プログラミング教育の補助システムの提案

広瀬研究室3年
C1172081 山口円馨

令和2年1月15日

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。現在、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は困難であり、学生数が多いと動作確認まで行い正確な評価をするのは更に困難を極める。本研究では、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、プログラミングにおける評価の補助システムの作成を目的とする。本研究の基本方策は、プログラミング学習における一連のサイクルを分析し、コンピュータの機能を使用して処理する部分は機能を使用し、人の手で行う部分にはより作業をしやすい表示を行う。まず最初に、学生側と教員側のサイクルの作業の切り分けを行った。この結果、コンピュータが処理する部分はプログラム文の実行や採点であると考えた。また、人の手で行う部分はプログラムの理解への説明や、個人単位の総合的な配点である。これに対してはコンピュータ上の表示により用意な環境を整えることが必要である。その次に、この設計に基づいて実装し、実装上の工夫について議論した。さらに実際の運用を経て課題を明らかにした。今後の課題については、学生側が見て理解できる表示や、教員側が採点をする際の表示を使用する側の目線から明らかにすること、さらに脆弱性を議論した。(600 文字)

目 次

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 年度から小中高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。なお、中学校から実際のプログラミングが始まるが、これに対して、教職員の実態と意識調査の結果では、98 %が「授業の実施に不安」を感じているといった結果が出た [7]。また、第 3 章で述べるアンケート結果により、プログラミングに対しては「難しい」、「将来役に立つのかわからない」「動くと楽しい」といった意見が見られた。この結果を受けて、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、プログラミングにおける評価の補助システムの重要性があると考え、検討した。

1.2　基礎プログラミング

東北公益文科大学 (以下本学) 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。当講義では、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エディタ (Emacs)、右側のプログラムの実行結果を見るターミナルエミュレータが 2 つ (Kterm)(console)、教員が作成した講義ノートを見る Web ブラウザ (Firefox) が表示されている (図 1.1)。学生は、講義の説明を受けた上でプログラミングを実際に行う。その際、一つのプログラムの構成を説明するため、短いプログラム文などが存在する。それらを Web ブラウザの画面を目視し、エディタへと書き込み、ターミナルエミュレータで実行する、といった一連の流れが発生する。講義の最後に課題が出され、学生は期限までに課題をメール (Mew) 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必要がある。教師側の一連の流れは以上になる。一連の流れとしての工数は多くないが、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に増加する。また、講義内で説明として表示されているプログラム文を実行する時間には個人差があり、考慮すると説明すべき点を十分に説明できない事態となる可能性がある。

これらの講義は必修科目であり、在学する学生が必ず受講する。初めての取り組みに苦戦し、プログラミング自体に苦手意識を持つ学生が見えた。さらに、プログラムのエラーで生じるエラー文を読み解くことができないことも苦手意識をもたせる原因となっている。

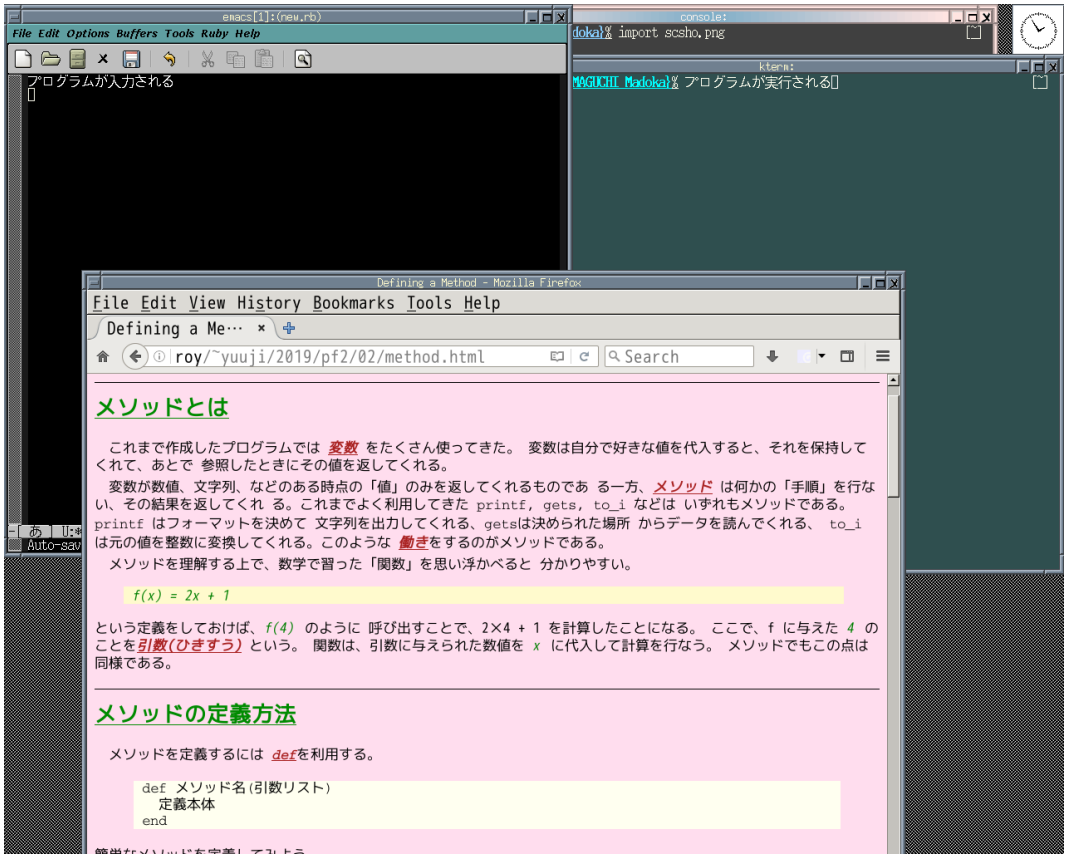

図 1.1: 基礎プログラミングの講義中の画面

1.3 目的

現状の課題を解決するために、プログラミングにおける評価の補助システムの作成を提案する。

1.4 研究内容

本研究は以下の手順を踏まえ、最終的な課題を明らかにしていく。

1. 問題点を見つける

   プログラミング教育の現状や既存のシステムや研究を比較することで、具体的な問題点を明らかにし対策を講ずる。

2. 基本アイデアの提案

   対策をもとに、システムの提案を行う。この際、どういった部分をコンピュータのシステムで補うか、また人の手で行う作業はどういった方法で補助するか、といった点を考える。

3. システムの設計

   前提と要件を決めた上で基本方策に則り、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. 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研究していく。

1.5 本稿の構成

本稿では、第 2 章では既存の研究やシステムを分析する。第 3 章では問題点を明らかにし、対策を講じる。それをもとに第 4 章ではシステムの方策を決めを、第 5 章ではシステムの設計を行い、第 6 章で開発、評価する。評価を元に第 7 章で考察し、第 8 章でまとめとする。

第2章　既存のシステムや研究

既存のシステムであるプログラミング学習サイトの説明を示す。

2.1　システム

本研究と同様のシステムを搭載している既存のシステムを説明する。主に企業が作成している。

2.1.1　paiza ラーニング

ギノ株式会社が開発した転職支援サイトである [8]。実行画面は図 2.1 のとおりである。目的別に名称が分かれており、プログラミングを行う学習サイトが paiza ラーニングである。動画を見ながら学習する。各言語の最初の動画では言語の説明や使うことで何が出来るか、採用されている例はどういったものかの説明が行われる。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。有料会員になることで実際のエンジニアに質問を行えるようになるなど、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出来る。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。

