

□ ない

質問3. 基礎プログラミングの難易度はどのぐらいですか。
簡単　　少し簡単　　普通　　難しい　　とても難しい
□ 1　　□ 2　　□ 3　　□ 4　　□ 5

質問4. プログラミングで特に難しいと感じた部分はどこでしたか。

□ 動かない・エラー文が読めない
□ メソッドが覚えられない
□ そもそも組み方がわからない

□ その他 (　　)

質問5. プログラミングを楽しいと思う・理解したきっかけは何でしたか。

□ 動いたとき
□ 難易度が高いプログラムを作れたとき
□ 緻密な作業 (デバッグなど)
□ その他 (　　)