プログラミング初学者用学習支援Webシステム ECLATの実装と評価

高原 渉*　小高 知宏*　黒岩 丈介**　諏訪 いずみ***　白井 治彦****

Implementation and Evaluation of The Programming Environment ECLAT for Novices

Wataru TAKAHARA*,　Tomohiro ODAKA*, Jousuke KUROIWA**,
Izumi SUWA** and Haruhiko SHIRAI***

(Received February 2, 2018)

In this paper, we developed programming environment using web system.Programming education are necessary to introduce an editor and a compiler to perform C language program learning. There are cases where complicated operations are required when executing.Therefore, we developed the system ”ECLAT” for improving the efficiency of these tasks.

In this system, students can learn programming using the web browser. Our system does not need to introduce editor and compiler. and thus students can learn instantly. Also, our system is compatible with both PC and smartphone, many students can use it.

In order to evaluate the system, we experimented by conducting programming education for students using our system. . As a result, we confirmed that our system can support programming education.

Key words : Web system,Programming environment,Learning support system

1. はじめに

現代では高度な情報化社会が進んでおり、実用的な情報教育において基本的な技能のひとつであるプログラミング能力の獲得が必要になってきている[1]。大学において、多くの学生は大抵の場合プログラミング初心者である。プログラミング初学者に対して、プログラムの動作について理解させることはとても難しい[2]。プログラミング初学者である学生に対して、教員がそれぞれの学生に丁寧な指導を行うという方法は望ましいと考えられる。しかし、プログラミング演習における教員やTA（ティーチングアシスタント）の数および演習の時間的な制限によって、学生全員に対して丁寧な対応ができない場合も出てくる。そのため、学生の中ではプログラミングについて分からない点や疑問点が放置されるようになってしまい、学習速度に遅れが出てしまう場合がある。[3]

またプログラミング演習における事前準備として、エディタやコンパイラ等のインストールといったプログラミングに必要な環境を構築しなければならない。その上プログラムの実行方法についてもターミナルのコマンド入力を用いた複雑な方法で行う場合があり、プログラミング本来の学習以外の事に気を取られてしまい授業の効率が悪くなってしまうと考えられる。

これらの問題を解決するためには、プログラミング初学者を支援するためのシステムが必要となる。プログラミング演習における準備や学習に必要な操作

* 大学院工学研究科 原子力・エネルギー安全工学専攻
** 大学院工学研究科 知能システム工学専攻
*** 工学部技術部
* Nuclear Power and Energy Safety Engineering Course, Graduate School of Engineering
** Human and Artificial Intelligence Systems Course, Graduate School of Engineering
*** Technical Division

が複雑であるという問題について、それらをできるだけ簡略化するシステムが必要である。

そこで、本研究では「ECLAT」という手軽にプログラムを実行可能な Web システムの設計、実装を行った。ECLAT は Web ブラウザ上で動作するため、特別なソフトウェアのインストールや設定の必要がないように設計されている。そのためスムーズに学習を行うことが可能となると期待できる。

本研究では、ECLAT を用いたプログラミング演習を大学 1 年次生を対象に行い、学生を対象にアンケートを集計しシステム評価を行った。また、プログラミングツールとして適切な速度で応答するかどうかを検証するため、システムのターンアラウンドタイムの測定を行った。それらの結果を分析することで、スムーズなプログラミング学習が可能であるかどうかを検証する。

2. 既存のシステムの問題点と解決法

Web ブラウザ上でプログラムを実行することができるサービスとして、「 ideone[4] 」や「 codepad[5] 」といったシステムが挙げられる。これらのシステムは入力されたソースコードをコンパイルし実行結果を表示する機能を持つが、図 1 に示すようにプログラムの実行前に標準入力に指定する文字列をあらかじめ入力しなければならないというバッチ処理的な入出力といった仕様を持っている。すなわち、通常のプログラミングの入出力では「あなた名前はなんですか？」という出力を確認してから、標準入力に自分の名前を入力するという対話的な入出力を行うが、これらのシステムでは名前を入力するための問いが出力されることを見越して、既に入力を終えてなければならない。また条件分岐や繰り返し処理によって入力が要求される内容や回数が学習者の入力によって変わるプログラムを実装した場合、事前に標準入力を指定する方法において達成できず、直感的な入出力ができない場合が考えられる。