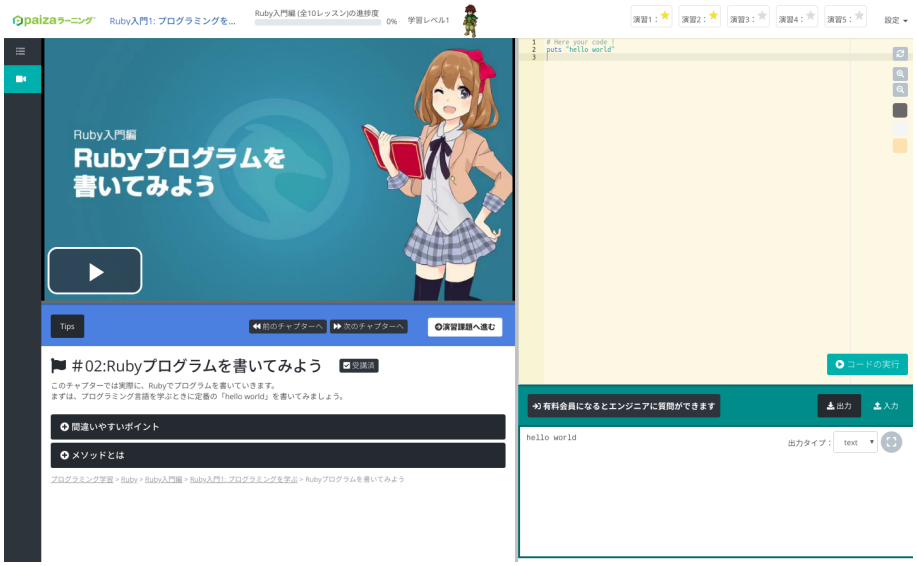

図 2.1: paiza の実行画面

2.1.2　Progate

株式会社 Progate が開発した、アプリケーション化もされており、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なプログラミング学習サイトである [9]。実行画面は図 2.2 のとおりである。はじめにスライド式で言語や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。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。

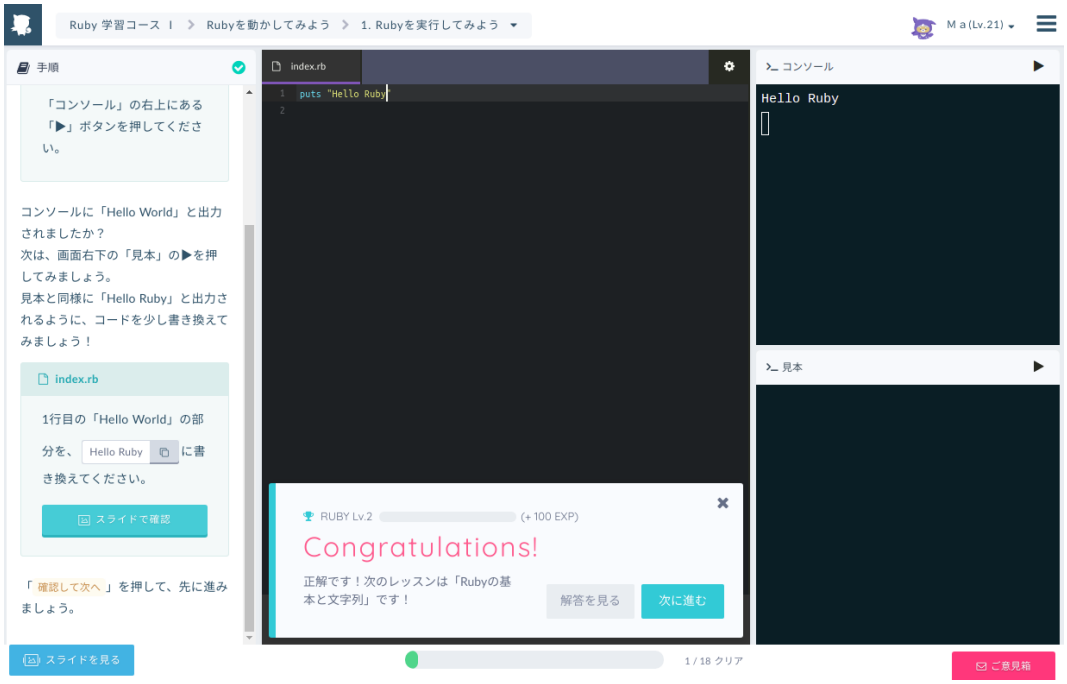

図 2.2: Progate の実行画面

2.2 類似研究

類似の研究では、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った。以下に説明を示す。

2.2.1 Wapen の改良

中西による Web ブラウザ上のプログラミング学習環境 WaPEN の改良 [10] では、プログラミング学習環境を Web ブラウザ上で実行できる環境として WaPEN [11] を開発し, 勤務校での授業で用いている。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。この開発環境では、プログラムをコードとしてではなく日本語としての説明として入力、フローチャート化するため、流れを理解するためのシステムである。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。

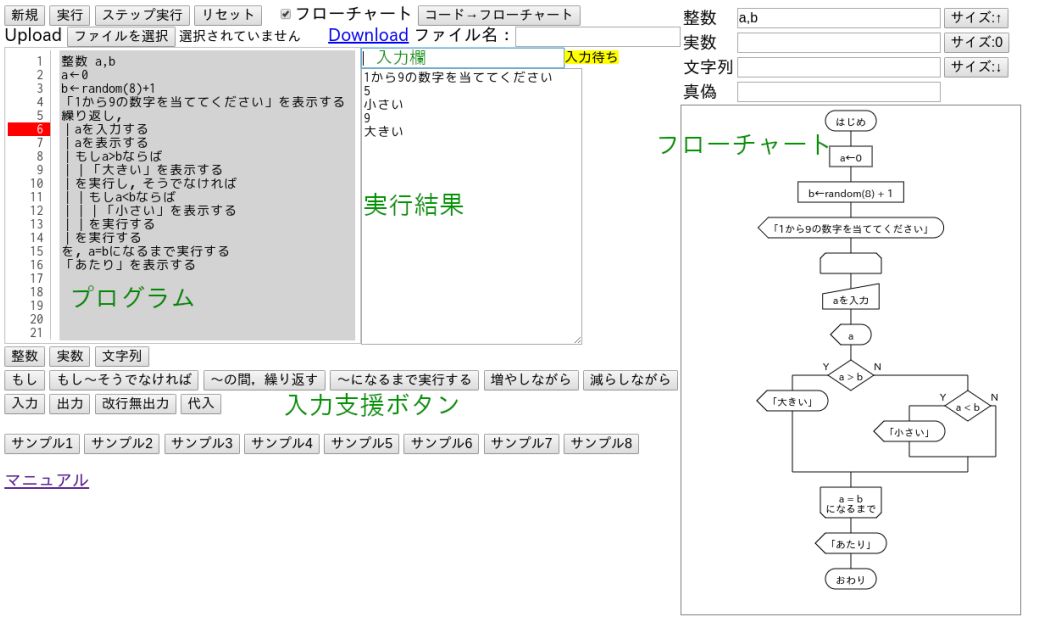

図 2.3: WaPEN の使い方

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 [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た料金プランが用意されている。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、または人にを対象にしているが、初学者が学ぶ際に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　プログラミング教育の問題および対策

本章では、プログラミング教育の問題点をアンケートにより明らかにし、その対策を議論する。

3.1　調査

本節では、初学者である基礎プログラミング受講者が授業を受ける際に、難しいと感じている現状や問題点をアンケートを用いて集計した。これにより問題点を明らかにした。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)

質問 3、質問 4 に対する結果は図 3.1,3.2 のとおりである (2019-11-19 時点)。

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、また特に難しいと感じた部分では選択肢を 3 つに絞った内、ほぼ均等に苦手意識が現れていることが分かった。第 2 章のようなプログラミング学習サイトを大学入学前に経験している学生の割合は 4.43%となった。アンケートの結果は図 3.3 である。

