オンラインプログラミング学習環境Bit Arrowの研究と開発

A Study and Development on Bit Arrow: An Online Programming Learning Environment

2022年3月　博士学位論文

東京農工大学 大学院 工学府 電子情報工学専攻

長島　和平

Kazuhei Nagashima

目 次

図 目 次

表 目 次

リスト目次

第1章　緒言

本章では研究の背景と目的について述べる．

1.1　研究の背景

2022年度から適用される新学習指導要領では，高等学校の必履修科目である教科「情報I」の中にプログラミングが含まれることとなった．しかし，これまで高等学校ではプログラミングを扱う科目「情報の科学」は各年次で10%程度しか開講されていなかった [1]．新学習指導要領の適用と同時に，多くの高等学校でこれまで扱ってこなかったプログラミングの授業が行われることになる．さらに,「情報I」においては，プログラミングを使用した活動を通じて,「アルゴリズム」「コンピュータの仕組み」「情報システム」「モデル化とシミュレーション」「オープンデータの分析」「センサデータの収集・分析」などを学習することが求められており，生徒や教員が使う実習環境にはこれらの活動を統合的にサポートすることが必要である．

一方，公立の高等学校などの学校教育では，実習用のPCは自治体によって導入され，教員はコンピュータの管理者権限を持っていなかったり，ソフトウェアの導入が制限されたりしていることが多い．そのため，プログラミングの学習のために特別なインストール作業が不要であるという理由でプログラミングの実習環境が選ばれてしまうことがある．実際，高等学校の教科「情報の科学」で扱われている現行の教科書には「Webブラウザを用いたJavaScriptの実習」や,「表計算ソフトを用いたBASICの実習」などが掲載されている [2],[3]．JavaScriptはOSに付属しているテキストエディタアプリでプログラムが記述でき，それをWebブラウザで実行できる．BASICも，多くのPC環境に標準でインストールされている表計算ソフトで編集と実行が行える．このように，教科書に掲載されている環境は多くのPCでインストールなどの準備をせずに使用できるが，プログラミングを学習するために開発された環境ではないため，生徒が利用するには適していない．

例えば，一般的なWebブラウザでは，JavaScriptの実行はできるが，そのコードを作成・編集するためにテキストエディタを用いなければならないため，Webブラウザだけで作業が完結しない．テキストエディタとWebブラウザの両方でファイルを開くような煩雑な操作が必要になる．また，多くの高校で導入されているWindows環境で標準でインストールされているテキストエディタ（メモ帳）は，プログラミングのための入力支援（インデント付け，対応する括弧の自動挿入など）をもっていない．さらに，エラーが発生したとき，学習者がエラーを直すためには，エラーメッセージや，エラーがプログラム中のどこで起きているかを学習者に通知する必要がある．しかし，Webブラウザのエラーメッセージは開発者ツールなどから見

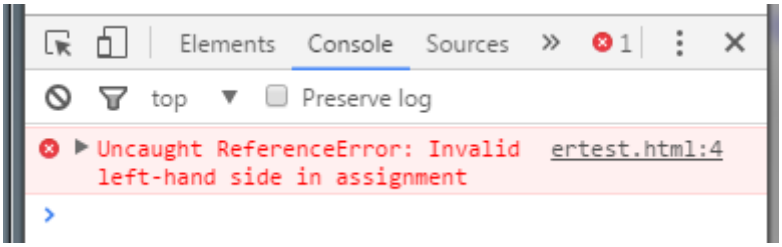

図 1.1.1 Web ブラウザ (Google Chrome) のエラーメッセージ

なければならず，エラーが発生しても自動的にエラーメッセージが表示されるわけではないため学習者がエラーの原因を探すことは容易ではなく，そのメッセージも初学者にとって分かりにくい（図 1.1.1）．Web ブラウザを用いて JavaScript のプログラミング授業を行った高校教員によると，エラーが発生したときに，エラーメッセージが表示されない現象があったため，学習者は何をすればよいのかが分からずに作業が止まってしまい，それらのエラーへの対応に追われ授業を成立させることが難しかったという問題がある．

さらに，エラーの修正やプログラムの書き方，概念の理解などに苦労し，学習者が教員やティーチングアシスタントなど（以降「教授者」と総称）から何らかのサポートを必要とする状況（つまずき）に陥ったとき，教授者は適切に指導して学習者をサポートする必要がある．しかし，高等学校では 40 人程度の教室に教授者が 1 人か 2 人であることが多い．大学においても大きい演習室に教授者が 1 人と，すでにその講義を履修した学生などのティーチングアシスタントが数名配備されているだけであることが多く，すべての学習者について状況を把握することが難しい．学習者はそれぞれのコンピュータのローカル環境で作業をしているので，学習者が順調に進められているのか，エラーで困っているのかといった情報は机間巡回をして一人ずつチェックしなければ知ることができない．

また，指導要領が求めている活動内容には，ネットワークに接続してデータを取得し，実際のデータに基づいた分析を行ったり，身の回りの事象を計測して実世界のデータを集めたりするなどの活動が含まれており，それらの活動の中でもプログラムを使ってデータを分析することが望ましい．このような状況では，データを入手・共有・処理する作業の途中でも問題が発生しうる．例えば，分析を行うプログラムが正しく書かれていたとしても，入手したデータを適切な場所に適切な名前で保存することができていなければ，正しい実行結果が得られなくなってしまう．

1.2 本論文の目的

そこで，本研究では，演習環境に自由にソフトウェアをインストールできない，受講者のほとんどがプログラミングの初学者であり，プログラミングへのサポートが必要である，そのような受講者が同時に多数受講していて，状況が把握できないような教室環境でも，高校の「情報」科目で必要とされる学習内容を必要十分に学習可能なプログラミング環境の構築を目指す．

具体的には，演習環境をインストールが不要な Web アプリケーションとして提供し，プロ

グラミング初心者にとってつまずきの原因となるエラーの発生を抑え，エラーが発生したとしても素早く修正ができるようにする．また，つまずきに陥っている学習者を教授者が容易に把握できるようにする．さらに，演習環境にはプログラミングを行う環境に加えて,「情報」科目で必要とされているデータの収集・整理・分析を統合的にサポートする．

1.3 本論文の構成

本論文は，特に初学者へのプログラミング教育を目的に開発されたプログラミング学習環境に関する既存研究を調査し，初学者のプログラミング学習を支援するために必要とされる機能や要件，課題について明らかにしたうえで，それらの要件を満たす新しい学習環境である Bit Arrow を提案する．

2 章では，高校の教科「情報」の指導要領の調査を通じて，必要となる学習プラットフォームについての要件を整理する．3 章では，プログラミング教育を支援するための先行研究を，学習者に対する支援と教授者に対する支援に分類し，既存研究が行ってきた支援の手法をまとめ，プログラミング学習環境に必要な特徴を整理する．2 章と 3 章から，Web ベースのプログラミング学習環境への要求として「高等学校の学習内容や目的に応じて適したプログラミング言語を利用できる.」「高等学校で行われるデータの収集・整理・分析という学習を統合的に支援する.」「学習者自身によるエラー修正を支援する.」「教員による学習者の状況把握を支援する.」が必要であることが分かった．

4 章では，3 章までに明らかになった課題を受け，本研究の目標として,「エラー発生防止・学習者による自力でのエラー修正」「高校の学習指導要領に対応した活動の支援」「演習内容に依らない，つまずいている学習者の効率的な発見」を挙げる．

5 章では，この要求を踏まえ,「Web ブラウザで実行できる.」「複数のプログラミング言語に対応する.」「エラーを減らすための入力支援やエラーメッセージの表示を行う.」「IoT 機器や外部データベースとの連携や，データ分析ライブラリを提供する.」「学習者のログを収集し，教員に提供する.」という機能を持ったプログラミング学習環境 Bit Arrow の設計を述べる．6 章では，5 章の設計に沿った Bit Arrow の実装について述べる．

7 章では，Bit Arrow を高等学校の教科「情報」で利用することを想定し，文部科学省から公開されている資料や現行の教科書の内容に対して，Bit Arrow で実装することができるソースコードや仕組みといった実例を調査する．

8 章では，学習者に対する支援機能を検証するため，Bit Arrow を高等学校で行われた実践授業で利用した結果を示す．この授業における学習者の学習履歴を調査し，これまでの教科書で使われていたテキストエディタとブラウザを用いた環境との比較を行う．また，教授者への支援として提供しているログを用いた学習者の状況把握について，授業で運用されている実例を示す．9 章では，本研究の成果と，今後の課題をまとめる．

第2章　高校教科「情報」に関する調査

本研究は，高校教科「情報」で習得すべき技能を統合的に学べるWeb上のプログラミングプラットフォームの構築を目指している．

ここでは，まず高校においてプログラミングを学ぶ理由としてたびたび議論されている「プログラミング的思考」という能力について整理を行った後，高校教科「情報」（以下「教科情報」）が定めている，生徒が学ぶべき内容，および2022年度の施行に備え教員向けに作成された研修教材について見ていく．また，日本以外の国での「プログラミング的思考」の扱いや，情報系科目のカリキュラムの比較を行った上で，最後にプラットフォームが備えるべき要件についてまとめる．

2.1　プログラミング的思考とComputational Thinking

プログラミング的思考は，文部科学省の教育課程部会において次のように定義されている．

> 自分が意図する一連の活動を実現するために、どのような動きの組合せが必要であり、一つ一つの動きに対応した記号を、どのように組み合わせたらいいのか、記号の組合せをどのように改善していけば、より意図した活動に近づくのか、といったことを論理的に考えていく力

なお，この定義は「小学校段階におけるプログラミング教育の在り方について」[4]という資料の中で行われていたものであるが，プログラミング的思考は小学校段階に限ったものではなく，高校の指導要領解説にも次のように取り上げられている．

> プログラミングを，自分の意図した活動に活用し（中略）将来どのような職業に就くとしても，時代を超えて普遍的に求められる「プログラミング的思考」などを育むプログラミング教育の実施を（中略）発達の段階に応じて位置づけ（中略）小・中・高等学校を見通した学びの過程の中で,「主体的・対話的で深い学び」の実現に資するプログラミング教育とする．

この定義によれば，プログラミング的思考は実際にプログラミングを行うこととは独立に，あらゆる局面で必要になるものであり，教育の現場では，すべての教科においてプログラミング的思考を身に着けることが要求されている．[5]

一方，プログラミング的思考について高校の段階で何を身に着けさせるべきか，という議論は国内ではあまり行われていない．

中島 [6] は,「Scratch や Google Blockly のようなビジュアル型（ブロック型）プログラミング言語は，初心者にとってプログラミング的思考を学修するために適切な言語である」としながらも，ブロック型言語は「ブラックボックス化」された処理系であり，コンピュータの仕組みを知るためには限界があるとしている．内田ら [7] は，中島の論文を引用しつつ，高校の段階になっても，小学生向けとされているブロック型言語が使用されている場合があり，ブロック型言語からテキスト型言語への移行が円滑に進んでいない点を問題視している．しかしながら，これらの議論を通じてもブロック型言語で学べるものと，テキスト型言語で学べるもの（ひいては，初等教育と中等教育で学ぶべきもの）の違いについては明確ではない．

太田 [8] は高校段階におけるプログラミング教育にあたり，プログラミングを，従来の「アルゴリズムを考えて，それを実現するために個々の命令を（トップダウン的に）使う」ことから「すでに知識としてあるプログラミング部品を組み合わせて問題を解決するボトムアップ的な認知過程であると再定義」することを提案している．その上で，プログラミング的思考については「単なるアルゴリズムの学習とプログラミング的思考の根本的な違いが現状では明確に論議・定義されていない」としている．

一方で，海外に目を向けると,「プログラミング的思考」のもととなった概念として「Computational Thinking」が提案されている．Computational Thinking は，Wing[9] によって提案された概念であり，明確な定義として書かれてはいないが，次のような性質をもつものとしている (訳文 [10] より引用)

- コンピュータ科学者だけではなく，すべての人にとって基本的な技術．
- 問題解決，システムのデザイン，そして基本的なコンピュータ科学の概念に基づく人間の理解などを必要とする．
- 再帰的に考えること．並列処理であり，命令をデータとし，データを命令とすること．
- 巨大で複雑なタスクに挑戦したり，巨大で複雑なシステムをデザインしたりするときに，抽象化と分割統治を用いること．
- 予防，防御，そして最悪のシナリオからの復帰という観点を持ち，そのために冗長性，故障封じ込め，誤り訂正などを用いること．
- ヒューリスティックな推論により解を発見すること．

「プログラミング的思考」の定義に比べると，コンピュータを職業として携わる人以外にも必要な素養であるという共通点をもちながら,「単なるアルゴリズムの学習」の範疇を超えて，システムの設計や複雑なデータ構造など，情報科学の広い分野に立ち入った定義をしようとしていることが読み取れる.「プログラミング的思考」を高校での活動に拡張した場合，多少なりともこのような概念を取り入れる必要があると考えられる．

Computational Thinking を高校の授業に取り入れた実践 [11] においては，学習目標に次のようなものを掲げている．

- パターンと規則性を認識し、問題をより小さなパーツに分解し，単純化された問題を定式化して解決し，解決策を一般化・カプセル化する．
- 複雑な式の代数的操作を体験する．
- 数学関数を使って，図やアニメーションなどの成果物をモデル化することができる．
- 多数のデータアイテムを単一のエンティティであるかのように変換する．
- 階層的にデータを整理することができる．
- 税金や割引などのレートを使用して，合計，平均，数量を計算する．
- ランダムサンプリングを使用して，関係性の事例を調べたり 一般的なケースを見つけることができる。

その上で，これらの考え方はプログラミングに限らず適用可能であり，Computational Thinking はプログラミングの体得が目的ではない，それにもかかわらず，プログラミングを行うことなくこれらの概念を体験することは困難である，としている．

また，Computational Thinking を高校の情報系以外の科目にも適用した実践も報告されている [12]．例えば，国語（英語）では文学に対してテキスト解析の手法を取り入れ，歴史においてはノートを取る際の情報の整理の仕方に着目させ，外国語（ラテン語）では構文定義・文の図式化を実施，美術では 3D プリンタを活用して造形を行う，といった具合である．そして，情報系の授業においては，Logo などの教育用言語を用いてインタラクティブなゲーム制作を通してストーリーを組み立てる，HTML や CSS，Python などの「実際の言語 (原文では”real language”)」を使った問題解決の手法を学ぶ，としている．

以上をまとめると，Computational Thinking の学習について次のような方針で実践を行っているといえる．

- 情報系以外の科目では，プログラミングを用いずに Computational Thinking を学習する．
- 情報系の科目では，教育用言語,「実際の言語」を用いたプログラミングを交えた活動で Computational Thinking を学習する．

裏を返せば，高校段階における Computational Thinking（日本では「プログラミング的思考」）の学習については，科目ごとに次のような方針で行うべきと考えられる．

- プログラミングを用いなくても Computational Thinking は学習できるが，それらの活動は情報系以外の科目で行うべきである．
- 情報系の科目では，プログラミングで Computational Thinking を学習するべきである．プログラミング言語は教育用言語だけでなく「実際の言語」も併用すべきである．

2.2 高校教科「情報」で学ぶ内容

教科「情報」は，必修科目の情報Iと選択科目の情報IIに分けられている．本研究では，情報Iで学習する内容について詳しく見ていく．

情報Iにおいては，目標に,「効果的なコミュニケーションの実現，コンピュータやデータの活用について理解を深め技能を習得するとともに，情報社会と人との関わりについて理解を深めるようにする」と挙げられている [13]．これは，次のような事柄について「理解し，技能を身に付ける」ことを目標としている [14].

- コンピュータを活用するために必要な情報が処理される仕組み
- データを活用するために必要な収集，整理，分析の方法
- プログラム
- モデル化とシミュレーション
- ネットワーク
- データベース

ここで，目標としているものには,「プログラム」の理解・習得だけではなく，データの「収集，整理，分析の方法」の理解・習得も含まれている点に着目したい．つまり，教員や生徒が活動を行うために必要な環境は，プログラミング環境を提供するだけでなく,「収集，整理，分析」を支援する環境である必要がある，と言うことができる．

情報Iは大きく「(1) 情報社会の問題解決」,「(2) コミュニケーションと情報デザイン」,「(3) コンピュータとプログラミング」,「(4) 情報通信ネットワークとデータの活用」，に分類される．

これらの中から，特に，プログラミングが密接に関わってくると思われる,「(3) コンピュータとプログラミング」,「(4) 情報通信ネットワークとデータの活用」について，次に詳しく見ていく．

2.2.1 コンピュータとプログラミング

情報Iの分類のうち「(3) コンピュータとプログラミング」については，次のようなことを理解し，技能を身に着けるよう指導することが求められている[1].

- **(ア)** コンピュータや外部装置の仕組みや特徴，コンピュータでの情報の内部表現
- **(イ)** アルゴリズムを表現する手段，プログラミング
- **(ウ)** 事象をモデル化する方法，シミュレーション

[1]指導要領ではこれらを大項目「ア：知識及び技能」および「イ：思考力，判断力，表現力等」に分類した上で，それぞれを枝番（ア）（イ）（ウ）に細分化しているが，ここでは各大項目の枝番（ア）（イ）（ウ）のみをそれぞれ要約して載せている

(ア) については,「コンピュータの基本的な構成や演算の仕組み（中略）コンピュータ内部でのプログラムやデータの扱い方，値の範囲や精度について」の理解が含まれており，**コンピュータの仕組み**の基本的な理解が必要とされている．

（イ）については,「アルゴリズムを文章，フローチャート，アクティビティ図などによって表現する方法（中略）**外部のプログラム**との連携を含めたプログラミング」や「アルゴリズムによって処理の結果や**効率に違いが出る**こと」，また,「プログラムを作成する力，作成したプログラムの動作を確認したり，不具合の修正をしたりする力」などを養うことを目的としており，**アルゴリズム**（アルゴリズムの**評価**を含む），**デバッグ**などに言及している．指導要領解説においては，活動の例として,「気象データや自治体が公開している**オープンデータ**などを用いて数値の合計，平均，最大値，最小値を計算する単純なアルゴリズムや，**探索や整列など**の典型的なアルゴリズムを考えたり表現したりする活動」や「プログラミングで（中略）**アプリケーション**ソフトウェアが持つ（中略）機能の一部を実現したり,（中略），カメラや**センサ及びアクチュエータ**を利用したり，画像認識や音声認識及び**人工知能などの既存のライブラリ**を組み込んだり，**API** を用いたりすることなどが考えられる」などを挙げている．

（ウ）については,「実際の事象を図や数式などにモデル化して表現する方法」「モデル化した事象を (中略) 表現し条件を変えるなどしてシミュレーションする方法」「シミュレーションを通じてモデルを改善する方法」などについて,「**プログラミング**，**シミュレーション専用ソフトウェア**，表計算ソフトウェア」を用いて学ぶとあり，題材には,「数学科と連携し，不確実な事象を含む確率的モデル」「数学や物理などの事象」「乱数を用いたシミュレーション」などが例示されている．

また,「(3) コンピュータとプログラミング」全体にまたがる活動について,「コンピュータや携帯情報端末などで使われている**アプリケーションソフトウェアの機能の一部**について，内部ではどのようなプログラムが働き情報が処理されているのか考え，**プログラミング言語で表現する**学習活動」などが例示されている．

2.2.2　情報通信ネットワークとデータの活用

情報 I の分類のうち「(4) 情報通信ネットワークとデータの活用」については，次のようなことを理解し，技能を身に着けるよう指導することが求められている

- **(ア)**　情報通信ネットワーク，プロトコルの役割及び情報セキュリティを確保するための方法
- **(イ)**　データを蓄積，管理，提供する方法，情報通信ネットワークを介して情報システムがサービスを提供する仕組み
- **(ウ)**　データを表現，蓄積するための表し方と，データを収集，整理，分析する方法

(ア) については，ネットワーク通信の基本となる「プロトコル」や,「LAN 等の小規模な情報通信ネットワークの仕組み」の理解，例として「**電子メールを送受信するときの情報の流れ**」などを例示している．

(イ) については,「情報システム」についての理解が求められている.「情報システム」が何かについては明確な定義はされていないが，例として,「POS システム（Point Of Sales system）や ATM（Automatic Teller Machine)」が挙げられていることや，情報システムを理解するには「データベース管理システム」や「情報システムが提供するサービスの多くが情報通信ネットワーク上のシステムで稼働していること」の理解が必要としていることから,「情報システム」は**ネットワークで接続**されている複数のコンピュータからなる，**データを収集・共有・処理**する仕組み，と言うことができる.

(ウ) については，データベースの中でも特に「分析」に関わるもの，具体的には，データの「名義尺度，順序尺度，間隔尺度，比例尺度」や「欠損値や外れ値の扱い」など主に**統計的な処理**についての理解を述べている.

2.3 教員研修教材について

『高等学校情報科「情報 I」教員研修用教材』（以下,「研修教材」）[15] は，文部科学省から公開されている,「情報 I」を担当することになる教員向けに，指導要領や指導要領解説で必要とされている活動について，実際の演習例を交えて紹介をした資料である．この資料には，教員自身が研修で行うことを想定した演習項目の他に,「学習活動と展開」という実際の授業において教員が参考資料として活用することを想定した内容が掲載されている.「学習活動と展開」には，ソースコードなどは掲載されていないが，学習の例として書かれている活動は，教員向けの演習で取り上げられていた内容であることが多い．研修教材の 3 章と 4 章の「学習活動と展開」においてプログラミングを用いた授業が例として挙げられている学習について次に示す.

- 「学習 11 コンピュータの仕組み」
  - プログラミング言語を用いて計算誤差の問題を解決する
- 「学習 12 外部機器との接続」
  - センサとアクチュエータを使って身近な問題点を解決する方法を考える
  - プログラムに問題が発生した場合に，原因を見つけ解決する
  - センサから入力された値に応じて動きが変わるプログラムを作成する
- 「学習 13 基本的プログラム」
  - 分岐のプログラムを作る
  - 反復のプログラムを作る
  - 分岐と反復を組み合わせたプログラムを作る
- 「学習 14 応用的プログラム」
  - リストを用いたプログラムを作る

  - 乱数を用いたプログラムを作る
  - プログラムを関数で分割する

- 「学習 15 アルゴリズムの比較」
  - 線形探索のプログラムを作る
  - 二分探索のプログラムを作る
  - 最大探索回数を比較する

- 「学習 16 確定モデルと確率モデル」
  - 一様乱数を使ったプログラミングでシミュレーションし，結果を考察する

- 「学習 17 自然現象のモデル化とシミュレーション」
  - 放物運動についてプログラムなどを作ってシミュレーションを行う
  - 物体を遠くまで飛ばす方法について数式に入れる値を変えて見つける

- 「学習 21 さまざまな形式のデータとその表現形式」
  - キー・バリュー形式で表現されたデータを図示したり，プログラムで扱ったりする

- 「学習 23 質的データの分析」
  - テキストマイニングの基本的な操作について，プログラミング言語やアプリケーションを用いて身に付ける

データ分析や可視化については，表計算ソフトなどを利用して学習することも考えられる．研修教材では教員向けの演習においても表計算ソフトを利用する例が掲載されている．しかし，研修教材に掲載されている「全体を通じた学習活動の進め方」において,「分析や可視化に関しては,(3) のプログラミングと関連して，プログラミング言語を活用してもよいが,(中略) データ分析に適したプログラミング言語を選択することも必要である.」との記述もある．このことから，教員や生徒の習熟度によってはデータ分析や可視化についても，プログラミングが活用されることが想定される．研修教材に掲載されたデータの分析や可視化についての学習活動で，プログラミングを活用する可能性がある内容は次のようなものがある．

- 「学習 20 情報システムが提供するサービス」
  - 国や地方公共団体，民間企業が公開するオープンデータには，どのようなデータがあるか確かめる
  - 複数のデータを組み合わせて分析する

- 「学習 22 量的データの可視化」

  - 複数のデータ項目から関係性が強いデータを見つける
  - 体力測定の項目間の関係を調べる
- 「学習 24 データの形式と可視化」
  - 可視化手法に関して，目的や特性について理解させる
  - グラフから見える特徴や外れ値，傾向などを読み取り，より良い可視化手法を用いて改善や問題の明確化を行う方法を学習する

研修教材の 3 章と 4 章においては，具体的な Python や R 言語のプログラムのソースコードが掲載されている．なお，3 章に関しては，Python 以外の選択肢として JavaScript，VBA，ドリトル，swift を扱った資料がそれぞれ公開されている．研修教材ではこれらのプログラムは基本的に教員自身が書いたり実行したりすることを想定しているものの，実際の学習活動においては生徒がこれらのプログラムを少なくとも（教員から配布されて）実行したり，場合によっては数値などを変更させて試行錯誤させたりすることは十分に考えられる．特に「学習活動と展開」に掲載された内容は実際の授業で参考資料として活用されることが想定されており，これらのプログラムが動作可能であることが，高等学校におけるプログラミングの授業を支援できるかどうかの指標となると考えられる．

2.4 海外の指導要領や実践例との比較

ここまで，日本国内の高校の情報科目について見てきたが，海外における中等教育の現状および，日本で必履修となっている科目「情報 I」の指導要領（以下単に「指導要領」）との比較も行い，先述した Computational Thinking，あるいは「プログラミング的思考」を身につけさせるにあたり，日本ではどのような点に留意すべきかを議論する．

指導要領からは，プログラミングの学習は，アルゴリズムの学習にとどまらず，情報システムやデータの分析など，実用的な領域まで踏み込んだ学習が盛り込まれており，Computational Thinking の節 (2.1) で目指しているような，情報科学の広い分野について学ばせようとしていることが読み取れる．

一方で，情報系科目の標準カリキュラムとして ACM が策定している Computing Curricula[16] と指導要領とを比べた場合には，情報科学として学ぶべき素養として，指導要領には書かれていないものも多く見受けられる．初等中等教育向けの Computing Curricula の標準を策定した CSTA K–12 CS Standards[17] において,「すべての学生が習得すべき内容」として挙げられており，かつ日本の高校生にあたる Level 3A(Year9-10) の項目を見ると,「手順、モジュール、オブジェクトなどの構成要素を用いて、体系的な分析により問題をより小さなコンポーネントに分解することができる。」「ユーザーからのフィードバックを取り入れて、幅広い層に向けたプログラムを体系的に設計・開発できる。」「ライブラリなどのリソースを使用する際に、計算機上の成果物の使用を制限するライセンスを評価することができる。」など，指導要領にはない高度な内容が取り扱われている．一方で，中学生にあたる Level 2(Year6-8) の項目を見ると

「フローチャートや疑似コードを使って、複雑な問題をアルゴリズムとして扱うことができる。」「入れ子になったループや複合条件式などの制御構造を組み合わせたプログラムを設計し、繰り返し開発することができる」「既存のコード、メディア、ライブラリなどをオリジナルのプログラムに組み込む」など，指導要領に近い内容が見受けられる．

これらのことから，日本の高校で今後行われるプログラミング教育は，世界的なレベルでみた場合には中学生程度のものを想定しておく必要がある．とはいえ，中学生向けに開発された，ブロック型言語をはじめとする教育用プログラミング言語だけを用いていると，実用的な情報システムの体験や実際のデータに基づいた分析などを十分に行えない可能性があり,「実際の言語」での学習は必須であると考える．

したがって，日本の高校におけるプログラミング学習は,「実際の言語」を利用しながらも，ブロック型言語をはじめとする教育用の言語と同等のサポートを得られるような環境が必要と考えられる．

ブロック型ではあまり問題にはならず，実際の言語を使うときに大きな問題となるものとして，実行前のエラー（コンパイルエラー）を挙げることができる．その原因のほとんどが，文字の打ち間違いなどに起因する文法エラーである.「プログラムというものは，軽微な打ち間違いがあっても動かなくなる」ということ自体は経験はさせる必要はあるものの，その解決にあまりに時間がとられてしまうと，アルゴリズムを構築したり，センサデータを集めたり，データを分析したりといった，本質的な活動を行う時間がなくなってしまう恐れがある．

これらのことから，生徒が利用する演習環境では,「実際の言語」を利用でき，かつ，生徒たちが本質的な活動に専念できるよう，軽微な間違いに対する支援を十分に行う必要があるといえる．

2.5　プラットフォームが備えるべき機能

これまで見てきたように，高等学校では実際の言語を用いた活動が求められるが，それらのプログラムを動作させるためには，対応した言語の処理系がインストールされていることが前提とされている．1.1 で述べた通り，高校ではソフトウェアのインストールが制限されている環境が多く，研修教材で提示されている活動が実際にはできない可能性がある．

そのため，指導要領で必要とされる演習活動ができるプログラミング環境を，インストール不要で提供する必要があると考えられる．

したがって，生徒が使用する環境は次のような特徴を持つべきと考えられる．

- インストール不要
- エラー対策：生徒自身による修正を支援する，教員が困っている生徒を見つけられる
- 指導要領で学ぶものを網羅

指導要領で学ぶものを網羅するために必要であると考えられる機能について次に挙げる．

- **コンピュータの仕組みの学習ができる**

「コンピュータとプログラミング」では,「外部装置の仕組みや特徴」や「情報の内部表現」，つまり，コンピュータの仕組みを理解することが求められている．このため，メモリに対する比較的低レベルな操作を体験できるような言語とライブラリが必要となる．

- **アルゴリズムの学習と評価ができる**

  「コンピュータとプログラミング」では，アルゴリズムを構築するだけでなく,「アルゴリズムによって処理の結果や効率に差が出る」点も理解させることが求められる．そのため，作成したアルゴリズムについて速度や実行ステップ数，使用メモリ量などを簡単に評価できる仕組みが必要となる．

- **グラフィックスの描画やアニメーションの作成ができる**

  「コンピュータとプログラミング」の分野において，アプリケーションソフトウェアの一部をプログラミング言語で実現する活動が例示されている．このときのアプリケーションには，生徒が想像しやすいものを選ぶべきである．アニメーションやゲームは，生徒が最も親しんでいるアプリケーションと言える．これらのアプリケーションを容易に作成できるようにする必要がある．

  また，グラフィックスやアニメーションは，実行結果をわかりやすく可視化を行ううえでも重要な技術であり，数値をグラフ化したり，時系列に変化する状況をアニメーションで示したりすることを簡単に行えることは，データの分析やシミュレーションを行う上でも有用である．

- **Web アプリケーションの構築ができる**

  前述に関連して，生徒にとってSNSや検索サイトに代表されるWebアプリケーションも「身近なアプリケーション」の1つであり，ゲームなどと同様に作成を容易に行える必要がある．また，Webアプリケーションは，フォームを通じたデータ入力ができるため，後述するデータ収集の窓口としても重要である．

- **スマートフォンや組み込み機器での実行ができる**

  「コンピュータとプログラミング」の分野において，センサおよびアクチュエータを用いたプログラミングが例示されている．スマートフォンには加速度センサやジャイロセンサが搭載されている，また組み込み機器には，温度センサ，湿度センサ，照度センサなどの多様なセンサを用いることができる．実習環境は，PC上での実行だけでなく，これらの機器での実行もサポートする必要がある．

- **ネットワーク経由でのデータの収集・蓄積・共有ができる**

  「情報通信ネットワークとデータの活用」では「情報システム」つまり複数のコンピュータがネットワークで接続されたシステムの理解が求められている．そのようなシステムを構築するために，ネットワーク経由でのデータを蓄積したり，そのデータをプログラミングを用いて読み取り可能にできる仕組みが必要となる．

- **Web API を用いたデータの送受信ができる**

  「コンピュータとプログラミング」の活動例に，Web API を用いることが言及されている．オープンデータの入手，外部のデータベースと連携することで，実社会とのつながりを意識したプログラミングができるようになる．

- **データを読み込んで統計処理などの分析ができる**

  「情報通信ネットワークとデータの活用」では，データの統計的な分析方法についての理解が求められる．先述したように，収集したデータをプログラムから読み出すことができるようにすることに加え，統計処理やグラフ化を簡単に行えるライブラリが必要となる．

これらの活動で使用されるプログラミング環境が複数の環境に渡っていると，生徒の負担，教員の指導の負担となりうる．例えば，組み込み機器などから入手したセンサデータを一旦 USB メモリなどに保存して，PC のプログラミング環境から読み込むために，USB メモリの内容をコピーし，プログラムで実行した結果を一旦ファイルに保存し，表計算ソフトから読み込んでグラフ化する，という作業であると，ファイルの移動・コピーという本質的でない作業に手間がかかってしまうことになる．

したがって，各段階で生徒や教員が使用する環境はなるべく統一するべきであり，各段階のプログラミングをなるべく広くサポートする実習環境を用意すべきである．

第3章　プログラミング教育のための学習環境に関する関連研究

本章では，これまでプログラミング教育を支援するために行われてきた既存研究について，学習者への支援に着目したものと，教員の支援に着目したものに分類して調査する．また，主に高等学校におけるデータの収集・整理・分析といった演習の実践例を挙げる．

3.1　学習環境の評価

3.1.1　アルゴリズムの学習の支援に関する研究

プログラミングの演習を行う主な目的の一つは，アルゴリズムの学習を行うことであり，そのための支援環境が多数開発されている．ここでは，プログラミングの演習における学習者の支援となる環境についてサーベイする．なお，ブロックなどを用いた初学者向けのビジュアルプログラミング環境も存在するが，小中学校においてもプログラミング教育が必履修化されるという背景から，高等学校においてはより実際の社会で利用されているものに近いプログラミング環境を扱うことが多くなると考えられるため，テキストでプログラムを記述する環境について調査を行う．

初学者用プログラミング言語と環境

プログラミングの初学者向けに，教育に特化した専用の言語を作った例を挙げる．西田らはPEN，荻野らは「ますめ」を開発した，次に詳細を述べる．

- PEN[18]
  西田らは，プログラミングの入門養育の目標を職業プログラマの養成ではなく，プログラミングとは何かの理解とコンピュータの本質を理解することと定め，そのためのツールとしてPEN(Programming Environment for Novices) を開発した．必要な要件に，前提知識がほとんどなくても読めるような分かりやすい言語の用意と，タイプミスなどに起因する文法エラーなどで躓かないように支援する機能，そして論理的な間違いを発見しやすいようにプログラムの実行状況を観察できる機能が必要であるとした．

  PENで扱われる言語は，大学入試センターの入試科目で用いられている手順記述型言語DNCLがベースとなっている．開発環境には，プログラムのタイプミスなどを減らすためにプログラム入力支援ボタンが用意されている．繰り返しや条件のテンプレートを自

動で挿入したり，インデントの自動挿入も行うこともできる．プログラムの流れの理解のために，1 行ずつ実行するステップ実行も可能で，変数の中身の変化を確認できるようになっている．

PEN の利用について，大阪大学で JavaScript を用いた授業との比較を行った結果，自己評価では，ともに 3 回目の授業を終えた時点で PEN を用いた方が高かった．試験では，資料の持ち込みおよびプログラムの実行を許可し，互いに対応した問題を出題した．平均を調べると，PEN を用いた方が点数が高く，自己評価および試験成績両方で良い値が得られた．自己評価という心理的な指標については，プログラミングに対するポジティブなイメージにつながり，今後学習を進めるうえでの態度と関係すると考えられるため，そうした側面でも PEN が良い効果をもたらすことを示している．

- ますめ [19]:
  荻野らは，問題解決においてモデル化とシミュレーションを支援する環境「ますめ」を開発した.「ますめ」は，表計算ソフトにおけるセルを変数とみなすことで，変数の視覚化を行う．また，各セルにプログラムを記述することができ，他のセルの値などを参照して式などを記述できる．参照しているセルの値が変更されると，自動で再計算が行われる．プログラム編集画面では，エラーがある場合はセルを赤く表示してユーザに通知する．セルの値を数値で表示するだけでなく，グラフィックオブジェクトの座標や大きさ，色などに関連付けることで理解をサポートする．インストールの手間を省くため，ブラウザ上で動作する環境としている．

既存の処理系で興味をもたせる内容を扱えるライブラリ

教育用に特化した言語ではなく，実用的に使われている実際の言語を用いた学習において，初学者にとって興味を惹く機能を，短い命令で実現できるライブラリを実装した例として，Alice がある．

- Alice[20]
  Cooper らは，情報系の学部に入学した学生が問題解決の手法やプログラミングに必要な概念とスキルを学習できる環境 Alice を提供した．

  Alice は Python を元に開発した 3D アニメーションツールで，各部品がどういった仕事をしているか分かりやすくなるように考えられている．簡単なスクリプトで 3D オブジェクトを出現させたり動かしたりでき，実行中にはマウスやキーボードで操作できる．オブジェクトを操作する命令が用意され，分岐や反復，関数なども利用できる．一方で，座標系と空間内での他のオブジェクトとの関係性，各オブジェクトの向きによって方向が異なることは理解する必要があり，オブジェクトの概念の学習を促進する．ただ，Python を元にしているため，まれに意味が分かりにくいエラーメッセージに遭遇することもある．

  高校生への授業実践を通じて，学習者が熱中して時間が終わっても作業をしていたことからモチベーションの向上が見られ，自己評価ではプログラミングスキルへの自信がつ

いている様子が読み取れた．また，オブジェクトやメソッドなども使いやすかったという感想も得られた．

アルゴリズムの問題の解答を支援する環境

アルゴリズムの理解には，実際にアルゴリズムの構築をする経験が重要であり，アルゴリズムの問題をプログラムを書いて解答するシステムが数多く提案されている，これらのシステムでは，環境の準備や，単純な書き間違いによるエラーなど，アルゴリズムの構築の本質でない部分でトラブルを起こしにくくする工夫を行っている．

- ELP[21]
  Truong らは，インタラクティブな Web ベースプログラミング環境 ELP(Environment for Learning to Program) を開発した [21]．ELP はインストールや Java のコンパイラのような考え方を排除し，問題解決スキルを育む事に集中させようとした．

  ELP は Web で利用できるエディタで，テンプレートを埋める形式の演習を用意している．すでに学習した概念はひな形が配られるため，新しい概念の習得を助け，プログラムを書く複雑さを減らすことができる．また，Web ベースなので学習者の PC には何もインストールする必要がない．システムは，大学のカリキュラムと結びついていて．演習ページでは目的となるプログラムの構造チャート，クラス図，擬似コード，期待される動きを確認しながらプログラムを作成できる．コンパイルと実行はサーバサイドで行い，実行結果がブラウザに返却される．エラーがあると，メッセージがコード画面とは別ウィンドウのエラーページに返ってくる．エラーを発見しやすいように，ソースコードの横には行番号を表示している．補習授業で 30 人の生徒が 2 時間このシステムを使った．その後，フィードバックフォームから 12 人の学生が質問に回答した．回答した学生はそれまで科目の単位を取ることが難しいと考えていたが，Java プログラミングが簡単になったか，理解しやすくなったか，残りの期間もこの環境を使いたいか，という質問にポジティブな回答が得られた．自由記述の設問では，記述と実行を楽にしたことがよかったという意見や，空白を埋めることで本質に集中できることが良かったという意見が得られた．

- PLM[22]
  Quinson らが開発した PLM はプログラミングやアルゴリズムを学習するための環境で，短いフィードバックを繰り返し提供することで自主的な学習をサポートできる．主なターゲットは大学などで実習を受けている学生だが，もっと若い生徒にも使用できる．演習が用意されており，プログラミングの基礎からソーティングアルゴリズム，インタラクションのあるもの，タートルグラフィックスを扱うものや再帰など 200 の課題がテーマごとにまとめられている．また，教員はこれらの教材をアレンジしたりオリジナルで作成したりできる．

  演習に関する説明と問題文，最終的な求められる結果を確認することができ，コンパイルエラーと実行時エラーはウィンドウ下部のコンソールに表示される．実行結果にはグ

ラフィックスが含まれており，ソースコードの実行は通常の実行だけでなくステップ実行も行うことができる．また，不正解だった場合は問題点を通知するメッセージが提供される．言語は Java，Python，scala の三種類が標準で利用できる．テンプレートを提供することで，重要でない難しい概念は隠している．

PLM を利用した 53 人の学生によるレビューでは，教育効果について否定的だったのは 3 人だけで，34 人が強く肯定的だった．面白さについては 31 人が肯定的で，強く肯定的だったのが 16 人だった．また，興味を惹くものであったかという問いに，80 パーセントの学生が肯定的か強く肯定的だった．初心者は構文に早く自信を持つようになり，同じ学習時間でより進んだことを学習できるようになった．

- CloudCoder[23]:
  Papancea らは，課題などを自由に再利用でき，学習者のプログラミングの学習過程を研究するためのデータを収集できるプログラミング演習環境 CloudCoder を開発した．CloudCoder は一つのメソッドか短いプログラムで完成する短い課題を管理するオープンプラットフォームである．エディタはシンタックスハイライトや自動インデントが使える．学習者が課題を提出すると一連のテストケースに対してコードを実行し，結果はブラウザに表示される．C と C++，Java，Python，Ruby が使用でき，実行はビルド及びテスト用のサーバで行われる．

  Papancea らは良い演習課題を作ることの難しさも挙げ，課題を教員が作れるだけでなく，様々な機関の様々なユーザによって書かれた良い課題のリポジトリを作ることも目標とした．リポジトリに登録された課題は誰でも使うことができ，この時点で，Java は 50，C と C++は 60 の課題が 5 人のユーザによって作られていた．関数の中身を実装させる課題と，プログラム全体を書かせる課題を利用できる．関数を実装する課題の採点は，テストケースを用意して戻り値をチェックするが，一部のテストケースは隠蔽され，特定の入力だけに対応するプログラムを書くことがないように工夫されている．プログラム全体の記述を行う課題の採点は，出力を正規表現で確認する．

  利用には，ウェブアプリとデータベースのホスト用と，ビルドとテスト用の 2 つの Linux サーバが必要である．インストールに関しては，障壁となることも多いため，より簡単に導入する方法を調査中である．

  学習者のログはキーストロークに近いレベルで収集していて，編集とすべての提出とその出力が集められている．ログから，どの学習項目への取り組みか，学習履歴を知ることができる．

- SICAS[24]

  Gomes らは，学習者がアルゴリズムを作るときに感じる困難は，プログラミングに限らず他の領域も含めた問題解決能力の低さに原因があるとし，それを伸ばすことでプログラミング能力を向上させることができないかと考えた．そこで，学習を通じて学習者の問題解決能力を向上させることに主眼を置いたシステム SICAS を提案した．

SICASは，フローチャートをGUIで編集し，フローチャートで構築されたアルゴリズムの実行をシミュレートし，アニメーションで見ることができる．作成されたアルゴリズムは，C言語やJavaのコードに自動的に変換することができる．この機能を通じて，アルゴリズムは簡単に複数のプログラミング言語に翻訳でき，アルゴリズム設計で最も重要な要素はその構想であり，コード化されたプログラムではないことを学習者に示すことが重要であるとしている．このシステムを提案した上で，学習環境に重要な特徴としていくつかの機能を挙げた．一つは，学習者の知識レベルを継続的に分析し現在の状態を知り，学習者への効果的な指導が可能であるとした．二つ目は，学習者の学習スタイルの考慮で，学習スタイルのモデルからそれに合致した適切なフィードバックを与えることを目的とした．三つ目はひな形を利用し，学習者が問題解決に詰まってしまったときに，必要な部品をシステムから与えて支援する．四つ目はゲーム要素で，遊びの要素やゲームを題材に含めることで学習者のやる気を引き起こすことが期待される．最後に，知識がついた後にアルゴリズムを構築する環境を提供する．こうした特徴をもつシステムで学習することで，学習の終盤にはプログラミング的な方法で問題解決の手法を考えられるようになる，としている．