そのため、ideone や codepad では入出力に関してプログラミング初学者が直感的に学習する上では不向きであると考える。よって本研究では、初学者が直感的にプログラミング学習を可能とする環境の構築を目標に、対話的な入出力を実現するシステムや実行結果からソースコードへフィードバックしやすいシステムの開発を行った。

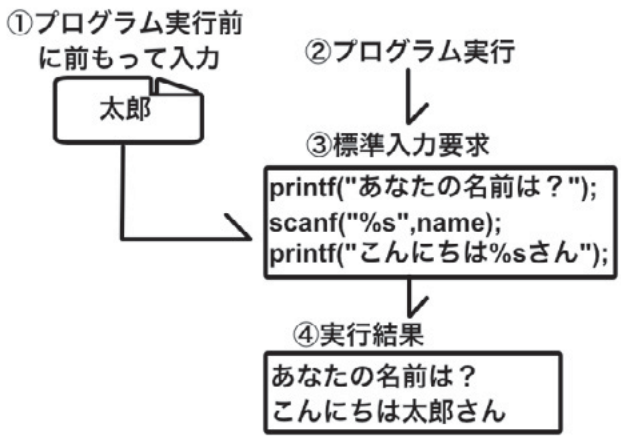

図 1 バッチ処理的な入出力

3. C 言語学習 Web アプリケーション「ECLAT」

ECLAT は、Web ブラウザ上で実行可能な C 言語プログラミング学習アプリケーションである。簡単な操作でプログラミング学習を行う事ができ、コンソール上で実行する場合と同じような対話的な入出力を行うことが可能である。プログラミングを行うために必要な環境の構築や複雑な操作を取り除くことで、スムーズな学習の実現を目的としている。また、PC とスマートフォンの両方に対応しているので、多くの学生が利用することができる。

3.1 システム構成

ECLAT のシステムは、クライアント側でソースコードの記述や実行結果などの表示、サーバ側でコンパイルやプログラムの実行などの処理を行う。ユーザがプログラムを記述し、実行するまでの流れを図 2 に示す。プログラムの記述を終えたら、プログラムを実行するためにソースコードの内容をサーバに送信する。次に受け取ったデータからファイルを作成し、コンパイルが成功した場合プログラムをサーバ側で実行する。実行中、C 言語における scanf 関数といった標準入力の要求を検知した場合、クライアント側に対して入力要求を行う。全ての処理が完了したら終了処理を行い、クライアント側に実行結果を表示する。またコンパイルに失敗した場合、エラーの内容をクライアント側に送信し、実行できないことを示す。