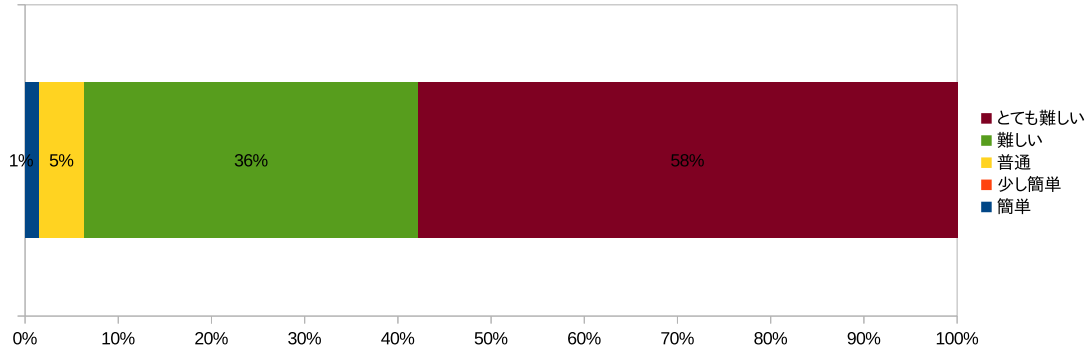

図 3.1: 基礎プログラミングの難易度

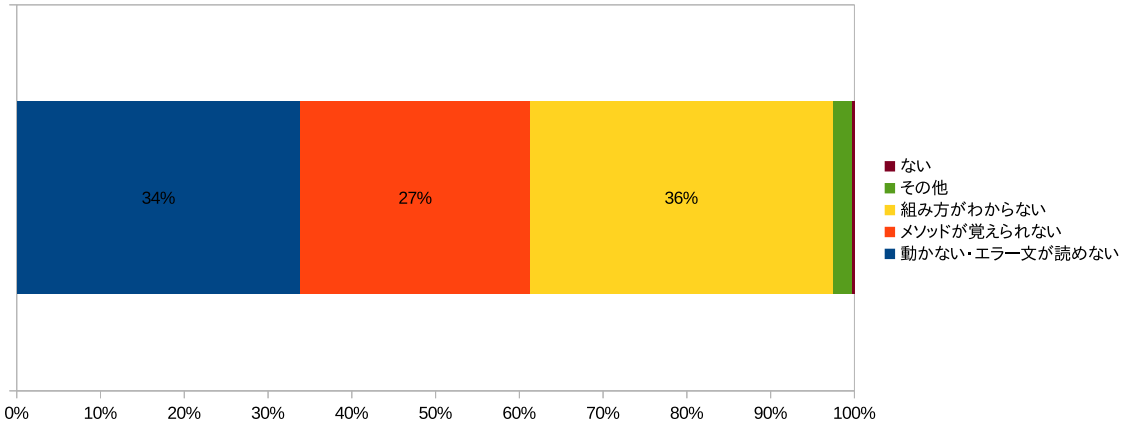

図 3.2: プログラミングで特に難しいと感じた部分

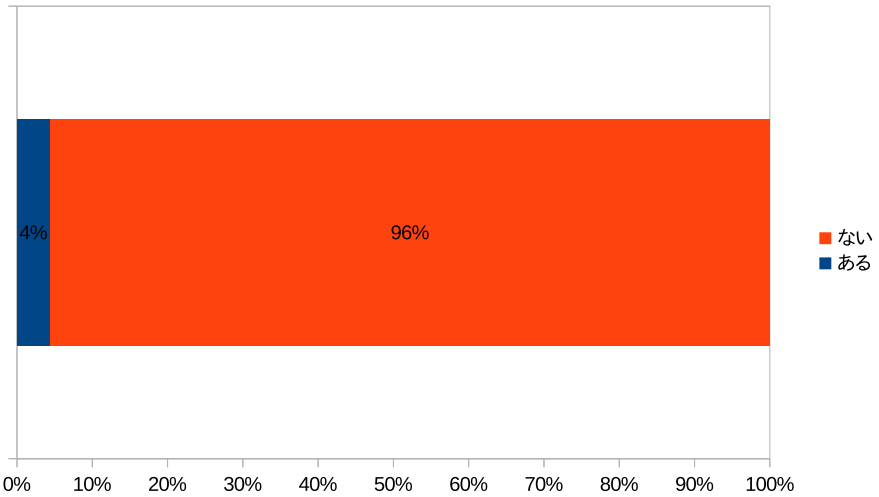

図 3.3: プログラミング経験の有無

3.2 問題

第 3.1 章と第 1.2 章、第 2 章の結果を経て学生側と教員側、プログラミング学習サイトの問題を提示する。

3.2.1 学生側

学生側の問題としては以下が挙げられる。

問題 1 プログラムを活用する機会が分からず、意識の向上がない。

プログラミング教育を受ける際に、プログラミングをどこで活かすのか分からないといった声がアンケートで上がった。これにより学習の意欲が薄い。

問題 2 苦手意識がある。

基礎プログラミングを受講する学生はアンケート結果により、プログラミングを経験していない初学者である。そのため、初めて経験するプログラミングに対しての苦手意識がある。

問題 3 エラー文の解読ができない。

問題 1 に関連する。初学者は英語のエラー文が出た時点でプログラムの作成を諦める傾向があった。検索エンジンを使用するという意識も薄い。そのため、エラー文で挫折する学生がいた。

3.2.2 教員側

教員側の問題としては以下が挙げられる。

問題 4 プログラムの評価に手間がかかる。

教員はメールで送信された課題に対して、目視した上で実行が可能か不可能かひと目で分からないものは実際に動かす必要がある。学生が全員プログラムが動作する状態で送っているなら目視のみで終わるが、実行不可能なプログラムがあると評価には極端に時間がかかる。

3.2.3 プログラミング学習サイト

プログラミング学習サイトの問題としては以下が挙げられる。

問題 5 決まった問題にしか取り組むことが出来ない。

プログラミング学習サイトではあらかじめ問題が設定されており、任意のプログラムを書くことには適していない。

問題 6 企業が制作しているため、本格的な学習を行うには料金が発生する。

学生が独自に学ぶ際にも、料金が発生していることはあくまで学生生活のみでプログラミング学習をすることを前提としている初学者には使うことをためらう要因である。

3.3　対策

第 3.2 節で示した問題に対しての対策案を以下に示す。

3.3.1　学生側

学生側には表示によって視覚的に情報を伝え、また苦手意識の原因となる部分を改善していく。学生側の問題への対策を以下に示す。

対策 1 プログラミングはどういったものか、説明を視覚的に表示する。

対策 2 エラー文をより分かりやすく、調べやすく表示する。

3.3.2　教員側

教員側のヒューマンリソースを鑑みて、評価までの一連のプロセスをシステム化することで手間の軽減を図る。教員側の問題への対策を以下に示す。

対策 3 簡易なプログラム文を実行できるエディタの用意。

対策 4 プログラム文の判定を自動で行う機能の実装。

3.3.3　プログラミング学習サイト

プログラミング学習は教員によって指導方法が異なる。どのような方法にも柔軟に応じるために、問題文は固定したものではなく自由に変更できる必要がある。プログラミング学習サイトで起こる問題に対しての対策を以下に示す。

対策 5 問題を教員側が自由に設定できるシステムの実装。

第4章　プログラミングにおける評価の補助システムの提案

本章まで論じた現状や問題、対策を踏まえた上で、プログラミング教育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。システムの名前は leafR(Less offort and full Results) とし、今後本システムを leafR で表記する。