課題の出題方法についての研究

前節でみたようなアルゴリズムの演習課題を出題させる形式のシステムにおいては，演習課題の作問に対する負担があったり，学習者に適切な難易度の演習課題を出題することが難しかったりする．このような問題を対応するためのシステムも開発されている．

- CodeWrite[25]:
  Dennyらは，学習者に課題とテストケースを作らせ，他の学習者がつくった課題に解答するとすぐにフィードバックがもらえる演習を行うためにCodeWriteを開発した．CodeWriteには短い課題が集まった大きなリポジトリがあり，ブラウザとネット環境だけでコーディングできる．一度正解した問題は他の学習者の解答も見ることができ，他人の解答を通じてより効率の良いコードなどを理解することにもつながる．

  課題の作成時には，説明文と実装させるメソッドの名前と戻り値の型，10個までのテストケースを設定し，作者自身で問題を解くことで公開される．

  これをAuckland大学の12週間のJavaクラスで用いた．CodeWriteは4週目の終わりに簡単なドキュメントを配り，課題の作り方や回答の仕方の説明を行った．学生には1つの課題作成と10個の課題の正解を課した．期限までに，437人中299人の学生は少なくとも一つの課題を正解したが，10個の課題を正解するなどの要求を満たした学生は138人と少なかった．

  課題で用いられていた概念を調べたところ，代入，分岐，反復，配列など教えられた概念を全て使うのに十分な機会を与えているといえる．解答プログラムの行数を調べると，ほとんどの課題で十分にバリエーションがあったため，課題に正解した後に異なる解答

や考え方に触れることができたと考えられる．例えば，ある学生が，多くの if 文を使っていた課題で，他の学生は反復の中で判定していた．このコードを見比べることで効率よい回答やその考え方に触れることができる．一方，テストケースだけに対応する解答も見られたため，一部のテストケースのみを見せるような工夫を行う必要がある．

- 学習者に適した課題を提案するシステム [26]
  田口らは，理解が追いつかずに挫折してしまう学習者を助けるために，学習者ごとの理解状況を把握してそれに沿った最適な課題を出題する手法を提案した．過去の授業で理解度の自己評価が低かったポインタの単元の提出課題を見ると，ほぼ白紙や教科書を写しただけの学習者が見られ，その多くはその後の課題も白紙やそれ以降提出がなかったりと，学習意欲が失われたと考えられる．自己評価が低くても自分なりにプログラムを作成した学習者もいて，その多くはそれ以降も積極的であったことが見て取れた．

  この手法では，あらかじめ学習内容ごとに難易度に幅を持つ演習課題を豊富に用意しておき，学習者のこれまでに作成したプログラムから判定した学習意欲を元に課題を出題する．課題の選択のために，まず過去のデータを用いて学習者の傾向や嗜好を推測する協調フィルタリングを用いて，課題ごとの得点の推測値を算出する．その後，学習者の意欲をこれまでの課題から教員が評価し，意欲があれば難しい問題を，意欲が低ければ易しい問題を出題する．

  演習では，この手法で選択された課題を推薦課題として提示し，それ以外の課題も選択可能とした．選択課題から，推薦課題を選択したグループ，それ以外の課題を選択したグループ，いずれも未提出のグループに分け，各グループが翌週も課題を提出していたかチェックした．その結果，推薦課題を選択していたグループは，翌週も課題を解いていた割合がほかのグループよりも高かった．一方，未提出であったグループは翌週の課題提出者の割合が著しく低く，一度行き詰まると学習意欲を失ってしまうおそれが非常に高いことも確認できた．また，手法を用いて選出された課題を解いた回数が多い学習者ほど演習全体の達成度も高かった．

3.1.2 センサを用いた演習やデータの収集分析の支援に関する研究

前節に挙げたようなアルゴリズムの演習以外にも，データの収集・分析にもプログラミングは活用されている．ここでは，(1) 組み込み機器などのセンサデータを題材としたデータ収集の事例，(2) オープンデータを用いたデータの分析，(3) 先述した (1)(2) を組み合わせて，センサデータを分析する事例について調査を行った．

センサ・アクチュエータを用いた実践例

ここで挙げるのは，組み込み機器に接続されたセンサデータとアクチュエータを組み合わせた計測と制御をプログラミングで行った実践例である．これらの実践では，基本的に組み込み

機器内で閉じたプログラムを作成しており，センサから得たデータを比較的短い時間で（多くの場合即時）アクチュエータに反映させるため，データの蓄積を行ったり，他のデバイスにデータを送信したりはしないことがほとんどである．

- 高橋らは，センサの値を使ってロボットを制御する体験授業を高校生向けに開いたことを報告している [27]．LEGO mindstorm を用いたロボットプログラミングは多くの教育現場で扱われており，高橋らの報告でも利用されている．この講座は事前に高校で行う出張講義が 2 時間あり，そこでロボットや開発環境についてと，本番で行う課題などが解説される．本番は大学で 5 時間程度の作業時間で，いくつかのプロジェクトに取り組ませる．各プロジェクトは 10 分程度の技術解説があり，その後数問の基本問題，応用課題をグループで取り組むことになる．複数年にわたり高等学校でこの講座は開かれているが，いずれも理系高校生が対象となっているため，一般科目としてこのような内容を扱うためには，難易度を調整する必要がある．

- 大見らは，高等学校で加速度センサを用いて体感的な操作を行うプログラム作品を作る実践を行った [28]．理系の高校生が対象となった授業ではあるが，Procesing を用いたプログラミングも扱われており，生徒の興味を惹く目的で加速度センサが使われている．この授業は希望者向けの課外授業として開かれており，2 時間半程度の講座が 12 から 15 回にわたっている．前半部分を一斉授業によるプログラミング学習としており，後半で作品制作を行いつつ，適宜個別指導を行う形式となっていた．最終的に，加速度センサを活かしたゲームなどを制作したが，時間の都合上，教員側でコーディングをした例があったことも報告されている．普通科高校においては，通常これだけの時間をプログラミングに割くことができないと考えられるため，学習支援環境が必要である．

- 山本らは，小学校 6 年生の理科の単元「電気の働き」においてプログラミング体験をとり入れる授業を報告している [29]．授業には，タブレット端末と Bluetooth で連携できる MESH という無線電子ブロックが使われた．MESH にはセンサ，ボタン，ライトなどがついており，iPad のアプリ上からプログラミングを行うことができる．プログラミングはノードをつなぐだけで行うことができ，タイピングなどは必要とされない．授業内では，センサの値によってモーターを制御する演習を行い，すべてのグループがモーターを制御できた．

- 河村は，小学生向けの公開講座で利用する学習教材の開発を行った [30]．参加者は，タブレット端末から Wi-Fi を経由して Raspberry Pi につないだ LED などを制御する．電子工作とプログラム制御を通じて身の回りの技術へ目を向けるきっかけを与えるため，ブレッドボードへの配線などの電子工作も参加者自身が行うこととなっている．プログラミングは ScratchGPIO8 を使用しており，タブレット端末からのアクセスには Python と Scratch 間の連携ライブラリの scratchpy を用いて作成された Web アプリが使用される．公開講座は 2 回開かれており，1 回目ではキーボード入力に苦戦した様子が見て取れたた

め，2回目は命令を選択するだけでプログラムを作成できるようなサポートを行い，スムーズに授業が行えたことが報告されている．

データ分析の実践例 (DS)

ここで挙げる実践例は，オープンデータなどの既存のデータや，アンケートなどによって学習者が自ら収集したデータを分析する，いわゆるデータサイエンス（DS）の分野に関連した実践例である．分析には表計算を利用した実践や，Python などのプログラムを利用して行った実践がある．学習者が自ら課題を設定して分析を行う場合，分析手法も学習者に選択させているものもあった．

- 春日井は，高等学校における教科「情報 I」で学習される予定となっている，データサイエンスや人工知能と関わりが深い機械学習について研究を行っている [31]．機械学習に関連するいくつかのアルゴリズムに対して，概念の理解に必要となる知識と，高等学校で学習する内容を照らし合わせて学習可能かどうかを検討した．この研究では，高等学校での実践授業を行っており，第 3 学年の生徒に自然言語処理と単純ベイズ分類器を教材として使用した．形態素解析や単純ベイズ分類器，ワードクラウドの作成などのために，Python のプログラミングを配布し，プログラムを用いて処理ができることを体験させている．しかし，学習者のプログラミング経験が少ないため，ワードクラウドで表示したくない単語を追加させる程度の改変をさせるにとどまっていた．

- 阿部の勤務する高等学校では，表計算ソフトを用いたデータ処理ではなく，Python を用いたデータ処理を 1 年次に学習させている [32]．50 分の授業 5 コマを割いており，3 コマ目までに最頻値や中央値，平均，分散，相関係数といった代表値や，データを表現するグラフの種類や特徴などを学習させ，Python や表計算ソフトを用いた統計処理の実習を行った．残りの 2 コマでクラス内へのアンケート調査やインターネット上で収集したデータを，Python や Google スライド，表計算ソフトを使って分析し，その結果を発表した．生徒が分析に取り組んだ全 32 件のテーマでは，Google スライドを利用した分析が大半を占め，Python が利用されていたのは 4 件だった．

- 林は，データサイエンティスト協会から提供されているデータサイエンス教材の教育効果について検証した [33]．教材は，コンビニを出店する場所を候補地の中から選定するもので，授業用スライド，候補地決定のためのデータが入ったエクセルファイル，教材のシナリオが記されたエクセルファイルからなっている．シナリオ通りに進めると，85 分間で完結する内容である．林は，一部のスライドを除いて提示することで，可視化の手法や表計算ソフトによる散布図の作成などを学習者自身に作業させることを提案し，この教材を使った実践授業を高等学校 1 年生向けに行った．授業を通じて，データ分析と結果を読み取る力の成長を学習者が実感している．

- 林は，SSH に指定された高等学校において，データサイエンス教育のカリキュラム開発

に取り組んでいる [34]．3 年間にわたるカリキュラムで，1 年次に準備，2 年次に実践，3 年次に展開という流れで行われる．1 年次に開講される授業の内容が報告されており，1 年をかけて 4 単位の授業になっている．データ収集には，オープンデータを集約したページを提供し，政府の統計や，地域経済分析システムなどが利用された．収集したデータの研磨には，表計算ソフトの基本操作から，度数分布表とピボットテーブルの作成，ヒストグラムや散布図の描画などデータの視覚化，代表値や標準偏差などの求め方などを扱い，Python によるデータ整理・整形も行われている．こうして可視化されたデータを用いて，内容を読み取ったり，意味を理解したりする能力を育成する．2 年次の授業では，探究活動に活用できる統計手法の指導と，Python の理解・活用が予定されており，Python については元となるサンプルコードを加工して扱えるようになることを目指すとされている．

- 岡本は，情報デザイン，データサイエンスなどの学習のために紙飛行機の制作をテーマとした授業を行った [35]．対象となったのは高校 2 年生の選択科目で，4 から 5 人のグループで作業を行った．合計 11 時間を費やしており，紙飛行機を設計し，その制作方法を示した手順書を分かりやすく作成することで情報デザインを，複数回紙飛行機を飛ばした距離のデータを分析してデータサイエンスを学習させることとした．紙飛行機の飛距離はメジャーで計測して記録しておき，それを表計算ソフトに入力して分析を行った．平均値，中央値，標準偏差を解説し，性能の比較や飛距離の分布から個人で考察をさせている．その後，分析結果を受けてもう一度設計から分析と考察までを行わせている．

- 才田らは，高等専門学校の研究題材として，オープンデータを Web ベースで可視化するツールを開発した [36]．ツールの利用者としては，他分野の研究者やこれから研究を始めようとしている人を想定しており，様々な人が容易に利用できることを目指している．報告の時点ではあらかじめツールのデータベースサーバに登録されている地磁気データのみが表示されているが，今後の展望として，教育研究機関が運用するサーバや Dropbox などから自動的にデータ取得を行い，リアルタイムでデータを更新するシステムの構築が構想されている．解析手法も，報告時点での実装は高速フーリエ変換のみとなっており，今後 R を使った拡張が予定されている．グラフ上でデータをインタラクティブに操作するため，JavaScript のライブラリ Plotly.js が用いられている．

- 藤本らは，文系学部の大学生を対象とした人工知能におけるデータの扱いに関する知識を身に着けることができる教材の開発を行った [37]．授業では学習の過程，集めるべきデータの内容，データ分析の手順などを取り上げている．表計算ソフトだけでなく Web ブラウザで Python を実行できる Google Colaboratory が利用されている．Google Colaboratory では，事前にノートブックの作成・配布を行っており，クリック操作のみでデータ分析を実行できる環境を整えている．分析に使用したデータは，架空のデータセットが用いられており，教師データとして過去の大学の卒業生の進路を，検証データに今年度卒業する学生のデータを用意し，卒業後にプロスポーツ選手になるかどうかを判断する．教員の専門性や，表計算ソフトと Google Colaboratory のどちらを利用するか

によって学習効果に差異が生じるかどうかの検証では，確認テストの点数から有意な影響はなく，記述式の設問ではGoogle Colaboratoryを用いた情報を専門とする教員のクラスで具体的な数値を読み取れていることが多かったことが見て取れた．一方で，Pythonによる可視化のコードは事前に用意していたため，表計算ソフトよりも簡単になっていた可能性もあり，今後さらなる検証が必要とされている．

センサデータ収集とデータ分析を組み合わせた実践例

先述した手法を組み合わせて，組み込み機器からセンサデータを収集したうえで，それをPCなど別のデバイスに送信・蓄積して分析を行った事例もある．

- 細合らの報告はIoTを活用できる人材育成のための教材開発に関する内容である．授業の対象となったのは大学院生で，組込みハードウェアや組込みソフトウェアに関するスキルやWebアプリケーション開発，ロボット制御ソフト開発の技術の習得が到達目標としてあげられており，すでにある程度の知識をもった学習者向けのものである．チュートリアルとしてRaspberry Piのセンサ値をネットワークで送信し，送信されたデータをWebアプリケーションで表示するといった演習が行われている．[38].

- 福地らの研究では，小学校高学年向けの科学教室で計測デバイスにデータを保存し，それをPCと接続して表計算ソフトでグラフを作成させる実践の報告があった[39]．この教室は夏季休暇中に4時間程度の授業時間で開かれており，身近な様々な現象について計測を体験させることに焦点が当てられている．この活動を通じて，センサを使うことで計算機が実世界の事象を対象にできることや，コンピュータの主要機能に触れること，データの加工を通じて情報処理におけるコンピュータの役割の一端に触れることも目的としていた．データの収集のためにデータ蓄積サーバなどは用意されておらず，計測用のデバイスにデータがCSVファイルで保存される仕組みとなっていた．収集されたデータは，コンピュータにUSB接続することでアクセスすることができ，小学生はそれを表計算ソフトでグラフにするだけで分析ができる．対象が小学生であることや，時間が4時間ほどしかないことから，データの収集などに必要なプログラミングは授業スタッフが事前に行っていた．

- 様々な地点のデータを収集できるプロジェクトとしてLive E!というものもあり，デジタル百葉箱で収集された気温，湿度，気圧，雨量，風向，風速，二酸化炭素濃度がLive E!のサーバにアップロードされている[40]．Live E!のデータを利用するためにはLive E!プロジェクトの会員への登録が必要である．会員にはデータをアップロードおよびダウンロードするためのライブラリが提供されている．

  滑川らは，このデータを用いて，情報理数科の生徒がPHPでデータを可視化するサイトを作成した実践を行っている[41]．この演習は情報理数科の生徒が対象で，1年次にProcessingを学習し，3年次には自宅でLinuxサーバを立ててPHPやCのプログラムを

書いている生徒もいるなど普通高校よりも情報科学に対する知識がある学習者に向けたものである．演習では，二酸化炭素センサの値をインターネットにアップロードし収集しており，データ分析のためのプログラム作成や，データの可視化サイトを作成している．これまではProcessingやVBAなどローカル環境で完結する作業が多かったため，ネットワーク上のセンサやサーバのデータを取得するプログラミングは，できることの広がりに大きな刺激を受けていたことが報告されている．

- 間辺らは，共通教科情報に関して，IoTを活かした授業実践を行っている [42]．この授業はスマートフォンのSNSなど，高校生が普段から利用しているもののデータの流れの理解や，気温などのデータを分析する必要性の理解などを狙っている．この授業では，事前に教室の気温や気圧のデータをラズベリーパイに取り付けたセンサで収集し，データ蓄積サーバに蓄積している．生徒は，収集されたデータをWebブラウザなどから確認し，表計算ソフトでグラフ化している．この授業は1コマ (65分) と短いものであり，生徒には実際のプログラミングなどはさせず，データの流れや教室の気温の推移などを実際に考えさせる形式の授業であった．アンケートベースで理解度を測ったところ，授業後もデータが「スマートフォンからスマートフォン」に直接流れていると答えた生徒などもおり，ネットワークの仕組みを学ばせる教材には改善が必要であると考察されている．また，分析されるデータに身近な題材を使うこと，グラフ化して分かりやすくすることは，生徒の反応からもデータ分析教育の入り口として適していることが報告されている．

- 渕らは，高校生から大学一年生までを想定したデータサイエンス教育の基本カリキュラムの検討を行っている [43]．収集するデータは慣性計測センサによるもので，これを搭載しておりWi-FiとBluetoothに対応しているマイコンのM5Stackを用いている．データ取得にはArduino言語を用いて処理を書く必要がある．これを単一方向に動かした時の値の変化や複雑な動きをした時の値の変化からデータについての理解を深める演習を提案している．データのグラフ化や，フーリエ変換などにはPythonのプログラムを利用している．

3.1.3 教員が適切な指導を行うための支援に関する研究

教室などの多人数の受講者がプログラミングを行う際には，エラーの修正ができなかったり思い通りの動作を実現できなかったりしている（つまずいている）受講者を発見して，個別に修正方法を提案したり，同じ原因でつまずいている受講者が多数いる場合に全体に注意を促したりするなど，適切な指導を行う必要がある．このようなつまずいている学習者を迅速に発見するため，教員が個別に巡回をすることなく学習者の進捗を一括で把握できるシステムが提案されている．

課題の成否をもとに学習者の進捗を把握するシステム

アルゴリズムの演習課題では，明確な正解が設定されていることがほとんどであり，学習者が書いたプログラムに対する正誤判定を自動化することで，学習者への素早いフィードバックができるだけでなく，教員に学習者の課題の成否を機械的に通知できるようになる．課題が複数ある場合には，課題の達成数によって進捗を計算できるし，単独の課題であっても複数のテストケースを設定することで，部分的に仕様を満たしているなどの詳細な進捗も計算できるようになる．ここでは，そのような手法を用いたシステムの例を挙げる．

- tProgrEss[44]

  西村らは，小コンテスト形式の演習システム tProgrEss において初心者が途中であきらめてしまわないよう部分点を与える実行テスト系列を導入した．この環境において，教師側では提出状況や着手状況を時系列のグラフで表示させ，ソースコードを閲覧して自動判定では見逃される不十分なコードや理解不足の点に注意を与えることができていた．

  それに加え，教師が注意を払うべき部分を発見しやすくするモニタリング機能と，学習者側のミスや躓きを判断して個別指導を支援したり半自動化するチュータリング機能の実装に取り掛かった．モニタリング機能では，全体の進捗を一覧できるページ，提出履歴を閲覧できるページ，提出状況を確認できるページなどがある．提出状況ページでは，詳細な状況の把握のために提出時刻や問題番号，正誤判定の結果などの情報も併せて表示させた．

  教師への効果として，必要に応じて解答を目視確認したり，TA と連携して躓きに迅速に対応したりと，時間を効率的に使うことができた．

- vRoundEd[45]

  太田らは，小コンテスト形式の演習において，授業を進行する教員以外で教室を巡回する補助者を支援するためのシステム vRoundEd を開発した．対応の時間短縮のため，座席表の管理，受講者からの呼び出しと補助者による対応の管理，受講者と補助者間での指導チャット，小コンテストの提出や得点などの進捗閲覧という機能を実装している．

  受講者は事前に着席位置を登録しておき，疑問が生じた場合は挙手ボタンを押すことで補助者を呼び出すことができ，補助者は応対ボタンを押してその挙手に対応する．個人チャットで質疑応答を行うこともでき，簡単な質問は直接の応対が必要ないこともある．得点などを見ることで，補助者は進行の遅い受講者を見つけることができる．座席表をクリックすると受講者の詳細な提出履歴を確認することができる．

  実際の授業で用いたところ，座席表から出席者を管理する機能はおおむね好評であった．また，受講者の呼び出しと補助者の応対を管理する機能も好評で，座席表で受講者の挙手を可視化することは効果があったといえる．チャット機能については，評価は高かったものの，利用頻度は少なかった．

- Hercules[46]
  内藤らは，学習者の進捗状況を把握し，多くの学習者が犯している誤りを発見してその原因を理解したうえで注意やヒントを与えるためのシステム Hercules を開発した．Hercules は，評価スイートを用いてそのプログラムを評価して結果を保存する．評価スイートの達成率から最新の進捗情報がパーセントで教員に提示される．また，評価スイートが検出した誤りをそれを発生させた人数の多い順に並べることができる．

  評価スイートは，静的な評価は事前テスト，必須テスト，補足テストに分けて記述し，動的な評価は実行テストに記述する．事前テストでは例題をコピーしただけのものかどうか，必要なヘッダファイルをインクルードしているか，コンパイルエラーが発生していればその種類を調べる．必須テストでは，要件を満たしているか，例えば，構造体が学習目標であれば，要求されたメンバを持つ構造体があるかなどを調べる．実行テストでは任意の個数のテストケースを用意し出力結果を審査する．補足テストでは，余分なコードが書かれていないかを検査する．ただし，評価スイートはテストケースの妥当性や記述漏れがないかなどのチェックが必要で，完成に 1 時間ほどかかる．完全な評価スイートを作成しようとすると検査項目が 100 個にもなる課題もあった．

- IDISS[47]
  長谷川らは，授業中にリアルタイムに課題を提示し，提出されたソースコードを評価して通知する統合リアルタイム授業支援システム「IDISS:Integrated and Dynamic Instruction Support System」を開発した．解析内容は教授者ならびに TA に通知することで，授業進行をリアルタイムに支援する．ソースコードの評価は，提出されたソースコードがコンパイル可能であるか，インデントが正しくできているか，与えられた UML のクラス図から骨格を形式的に導き出し記述できているか，各メソッドが仕様通りの振る舞いをするかという観点で進行度を計算する．

  教員は，座席位置と対応して進行度を確認でき，携帯端末のブラウザからも進行度がリアルタイムに確認できる．

  検証から，提出履歴から学習者の試行錯誤が詳細に読み取れることを確認できた．また，問題に対する平均提出回数と提出された解答における評価項目別エラー履歴の傾向から，試行錯誤しているがなかなか正解できない学習者の前兆を予測できる可能性が示唆された．

学習者の行動からつまずきを把握するシステム

プログラミングの演習では，先述したデータ分析などのように，明確な正解を設定できる題材を扱うだけではない．このような正解が設定できない状況であっても，学習者のエラーの発生率や，演習にかけている時間などから進捗を割り出す手法が提案されている．また，学習者のソースコードの変更履歴，打鍵の様子などを可視化することで，教員や学習者自身が演習の振り返りを行うことができるシステムも提案されている．

- C3PV[48]

井垣らは，学習者のコーディング過程を記録し，ログをリアルタイムに計測，ランキング化して講師が学習状況を把握することをサポートするシステム C3PV を開発した．コーディング過程は，学習者ごとにプログラム行数，課題ごとのコーディング時間，単位時間当たりのエディタ操作数，課題ごとのエラー継続時間の 4 種類を計測する．これらを表にして教員に提供することで，相対的に進度が遅い学習者をシートマップと対応して確認できる．

実験では，演習開始から 1 時間後に限ればほぼすべてのケースで実際に支援を必要としていた学習者に直接対応ができた．中には，課題をあきらめて他のことをやっていた学習者への注意も含まれていた．学習者へのアンケートにおいても，このシステムに対する肯定的な回答が多く，否定的な回答をした学習者は元々優秀な学習者であったことも分かった．これらのことから，理解不足で悩んでいる学習者を効率的に早期発見することが可能になったと考えられる．

- 加藤らの授業支援システム [49]
  加藤らは，プログラミング演習における学習者の学習状況を取得しそれを教員に提供する学習状況把握支援機能を実装した．分析機能として，エラーの種類を分類する「コンパイルエラー分類機能」，エラーが発生した行が模範解答のどの行に当たるかを差分から計算する「エラー行分類機能」，コンパイルや提出などの時刻が外れ値かどうか判定することで進みの遅い学習者や解答に行き詰まった学習者を発見する「行き詰まり検知機能」を有し，これらの分析結果が教員に提供される．

  実際の授業にシステムを用い，支援機能を用いなかった授業と比較したところ，指導回数には差が見られなかった．指導の内容を調べたところ，全体の進度から遅れている学習者や，コンパイルの時間間隔が長い学習者へのサポートや指摘が行われており，支援機能によってしか把握できない状況への指導が見られた．

- 市村らの授業支援システム [50]
  市村らは，挙手をしない学習者や講師が想定しないポイントで躓いている学習者を発見することが難しいという問題に対し，プログラミング初学者が抱えている問題を早期に発見するためのプログラミング学習支援システムを開発した．

  コンパイル時のエラーメッセージや実行結果をログとして収集し，同一エラーを継続している時間やエラーを繰り返し出している学習者を教員に提示するようにした．共通して抱える問題に対して，発生したエラーごとの解決時間の平均や分散も集計するようにした．

  授業実践では，挙手の前に個別対応ができたことが学習者と TA 両者のアンケートからわかり，躓いている学習者の早期発見に効果があったことが示唆された．共通して抱えている問題への対応については，教員が提示したヒントが自分が悩んでいるところと同じであったという学習者からのアンケート回答が一定数得られたことから，全体への適切な指導にも効果があると考えられる．

- ますめのフィードバック機能 (荻野ら)[51]

荻野らは，全体の授業の進行を止めることなく生徒の状況を把握するため，ますめ [19] に生徒の学習活動を記録して再利用することを目指した．学習活動を再構築するため，クリック操作やキーボード入力，セルの選択，実行やリセットなどのシステムの操作処理などのアクションをすべてキャプチャした．

リアルタイムに提供するデータは，生徒が活発にアクションを起こしているか否かということだけでも教師には有用であると考え，収集したデータからアクションの発生回数を分析した．アクションの発生回数を 5 から 10 秒間隔で集計することで活動状況を効果的に提示できるとした．

非リアルタイム型のデータ分析では，すべてのアクションを記録したことで，任意の時点での生徒の状況を再構築できるようになった．しかし，現時点では，状況を再生しようとすると活動がない期間も動かない画面を見ていなければならず，改善の余地がある．

- PPV[52]
  松澤らは，学習者自身によるプロセス分析を行うためのツール Programming Process Visualizer を開発した [52]．このツールは，操作ログをローカルファイルに保存し，活動後のプロセス観察を目的としている．Eclipse および松澤らが開発している教育用ツールに埋め込みが可能で，一文字の編集からログ収集ができる．

  ますめと同様，作業過程を再現する機能を有し，加えて，作業項目の分析として，何を目的とした作業であるかという情報を付与することができる．はじめに作業項目と見積データを入力し，演習後に作業範囲を設定することで作業項目ごとの実績データを得ることができる．ユーザは各作業項目データに基づき，見積もりと実績の比較ができる．

  このツールを，大学でプログラミング入門教育を終えた学生の履修する科目において用いた．作業時間については，課題の難易度が高いほど見積もりの精度が悪いことが分かった．作業項目については，観察の仕方やタスクの区切り方に個人差が見られ,「メソッドを作る」や,「希望通りに動かす」など抽象的なタスクも多く，作業分割が意味をなしていないケースが散見されたため，ガイドラインの作成などが必要であると考えられる．また，アンケートによると，分析を行うため，課題に要した時間の 35%程度の時間が必要であった．このオーバーヘッドは小さくないが，大学の授業で実現可能な範囲であるとしている．

- ClockIt[53]
  Norris らは，学習過程モニタリングツール ClockIt を開発した．ClockIt では，初学者の演習の様子を，経験のある開発者と視覚的に比較することができる．収集したログから課題にかかる時間などを計測することで，異常に長い時間をかけていた学習者などを判断することができる．ログとして，プロジェクトやパッケージのオープンやクローズ，コンパイルの成否とエラー発生時はエラーの情報，起動とファイル操作が収集される．可視化機能は，時間別のアクションの数と，プロジェクトの作業経過がグラフで表示され，教員も学習者も確認できる．

  演習にこのシステムを利用し，成績の上位・中位・下位の 3 人の学習者のログを観察し

た．上位の学習者は，中位の学習者と同程度の時間を費やしているが，コード行数は少なく，中位の学習者は無駄なコードが多いことが分かる．下位の学習者は費やした時間も短く，最終的なコードの行数も少ない．また，成績が下がるにつれてコンパイルしたときにエラーを起こす割合も多くなっていて，下位の学習者は上位の学習者が起こしていないエラーを発生させていることが分かった．

3.2 既存研究で行われていた支援

本節では，3.1 で挙げた既存研究が，プログラミング教育支援のためにとっていた手法をまとめる．

3.2.1 学習者のプログラミングをサポートするための支援

表 3.2.1 に既存の研究で学習者に向けて提供されてきたプログラミング学習支援機能のまとめを示す．本項では，3.1.1 で調査した研究から，学習者を支援するためにどのような手法が取られていたかをまとめたこれらの項目について考察する．

表 3.2.1 既存研究で提供されている学習者への支援機能のまとめ

各研究	日本語	テンプレート	興味を惹く題材	Web 環境	ステップ実行	エラー修正
PEN[18]	✓	✓			✓	
ますめ [19]			✓	✓	✓	✓
Alice[20]			✓			
ELP[21]		✓		✓		
PLM[22]		✓	✓			
CodeWrite[25]		✓		✓		

日本語の利用

日本における研究では，英語に慣れ親しんでいない生徒のために日本語を使える環境もある．西田らの PEN は日本語で処理を記述する [18]．英語がベースとなっている命令に比べて動作の理解がしやすくなるという効果が期待できる．

テンプレートの提供

学習者がエラーを起こすことを未然に防ぎたいという考えや，学習したい概念に集中させたいといった考えに基づき，ゼロからプログラムを組ませるのではなく，テンプレートを用意し

ている環境が多数あった．Gomes らの研究 [24] では必要なフローチャートの部品を環境側から与えることで解決の支援を行っている．Truong らは，以前学習した項目はすべてひな形にして学習者に配布することで，学習者は新しい学習項目だけに集中できるようにしている [21]．Quinson らの研究 [22] では，Java のメイン文に代表されるような学習の序盤では理解が難しい概念を隠蔽するためにテンプレートを用いていた．Denny らの開発した CodeWrite[25] も，Java のメソッド実装させる課題を扱っており，一つの課題で学習者が実装するのは一つのメソッドだけになっている．

西田らの PEN[18] では，テンプレートが与えられて課題を解くという形式ではないものの，長い日本語のキーワードをタイプするときのミスを防ぐためにボタン一つで繰り返し文や条件文のテンプレートが挿入されるようになっており，入力補助の役割を果たしている．テンプレートを配布することで学習者にある概念の習得に集中させることは有効であると考えられる．ただし，教員によってはすべてを書かせたいと考える場合もあるので，授業の方針によって選択できることが望ましい．

興味を惹く題材の適用

学習者の意欲を高めるために，グラフィックスを含む実行結果が出力される環境を実装した研究があった．荻野らが開発したますめ [19] は，表計算ソフトのセルにあたる箇所にコーディングをすることができ，実行結果をグラフィックスオブジェクトの座標や大きさなどと関連付ける工夫がなされている．モチベーションを上げるだけでなく，実行結果が一目見て分かるようになることで動作の理解をサポートするとしている．Gomes らの研究 [24] でも，学習者の意欲を引き出すために遊びやゲームといった要素を含めることが重要とされている．Cooper らの開発した Alice[20] は，3D オブジェクトを扱えるツールで，高校生への授業実践では学習者が熱中して授業時間が終わっても作業をしていたことが確認されており，自己評価でもプログラミングスキルに自信がついたという結果が得られている．Quinson らが開発した PLM[22] では，用意している課題の中にオブジェクトを動かしたり壁にぶつかったりというインタラクションがあるものや，LOGO のようなタートルグラフィックスを扱ったものがある．こうした課題を扱うことで，多くの学習者が面白さを感じたり，興味を惹かれたりしたことがアンケートからわかっている．

このような研究から，グラフィックスを扱うことが学習者のモチベーションに好影響を与えることが分かる．結果がグラフィックスで表示されるだけでなく，インタラクションがあるゲームのような題材も扱えることが望ましいと思われる．

また，グラフィックスだけでなく，ネットワークを用いたアプリケーション，機械学習などの先進的なトピックを題材として扱えることも，学習者の興味を惹くことにつながると考えらえる．これらの多様な題材を扱うためには，ライブラリの存在が不可欠であり，ライブラリの選択の幅を広げるには，多数の言語に対応しておくことも重要であると思われる．

インストール不要

学習環境の準備は教員にとっても学習者にとっても容易ではないことが複数の研究で指摘されている．教員は演習用の PC へのインストールをしなければならないことや，学習者が演習室以外で作業ができないことが課題として挙げられる．ますめ [19] は，ブラウザ上で動作するためインストールが不要になり，作業内容をサーバに保存することで演習室以外の場所でも作業の続きを行うことができる．また，ブラウザで実行を行うために仮想マシンを JavaScript で実装している．ELP[21] も同様に，Web ベースのプログラミング環境になっている．事前の準備なしで自宅学習などを行うことができるが，ますめとの相違点として，プログラムの編集はブラウザ上で行うが，コンパイルはサーバサイドで行うところが挙げられる．CodeWrite[25] も，ブラウザとネット環境だけでコーディングができるとしている．実際に授業に用いた際には，使用方法を説明した後に期限を設定した課題を出しているが，演習室でしか作業ができない環境ではこのように課題を出すことも難しくなると考えられる．

インストールなどの事前準備が必要な環境や，作業可能な場所が限定されている環境は，授業への導入の敷居が高くなってしまう．また，授業時間外の活動として宿題や課題を出すことも難しくなる．Web ブラウザで編集や実行ができる環境は，こうした問題を解消するために有用であると考えられる．

プログラムの動作理解の支援

従来の演習では，プログラムを実行するとエラーメッセージや結果が出力される．初学者にとっては最終的な結果を見るだけではプログラムの流れを理解することが難しいとする考えもある．PEN[18] は，その理解を助けるために 1 行ずつステップ実行を行うことができる．この機能では，現在実行されている行にマークが表示され，変数の中身がどうなっているかということも確認できるようになっている．ますめ [19] は，表計算ソフトのセルにあたる領域にソースコードを記述するため，どこが参照されているか，あるいは循環参照などが分かるようにステップ実行ができるようになっている．PLM[22] でも通常の実行とステップ実行を選択することができる．また，グラフィックスを用いているため，オブジェクトをアニメーションさせた時の動きが見られることも，プログラムの動作を理解するのに役立つと考えられる．

ステップ実行やアニメーションのように少しずつプログラムが動いていく様子を見せることで，最終的な結果だけを見て動きを推測するよりも理解が容易になると考えられる．また，もしソースコードの中に間違いがあった場合は，ステップ実行のような実行方法を使うことで，どこで動きがおかしくなっているか判断するのに役立つと考えられる．

エラー発生防止およびエラー修正の支援

特に初学者において，文法エラーが発生したときにその原因を突き止め，正しく修正することが難しい場合がある．エラーが発生したとき，たいていの環境ではコンパイラからのエラーメッセージを表示するようになっているが，日本人の場合には，英語のエラーメッセージを読

むことも一つの壁となる場合がある．

PEN[18] では，文法エラーの発生そのものを抑制するために入力支援ボタンが用意されている．キーボードに慣れ親しんでいない学習者に長い命令やキーワードをタイプさせる必要がなくなるので，より本質的な概念の学習に集中させることができるようになる．ますめ [19] は，プログラムが編集されるたびにコンパイルし，間違いがあるセルに色を付けることでエラーを知らせている．

初学者にとっては，ちょっとした間違いから起きている文法エラーでも自力での修正が難しい場合もある．もしエラーの修正に時間を使いすぎてしまえば，学習させたい本質的な概念などに触れる時間も短くなってしまう．そのような事態を防ぐためにも，エラーを起こさせないための工夫を施すことが重要であり，エラーを起こしたときにはそれを自力で修正できるようなヒントを与えることが必要であると考えられる．

学習用ライブラリの提供

プログラミング初学者に対して興味を惹く内容を扱いたい一方で，長い命令文を打たせたり，難しい概念を用いたプログラムを作らせたりということはなるべくさせたくない．出力結果にグラフィックスを扱うために，その表示のためのプログラムをすべて書かせることは現実的ではない．

例えば Alice[20] では，3D オブジェクトを出現させたり動かしたりする演習を行うが，それを初学者でも扱えるように簡単なスクリプトで記述できるようなライブラリを提供している．ますめ [19] や PLM[22] のようなグラフィックスを扱う環境でも同様にライブラリを提供していると考えられる．興味を惹く内容を扱うためには，それによって難易度を上げすぎてしまわないようライブラリを整備することが必要である．

他人の解答を通じた学習

学習者はたいてい，自らの解答を提出し，それが正解していれば次の課題へ移るなどして学習を進めていく．しかし，プログラミングの課題のほとんどは解が一つになるとは限らない．Denny らの CodeWrite[25] は，課題を作らせたり，他の学習者が作った課題を解くことで学習を進めていく環境である．この環境の特徴として，正解した課題については他人の解答を見ることができることが挙げられる．他の学習者の解答を見ることで，自分が考えていたよりも効率よく記述できることに気づいたり，違うアプローチでの解答が見られたりすることで，様々な考え方に触れることができる．

他人が記述したプログラムを見られることで，新しい発見や学びが得られる可能性がある．CodeWrite の実践では，同一の課題について自分と他人のプログラムを比較する手法をとっていたが，作品制作などで自由にプログラムを書かせたときに，他の学習者の作品がどのように動いているか知ることができれば，それも学習に役立つのではないかと考えられる．

3.2.2 センサやデータ分析のための支援

表3.2.2にこれまでに行われてきたセンサやデータ分析を扱った授業における学習支援機能のまとめを示す．本項では，3.1.2で調査した先行研究から，学習者支援の手法についてまとめ考察する．

表 3.2.2 既存研究で提供されているセンサやデータ分析学習支援のまとめ

各実践	組込み機器	データ蓄積	分析ツール	対象
細合ら [38]	✓	✓	✓	大学院
福地ら [39]	✓	△[1]	△[2]	小学校
高橋ら [27]	✓			高校 (理系)
大見ら [28]	✓			高校 (理系)
滑川ら [40]	✓	✓	✓	高校 (情報系)
間辺ら [42]	✓	△[1]	✓	高校
山本ら [29]	✓			小学校
河村 [30]	✓			小学校
春日井 [31]			✓	高校
阿部 [32]		△[3]	✓	高校
林 (DS 導入)[33]		△[4]	△[2]	高校 (SSH)
林 (DS 研究)[34]		✓	✓	高校 (SSH)
渕 [43]	✓		✓	高校-大学
岡本 [35]			△[2]	高校
藤本ら [37]		△[4]	✓	大学 (文系)

[1] ネットワークを通じたやりとりはしていない
[2] 表計算ソフトを利用
[3] 生徒ごとにアンケート調査やオープンデータの活用
[4] サンプルデータ配布

教育用組み込み機器

mbed[39] ,Gainer[28], Raspberry pi[42] など，初等中等教育での利用を想定した組み込み機器が多数開発されている．これらの機器は，USB接続で簡単にPCとデータを交換できる，キーボードやディスプレイを接続すればそれ自身がPCのように動くなど，PCからの接続・開発がしやすい，はんだ付けが不要でセンサ・アクチュエータを接続可能で，電子回路の詳しい知識がなくても機器の拡張が容易である，などの特徴がある．

データの蓄積・共有

間辺らの研究 [42] では，組み込み機器から収集したデータを記録するサーバを用意し，プログラミングから簡単に送信できる API を用意していた．

センサネットワークから気候などのデータをアップロードおよびダウンロードできるサービスとして Live E!があり，会員向けに提供されている [40]．会員には，データのアップロードとダウンロードを行うためのライブラリが提供されており，滑川らの研究 [41] では，この Live E!のサーバへデータをアップロードしたり，サーバに蓄積されたセンサネットワークから収集した気候データなどから，API から使用してデータの可視化をするプログラムの作成演習を行ったりしている．

すでに蓄積されているオープンデータを用いて分析などに活用している事例もある．阿部の研究では，気象庁や統計局が収集・提供しているデータを利用している．林の研究においても，高等学校データサイエンス教育研究会の Web サイトに集約されたオープンデータのページから，政府統計や地域経済などのデータについて利用法の指導が行われている．オープンデータを提供している主なサイトに，RESAS[54] や SSDSE[55]，e-Stat[56] などが挙げられる．RESAS では，地域経済に関するデータが提供されており，Web ページ上でグラフ表示や地図上へのヒートマップ表示といった分析支援機能を提供している．

分析ツール

先述した各種組み込み機器から収集したデータは，USB 経由で機器から PC にダウンロードしたり，組み込み機器からアップロードされたデータ収集サーバからダウンロードしたりして入手する．

入手したデータを用いて行う分析作業には，グラフの作成，平均値や分散などの統計値の算出が多く，表計算ソフトが使われることがほとんどであるが，機械学習などの高度な作業を行う場合には，Google Colaboratory などを用いてプログラミングをする場合もある．このときに使われる Python などのプログラムは，そのままでも動かせるサンプルを配布して，実際に分析が行われていることを確認・体験させることがほとんどである．

情報系高校や，SSH(スーパーサイエンスハイスクール）など，先進的な教育を行っている高校では，PHP や，C#などを使った可視化分析プログラムを作成している例もある．

3.2.3　教員をサポートするための支援

本項では，3.1.3 で調査した研究から，教員を支援するために取られてきた手法をまとめ，考察する．教員の支援では，学習者の状況を教員に把握させるためにいくつかのやり方でログの集計や分析が行われていた．また，授業中に行われるリアルタイムな支援と，授業後の振り返りに使われる非リアルタイムな支援があった．既存研究で進捗を管理するために利用されていた情報を表 3.2.3 に示す．

表 3.2.3 既存研究で提供されている教員への支援機能で利用されていた情報のまとめ

各研究	利用する情報						ログ収集
	課題	テストスイート	ソースコード	エラー	差分	時刻	
tProgrEss[44]	✓	✓				✓	提出
vRoundEd[45]	✓	✓					提出
Hercules[46]	✓	✓					
IDISS[47]	✓		✓			✓	提出
C3PV[48]			✓[1]			✓	キー操作
加藤らのシステム [49]	✓			✓	✓[2]	✓	実行
市村らのシステム [50]				✓		✓	実行
ますめ [51]						✓	キー操作
PPV[52]			✓	✓		✓	キー操作
ClockIt[53]			✓		✓	✓	実行
CloudCoder[23]			✓				キー操作