3.2 システム実装

クライアント側のシステムは主にプログラムの記述と実行結果の表示機能を持っており、HTML、

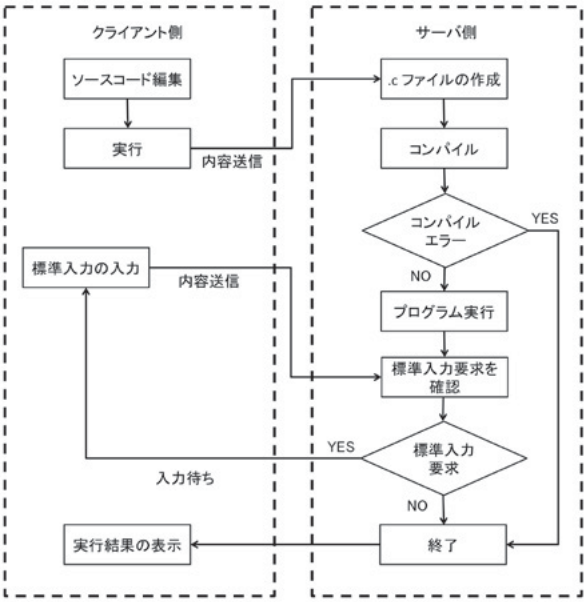

図 2 ECLAT のクライアント側とサーバ側の処理

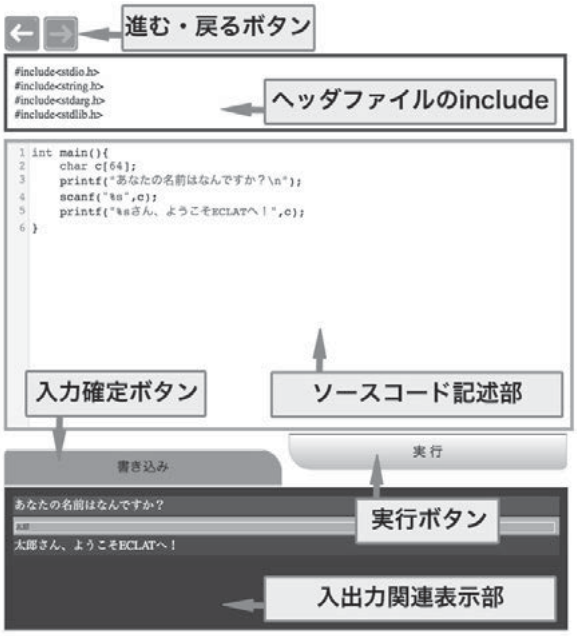

図 3 ECLAT の画面説明

javascript、jQuery を用いて実装されている。またサーバ側ではプログラムのコンパイルや実行を行う機能を持っており、PHP を用いて実装されている。ECLAT の特徴として、C 言語における scanf 関数といった標準入力を利用するプログラムを実行した場合、コンソール上で実行する挙動とほぼ同じ働きをする対話的な入出力が可能となっている。

利用可能なプラットフォームとして、PC やスマートフォンなど Web ブラウザを扱えるほとんどの端末をサポートしている。ページを表示するためのスタイルは PC 版とスマートフォン版でそれぞれ最適化されており、利用しやすいように設計した。

3.2.1 画面設計

図 3 に、ECLAT のページにアクセスした際に表示される画面を示す。主な要素はソースコード記述部、入出力関連表示部、実行ボタン、入力確定ボタン、戻るボタン・進むボタンとなっており、この一画面の中にプログラムの記述や、実行、実行結果の表示といった基礎的な C 言語プログラムが可能であるように設計されている。

- ソースコード記述部

  ユーザーがソースコードを記述するテキストエリアである。本システムの仕様上、強制的にインクルードされるヘッダファイルはソースコード記述部の上に記載されている。コンパイルエラーを検知した場合、エラーが起こっている行の箇所を、赤く強調する機能を持つ。図 4 に、コンパイルエラーが発生するソースコードを記述し、ECLAT で実行した際に表示される例を示す。この図の上部に示すプログラムは、整数型 a を宣言しているが、誤って宣言していない b という変数を呼び出している。このプログラムをサーバ側でコンパイルすると、図 4 の下部に示すように、コンパイルエラーのログが出力される。図中の矢印で示されているように、この出力の中にはエラーが発生している箇所である行番号が示される。本システムでは、コンパイラが示すエラーが発生している行番号とユーザの記述するエディタの行番号と関連付けて強調する機能を持つ。この機能により、ユーザーはエラーの箇所を素速く特定することができ、よりスムーズな学習に取り組むことができると考えられる。

  また、ソースコードを編集するエディタ部には CodeMirror と呼ばれるプログラムの記述をサポートする様々な機能を持つ Javascript コンポーネントを用いている。本システムで導入している CodeMirror の機能については、次の節で説明する。

- 入出力関連表示部

  プログラム実行後、実行結果の標準出力や例外出力、コンパイルエラーの内容を示すエリアである。出力される文字列は一行ずつ最下部に一定間隔で挿入され、一定以上出力されると古

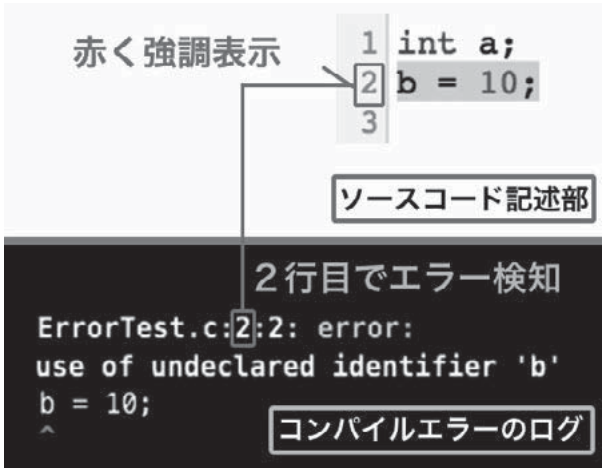

図 4 エラー行の強調

い出力データは上部へと隠れるようになっている。隠れたデータはスクロールすることによって確認することができる。

標準入力を用いる場合（scanf 関数の使用時）、それまでの標準出力が完了した後、入力欄が挿入される。入力欄に文字列を入力した後、「入力確定ボタン」もしくは Enter キーを押下することで、次の処理に進むことができる。