4.1　問題の整理

第 3 章であがった問題では、学生側に苦手意識があると論じた。これらの原因はそもそもプログラミングを行うことに対して意義を感じていない意見があったことも考えられる (図 4.1)。学ぶ意味を理解していないことによって向上意識も起きず、また意欲も出ない。加えて、プログラムのエラー文は解読が難しく、種類も多い。それに対して検索エンジンを使用して調べている学生は見えなかった。これもまたプログラミングへの苦手意識へと繋がる。

教員側の問題としては簡易なプログラムを実行する際の学生の手間を踏まえた上で、決められた時間内に説明を終える必要性がある点と、評価である。仮に時間内に理解できない学生がいた場合はやむなく次の説明へと移るが、学生は理解していないために講義自体についていくことができなくなる。さらに、教員の確認方法、評価方法で変わるものの、課題の評価では学生分のプログラムを目視し更に実行まで行った場合の手間に苦しんでいる。

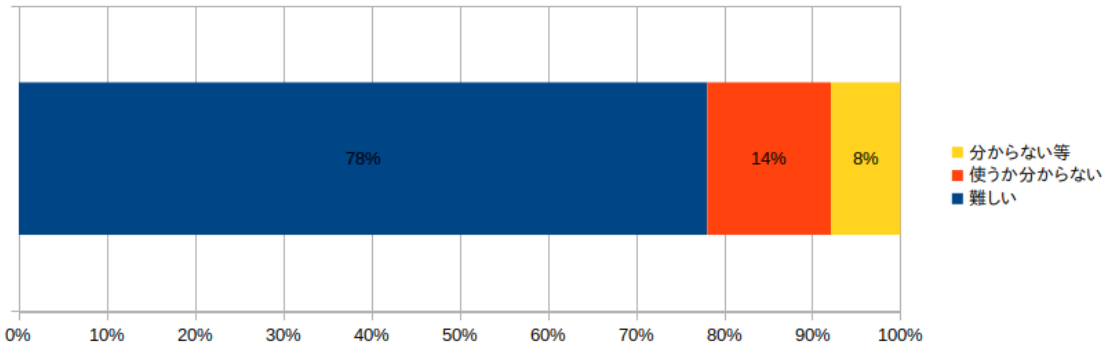

図 4.1: 自由記載欄の回答

4.2　サイクル

プログラミング教育では、一連のサイクルが発生している。学生側と教員側のサイクルをもとに説明する。

4.2.1　学生側

学生側のサイクルでは説明を聞く、簡易なプログラムを実行する、課題に取り組み結果をメールで送信する、の 3 つの工程に分けられる。流れは図 4.2 のとおりである。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。これを実行する際の手順もまた一連のサイクルとなっているが、3 つの画面内の表示を見る必要があり、効率的ではない。

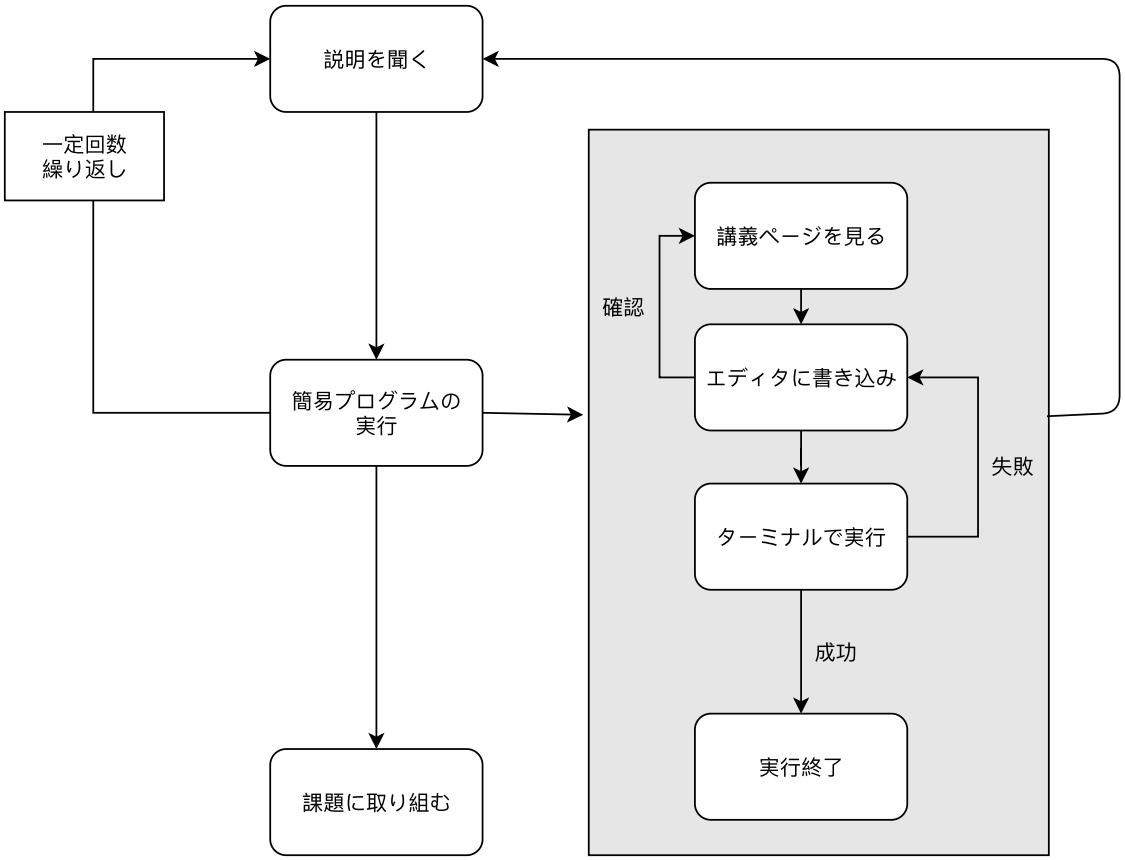

図 4.2: 学生側のサイクル

4.2.2 教員側

教員側のサイクルでは、メールが届き開く、目で見て実行可能かの判断、評価が大筋となる。流れは図 4.3 のとおりである。その際に、目で見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、メールをいちいち開く、見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。

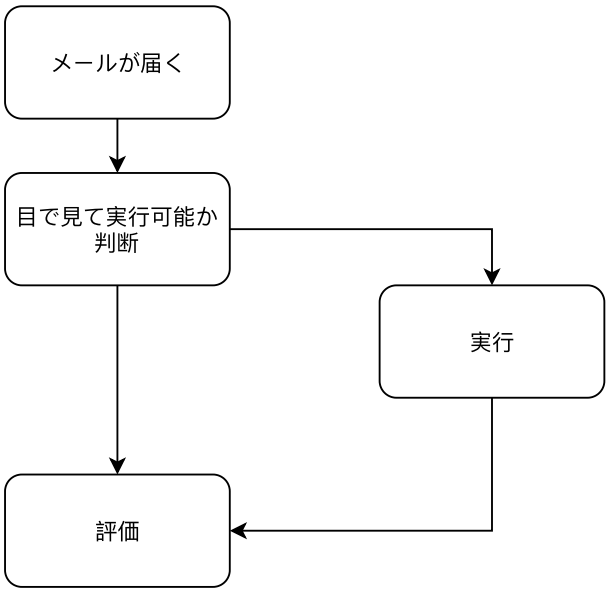

図 4.3: 教員側のサイクル

4.3 サイクルの分割と方策の提案

これらのサイクルをコンピュータの機能で実行した方がよい場合と、人の手で実行した方がよい場合で分割して考える。それぞれ、コンピュータの機能で補う部分と、人の手で実行する部分をにより簡易にする表示の方法を示す。