[1] 行数のみ利用

[2] 模範解答との比較

ログの収集

学習者の状況を分析するために，多くの環境では学習者のプログラムや実行結果などを収集している．ログの収集間隔には研究ごとに差があり，最も粒度の細かい方法では，キーボード入力やクリック操作などの単位で収集されていた．C3PV[48] では，操作回数を元に手が止まっている学習者などの判断に活用していた．細かい粒度で収集することで，学習者の作業の経過をほぼ完全に再現できるという利点もあり，PPV[52] や CloudCoder[23] のように授業後の振り返りのために提供されている環境でキー操作単位でのログ収集が行われていた．しかし，リアルタイムで支援を行う環境では，手を動かしているかどうかの判断にしか利用されていなかった．アクションの発生頻度をどれぐらいの時間で区切って分析するかという点では，間隔が短すぎると変更が目まぐるしく変わり，長すぎると差があまり出ないということが荻野らの研究 [51] から指摘されている．

最も粒度が荒いと思われるのは提出を行ったタイミングでの収集である．この方法は課題の

達成度を元に進捗度を計算していた研究 [44]，[45]，[47] で用いられていたが，課題に対して学習者が完成したと考えたときのログしか集めることができないため，例えば課題に取り組んでいる間にどんなプログラムを実行してきたかという経過を見ることができなかったり，未提出の学習者が全く手を付けていない状態なのか，もう少しで完成できそうな状態なのか，あるいはエラーを出してしまい悩んでいるのかということも見分けることができない．また，Herucles[46] では，学習者のホームディレクトリにあるファイルを一定間隔で調べるという仕様で，ログの収集自体を行っていなかった．

両者の中間の粒度として，コンパイルや実行単位でのログ収集が挙げられる．この粒度のログであれば，学習者が課題を解く過程でどんなプログラムが書かれていたのかを追うことができ，エラーが出ている場合にもその情報を分析することができる．市村らの研究 [50] ではコンパイルを行うたびにログを収集することでエラーを繰り返し出している学習者の特定が行えている．

進捗状況の算出

教員向けの支援では，学習者の進捗度合いが算出され，それを教員に提示することで指導を促す仕組みが提供されていた．進捗の把握による支援を目的とした研究では，多くの場合で課題に対する取り組みを元にして進捗が計算されていた．西村ら [44] や太田ら [45] のようにコンテスト形式の演習がターゲットになっている研究や Hercules[46] などがこれにあたる．課題の成否や，テストスイートで何点取れているか，提出がされているかどうかといった点を元に学習者を順位付けすることで，遅れていると思われる学習者を特定していた．中には長谷川ら [47] のようにソースコードがコンパイル可能か，形式的に記述できているか，インデントが正しいかといった細部までチェックしたうえでの進捗度計算を行っていた研究もあった．しかし，多くの場合は内容をそこまで詳細に見るのではなく，課題を元に進捗度を算出した結果から，遅れている学習者のコードが確認できるようになっていたり，その学習者の座席位置を表示し直接対応を促したりという方法が取られていた．

井垣らの研究 [48] では，課題の成否ではなくプログラムの行数や課題に取り組んでいる時間などを相対的に計算することで，クラス内で遅れている学習者を算出する方法が取られていた．しかし，この方法では授業開始直後に支援を必要としていない学習者が遅れていると指摘されてしまう問題が報告されている．

荻野らの研究 [51] では，単純に学習者がどれだけアクションを起こしたかという点だけに着目して進捗を把握しようとした．しかし，アクションの発生度合だけでは作業をしているかどうかが分かるだけで，順調であるのか，何度やってもうまくいかないのかという詳しい状況を判別することはできないと考えられる．

エラーの分析

進捗状況の把握では全体を見たときに遅れている学習者を見つけて個別指導を行うことに重きが置かれていたのに比べ，エラーの分析では教室内で多発しているエラーの種類を調べた

り，長くエラーに苦しんでいる学習者を見つけたりすることが目的となっていた．

加藤らはエラーメッセージからエラーの分類を行うと同時に，学習者のソースコードと教員が用意した模範解答との差分を比較してどの部分でエラーが発生しているかを計算した [49]．しかし授業で用いた結果は，コンパイルした時間を分析して割り出した進みの遅い学習者への指導が行えたという報告で結ばれており，エラーの分類や差分を元にしたエラー発生場所の特定機能は大きな成果を挙げられなかったと考えられる．特にエラー行を模範解答との差分から見つけ出す手法は，学習者のプログラムが必ずしも模範解答と同じになるとは限らないことから，効果が薄いことが推測できる．

市村らは，同一のエラーを継続している時間やエラーを繰り返し出している学習者を教員に提示した [50]．また，教室内で共通しているエラーについては，発生したエラーごとに解決時間を集計するなどして教員に通知するようにした．結果，学習者が挙手する前に対応ができたり，全体で困っている人が多いエラーなどへのアドバイスが適切にできたことが報告されている．

授業の振り返り

非リアルタイムでの支援として，収集したログを用いて授業の振り返りに利用している研究があった．ClockIt[53] は，コンパイルの回数や実行結果などの他に学習者がプロジェクトに取り組んでいた様子が，ソースコードの差分やテストの様子などから見られるようになっている．なお，この情報は教員だけでなく学習者自身でも確認ができる．PPV[52] は学習者が自身の作業を分析するために開発されたツールであるが，ソースコードの変遷やコンパイルの結果などの情報を確認することができるため，教員が振り返りのために利用することも可能であると考えられる．CloudCoder[23] も学習者の学習過程を研究するためのデータを収集するために開発されたツールであり，ログから学習過程を追うことができる．振り返りでの利用が考慮されている環境では，リアルタイムでの支援に比べて多くの情報を提供している傾向にある．

3.3 プログラミング教育支援の課題

3.1 と 3.2 では，初学者へのプログラミング教育支援環境における，学習者の活動を支援するための手法と教員を支援するための手法，センサデータの収集やデータ分析を題材とした実践についてまとめた．サーベイを通じて，教員や学習者にとってインストールの必要がない Web アプリケーションとして環境を構築することや，グラフィックスなどを扱い学習意欲を引き立てること，学習者の状況を教員に把握させることなどが必要であることが分かった．一方で，本サーベイを通じてプログラミング教育には次のような課題が残っていると考えられる．

エラーの修正が困難

サーベイを通じて，PEN[18] のように入力補助を用いて文法エラーの発生を抑制したり，ますめ [19] のようにエラーのあるセルを赤くして通知するなどの方法でエラーへの対応を行っている環境があることが分かった．それ以外の多くの環境では，コンパイルエラーが起こった時

にはコンパイラが出したエラーメッセージを表示して対応を行っていたが，これらのコンパイラは教育用に作られたものではないため初心者には理解が難しいことに加え，特に日本人にとって英語で書かれたエラーメッセージを元にソースコードを修正するのは難しいと考えられる．また，ますめではエラーのあるセルを赤く表示しているが，その中の何行目でエラーが起きているのかは自力で探さなければならない．

エラー以外のつまずき把握が困難

教員へ学習者の進捗状況を提示したり，エラーの分析を行い，エラーが多発している学習者へのサポートを行う研究もあったが，エラーが発生していなくても，思い通りの結果にならずに試行錯誤している学習者がいた場合にはサポートが難しい．テストケースを利用して進捗を判断し，進捗が送れている学習者へのサポートを行う研究もあったが，高校でのプログラミング活動においては，必ずしも正解が一意に定まらないような活動も想定しなければならない．例えば，センサデータやアンケートの分析などは，データ自身を学習者が集め，その分析結果も様々であるため，明確なテストケースを設定できない場合がある．このような活動においても，学習者の進捗を教員が的確に把握する手段を提供することで，授業を円滑に進めることができるようになると考えられる．

指導要領で求められる活動を網羅できていない

3.1.1 で調査したような既存のプログラミング支援環境においては，主にアルゴリズムの構築や評価に焦点を当てているものが多い．しかし，高校の「情報 I」においては，アルゴリズムの構築や評価も重要な単元の一つではあるものの，他にも「アプリケーションの機能を再現する」「データの収集・整理・分析」「モデル化とシミュレーション」「情報システム（ネットワークで接続されたコンピュータ）」なども重要な単元として位置づけられている．

特に，今回焦点を当てている Web ベースで動作する環境においては，Web サーバにプログラムを送信して実行する方式が多い．しかし，Web サーバで実行すると，結果を受け取るまでにある程度時間がかかるため，入力を頻繁に受け取って出力結果を変化させるインタラクティブなプログラムを構築するのが難しい．このため，ゲームのような生徒にとって身近な題材を扱ったり，シミュレーション結果をアニメーションで表示したりすることが困難である．

また，3.1.2 で調査したセンサやデータ分析のための実践報告の中では，データ収集・分析を行う場合，オープンデータを用いたものが多いが，教員や生徒が独自にデータを集めて分析させることによって，より身近なデータ分析を体験させることが可能になると考えられる．そこで，組み込み機器のセンサや，Web フォームなど，様々なデータソースからデータを入力でき，ネットワークを通じて共有できることが望ましい．このような，データを独自に収集し，プログラミングによって分析までをも行う演習は，理系高校や SSH，大学では実践例はあるものの，普通高校では実践された例は見当たらなかった．その理由として，データの取り扱いとプログラミングとを連携させることが煩雑であることが考えられる．例えば，センサや Web フォームなどデータを集める機器，プログラミングを行う PC とが別のコンピュータである場合，データを適切に転送する必要があるが，初学者にはその作業自体が難しく，うまく動作しないときに，プログラムの誤りが原因か，データの不備が原因などを切り分けることは難しい．多くの受講者がいる中で，教員がそのような状況に陥っている学習者を把握することも難

しいと考えられる.

第4章　本研究の目標

本研究では，高校の「情報」科目におけるプログラミングを用いた活動を総合的に支援するWebアプリケーションの構築を目指している．2章では，高校の「情報I」の指導要領で求められる学習内容について調査を行い，3章では，既存のプログラミング環境や授業実践で行われている内容について調査を行い，問題点を示した．これらの課題を次にまとめる．

- **初学者にとってエラー修正が困難な処理系・開発環境**

  既存の研究で開発されていたWebアプリケーションではエラーを含むプログラムを実行したとき，処理系が出したエラーメッセージをそのまま表示することが一般的であった．しかし，初学者にとって処理系が出す生のエラーメッセージを読んでプログラムの該当箇所を修正することは容易ではない．また，タイピングに不慣れな学習者のために，エラーの発生そのものを減らすための工夫も必要である．

- **指導要領で求められる活動を網羅できていない**

  高等学校の学習指導要領においては，2章で見たように，プログラミングを広範囲に活用することが求められている．例えば，ゲームやシミュレーションなどのインタラクティブなプログラムも実行させられることが望ましいが，既存のWebベースの環境では実行が難しいものが多い．また，APIやオープンデータの活用，センサやアクチュエータの使用，情報システムの構築などは，一台のPCで完結するものではなく，複数の異なる機器やサービスと連携をとりながら実行する必要がある．このような演習では，個別の機器やサービスの登録作業・設定作業や，データの受け渡しなどを行う作業が煩雑になる可能性がある．

- **教員によるつまずき把握が困難な演習内容**

  先述したようなエラーの発生を防止し，エラーの修正を支援したとしても，学習者が多数いる教室ではやはりエラーなどのトラブルが解決できずつまずきの状態に陥る学習者（つまずいている学習者）が同時に多数発生することが考えられる．教授者は，つまずいている学習者を的確に把握し，適切に指導することが求められているが，教授者が学習者一人一人を巡回して確認・指導するだけでは，多数の学習者のつまずきに対処しきれない恐れがある．巡回せずとも教授者が手元のPCだけで確認ができる仕組みも既存研究で提案されていたが，エラーの発生数だけではエラーにはならないが思い通りの結果にならない場合に対処できず，テストケースによる進捗状況の算出は，データの分析のように学習者それぞれが異なる実行結果を出力させるような演習には向かない．

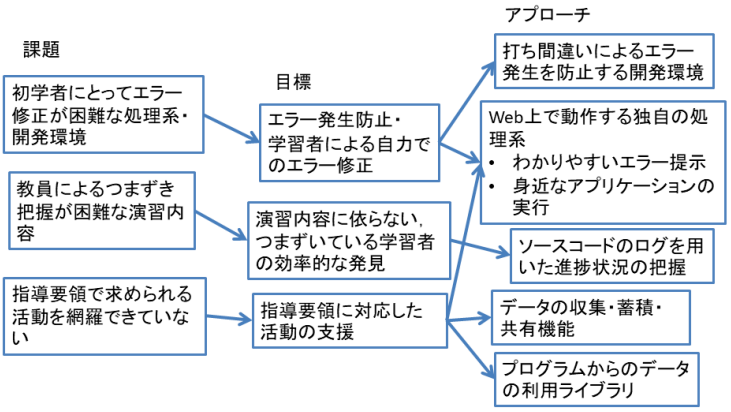

図 4.1.1 本研究の目標と解決方法

4.1 本研究の目標

本研究では，これらの課題を解決し，高校の生徒や教員にとって必要十分な活動を行うことができるプログラミング環境を構築する．それぞの課題に対応する目標を示す．

- エラー発生防止・学習者による自力でのエラー修正

  エラーの原因の大半は単語や記号の打ち間違いによるものである．これらのエラーが頻発することで，アルゴリズムの構築やデータの分析などの本質的な作業の妨げとなりうるため，打ち間違いによるエラーは極力起こさないような防止策を講じる．それでもエラーが起きた場合は，エラーを素早く修正できるように支援する．

- 高校の学習指導要領に対応した活動の支援

  指導要領で求められていることは，単にアルゴリズムをプログラムで実現させることにとどまらず，身近なアプリケーションの機能の一部を再現する活動や，プログラムを使ってデータを収集・整理・分析する活動が求められる．このようなプログラムを実行するために必要な仕組みを提供する．

- 演習内容に依らない，つまずいている学習者の効率的な発見

  テストケースが明確に設定できない演習に対しても，学習者がエラーを起こしていたり，手が止まっていたり，進捗が遅かったりするなどの状況を教授者が学習者一人一人の PC を巡回することなく，教授者の手元の PC で閲覧できるようにする．

4.2 本研究の設計方針

次に，目標を実現するための具体的な設計方針を示す．

- エラーの発生を防止し，エラーを学習者自身が修正できるようにするため，なるべく短いプログラムので目的の動作を実現できるような言語やライブラリを設計し，開発環境による入力支援によって打ち間違いを防止する．

- JavaScrtipt や Python などのプログラムを Web ブラウザで直接実行できる処理系を独自に実装し，エラーが起きた場合には，そのエラーの発生個所や原因をわかりやすく提示する．また，実行中に Web サーバを介さないことで，インタラクティブなプログラムを実行させることを可能にする．

- 教員による学習者の状況把握を支援するため，学習者のソースコードの編集や実行の作業をログとして収集し，教員が各学習者のログの閲覧や，進捗状況の確認ができる画面（ログ閲覧画面）を提供する．また，ソースコードの変更履歴から，その学習者が順調にプログラミングを行っているか，うまく実行できずにつまずいているかを数値化して提示する．

- 高等学校や大学の授業での利用を念頭に，その学習内容や目的に応じてそれに適したプログラミング言語を利用できるようにするため，複数のプログラミング言語をサポートし，データの収集・整理・分析に対応したライブラリを用意する．

ここまでの課題，目標，設計方針（アプローチ）についてまとめたものを図 4.1.1 に示す．本研究では，このような設計方針に基づき，Web アプリケーションによるプログラミング環境 Bit Arrow を開発・実装した．

Bit Arrow のこれらの設計方針がそれぞれの目標を達成できたかの評価を次のように行うことにする．

- Bit Arrow によって打ち間違いによるエラーの発生率が抑えられ，エラーの修正時間が短くなるかどうかを高校での授業実践において，Bit Arrow を使用したクラスとそうでないクラスで比べて検証する．

- 高校の学習指導要領に対応した活動が支援できているかどうかを，指導要領に合わせて発行されている『高等学校情報科「情報 I」教員研修用教材』[15] に掲載されているプログラムを実行できるかどうかによって検証する．

- ログを用いた教員による学習者の状況把握ができているかどうかを，プログラミングの授業における教授者から受講者へのサポート履歴と，Bit Arrow の収集したログとを比較し，サポートを行ったタイミングにおいてログに共通の傾向が見られるかどうか，また教授者がその傾向をログ閲覧画面で確認できるか検証する．

4.3 本研究の全体構成

本研究では，4.2 で述べた設計方針を満たすプログラミング学習環境 Bit Arrow を開発する．5 章では，本システムの詳細な設計を示す．6 章には，設計に沿った実装を示す．7 章には，文部科

学省から公開されている『高等学校情報科「情報 I」教員研修用教材』に掲載されている内容について，Bit Arrow で実装することができるソースコードや仕組みといった実例を挙げることで，高等学校での利用可能性を評価する．8 章では，実際に高等学校で行った実践授業から学習者によるエラー修正の支援効果について検証し，成果を明らかにする．また，収集したログを用いた教員による学習者の状況把握支援について，大学における授業での実践から評価する．

第5章　プログラミング学習環境「Bit Arrow」の設計

本研究では3.2で述べたような，プログラミング教育において要求される特徴をもったプログラミング学習環境Bit Arrowを提案する．本章ではまずBit Arrowの設計にあたっての要件について述べ，次いで設計について述べる．

5.1　プログラミング学習環境「Bit Arrow」の設計方針

3章で取り上げた多くの環境と同じく，Bit Arrowは教員や生徒にとってインストール不要なWebアプリケーションとして動くようにする．その上で，次のような方針で設計を行った．

5.1.1　Webブラウザ上での複数プログラミング言語の実行

3章で取り上げた環境では，対象となるプログラミング言語は多岐にわたっていたことから，教員や目的によって異なるプログラミング言語が利用されると考えられる．そのため，複数のプログラミング言語を実行できるようにする．また，教員からの要望に応じて新しいプログラミング言語に対応できるように設計する．

既存のWebブラウザ上の学習環境では，プログラムを実行するときにソースコードをサーバに送信し，サーバでコンパイルと実行を行ってから結果を受け取りブラウザに表示している．しかし，実行結果をサーバから受け取る方法では，実行中にインタラクティブに入力を受け付けるプログラムは実行することができない．このため，次節で述べるような，アニメーションを用いたり，ゲーム性があったりといった学習者の興味を惹くプログラムを扱うことができなくなってしまう．また，高等学校などではネットワークの速度が不足したり，一度に多くの接続を受け付けることが難しかったりする場合がある．

そこでBit Arrowでは，既存の環境に見られるような，プログラムをサーバに送信して実行する方式をサポートするだけでなく，プログラムをサーバへ送信することなくWebブラウザの内部で実行するようにし，インタラクションのあるプログラムも実行できるようにする．

一方で，多数の処理系をWebブラウザの内部で動かすようにするためには，各言語をWebブラウザの処理系（JavaScript）に変換して動かす必要があり，そのための独自の処理系を新たに用意する必要がある．既存の処理系がもつライブラリの機能を十分に利用するためには，既存の処理系をそのまま動かす必要も残っており，それらの既存の処理系が動かせるWebサーバ側での実行も必要に応じて可能なようにしておく．

システムで演習可能プログラミング言語は，高校教科「情報」で学ぶべき内容を網羅する必要がある．具体的には，2.5 で述べたような次のような活動ができるようなプログラミング言語を複数揃えておく．

- **コンピュータの仕組みの学習ができる**
- **アルゴリズムの学習と評価ができる**
- **グラフィックスの描画やアニメーションの作成ができる**
- **Web アプリケーションの構築ができる**
- **スマートフォンや組み込み機器での実行ができる**
- **ネットワーク経由でのデータの収集・蓄積・共有ができる**
- **Web API を用いたデータの送受信ができる**
- **データを読み込んで統計処理などの分析ができる**

5.1.2 エラーへの対策

学習者がエラーを発生させたとき，それを修正するためにはそのエラーがプログラムのどこで起きたものか理解しなければならない．既存の研究では，エラー発生時の対応として，教育用ではないコンパイラが生成したエラーメッセージを表示するだけのものが多かった．しかし，プログラミングに慣れていない学習者にとって，このエラーメッセージから原因と場所を特定することは難しい．

Bit Arrow ではエラーが発生した際に，原因を示すエラーメッセージを表示させるだけではなく，エラーが発生した場所を明確に示すようにする．エラーメッセージについても，その原因が分かりやすいよう配慮を行う．これにより，エラーが起こった原因やその場所の特定を支援することができる．

そのほかにも，Bit Arrow では括弧やダブルクォーテーションのように対で使われる記号について，入力時に対応する記号の入力を自動で行う．これにより括弧の閉じ忘れなどによるエラーの発生も抑えることができると考えられる．初学者が正確につけることが難しいインデントも自動でつけるようにする．正しいインデントがつけることで，文法エラーはないがアルゴリズムがおかしいプログラムを見た時に，その原因となっている場所が分かりやすくなると考えられる．

5.1.3 データの収集・整理・分析の支援

2 章で議論した通り，指導要領で想定している生徒が作るプログラムは，それ単独で動くものだけはなく，入力となるデータがあり，データを整理・分析した結果を出力するプログラム

も作らせることが想定されている．また，組み込み機器からセンサデータを収集するなど，入力となるデータを収集させること自身を活動として行うことも想定されている．

そこで，Bit Arrow には，次のような仕組みを用意する．

- 学習者が分析対象とするデータ（これを「素材」と呼ぶ）を保存・共有し，プログラムから読み書きできるようにするための領域を持たせるようにする．この領域は 5.1.4 で述べるユーザ管理・クラス管理の仕組みと連動させ，学習者ごとに異なる空間を用意するとともに，クラス内で素材を共有できるようにし，教員からのデータ配布や，グループワークにも対応させる．
- 組み込み機器からの素材を集めやすくするため，先述のデータを保存・共有する領域に対して，ネットワークを通じて組み込み機器からの素材を受け付けたり，手元の PC に接続された組み込み機器からの素材入力をサポートする
- Web API を用いて，オープンデータを提供している他の Web サイトから素材を収集したり，外部の Web サービスに素材を送信したりできるようにする．
- 素材をプログラムで分析した結果をわかりやすく提示するために，グラフライブラリやアニメーションによるシミュレーション結果の表示をサポートする

5.1.4 教員によるクラス・学習者・教材の管理

教員には，クラス・学習者管理のための機能を提供する．受け持つクラスが一つとは限らないことを考慮し，教員のアカウント一つで，複数のクラスを管理できるようにする．教員はクラスの作成と，そのクラスに属する学習者の追加を行えるようにする．

一般的なアカウント管理においては，各ユーザのメールアドレスでユーザを登録し，パスワードを設定，パスワードを忘れたらメールアドレスを使って再発行する，というものが主流である．基本的にアカウント管理は個々のユーザに委ねられていることが多い．

しかし，これを高校などでの授業で行おうとすると，普段コンピューターを使い慣れていない受講者においては，アカウント登録，パスワードの再発行などのアカウント管理そのものが難しく，プログラミングと関係のない作業によって余計な時間をかけてしまう問題点がある．例えば，一般的なシステムにおけるユーザのパスワードは，他ユーザによるなりすましを防ぐため，管理者を含めた他ユーザからは確認できないような仕組みになっていることが一般的であり，その都度再発行の手続きを行う必要がある．また，再発行の際にはメールアドレスを使うことが一般的である．しかし実際の高校の授業においては，生徒がパスワードを忘れてログインできなくなってしまうことが多数発生すると考えられる．また，生徒にメールアドレスを付与していない高校もあるため，再発行のためには，教員などが再発行の手続きを代行する必要もあると考えられる．しかし，複数の生徒がパスワード忘れによってログインできなくなった場合，その一人一人に対してパスワードの再発行手続きを行わなければならない．通常これに対処するのは授業を受け持っている教員で，この対応は教員への大きな負担になるうえ，授業時間もその分短くなってしまう．

そこで，Bit Arrowでは，アカウントに関する運用ポリシー（アカウントポリシー）をクラスごとに設定可能にし，学習者の習熟度に応じてアカウント管理を変えられるようにする．例えば，パスワードを含めたアカウント登録を教員が一括して行ったり，パスワード再発行を簡略化できたり，演習内容によってはパスワードそのものを不要にする設定をしたりすることが可能にする．

5.1.5 教員による課題の管理

Bit Arrowでは，課題や演習に際して，教員がその元となるテンプレートファイルをクラスに登録した学習者全員に配布できるようにする．例えば，あるメソッドの中身だけを実装してほしい場合には，実装してほしいところだけを削除したファイルを配布することで特定の概念に集中して学習させることができる．ファイル配布時にはファイル名を設定するため，進捗を管理するときにはどの課題への取り組んでいるかということが分かりやすくなる．

また，教員は課題を設定できるようにする．このとき，入力のセットと出力結果やソースコードを検査するためのテストスイートを任意で設定するようにする．学習者が提出した課題については，ソースコードとテストスイートの入力に対する出力をチェックすることで，採点やフィードバックを行うことができる．

5.1.6 作業状況把握の支援

教員は学習者の作業の状況を把握することが必要である．

学習者の状況を把握することで多発しているエラーが分かったり，同じエラーで長時間悩んでいる学習者を見つけたりすることができ，それらの問題に対して適切な指導を行うことができる．そこで，Bit Arrowでは次のような状態を把握する支援をする．

- エラーが多発している
- 同じエラーで長時間悩んでいる
- 手が止まっている
- 他の学習者より進みが遅い
- そのような学習者が記述したプログラムやエラーの詳細など

これを支援するためにBit Arrowでは，学習者がプログラムを実行するたびに活動状況を収集し，それを教員側から確認することができるようにする．また，学習者の実行の履歴を記録しておくことで，授業後に学習者がどのような行動をとっていたか観察することも可能にする．これは教員に限らず，学習者が自身の過程を振り返るのに用いたり,（クラスのポリシーにより許可があれば）他の学習者がどのような考え方で学習していたかを相互に学んだりするためにも用いることができる．

5.2 Bit Arrow の設計

Bit Arrow が学習者向けに提供する機能を図 5.2.1，教員向けに提供する機能を図 5.2.2 に示し，それぞれの機能について示す．

- **ユーザ認証**は，ユーザ（学習者および教員）およびそのユーザが属するクラスについて認証を行う．認証方式はクラスの運用方法に応じて複数の方式が選択できるようにする．
- **クラス・ユーザ管理**は，教員ユーザがクラスや学習者ユーザを追加したり，パスワードの再発行を行ったりする．
- **ユーザデータ管理**は，サーバに蓄積されるユーザプログラムと素材の保存・読み出しを行う．
- **IDE** は，Web ブラウザでのユーザプログラムの編集を行い，言語処理系に実行の指示を行う．
- **言語処理系**は，Web ブラウザまたは Web サーバで，各言語プログラムをコンパイル・実行する．
- **外部機器・外部サイト連携**は，組み込み機器からのデータ入力を受け付けたり，外部サイトに WebAPI を発行したりする
- **ログ収集**は，IDE でユーザが行った操作をログとしてサーバに送信し，サーバに蓄積する．
- **ログ閲覧**は，教員ユーザによって，クラスの受講者のログを閲覧する機能を提供する．
- **課題管理**は，教員ユーザによって，課題ファイルの配布，テストケースの作成，採点フォームの生成を行う機能を提供する．

なお，一部で 6 章で後述する実装したシステムの画面を用いているが，これは実装を示すためのものではなく，設計内容をわかりやすく伝えるための補助として提示している．

5.2.1 ユーザ認証，クラス・ユーザ管理

Bit Arrow における，アカウントの種類を図 5.2.3 に示す．

- **教員ユーザ**は，複数のクラスを作成・管理することができる．
- **管理者ユーザ**は，教員ユーザの一種で，教員ユーザの登録を行うことができる．
- **クラス**には，複数の一般ユーザを登録することができる．
- **一般ユーザ (学習者)** は，ある 1 つのクラスのみに属している．すなわち，同一人物がシステム内の複数のクラスを利用する際には，それぞれ別のユーザを作成する必要がある．

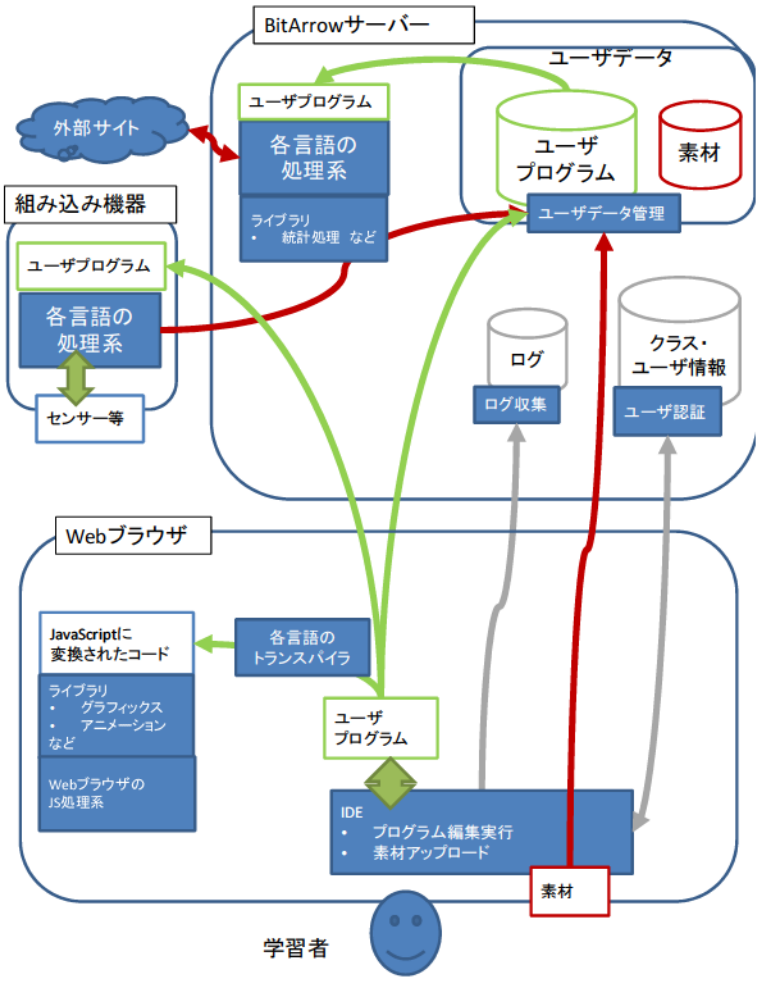

図 5.2.1 Bit Arrow の学習者向け機能

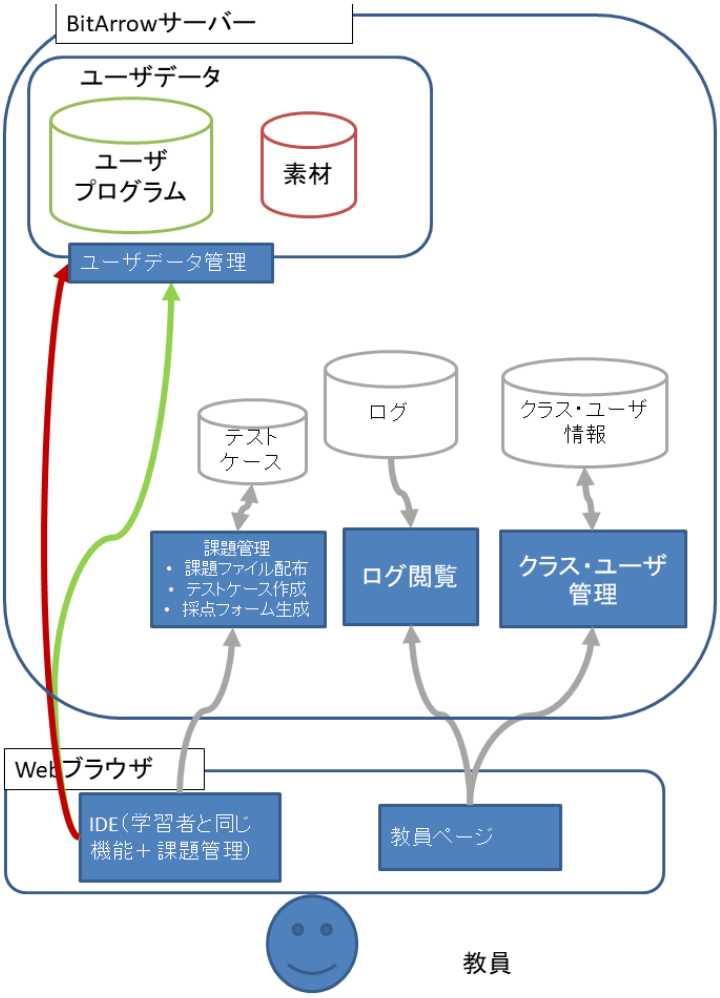

図 5.2.2 Bit Arrow の教員向け機能

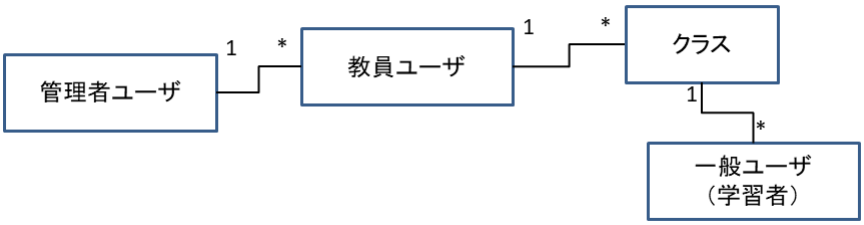

図 5.2.3 Bit Arrow におけるアカウントの構成

Bit Arrowでは，アカウントに関する運用ポリシー（アカウントポリシー）を，クラスごとに設定可能にしている．これによって，クラスの性質により，例えば次のようなアカウントポリシーを使い分けることができるようになる．

- **Bit Arrowによる演習が主体の授業**においては，Bit Arrow上の成果物が成績など重要な情報に結び付く可能性があるため，ユーザ名やパスワードを教員が厳重に管理する．
- **Bit Arrowを授業のごく一部で利用する授業**，例えばプログラミングを少しだけ体験させる授業においては，成果物にそれほど機密性が必要でない場合がある（むしろ，他の人と共有させることを目的とする場合もある）ため，パスワードを使用せずに利用させるようにしてもよい．受講者は決まっているため，教員は受講者のユーザー名のみを予め登録しておく．
- **Bit Arrowの利用を検討している教員に対する研修や，自由参加型のワークショップ**においては，そもそも参加者が誰かが確定していない場合もあるため，ユーザ登録をユーザ自身に行わせることも考えられる．

クラスにおけるアカウントポリシーは，そのクラスを管理可能な教員ユーザによって次のような項目を設定可能とした．

- **「パスワード設定」**は,「使用しない」「使用する」から選ぶことができ,「使用しない」を選んだ場合，そのクラスに登録したユーザは，ログイン画面でパスワードを入力することなくログインすることができる．
- **「ユーザ登録方法」**は,「教員による登録のみ」「ユーザ自身または教員による登録」から選べる．前者を選んだ場合，先述した教員が一括で登録を行い，登録されていないユーザではログインできない．一方，後者を選んだ場合，教員による一括登録に加え，そのクラスに存在しない名前のユーザでログインしようとすると，その場でユーザ登録を要求する（このとき，先述の「パスワード設定」によっては，パスワードの登録も要求する）．

一般ユーザ向けのログインでは，図5.2.4に示すように，クラス名とユーザ名の入力が必須となる．パスワードは，そのクラスで「パスワード設定」が「使用する」になっている場合は入力する必要がある.「使用しない」になっている場合はパスワードは空欄でもログインできる．この場合，クラス名とユーザ名が判明していれば，誰でもログインできることになる．

Bit Arrowでは，受講者のパスワードは，教員からは確認できるようにしている．教員が受講者のパスワードを見ることができれば，再発行などの手続きをせずにそれを受講者に改めて伝えるだけでパスワード忘れにすぐに対応することができる．また，確認できるのが教員だけであれば，受講者のパスワードが悪用されることも考えにくい．教員による受講者のパスワード確認を許可した場合，Bit Arrowの教員用ページにパスワードを表示させる必要がある．このとき，教員は自身の画面をPC教室のモニタに表示させて演習を行っていることも想定される．そのため，うっかり教室全体にパスワードを漏洩させてしまわないよう，確認ページでは教員が表示するというボタンを押すまで表示をしないようにする．また，他人にパスワードが知られてしまった場合などにも備え，教員側からのみ受講者のパスワードの再設定も行えるようにする．

図 5.2.4 ログイン画面

5.2.2　ユーザデータ管理

一般ユーザによって作成・編集可能なデータを「ユーザデータ」と呼ぶ．ユーザデータは，「ユーザプログラム」と「素材」に大別され，さらにユーザプログラムは「非公開プログラム」と「公開プログラム」に，素材は「素材ファイル」と「簡易 DB」に区分される．

それぞれの管理方法について述べる．

非公開プログラムの管理

非公開プログラムの保存領域はクラス毎に区分され，そのクラス内の領域は，ユーザ毎に区分されている．さらに，各学習者が作成する非公開プログラムは「プロジェクト」という単位で区分されている．1 個のプロジェクトはプログラムは 1 つのプログラミング言語に関連づけられ，そのプログラミング言語で書かれたプログラムを複数個格納できる．また，作成できるプロジェクトに上限はない．

非公開プログラムは 5.2.3 で後述する IDE によって，Web ブラウザに一時的に保存され，定期的に Web サーバと同期される．これにより，作業中にネットワークが一時的に遮断されてもクライアント側だけで作業が行える．

公開プログラムの管理

非公開プログラムは，ユーザの操作によって公開することも可能である．公開されたプログラム（公開プログラム）には URL が付与され，Bit Arrow のアカウントを持たないユーザでも

URL を知っていれば誰でもアクセス可能とする．これにより，Bit Arrow を使用していないユーザや，組み込み機器などの HTTP クライアントなどからのアクセスも可能である．

また，公開プログラムに付与された URL と，そのプログラムを作成したユーザの属するクラスは，システム内部のデータベースにおいて紐づけされる．これは,「簡易 DB 管理」の項目で後述するように，システムが URL からクラス名を割り出して，特定のクラスに対応したデータを保存させるためである．ただし，ユーザからはこの紐づけを参照することはできず，ユーザは URL だけからはその名称から作成したユーザやそのユーザの所属するクラスが特定できないようになっている．これは，5.2.1 で述べた通り，パスワードを必要としない設定のクラスにおいては，ユーザ名とパスワードが判明すれば誰でもログインできてしまうため，外部ユーザがなりすましをしにくくするための配慮である．

素材ファイル管理

素材ファイルは，ユーザによってアップロードすることができ，ユーザプログラムからも読み書きすることが可能なファイルである．

素材ファイルの格納先は，各ユーザが個別にファイルを作成・使用する「user フォルダ」と，同一クラス内で共有可能な「class フォルダ」がある．user フォルダはユーザごとに異なるファイルが格納されており，ユーザが削除や上書きを行っても他のユーザには影響しない．一方 class フォルダ内のファイルは同一クラスのユーザであれば学習者・教員問わず他のユーザのファイルを削除したり上書きしたりできる．

各言語の処理系からは，user フォルダ，class フォルダのアクセスはそれぞれ，`"user/foo.txt"`, `"class/bar.txt"`のようにアクセスできる．

簡易 DB 管理

5.1.3 で述べたような組み込み機器などから送られてくるデータや，拡張版 JavaScript などで作成した Web フォームを通じて送られるデータを保存・共有する領域として,「簡易 DB」を提供する．簡易 DB はクラスごとに分けて収集され，データの保存方式には上書き型の KVS(Key Value Store) と追記型のログ形式の 2 種類を用意している．

上書き型ではデータを送信するときにはキーと値をセットで書き込み，そのデータを Bit Arrow のプログラムで取得するときにはキーで検索して値を受け取る．追記型では，データをテーブルごとに分けて記録できる．1 回の送信で送られるデータ（レコード）の内容は数値または文字列の配列であり（6 章での実装では 4 個まで），属性名は指定せず「属性 1」「属性 2」のように呼ぶ．テーブルごとのスキーマの定義を省略し，簡便に使えるようにこのような設計とした．また，送信された時間の情報が自動的に付与される．

また，上書き型でも追記型でも,「グループ」を指定することが可能である．グループが異なる値やレコードにおいては，別の値を格納することができ，グループワークにも対応する．なお，グループ名を指定しない場合は「default」というグループが予め指定されており，クラス

全体の演習においてはグループの概念は意識する必要がない．

これらのデータを Bit Arrow のプログラムから読み出す関数群が各言語むけに提供されており，データを活用したプログラムを作成できる．また，そのプログラムを公開プログラムとして公開した場合，そのプログラムの作成者のクラスに属さないユーザが使用しても（すなわち，そのクラスのユーザであることの認証を行わなくても），そのクラスにデータが届くようにする．これは,「公開プログラムの管理」の項目で述べた，公開プログラムに付与された URL とクラスの紐づけによって実現する．この仕組みにより，例えば，不特定多数のユーザにが利用可能なチャットシステムやアンケートシステムを作成することもできる．その際，クラス関係者でないユーザから具体的なクラス名を特定できないようになっている．

さらに，組み込み機器などから収集したセンサデータなどを，簡易 DB の追記型のテーブルに格納するための Web API（簡易 DBAPI）も提供する．API を使用するにあたっては，API キーは不要であるが，公開プログラムの URL をパラメータに含める必要がある．これによって，その URL と紐づけされたクラスに属するテーブルのレコードとして格納される．パラメータの詳細は 6 章で述べる．

5.2.3 IDE

Bit Arrow ではブラウザ上でプログラムの編集と実行を行う IDE を提供する．学習者はその IDE を操作してプログラミングを行う．エディタは対応する括弧の補完，インデント付けなどの入力補助を行い，事前に打ち間違いによるエラーを防ぐようにする．また，プログラムの実行（デバッグ）も同じページで行えるようにする．

プログラムを書く以外にも，データ分析に必要な素材ファイルを管理する仕組みも提供する．

これらのプログラムファイル，素材ファイルは最新の状態をデータストアに送信し，学習者はどこからでも同じファイルを編集することができるようにする．また，保存や実行がされるたびに，ソースコードや実行結果のログをデータストアに送信する．これらは後述するログ閲覧機能によって後から確認できるようにする．

プロジェクトの作成

5.2.2 で述べた通り，プロジェクトは 1 つのプログラミング言語に関連づけられるため，プロジェクトを作成する際には，図 5.2.5 のように，そのプロジェクト内で使用するプログラミング言語を 1 つだけ指定して作成する．図 5.2.6 のように，1 人の学習者はプロジェクトを複数個作成でき，異なる言語によるプロジェクトを混在させることが可能である．

プログラムの編集

プロジェクト内のユーザプログラムを編集する際に用いるテキストエディタには，初学者が起こしがちな文法エラーや意味エラー，アルゴリズム上の間違いを未然に防ぐ機能を備えるよ

図 5.2.5 プロジェクトの作成

図 5.2.6 プロジェクト一覧

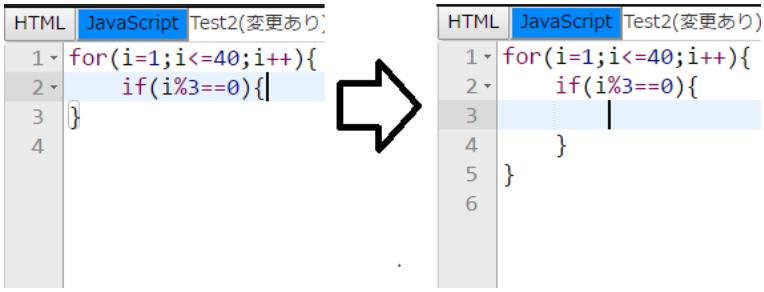

図 5.2.7 括弧の自動入力

うにする．

例えば，if 文や for 文のブロックを表す開き波括弧を書いた後に，初学者は閉じ波括弧を書き忘れることによる文法エラーをたびたび起こす．しかし，波括弧の数が合わないとき，その原因がプログラムの途中にあったとしても，エラーメッセージではプログラムの最後で閉じ波括弧が足りないかのような指摘をされることもある．そこで，図 5.2.7 に示すように「{」を開いた後に改行するタイミングで対応する「}」を自動入力することで，打ち忘れを防ぐようにする．また,「{」や「}」が入力されたタイミングでインデントを自動調整するようにし，変数の範囲を明確にすることで意味エラーを防いだり，プログラムの構造をわかりやすく示しアルゴリズム上の間違いに気づかせたりできるようにする．

これにより，エラーや間違いの発生数の減少と，それらの修正時間の短縮を支援する．

また，Bit Arrow がサポートする言語の 1 つである拡張版 JavaScript（詳しくは後述）は，「HTML」と「JavaScript」という 2 つのファイルで構成されるが，それらを 1 つのファイルのように見せることで，2 つのファイルの関連性を意識してプログラムを書かせるようにしている．

プログラムの実行

IDE で編集したユーザプログラムは，図 5.2.8 のように，IDE と同じ画面に実行画面ダイアログを表示し，ソースコードと実行結果を比べながら確認することができる．

IDE と同じ画面で実行されるユーザプログラムは非公開プログラムであるが，これを公開プログラムとして公開することも可能である．公開プログラムには認証不要でアクセス可能な URL が付与され，IDE とは別のウィンドウ（タブ）で表示を確認できる．スマートフォンなどのデバイスからの実行を簡単に行うため，図 5.2.9 のように URL を QR コードを生成する仕組みもある．

エラーの表示

エラーが発生した際には図 5.2.10 のようなエラーダイアログを画面上に表示することで学習者へ通知を行う（表示される内容は言語によって異なる）．このダイアログは，文法エラーが

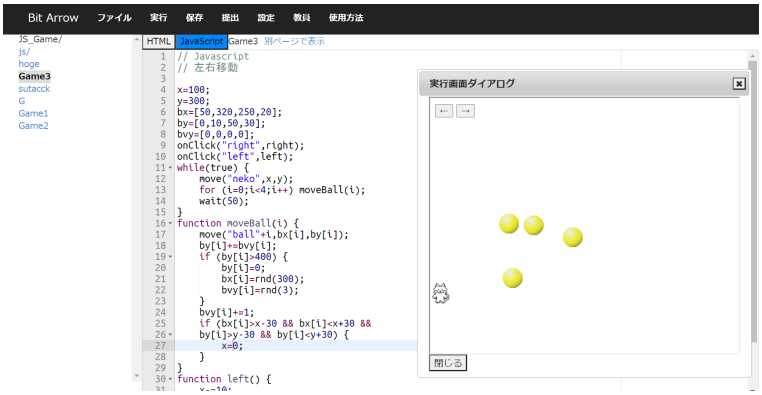

図 5.2.8 実行画面ダイアログ

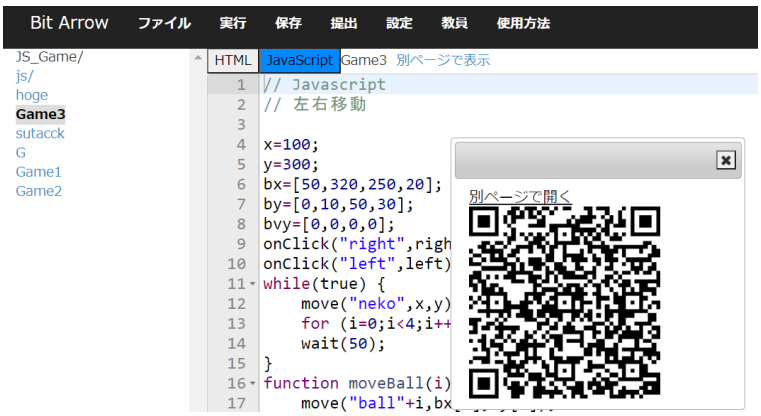

図 5.2.9 QR コードの生成

図 5.2.10 エラーダイアログ

発生するファイルを実行したときすぐに表示される．このことで，エラーの有無や原因の確認のために開発者ツールやコンソールを開く必要がなくなる．また，ダイアログによる通知の内容は，原因を示すエラーメッセージを表示させるだけではなく，エラーが発生した場所周辺にマークを同時に表示させた．これを学習者に提示することで，エラーが起こった原因やその場所の特定を支援することができる．これにより，従来の環境よりも簡単にエラーを確認することができ，エラーの修正を支援する．

教員によるユーザプログラムの配布

教員からも学習者と同じ IDE を利用できる．それに加え，クラスに登録している学習者へのユーザプログラムファイルの一斉配布機能も提供する．教員でログインしている場合のみ，IDE に配布のためのメニューが追加され，現在教員が編集しているファイルを，同一のプロジェクトの同一のファイル名で，クラスに登録している全ユーザに配布することができる．なお，当該ファイルと同じプロジェクト名・同じファイル名のファイルがすでに存在するユーザに対して上書きを行うかどうかは配布時に選択できるようにする．

また，一斉配布した課題ファイルに対して，学習者による提出を行わせることも可能で，提出されたファイルに対してテストケースを設定し，各学習者の提出したプログラムに対する実行結果を一覧できる Web ページ（採点フォーム）を生成可能である．採点フォームには，各プログラムに対するコメント・フィードバックを書き込んで送信することも可能である [57].