- 実行ボタン

  ソースコードの記述を終えたら、このボタンを押すことでソースコード記述部の内容をサーバに送信し、実行結果を表示する。実行の開始から実行結果の表示までである程度の時間が必要となる。したがって連続でこのボタンを押すことで、プログラムの実行中に繰り返しサーバに対して実行要求を行ってしまうため、負荷がかかってしまう危険性が考えられる。そこで、サーバにデータを送信し、サーバから実行結果を受け取るまでは、このボタンを無効化し、押せなくなるように実装した。

- 入力確定ボタン

  プログラムの実行中、標準入力の要求が発生した場合、実行結果表示部に入力欄が挿入される。ユーザはここに文字列を入力したのち、この入力確定ボタンを押すことでサーバに入力情報を送信することができる。入力確定ボタンの他に、キーボード上の「Enter キー」でも同様の処理を実行することができる。また、標準入力の要求がされていない状態の場合は押せないように無効化している。

- 戻るボタン・進むボタン

  CodeMirror のコードエディタの機能を用いることで、ソースコード記述部の書き込み履歴を参照し、ソースコードを前の状態に戻したり、進めたりすることができる。初期状態ではどちらのボタンも無効化している。戻るボタンについて、ソースコードが変更されたタイミングでボタンの押下が可能になり、これ以上戻せない状態になった場合はボタンが無効化される。進むボタンについて、戻るボタンが押下されたタイミングで押下可能になり、これ以上戻れない場合はボタンが無効化される。

3.3 サーバ側の実装

サーバ側では、MAC アドレスと関連付けたユーザの管理や、ユーザから送信されたソースコードのコンパイルやプログラムの実行を行い、実行結果を送信する役割を持つ。本システムでは、実装に PHP を用いて構成した。

3.3.1 ユーザ管理

図 5 に示すように、本システムではユーザの使用する端末の MAC アドレスをもとに、ユーザ毎に作業フォルダを割り当てる。この処理を行うことで以下の作業を実現できる。

- コンパイル・実行の処理の並列化
- テキストデータの保持

コンパイル・実行の処理の並列化について、一つのソースコードのファイルや実行ファイルで複数のユーザの処理を行うと、処理が混雑してしまって期待通りの実行結果にならない場合がある。ユーザ毎に作業フォルダを分けることで、円滑にコンパイル・実行の処理を行うことが可能になる。　テキストデータの保持について、保存されている作業フォルダと同じ MAC アドレスからログインされた場合、前回編集していたソースコードの情報を送信することができる。そのため、誤って作業中に Web ブラウザを閉じてしまった場合でも、スムーズに作業を再開することが可能である。