4.3.1 学生側

学生側のサイクルを分割して考える。また、それを改善するための手法を示す。

方策 1 簡易プログラムの実行の際の表示画面を 1 つにまとめる

コンピュータの機能でより簡易に実行する。学生側のサイクルにある簡易なプログラムの実行に関しては、3 つの画面内の表示をいちいち見るといった動作は、その 3 つの表示を一つの画面内に展開することで解決できる。

方策 2 課題の送信にメールを用いない

コンピュータの機能でより簡易実行する。課題の送信はメールソフトを起動せず、プログラムが実行した時点で教員側から目視できる形で情報を保存することで、学生側の動作を省き単純化できる。

方策3 意識の向上を補う説明の表示

人の手で実行する際により簡易にする表示である。4.1にあった問題を解決するためには、プログラミングの意義の説明やエラー文への説明を随時行う必要がある。しかし、プログラムの説明に加えると講義の時間がさらに圧迫されることは明確である。そのため、コンピュータ画面に表示するといった形で視覚的に伝える。また、エラー文は手間をかけずに理解する、といった動きを作る。

4.3.2　教員側

教員側のサイクルを分割して考える。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が前提

コンピュータの機能でより簡易に実行する。教員側の評価のサイクルでは、目で見て実行可能か判断する、という工程が作業が増える起点になる。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という前提であれば、プログラムの動作をいちいち確認する必要はなくなる。

方策2 個人の評価をより簡単にする表示

人の手で実行する際により簡易にする表示である。アイデア1の前提を踏まえた上で、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。これによっていちいち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である leafR 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に表示する。

要件 2 エラーの場合は、エラーの情報を提示する。

要件 3 成功するまで入力できる。

要件 4 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される。

要件 5 ユーザ名が入力されていない場合は情報を保存しない。

要件 6 教員が情報を利用し、採点が出来る。

要件 7 脆弱性について可能な限り対処する。

5.2　システムの設計

システムを設計する。一連の流れは図 5.2 のとおりである。図 5.2 にあるエディタページと実行ページの関係性は図?? である。本システムは、課題となる問題に答え、成功した場合にデータベースへと格納する課題システムと、講義中の簡易なプログラムを実行する講義システムの 2 つに分けられる。以下の機能をつける。