素材ファイル管理・出力の共有

5.2.2 で述べたように，ユーザプログラムから素材を扱えるようにするため，IDE には素材のアップロード機能を備える．アップロード先には user フォルダ（各ユーザのみが使用可能）

か class フォルダ（クラス全体で使用可能）を選ぶことができる．

また，実行画面ダイアログから「出力の共有」という機能を用いることで，出力結果を素材ファイルにアップロードすることも可能である．出力結果は，表形式として解釈可能なもの（カンマやタブなどで区切られている）であれば，簡易 DB の追記型のテーブルに追加したり，Web API を通じて外部のデータベースサーバに送信したりすることも可能である．

5.2.4 処理系とライブラリ

Bit Arrow でサポートする言語は原則，プログラムの実行を Web ブラウザ上で行えるようにする．そのため，各言語から JavaScript へ変換するトランスパイラを用意する．また，そのトランスパイラ自身も JavaScript で実装するようにし，トランスパイルも Web ブラウザにて行うことにする．トランスパイルから実行までの処理をすべてブラウザ上で行うことで，ネットワークの負荷を減らし，素早い実行に対応するとともに，インタラクティブなプログラムを作ることもできるようになる．

これらのトランスパイラのプログラムファイルは，Web サーバ側には単なる静的な JavaScript ファイルとして配置され，将来，新しい言語を利用する需要が発生した場合には，トランスパイラの JavaScript ファイルを Web サーバに追加するだけで，対応言語を拡張可能とする．

また，一部の言語では，既存のシステム同様，Web サーバ側でのプログラムの実行もサポートする．

ライブラリには，学習者の興味を惹く題材として，アニメーションを行えるライブラリを提供したり，統計ライブラリを提供し，統計処理やグラフ化の演習を行ったりすることができるようにする．統計処理の際には，処理の対象となる素材ファイルを読み書きできるライブラリを提供したり，必要に応じて Web API を使って他のシステムに接続できたりするようなライブラリも用意する．

これらの題材の違いにより，Web ブラウザでの実行，Web サーバでの実行を使い分けて実行できるようにする．

なお，Web ブラウザ側の実行を司る処理系では，教育的配慮を行った独自の言語処理系の実装を行う．例えば，学習者が編集したプログラムに文法エラーがあった場合，独自実装の処理系では，従来のコンパイラが表示するエラーメッセージよりも分かりやすいメッセージを学習者に提示するように実装してもらうことなどが考えられる．

Bit Arrow では，高校での活動を網羅するため，次の言語を標準で利用できるようにした．

- **拡張版 JavaScript** は，JavaScript をベースとし，初心者向けの配慮を行った言語である．アニメーションやゲーム，Web フォームなど，学習者にとって身近なアプリケーションの作成が容易である．

- **ドリトル**は，拡張版 JavaScript と同様，学習者にとって身近なアプリケーションの作成が容易であることに加え，日本語の命令セットでのプログラミングが可能で，プログラ

ミングの初学者でも簡単にプログラミングができるため，拡張版 JavaScript を始める前段階の入門言語として使用させることができる．

- **簡易 C** は C 言語のサブセットである．C 言語によるメモリアクセスの仕組みを再現し，コンピュータの仕組みについての理解を深めることができる．
- **Python** は，統計処理，機械学習，ネットワーク通信のライブラリが豊富であることや，組み込み機器での実行もサポートすることから「データの整理・分析」の中核となる言語と位置づけられる．各種ライブラリが使用可能なフルセットの Python 言語に対応するため，Web サーバ側での Python 処理系実行をサポートする．また，組み込み機器でも実行できる Python のサブセット言語である MicroPython の実行もサポートし，センサやアクチュエータの制御も演習可能とする．さらに，他の言語同様 Web ブラウザでの直接実行を可能とする BA-Python という言語を開発した．これは Python の入門用として，Web ブラウザで手軽に実行できる言語である．
- **DNCL（どんくり）**は，大学入試センター試験の「情報関係基礎」科目で使用されてきたアルゴリズムの記述言語で，可能な限り日本語に近い表現で記述が可能で，アルゴリズムの構築・評価が容易である．

Bit Arrow で用意する言語と，教科「情報 I」で学習されるプログラミングに関する活動との対応を表 5.2.1 にまとめる．それぞれの言語が特に得意とする項目が各学習項目にあるため，これらを網羅することで学習したい内容に応じて適切なプログラミング言語を選択することができる．

表 5.2.1 Bit Arrow で対応するプログラミング言語と教科「情報 I」で学習される内容との対応

	JavaScript	ドリトル	C	Python	DNCL
コンピュータの仕組み	△	△	◎	△	△
アルゴリズムの構築・評価	○	○	○	○	◎
ソフトウェアの機能の一部を実現	◎	◎	○	○	△
センサ及びアクチュエータの使用	△	◎	-	◎	-
ネットワーク接続	◎	○	-	○	-
データを活用するために必要な収集，整理，分析の方法	○	-	-	◎	-

次に，各言語の設計について示す．

拡張版 JavaScript

拡張版 JavaScript は高校での活動を想定し，アニメーションやゲームの制作，ユーザインタフェースの構築などを初学者でも行いやすくする配慮を行った言語とライブラリのセットであ

る．JavaScript という名称を含んでいることから，基本的な文法は可能な限り JavaScript に似せてあるものの，言語仕様の互換性はない．つまり，JavaScript で書いたものをそのまま拡張版 JavaScript で動かせたり，その逆ができたりすることは一般的にはない．そのため，拡張版 JavaScript で書いた言語を Web ブラウザで実行する際には，拡張版 JavaScript から JavaScript へのトランスパイルを行ってから実行を行う．

以降は，拡張版 JavaScript の言語およびライブラリの特徴について述べる．まず，拡張版 JavaScript では DOM 操作などの頻繁に使われる命令を短く書くためのライブラリを用意している．例えば，リスト 5.1 は一般的な JavaScript で書かれたプログラムであり，乱数に応じて 3 種類の文字列を出力させるものである．このプログラムには，次のような点で初学者にとって難しいと考えられる．

- HTML の script タグの中に JavaScript のプログラムが書かれているため，それぞれ異なる文法が一緒になってしまい初学者が理解することは容易ではない．JavaScript を別ファイルに分けて書くことも可能だが，ファイル管理が煩雑になってしまう．
- 結果の表示に document.getElementById が用いられているが，文字数が長く，初学者に書かせることで，打ち間違いによるエラーが増加するおそれがある．同様に，Math.random や Math.floor も命令が長く，式も複雑で打ち間違いやすい．

リスト 5.1 一般的な JavaScript で書いたプログラム

```
1  <html>
2  <body>
3     <div id="t"></div>
4  </body>
5  <script>
6  jcode=Math.floor(Math.random()*3);
7  if(jcode==0){
8      document.getElementById("t").innerText="Gu-";
9  }else if(jcode==1){
10     document.getElementById("t").innerText="Choki";
11 }else{
12     document.getElementById("t").innerText="Pa-";
13 }
14 </script>
15 </html>
```

リスト 5.1 と同じプログラムを拡張版 JavaScript で記述した場合，図 5.2.11 のようになる．次のような特徴により，前述した問題点を解決している．

- HTML ファイルと JavaScript を別ファイルに分けて書けるようにしている．ファイル管理が煩雑になるのを避けるため，HTML と JavaScript のファイルが一対一に対応するようにし，タブにより容易に切り替えられるよう IDE 側でサポートを行う．

図 5.2.11 教育用 JavaScript で記述したプログラム (上：HTML，下：JavaScript)

- 結果の表示に setText，整数乱数の生成に rnd など，よく使う命令を簡便に書けるようにしている

また，アニメーションなどを用いた作品制作を補助するためのサポートを言語レベルで行う．アニメーションとは，少しずつ異なる画像の描画を一定間隔で待機しながら繰り返し行うことで，画像が動いているように見せる技法である．高校の教科書など一般的に教えられているプログラミングの考え方は,「順次，選択，反復」という概念から導入するのが一般的であり，この考え方からすればアニメーションの動作は反復，すなわち while 文や for 文などの制御構造を用いるのがわかりやすいが，JavaScript の仕様上，while 文や for 文の途中で一定間隔で待機を行うことができない．このため，JavaScript でアニメーションを，setInterval を用いるのが一般的である．例えば，一般的な JavaScript で書いた，画像を右に動かし続けるプログラムをリスト 5.2 に示す.

しかし，setInterval の使用には，一定間隔で動作する内容を指定するために，コールバック関数を渡す必要があり，while 文や for 文とはまったく違う書き方を要求されてしまう．最近の JavaScript では構文が拡張され，async，await などのキーワードを利用して，while 文や for 文の途中で一定時間待機をさせる手法も出てきているが，async や await などの意味を初学者に理解させるはやはり困難であるといえる．

リスト 5.2 一般的な JavaScript で記述したアニメーション

```
1 <html>
2 <body>
```

```
<img id="n" src="images/neko.png">
<script>
x=10;
s=document.getElementById("n").style;
s.position="absolute";
s.top=50;
setInterval(function(){
  s.left=x;
  x+=5;
},100);
</script>
</body>
</html>
```

そこで，拡張版 JavaScript では，同じ動きをするプログラムをリスト 5.3 のように記述できるようにしている．拡張版 JavaScript では，wait という関数を用いるところで，while 文や for 文の中であっても一定時間処理の中断を行えるようにしている．async や await などの予約語も不要になっている．また，リスト 5.1 と図 5.2.11 の例と同様，画像への操作を簡単に行える move 関数を提供している．

リスト 5.3 拡張版 JavaScript で記述したアニメーション

```
x=10;
while(true){
    move("n",x,50);
    x+=5;
    wait(50);
}
```

その他，簡易 DB を操作するためのライブラリである BA-DB や，グラフを表示するためのライブラリも備えている．これらを利用することで，Web フォームからのデータ入力・共有をしたり，データの分析結果を可視化することにも対応する．

Python

Bit Arrow で動作する Python は，Web ブラウザ，Web サーバ，組み込み機器のいずれかで動作させることができる．Python のユーザプログラムをどこで実行するかは，ユーザ自身にも意識させる必要があると考え，図 5.2.12 のようにユーザに明示的に選択させている[1]．

Web ブラウザで直接動作する Python として，BA-Python という Python のサブセット言語を新たに用意した．BA-Python のユーザプログラムは JavaScript にトランスパイルして実行する．また，組み込み機器での実行は，組み込み機器にユーザプログラムを転送して行う．こちらについては 5.2.5 でも述べる．

さらに．統計処理や機械学習などの既存のライブラリの機能を十分に活かせるように，Web

[1]実装では，組み込み機器での実行メニューは組み込み機器を接続している状態でのみ表示されるようにしている．

図 5.2.12 Python の実行メニュー

ブラウザ側の実行だけでなく，Web サーバ側での実行もサポートできるように設計した．本節では主にこのサーバ実行の仕組みについて述べる．

統計処理や機械学習においては，入力データを与えて，それに対応する出力結果を得る，というインタラクションはあるものの，その頻度は 1 回の実行につき数回程度（多くの場合 1 回だけ）であることが多い．ゲームなどで，1 秒間に数十回のインタラクションをすることと比較すれば，リアルタイムなインタラクションは必ずしも必要でないことが多い．したがって，Web ブラウザ側で動かす必要性が比較的乏しい題材であることから，サーバ側に用意した Python 処理系を用いて動かすことが望ましい．

なお，以降「Python 処理系」は，www.python.org で配布されている処理系 (いわゆる CPython) のことを指すものとする．

サーバ側での動作を行うにあたって，次のような点を重視して設計した．

- **危険なプログラムを動作させないようにする**

  Python 処理系は，ファイルなどの各種リソースに直接アクセスする API をもっているため，悪意のあるなしにかかわらず，学習者がプログラムを通じてサーバのリソースを破壊・漏洩させる可能性を考慮しなくてはならない．したがって，API の使用を制限したり，実際の API の動作に変更を加えたりして，不用意なアクセスから守る必要がある．そこで，BA-Python ではサーバ側での実行を安全に行うため，実際のライブラリの機能を一部制限したラッパーライブラリを使用したり，仮想環境（Docker）上で実行する仕組みを用意したりしている．

- **エラーを Web ブラウザ側でチェックする**

  Python のプログラムを Web サーバーで実行する場合，エラーの含まれているプログラムを Web サーバに送信して実行した場合，エラーメッセージが含まれた結果が返ってくる．この場合，エラーメッセージは通常の Python 処理系のものになるため，教育的な配慮がされているものではなく，学習者にとって難解なものになる可能性がある．また，エラーのあるプログラムを送信することによって，Web サーバに不要な負荷をかけてしまう可能性もある．

  このため，実行前に Web ブラウザ側でもエラーのチェックを行い，エラーがある場合は Web サーバへの送信を行わないようにした．また，エラーのチェックは Bit Arrow のために独自に開発した BA-Python を使用するため，エラーメッセージに教育的な配慮を行

いながら表示することもできる．

- **データの収集・整理・分析に必要なライブラリを備える**

  Python は「データの収集・整理・分析」の中核となる言語であり，そのために必要な「データ読み込み」「数値・科学技術計算」「画像処理」「機械学習」「自然言語処理」「グラフ描画」「地図の表示」などのなどのライブラリを使用できるようにする．

- **Python プログラムから素材ファイルへのアクセスを提供する**

  統計処理や機械学習には，入力となるデータ（表形式や画像など）が必要になることが多い．これらのデータは素材ファイルとしてアップロード・共有可能である．したがって，Python プログラムから素材ファイルへのアクセスを提供している．前項のラッパーライブラリにおいて，素材ファイルのフォルダ（user と class）へのアクセスを許可し，それ以外のフォルダへはアクセスできないように制御を行っている．

- **出力結果にグラフや画像を含められる**

  統計処理ライブラリおいては，結果をグラフに表示することが必要になる．また，画像処理を行った結果を表示するときには画像を結果として表示する必要もある．通常の Python の処理系では，処理系が動作している Web サーバのローカルな画面にグラフなどを表示させるようになっており，Web ブラウザ側には画像が表示されないため，出力結果の表示方法を変更して Web ブラウザにも結果を戻すようにしている．

Web サーバ側での実行の流れを，仮想環境を使わない場合と使う場合とで，それぞれ図 5.2.13 と図 5.2.14 に示す．

仮想環境を使わない場合は，ユーザプログラムから呼び出すことができるのは，ラッパーライブラリに限定される．ラッパーライブラリは，本来の Python 処理系が備えるライブラリ（ネイティブライブラリ）の中から，システムに対する危険な動作（OS のシステムファイルの読み書きや，不正なネットワークパケットの送信など）を行わないものに限定して呼び出せるようにしたライブラリである．また，一部の命令については，ネイティブライブラリの動作を変更して呼び出せるようにしている．例えば，ファイルの読み書きについては，書き込む先を素材ファイルのフォルダだけに限定するよう，アクセス先を付け替える処理を行う．IDE から実行の指示があると，まずブラウザ側のエラーチェッカを起動し，ユーザプログラムにエラーがないか判定する．ここでいうエラーには，文法的な誤りに加えて，ラッパーライブラリ以外のライブラリを使用 (import) しようとすることも含まれる．エラーがないと判断されたユーザプログラムは，Web サーバに送信され，Python 処理系で実行される．

仮想環境を使う場合は，IDE 側の動作は仮想環境を使わない場合と同様であるが，Web サーバでは，仮想環境内にインストールされた Python 処理系を用いて，ユーザプログラムを実行する．その際ラッパーライブラリは使わず，ユーザプログラムからネイティブライブラリを直接呼び出すことができる．ユーザプログラムがどんな動作をしようとも，影響は仮想環境内にとどまるので，Bit Arrow の動作するシステム本体（ホスト）には影響がない．ただし，

素材ファイルのアクセスはできるように，ホスト側に設置されている素材データのフォルダだけは，仮想環境内からアクセスを許可している．

これらの動作について，詳しくは 6 章にて述べる．

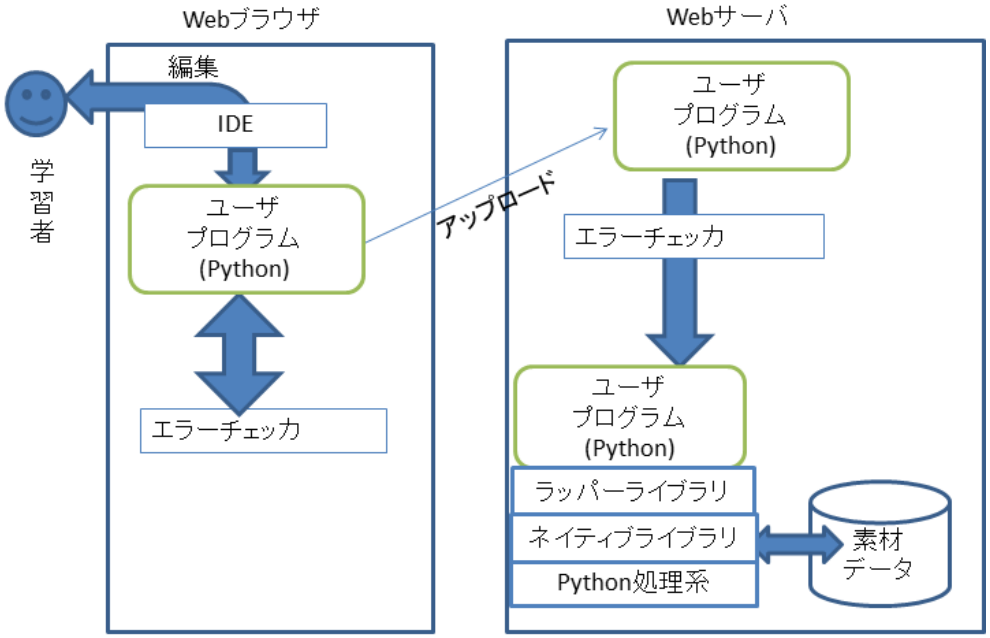

図 5.2.13 BA-Python のサーバ側での実行（仮想環境不使用）

C

Bit Arrow は，C 言語でのプログラミング学習にも対応している．

Bit Arrow 上で動作する C 言語 (簡易 C) は，C 言語の制御構造，変数，関数，構造体，ポインタなどをサポートしている．また，文字の入出力を行なうような標準的なライブラリだけではなく，初学者にも簡単に扱えるグラフィックスライブラリを備えており，興味を持たせる題材を提供することができる．

C 言語はプログラミングの初学者にとっては難しい点が多い．例えば，変数の初期化忘れや，メモリ範囲外のアクセスなど，実行時のエラーに対するチェックがないため，原因不明のエラーに遭遇してしまうこともある．

そこで，BA-C では初学者に向けた支援機能を用意している．例えば，C 言語のポインタを用いて，配列の範囲を超えてアクセスを行った場合は,「配列の添字 [10] にアクセスしようとしました．この配列の有効な添字は [0] から [9] までです」などのように，わかりやすいエラーメッセージを表示したり，ローカル変数に対して初期化を行わずに計算を行った場合，変数が初期化されていない旨の警告を行なったりするようにしてある．[58]

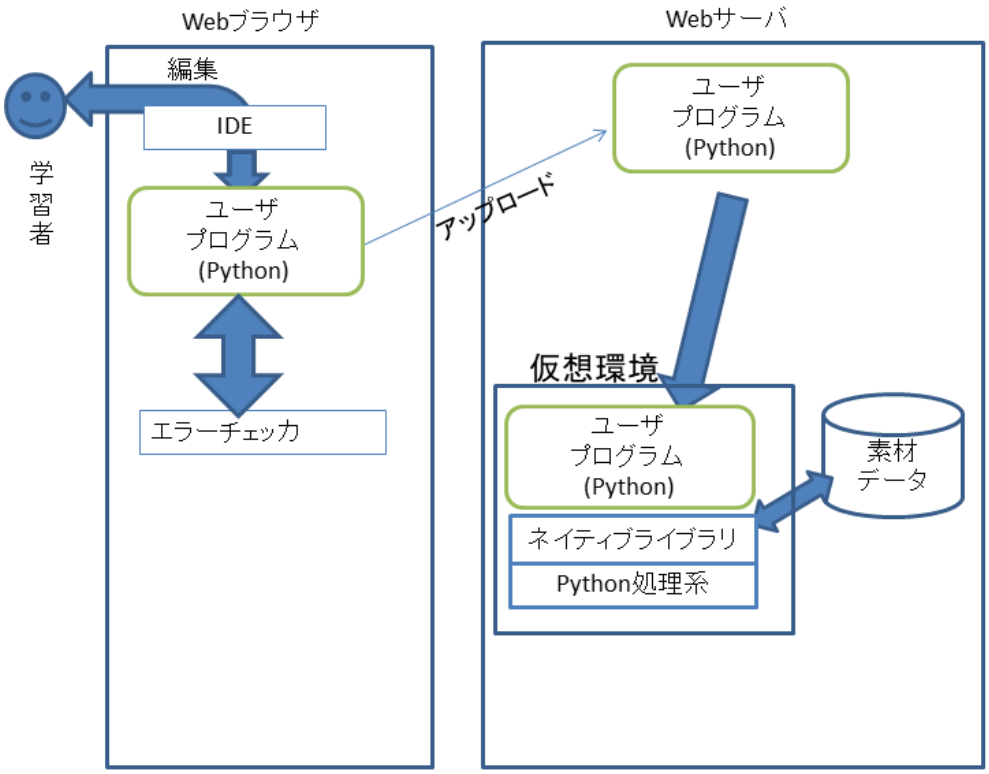

図 5.2.14 BA-Python のサーバ側での実行（仮想環境使用）

ドリトル

ドリトルは教育用に設計された，オブジェクト指向言語である．ドリトルはプロトタイプベースの言語であり，クラスの概念がないことから，初学者がオブジェクト指向を学ぶのに適している [59].

ドリトルは従来，Java アプリケーションとして提供をしており，独自の処理系を Java で実装していた．Bit Arrow で動作するドリトルは，Java 版とは異なる実装とし，他の言語同様ドリトル言語を JavaScript に変換して Web ブラウザ上で実行するようにした．

Bit Arrow で動作するドリトルでは，基盤となる Bit Arrow の機能や Web ブラウザ特有の機能を利用することで，URL を伝えることで作品を公開できたり，スマートフォンやタブレットでの実行に対応したり，さらにスマートフォン内蔵の加速度センサを用いたアプリケーションを作成できるなど，Java アプリケーションにはない機能を提供するようにしている．

DNCL(どんくり)

DNCL（大学入試センターランゲージ）は，大学入試センター試験の入試科目である情報関係基礎におけるプログラミング問題の記述言語である．

DNCL の主な特徴を次に挙げる [60]

- 変数宣言は不要であり,「i ← 3」のような代入文を書くことで変数が使用可能になる.
- 配列は a[i] のように記述し，最初の要素の添字は 1 である.
- 「i を表示する」「i を 1 増やす」など，主な命令が日本語に近い構文で記述できる.
- 制御構造には while,for,if に相当する「i<10 の間……を繰り返す」「i を 1 から 30 まで 1 ずつ増やしながら……を繰り返す」「もし i>j ならば……を実行する」などが用意されている
- 変数はすべてグローバル変数である.
- 関数は『指定された値の二乗の値を返す関数「二乗」を用意する。』など，日本語で定義が導入され，使用するときは「二乗 (x)」のように表記する.

「どんくり」はこのような DNCL の特徴を意識し，並び替えや探索などのアルゴリズムの記述を，プログラミング経験のない学習者でも読み書きしやすくすることを目的に設計された言語である.

どんくりには DNCL の仕様にはない独自機能の拡張もある.

- 関数定義を日本語ではなくプログラムで行う構文．なお，関数の仮引数はその関数の呼出中にのみ有効なローカル変数として振る舞う.
- 「sin」「cos」「切り捨て」「四捨五入」などの数学関数
- 「入れる」「取り出す」「入れ替える」などの配列操作関数
- 変数と値の一覧を表示する「確認」関数
- プログラムの実行時間，繰り返し回数，関数の呼び出し回数などをプロファイルする関数

どんくりのプログラムの例を図 5.4 に示す．これは選択ソートのアルゴリズムを「ソート」という関数で記述し，長さの異なる 2 つの配列に対して「ソート」を適用し，性能を比較したものである．図 5.5 のように，ソートの結果に加えて，実行時間，関数や制御構造の実行回数を表示している．このようなプログラム例を通じて，データ量に対する処理時間の概念や，アルゴリズムによる性能の違いなどを考察させるような演習に活用することが期待されている.

リスト 5.4 DNCL **による並び替えプログラム**

```
1 ソート (hako)は
2     okisa ← 要素数(hako)
3     i ← 1
4     j ← 1
5     i を 1 から okisa まで 1 ずつ増やしながら、
6         min ← hako[i]
7         basyo ← i
```

```
        jをiからokisaまで1ずつ増やしながら、
            もしmin>hako[j]ならば
                min←hako[j]
                basyo←j
            を実行する
        を繰り返す
        入れ替える(hako,i,basyo)
    を繰り返す
    iを1からokisaまで1ずつ増やしながら、
        iと「番目に小さい値は」とhako[i]を表示する
    を繰り返す
を実行する

ソート({10,3,1,4,2})の性能を確認する
ソート({10,6,7,3,1,4,2})の性能を確認する
```

リスト 5.5 DNCL **の出力結果**

```
1番目に小さい値は1
2番目に小さい値は2
3番目に小さい値は3
4番目に小さい値は4
5番目に小さい値は10

統計情報------------------
(実行時間)
0.004秒
(実行回数)
for1 : 5
for2 : 15
for3 : 5
if1 : 比較 15, 真 5, 偽 10
(呼び出し回数)
ソート : 1
要素数 : 1
入れ替える : 5
--------------------------

1番目に小さい値は1
2番目に小さい値は2
3番目に小さい値は3
4番目に小さい値は4
5番目に小さい値は6
6番目に小さい値は7
7番目に小さい値は10

統計情報------------------
(実行時間)
0.009秒
```

```
32 (実行回数)
33 for1 : 7
34 for2 : 28
35 for3 : 7
36 if1 : 比較 28, 真 9, 偽 19
37 (呼び出し回数)
38 ソート : 1
39 要素数 : 1
40 入れ替える : 7
41 -------------------------
```

5.2.5 組み込み機器連携

Bit Arrow では，次の 2 つの方式で組み込み機器との連携を行える．

- 簡易 DBAPI によるデータ送信
- Web ブラウザが動作している PC への USB 接続

前者の方式は，組み込み機器自身に HTTP クライアントの機能がある場合に利用できる．5.2.2 で述べた簡易 DBAPI を通じてセンサデータを送信するプログラムを，組み込み機器の内部で実行する．

後者の方式は，HTTP クライアントを含めたネットワーク機能のない機器でも使用可能である．Bit Arrow にアクセスしている Web ブラウザと同じ PC（ローカル PC）に USB 経由で組み込み機器を接続することで，Web ブラウザの JavaScript から組み込み機器を制御可能である [61]．

Bit Arrow では，ローカル PC に USB 接続された組み込み機器を次の 2 つの方法で制御できる．

- **機器内実行** 組み込み機器上で動作するユーザプログラムを IDE で作成・編集し，組み込み機器に転送して実行する．
- **ブラウザ内実行** Web ブラウザ側で動作しているユーザプログラムから，組み込み機器に対してセンサやアクチュエータへのデータの送受信を行う．

機器内実行では，IDE で編集しているプロジェクトの言語と，組み込み機器上で動作可能な言語を一致させる必要があるが，一度転送されたプログラムは組み込み機器内で完結して動作させることができる．電源などを適切に確保できていれば，プログラム転送後に USB を切断しても動作させ続けれることもできる．また，USB を接続している間であれば，実行結果を IDE の実行画面ダイアログで逐一リアルタイムに確認することも可能である．さらに 5.2.3 で述べた「出力の共有」 を利用して Web サーバの素材ファイルや簡易 DB などに実行結果を送信することも可能である．

ブラウザ内実行は，USB は常に接続した状態で実行させる必要があるが，JavaScript にトランスパイルされた言語であれば，組み込み機器側で解釈可能な言語以外で書かれたユーザプログラムでも実行可能である．

5.2.6 Web APIの実行

外部サイトが提供しているWeb APIへのアクセスについては，次の点に注意する必要がある．

- WebブラウザのJavaScriptプログラムからのWeb APIの呼び出しは，Bit Arrowが動作してるWebサーバと異なるドメインへは直接行うことができない．
- ユーザプログラムの外部サイトへの接続を無制限に許可すると，ユーザの意図するしないかかわらず，過剰なアクセスを行ってしまう可能性がある．例えば，誤って無限ループの中でリクエストを送信し続けるのは避けなければならない．また，APIを提供していない無関係なサイトへのアクセスを避ける必要がある．

これらの点を解決するため，外部サイトが提供しているWeb APIへのアクセスは，ユーザプログラムから直接外部サイトにアクセスさせるのではなく，図5.2.1に示したBit Arrowのサーバ内の「WebAPI呼出」機構で一旦リクエストを受け付け，そこから，システムが許可したサイトへのアクセスのみを，決められた回数のみ行えるようにする．また，WebブラウザのJavaScriptプログラムからも，Bit Arrowのサーバに配置されたWeb API呼出機構であれば同一ドメインであるためアクセスが可能である．

5.2.7 ログの収集と閲覧

Bit Arrowは，ユーザの操作に応じて，実行された時間や実行結果，プログラムのログを収集する「ログ収集」機能，および収集したログを，教員および学習者が閲覧する「ログ閲覧」機能をもつ．

ログ収集

「ログ収集」機能は，IDE内で次の操作を行ったタイミングで起動され，IDEで生成したデータ（ログ要素）をBit Arrowサーバに送信・蓄積する仕組みである．

- プログラムを保存したとき
- プログラムの実行を指示したとき
- 実行しようとしたプログラムにコンパイルエラーがあったとき
- プログラムが正しく実行され，実行結果が返却されたとき
- プログラムが実行されたが，実行時エラーがあったとき
- プログラムの書き換えを行い，一定時間保存操作がなかったとき
- その他，名前の変更，削除など

1つのログ要素に含まれるデータには，次のものがある．

- ユーザ名
- ユーザの属するクラス名
- 操作を行った時刻
- 操作したプログラムのファイル名
- 操作したプログラムのソースコード
- （実行時）出力結果，エラーメッセージ

ログ閲覧

「ログ閲覧」機能は,「ログ収集」機能で収集した情報を閲覧できるページを提供する．このページから，最終実行からの経過時間を元に手が止まっていそうな学習者を把握できたり，実行履歴を元にエラーが続いている学習者を把握できたりという支援が行える．また，各学習者の実行したコードや実行結果を詳細に確認できるようにすることで，どんなエラーに悩んでいるかという情報や，そのエラーを修正するためにどのような変更をしたかという学習者の編集履歴を追跡できるようになる．

また，学習者からも自分自身のログを閲覧することを許可している．教員がクラスの設定を行うことで，他の学習者のログを閲覧することも可能になる．

学習者の状況一覧では，履修している学習者の今の状況を把握することを支援し，現在困っている学習者や作業が止まっている学習者を見つけることができる．また，クラス内で頻発しているエラーの情報を元にクラス全体に注意や指導を行うことができる．

教員は授業において，教室全体の進捗状況と学習者個人の進捗状況の両方を把握しなければならない．そこで，ログの閲覧のためにクラス全体と学習者ごと個別の二種類の画面を提供することとする．クラス全体の状況を把握する画面では，図 5.2.15 のように，特にサポートが必要な学習者を見つけ出すために次のような情報を提供する [62].