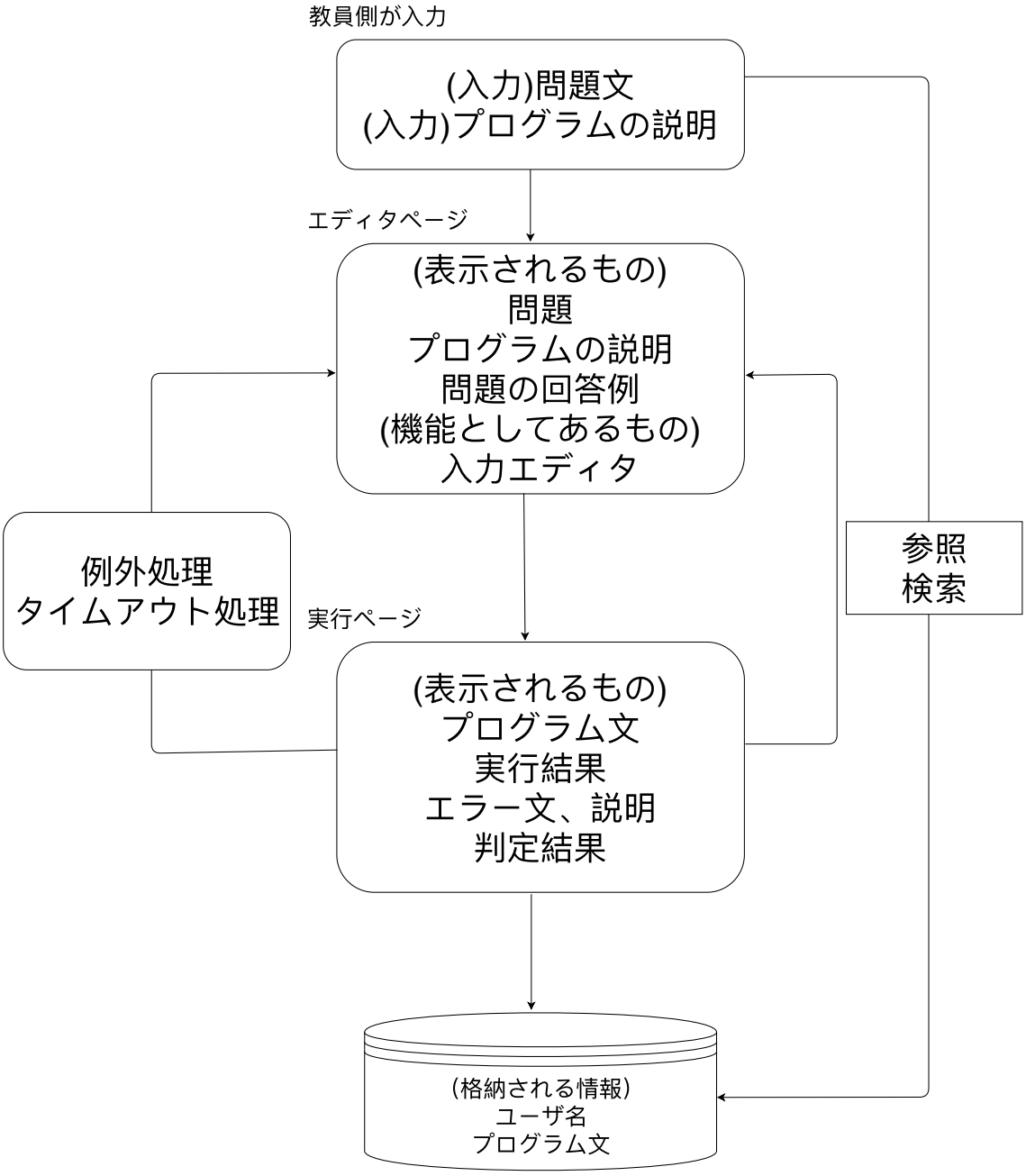

図 5.1: システムの流れ

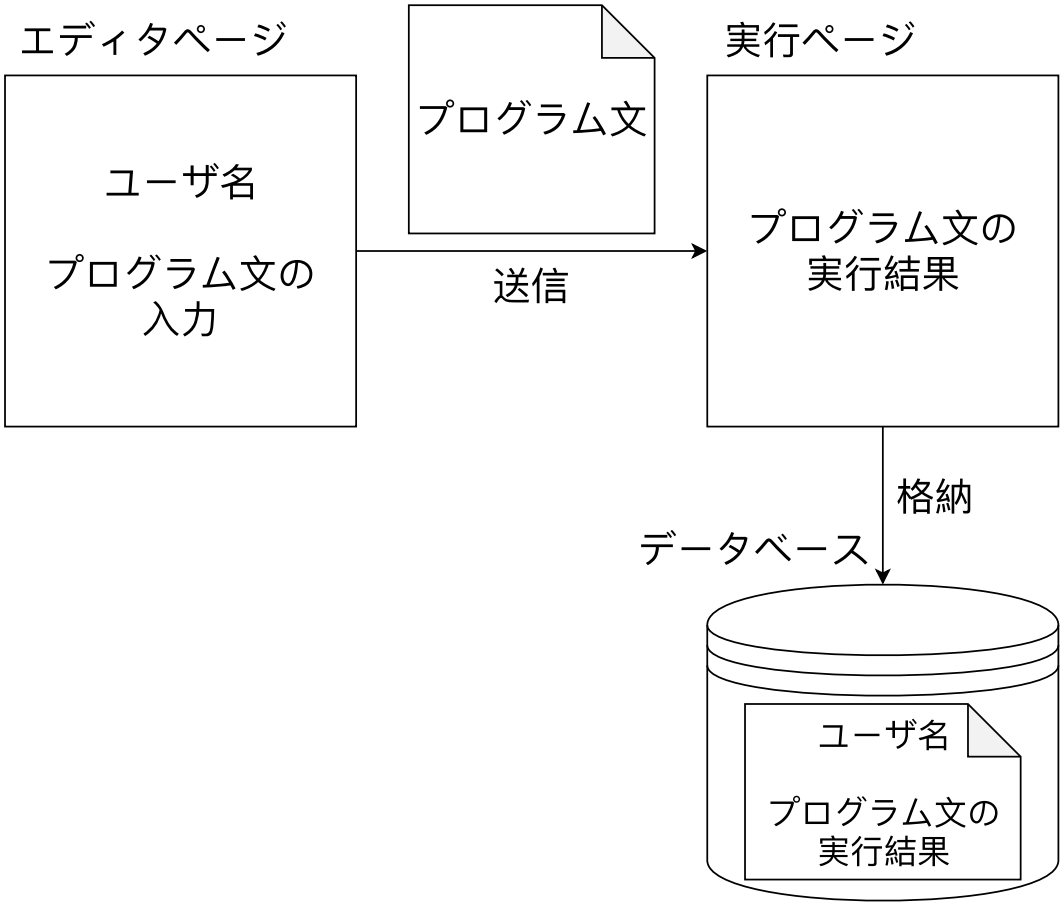

図 5.2: 各ページの関係性

5.2.1 エディタページ

課題システム、講義システムどちらにも実装する。このページでは以下の機能が含まれる。これらの機能で表示する内容は教員が入力できる仕組みになる。

- メソッドと想定の説明

  問題を解く前にどういった場面で使うかの想定の説明をエディタページに表示することにより、どんな場面で使うかわからない、といった疑問に対して答えることを想定している。

- 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見ながらプログラムを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。

- 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、逆にこれが出なければ成功していない、と伝えるために表示する。

5.2.2　実行ページ

課題システム、講義システムどちらにも実装する。このページでは以下の機能が含まれる。

- エラー文の表示、説明

  アンケート結果にもある通り、エラー文が読み解けずに挫折する学生が多いため、出力頻度の高いエラー文には対策を講じた文章を表示する。その他、環境によってエラー文が英語で表示される場合がある。慣れていない文に対して苦手意識が多く見られるため、対策の説明は日本語で行う。

- データベースへの格納

  講義システムには実装しない。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。格納の形態は図 5.3 のとおりである。

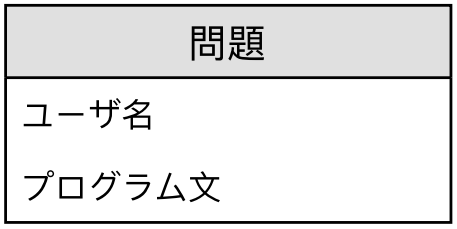

図 5.3: データベースの格納

5.2.3　個人評価確認ページ

課題システムに実装する。このページでは以下の機能が含まれる。

- データベースに格納された情報の表示

  実行ページにある機能であるデータベースの格納によって保存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。

- データベースの情報の検索

  データベースには人数を問わず、実行に成功したユーザ名とプログラム文を格納する。その際、母数が多くなるほど個人の学生を見つけるのは困難になる。それを防ぐため、検索欄を設けることで一個人がどういったプログラム文を入力したのかひと目で分かるページの表示を行う。

第6章　システムの開発

本章では、以上の機能を含めたleafRの開発を行う。

6.1　開発環境

開発環境は以下の通りである。

- Ruby

  Rubyとは、1995年にまつもとゆきひろによって開発されたオープンソースの動的なプログラミング言語である。本システムはruby 2.5.1p57を使用した。 [14]

- SQLite3

  SQLiteとは、軽量で速く、高い信頼性を持つSQLデータベースエンジンを実装するC言語ライブラリである。本研究でのバージョンは、SQLite3.22.0である[16]。

6.2　システムの作成

実装した部分を以下に示す。また、その際にどういったことを目的としているかを述べる。

6.2.1　エディタページ

学生側が最初に見るページである。このページでは、プログラムを作成する前に行うべき説明や、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いことを目的としたページにした。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。実際に作成した画面が図6.1である。

図 6.1: エディタページ

実装部分 1 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。本学では学生 1 人につき学籍番号が割り振られており、基礎プログラミングなどではそれを入力する。これによってプログラムの情報を格納した際に教員側は個人が作ったプログラムを検索することが容易になり、評価の補助に繋がる。

実装部分 2 入力エディタ

プログラム文を入力するテキストエリアを作成した。エディタの役目を担う。プログラム文を入力して送信ボタンを押すと、実行ページで評価を行う。送る文字列には CGI ライブラリの機能の一つである CGI.escapeHTML() を使用し、特定の意味を持つ文字を無効化するためのエンコード処理を行っている。

実装部分 3 問題文・出力結果例・メソッドや想定の説明の表示

問題文には講義で取り組む簡易プログラムの問題文や課題文を挿入する。出力結果例は、教員が用意したプログラムを実際に動かした場合の結果を表示している。学生にあらかじめ答えを認識してもらうことでプログラム作成のイメージを与える。メソッドや想定の説明では、今から作るプログラムをどういった場所に使うか、また、どのようなプログラム文で作ると良いか、などヒントにもなり得る文章を挿入する。入力

は教員が行えるように、問題文を question 変数、出力結果を出すためのプログラム文を code 変数、メソッドや想定の説明を method 変数として扱っており、より簡易プログラムの作成や課題の作成といった様々な場面に適応することができる。ソースは以下のとおりである。

```
#question は問題文
question = "ここが問題文"

#code は問題の出力結果例
code = "i = 1
        while i <= 10
        puts i
        i += 1
        end
"

#method はメソッドと想定の説明の変数
method = "ここがメソッドの説明の変数"
```

6.2.2　実行ページ

エディタページにプログラムを入力し、その結果が出力されるページである。ここでは、学生が入力したプログラムに対しての評価や訂正箇所を伝える表示を主に行う。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。実際に作成した画面が図 6.2 である。

図 6.2: 実行ページ

実装部分 4 プログラムのコードを表示

自分が打ったプログラムを再度表示することで、どういった問題に取り組んでいるかの認識や、下記に説明しているエラー文は何行目が問題なのか、といった点を探しやすくすることができる。

実装部分 5 実行結果の表示

ターミナルエミュレータと似た表示を行うために背景は黒に文字が白である。これはターミナルエミュレータの役目を担う。この表示によりひと目でプログラムが動いているかどうかを判断できる。

実装部分 6 判定結果の表示

作成していたプログラムが実行され、成功していた場合と、失敗していた場合で表示が異なる。成功した際にはそのままデータベースへとユーザ名とプログラム文が格納された旨が表示される。成功していた場合は図 6.3 である。

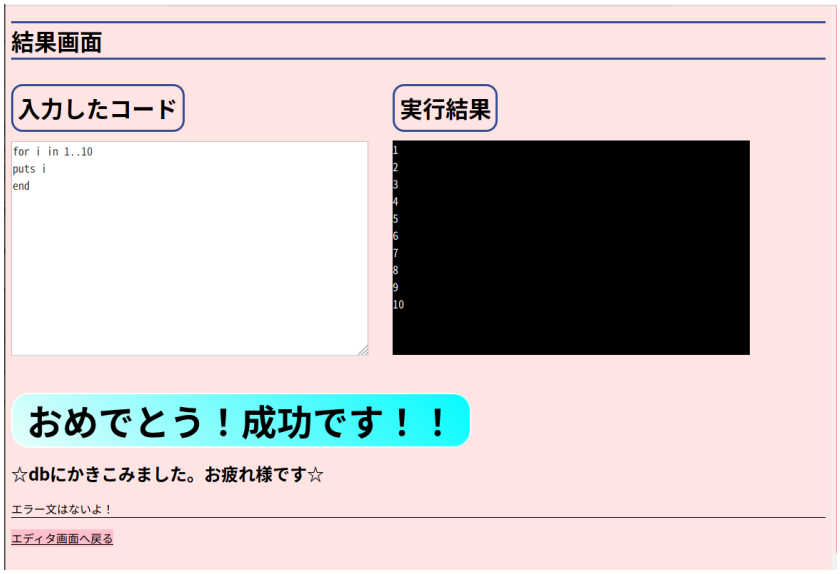

図 6.3: プログラムの実行に成功した場合の表示

実装部分 7 エラー文の表示

実行に失敗していた場合は図 6.5 である。出現する頻度の高いエラー文に対して、出現した際にはどういったことが原因で起きているかの説明の文章を表示した。また、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。例えば、指定した変数を間違えている NameError のエラーが出た場合には、NameError ruby で検索する URL が作成される。また、エラー文のみではプログラミング以外の他の情報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” を文字列に追加した。

図 6.4: プログラムの実行に失敗した場合の表示

図 6.5: エラー文の検索画面

6.2.3 個人評価確認ページ

データベースの中身を見るページである。教員側のみが閲覧可能であり、個人の評価を容易に行うことを可能にするため、ユーザ名でこれまで登録されたプログラムの情報を検索することも出来る。機能は以下にあげたものを実装している。実際に作成した画面が図 6.6 である。

図 6.6: 個人評価確認ページ

実装部分 8 データベース内の情報の表示

エディタ画面で入力されたユーザ名とプログラム文を表示する。左側に全ての情報を掲示している。同じユーザ名が重複した場合でも上書き保存はされない。ここに表示されている時点でプログラムの実行に成功しているため、ユーザ名を探せば評価を行うことが可能になる。

実装部分 9 データベース内の情報の検索

データベースの情報量が多くなるほど探しにくくなる。そのため、ユーザ名で検索を行う欄を設けた。実際にユーザ名を入力した際の検索画面は図 6.7 になる。これにより、個人単位の評価も行いやすくなる。

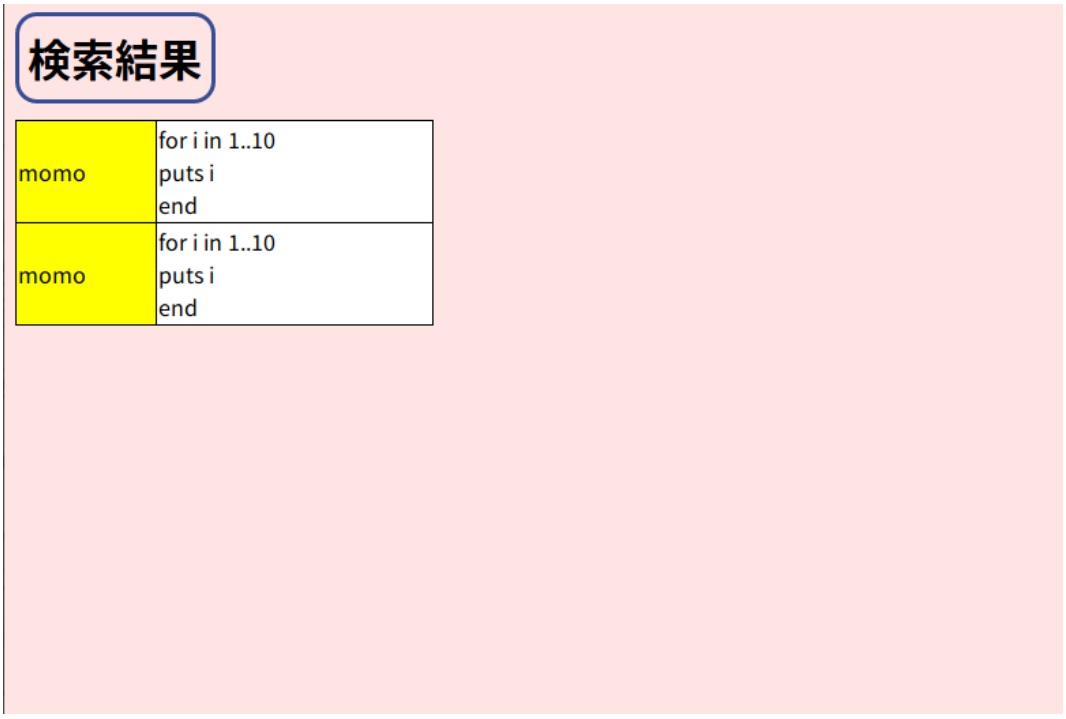

図 6.7: データベース検索結果

6.3　評価

実装し、要件と照らし合わせて評価を行う。

6.3.1　実装評価

エディタページ、実行ページ、個人評価確認ページにそれぞれ提案した方策を元にシステムを実装した。その結果、第 5.1 節で述べた要件のうち、1 から 6 までを満たすことができた。要件 7 にある脆弱性は現時点では対策を施している部分は一部分である。

6.3.2　動作評価

講義内では 4 回、約 10 分程度の時間を使って本システムを学生 40 人程度を対象に運用した。使用した環境は本学の情報教室である。Web ブラウザはバージョンを使用し、実行ページで成功した表示が出た場合は、本学の SNS システムにプログラムを投稿するといった形式である。問題文はそれぞれ以下のとおりである。

1 回目　1 から 1000 の数を表示しなさい。

2 回目　1 から 1000 の数のうち、偶数のみ表示しなさい。

3 回目　1 から 1000 の数のうち、3 の倍数のみ「ばか」と表示しなさい。

4 回目 1 から 1000 の数のうち、3 の倍数のみ「わん」と表示しなさい。

1 回目はエディタに入力したプログラムを判定するのみのシステムを、2 回目、3 回目にはプログラムが成功した場合にデータベースに格納するシステムを、4 回目ではエディタページにヒントを表示して運用した。4 回目の時点で、成功する画面と失敗した画面の表示、データベースへの格納を行うことが確認できた。

一方で、プログラムが実行されていない (失敗している) にも関わらずエラーが発生しないプログラム文に対しては、学生側へのサポートを実行ページのみでは行えなかった。加えて、今回の運用では学生側にではユーザ名を学籍番号で固定することを伝えずに使用したため、ユーザ名検索が意味をなさなかった。

システム上の問題としては、40 人が一斉に使うことで、サーバー過負荷によりシステム自体がダウンしてしまう可能性を発見した。

第7章　考察

本章では、第 6 章と第 6.3 節の結果を踏まえて考察する。システム自体は動作を確認できたことで、方策を実現したシステムを実装することに成功したと言える。また、第 6.3.2 節で得られた結果により、実際の講義でも利用可能なことが分かった。その際に起こった問題について深く考察していく。

問題 1 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。今回の運用では制御構造の while や for、upto を用いて問題に答える内容だったが、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 7.1 を参照)。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。