- 最近 10 分間など，任意の時間の各学習者の実行結果履歴の概要
- 最後に実行してから経過した時間
- 最後に実行したファイル名

任意の時間における実行結果の履歴を提示することで，エラーが続いている学習者を発見することができる．また，最後に実行してから経過した時間からは，作業の手が止まっている学習者を見つけることに役立てられる．複数の課題が出され，それを順に解いていくというスタイルで授業が展開されるとき，最後に実行したファイル名を一覧することで，それぞれの学習者

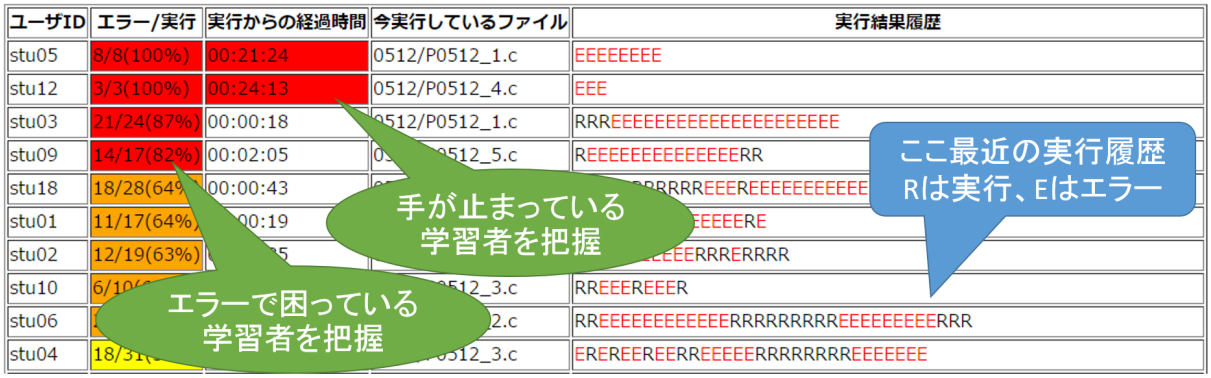

図 5.2.15 学生の実行状況把握画面

が今どの課題に取り組んでいるかを把握することができる[2]．この情報をクラス内で比較することで，他の学習者に比べて遅れている学習者を把握できる．

学習者ごと個別の作業を確認する画面では，より詳細に作業内容を調査するために次のような情報を提供する．

- ソースコードの内容と差分の時系列表示
- 実行結果・エラーメッセージ
- 進捗の詳細

ソースコードの内容と差分を時系列で提供することで，学習者の編集履歴などの作業工程を詳細に調査することができる．実行結果やエラーメッセージから，学習者がつまずいている原因を確認することができる．実行結果から，エラーが出ていないが課題の要求を満たすプログラムが書けていない状態も把握することができる．進捗の詳細として，学習者が書いたプログラムの差分を数値化して提供する．最後に実行したプログラムを完成品と考え，そのプログラムとそこに至るまでの各地点におけるプログラムとの差分を取り，同一行数の増減を元に進捗度を計算する (この同一行数を，本研究では「正解行数」と呼ぶ)[63]．順調に作業が進んでいる場合，この正解行数が単純増加していくと考えられ，正解行数が減少したり横ばいになっているところでは学習者が順調に作業をできていなかった可能性がある．

5.3 設計方針と設計の対応

ここまでで述べた Bit Arrow の設計が，5.1 で述べた設計方針に沿っていることを示す．

- **Web ブラウザ上での複数言語の実行**については， 高校教科「情報」で必要とされる活動を網羅するように言語を選定した．具体的には,「コンピュータの仕組み」には C 言語,

[2]ファイル名と課題とがユーザごとに統一されていることが前提であるが，5.2.3 で述べたファイル配布機能を使って課題のファイルのテンプレートを配布すると，容易に統一することができる

「アルゴリズムの構築評価」にはDNCL,「ソフトウェアの機能の一部を実現」には拡張版JavaScriptやドリトル,「センサおよびアクチュエータの使用」にはPythonやドリトル,「情報の整理，分析」にはPython，などのように，それぞれの活動に対して適切な言語を選んで演習できるようにした．また，ユーザプログラムをJavaScriptにトランスパイルすることで，Webブラウザ上のJavaScriptの処理系で動作させ，インタラクティブなプログラムを動作させることができ，一方既存のライブラリ資源を利用する場合にはユーザプログラムをWebサーバ側で実行することも可能とした．これらの実行方式を，活動の種類に応じて使い分けて利用できるように設計した．将来新しい言語が主流になった場合に備えて，処理系を拡張できる仕組みにも対応している．

また，これらの活動を行うための実行方式として，Webブラウザでの実行とWebサーバでの実行それぞれに対応した．Webブラウザでの実行では，アニメーションなどのインタラクションのあるプログラムを実行可能であり，Webサーバでの実行では，統計処理などの既存のライブラリ資源を利用したプログラムを実行可能である．

- **エラーへの対策**については，まず，エラー自体を起こしにくくする対策として，拡張版JavaScriptにおいては，教育用に特化されたライブラリを用意した．このライブラリを用いることで，命令文が短くなりエラーを減らすことができる．ドリトルにおいては，命令文を日本語で書けることにより，綴りの間違いを減らす効果も期待できる．次に，エラーが起きたときの修正の支援として，IDEによるエラーメッセージや発生箇所をわかりやすく提示することや，Webブラウザで動作する処理系に対してはエラーメッセージを初学者向けにする配慮を行っていることなどが挙げられる．

- **データの収集・整理・分析の支援**については，収集するための手段として，Webフォームの作成，WebAPIを使用したオープンデータの入手，組み込み機器からのセンサデータ入力に対応した．また，収集したデータを蓄積する素材ファイルや簡易DBを用意した．分析については，主にPythonの統計・グラフ化ライブラリに対応することでサポートした．拡張版JavaScriptにも，グラフ化のための関数を用意している．

- **教員によるクラス・教材の管理**については，教員ユーザがクラスを登録し，クラスにユーザを登録する機能，テンプレートとなるファイルを配信する機能によって実現している．特に，ユーザ管理については，クラスの運用形態に応じて，パスワードの管理の手間を軽減させる仕組みを用意した.「パスワードを不要にする設定」「パスワードを学習者に見せる機能」などについては，一般的なシステムではあまり見られないものであるが，高校での実際の運用に合わせて検討した結果，導入した方が望ましいと判断した．これらの設定や機能を使用するにあたっては，パスワード漏洩やなりすましが起きたときのリスクの程度を考えながら，慎重を期して使用すべきである．

- **作業状況把握の支援**については，IDEでの操作（編集，実行結果など）を逐一ログに保存し，クラスのユーザのログを閲覧する機能を備えることで，エラーが頻発したり，手がとまっていたりするなどの，つまずいている生徒の把握ができるようにした．

5.4 先行研究と筆者の研究との比較

3.2 で述べた先行研究で行われてきた支援から，学習者へ向けて提供されていた支援の比較を表 5.4.1 に示す．提案手法は，Web ブラウザで利用できることや，出力結果にグラフィックスを使えることなど，先行研究で提供されてきた支援内容もカバーしている．日本語の利用については，現在ドリトルが提案環境上で動作するようになっているほか，PEN でもベースとして利用されていた大学入試センター試験用の言語 DNCL も動作する．

エラーへの対応については，長い命令を短く書けるライブラリや対になって使われる記号の自動入力，教員側からテンプレートファイルの配布といった機能を提供することで，エラーの発生を抑制することができると考えられる．また，エラーが発生したときには，エラーメッセージとエラーが発生した場所を学習者に通知することで，学習者が自力でエラーを直せるようなサポートを提供する．

表 5.4.1 既存研究で提供されている学習者への支援機能の提案環境との比較

各研究	日本語	テンプレート	興味を惹く題材	Web 環境	ステップ実行	エラー修正
PEN[18]	✓	✓			✓	
ますめ [19]			✓	✓	✓	✓
Alice[20]			✓			
ELP[21]		✓		✓		
PLM[22]		✓	✓			
CodeWrite[25]		✓		✓		
提案環境	✓	✓	✓	✓	✓	✓

表 5.4.2 には既存の研究で提供されていた教員への支援に用いられた情報と提案環境の比較を示す．提案手法は，現在実行しているファイル名から取り組んでいる課題を知ることができる．すべての実行のログを収集することで，ソースコードやそのエラー内容，前のコードとの差分などから学習者の学習履歴を知ることができる．実行単位のログ収集であっても，最終実行時刻を見ることで手が止まっている学習者を判別することができると考えられる．課題の達成度も，テストスイートを設定することにより実行結果を一覧できる．また，テストスイートが設定できないような課題についても，ソースコードなどのログを活用することで進捗の管理を支援することができる．

表 5.4.2 既存研究で提供されている教員への支援機能で利用されていた情報と提案環境との比較

各研究	利用する情報						ログ収集
	課題	テストスイート	ソースコード	エラー	差分	時刻	
tProgrEss[44]	✓	✓				✓	提出
vRoundEd[45]	✓	✓					提出
Hercules[46]	✓	✓					
IDISS[47]	✓		✓			✓	提出
C3PV[48]			✓[1]			✓	キー操作
加藤らのシステム [49]	✓			✓	✓[2]	✓	実行
市村らのシステム [50]				✓		✓	実行
ますめ [51]						✓	キー操作
PPV[52]			✓	✓		✓	キー操作
ClockIt[53]			✓		✓	✓	実行
CloudCode[23]			✓				キー操作
提案環境	✓	✓	✓	✓	✓	✓	実行

[1] 行数のみ利用

[2] 模範解答との比較

表 5.4.3 既存研究で普通科高校で行われていたセンサやデータ分析学習支援と提案環境との比較

各実践	組込み機器	データ蓄積	分析ツール	対象
間辺ら [42]	✓	△[1]	✓	高校
春日井 [31]			✓	高校
阿部 [32]		△[3]	✓	高校
渕 [43]	✓		✓	高校-大学
岡本 [35]			△[2]	高校
提案環境	✓	✓	✓	高校-大学

[1] ネットワークを通じたやりとりはしていない

[2] 表計算ソフトを利用

[3] 生徒ごとにアンケート調査やオープンデータの活用

普通科高校において行われていたデータの収集・整理・分析の支援については，表5.4.3に示すように,「収集」「整理」「分析」のそれぞれのフェーズを個別サポートするシステムの提案や，複数のツールを組み合わせて実践を行った例の報告はあったが，すべてのフェーズに渡って統合的，横断的に使えるプラットフォームは少ない．Bit Arrowを用いることによって，複数のツールを組み合わせなくても，これらの活動ができると考えられる．

以上の点から提案手法は，3.3で述べた課題である，学習者の自力でのエラー修正のサポートや，学習内容に依らない学習者の進捗の把握，データの収集・整理・分析の演習といった内容を含む高等学校の学習内容を網羅する，といった課題に対して有効であると期待される．

第6章　実装

本章では，5章で述べた設計に沿って実装したBit ArrowのIDE，言語処理系・ライブラリ，ユーザデータ管理機能，ログの収集・閲覧などの各機能の詳細について示す．

6.1　ソフトウェア構成

Bit Arrowサーバーのソフトウェア構成を図6.1.1に示す．Bit Arrowサーバーは Unix系OS(Ubuntu，CentOSなど）上にWebサーバを立て，Webアプリケーションとして動作している．システムプログラムは，サーバ側で動作する部分をPHP，Webブラウザにダウンロードされて動作する部分をJavaScriptで実装した．dataフォルダには，ユーザプログラム，素材ファイルが格納されている．データベースはMySQLを使用し，クラス・ユーザ情報，ログ，簡易DB，素材ファイルの管理情報（後述）が格納される．ユーザプログラム実行系は，ユーザプログラムをWebサーバ側で動作するための処理系とライブラリのセットである．現状はPythonのみに対応し，Pythonの処理系（Python3）と，ユーザプログラムから使用可能なライブラリを設置している．処理系に仮想環境を用いる場合はこの部分はDockerコンテナに格納される．

6.2　アカウント管理

Bit Arrowにおけるアカウントは5.2.1の図5.2.3に示した通り，管理者ユーザ，教員ユーザ，クラス，一般ユーザから構成されるが，実装上はMySQLの次のテーブルによって管理されている．

- **class** クラスの情報
- **user** ユーザの情報
- **teacher** 教員および管理者ユーザの情報
- **role** 教員ユーザの管理権限についての情報

このうち，roleテーブルには,「ある教員がどのクラスを管理可能か」という情報と,「ある教員が管理者ユーザか」という情報が格納されている．ある教員ユーザがクラスを作成した場合，そのクラスはその教員によって管理可能になるように自動的にroleのレコードが追加される．また，教員ユーザでログインした場合，教員メニューにはroleに登録された管理可能なクラスの一覧が表示されるようになっている．roleのレコードを手動追加すれば，複数の教員で同じクラスを共同で管理することも可能である．

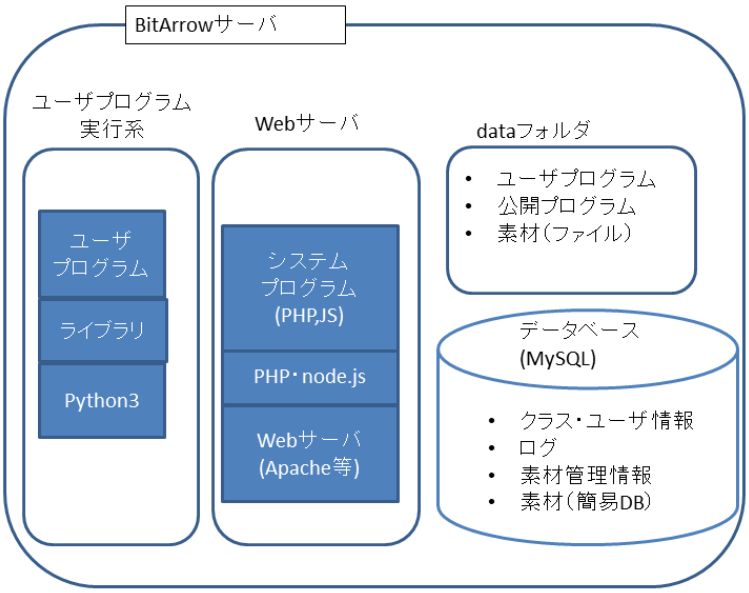

図 6.1.1 BitArrow のソフトウェア構成

クラス ID は教員によらず，システム全体で一意である（同じ ID のクラスを作成しようとするとエラーになる）が，ユーザ名はクラス内で一意であり，異なるクラスで同じユーザ名のユーザがいても異なるユーザとして扱われる[1]．

パスワードは，教員ユーザについては不可逆ハッシュ，一般ユーザについては可逆暗号（openssl_encrypt/openssl_decrypt）を用いて格納している．一般ユーザについては，所属するクラスを管理可能な教員ユーザによって，パスワードを確認できるようにするためである．

6.3 ユーザデータ管理

ユーザデータは「非公開プログラム」「公開プログラム」「素材ファイル」「簡易 DB」から構成される．このうち，簡易 DB 以外はファイルとして保存される．これらのファイルの保存場所を「`data` フォルダ」と呼ぶ．

`data` フォルダはさらに，非公開プログラムを格納する **home** フォルダと，公開プログラムおよび素材ファイルを格納する **pub** フォルダに分けられている．

6.3.1 home フォルダ

非公開プログラムは，**home** フォルダに，次のようなパスで保存されている．

data/home/クラス ID/ユーザ名/プロジェクト名/

home は Web サーバから直接参照はできず，ログインしたユーザのみが当該ユーザのフォ

[1] この一意性の違いを明示するため，クラスには「ID」，ユーザには「名」という接尾辞を用いている．

ルダにIDEを通じてアクセスできる．

6.3.2 pubフォルダ

一方，**pub**フォルダはWebサーバの設定により公開され，Bit Arrowのアカウントを持たないユーザでもURLを知っていれば誰でもアクセス可能である．pubフォルダの中はさらにサブフォルダに分けられ，素材ファイルや公開プログラムを格納する．このサブフォルダの名称は,「キー」と呼ばれる文字列をもとにユニークかつランダムに生成される．すなわち，あるキーkに対応するフォルダ名が初めて生成されるときには，すでに他のキーに対して生成されたフォルダ名とは異なるランダムな文字列fが生成され，それ以降はkに対応するフォルダ名はfで固定される．

「キー」は，次のように決定される．

- クラス共有の素材ファイル(classフォルダ)の場合，クラスID
- ユーザ個人の素材ファイル(userフォルダ)の場合，クラスID・ユーザ名の結合
- 公開プログラムの場合，クラスID・ユーザ名・プロジェクト名の結合[2]

フォルダ名とキーの対応関係は，MySQLデータベース内の素材ファイルの管理情報によって管理され，ユーザからは直接アクセスできないようにしている．これにより，サブフォルダの名称（ひいては，公開プログラムや素材ファイルのURL）だけからは，キーすなわちクラスIDやユーザ名を特定できないようになっている．これは，パスワードを必要としないクラスにおいて，クラスIDとユーザ名の組み合わせを他者が特定することにより，他者によるなりすましログインを防ぐためである．

dataフォルダの例を図6.3.1に示す．homeフォルダに，クラス「class_n」に属するユーザ「user_n」が作成したプロジェクト「project_n」のフォルダがあるとする．このフォルダ自身はWebサーバ上には公開されておらず，ユーザが公開の操作を行うことで，pubフォルダに実行可能な形式のファイル（この場合，C言語をトランスパイルしたJavaScriptファイルと，実行結果をWebページとして表示するためのHTMLファイル）が公開される．ただし，pubフォルダ上には「109e8a3」というサブフォルダに作成されるため，このフォルダ名を含んだURLからもとのクラス名やユーザ名は推測できない．また，このプログラムが素材ファイルを使用する場合は，公開プログラムのフォルダとは別のサブフォルダ「fd9fc9」に素材ファイルが置かれる.「109e8a3」が当該プロジェクト(class_n/user_n/project_n)のユーザプログラムであること,「fd9fc9」が同ユーザ(class_n/user_n)の素材ファイルのフォルダであることは，MySQLのテーブルで管理している．

素材ファイルへのアクセスをユーザプログラムから行うAPIは，現状拡張版JavaScript，簡易C，BA-Python，ドリトルにおいて提供されている．6.5にて後述する．

[2]各名前間の結合には，各名前で使用されていない文字を用いる

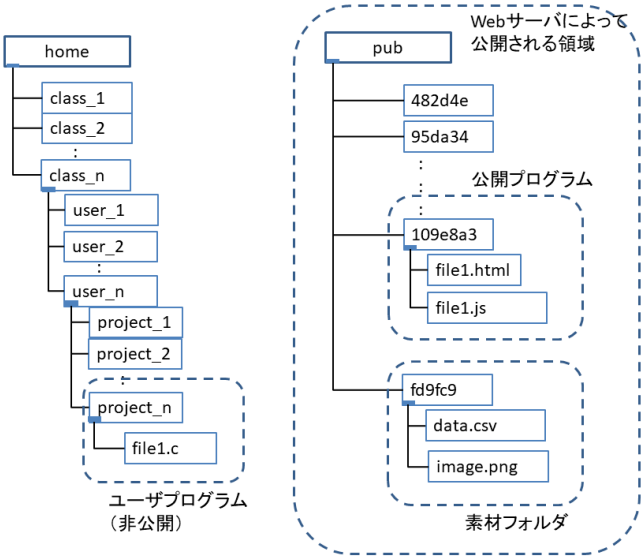

図 6.3.1 data フォルダの構成

6.3.3 簡易 DB

簡易 DB には，追記型と上書き型（KVS）があり，mySQL 内のテーブルとして保存される．追記型のスキーマは次のようになっている．

```
CREATE TABLE ‘bigdata‘ (
  ‘id‘ int(11) NOT NULL AUTO_INCREMENT,
  ‘time‘ int(11) DEFAULT NULL,
  ‘class‘ varchar(32) DEFAULT NULL,
  ‘group‘ varchar(32) DEFAULT NULL,
  ‘practice‘ varchar(32) DEFAULT NULL,
  ‘str1‘ varchar(50) DEFAULT NULL,
  ‘num1‘ double DEFAULT NULL,
  ‘str2‘ varchar(50) DEFAULT NULL,
  ‘num2‘ double DEFAULT NULL,
  ‘str3‘ varchar(50) DEFAULT NULL,
  ‘num3‘ double DEFAULT NULL,
  ‘str4‘ varchar(50) DEFAULT NULL,
  ‘num4‘ double DEFAULT NULL,
  PRIMARY KEY (‘id‘)
) DEFAULT CHARSET=utf8;
```

- **id** レコードの ID

- **time** 送信時刻
- **class** このレコードの属するクラス
- **group** グループ名
- **practice** テーブル名
- str_n, num_n 値．文字列または数値を 4 つまで格納可能．

上書き型のスキーマは次のようになっている．

```
CREATE TABLE ‘keyvalue‘ (
  ‘id‘ int(11) NOT NULL AUTO_INCREMENT,
  ‘class‘ varchar(32) DEFAULT NULL,
  ‘key‘ varchar(128) DEFAULT NULL,
  ‘value‘ mediumtext,
  ‘group‘ varchar(50) DEFAULT NULL,
  ‘time‘ int(11) DEFAULT NULL,
  PRIMARY KEY (‘id‘)
) DEFAULT CHARSET=utf8;
```

- **id** レコードの ID
- **class** このレコードの属するクラス
- **group** グループ名
- **key** キー名
- **value** 値

簡易 DB へのアクセスを行うための Web API を Bit Arrow に実装した．これを簡易 DBAPI と呼ぶ．簡易 DBAPI は，任意の HTTP クライアントから，Bit Arrow アクセスすることができる．

追記型のデータを追加する際のパラメータは，次のものになる．

- **group** グループ名
- **practice** テーブル名
- **data1...data4** 追加するデータの内容．4 個まで．送信時には文字列しか送信できないが，数値に見えるものは数値として格納される
- **url** (後述)

上書き型のデータを追加（更新）する際のパラメータは，次のものになる．

- **group** グループ名
- **key** キー名
- **value** 値．文字列のみ
- **url** (後述)

送信時のパラメータには指定されていないが，実際にデータが保存されるときに付与される情報として,「送信時刻」と「データの属するクラス」がある．データの属するクラスについては，Bit Arrow 上で開発中のプログラムからの送信であれば，ログインしているユーザの属するクラスをそのデータの属するクラスと見なすが，ログインしていない HTTP クライアント（組み込み機器など）からの送信の場合，パラメータ **url** に公開プログラムの URL を付与することで，その公開プログラムを作成したユーザのクラスに属するものと見なされる[3]．したがって，公開プログラムの URL を知っていれば，誰でもそのクラスに対してデータを送信可能である．ただし，前述した通り URL だけからはクラス ID は特定できないため，クラスの関係者でないユーザが特定のクラスを狙って送信することは困難である．

これらのデータを Bit Arrow のプログラムから操作する API(表 6.5.1) は，現状 Python および，拡張版 JavaScript で提供されている．addLog や putToServer でデータを追記する際には，url パラメータを呼び出し時に JavaScript の `location.href` を参照して付与するようにしている．このため，これらの API を使用する拡張版 JavaScript のプログラムを公開すれば，ログインしていない状態でも当該プログラムの製作者のクラスに対してデータを送付可能である．

6.4 IDE

IDE は，プロジェクトの作成，プロジェクト内のファイルの編集・実行，素材ファイルの管理などを行うページである．IDE はプロジェクトの作成・選択・削除を行う「プロジェクト一覧画面」（5 章の図 5.2.6）と，プロジェクト一覧画面で選択したプロジェクトを編集する「プロジェクト編集画面」（図 6.4.1）から構成される．

6.4.1 ローカルファイルシステム

Bit Arrow は，ネットワークの一時的な寸断があってもユーザプログラムを保存・実行できるように，Web ブラウザの localStorage を用いたファイルシステム（ローカルファイルシステム）をもっている．IDE で編集を行ったプログラムは，一旦ローカルファイルシステムに保存され，ネットワークに接続されている場合は Web サーバの非公開プログラムの保存領域と同期を行う．

[3]そのクラスで複数の公開プログラムが公開されている場合，どのプログラムの URL でもよい．公開プログラムではなく素材ファイルの URL でもよい．

図 6.4.1 プロジェクト編集画面

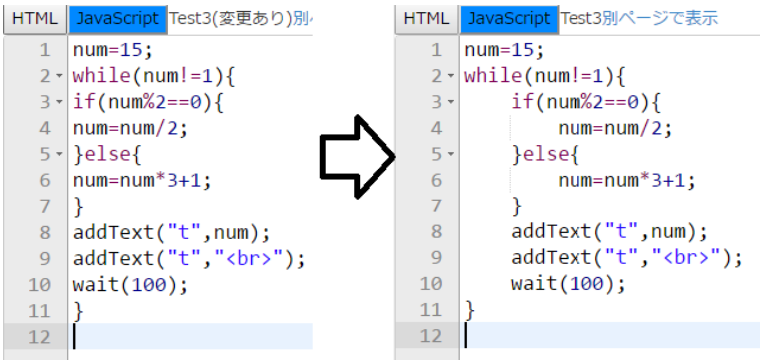

図 6.4.2 自動インデント

6.4.2 プログラムの編集

プロジェクト編集画面におけるユーザプログラムのエディタ部分には Ace editor [64] を採用した．Ace editor は対応する括弧や閉じタグの補完やインデンテーションなど，プログラミング言語ごとの入力補助を行うことができ，初学者が起こしがちなエラーを未然に防ぐ機能を備えている．ただし，Ace editor はインデンテーションについては，開き波括弧やコロン（Python の制御構造の始まり）を入力し，改行したタイミングでしか行わない．初学者の中にはせっかく付けたインデントを削除したり，余計に追加したりするなどの操作をすることがあり，適切にインデントを維持できないことが多いため，Bit Arrow では図 6.4.2 に示すように，実行または保存を行うタイミングでも，波括弧の個数をもとに自動的にインデントを修正するようにしている（ただし，Python は波括弧がないため修正を行わない）．

6.4.3 プログラムの実行

ユーザプログラムをブラウザ側で実行する際は，各言語のトランスパイラをブラウザ側で起動して，ユーザプログラムを JavaScript に変換し，その JavaScript を呼び出す Web ページを

生成する．

各トランスパイラはBuilderと呼ばれる統一されたインターフェースをもって開発されており，所定のプログラムフォルダに新しいBuilderを配置すれば新しい言語に対応可能となっている．プロジェクト編集画面を開いたときに，そのプロジェクトの言語に対応するBuilderが動的にロードされ，実行する際にはBuilderの所定のメソッドを呼び出すことで，必要なJavaScriptやHTMLのファイルを生成する．

生成されたプログラムをIDEのプロジェクト編集画面と同じ画面で行えるようにするため，生成したWebページは一旦，先述したローカルファイルシステムに保存し，動的に生成したiframe内で表示させている．このため，ブラウザ側でユーザプログラムを実行する際には，プログラムを実行するためのWebサーバとの通信は一切行う必要がない．

ユーザプログラムをサーバ側で実行する場合もBuilderを起動する．この場合Builderは，ユーザプログラムに対して必要なチェック作業（詳しくは後述）を行ってから，サーバにアップロードを行い，アップロードしたプログラムを呼び出すだけのWebページを生成する．

6.4.4 プログラムの公開

IDEで編集するユーザプログラムは基本的には非公開プログラムである．非公開プログラムを他のユーザから使えるよう，公開プログラムとして公開する場合，ブラウザ側で動作させるものでも，サーバ側で動作させるものでも，Builderが生成したWebページを`data/pub/`フォルダ内の特定の名前のサブフォルダ（名前の決定方法は6.3.2を参照）にアップロードする．`data/pub/`フォルダ内はWebサーバを通じて公開されているため，対応するURLを知っているユーザであれば誰でもアクセス可能となる．図5.2.9のように，URL示したQRコードを生成することもでき，実行結果をスマートフォンなどのPC以外のデバイスでも確認できる．

6.4.5 素材管理

5.2.3で述べた素材ファイル管理機能には，IDEから図6.4.3のようなダイアログ画面からアクセスできる．userフォルダかclassフォルダを選択して，ローカル（Bit Arrowにアクセスしているブラウザが動作しているPC）にあるファイルをアップロードできる．

また，Bit Arrow上で実行したプログラムの出力結果（printなどの標準出力命令で出力したもの）を素材ファイルとして送信する機能ももつ．次のような手順で送信可能である．

- Bit Arrow上でプログラムを実行する．
- プログラムを実行後，図6.4.4の実行画面ダイアログの下部にある「出力を共有……」を選択する．
- 「出力の共有」ダイアログが表示され，素材ファイルの送信先を選んでアップロードする

なお，図6.4.4のダイアログは，Bit Arrowでサポートするすべての言語の出力結果を送信可

図 6.4.3 素材ファイル管理ダイアログ

能であり，上記の形式で出力した結果はどの言語からでもアップロード可能である．また，同じダイアログから 6.6 で後述する ConnectDB にデータを送信することも可能である．

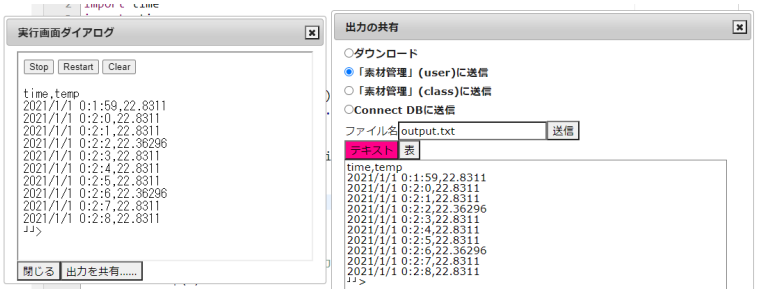

図 6.4.4 出力の素材ファイルへの送信

6.5 言語とライブラリ

ここでは，各言語のトランスパイラの実装や，対応する命令について示す．Python については Web サーバ側での実行機構の実装についても示す．

6.5.1 拡張版 JavaScript

拡張版 JavaScript の処理系はベースに Tonyu2[4] の処理系を用いている．Tonyu2 は，ゲームエンジンの Tonyu System 2[65] のスクリプト言語であり，基本的には JavaScript の文法に準拠しているが，`wait` メソッドなど，処理を途中で中断する命令を実行できるように，コードを変形して生成する．例えば，リスト 6.1 のコードはリスト 6.2 のように変換されて実行される．もともとのコードでは while 文の中で `wait` メソッドが呼び出されているが，変換されたコードにおいては while 文がなくなり，switch 文を用いて各ステップごとに分解されたコードとなる．これによって，処理の中断を可能にしている．

[4]https://github.com/hoge1e3/tonyu2-compiler

リスト 6.1 拡張版 JavaScript で記述したアニメーション

```
x=10;
while(true) {
    move("n", x, 50);
    x+=5;
    wait(50);
}
```

リスト 6.2 トランスパイラによって変換された JavaScript コード (一部)

```
      fiber$main :function _trc_Animation_f_main(_thread) {
        "use strict";
        var _this=this;
        var __pc=0;
        _this.x=10;
        _thread.enter(function _trc_Animation_ent_main(_thread) {
          if (_thread.lastEx) __pc=_thread.catchPC;
          for(var __cnt=100 ; __cnt--;) {
            switch (__pc) {
            case 0:
            case 1:
              _this.fiber$move(_thread, "n", _this.x, 50);
              __pc=2;return;
            case 2:
              _this.x+=5;
              _this.fiber$wait(_thread, 50);
              __pc=3;return;
            case 3:
              __pc=1;break;
            case 4 :
              _thread.exit(_this);return;
            }
          }
        });
      },
```

拡張版 JavaScript で使用できる命令を表 6.5.2 および表 6.5.3 に示す．DOM 操作，グラフィックス描画などの UI 構築に必要な命令や，シミュレーションに必要な数学関数，素材データにアクセスするための命令などを，ユーザプログラムから直接（オブジェクトの生成などの手続き不要で）行うことができる．ただし，グラフ描画についてはグラフオブジェクトを生成してから，そのグラフオブジェクトに対してメソッドを呼び出して描画処理を行う．グラフオブジェクトに対して使用できるメソッド表 6.5.4 に示す．グラフオブジェクトのバックエンドには JavaScript 用のグラフライブラリである Plotly.js[5]を使用した [66].

表 6.5.1 拡張版 JavaScript で提供しているデータの送信と取得に関するライブラリ

命令	動作
setGroup(グループ名)	下の命令で送信したり取得する対象のグループを設定する．省略すると”default” になる．
addLog(テーブル名, 値 1 [, 値 2[...[, 値 N]]])	テーブル名を指定して値を送信することができる．値は 1 から順に”data1”,”data2”,...,”dataN” として送信される．サーバへは追記型で保存される．
findLog(テーブル名,[値 1 [, 値 2[...[, 値 N]]]])	addLog で蓄積されたデータから指定したテーブル名と合致する追記型で保存されたデータを取得する．値が設定された場合はそれらの値を検索条件として条件に一致するものだけを取得する．
putToServer(キー, 値)	指定されたキーと値を送信する．サーバへは KVS（上書き型）で保存される．
getFromServer(キー)	指定されたキーと合致する KVS（上書き型）で保存されたデータを取得する．

6.5.2 Python

Python は，Web ブラウザでの実行と Web サーバでの実行を両方サポートしている．また，組み込み機器の Raspberry Pi Pico からの実行もサポートするが，これについては 6.7 にて述べ，ここでは，Web ブラウザ側，Web サーバ側それぞれでの実行系の実装を示す．なお，Web サーバ側の実行には仮想環境を使用する場合と使用しない場合で実装が異なるのでそれぞれ別の節にして述べる．

[5]https://plotly.com/javascript/

Web ブラウザ側での実行

BA-Python を Web ブラウザで実行させるため，他の言語と同様，Python から JavaScript へのトランスパイラを実装した．Python は JavaScript と比べると，動的な型付けのオブジェクト指向言語という共通点こそあるが，クラス定義やオブジェクトの動作については次のような点で違いが多い．

- メソッドの第 1 引数に，呼び出したオブジェクト (`self`) が渡される (JavaScript では `this` が引数経由ではなく暗黙的に渡される）
- 存在しないメンバーや配列要素にアクセスするとエラーになる（JavaScript は `undefined` を返す）
- 演算子の適用，関数呼出，メンバー・配列要素へのアクセスなど（ここでは「基本操作」と呼ぶ）を行う際の振る舞いオーバーロード可能（JavaScript では原則不可能）

これらの違いを吸収するために，Python 言語の補助ライブラリを JavaScript で実装した．これは，クラス定義の関数（第 1 引数に this を渡すように定義）を提供し，Python 特殊メソッド（`__add__`，`__getattr__`など，基本操作の振る舞いを定義する関数）を JavaScript の各種オブジェクトに定義するものである．また，ユーザプログラムからトランスパイルされる JavaScript では，`a+b` や `obj.x` などの基本操作を，これらの特殊メソッドを使った呼び出しに置き換えて出力し，JavaScript が本来提供している基本操作の機能を直接使わないようにしている．

Web サーバ側での実行（仮想環境不使用）

Web サーバ側で仮想環境を使わずに実行する際には，ユーザプログラムから Web サーバのリソースへのアクセスを制限させるため，ユーザプログラムから使用（import）できるライブラリは後述するラッパーライブラリのみになる．

動作の流れを図 6.5.1 に示す．まず，ブラウザ側で，エラーチェッカを用いて構文のエラーや未定義の変数や関数の使用などの意味的なエラーをチェックする．このとき，ラッパーライブラリ以外のライブラリを import している場合も，エラーとなる．エラーがなければサーバにユーザプログラムを送信する．

しかし，サーバに送信されるコードが，ブラウザ側でのエラーチェック（使用しているライブラリのチェックも含む）を行っていることは保証されない（任意の HTTP 接続ができるプログラムによるエラーチェックを行っていないコードの送信も可能）ため，サーバ側でも同様のチェックを行ってから，Python 処理系を使って実行される．実行する際には，Python 作業用フォルダ内のサブフォルダ（ユニークなフォルダを作成）を用いる．サブフォルダ内にユーザプログラムをコピーし，ユーザの素材ファイル（user および class）のフォルダへのシンボリックリンクを張った状態で実行を行う．

エラーチェッカはすべて JavaScript で実装されており，ブラウザ側ではブラウザの JavaScript 処理系を用い，サーバ側では node.js を用いて動かしており，コードの内容はほとんど同一である．

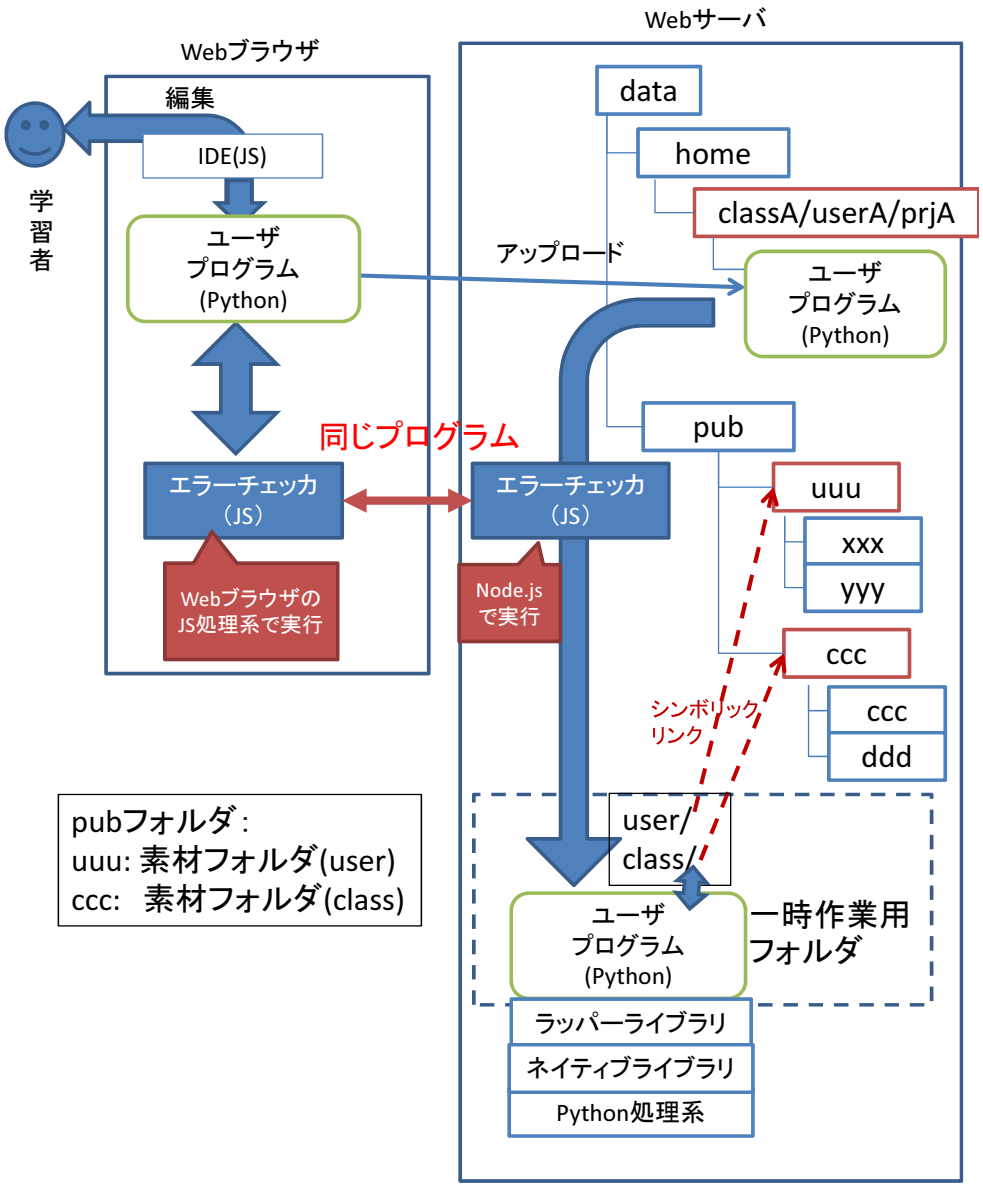

図 6.5.1 BA-Python のサーバ側での実行（仮想環境不使用）

さらに，実行のリクエストが集中してサーバの負荷が上がるのを防ぐため，一定数の Python 処理系のプロセスが起動していたら実行を中止させる．また，すでに起動している Python 処理系のプロセスが一定時間経過しても終了しない場合は，強制終了させる仕組みも用意されている．

ラッパーライブラリは，Python 処理系に用意されているライブラリ (ネイティブライブラ

リ) の中から，一部の機能（OS 等のリソースを破壊する危険がないと認められるもの）を選んで呼び出せるようにしたものである．ユーザプログラムから import を記述する際には，一般的な Python 処理系で使われているライブラリと同じ名前を記述すればよいが，実行する際に実際に読み込まれるライブラリの実体はラッパーライブラリであり，そのラッパーライブラリから実際のネイティブライブラリを呼び出している．

現在，ラッパーライブラリがサポートする関数・クラスは次のものになる．

- 組み込み関数は range, input, str, int, float, len, type, quit, exit, sorted が使用できる．
  - print については，Python2 と Python3 で記法が異なるが、いずれの記法でも動作するようになっている（図 6.5.2 参照）．
  - input については，ブラウザ側で実行した場合は input を呼び出すたびにブラウザ標準のダイアログで入力をさせる．サーバ側で実行する場合は入力される予定の値を実行前にすべて入力しておく必要がある（したがって，input が何回呼ばれるか事前にわからないプログラムは正しく実行できない）．
- ライブラリは，表 6.5.5 に挙げたライブラリをサポートする．

```
print("hello") #python 2,3で共通
print "hello" #python2 のみ
print "hello", #python2 のみ．改行なし
print("hello",end="") #python3  改行なし
```

図 6.5.2 print の記述方法

また，いくつかのメソッドにおいては，ネイティブライブラリの入出力を加工するものがある．

- ファイル操作系のメソッド (open や matplotlib の save など) は，アクセス先のパスをチェックし，素材ファイルのフォルダ（user フォルダおよび class フォルダ）以外へのアクセスを行おうとするとエラーにするようにしている．
- matplotlib におけるグラフ表示命令 (pyplot.show) は，通常はサーバ側のグラフィックス画面 (X-window サーバなど）へ表示を行うが，Bit Arrow においては結果をブラウザに表示させる必要があり，出力を一旦画像ファイルに変換して，出力結果の中に img タグで埋め込んで表示する．
- folium における地図表示は，通常は地図を表示する html ファイルを生成するが，Bit Arrow で表示する際には，生成した HTML を iframe タグで埋め込んで表示する．なお，表示のために，ラッパーライブラリの folium.Map オブジェクトに，ネィティブライブラリには存在しない show というメソッドを追加している．

既存のライブラリの他に Bit Arrow が提供する機能にアクセスするために作成したライブラリとして，badb と cdb がある．cdb については 6.6 にて後述し，ここでは badb について述べる．これは簡易 DB へのアクセスライブラリであり，上書き型には badb.getFromServer(キー），badb.putToServer(キー, 値）によって読み書きを，追記型には，badb.addLog(テーブル名，値，値，…），badb.findLog(テーブル名，検索値，検索値，…），によって追記と読み出しを提供する．

Web サーバ側での実行（仮想環境使用）

仮想環境を使用する場合の動作を図 6.5.3 に示す．仮想環境には Docker を使用している．Docker は，コンテナと呼ばれる仮想的なシステムを複数起動可能であり，1 つのクラスごとに 1 つのコンテナを起動する．あるクラスで Python プログラムのサーバ側での実行要求があった段階で，そのクラスを担当する Docker のコンテナが起動され，以降はそのクラスの Python プログラムはすでに起動しているコンテナで実行する．他のクラスでも同様に実行要求があった場合は，そのクラスのための別のコンテナを起動するが，コンテナのイメージ（コンテナの OS やデータを定義したファイル）はクラスによらず同じものである．各コンテナには Python の処理系とネイティブライブラリがインストールされており，ラッパーライブラリは使用しない．

仮想環境を使用する場合，コンテナは，サーバ本体（ホスト）側にある非公開プログラムのフォルダ内の，コンテナが担当するクラスに対応するサブフォルダをボリュームマウントし，その中に書かれている Python プログラムを，コンテナ内の Python 処理系で実行する．

また，仮想環境を利用する場合は，素材ファイルのフォルダ構成が先述したものと少し異なる．まず，pub フォルダの直下にクラス ID から生成されたキーでサブフォルダが作成され（図 6.5.3 の aaa)，その中に class フォルダ (ccc，クラス ID から生成されるキーだが，aaa とは異なるものが生成される) と，各ユーザの user フォルダ (uuu，クラス ID とユーザ名から生成されるキー) が生成される．aaa フォルダをボリュームマウントすることにより，Docker コンテナからそのクラスのすべての素材フォルダ（class フォルダおよび，各ユーザの user フォルダ）を参照させつつ，他のクラスの素材フォルダへのアクセスができないようにしている（ただし，同じクラス間の他のユーザへのフォルダはアクセス可能になる）．実行前に，実行するユーザプログラムの所有者の user フォルダおよび class フォルダへのシンボリックリンクが作成されるので，open などの命令を使用する際にはパス名に"user/"または"class/を付与してアクセスを行ことができる．

また，matplotlib におけるグラフ表示命令 (pyplot.show) は，仮想環境不使用のときと同様，ブラウザに画像としてグラフを表示できるように，matplotlib.backends を用いて動作を変更している．

なお，仮想環境を利用しない場合に提供しているライブラリ（6.5.2 参照）の一部は，仮想環境を利用する場合には現状対応していないものがある．例えば，地図ライブラリの folium，Bit Arrow 独自のライブラリである badb，cdb などは現状利用できない．

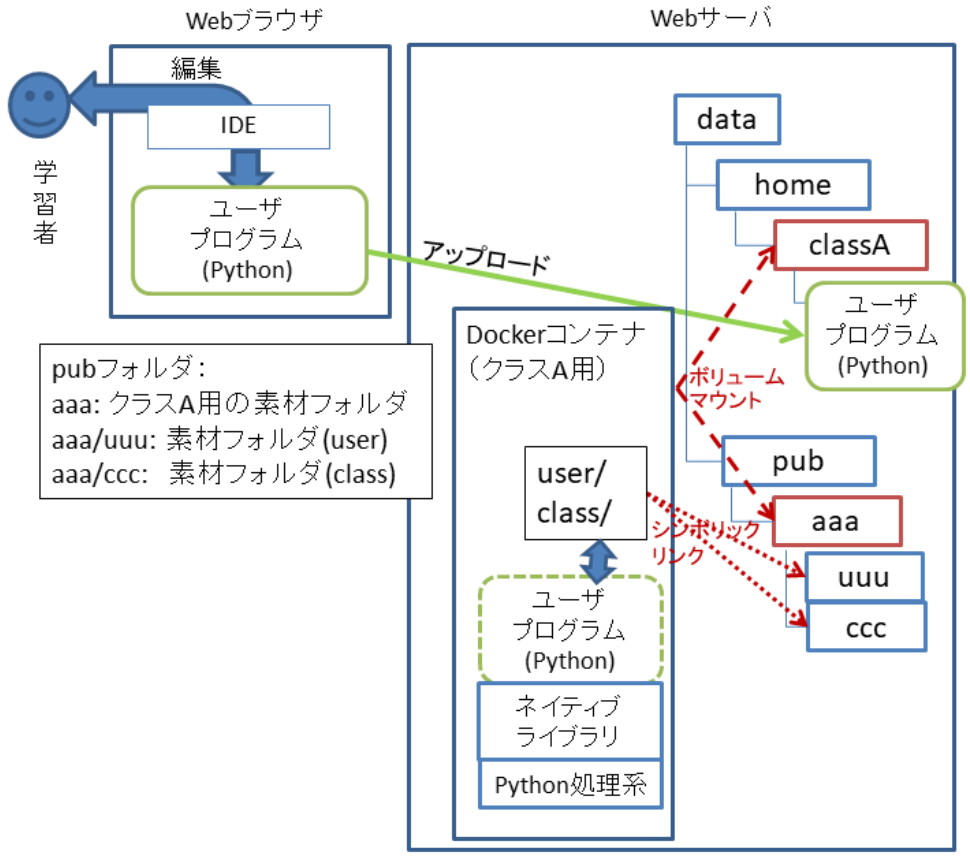

図 6.5.3 BA-Python のサーバ側での実行（仮想環境使用）

6.5.3 ドリトル

ドリトルのプログラムの例をリスト 6.3 に示し，ドリトルの言語としての特徴についていくつか述べ，JavaScript にトランスパイルする際の注意点を示す.

- 関数オブジェクト（ドリトルでは「ブロック」と呼ぶ）に対して「実行」「繰り返す」「なら」などのメソッドを用いて，繰り返しや条件分岐などの制御構造を実現する仕組みがある．2 行目のように，ブロックに対して「繰り返す」というメソッドを発行することで繰り返しを行える.

- オブジェクトに対してブロックが値となるようなプロパティを付与することでそのオブジェクトにメソッドを追加できる．3 行目のように,「かめた」に対して「三角形」というメソッドを定義できる.

- オブジェクトに対して「作る」というメソッドを発行すると，そのオブジェクトを「親」として別の「子」となるオブジェクトを作成する．子は親のもつメソッドをすべて使う

ことができる．8行目のように,「かめた」から作られた「かめこ」も「三角形」メソッドが使用できる．

リスト 6.3 ドリトルのプログラム

```
1 かめた＝タートル!作る.
2 「かめた!１００ 歩く １２０ 左回り」!３ 繰り返す.
3 かめた:三角形＝「|長さ|
4     「!（長さ）歩く １２０ 左回り」! ３ 繰り返す
5 」.
6 かめた! 50 三角形.
7 かめこ＝かめた!作る ６０ 右回り.
8 かめこ! 150 三角形.
```

ドリトルは従来，Java アプリケーションとして実装されていたが，ドリトルを Bit Arrow 上で動作させるため，JavaScript に変換するためのトランスパイラが提供された [67] [68].

ドリトルも JavaScript も，ともにプロトタイプ方式のオブジェクト指向言語であることから，比較的簡単に変換が可能である．ただし，変数へのアクセス方法に次のような違いがある．

- JavaScript では，ある名前の変数にアクセスする際，まず引数およびローカル変数の宣言に指定の名前があればそれを参照し，ない場合はグローバル変数を参照する．また，メソッド内における現在の（メソッド呼出で対象となっている）オブジェクトのプロパティへのアクセスは必ず `this` をつける必要がある．

- ドリトルでは JavaScript 同様，まず引数およびローカル変数を参照するが，該当する名前がそれらの宣言にない場合，現在のオブジェクトのプロパティを参照し，それもない場合は現在のオブジェクトの親を順番に辿る．最終的には「ルート」と呼ばれるオブジェクトにたどり着く.「ルート」は，あらゆるオブジェクトの親であるとともに，グローバル変数としても機能する．この仕様のため，現在のオブジェクトのプロパティへのアクセスにおいて `this` は省略可能である（「自分」という変数を明示的に使うことも可能）．

これらの差異を吸収するため，JavaScript を生成する際には，引数やローカル変数以外の変数に対しては必ず `this` を介してアクセスするようにしている．また，ブロックを制御構造として使う場合と，メソッドとして使う場合それぞれの状況に対応させるための変換も行っている．

リスト 6.3 を JavaScript に変換したプログラムをリスト 6.4 に示す．引数（「三角形」メソッドにおける「長さ」）以外の変数は `this` を使ってアクセスされていることがわかる．プログラムが開始されると，まず，`this` を「ルート」にして関数を呼び出す．このため「かめた」「かめこ」などは「ルート」のプロパティとして登録される．また，ブロックオブジェクトを JavaScript の関数オブジェクトで表現する際には，`dtlbind` という組み込み関数を使って生成する．`dtlbind` は，関数オブジェクトに対して先述した「実行」「繰り返す」「なら」などのメソッドが呼び出された場合に，第一引数のオブジェクト（ここでは `this`，すなわち「ルート」）を `this` として呼び出すように変換をかけるものである．ただし，その関数がメソッドとして

呼ばれた場合（例：this[’かめた’][’三角形’]();）は，そのメソッドの呼出対象となるオブジェクトがthisになるように呼ばれるようにしている．

リスト6.3の例では，2行目のブロックはthisが「ルート」になるように呼び出されるが，3行目から5行目のブロックは，thisが対象となるオブジェクト（「かめた」または「かめこ」）になるように呼び出される．

リスト 6.4 ドリトルのプログラムを JavaScript に変換した例 (一部コード片抜粋)

```
1  root.system.run(function() {
2      this['かめた'] = this['タートル']['作る']();
3      dtlbind(this, function() {
4          return this['かめた']['歩く']((100))['左回り']((120));
5      })['繰り返す']((3));
6      this['かめた']['三角形'] = dtlbind(this, function(長さ) {
7          return dtlbind(this, function() {
8              return this['歩く']((長さ))['左回り']((120));
9          })['繰り返す']((3));
10     });
11     this['かめた']['三角形']();
12     this['かめこ'] = this['かめた']['作る']()['右回り']((60));
13     return this['かめこ']['三角形']();
14 });
```

また，Bit Arrow上で動作するドリトルは，Javaアプリケーション版にはない次のような機能が新たに使えるようになっている．

- 「通信 (Ajax)」オブジェクト．Web APIの呼出.「データ」「読む」「書く」などのメソッドを提供する．
- 「素材 (Assets)」オブジェクト．素材ファイルへのアクセス.「一覧」「読む」「書く」などのメソッドを提供する．
- 「加速度センサ (accelerationSensor)」「ジャイロセンサ (gyroSensor)」オブジェクト．各種センサへのアクセスメソッドを提供する．
- 「ラズパイ」オブジェクト．Raspberry Pi Picoとの接続を行い，GPIOの入出力を行う．内蔵されている温度センサ，LEDについては「温度」「LED」などの簡便な命令を用意する．
- 「タッチセンサ (touchSensor)]」オブジェクト．画面タッチの検出の有無や座標を取得するメソッドを提供する．

6.5.4 DNCL(どんくり)

DNCLの処理系は，上で述べたドリトルの処理系をベースにして，当初はBit Arrowとは別のスタンドアロンなWebアプリケーションとして開発された．後にDNCLの処理系部分と

Bit Arrow とを接続する Builder を実装し，Bit Arrow 上で動作するようにした．処理系の詳細は処理系開発者の文献 [69] を参照されたい．

6.5.5 C

変換規則

C 言語を JavaScript に変換する際には，JavaScript と C 言語で似ている点についてはなるべくそのまま変換し，異なる点についてその差異を吸収するような変換を行うよう設計した．[58]

- **制御構造**

  JavaScript は C 言語の文法を踏襲しているため，式文，条件分岐，繰り返し，関数などの制御構造はそのまま JavaScript でも同様に扱えるものが多く，それらについては対応する同じ要素に変換している．

- **数値の型**

  C 言語が扱う数値には整数，実数などの区別や，符号の有無やビット長の区別などがあるが，JavaScript で扱う数値は Number 型という 1 種類しかない．このため，コンパイル時に検出した型の情報を用いて，型の違うもの同士での演算が発生した際に変換操作を行なうようにしている．これにより，整数同士で割り算をした場合に余りの部分を無視して整数の結果を得るなど（JavaScript では小数点つきの値になる）C 言語本来の挙動に近い動作を実現している．

- **変数とポインタ**

  C 言語においては，任意の変数についてそのポインタを取得することができる．一方，JavaScript はポインタと同等の機能は直接用意されていないため，オブジェクト[6]への参照を用いてポインタ相当の機能を再現している．変換後の JavaScript においては，関数（正確にはブロック）ごとに変数を「変数オブジェクト」というデータ構造に格納しており，ポインタが必要な場合は「ポインタオブジェクト」を生成する．ポインタオブジェクトは，ポインタが指し示す対象の変数に対する操作（読み取りや書き込み）機能を提供する．図 6.5.4 に例を示す．`main` 関数のローカル変数は `v_m` という変数オブジェクトに格納される（生成後の JS2 行目から 4 行目）．このうち変数 p についてはポインタオブジェクトを生成・参照し（4 行目），変数 a に対する操作を提供する．これを関数 f に渡して，f から変数 a の値を書き換える（10 行目）ことが可能である．

- **配列**

  C 言語における配列とは，メモリ空間のある領域を指すポインタであるが，JavaScript にはメモリ空間を直接扱う機能はなく，代わりに動的に確保可能な配列オブジェクトが

[6] JavaScript のオブジェクトの実体はキー（文字列）と値の対応づけであり，どのオブジェクトに対しても任意のキーと値とを複数格納可能である．また，キーと同名の文字列を用いて任意の値を読み書きできる．

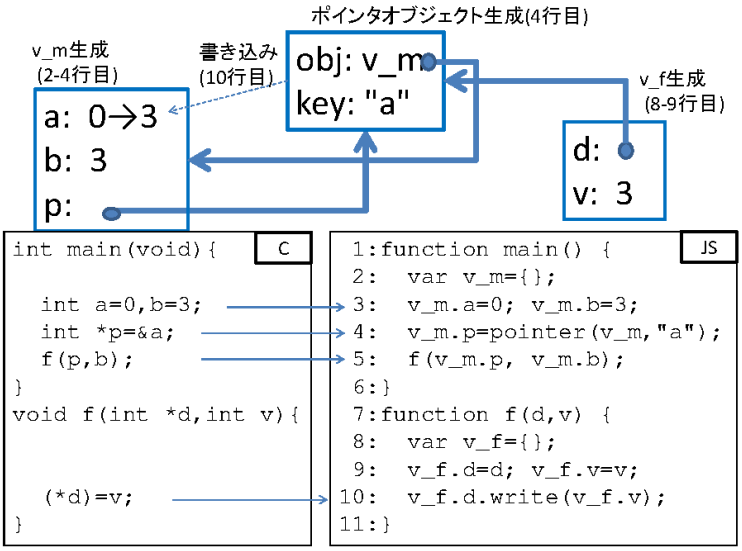

図 6.5.4 簡易 C におけるポインタの表現

利用できるため，これで配列を再現する．前述のポインタオブジェクトは配列の要素を指し示すことも可能であり，特定の要素をポインタを用いて読み書きすることや，そのポインタから指定されたオフセット分前後した要素を指し示すポインタを生成すること（本稿ではこれを「相対アドレッシング」と呼ぶ）も可能である．

- **構造体の値渡し**

  簡易 C では構造体を JavaScript のオブジェクトで表現するが，C 言語の構造体はそれ自身が値であり，他の変数へ代入したり，関数に渡したりするときにはコピーが発生する（値渡し）．一方，JavaScript のオブジェクトは参照であり，代入したりや関数に渡したりしたときにコピーが発生しない（参照渡し）．これらの差異を吸収するため，コンパイル時に型を判定し，構造体同士の値の受け渡しをする際には事前にコピーを行なうようにしている．

- **同期処理の実行**

  C 言語は，処理中に I/O などで待ち時間が発生した場合に，任意の箇所で処理を中断し，入力などのイベントがあるまで待機した後に処理を再開するという，いわゆる同期処理を行なうことができる（`scanf` など）．一方，JavaScript はイベントの処理を非同期に行なう．すなわち，イベントに対するイベントハンドラを定義して，イベント発生時にイベントハンドラの呼出を行なう．さらに，イベントハンドラの処理中に待機をし，他のイベントを受け取ってから処理を再開する仕組みを基本的にはもっていない．そこで，簡易 C では一部の Web ブラウザに搭載されているジェネレータ[7] の仕組みを利用して，

[7] https://developer.mozilla.org/ja/docs/Web/JavaScript/ Guide/Iterators_and_Generators

1つの関数を複数のイベントに渡って呼び出せるようにしている．ただし現時点（2017年5月）では非対応のWebブラウザも残っているため，それらのWebブラウザにおいては代替の命令（例えば，scanfはJavaScriptのprompt関数で代用）を用意したり，プログラムによっては正しく動作しなかったり（後述のupdateなど）する場合もある．

初学者へのサポート機能

簡易Cでは，一般的なC言語との互換性を保つだけでなく，初学者がつまずきやすい点に関しては独自にサポートを行なうようにした．

- **メモリの範囲外へのアクセス対策**

  C言語のポインタは，相対アドレッシングを用いることで任意のメモリ領域にアクセスができるが，使い方を誤ると，正規に確保されたメモリ領域以外へのアクセスを行ってしまい，思わぬ動作を引き起こす．初学者にはその原因を特定するのが困難なことが多い．

  そのため，簡易Cでは相対アドレッシングを利用できるポインタは配列へのポインタだけに限定している[8]．つまり，単純変数や構造体のメンバへのポインタは，生成はできても相対アドレッシングをしようとするとエラーが発生するようにしている．例えば図6.5.5のプログラムはエラーになる．

```
typedef struct {
    int x,y;
} Vec2;
int main(void) {
    Vec2 v;
    int *pv=&v.x;
    *pv=3;
    pv++;//エラー（&v.yを指すことはできない）
    *pv=5;
}
```

図 6.5.5 相対アドレッシングが禁止される例

  相対アドレッシングが可能な配列へのポインタについても，JavaScriptの配列オブジェクトが実行時に要素数の情報を保持していることを利用して，相対アドレッシングによって配列の範囲外を指すポインタオブジェクトが生成され，それを使ったアクセスが行われた場合は実行を中断し，学習者に注意を促すようにしている．

- **無限ループ対策**

  学習者が無限ループを起こすようなプログラムを書いた場合，Webブラウザが固まってしまい演習を続けることができなくなる場合がある．これを防止するため，実行されて

[8]動的メモリ確保には未対応 (現状の簡易Cは動的メモリ確保の仕組み自体が未実装)

からの経過時間をチェックし，5 秒以上経過すると自動的に実行を停止するようにしてある．このチェックを行なうための処理を，繰り返し構造のたびに挟み込んである．なお，前述のような途中で処理を中断し，次のイベントを待つ処理が実行された場合は，待っている時間は経過時間をカウントしないようにしてある．

- **初期化忘れ対策**

  C 言語は，ローカル変数に対して明示的に初期化を行わない場合の変数値は保証されておらず，いわゆる「ゴミ」の値が入っている．C 言語の動作を正しく学ばせるために，簡易 C はこの動作も再現しているが，初学者にとってはゴミの値が表示されることで混乱を来す可能性があるため，代入や演算の処理中にゴミの値を検出した場合は変数が初期化されていない旨の警告を行なうようにしてある．

標準関数

現在，一般的な C 言語と共通して簡易 C で使える関数は次のものになる．

- 標準入出力（stdio.h）は `printf` と `scanf` と `fopen` に対応している．`printf` のフォーマット指定は`%f`, `%d`, `%c`, `%s` に対応し，桁数指定子には未対応．`scanf` のフォーマット指定は`%f`, `%d`, `%c`, `%s` に対応し，桁数指定子には未対応．変数は 1 つだけ指定できる．`fopen` は，ファイルパスに `user` および `class` を指定すると，素材ファイルへのアクセスになる．それ以外のパスは，ブラウザの localStorage へのアクセスになり，他のブラウザとは共有されない．

- 数値関数（math.h）は次の関数に対応している：`abs`, `acos`, `asin`, `atan`, `atan2`, `ceil`, `cos`, `exp`, `floor`, `log`, `max`, `min`, `pow`, `round`, `sin`, `sqrt`, `tan`

- 文字列関数（string.h）は次の関数に対応している：`strlen`, `strcpy`, `strncpy`, `strcmp`, `strncmp`, `strcat`, `strncat`, `memset`, `index`, `rindex`, `memcmp`, `memcpy`, `strstr`

- 標準ライブラリでは (stdlib.h) は `rand` が利用できる．`srand` は用意されていないが，乱数系列の初期化は自動的に行われる．

なお，これらの命令を利用するためには，対応するヘッダファイルを`#include<stdio.h>` などの形式でインクルードしないとエラーになる（一部の C コンパイラでは警告が出るだけの場合もあるが，簡易 C では厳密に書かせるようにしている）．

グラフィックスライブラリ

簡易 C は標準関数のほかにグラフィックスやアニメーションの制作を支援するライブラリが用意されている．これは，HTML5 の Canvas 機能を利用して図形の描画を行なうものである．表 6.5.6 にグラフィックスライブラリの命令を示す．なお，これらの命令を利用するに

は#include<x.h>というインクルードが必要になる．図 6.5.6 にグラフィックスライブラリの使用例を示す．このプログラムでは，四角形が右に移動するアニメーションを表示している．

```
#include<x.h>
int main(void) {
    int x,y;
    setColor(255,0,0);
    x=30; y=50;
    while (x<300) {
       clear();
       fillRect(x,y , 50, 50);
       x+=2;
       update();
    }
}
```

図 6.5.6 アニメーションの例

6.6 Web API・外部データベース連携

5.2.6 において，Web API の呼出は Bit Arrow のサーバ内で一旦リクエストを受け付け，サーバが呼び出すことを許可した Web API のみを呼び出せるようにすると述べたが，現在の実装でこの方式で呼び出せる API は，教育用データベースの ConnectDB[70] である．ConnectDB は次のような特徴をもつ [61].

- PC やスマートフォンの Web ブラウザで動作する
- ログインなしでも，サンプルデータや計測データを利用可能．ログインするとでデータをクラウド上に保存可能
- ログインすることで，クラス単位でのユーザ管理，教員によるデータの配布や共有に対応
- グラフ作成が簡単に可能
- スマートフォンの内蔵センサ（加速度・ジャイロ・GPS）を利用してセンサデータを作成可能
- Web API が提供されている．テーブルごとに API キーが割り当てられ，API キーを知っていれば認証手続不要でデータの追記と読み出しができる．

ConnectDB へのアクセスは，現状の実装では，次の言語に向けてライブラリを提供している．

- Python では，cdb というパッケージを提供し，cdb.post(API キー, 値) , cdb.get(API キー），によって読み書きができる

- 拡張版 JavaScript においては，`addCDB(API キー, 値)` , `findCDB(API キー)` ，によって読み書きができる．

また，5.2.3 で述べた,「出力の共有」機能において，実行結果を送信する先として，Bit Arrow が管理する素材ファイルや簡易 DB のほかに，ConnectDB[70] にも API キーを指定して送信を行うことができる．

ConnectDB 以外の Web API については，Python のサーバ実行において (仮想環境使用，不使用両方)，request モジュールを使用して通信を行うことで，各種 Web API を提供するサイトに対してアクセスが可能である．現状は，利用者が多くないこともあり使用できるサイトの制限や，発行可能回数などの設定はしていない．

6.7 組み込み機器接続

組み込み機器からのセンサデータの収集は，HTTP 通信が可能な組み込み機器については，6.3.3 で述べた簡易 DBAPI を利用して行うことができる．

また，HTTP 通信ができない組み込み機器については，Bit Arrow が動作している PC に USB ケーブルで接続を行い，Web ブラウザの JavaScript プログラムから USB 機器にアクセスする手法として策定されている WebUSB[9]を用いて操作ができるようにした．現段階での Bit Arrow の実装では，Raspberry Pi Pico を操作できるようにしている [61]．また，研究室の学生の協力により，micro:bit[71] でも動作可能であることを試験的に確認しているが，必要なライブラリの整備をしている途中である．

Raspberry Pi Pico は，PC から USB シリアル接続で MicroPython のプログラムを書き込んで実行することができ，実行結果（標準出力・標準エラー出力の内容）はリアルタイムに USB シリアル経由で送信される．また，REPL(Read-eval-print-loop) 環境を備えているため，短いプログラムを送信して，実行結果をその都度確認することもできる．これを利用して，Web ブラウザで動作しているプログラムから，センサやアクチュエータのデータを一定間隔（数秒に 1 回程度）で送受信することも可能である．したがって，5.2.5 で述べた機器内実行，ブラウザ内実行いずれの方法でも動作させることができる．次に，それぞれの実行方式の実装について述べる．

6.7.1 機器内実行

Bit Arrow の Python プロジェクトにおいて，IDE で作成した Raspberry Pi Pico 向けの MicroPython プログラムを実行し，IDE の実行画面ダイアログに Raspberry Pi Pico からの実行結果を表示することで機器内実行方式に対応する．

IDE で Python プロジェクトを開くと,「Raspberry Pi Pico 接続」というメニューが選択可能になっている．Raspberry Pi Pico をローカル PC の USB の接続した状態でそのメニューを使用すると，Web ブラウザによる USB 接続確認を承認したのち，Raspberry Pi Pico と接続され

[9]https://wicg.github.io/webusb/

た状態になる．この状態で，ユーザプログラムの実行を行おうとすると，図 5.2.12 で示したような実行先を選択するメニューにおいて,「Raspberry Pi Pico で実行」というサブメニューが追加される．このサブメニューを選ぶと，現在開いているユーザプログラムを Raspberry Pi Pico に転送し実行する，実行ダイアログには，USB 経由で送られてくる Raspberry Pi Pico の実行結果（エラーメッセージも含む）をリアルタイムに確認することができる．

6.7.2 ブラウザ内実行

拡張版 JavaScript，ドリトルのライブラリにおいて，Raspberry Pi Pico の各種 GPIO ピンの入出力を行う関数を用意し，関数を呼び出すたびに Raspberry Pi Pico へ短いプログラムを送信してセンサやアクチュエータを操作することができる．

6.8 ログの収集・閲覧

6.8.1 ログ収集機能・ログデータ

IDE は，ユーザの次の操作を検知すると，ログをサーバに送信する．なお，カッコ内は後述する result フィールドに格納される識別子である．

- プログラムを保存したとき (Save)
- プログラムの実行を指示したとき (Build)
- 実行しようとしたプログラムにコンパイルエラーがあったとき (Compile Error)
- プログラムが正しく実行され，実行結果が返却されたとき (Run)
- プログラムが実行されたが，実行時エラーがあったとき (Runtime Error)
- プログラムの書き換えを行い，一定時間保存操作がなかったとき (Unsaved)
- その他，名前の変更 (Rename)，削除 (Remove) など

上記のタイミングで収集される１つ１つのデータを「ログ要素」と呼ぶ．ログ要素は mySQL のテーブルに次のようなスキーマで保存される．

```
CREATE TABLE ‘log‘ (
  ‘id‘ int(11) NOT NULL AUTO_INCREMENT,
  ‘time‘ int(11) DEFAULT NULL,
  ‘class‘ varchar(32) DEFAULT NULL,
  ‘user‘ varchar(32) DEFAULT NULL,
  ‘lang‘ varchar(10) DEFAULT NULL,
```

```
  `filename` varchar(255) DEFAULT NULL,
  `result` varchar(40) DEFAULT NULL,
  `detail` text,
  `raw` text,
)
```

- **id** ログ自身の ID
- **time** 時刻 (UTC 秒)
- **class** 操作したユーザのクラス ID
- **user** 操作したユーザのユーザ名
- **lang** プロジェクトの言語
- **filename** ファイルパス (クラス ID/ユーザ名/プロジェクト名/ファイル名)
- **result** 言語名+上記識別子
- **detail** エラーの場合はエラーメッセージ
- **raw** ソースコードを表す JSON．キーは「HTML」とプロジェクトの言語（”js” や ”python” など）.「HTML」は拡張版 JavaScript の場合のみ使用され，各プログラムの HTML 部分が保存される．

6.8.2 ログ閲覧画面

実装した学習者の状況を把握する画面を図 6.8.1 に示す．学習者の状況把握画面では，学習者ごとに，指定された時間内の実行数とエラー数，前回実行してから今までの経過時間を表示させる．最近 30 分，60 分，90 分の期間をボタン一つで選択できるほか，任意の期間を選択することもできる．また，一番最近実行したファイル名も提示することで，他の学習者に比べて作業が遅れている学習者も見つけることができる．苦労している学習者の情報を分かりやすく提供するため，エラーの発生率が高い学習者や前回の実行からの経過時間が長い学習者は表に色をつけて表示させている．表示される情報は，エラーの発生率，実行からの経過時間でソートすることもできる．図 6.8.1 で示した例では，このページを読み込んだ時点での実行からの経過時間が長い順に並べ替えられており，手が止まっている学習者を把握することができる．また，時間内の実行結果履歴をエラーが発生していたら E，エラーが出なければ R，保存を行ったときには S と表示させ，それをクリックすることでソースコードや実行結果，エラーメッセージなどの詳細を確認できるようにした．

図 6.8.1 の画面から，学習者を 1 人選ぶことで，図 6.8.2 の学習者個人の実行状況把握画面に移動できる．この画面では，左側に指定された（全体の画面で見ている範囲の）日付の ログ要素が一覧表示される．

2018/05/18 17:30:00 から 2018/05/18 17:40:00までの実行状況 集計....

ユーザID	エラー/実行	実行からの経過時間	今実行しているファイル	実行結果履歴
user19	0/1(0%)	00:07:13	0518/P0518_6.c	R
user58	0/0(--%)	00:06:40	0518/P0518_6.c	S
user12	2/3(66%)	00:06:31	Test/Test.c	ESERS
user33	0/8(0%)	00:05:10	0518/P0518_5.c	RRSSRRRRSRRS
user38	3/9(33%)	00:04:15	0518/P0518_6.c	RRSESRRSRRSESES
user84	5/19(26%)	00:04:02	0518/P0518_6.c	RRSRRRSRRRRSERERSERSERERS
user15	1/14(7%)	00:03:46	0518/P0518_6.c	RRRRRRSRRRRSERSRR
user65	0/5(0%)	00:01:55	0518/P0518_6.c	RRSRRRSSSSSS
user39	4/15(26%)	00:01:47	0518/P0518_6.c	RRRSESRRSSERSERERSRSRRS
user11	67/78(85%)	00:01:35	0518/P0518_4.c	RRRRRSRRSERSERSERSEEEEEEEEEEEEEEEEEEEEEEEEEEEEE
user55	6/8(75%)	00:01:29	0518/P0518_2.c	RSRSESEESESESES
user82	7/18(38%)	00:01:20	0518/P0518_5.c	ESESESESSSRSRSSRRSERSERSRRSERSRRSS
user68	12/14(85%)	00:01:13	0518/P0518_6.c	ESESEESESESESESRRSESESESSES
user22	2/5(40%)	00:00:39	0518/P0518_2.c	RRSSESRSES
user43	0/14(0%)	00:00:37	0518/P0518_6.c	RRRRRRRRRRRRSSSSRRS

図 6.8.1 学習者全体の実行状況把握画面

一覧表示されているそれぞれのログ要素について，次の情報が表示される．

- ファイル名
- 操作の種類（Result フィールドの内容によりファイル名の文字色が変化）
- 正解行数/ プログラム行数 / 当該ファイルの最後のプログラムの行数
- 正解行数の増減

「正解行数の増減」については，正解行数が当該ファイルの最大値を更新した瞬間を「進展した」とみなし，♪マークで示している．また，正解行数が当該ファイルの直前の値より減っている場合は「後戻りが発生した」とみなし，★マークで示す．また♪マークが表示されてから一定の時間が経過すると，背景色を変えて，進展していない旨を警告するようにしている．

ログ要素を 1 つ選択すると，次の内容を表示する．

- ソースコード
- 同一ファイルの直前のソースコードとの差分
- 実行結果，エラーメッセージ

図 6.8.2 学習者個人の実行状況把握画面

6.9 性能

現在サービスを提供している Bit Arrow のサーバ[10]は，conoha VPS[11]を使用し，CPU4core、メモリ 4GB で運用している（2021 年 6 月現在）

Bit Arrow のサーバへのアクセス数を，ログ要素への書き込み件数をもとに，2021 年 6 月の範囲で 1 時間おきに（毎時 00 分から 59 分）時間帯を区切って測定したところ，アクセス数が最大となった時間帯（2021 年 6 月 25 日の 14 時台）において，アクセス数は 8956 件，ユニークユーザ数は 86 人（こちらもこの時間帯が最大）であった．また，1 日あたりの平均アクセス数は 8142.1 人であった．

これらのリクエストに対して，特段の障害を起こすことなく運用ができている．

[10]https://bitarrow.eplang.jp/bitarrow/
[11]https://www.conoha.jp/vps/pricing/

表 6.5.2 拡張版 JavaScript で使用できる命令 (1: DOM 操作，入力，グラフィックスなど)

命令	説明
setText(要素名, 文字)	要素に文字を設定
addText(要素名, 文字)	要素に文字を書き足す
getText(要素名)	要素に書かれた文字を取得
getNumber(要素名)	要素に書かれた値を取得
setNumber(要素名，値)	要素に値を設定
onClick(要素名, 命令)	要素が押されたときの動作を設定
waitClick(要素名)	要素がクリックされるまで待つ
findElement(要素名)	指定された要素名を持つ要素を取得する
newElement(タグ名，要素名)	動的に新しい要素を作成
isFormElement(要素名)	指定した要素がフォームかどうか調べる
clearContent(要素名)	指定された要素の中身を空にする
changeImage(要素名，画像ファイル名)	要素の画像を差し替える
move(要素名,x,y)	要素を x,y 座標に移動
rotate(要素名, 角度)	要素を指定の角度回転
transform(要素名，角度，横，縦)	要素の角度と拡大/縮小比率を設定
resize(要素名, 横, 縦)	要素を指定の比率に拡大/縮小
getkey(キー)	キー入力の値を取得 (押されている:1，押されていない:0)
onTouch(命令)	画面が押されたときの動作を設定
setCanvas(キャンバス)	図形を描画する対象のキャンバスを指定
searchCanvas()	描画対象のキャンバスを返す
setColor(r,g,b)	キャンバスに描画する色を設定する
fillRect(x,y,w,h)	キャンバス上の x,y 座標から幅 w，高さ h の四角形を描画する
fillOval(x,y,w,h)	キャンバス上の x,y 座標から幅 w，高さ h の円を描画する
drawLine(x1,y1,x2,y2)	キャンバス上の x1,y1 座標から x2,y2 座標までの直線を引く
clearRect(x,y,w,h)	キャンバス上の x,y 座標から幅 w，高さ h の四角形の形に描画を消す
fillText(文字列,x,y)	キャンバス上の x,y 座標に文字列を表示する
setBGColor(色)	画面の背景色を設定する

表 6.5.3 拡張版 JavaScript で使用できる命令 (2: 動作制御，数学関数，データ操作など)

命令	説明
wait(時間)	指定された時間 (ms) 処理を中断
setInterval()	JavaScript の setInterval と同等
setTimeout()	JavaScript の setTimeout と同等
rnd(値)	0〜値未満の整数乱数を返す
dist()	二点間の距離を返す
angle(x,y)	(0,0) から (x,y) までの直線と x 軸の角度を度数法で返す
rad(d)	度数法の角度をラジアンに変換して返す
deg(r)	ラジアンの角度を度数法に変換して返す
sqrt(値)	値の平方根を返す
sin(d)	度数法の角度 d の正弦を返す
cos(d)	度数法の角度 d の余弦を返す
tan(d)	度数法の角度 d の正接を返す
putToServer(キー, 値)	指定されたキーと値をサーバに送信し，上書き型で保存する
setGroup(グループ名)	データの送信や取得に使うグループ名を指定する
getFromServer(キー)	指定されたキーで保存されている上書き型データを取得する
addLog(キー, 値 1[, 値 2[...[, 値 N]]])	キーを指定して値を送信し，追記型で保存する
findLog(キー)	指定されたキーで保存されている追記型データを取得する
curProject()	現在開いているプロジェクトの情報を取得する
createGraph(要素名 [, データ])	描画領域を指定してグラフオブジェクトを返す
readFile(ファイルパス)	素材ファイル管理にアップロードしたファイルの内容を取得する
writeFile(ファイルパス, データ)	素材ファイル管理にデータをアップロードする

表 6.5.4 拡張版 JavaScript で使用できる命令 (3: グラフオブジェクト，統計関数など)

命令	説明
setData(データ)	グラフにデータをセットする
addData(データ)	グラフにデータを追加する
setAxisText(x 軸,y 軸)	x 軸と y 軸のラベル設定
setXAxisText(x 軸)	x 軸のラベル設定
setYAxisText(y 軸)	y 軸のラベル設定
setAxisType(x 軸,y 軸)	x 軸と y 軸のタイプ (日付など) の設定
setXAxisType(x 軸)	x 軸のタイプ (日付など) の設定
setYAxisType(y 軸)	y 軸のタイプ (日付など) の設定
setXRange(min,max,logScale)	x 軸の範囲と対数スケールの設定
setYRange(min,max,logScale)	y 軸の範囲と対数スケールの設定
line(x,y)	折れ線グラフの描画
bar(x,y)	棒グラフの描画
scatter(x,y, 回帰直線)	散布図を描画する．回帰直線はオプションで表示するか決める
pie(x,y)	円グラフを描画する
vec(k)	全データのフィールド k を配列で返す
min(k)	フィールド k の最小値を返す
max(k)	フィールド k の最大値を返す
med(k)	フィールド k の中央値を返す
mode(k)	フィールド k の最頻値と度数を返す
freq(k)	度数分布を返す
sum(k)	k の総和を返す
mean(k)	k の平均を返す
dev(k)	k の偏差を返す
vari(k)	k の分散を返す
std(k)	k の標準偏差を返す
cov(k1,k2)	k1 と k2 の共分散を返す
corrcoef(k1,k2)	k1 と k2 の相関係数を返す

表 6.5.5 Python のサーバ実行で対応しているライブラリ

ライブラリ	用途	対応メソッド
datetime	日付や時刻の操作	全部
random	乱数の生成	全部
math	数学	全部
re	正規表現	全部
os	OS の機能	getcwd(),path.exists()
matplotlib	グラフ描画	pyplot,dates などの一部
numpy	数値計算	基本的な数値計算用メソッド
OpenCV	画像処理	画像の読み込み・処理・出力に関するメソッド
sklearn	機械学習	サンプルデータセット，SVM，k 近傍など
pandas	データ解析支援	DataFrame,Series,csv と excel ファイルの読み込み
scipy	科学技術計算	音声ファイル操作，信号処理など
BeautifulSoup	スクレイピング	find(),get_text() など
folium	地図の表示	Map クラス,Marker クラスの一部
MeCab	自然言語処理	Tagger クラス

表 6.5.6 簡易 C のグラフィックスライブラリの主な関数

関数	機能
fillRect	四角形を描く
fillOval	楕円を描く
clear	画面を消去する
setColor	描画色を設定する
drawGrid	座標を判別するための枠線を描画する
setPen	線分を描画する（開始点の指定）
movePen	線分を描画する（終了点の指定）
getkey	キーボードの押下状態を取得する
update	一定時間（1/60 秒程度）待機する

第7章　Bit Arrowによるプログラム例と教材

本章では,「情報 I」の指導要領に沿って作成された『高等学校情報科「情報 I」教員研修用教材』（以下,「研修教材」）[15] において紹介されている演習内容を中心に，高校で実施されるであろう演習を Bit Arrow を用いて行うことができるかどうかを検証する．

研修教材は,「第 1 章　情報社会の問題解決」「第 2 章　コミュニケーションと情報デザイン」「第 3 章　コンピュータとプログラミング」「第 4 章　情報通信ネットワークとデータの活用」の 4 つの章からなり，それらの中でも 3 章と 4 章においてプログラミングを使用した活動が記載されている．この資料は教員研修で利用されることを目的としたもので，生徒に対する指導案そのものではないため，掲載されたプログラムも基本的には教員自身が学習のために扱うものである．しかし「学習活動と展開」という項目は，実際の授業における指導案を作る上での参考資料として活用されることが想定されており，そこで教員が生徒に対して示す，操作させる，あるいは多少の改造をさせるプログラムは研修教材に掲載されているものに基づいて作成されることが想定される．そのため，この資料の内容が実際の授業において生徒によってプログラミングされたり，教員によって生徒に配布され実行結果の確認を行わせたりする可能性もある．研修教材の掲載内容について，可能な限り Bit Arrow のみを使用して活動を構築していくことを試みる．

また，研修教材以外にも，Bit Arrow を活用した教材が公開されている．本章では，Bit Arrow 公式ページに記載されている教材，東京書籍で作成した「プログラミング事例集」(以下「事例集」)[72]，Bit Arrow の C 言語を題材とした書籍「楽しく学ぶ C 言語」[73] について述べ，高校や大学初年次の授業においても Bit Arrow が活用できる可能性についても述べる．

2.3 に示した通り，研修教材に掲載された「学習活動と展開」には「プログラムを作る」などと記述された学習（プログラミングが用いられる学習）と，プログラムに関する記述はないがプログラミングを用いてもよいとされる学習がある．ここでは，プログラミングが用いられる学習と，プログラミングを用いてもよいとされる学習に分けて検証を行う．

7.1　プログラミングが用いられる学習

研修教材の「学習活動と展開」から，プログラミングが用いられる学習は次の通りであった．

- 学習 11 コンピュータの仕組み
- 学習 12 外部機器との接続
- 学習 13 基本的プログラム

- 学習 14 応用的プログラム
- 学習 15 アルゴリズムの比較
- 学習 16 確定モデルと確率モデル
- 学習 17 自然現象のモデル化とシミュレーション
- 学習 21 さまざまな形式のデータとその表現形式
- 学習 23 質的データの分析

順に Bit Arrow での実行可否について見ていく.

7.1.1 オーバーフローや計算誤差

研修教材の 3 章,「学習 11　コンピュータの仕組み」においては,「コンピュータの構成，各構成要素の働きを理解する。」「コンピュータは（AND，OR，NOT などの）論理回路などから構成されており，演算の基本は論理演算であることについて理解する。」「コンピュータが数値計算で誤差が生じるのは，取り扱うデータがビット数の限られた有限のものであることについて理解する。」「プログラミング言語を活用し，計算誤差の原因と問題点を生徒に考えさせる授業ができる。」などの目標が設定されている.

プログラムの例として，リスト 7.1 やリスト 7.2 のような，Python でのオーバーフローや計算誤差が発生する例が述べられている．Bit Arrow でこれらのプログラムを実行した場合でも，教材が想定している通りにオーバーフローや誤差が発生させることができる.

リスト 7.1 オーバーフロー (研修教材学習 11)

```
1 import sys # 変数や関数のライブラリを呼び出し
2 x=sys.float_info.max #float 型で表せる最大値をx に代入
3 print(x) # 画面にx の値を表示
4 x=1.797693134862315799999e+308 #float 型の小数点以下に 5 桁「9」の追加した値を
      代入
5 print(x) # 画面にx の値を表示
6 x=1.8e+308 #float 型の最大値より大きな値を代入
7 print(x) # 画面にx の値を表示
```

リスト 7.2 計算誤差 (研修教材学習 11)

```
1 x = 28-27 #x に 28-27 の計算結果を入れる
2 print(x) # 画面に x の値を表示
3 y = 0.28-0.27 #y に 0.28-0.27 の計算結果を入れる
4 print(y)
```

研修教材において使用されている言語は主に Python であるが，Bit Arrow はこれに加えて，C 言語のサポートをしており，コンピュータのデータ構造，メモリモデルの理解に活用している事例もある.