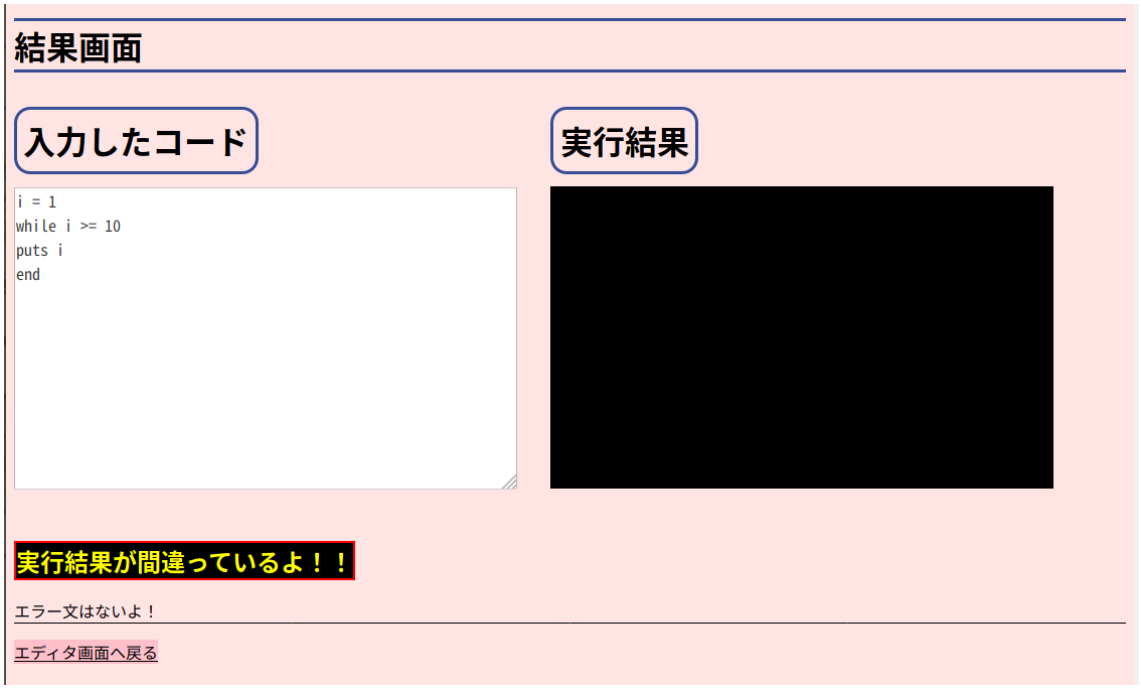

図 7.1: エラー文がでない場合

問題 2 ユーザ名検索を使用できない場合

使用する学生側にユーザ名の形式を伝えない場合、本人の名前以外にも学籍番号やオリジナルの仮名を使用する可能性がある。基礎プログラミングで運用する際には学籍番号をユーザ名として想定するが、実際に運用した際に本人のローマ字表記などで登録された場合、検索が困難になる。運用の際には口頭での説明や画面上に表示、さらに受け取る文字列に対して何らかの正規表現での制限を行うべきだと考えた。

また、ユーザ名の問題としては代筆も浮上した。本人以外が別の誰かの学籍番号を入力し、プログラムの作成に成功した場合はデータベース上では本人がやっていなくても成功した、という結果のみが格納されてしまう。これを防ぐためには、本学の個人に割り当てられている専用のメールアドレスの使用を考えた。本学のメールアドレスは**学籍番号@学年ごとの一文字.koeki-u.ac.jp** の形式である。教員側にデータベースに格納された旨の通知を送る際、実行した学生本人のメールアドレスから送ることで、そのメールアドレスの学籍番号の部分と登録されたデータベース上の学籍番号が異なった場合には、個人評価確認ページで特別な表示を行う、といった対策を考えている。

問題 3 サーバー過負荷の問題

本学の基礎プログラミングでは、一つの講義につき平均 40 人が受講している。今回は 1 つのクラスで 10 分間程度で運用したが、システムの動作が極端に重くなる場面があったりと、使用する際に課題となる部分が出た。これを 105 分の講義で使用する際には、さらに動作が遅くなることが想定される。後に記述する問題にもあるが、DoS 攻撃と同一の状態が発生しているため、対策が難しい。

第8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。

8.1　結論

本稿では、プログラミング教育での問題を明らかに、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、プログラミングにおける評価の補助システムである leafR の提案を行った。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。プログラムを入力するテキストエディタや、プログラムが実行された結果に対しての表示を行うことができた。また、実際の講義で運用した際には、実装したシステムが目的に沿って動作や表示することを確認した。しかし、運用していく際の問題点も明らかになった。システムの例外や問題を明らかにするために、さらに運用する機会を増やす必要がある。

8.2　課題

結論と第 7 章を踏まえて、今後の課題を明らかにした。

8.2.1　プログラミング教育の補助のためのシステム

プログラミングを学ぶために、必要だと思われる対象について具体的に探す必要がある。今回はシステムの機能を中心に実装、確認のための運用を行っているため、今後は使用する側である学生や教員の意見を聞きつつ、教育のために伝えやすい視覚的効果を探す必要がある。このためには、アンケートを行うことが有効であると考える。

8.2.2　脆弱性の問題

脆弱性による問題がある。運用前にテストを行って問題を見つけ、対策を講じる必要がある。また、プログラムだけではなく運用する環境に応じて対策の方針を変える。今回の運用はローカル環境のみの使用を想定しているため、最低限の対策のみを行っていく。以下にこれまでに発生した問題の説明を示す。

- インジェクション攻撃

  悪意のあるユーザーが、意味を持つ不正なプログラム文を投入することで、通常はアクセスできないデータにアクセスしたり更新する攻撃である。本研究ではインジェクション攻撃によって、HTML のタグが実行できた。実際に実行できたインジェクション攻撃の例を以下に示す。

  ```
  print("<h1>やっほーーーーーー</h1>")
  ```
  やっほーーーーーの文字列が HTML の<h1>タグとして判別されるため、
  大きく表示される。

- シェル実行

  Ruby には、‘(バッククォート)、system、ecex、open、といったシェル実行の文字がある。これによって、シェルスクリプトを実行し、本来の動きとは異なった表示を見せることが出来る。これは Ruby 言語で実行するからこその問題ということもあり、対策が難しい。実際に実行できた文字列を以下に示す。

  ```
  puts ‘ls‘
  ```
  ディレクトリに入っているファイル一覧を参照できる。

  ```
  ’’; ls # ’
  ```
  上の実行結果と同様。

- DoS 攻撃

  サーバーなどのネットワーク機器に大量のパケットを送るなどしてサービスの提供を不能にする攻撃である。運用の際に、多くの人数からのアクセスが起こることによってシステムの動作が遅くなることが発見できたため、この問題が起こる可能性がある。

8.3　今後の展望

提案した方策の実装までを具体的に行ったため、追加するシステムは実際に運用した上で見つけていく必要がある。そのため、実際に授業などで使うことで生徒側の問題点を更に発見する。また、教員側にも実際に使用してもらい、これまでの採点と比べて本システムを使うことによりどの部分が改善されたか、またどういったシステムがあることでより採点や個人への評価が簡易になるかといった点を明らかにした上でシステムの拡張を図る。本研究の段階では外部からアクセスできないローカル環境以外で使うことを想定していないため、ローカル環境以外で使う際にどういった問題が発生するか、それについての対策を講じながらより実用性の高いシステムにする。

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] ReseMom. “小学校のプログラミング教育、先生の 98 %が「授業の実施に不安」”. https://resemom.jp/article/2019/04/26/50334.html. (参照 2019-11-18).

[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 2019-12-3).

[15] Perl UNIX/Linux windows. “CGI の基礎知識 “. http://www.tryhp.net/first.htm. (参照 2019-12-3).

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

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 (　　)