「楽しく学ぶC言語」においては，Bit ArrowのC言語によるアニメーション・ゲームの製作を主軸としながら，構造体やポインタなど，データ表現に関するトピックを学べるように構成している．例えば，リスト7.3のプログラムは，playerという構造体を移動させる処理をmovePlayerという関数で行っており，mainの★で示す箇所で呼び出している．このとき，movePlayerにplayerを値渡ししていることから，移動後のplayerの状態をreturnで返却し，mainで受け取る必要がある．もし，★の式文の左辺player=を忘れてしまうと移動ができなくなる点が記載されている．一方で，リスト7.4のようなプログラムも掲載されている．これは，構造体の配列を関数に渡す例であり，配列は参照渡しとなるため，値の返却をしなくても構造体の内容が書き換えられることを利用している．このような，プログラムを通じて，学習者に内部でのメモリ表現を意識させるようにしている．

リスト 7.3 C言語の値渡し (「楽しく学ぶC言語」より抜粋)

```
1  typedef struct {
2      float x, y,vx, vy;
3  } Rect;
4  Rect movePlayer(Rect player) {
5      //省略(playerのメンバーを書き換える処理)
6      return player;
7  }
8  int main(void) {
9      Rect player;
10     //省略(playerのメンバー初期化)
11     while(1) {
12         clear();
13         player=movePlayer(player);//★
14         update();
15     }
16 }
```

リスト 7.4 C言語の参照渡し (「楽しく学ぶC言語」より抜粋)

```
1  typedef struct {
2      float x, y,vx, vy;
3  } Rect;
4  #define ENEMYCOUNT 10
5  void moveEnemies(Rect enemy[]) {
6      int i;
7      for (i=0 ; i<ENEMYCOUNT ; i++) { // 敵を描画して動かす
8          // enemy[i]のメンバーを書き換える処理
9      }
10 }
11 int main(void) {
12     Rect enemy[ENEMYCOUNT];
13     //中略
14     while(1) {
15         clear();
```

```
16          //中略
17          moveEnemies(enemy);
18          //中略
19          update();
20      }
21  }
```

7.1.2 外部装置との接続

センサデータのリアルタイムな表示・アクチュエータの使用

研修教材の第3章「学習12 外部装置との接続」は，センサデータを使った計測と，アクチュエータを使った制御についての仕組みを理解する内容となっており，micro:bit による内蔵されている加速度センサと LED を用いたプログラム例が掲載されている．

Bit Arrow での micro:bit のプログラムを動作させることについては，現在試験的に動作可能であることがわかっており，研修教材に掲載されているプログラムも動作を確認している．また，センサとアクチュエータが備わっている点で，Raspberry Pi Pico を使用することで同じような演習が可能である．Raspberry Pi Pico は，Bit Arrow が動作している PC に USB ケーブルで接続することで動作させることができる．

リスト 7.5 は，温度センサに反応して LED を点滅させるプログラムを Raspberry Pi Pico で動かしたものである．

リスト 7.5 Raspberry Pi Pico による，温度センサと LED の計測制御

```
1  import machine
2  import time
3  import utime
4  sensor_temp = machine.ADC(4)
5  conversion_factor = 3.3 / (65535)
6  def getTemp():
7      reading = sensor_temp.read_u16() * conversion_factor
8      temperature = 27 - (reading - 0.706)/0.001721
9      return temperature
10 led = machine.Pin(25, machine.Pin.OUT)
11 baseTemp=getTemp() #実行開始時点の温度
12 for i in range(1000):
13     temperature = getTemp()
14     #実行開始時点より温度が上昇していたらLED 点灯
15     if temperature>baseTemp:
16         led.value(1)
17     else:
18         led.value(0)
19     time.sleep(1)
```

また，Bit Arrow のドリトルを用いても Raspberry Pi Pico のセンサデータを制御できる．リ

スト 7.6 は，リスト 7.5 と同様，温度計測と LED の点滅を行うのに加え，Bit Arrow が動作している PC の画面上に，温度の値をリアルタイムに記録するタートルオブジェクトを表示させている．

リスト 7.6 ドリトルによる Raspberry Pi Pico のセンサデータのリアルタイム表示

```
センサ＝ラズパイ!作る 接続.
かめた＝タートル!作る.
温度表示＝ラベル!作る -100 100 位置.
初期温度=センサ!温度.
x=0.
タイマー! 作る 30 時間「
    y=センサ!温度.
    温度表示!(y+"度")書く.
    かめた!(x)(y*6)位置.
    x=x+1.
    「y>初期温度」!なら「
        センサ! 1 LED.
    」そうでなければ「
        センサ! 0 LED.
    」実行.
」実行.
```

今回の事例では，Raspberry Pi Pico に内蔵されているセンサとアクチュエータである温度センサと LED のみを使用したが，Raspberry Pi Pico に他のセンサやアクチュエータを接続することによって，さまざまな演習に対応させることが可能と考えられる．

センサデータの蓄積と分析

研修教材においては，センサデータを用いた演習は先述した micro:bit の事例のみであった．この演習では，組み込み機器で取得したセンサデータの内容を，同一の機器で即座にアクチュエータの制御に反映させる事例であった．

一方，3.1.2 で見たような IoT の実践事例の中には，センサデータを収集・蓄積し，PC など組み込み機器以外においてデータ分析を行う事例も紹介されている．このような事例にも Bit Arrow が対応できることを，いくつかの例を交えて示す．

Bit Arrow は，HTTP 通信が可能な組み込み機器からのセンサデータを受け取ることができる．リスト 7.7 は Raspberry Pi で動作するプログラムであり，温度データを HTTP 通信で Bit Arrow のサーバに送付している．ただし，このプログラム自身は Bit Arrow で編集実行したものではなく，Raspberry Pi をローカルに操作して編集実行したものである．送信されたデータは Bit Arrow 上でリスト 7.8 のようにグラフ化できる．

リスト 7.7 Raspberry Pi から Bit Arrow への温度送信

```
import os
import glob
from time import sleep

```

```
os.system('modprobe w1-gpio')
os.system('modprobe w1-gpio')

base_dir='/sys/bus/w1/devices/'
device_folder=glob.glob(base_dir+'28*')[0]
device_file=device_folder+'/w1_slave'

def read_temp_raw():
    f=open(device_file,'r')
    lines=f.readlines()
    f.close()
    return lines
def read_temp():
    lines=read_temp_raw()
    while lines[0].strip()[-3:] != 'YES':
        sleep(0.2)
        lines=read_temp_raw()
    equal_pos=lines[1].find('t=')
    if equal_pos!=-1:
        temp_string = lines[1][equal_pos+2:]
        temp_c=float(temp_string)/1000.0
        return temp_c
try:
    t=read_temp()
    os.system('wget -O res.txt "http://bitarrow.eplang.jp/beta2104/?BigData/add
        &group=sample&practice=thermo&data1=%f&data2=raspi&url=http://bitarrow.
        eplang.jp/beta2104/fs/pub/xxxxxx/Graph.html"'%t)
except KeyboardInterrupt:
    pass
```

リスト 7.8 Raspberry Pi **から送信された温度データの拡張版** JavaScript **でのグラフ化**

```
g=new GraphObject("graph");
setGroup("mygrp");
res=findLog("thermo");
g.setXAxisText("時間");
g.setYAxisText("気温");
g.line(res,"time","data1");
```

また，Raspberry Pi Pico でも同様にセンサデータを収集・送信できる．図 7.9 は，Bit Arrow の Python プロジェクトから Raspberry Pi Pico で実行可能なプログラムであり，Raspberry Pi Pico の内蔵センサである温度センサに接続し (sensor_temp)，値を読み出して温度（℃）の値に変換して，タイムスタンプとともに表示している．表示する際には，最初にヘッダ部分として time,temp を表示し，その後タイムスタンプと温度をカンマ区切りで 1 秒おきに表示している．この出力結果を「出力を共有」を用いて図 5.2.3 のように送信することができる．

さらに，ConnectDB において，グラフ化の操作を行うことで，図 7.1.1 のような温度変化のグラフを描くことができる．

リスト 7.9 Raspberry Pi Pico による温度データの収集

```
import machine
import time
import utime

sensor_temp = machine.ADC(4)
conv = 3.3 / (65535)

print("time,temp")
for i in range(100):
    reading=sensor_temp.read_u16()*conv
    temperature=27-(reading - 0.706)/0.001721
    (y, M, d, h, m, s, _1, _2)=utime.localtime()
    t="{}/{}/{} {}:{}:{}".format(y,M,d,h,m,s)
    print("{},{}".format(t, temperature) )
    time.sleep(1)
```

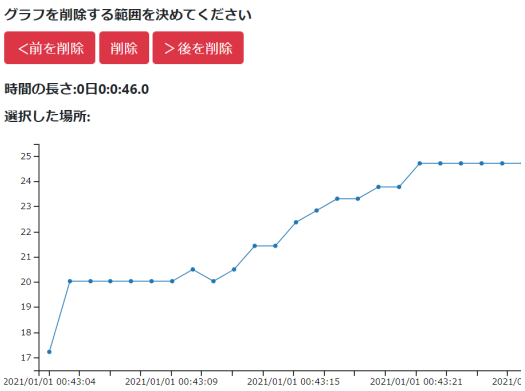

図 7.1.1 温度データのグラフ (ConnectDB)

また，リスト 7.10 のように ConnectDB に保存した温度データを，API を通じて再び Bit Arrow に取り込み，図 7.1.2 のようにグラフ化することも可能である．このとき，リスト 7.9 の温度データ収集プログラム（Raspberry Pi Pico で実行）と，リスト 7.10 のプログラム（サーバで実行）は同じプロジェクト内に置くことが可能であり，データの収集から分析までを 1 つの画面で演習することができる．

リスト 7.10 温度データのグラフ化プログラム

```
from urllib import request
import json
from dateutil.parser import parse
from matplotlib import pyplot
key="XXXXXXXXXXXX"
url='https://cdb.eplang.jp/api/get?key='+key
response=request.urlopen(url)
```

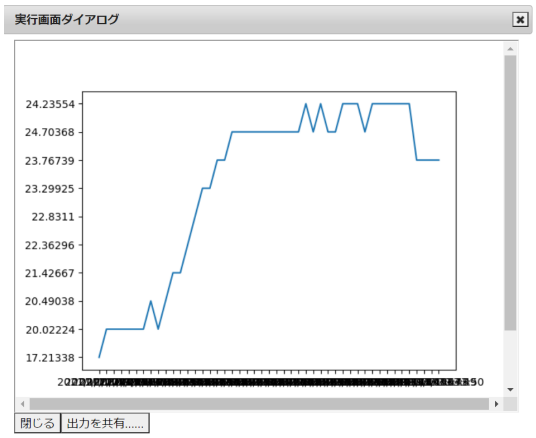

図 7.1.2 温度データのグラフ (Bit Arrow)

```
8  html = response.read().decode("utf-8")
9  response.close()
10 res=json.loads(html)
11 xs=[]
12 ys=[]
13 for e in res:
14     xs.append(parse(e["time"]))
15     ys.append(float(e["temp"]))
16 pyplot.plot(xs,ys)
17 pyplot.show()
```

7.1.3 プログラミングの基本とアルゴリズムの比較

研修教材の 3 章において,「学習 13　アルゴリズムとプログラム」「学習 14　応用的プログラム」「学習 15　アルゴリズムの比較」と，アルゴリズムについては相当の分量を割いて取り上げている．学習 13 においては，順次，選択，反復などのプログラムの基本要素について，学習 14 においては，配列などを使った比較的高度なアルゴリズムおよび Web API の利用について，学習 15 においては「データ数，探索する値が異なれば，処理時間が異なることを理解」させる活動について扱っている．なお，学習 14 の WebAPI については，7.2.1 で後述するように，研修教材の第 4 章でさらに詳しく取り扱っている．

これらの活動では Python のプログラムが掲載されているが，いずれも Bit Arrow の Python で動作することを確認している．さらに，Bit Arrow の Python では，matplotlib を用いたグラフ化機能があり，アルゴリズムの性能をグラフ化することもできる．例えば，学習 15 では選択ソートの例が出ているが，これを配列の要素数を変えながら並べ替え回数をプロットするプログラムを，リスト 7.11 のように書け，図 7.1.3 のようにグラフ化できる．

リスト 7.11 並べ替えアルゴリズムの性能グラフ化

```
import numpy as np
import numpy.random as rd
import matplotlib.pyplot as plt
swc=0
def selectionsort(a):
    global swc
    for i in range(0,len(a),1):
        for j in range(i+1,len(a),1):
            if a[j]<a[i]:
                swc+=1#並べ替え回数
                temp = a[i]
                a[i] = a[j]
                a[j] = temp
#配列数を増やしながら
#並べ替え回数をグラフ化
for n in range(5,15):
    a = rd.randint(1, 100, n)
    print(" ソート前 ",a)
    selectionsort(a)
    print(" ソート後 ",a)
    print("並び替え ",swc)
    plt.scatter(n,swc,color="blue")
plt.show()
```

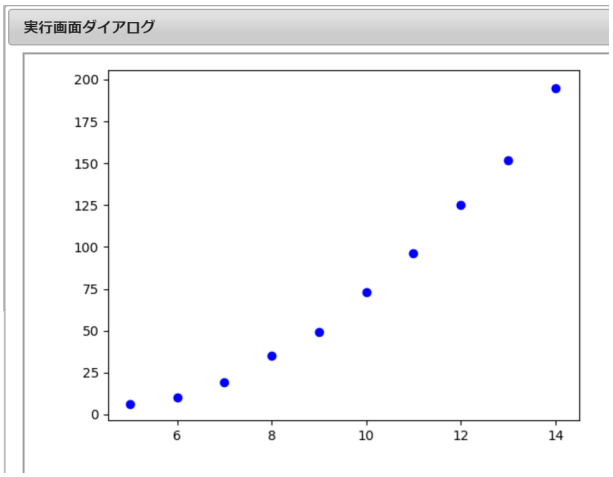

図 7.1.3 並び替えグラフ化の実行結果

7.1.4 モデル化とシミュレーション

研修教材の 3 章において,「学習 16 確定モデルと確率モデル」および「学習 17　自然現象のモデル化とシミュレーション 」において，モデル化とシミュレーションを取り上げている．学

習 16 では，複利計算，さいころの出目のシミュレーション，モンテカルロ法による円周率の計算が例示されている．また学習 17 では，物体の投げ上げ，生命体の増加，ランダムウォークを挙げている．

これらの学習で紹介されているプログラムは，Python のグラフライブラリである matplotlib を使用しているが，Bit Arrow では Web サーバでの実行を行うことでグラフ化を含めて問題なく実行することができる．

さらに，C 言語や拡張版 JavaScript を用いることで，これらのシミュレーションをアニメーションを使って表現することも可能である．例えば，事例集の Vol.3 において，モンテカルロ法による円周率の計算を C 言語で紹介している事例があるが，1 つずつ点をプロットしていく過程をアニメーションで表示している．また，リスト 7.12 は，研修教材の学習 17 の投げ上げシミュレーションを拡張版 JavaScript に書き直したプログラムである．実行すると，図 7.1.4 のように，画面に物体の軌跡をアニメーションで表示することができる．

リスト 7.12 投げ上げシミュレーション (拡張版 JavaScript)

```
dt = 0.01; //微小時間 ( 時間間隔 )
v0 = 30; // 初速度
g = 9.8; // 重力加速度
x = 0; // 水平位置の初期値 0
y = 0; // 鉛直位置の初期値は 0
angle = 45.0; // 投げ上げ角度
vx = v0*cos(angle); // 水平方向の初速度
vy = v0*sin(angle); // 鉛直方向の初速度
for (i=0;i<1000;i++) {
    pvy=vy;
    vy=vy-g*dt;
    x+=vx*dt;
    y+=(pvy+vy)/2*dt;
    fillRect(x,200-y, 1, 1);
    if (y<0) break;
    wait(1000*dt);//dt 秒待機する
}
```

7.1.5 データの扱い方

「学習 21 さまざまなデータの形式とその表現方式」では,「リレーショナルデータベース」「Web API によるデータの取得」「キー・バリュー形式のデータの処理・蓄積」などが扱われている．キー・バリュー形式のデータの処理・蓄積において，研修教材では Python でアドレス帳を作成するプログラムが掲載されているが，Python のオンメモリの配列でデータを初期化し，そこに append を用いて新しいメンバーを追加するという作業が行われているだけであり，実行が終わるとデータそのものは失われるし，他のユーザとデータを共有することもできないため，データベースの機能を再現するには至っていない．

しかし，Bit Arrow から ConnectDB を利用することで，Web API を利用して実際にデータ

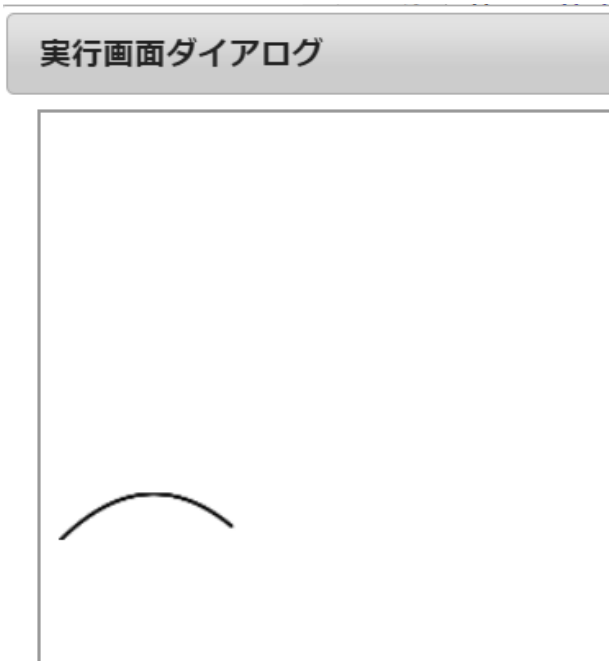

図 7.1.4 投げ上げシミュレーションの実行結果 (アニメーションで描画される)

ベースを操作する体験を学習させることができる．事前に ConnectDB 上に元となる数名分のデータを登録しておき，データベースを操作するためのキーのコピーを行う．リスト 7.13 に示したプログラム例のように，キーを指定し，BA-Python に実装した cdb ライブラリからデータを送信する．実行すると，図 7.1.5 のように，ConnectDB に登録されているデータが一覧で表示され，このプログラムで送信した C 山という名前の人物が登録できたことが確認できる．このデータは ConnectDB のページからも確認することができ，クラス内でキーを共有すれば同じデータベースを複数人で操作することもできる．これを演習に用いることで，実際にデータベースにキー・バリュー形式のデータを蓄積させる体験ができる．

リスト 7.13 住所録データベースの例

```
1  import cdb
2  key="XXXXXXXX"
3  cdb.post(key,{"name":"C 山",
4                  "mobilephone":"090-****-3333",
5                  "email":"cyama@****.com"
6                  })
7  members=cdb.get(key)
8  for member in members:
9          for key,value in member.items():
10                 print("{}: {}".format(key,value))
11         print()
```

「キー・バリュー形式のデータを用いた問題発見」の学習では，SNS などで見られる友達の

図 7.1.5 住所録データベースの実行結果

友達をおすすめのユーザとして提案する仕組みが例として扱われている．友達データベースを Bit Arrow に実装した KVS 型のデータを用いて登録・編集するためのプログラムを，拡張版 JavaScript で記述した．HTML には名前の入力フォームと友達の追加フォーム，現在登録されている友達のリストを編集する領域を設定している (リスト 7.14)．拡張版 JavaScript では，HTML 側で設定されたボタンが押された時の動作を設定しており，各ボタンが押される度に KVS で保存されたデータを取得したり，新たにデータを保存したりしている (リスト 7.15)．実行すると，友達データベースの登録・編集用のユーザインタフェースが表示される (図 7.1.6)．拡張版 JavaScript で登録を行ったデータに，BA-Python からアクセスするプログラムをリスト 7.16 に示す．BA-Python から Bit Arrow の簡易 DB を操作するためのライブラリ badb を使い，登録されている友達リストを呼び出し，おすすめのユーザを表示している (図 7.1.7)．

リスト 7.14 友達データベースの編集 (HTML)

```
1  <html>
2  <body>
3     自分の名前<input name="myname"/>
4     <button name="showlist">開始</button>
5  <h1>友達の追加</h1>
6     友達の名前<input name="friendname"/>
7     <button name="add">追加</button>
8
9  <h1>友達のリスト</h1>
10    <input size=80 name="list"/>
11    <button name="update">更新</button>
12 </body>
13 </html>
```

リスト 7.15 友達データベースの編集 (拡張版 JavaScript)

```
1  onClick("showlist", showList);
```

図 7.1.6 友達データベースの編集画面

```
2  onClick("add", add);
3  onClick("update",update);
4  function showList() {
5      myName=getText("myname");
6      list=getFromServer(myName);
7      setText("list",list);
8      if (list) {
9          friends=list.split(",");
10     } else {
11         friends=[];
12     }
13 }
14 function add() {
15     myName=getText("myname");
16     showList();
17     fName=getText("friendname");
18     if (friends.indexOf(fName)<0) {
19         friends.push(fName);
20     }
21     putToServer(myName, friends.join(","));
22     setText("friendname","");
23     showList();
24 }
25 function update() {
26     myName=getText("myname");
27     putToServer(myName, getText("list"));
28 }
```

リスト 7.16 友達データベースからの友達の推薦

```
1  import badb
2  def friends_of(id):
3      s=badb.getFromServer(id)
4      if s:
```

```
5         return s.split(",")
6     else:
7         return []
8 my_id="yamada"
9 my_friends=friends_of(my_id)
10 recommended_users=[]
11 for my_friend in my_friends:
12     friends=friends_of(my_friend)
13     for friend in friends:
14         recommended_users.append(friend)
15 for friend in my_friends:
16         if friend in recommended_users:
17                 recommended_users.remove(friend)
18 if my_id in recommended_users:
19         recommended_users.remove(my_id)
20 print(recommended_users)
```

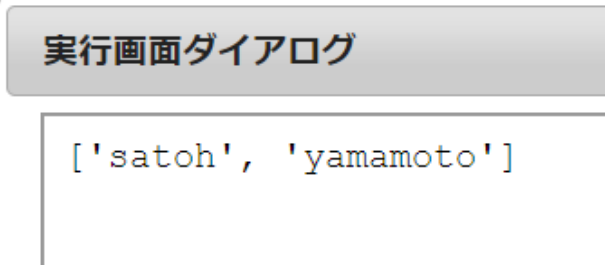

図 7.1.7 友達データベースからの友達の推薦の実行結果

7.1.6 テキストマイニング

「学習 23 質的データの分析」においては，自然言語解析を扱っており，形態素解析を用いて頻出する単語を調べるなどの簡単なテキストマイニングを行っている．形態素解析には MeCab と R を用いているが，Bit Arrow の Python においても，MeCab のライブラリを使用可能である．また，テキストデータのダウンロードには BeautifulSoup[1] を使用し，URL を指定してテキストデータを取得可能である．リスト 7.17 は，BeautifulSoup を用いて青空文庫から芥川龍之介の小説『杜子春』のテキストデータを取得し，MeCab を使って頻出する単語を抽出するプログラムである．図 7.1.8 に実行結果を示す．出現頻度が上位の単語を抽出されている．

リスト 7.17 テキストデータから頻出する単語を抽出するプログラム

```
1 from bs4 import BeautifulSoup
2 import MeCab
3 def sortByValue(x):
```

[1] https://www.crummy.com/software/BeautifulSoup/

```
4     return x[1]["freq"]
5
6  soup = BeautifulSoup('http://pubserver2.herokuapp.com/api/v0.1/books/43015/
       content?format=html')
7  for tag in soup.find_all(["rt", "rp"]):
8          tag.decompose()
9  text_list = soup.find('div', {'class': 'main_text'}).get_text().replace('\r
       ','').replace('\u3000','').split('\n')
10
11 m=MeCab.Tagger("-Owakati")
12 termFreq={}
13 for line in text_list:
14         words = m.parseToNode(line)
15         while(words):
16                 wf=words.feature.split(",")
17                 if wf[0]!='名詞' and wf[0]!='動詞' or wf[1]=='数' or wf[1]=='
                       非自立' or wf[1]=='接尾':
18                 words=words.next
19                 continue
20                 if wf[6] in termFreq:
21                         termFreq[wf[6]]["freq"] += 1
22                 else:
23                         termFreq[wf[6]]={}
24                         termFreq[wf[6]]["freq"] = 1
25                         termFreq[wf[6]]["feauture"] = wf
26                 words=words.next
27 tf=sorted(termFreq.items(),key=sortByValue,reverse=True)
28 for t in tf[:10]:
29         print(t)
```

図 7.1.8 テキストデータから頻出する単語を抽出するプログラムの実行結果

7.2 プログラミングを用いてもよいとされる学習

研修教材の「学習活動と展開」から，プログラミングを用いることが必須ではないものの，用いてもよいとされる学習は次の通りであった．

- 「学習 20 情報システムが提供するサービス」
- 「学習 22 量的データの可視化」

- 「学習 24 データの形式と可視化」

順に Bit Arrow での実行可否について見ていく.

7.2.1 オープンデータの分析

研修教材の 4 章では，学習 20 以降でオープンデータの分析を扱う.

「学習 20 情報システムが提供するサービス」では,「情報システムが提供するサービス」「オープンデータの活用」「オープンデータの可視化と問題発見」などを扱っている．演習例として，埼玉県のオープンデータポータルサイト[2]から AED の設置箇所についてのデータをダウンロードし，地図上に可視化する演習を取り扱っている．この地図の可視化には，jStat MAP という別のサイトを使用している.

Bit Arrow の Python（サーバ実行）には，地図の表示機能を持たせており，リスト 7.18 のようなプログラムによって，上記のサイトからダウンロードしたデータを表示させることができる．この例では，ダウンロードしたデータを aed.csv という名前で保存し，素材ファイル管理 (5.2.2) を用いて user フォルダにアップロードし，pandas から”user/aed.csv”という名前で読み込んでいる.

演習では「人口密度が多いにもかかわらず AED の設置が少ない箇所を探す」ことが提示されている．そこで，川越市の小地域 (町丁・字) 単位の人口密度[3]と位置情報[4]のデータをそれぞれダウンロードし，aed.csv と同じ user フォルダにアップロードした．このデータも pandas を用いて読み込み，地図上に各小地域の人口密度によって異なる色のマーカーを設置させている．このプログラムの実行結果を図 7.2.1 に示す．AED の設置場所には黒いマーカーが，人口密度の多い地域には赤，少ない地域には青いマーカーが設置されている．演習では地図上に可視化されたデータから，AED の設置が少ない箇所を生徒が目視で検出することを想定しているが，プログラムを作成することによって，検出を自動化するような演習にも発展させることができる.

リスト 7.18 AED 位置情報の地図への表示

```
1  import folium as f
2  import pandas as pd
3  import re
4
5  aeds = pd.read_csv("user/aed.csv",encoding="shift-jis")
6  m=f.Map(location=[aeds["緯度"][0],aeds["経度"][0]],zoom_start=15)
7  for i in aeds.T:
8          f.Marker(location=[aeds["緯度"][i],aeds["経度"][i]],popup=aeds["名
               称"][i],icon=f.Icon(color='black')).add_to(m)
9
10 dens=pd.read_csv("user/kawagoe_density.csv",encoding="shift-jis")
11 latlngs=pd.read_csv("user/kawagoe_latlng.csv",encoding="shift-jis")
```

[2]https://opendata.pref.saitama.lg.jp/data/dataset/aed
[3]研修教材に記載されている jStat-map から，人口密度情報をエクスポート可能
[4]https://nlftp.mlit.go.jp/isj/index.html

```
latlngs["地域名"]=""
for i in latlngs.T:
        latlngs["地域名"][i]=latlngs["都道府県名"][i]+latlngs["市区町村名"][i
            ]+latlngs["大字町丁目名"][i]

merged=pd.merge(dens,latlngs)
kawagoe=merged.query('市区町村名.str.contains("川越市")',engine='python')
for i in kawagoe.T:
        if kawagoe["密度(人口総数) "][i] < 1600:
                iconD=f.Icon(color='blue')
        elif kawagoe["密度(人口総数) "][i] < 5000:
                iconD=f.Icon(color='lightblue')
        elif kawagoe["密度(人口総数) "][i] < 8000:
                iconD=f.Icon(color='green')
        elif kawagoe["密度(人口総数) "][i] < 10000:
                iconD=f.Icon(color='orange')
        else:
                iconD=f.Icon(color='red')
        f.Marker(location=[kawagoe["緯度"][i],kawagoe["経度"][i]],popup=
            kawagoe["地域名"][i]+str(kawagoe["密度(人口総数) "][i]),icon=iconD
            ).add_to(m)

m.show()
```

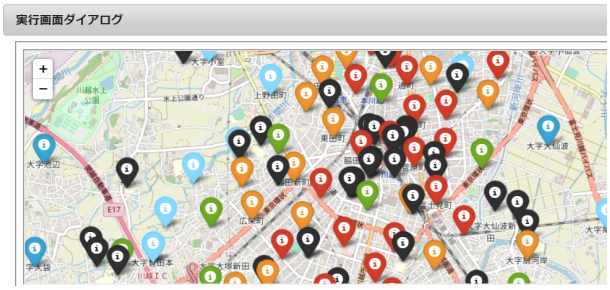

図 7.2.1 AED 位置情報の地図への表示の実行結果

7.2.2 量的データの分析

「学習 22 量的データの分析」においては，気象庁のサイトが提供するオープンデータを用いて，年ごとの気温の変化をグラフ化する演習や，e-Stat などから得られる運動テストから，テスト項目間の相関をグラフ化したり，相関係数を求めたりする演習が扱われている．これらの演習は，表計算ソフトを使用することを前提としているが，Bit Arrow においては，表計算ソフトのデータを読み込み，Python を使ってグラフ化や統計分析を行うことができる．

学習 22 の演習 1（熊谷市の 2017 年および 2018 年の夏季の温度データ[5]のグラフ化）を Bit Arrow の Python を使ってグラフ化したプログラムをリスト 7.19 に，実行結果を図 7.2.2 に示す．また，このプログラムにおいては，グラフ化を行うだけでなく，2017 年と 2018 年の気温の分布について t 検定を行い，2018 年のほうが有意に平均気温が高いことを示している．

リスト 7.19 年次気温データのグラフ化

```
import matplotlib.pyplot as plt
import pandas as pd
from scipy import stats

data=pd.read_csv("user/temp_kumagaya.csv",encoding="shift-jis",header=3)
data17=data[data["年月日"].str.contains("2017")]
data18=data[data["年月日"].str.contains("2018")]

print(data17["最高気温"].max(),data18["最高気温"].max())
print(data17["最高気温"].quantile(0.75),data18["最高気温"].quantile(0.75))
print(data17["最高気温"].median(),data18["最高気温"].median())
print(data17["最高気温"].quantile(0.25),data18["最高気温"].quantile(0.25))
print(data17["最高気温"].min(),data18["最高気温"].min())
print(data17["最高気温"].mean(),data18["最高気温"].mean())
print(data17["最高気温"].var(),data18["最高気温"].var())
print(data17["最高気温"].std(),data18["最高気温"].std())

print(stats.ttest_rel(data17["最高気温"],data18["最高気温"]))

fig,ax=plt.subplots()
bp=ax.boxplot([data17["最高気温"],data18["最高気温"]])
plt.show()
```

また，演習 2（運動テストにおける立ち幅跳びと 50m 走の成績[6]の相関の分析）を Bit Arrow で実装したソースコードをリスト 7.20 に，実行結果を図 7.2.3 に示す．このプログラムでは，それぞれの成績についての散布図と回帰分析，相関係数と寄与率を計算している．

リスト 7.20 運動テストの成績の分析

```
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np

data=pd.read_excel("user/04-03-02.xls")
h1=data[data["校種・学年"]=="高 1"]
h1m=h1[h1["性"]=="男"]

x = h1m["立ち幅跳び"]
y = h1m["50m走"]
```

[5]https://www.data.jma.go.jp/gmd/risk/obsdl/index.php
[6]データ入手先: https://rika-net.com/contents/cp0530/contents/04-03-01.html

```
37.8 41.1
34.575 37.375
32.15 35.349999999999994
29.05 31.85
23.0 23.4
31.659677419354836 34.25
13.111298254891592 15.932377049180332
3.620952672279989 3.991538180849625
Ttest_relResult(statistic=-3.769441642528996, pvalue=0.0003714642163341925 4)
```

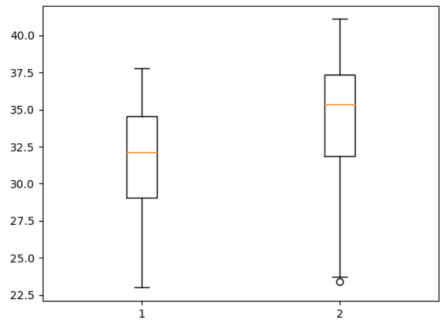

図 7.2.2 年次気温データの実行結果

```
11
12 p=np.polyfit(x,y,1)
13 print(p)
14 R=np.corrcoef(x,y)
15 print(R)
16 print(R**2)
17
18 plt.scatter(x,y)
19 plt.plot(x,x*p[0]+p[1])
20 plt.show()
```

7.2.3 データの可視化を通じた問題発見

「学習 24 データの形式と可視化」においては，質的データの分析や，可視化の手法と問題発見について扱われている．アメリカでのダイヤモンドの流通データ[7]から，カラット数ごとの流通数を可視化する演習が行われており，これを Bit Arrow の BA-Python を用いて実装したプログラムをリスト 7.21 に，実行結果を図 7.2.4 に示す．このときに用いたダイヤモンドのデータはあらかじめインターネットから入手し，Bit Arrow にアップロードして利用している．

リスト 7.21 ダイアモンドの流通データからカラット数ごとの流通数を可視化するプログラム

```
1 import matplotlib.pyplot as plt
2 import pandas as pd
3
4 diamonds=pd.read_csv("user/diamonds.csv",encoding="shift-jis")
5 diamonds=diamonds[diamonds.carat<3]
```

[7] https://github.com/seaborn/seaborn-data

```
[-0.01530446 10.73129815]
[[ 1.         -0.67421451]
 [-0.67421451  1.        ]]
[[1.         0.45456521]
 [0.45456521 1.        ]]
```

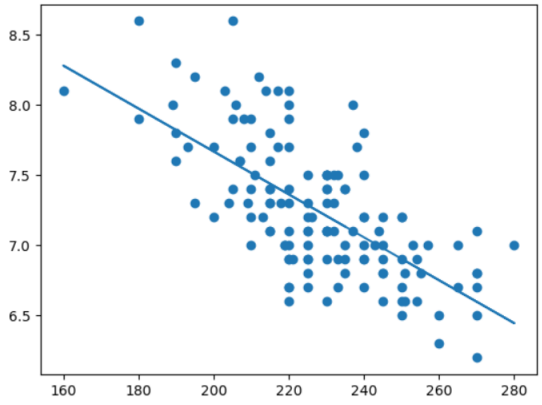

図 7.2.3 運動テストの成績の分析の実行結果

```
6 data=diamonds["carat"].value_counts()
7
8 plt.bar(data.keys(),data,width=0.01)
9 plt.show()
```

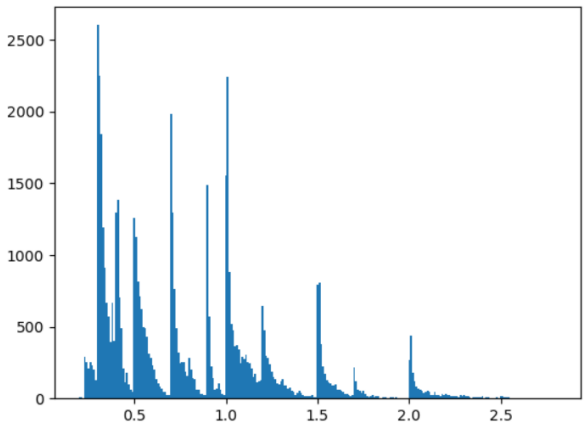

図 7.2.4 ダイアモンドの流通データからカラット数ごとの流通数の可視化の実行結果

7.2.4 身近なアプリケーションの構築

研修教材の第 4 章において「学習 20　情報システムが提供するサービス」が取り上げられている．2 章で述べたように,「情報システム」はネットワークで接続されている複数のコン

ピュータからなる，データを収集・共有・処理する仕組み，と言うことができる．

学習 20 では,「POS システム」「電子決済システム」「SNS（ソーシャル・ネットワーキング・サービス)」「e ラーニングシステム」などを情報システムの例として挙げているが，実際に生徒がプログラミングを行うのは「情報 II」の範囲となる．

この中でも，特に生徒にとって身近な存在といえる SNS について，その機能の基本的な部分を Bit Arrow を用いて構築した例が，事例集の Vol.3 にリスト 7.22 のように紹介されている．これは，簡易 DB の追記型テーブルを拡張版 JavaScript のプログラムから定期的に読み書きすることで，SNS における会話の機能を再現している．また，このプログラムを公開すればスマートフォンからも実行可能であり，生徒たちの（＝複数の）スマートフォン（＝コンピュータ）からメッセージをネットワークで送信し，データが共有される仕組み，すなわち情報システムを開発する体験をさせることができる.「情報 II」における実習での活用や,「情報 I」でもソースコードを配布したりすることで仕組みの体験をさせることが期待できる

リスト 7.22 チャットプログラム (事例集 Vol.3 より抜粋)

```
1  latestUpdate=0;
2  setGroup("sample");
3  onClick("send",send);
4  function send(){
5    var m=getText("message");
6    var n=getText("name");
7    addLog("chat",m,n);
8    setText("message","");
9    read();
10 }
11 function read(){
12   msg=findLog("chat");
13   var txt="";
14   for(var i=0;i<msg.length;i++){
15     txt+=msg[i].data1+"<BR>";
16   }
17   setText("talk",txt);
18 }
19 while(true){
20   read();
21   wait(10000);
22 }
```

また，Bit Arrow の公式サイト[8]には，各言語の使用法を掲載しているが，その中でも拡張版 JavaScript については，身近なアプリケーションの例として，ゲームアプリケーションの作成教材を掲載している．これは，落ちてくるボールをよけるゲームを作成するチュートリアル形式の教材であり，教材通りにプログラムを作成すると，図 7.2.5 のようになる．はじめはキャラクタを表示させるだけのプログラムから，段階的に反復や分岐，最後は配列の概念を用いてゲームを構築できる．

[8]https://bitarrow.eplang.jp/

HTML JavaScript Game別ページで表示

```
x=100;y=300;
bx=[0,100,200,300,400];
by=[0,10,20,30,40];
score=0;
onClick("right",right);
while(true){
    move("neko",x,y);
    moveBall(0);
    moveBall(1);
    moveBall(2);
    moveBall(3);
    moveBall(4);
    wait(50);
}
function moveBall(i){
    move("ball"+i,bx[i],by[i]);
    by[i]+=10;
    if(by[i]>400){
        by[i]=0;
        bx[i]=rnd(400);
        score+=10;
        setText("score",score);
    }
    if(bx[i]>x-30 && bx[i]<x+30 &&
    by[i]>y-30 && by[i]<y+30){
        y=-100;
    }
}
function left(){
    x-=10;
}
function right(){
    x+=10;
}
```

図 7.2.5 教材で完成するゲームのプログラム

表 7.3.1 文科省教材の動作可否

項目	学習内容	動作可否	備考
学習 11	コンピュータの仕組み	○	
学習 12	外部機器との接続	○	Raspberry Pi Pico にも対応
学習 13	基本的プログラム	◎	
学習 14	応用的プログラム	○	
学習 15	アルゴリズムの比較	◎	
学習 16	確定モデルと確率モデル	○	
学習 17	自然現象のモデル化とシミュレーション	○	
学習 18	情報通信ネットワークの仕組み	-	
学習 19	情報通信ネットワークの構築	-	
学習 20	情報システムが提供するサービス	○	
学習 21	さまざまな形式のデータとその表現形式	○	
学習 22	量的データの分析	△	研修教材では表計算ソフト
学習 23	質的データの分析	△	研修教材では R
学習 24	データの形式と可視化	△	研修教材では R

-:プログラム掲載なし，◎:サーバ側とブラウザ側両方で実行可能，○:サーバ側でのみ実行可能，△：代替環境で同様に演習可能，×:実行不可能

7.3 研修用教材との対応

研修教材に掲載されているプログラムと BA-Python での動作可否を表 7.3.1 に示す.

多くの Python プログラムについては，掲載されているプログラムをそのまま実行できることが確認された．micro:bit を使った演習（学習 12）では，正式な機能ではないものの，掲載されているプログラムを実行可能なことは確認できた．また，Raspberry Pi Pico を代替機器として実行することでも，センサとアクチュエータについての同程度の学習ができることが確認された．また，学習 22 から学習 24 にかけての，表計算や R を使った演習では，Python で同等のライブラリ，データを使用することでほぼ同じ内容の結果を得ることができた．

さらに,「AED の位置」「人口密度」という複数のデータを可視化した事例においては,「人口密度に対して AED が少ない場所の分析」を行うのは，生徒の目視によることを前提としているが，プログラミングを行うことで，分析自体を自動化するようなことも可能であることが示された．

また，シミュレーションの結果を可視化する例では，アニメーションを用いて表示してわかりやすく示すこともできた．データを独自に集めるためにフォームを使用したり，センサからデータを集め，分析を行うまでの過程も Bit Arrow で統一して行うことができた．

7.4 まとめ

本章では，高校教科「情報」で学習すべき内容である,「コンピュータの仕組み」「情報システム」「アルゴリズム」「モデル化とシミュレーション」「オープンデータの分析」「センサデータの収集・分析」「身近なアプリケーションの構築」などのうち，プログラミングが活用されると考えられる内容が Bit Arrow で学習できることを,『高等学校情報科「情報 I」教員研修用教材』(研修教材) のプログラミングに関係する章「第 3 章 コンピュータとプログラミング」「第 4 章 情報通信ネットワークとデータの活用」に掲載されているプログラムが Bit Arrow で動作させられることを通じて検証し，ほとんどの単元のプログラムをそのまま動作させることができた．そうでないプログラムについても，代替の機器やプログラムで同等の演習ができることも確認した．

また，同等の演習ができるだけでなく，Bit Arrow のデータ共有やアニメーションなどの仕組みを活用して，より発展的な活動にも利用可能であることが示唆された．研修教材をベースに，適宜 Bit Arrow で動作させられる内容と，これら発展的な内容を盛り込んだ教材を提案可能であると言える．さらに，Bit Arrow の使用を前提に作成された C 言語の教科書「楽しく学ぶ C 言語」，ゲーム作成の教材などもすでに発表されており，Bit Arrow は高校生を含めた初学者向けのプログラミング環境として実用に供するシステムであることも示した．

第8章　評価

8.1　Bit Arrowを利用した高等学校での授業実践

本節では，Bit Arrowの拡張版JavaScriptを利用して高校での授業を行った結果をもとに，Bit Arrowが先ほど述べたような高校の現場で起きている問題の解決に寄与していることを示す．高校での実践内容について報告し，この実践で収集したログを分析することで，Bit Arrowを利用することによって，既存の環境より学習の支援を行えていることを示す．

単元は二種類あり，一つは，分岐や反復といったプログラミングの基本的な概念を学習するための授業で，もう一つはアニメーションを用いたゲーム制作を行う授業である．基本的な概念を学習するための授業の評価は，5.2.7節で述べたログデータを元に行う．ゲーム制作を行った授業は，学習者へのアンケートを元に評価する[74]．

8.1.1　分岐や反復の学習による実践

分析の対象

Bit Arrowを利用していた神奈川県立柏陽高校1年生170名の実行時のログを高校教員と協力して調査した．授業は2015年10月下旬から11月上旬に行われた．ここでの利用者は，プログラミング経験がない初学者がほとんどである．はじめに「奇偶判定」,「じゃんけん」のプログラム(5章 の図5.2.11)が作成され，次に「FizzBuzz」のプログラム(図8.1.5)が作成されている．最後に，図8.1.1，図8.1.2のプログラムが作成されて，これを通じて分岐と反復を学習している．なお，拡張版JavaScriptを用いた授業を行った生徒には，一般的なJavaScriptとは異なる言語を用いていることを実習開始前に説明した．

比較対象として，同じようにプログラミング経験のない同じ高校の高校1年生40人が同様の内容をテキストエディタで行った際のデータを用いる．テキストエディタでは，Bit Arrowのようにログを収集することができないため，ファイルを保存するタイミングでログを収集することができるアプリケーションを開発・使用した．ここで開発したテキストエディタでは，以下のログが収集できる．

1. ユーザID
2. 実行を行った時刻
3. 実行時のプログラム

テキストエディタを用いた生徒の実行結果とエラーメッセージはログに残らないため，後からログに保存されているプログラムを担当教員が実行し直し確認した．

HTML JavaScript Kadai1別ページで表示

```
1  for(i=1;i<=50;i++){
2      if(i%2==0){
3          addText("t",
4          "<img src=images/neko1.png>");
5      }else{
6          addText("t",
7          "<img src=images/bluefish.png>");
8      }
9  }
10
```

図 8.1.1 交互に画像表示するプログラム

HTML JavaScript Kadai2別ページで表示

```
1  for(i=1;i<=50;i++){
2      if(i%2==0){
3          addText("t",
4          "<img src=images/neko1.png>");
5      }else{
6          addText("t",
7          "<img src=images/bluefish.png>");
8      }
9      if(i%10==0){
10         addText("t","<br>");
11     }
12 }
13
```

図 8.1.2 交互に画像表示し，一定間隔で改行するプログラム

エラーの修正支援の検証

はじめに，一画面上での編集と実行を実装したことにより学習者がプログラムを実行した平均回数が増加しているかを測るためすべての実行を集計した．さらに，短い命令を使えることでエラーの発生を減らすことができているかを確認するため，すべての実行をエラーが出たものと出なかったものに分けて集計した．そして，エラー発生時に学習者が自力でエラーを直すことができるよう表示させるエラーダイアログによってエラーの修正にかかる時間が短くなっているかを確認するため，エラーが発生してからそれを修正するまでの時間を算出した．これは，エラーが連続で発生したかどうかなどは考慮せず，前回実行時はエラーがなかった状態のときに，処理系がエラーを出した時間から，次にエラーがなく実行されるまでの時間を計測している．

Bit Arrow のユーザ 170 人が利用していた 10/6 から 11/8 までのログを調べると，実行の総数は，18282 回となり，約一ヶ月間で一人当たり 85.2 回の実行を行っていた．テキストエディタでは，同様の内容を行った期間で 2351 回の保存がされており，一人当たり 40.9 回となった．また，この実行のうち，エラーがなかったものとあったものの回数の割合を調べたところ，表

表 8.1.1 一人当たりのエラーの平均回数と修正時間

環境	エラーなし	エラーあり	修正時間
テキストエディタ	40.9 (69.6%)	17.9 (30.4%)	190.6 秒
Bit Arrow	85.2 (79.2%)	22.3 (20.8%)	105.0 秒

8.1.1 の結果となった．Bit Arrow ではエラーがなかったものの割合が約 79%となり，テキストエディタと比べてエラーの割合が少なかった．このことから，Bit Arrow では煩雑な操作を減らすことで多くのプログラムを記述しての実行や試行錯誤をさせることができていたといえる．

また，処理手順を簡潔に書けるよう用意したライブラリにより，エラーの割合を減らすことができたと考えられる．エラー発生時，その発生場所を表示してエラーの修正を支援できているか，エラーの修正にかかった時間を調べた．この時間は，テキストエディタを用いた授業では，約 190.6 秒かかっていたが，Bit Arrow では，エラーが発生してから修正するまでに約 105.0 秒と減少していた．Bit Arrow を用いることでエラーの修正時間を約 85 秒短縮させることができており，授業を担当した教員からも，これまでの環境ではエラーが起こるとエラーメッセージが表示されず何をすればいいか分からず固まっていた生徒が多かったが，ダイアログでエラーが分かることで前向きにそれを直そうとしている生徒が増えていた様子が見て取れたことから，生徒のエラー修正に向かう意欲を維持できていることを確認できた．

エラーの修正にかかる時間を短くすることができたことから，Bit Arrow でエラーの発生場所を表示することで，学習者が自力でエラーから抜け出すことを支援できたといえる．素早いエラーの修正を支援することで，プログラムの構造などの本質的な部分の理解に費やす時間を増やすことができる．また，生徒が自力でエラーから抜け出すことを支援することで，授業などにおいては生徒のエラーへの対処に関する教員の負担も軽減させることができているといえる．

一方，一部には JavaScript ファイルに HTML を記述したことによるエラーもあった．HTML と JavaScript のファイルを分けることでプログラムを分かりやすくする狙いがあったが，このログで用いられていたじゃんけんや FizzBuzz のような短いプログラムにおいては，一つのファイルに書くことのほうが簡単であった可能性もある．

課題作成中の実行の検証

エラーの対応にはあまり時間を使わせずに，実行はできるが思い通りに動かない原因を考えさせることが，アルゴリズムやプログラムの構造を考え，理解させることにつながると考え，課題のプログラムを作成している間の実行の成否を調査した．処理系はエラーを起こさないが期待した動作をしない場合，実行できたかどうかを見るだけでは検出することができない．そこで，利用者のログから，図 8.1.1，図 8.1.2 のプログラムを作る過程のプログラムを実行して観察した [75][76].

図 8.1.3 に，テキストエディタと Bit Arrow での実行の割合を示す．ここでは図 8.1.1 などの

プログラム作成に取り組んでいた間の結果のみを対象としたため，5.1.2 で述べた全体のエラーを出した割合とは異なる．エラーを出した割合は，分析期間全体の割合に比べ増加したが，テキストエディタでは 49.1%あったのが Bit Arrow では 36.1%と減少した．エラーはないが思い通りに動かなかった実行は 24.6%から 32.6%，思い通りの動きをするプログラムが 12.2%から 18.8%へ増加した．その他の実行の中身については，図 8.1.1 のプログラム作成前に作られた FizzBuzz のプログラムを実行していることが多く，これを実行しながらアルゴリズムを考えていたと推察できる．エラーの割合は，Bit Arrow で約 13%減らすことができた．

また，エラーはないが思い通りの動きをしない実行は約 8%増加した．思い通りの動きをしない実行は，どうすれば思い通りに動かすことができるかというアルゴリズムを構築する必要がある．Bit Arrow を利用することでエラーの割合が減り，思い通りの動きをしない実行と思い通りの動きをする実行が増えたことから，プログラムの構造の学習に時間を使わせることができたといえる．

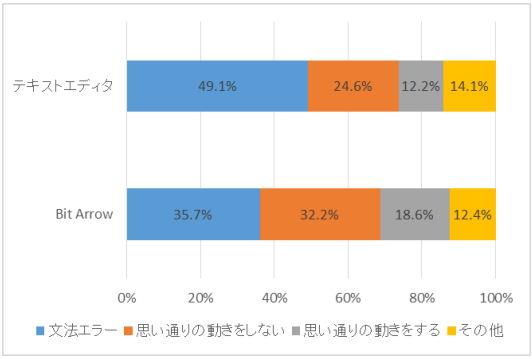

図 8.1.3 課題作成中の実行成否の割合

思い通りに動かない実行の原因の検証

実行はできるが思い通りに動かなかった原因が，実際にアルゴリズムやプログラムの構造に関わることかを調べるため，その原因を調べ分類した [77]．図 8.1.4 に，Bit Arrow での実行時，エラーはないが思い通りにいかなかった原因の内訳を示す．図 8.1.1，図 8.1.2 のプログラムを作成中の実行ログから，間違えた原因をアルゴリズム，構文や関数の使い方，打ち間違いに分け，それぞれの割合を示す．最も多いのはアルゴリズムの間違いであった．これは以前に作った FizzBuzz のプログラムを改造しながら作成している利用者が多く，FizzBuzz のプログラムが残っていたことが原因の多くを占め，53.2%となった．この間違いに対応するために，表示がどのタイミングで行われるか，いつ行うことが正しいかといったアルゴリズムを考えさせることができる．

構文や関数の使い方が原因となったものは 25.4%となった．この間違いの主な原因とそれが起きた回数を表 8.1.2 に示す．もっとも多かった原因は，二重ループに同じ変数を使用していたことであった．これは図 8.1.2 のように if 文で改行を判断せずに二重ループで改行を判断し

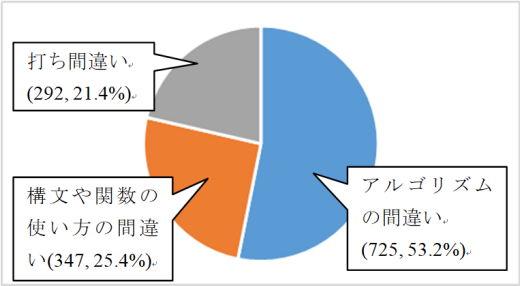

図 8.1.4 思い通りの動きをしない原因の内訳

表 8.1.2 構文や関数の使い方の間違いの主な原因と出現回数

原因	回数
二重ループに同じ変数を使用	88
ループ条件の誤り	53
ループ変数加算忘れ	43
ループ変数初期化なし	15
分岐判定とループ変数が違う	14
その他	134

ようとした学習者によるものであった．他にも，ループの条件やループ変数の初期化と加算が多く表れており，反復の使い方が原因となることが多かった．こうした反復や分岐の使い方に起因して思い通りに動かないプログラムを直そうとすることで，分岐や反復の概念を学習させることができていると考えられる．一方，打ち間違いが原因で思い通りに動かないプログラムも 21.4%あった．この原因を表 8.1.3 に示す．上位を占めたのは URL 間違いや img タグ内の src を scr と打ち間違えていたことなどであった．これは，課題のプログラムで画像を表示させようとしている部分で，文字列の中の HTML の記述を間違えていたものであった．JavaScript のメソッド名を間違えたり，セミコロンを忘れたりといった打ち間違いは，Bit Arrow でエラーとして通知することができるが，文字列の中身を間違えても，エラーメッセージは表示されないため，間違えた場所を探す必要がある．打ち間違いに起因するような多くのエラーを削減することはできたが，それでもなお，このような打ち間違いにより上手く実行ができないことがあることが分かった．これを修正する作業もプログラムの本質的な概念を学習できているとは言えない．

処理系がエラーを出さないが思い通りに動かない原因は，アルゴリズムの間違いと構文や関数の使い方の間違いが合わせて 3/4 以上になり，Bit Arrow を利用することで，分岐や反復の使い方やアルゴリズムの理解に取り組ませることができたといえる．一方，エラーを出さない打ち間違いもあり，これを減らすことで，よりプログラミングの構造の理解に時間を使うことができると考えられる．

表 8.1.3 打ち間違いの主な原因と出現回数

原因	回数
URL 間違い	79
src を scr と打ち間違い	60
> がない	23
++ と書くところを tt と書いている	23
メソッド名間違い	17
その他	90

8.1.2 アニメーションを用いた作品制作における実践

実践の対象

8.1.1 の分析対象とは別の，プログラミング経験のない高校 1 年生を対象に初学者用 JavaScript を用いてゲームを作成させた [78][79]．ゲームなどの身近な題材を扱う授業では，学習者の意欲を向上させることができると考えられる．この実践では，7.2.4 で述べた拡張版 JavaScript によるゲーム作成教材を用いた．実践は 1 回あたり 65 分の授業 2 回（チュートリアルと作品制作にそれぞれ 1 回ずつ）を用いて行った．

学習者へのアンケート調査

Bit Arrow の拡張版 JavaScript がゲーム制作の助けになっているかを確認するため，サンプル教材に倣ってゲームを制作できたか学習者に聞いた．ゲームを作成した学習者のうち 76 人にアンケート調査を行った．ゲームを教材のサンプルに倣って制作できたか，という問いに対して，67 人ができたと回答した．配列を用いる難易度の高いプログラムであったが，多くの学習者が教材通りにゲーム制作をすることができた．図 7.2.5 のプログラムで，通常の JavaScript では難しいゲーム制作のような題材が，move() のようなアニメーション制作を補助するライブラリを用意したことで行えるようになったことがわかる．次に，実際に初学者がプログラミングを学習する上で効果的であったサポート機能を調べるために，自由記述で Bit Arrow の使いやすかった点を聞いた．多く得られた回答を表 8.1.4 に示す．最も多かった回答は，エラーの表示がされることであった．実行すると，エラーの有無がすぐに分かり，エラーが発生していればその場所を通知する機能が，学習者の助けになっていたことが分かる．次に多かった回答は対応する記号の自動入力であった．この機能により，エラーを未然に防ぐことができた．実行画面をすぐに確認できること，HTML と JavaScript の切り替えができることが次に多かったことから，操作の煩雑さや，HTML と JavaScript の混同といった問題を解決できている．最後に，提案した支援の他に初学者が必要としている機能を調査するため，自由記述で Bit Arrow の使いにくかった点のアンケートをとった結果を表 8.1.5 に示す．自由記述であるため，回答の中には Bit Arrow という環境への指摘だけではなく，プログラミング全般に対する感想も見られ

図 8.1.5 初学者用 JavaScript で記述した FizzBuzz プログラム

た．最も多かったのは使いにくい点は特になかったという回答であり，Bit Arrow がプログラミング教育の助けになったと感じている学習者が多かったことが分かる．また，次にプログラミングが難しいことや，打ち込む量が多かったこと，ミスがあると動かないことなどが挙げられた．打ち込む量が多いという意見は，初学者用 JavaScript を用いても多く挙げられたが，教材として用意したプログラムが長かったことも要因として考えられる．ミスがあると動かない点は，プログラミング全般に言えることであり，これを学習することも必要な経験である．

他にも，アンケートでの回答にはなかったものの，現場にいた生徒の反応で特筆すべきものとして，演習中に図 8.1.5 の FizzBuzz プログラムを作成するにあたって，最初は 12 行目の wait メソッドのないプログラムを提示し，1 から 50 までの結果が一気に表示されるものを示した．そのあと，wait 命令を挿入して実行させたところ，生徒たちから「おおっ！」という声が挙がった．このプログラムを書く前には，事前に FizzBuzz を生徒間で口頭で交互に言う活動を行わせており，コンピューターが自分たちの活動を模して実行する様子に興味が湧いたと考えられる．この wait メソッドの挿入は，拡張版 JavaScript であれば簡単であるが，従来の JavaScript では setInterval に書き換える必要があり容易ではない．プログラムに一行命令を書き足すだけで，プログラムをゆっくり動かして動作を確認することができる，という特徴は生徒たちのプログラミングへの興味を喚起できると推察される．

8.2 ログデータを用いた授業支援

Bit Arrow にはログ収集機能とログ閲覧機能があり，これらの機能を用いて，大学の情報系

表 8.1.4 Bit Arrow の使いやすかった点のアンケート結果

使いやすかった点	人数
エラー表示 (ポップアップ，!マーク)	16
対応する記号の自動入力	15
実行画面をすぐに確認できる	8
HTML と JavaScript の切り替え	8
コメントや文字列の色付け	7

表 8.1.5 Bit Arrow の使いにくかった点のアンケート結果

使いにくかった点	人数
特になし	21
プログラミングが難しい・打ち込む量が多い	10
ミスがあると動かない	9
画面の切り替え	8
エラーの場所が分かりにくい	5

の授業の指導に活用した事例を示す.

近年，授業がオンラインで開催されることが多くなり，オンラインツールでのサポートが普及してきた．その中でも広く使われているツールの 1 つに Slack[1]がある．Slack は SNS の 1 つで，主に文字ベースでメッセージを送りあうことができる．チャンネル機能により，授業ごとに別のチャンネルを立てられることや，スマートフォンのアプリや通知機能により，受講者へのリアルタイムな対応が可能であることなどの特徴がある．また，Slack 用の API を用いて，プログラムからメッセージを送る Bot 機能を有するため，自動的にメッセージをやりとりすることも可能である．

2021 年 4 月から 7 月にかけて，明星大学情報学部のプログラミングの授業において，Bit Arrow と Slack を用いた完全オンラインの授業[2]を実施した．Bit Arrow を用いてプログラミングの演習を行い，Slack で質問を行うという形式で授業を行った．

この方式の授業は 2020 年度も実施していたが，そこで問題になったのは，トラブルが発生していて解決ができそうにない（つまずいている）にもかかわらず，質問をしてこない学生が多くいることであった [80]．今回の授業でも同様の問題が発生すると考え，Bit Arrow のログの閲覧機能を利用して，つまずいている兆候のある学生については，Slack の DM(ダイレクトメッセージ）機能を利用して，直接アドバイスを行えるような体制を構築することを試みた．

しかし，つまずいていることをログから判断するには，受講者のログを 1 つ 1 つ確認する必要があるため，つまずいていることの検知を自動化する必要が出てきた．それには，つまずい

[1] https://slack.com/intl/ja-jp/
[2] 第 1 回のみ対面だったが，第 2 回目以降は第 15 回まで対面なし

ている兆候をログから抽出する基準を定めなければならない．

その基準の候補として「トラブルが発生してから，一定時間の間に解決がみられなければ，その受講者はつまずいている，つまり質問などを行ってサポートを受ける必要がある」という仮説を立てた．

この仮説を検証するための実験として，Slack にて質問を行った学生を「つまずいている」とみなし，その学生のつまずきの原因となったトラブルの発生から，質問してくるまでの時間を計測した．

質問してきたタイミングについては，Slack のタイムスタンプで取得することができ，トラブルの発生のタイミングについては，当該学生の Bit Arrow ログを目視で確認し，それらのタイミングの差を計測した．その結果，約 50 件程度の質問について，トラブル発生から質問までの時間は，中央値ベースでおおむね 15 分であることがわかった．また，正解行数を利用することで,「正解行数が増加しない状態が長く続いていること」をトラブルが発生した兆候ととらえることができる可能性が示唆された [81]

現在，この結果を参考にして，ある課題に着手してから一定の時間が経過したあと，Slack の Bot 機能を利用して，受講者に自動的にメッセージを送れる機能を開発している途中である [82].

8.3 教育現場における利用実績

8.3.1 利用申請件数

Bit Arrow は管理者側で教員からの利用申請を受け付けて登録作業をおこなっている．運用を開始した 2016 年 8 月以来，登録申請の件数を，教員の所属別に表 8.3.1 に示す．なお，集計に当たっては，授業の準備のために年度の開始よりも早めに申請されることがあると考え，多くの教育機関で春休みに相当する 3 月から翌年の 2 月までを各年度とした．

表 8.3.1 Bit Arrow の登録申請数（各年 3 月〜翌年 2 月）

	中学校	高校	大学	高専	専門	小学校	その他	合計
2018 年度	4	24	4	0	0	0	1	33
2019 年度	11	45	5	0	1	2	7	71
2020 年度	7	45	13	2	1	1	4	73

2020 年度には，新型コロナウイルス感染症の影響でリモート授業が行われる機会が増加したことで，オンラインプログラミング環境の需要が高まった．実際，この時期に利用を申請してきた高専の先生から授業後に頂いたメールによると，急遽リモート授業導入が決まり，C 言語のプログラミング担当として困っていたところだったとのことである．この先生は，これまで C 言語でアニメーションをするプログラムを授業で扱っており，Bit Arrow を用いることで，オンラインでもこれまでと同等の演習を行うことができたという．JavaScript へトランスパイ

ルしてインタラクションのあるプログラムを扱えるようにしたことで，アニメーションを用いた授業をサポートすることができた．

8.3.2 利用言語

Bit Arrow を用いた活動で作成されたプロジェクト数を言語別に集計したものを表 8.3.2，高等学校において Bit Arrow を利用したクラス数を，各クラス内で「主に使用された言語」別，学期別（前期：4 月〜8 月，後期：9 月 翌年 3 月）に集計したものを表 8.3.3 に示す．ここでいう「主に使用された言語」とは，クラス内で作成されたプロジェクトの個数を言語別に集計し，ある言語の個数が全プロジェクトの 80%を越えた場合，その言語を「主に使用された」言語とし，どの言語も 80%以下であった場合は，そのクラスでは複数の言語を扱ったとみなし,「複数言語」という分類にした．また，学期についてはプロジェクトが作成された時刻をもとに判断した[3]．なお，クラスにおけるユーザ数が 3 名以下のクラスについては集計の対象外とした．

表 8.3.2 言語ごとの作成されたプロジェクト数 (年度ごと)

	C	DNCL	ドリトル	拡張版 JavaScript	Python	合計
2016 年度	380		3926	1606		5912
2017 年度	3897		5347	773		10017
2018 年度	7761	18	6187	351	18	14335
2019 年度	3750	167	3771	829	1718	10235
2020 年度	10109	141	2941	327	2769	16287

表 8.3.3 高校で各言語を主に利用したクラス数

	C	DNCL	ドリトル	拡張版 JavaScript	Python	複数言語	合計
2016 後期			3	3		2	8
2017 前期	1		1	1			3
2017 後期			6	2		1	9
2018 前期			2	1			3
2018 後期	1		5	2		1	9
2019 前期	3		2	1		1	7
2019 後期	4	1	8	3	2	4	22
2020 前期	4		1	1			6

[3]複数の学期に同一クラスでプロジェクトが作成された場合，それぞれの学期でそのクラスが集計されている

8.4 まとめ

本章では，Bit Arrow を実際の教育現場で用いた結果から，エラーの発生防止と学習者自身によるエラー修正の支援について評価した．Bit Arrow は Web ブラウザでコードの編集と実行を行うことができる．演習用 PC へのインストールのような準備を必要としないため，これまで教育現場で使われていた環境と同様に導入が容易であった．高等学校における実践においては，拡張版 JavaScript を利用して演習を行った．エラーが起こったときにはエラーの発生場所を通知し，エラーの発生を抑制するためにエディタに入力補助機能を持たせた．また，拡張版 JavaScript では，標準の JavaScript と比べて短い命令文で同等の処理を行えるライブラリを提供した．演習に Bit Arrow を用いた学習者と標準のテキストエディタを用いた学習者のログを調査した結果，Bit Arrow を利用した学習者は実行した際にエラーを出す割合が少なくなり，またエラーを修正するのにかかった時間が短いことがわかった．このことから，Bit Arrow はエラーの発生を減らし，学習者が自力でエラーを修正することを支援できたといえる．

本質的でないエラーの修正に費やす時間を減らすことで，アルゴリズムや分岐・反復といったプログラミングの基本的な概念の学習を支援することができていると考えられる．また，学習者にとって身近なアプリケーションの仕組みで使われているアニメーションを題材として扱うことができた．

高等学校における実践で学習者が記述したプログラムを調査するために，Bit Arrow で実行のたびに収集しているソースコードの内容や実行結果といった内容を持つログを用いた．この調査で，ログに残った情報から学習者が起こしたエラーや，思い通りにならない実行についてその原因を特定することができた．また，収集しているログの内容を時系列に表示するログ閲覧機能を提供したことでつまずいている学習者を発見する支援ができるようになった．ログを利用することにより，エラーの発生率やエラーの復帰時間を集計できるだけでなく，ソースコードの内容も記録されており，変更履歴を追うことも可能であるため，エラーではないが思い通りの結果になっていない学習者の状態も把握できるようになった．大学で開かれた授業においては，課題に着手してからの時間や，ソースコードの内容などの情報からトラブルが発生した兆候をとらえることに活用することができた．

第9章　結言

本章では，本研究の結論，成果を述べ，今後の課題についてまとめる．

9.1　研究成果

本研究は，高校における「情報」の授業における課題について，Web ブラウザで動作するプログラミング環境 Bit Arrow を提案し，次のようなアプローチによって解決を行った．

- **生徒のエラー発生防止と，エラー修正支援**

  高校の PC 教室などでインストール作業が不要でプログラミングが行える環境として，Web ブラウザと OS 標準のテキストエディタ（メモ帳など）をやむなく利用すると，入力ミスによるエラーが多発し，エラーメッセージの確認も煩雑であり，エラーの修正が困難であった．Bit Arrow を利用することで，IDE の入力支援とエラー表示機構によってエラーの発生そのものを抑えるとともに，エラーの箇所を明確に示すことができエラーの修正を支援することが確認された．

  実際の高校の実践において，Bit Arrow による演習と，メモ帳と同等の機能しかないエディタを使って演習を行った場合と比較して，一人あたりのエラーの発生する割合が，Bit Arrow を用いた場合のほうが 10 ポイント 減少し，修正時間も 90 秒短縮されることが確認できた．

- **指導要領で求められるプログラミング活動の支援**

  2022 年度からの高校教科「情報 I」においては，ネットワークで相互接続された情報システムを構築したり，データを収集・整理・分析したりするなどの活動にプログラミングに活用することが求められている．組み込み機器のセンサデータを収集したり，Web フォームからデータを入力したり，WebAPI を経由してオープンデータを入手したりすることを，可能な限り同一のプログラミング環境を用いて行うことで，データのやりとりを円滑に行えると考えられる．Bit Arrow では，Raspberry Pi や Raspberry Pi Pico などの組み込み機器からのデータ入力，素材ファイルや簡易 DB によるデータの蓄積，WebAPI の発行，グラフ描画ライブラリ，アニメーションなどをサポートすることで，**データの収集・整理・分析**の一連の活動を総合的にサポートするプラットフォームになっている．また，**アルゴリズム**や**コンピュータの仕組み**の学習，生徒にとって身近なアプリケーションであるゲームや SNS などの構築を通じて，**アプリケーションソフトウェアの一部を実現する**活動などもサポートしている．

これらの学習項目に対応できるように，C，拡張版 JavaScript，Python，ドリトル，DNCL などの各言語を同一環境で使用でき，実行方式も，プログラムの特性に合わせて Web ブラウザ，Web サーバ，組み込み機器での実行をサポートした．

研修教材におけるプログラミングに関する活動である第 3 章と第 4 章の演習で取り扱っているプログラムを Bit Arrow で動作させることが可能かを検証したところ，ほとんどのプログラムについてそのまま動かせる，あるいは同等の動作をする機器やプログラムを動作可能であった．それだけでなく，複数のデータを重ね合わせることにより，プログラミングを用いた自動的な分析を行ったり，アニメーションで結果をわかりやすく表示させたり，実際にネットワークを通じてデータベースを共有する活動を体験させたりするなど，発展的なプログラムも構築可能であることがわかった．

- **教員による生徒のつまずき発見の支援**

  教員は，多人数の生徒の活動を把握し，生徒のつまずき，すなわち手が止まっている，エラーが多発している，他の生徒と比べて進捗が遅い，などの状態を見つけて，適切なサポートを行う必要がある．

  Bit Arrow では，ユーザプログラムの保存や実行のたびに，ソースコードの名前・内容や実行結果をログとして保存する仕組みをもち，ログの内容を時系列に表示するログ閲覧機能をもつことで，つまずいている学習者を発見する支援ができるようになった．

  ログを利用することにより，エラーの発生率やエラーの復帰時間を集計できるだけでなく，ソースコードの内容も記録されており，変更履歴を追うことも可能であるため，エラーではないが思い通りの結果になっていない学習者の状態も把握できるようになった．具体的には，最後に実行したプログラムを完成品と考え，そのプログラムとそこに至るまでの各地点におけるプログラムとの差分を取り，同一行数の増減を元に進捗度を計算できる可能性が示唆された．これをもとにつまずきの自動的な検出を行うシステムの構築が可能と考えられる．

9.2 今後の課題

Bit Arrow は多くの高校で実際に利用されているものの，Bit Arrow を使用したことによる学習効果の検証は，8.1 で述べたようなプログラミング環境，その中でも「エラーの発生防止とエラー修正の支援」についてはある程度実証できているが,「データの収集・整理・分析」については．7 章で見たような教員研修教材に則って作成した，学習者向けの教材を充実させてこれから実践を行う予定である．

「教員による生徒のつまずき発見の支援」については，現状では教員がログ閲覧機能を使って，手元の PC で把握が可能であるが，つまずいている学習者を発見するには，学習者個別のログを詳細に確認する必要もある．このため．ログデータに含まれるソースコードの内容および，時間経過などの情報からつまずきの自動発見を行うことや，さらにはつまずいている学習

者への自動的なアドバイスを行う機能についても開発していく予定である.

謝辞

本研究を行うにあたり，多くの方々からご指導とご助言をいただきました．ここに深く感謝申し上げます．

はじめに，並木美太郎教授に感謝申し上げます．先生には筆者が明星大学の修士課程に在籍している当時から研究へのご助言をいただき，博士後期課程での研究の機会をいただきました．博士後期課程として東京農工大学工学府に進学後も，研究に関して様々なご指導をいただき，研究の方向性について多くの示唆をいただきました．ここに深く感謝申し上げます．

修士課程修了まで指導教員としてお世話になった明星大学の長慎也教授に感謝申し上げます．本研究は修士課程在籍中から続けているもので，筆者が東京農工大学の博士後期課程に進学した後も，先生にはシステムの開発や研究についてご指導をいただきました．深く感謝申し上げます．

共同研究者である大阪電気通信大学の兼宗進教授に感謝申し上げます．長らく情報科学分野における初等中等教育に携わってこられた経験から，本研究や論文の投稿にあたって多くのご助言をいただきました．また，本研究で開発したシステムの運用，改善や普及にもご尽力いただきました．ここに深く感謝申し上げます．

同じく共同研究者である四天王寺大学の間辺広樹教授に感謝申し上げます．先生が高等学校で情報科の教諭として勤務されていた際，高校教員の視点から現場における問題点をご指摘いただきました．また，本研究で開発したシステムを使った実践授業を行っていただき，本研究の評価を行うことができました．深く感謝申し上げます．

本研究は東京農工大学の並木研究室，明星大学の長研究室，大阪電気通信大学の兼宗研究室の共同研究として行いました．システムの運用などでお世話になった各研究室の学生の方々にも感謝申し上げます．

最後に，博士後期課程までの進学を許し支えてくれた両親に感謝します．博士後期課程への進学をはじめ，人生の決断を常に応援してくれたことで研究に集中することができました．両親の協力に心から感謝の意を表します．

参考文献

[1] 文部科学省. 高等学校の教育課程等に関する資料. https://www.mext.go.jp/b_menu/shingi/chukyo/chukyo3/053/siryo/__icsFiles/afieldfile/2015/06/05/1358298_02_01_02.pdf.

[2] 水越敏行, 村井純, 生田孝至ほか. 情報の科学. 日本文教出版, 2012.

[3] 赤堀侃司, 永野和男, 東原義訓ほか. 情報の科学. 東京書籍, 2012.

[4] 文部科学省. 小学校段階におけるプログラミング教育の在り方について. https://www.mext.go.jp/b_menu/shingi/chukyo/chukyo3/053/siryo/__icsFiles/afieldfile/2016/07/08/1373901_12.pdf.

[5] 尾崎光, 伊藤陽介. 小学校におけるプログラミング教育実践上の課題. 鳴門教育大学情報教育ジャーナル = Journal of information education, Naruto University of Education, No. 15, pp. 31–35, dec 2017.

[6] 中島敏. ビジュアル型言語からテキスト型言語への橋渡し教材の開発. 群馬高専レビュー, No. 37, pp. 65–74, apr 2019.

[7] 内田保雄, 玉城龍洋, 大西淳. アルゴリズム教育におけるブロックプログラミング言語の利用. 情報処理学会研究報告, Vol. 2021-CE-162, No. 20, pp. 1 – 6, nov 2021.

[8] 太田剛. 認知過程に考慮し，並び替えを目指す高校情報科のアルゴリズムプログラミング教育の開発と実践. Vol. 2021, pp. 85–92, aug 2021.

[9] Jeannette M. Wing. Computational thinking. *Commun. ACM*, Vol. 49, No. 3, p. 33–35, mar 2006.

[10] Jeannette M. Wing, 翻訳：中島秀之. Computational thinking 計算論的思考. 情報処理, Vol. 56, No. 6, pp. 584–587, may 2015.

[11] Fernando Alegre, John Underwoood, Juana Moreno, and Mario Alegre. *Introduction to Computational Thinking: A New High School Curriculum Using CodeWorld*, p. 992–998. Association for Computing Machinery, New York, NY, USA, 2020.

[12] Amber Settle, Baker Franke, Ruth Hansen, Frances Spaltro, Cynthia Jurisson, Colin Rennert-May, and Brian Wildeman. Infusing computational thinking into the middle- and

high-school curriculum. In *Proceedings of the 17th ACM Annual Conference on Innovation and Technology in Computer Science Education*, ITiCSE '12, p. 22–27, New York, NY, USA, 2012. Association for Computing Machinery.

[13] 文部科学省. 高等学校学習指導要領（平成30年告示）. https://www.mext.go.jp/content/1384661_6_1_3.pdf.

[14] 文部科学省. 高等学校学習指導要領（平成30年告示）解説情報編. https://www.mext.go.jp/content/1407073_11_1_2.pdf.

[15] 文部科学省. 高等学校情報科「情報i」教員研修用教材. https://www.mext.go.jp/a_menu/shotou/zyouhou/detail/1416756.htm.

[16] Association for Computing Machinary. Curricula recommendations. https://www.acm.org/education/curricula-recommendations.

[17] Computer Science Teachers Association. Cs standards. https://www.csteachers.org/Page/standards.

[18] 西田知博, 原田章, 中村亮太, 宮本友介, 松浦敏雄. 初学者用プログラミング学習環境 pen の実装と評価. 情報処理学会論文誌, Vol. 48, No. 8, pp. 2736–2747, aug 2007.

[19] 荻野哲男, 藤岡健史, 柳瀬大輔. 教育現場での実践に向けたプログラミング実行環境「ますめ」の試作. 情報処理学会研究報告, Vol. 2011-CE-111, No. 5, pp. 1 – 6, oct 2011.

[20] Stephen Cooper, Wanda Dann, and Randy Pausch. Alice: A 3-d tool for introductory programming concepts. *J. Comput. Sci. Coll.*, Vol. 15, No. 5, p. 107–116, April 2000.

[21] Nghi Truong, Peter Bancroft, and Paul Roe. A web based environment for learning to program. In *Proceedings of the 26th Australasian Computer Science Conference - Volume 16*, ACSC '03, p. 255–264, AUS, 2003. Australian Computer Society, Inc.

[22] Martin Quinson and Gérald Oster. A Teaching System To Learn Programming: the Programmer's Learning Machine. In *ACM Conference on Innovation and Technology in Computer Science Education 2015*, Vilnius, Lithuania, July 2015. ACM.

[23] Andrei Papancea, Jaime Spacco, and David Hovemeyer. An open platform for managing short programming exercises. In *Proceedings of the Ninth Annual International ACM Conference on International Computing Education Research*, ICER '13, p. 47–52, New York, NY, USA, 2013. Association for Computing Machinery.

[24] Anabela Gomes and António José Mendes. An environment to improve programming education. In *Proceedings of the 2007 International Conference on Computer Systems and Technologies*, CompSysTech '07, New York, NY, USA, 2007. Association for Computing Machinery.

[25] Paul Denny, Andrew Luxton-Reilly, Ewan Tempero, and Jacob Hendrickx. Codewrite: Supporting student-driven practice of java. In *Proceedings of the 42nd ACM Technical Symposium on Computer Science Education*, SIGCSE '11, p. 471–476, New York, NY, USA, 2011. Association for Computing Machinery.

[26] 田口浩, 糸賀裕弥, 毛利公一, 山本哲男, 島川博光. 個々の学習者の理解状況と学習意欲に合わせたプログラミング教育支援. 情報処理学会論文誌, Vol. 48, No. 2, pp. 958–968, feb 2007.

[27] 高橋 知希, 富永 浩之. 高校生への導入体験としての lego プログラミング演習の支援 – 高大連携の lego 講座における教育実践 –. 情報処理学会研究報告, Vol. 2013-CE-118, No. 18, pp. 1 – 7, feb 2013.

[28] 大見 嘉弘, 滑川 敬章, 永井 保夫. 情報系高校におけるセンサを利用したプログラミング教育の実践. 情報処理学会研究報告コンピュータと教育（CE）, Vol. 2012-CE-114, No. 5, pp. 1–7, mar 2012.

[29] 山本朋弘, 堀田龍也. 小学校理科での iot 教材のセンサーを活用したプログラミング体験に関する考察. 日本科学教育学会年会論文集, Vol. 43, pp. 441–444, 2019.

[30] 河村麻子. 小学生がプログラミング的思考を身近に感じる iot 学習教材. 第 81 回全国大会講演論文集, Vol. 2019, No. 1, pp. 365–366, feb 2019.

[31] 春日井優. 高等学校における機械学習についての指導の可能性と授業実践. 情報処理学会研究報告, Vol. 2018-CE-143, No. 19, pp. 1 – 8, feb 2018.

[32] 阿部百合. 情報の授業をしよう!：高等学校教科情報における python を利用した統計学習-問題解決能力の向上を目指して-. 情報処理, Vol. 60, No. 9, pp. 900–906, aug 2019.

[33] 林宏樹. 高等学校データサイエンスにおける導入教材の教育効果と有効性. 情報処理学会研究報告, Vol. 2021-CE-160, No. 7, pp. 1 – 5, may 2021.

[34] 林宏樹. 情報の授業をしよう !: 高等学校におけるデータサイエンスを基盤とした問題解決実践-ai 時代を切り拓くための創造力の育成を目指す-. 情報処理, Vol. 62, No. 10, pp. 560–565, sep 2021.

[35] 岡本弘之. 情報の授業をしよう!：情報 i を意識した授業をしよう！-紙飛行機制作を通し情報デザイン・データサイエンスに取り組む-. 情報処理, Vol. 61, No. 4, pp. 402–405, mar 2020.

[36] 才田聡子, 柏田元輝, 外村慶明, 尾花由紀, 北村健太朗, 古賀崇了. データサイエンス教育の題材としてのオープンデータ可視化 web アプリケーションの開発. 情報教育シンポジウム論文集, Vol. 2019, pp. 17–23, aug 2019.

[37] 藤本雄紀, 日下恭輔. 2 年次基礎ゼミナールにおける google colaboratory を利用したデータサイエンス教育の実践. 情報処理学会研究報告, Vol. 2021-CE-159, No. 28, pp. 1 – 8, mar 2021.

[38] 細合 晋太郎, 石田 繁巳, 亀井 靖高, 鵜林 尚靖, 福田 晃. 自律走行ロボットを用いた iot 開発 pbl に向けた教材開発. 組込みシステムシンポジウム 2015 論文集, Vol. 2015, pp. 40–45, oct 2015.

[39] 福地 健太郎, 茂木 大佑. センサによる計測を題材とした小学校高学年向け教材の開発とその活用事例. 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）, Vol. 2011-HCI-145, No. 3, pp. 1–8, oct 2011.

[40] 落合 秀也, 松浦 知史, 山内 正人. センサネットワーキングの新たな展開を目指して-live e! workshop in apng camp 活動報告-. 情報処理, Vol. 50, No. 1, pp. 55–63, jan 2009.

[41] 滑川 敬章, 落合 秀也, 山内 正人, 高岡 詠子, 中山 雅哉, 江崎 浩, 砂原 秀樹. 情報系高校における環境情報を計測・可視化する実用的なプログラミング教育の実践. 情報処理学会研究報告コンピュータと教育（CE）, Vol. 2012-CE-116, No. 16, pp. 1–8, oct 2012.

[42] 間辺 広樹, 大村 基将, 林 康平, 兼宗 進. 情報科教育における iot 学習環境の利用方法の検討. 情報教育シンポジウム 2016 論文集, Vol. 2016, pp. 98–105, aug 2016.

[43] 渕崇洋, 是常友哉, 白濱勝太, 上東亜佑稀, 上善恒雄. データサイエンス教育を目的とした慣性計測センサを用いた学習教材の検討. 情報処理学会研究報告, Vol. 2020-CE-153, No. 5, pp. 1 – 4, feb 2020.

[44] 西村智治, 青木辰徳, 富永浩之. 小コンテスト形式の初級 c 演習における教師支援 − 解答プログラムの提出状況と得点推移によるモニタリング機能 − . 情報処理学会研究報告, Vol. 2014-CE-119, No. 19, pp. 1 – 8, mar 2013.

[45] 太田翔也, 富永浩之. プログラミング演習における補助者の巡回指導のためのタブレット pc 上の支援ツール - 小コンテスト形式の初級 c 演習での実践におけるツールの操作ログの分析 - . 情報処理学会研究報告, Vol. 2014-CE-137, No. 1, pp. 1 – 8, nov 2016.

[46] 内藤広志, 齊藤隆. プログラミング演習のための進捗モニタリングシステム. 情報処理学会研究報告, Vol. 2008, No. 13(2008-CE-093), pp. 33 – 40, feb 2008.

[47] 長谷川伸, 松田承一, 高野辰之, 宮川治. プログラミング入門教育を対象としたリアルタイム授業支援システム. 情報処理学会論文誌, Vol. 52, No. 12, pp. 3135–3149, dec 2011.

[48] 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, Vol. 54, No. 1, pp. 330–339, jan 2013.

[49] 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, Vol. 55, No. 8, pp. 1918–1930, aug 2014.

[50] 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, Vol. 54, No. 12, pp. 2518–2527, dec 2013.

[51] 荻野哲男, 藤岡健史. 教育用プログラミング実行環境「ますめ」における活動記録を活用したフィードバック機能の設計. 情報処理学会研究報告, Vol. 2014-CE-124, No. 7, pp. 1 – 8, mar 2014.

[52] 松澤芳昭, 岡田健, 酒井三四郎. Programming process visualizer：プログラミングプロセス学習を可能にするプロセス観察ツールの提案. 情報教育シンポジウム 2012 論文集, Vol. 2012, No. 4, pp. 257–264, aug 2012.

[53] Cindy Norris, Frank Barry, James B. Fenwick Jr., Kathryn Reid, and Josh Rountree. Clockit: Collecting quantitative data on how beginning software developers really work. In *Proceedings of the 13th Annual Conference on Innovation and Technology in Computer Science Education*, ITiCSE '08, p. 37–41, New York, NY, USA, 2008. Association for Computing Machinery.

[54] 内閣府 地方創生推進室 ビッグデータチーム 経済産業省地域経済産業調査室. Resas - 地域経済分析システム. https://resas.go.jp/.

[55] 独立行政法人統計センター. Ssdse（教育用標準データセット）. https://www.nstac.go.jp/SSDSE/.

[56] 独立行政法人統計センター. e-stat 政府統計の総合窓口. https://www.e-stat.go.jp/.

[57] 堀越将之, 長島和平, 長慎也, 兼宗進, 並木美太郎. プログラミング学習環境「bit arrow」における採点支援機能. 情報処理学会研究報告, Vol. 2018-CE-144, No. 9, pp. 1 – 8, mar 2018.

[58] 長慎也, 長島和平, 堀越将之, 兼宗進, 並木美太郎. オンラインプログラミング環境 bit arrow を用いた c 言語プログラミングの授業実践. 情報教育シンポジウム論文集, Vol. 2017, No. 17, pp. 121–128, aug 2017.

[59] 兼宗進, 御手洗理英, 中谷多哉子, 福井眞吾, 久野靖. 学校教育用オブジェクト指向言語「ドリトル」の設計と実装. 情報処理学会論文誌プログラミング（PRO）, Vol. 42, No. SIG11(PRO12), pp. 78–90, nov 2001.

[60] 独立行政法人大学入試センター. センター試験用手順記述標準言語 (dncl) の説明. http://www.dnc.ac.jp/albums/abm.php?f=abm00004841.pdf&n=H23_dncl.pdf.

[61] 長慎也, 岸本有生, 長島和平, 兼宗進, 並木美太郎. Bit arrow における組み込み機器実行機能と，収集データの共有・分析機能との連携. 情報処理学会研究報告, Vol. 2021-CE-161, No. 7, pp. 1 – 7, oct 2021.

[62] 長島和平, 堀越将之, 長慎也, 間辺広樹, 兼宗進, 並木美太郎. プログラミング学習支援環境 bit arrow の教員支援機能の設計と試作. 情報教育シンポジウム論文集, Vol. 2017, No. 18, pp. 129–136, aug 2017.

[63] 浦上理, 長島和平, 並木美太郎, 兼宗進, 長慎也. プログラミング学習者のつまずきの自動検出. 情報処理学会研究報告, 2020-CE-154(4), pp. 1 – 8, mar 2020.

[64] Ace. Ace editor. https://ace.c9.io/.

[65] 長島和平, 長慎也. Tonyu system 2 ゲーム制作を通じたプログラミング学習に適したフレームワーク. 情報処理学会研究報告, Vol. 2015-CE-129, No. 2, pp. 1 – 8, mar 2015.

[66] 長島和平, 長慎也, 兼宗進, 並木美太郎. プログラミング学習環境 bit arrow でのセンサデータ収集と可視化ライブラリ. 情報処理学会研究報告, Vol. 2018-CE-143, No. 8, pp. 1 – 8, feb 2018.

[67] 本多佑希, 大村基将, 長慎也, 久野靖, 並木美太郎, 兼宗進. Dolittle のオンラインプログラミング環境の開発. 情報処理学会研究報告, Vol. 2016-CE-134, No. 25, pp. 1 – 5, feb 2016.

[68] 本多佑希, 長慎也, 長島和平, 大村基将, 島袋舞子, 並木美太郎, 兼宗進. Javascript 版ドリトルのタブレットでの利用可能性の提案. 情報処理学会研究報告, Vol. 2016-CE-136, No. 6, pp. 1 – 6, oct 2016.

[69] 本多佑希, 兼宗進. ブラウザ上で動作する dncl 学習環境「どんくり」の開発. 情報処理学会研究報告, Vol. 2018-CE-147, No. 10, pp. 1 – 4, nov 2018.

[70] 岸本有生, 本多佑希, 兼宗進. 計測データのクラウド保存と分析が可能な iot 学習教材の提案. 情報教育シンポジウム論文集, Vol. 2020, pp. 139–145, dec 2020.

[71] Micro:bit Educational Foundation. micro:bit. https://microbit.org/.

[72] 兼宗進, 並木美太郎, 長慎也, 長瀧寛之, 長島和平, 小林史弥, 本多佑希, 林康平. Bit Arrow で始める プログラミング事例集 Vol.3. 東京書籍, 2019.

[73] 飯塚康至, 長慎也. 楽しく学ぶ C 言語. 技術評論社, 2020.

[74] 長島和平, 長慎也, 間辺広樹, 兼宗進, 並木美太郎. Web ブラウザを用いたプログラミング学習支援環境 bit arrow の設計と評価. 情報処理学会論文誌教育とコンピュータ（TCE）, Vol. 4, No. 1, pp. 57–69, feb 2018.

[75] 長島和平, 長慎也, 間辺広樹, 兼宗進, 並木美太郎. Web ブラウザを用いたプログラミング学習支援環境 bit arrow の設計と評価. 情報処理学会研究報告, Vol. 2017-CE-138, No. 2, pp. 1 – 8, feb 2017.

[76] Kazuhei Nagashima, Shinya Cho, Masayuki Horikoshi, Hiroki Manabe, Susumu Kanemune, and Mitaro Namiki. Design and development of bit arrow: A web-based programming learning environment. In *Proceedings of the 10th International Conference on Education Technology and Computers*, ICETC '18, p. 85–91, New York, NY, USA, 2018. Association for Computing Machinery.

[77] 長慎也, 長島和平, 間辺広樹, 兼宗進, 並木美太郎. 初学者向けプログラミング授業における活動ログの評価支援機能. 情報処理学会研究報告, Vol. 2017-CE-138, No. 4, pp. 1 – 8, feb 2017.

[78] 長島和平, 長慎也, 間辺広樹, 並木美太郎, 兼宗進. Jslesson - 高校生向け javascript 学習環境. 情報処理学会研究報告, Vol. 2016-CE-134, No. 16, pp. 1 – 9, feb 2016.

[79] 間辺広樹, 長島和平, 並木美太郎, 長慎也, 兼宗進. 自宅で行うオリジナル作品制作の学習効果と問題点〜オンラインプログラミング学習環境を用いて〜. 情報教育シンポジウム論文集, Vol. 2017, No. 15, pp. 101–109, aug 2017.

[80] 長慎也, 浦上理, 長島和平, 兼宗進, 並木美太郎. 完全オンライン授業における php プログラミング実践と実習環境. 情報処理学会研究報告, Vol. 2021-CE-158, No. 10, pp. 1 – 9, feb 2021.

[81] 長慎也, 浦上理, 澤本直輝, 市石舜也, 長島和平, 兼宗進, 並木美太郎. プログラミングにおけるログとつまずきの相関 − slack のサポート履歴を利用した「15 分ルール」の妥当性分析. 情報教育シンポジウム論文集, Vol. 2021, pp. 14–21, aug 2021.

[82] 浦上理, 長島和平, 並木美太郎, 兼宗進, 長慎也. Slack を用いたプログラミング学習者のつまずきの検出支援. 情報教育シンポジウム論文集, Vol. 2021, pp. 9–13, aug 2021.