ユーザ毎に管理しているファイルを以下に示す。

- ECLAT 用編集ファイル
- 本システムで機能するソースコードファイル

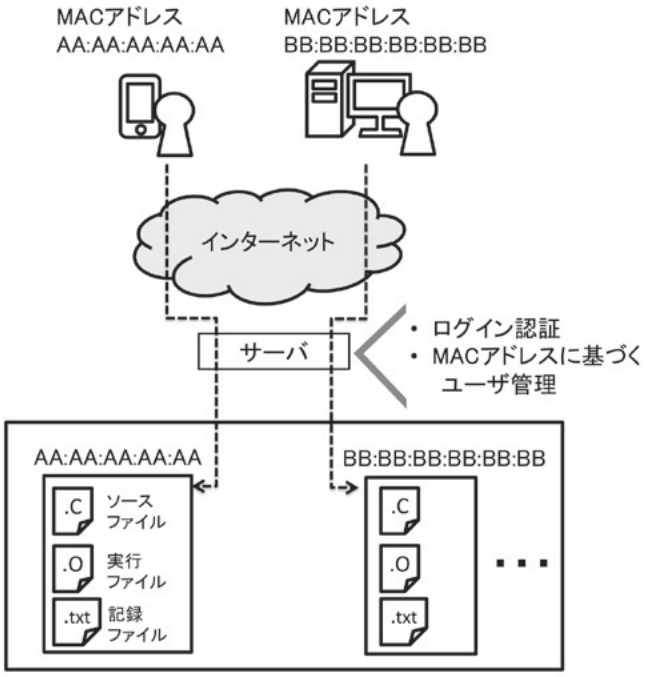

図 5 MAC アドレスに紐付けたユーザ管理

- 実行ファイル
- 出力用テキストファイル
- 入力用テキストファイル
- プログラム実行状態管理ファイル

ECLAT 用編集ファイルは、ユーザがブラウザ上で編集する部分の内容をそのまま記録している。ログイン要求のあったユーザの持つ MAC アドレスと作業フォルダの情報が一致した場合、このファイルの内容をクライアント側に送信する。送信されたデータはソースコード記述部に表示され、前回操作した状態から編集を始めることができる。

本システムで機能するソースコードファイルについて、ECLAT 用編集ファイルをコンパイルし実行した場合、特に入出力に関してシステムの仕様上機能しない。そのため、サーバ側でこのファイルを書き換えることで、本システムで機能するソースコードファイルにする必要がある。

実行ファイルは、コンパイルが成功した場合に生成される。実行したタイミングで時間を計測し、一定時間を超えた場合はプログラムを中断することで、サーバに対する負担軽減を図る。その際はクライアント側にそれまで出力された結果の後に「タイムアウト」と表示される。

出力用テキストファイルについて、プログラムを実行する際のコマンドに対して、標準出力や例外出力をこのファイルに書き込む。また、コンパイルエラーが発生した場合も、エラーの内容をこのファイルに記録する。このファイルの内容をクライアント側に送信することによって、実行結果を表示することが可能になる。

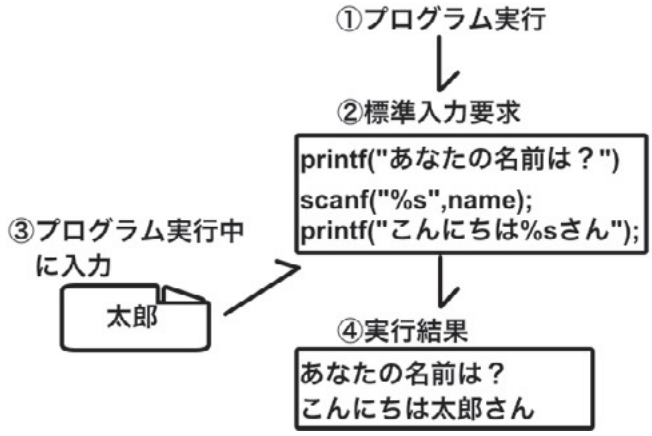

図 6 対話的な入出力

入力用テキストファイルは、クライアント側から送信された標準入力に対する入力の内容を記録するファイルである。入力待ち状態であるプログラムが再開されると、このファイルを読み込むことで処理を進めることができる。

プログラム実行状態管理ファイルは、プログラムが実行され、どのような状態であるのかを示すファイルである。プログラム実行中、このファイルを参照することで標準入力の確認を行うことができる。

3.3.2 対話的な標準入出力

標準入力とは、C 言語における scanf 関数などのようにプログラム実行中にユーザからの入力を受け付ける機能のことである。図 6 に示すようにプログラムの実行中に標準入力に指定する文字列を入力できるという対話的な入出力ができ、直感的な操作を実現する。

図 7 に標準入力を含むプログラムを実行した時の処理の流れを示す。基本的に、標準入力を用いるプログラムはユーザが何を入力すればよいのかを提示する文字列を先に出力しておく必要がある。そのため、図 7 で示すプログラムは、最初に名前の入力を促す文字列を表示し、その後ユーザに対して標準入力を要求するものを使用した。プログラム実行中、サーバは常に実行状態を管理するファイルを監視する。scanf

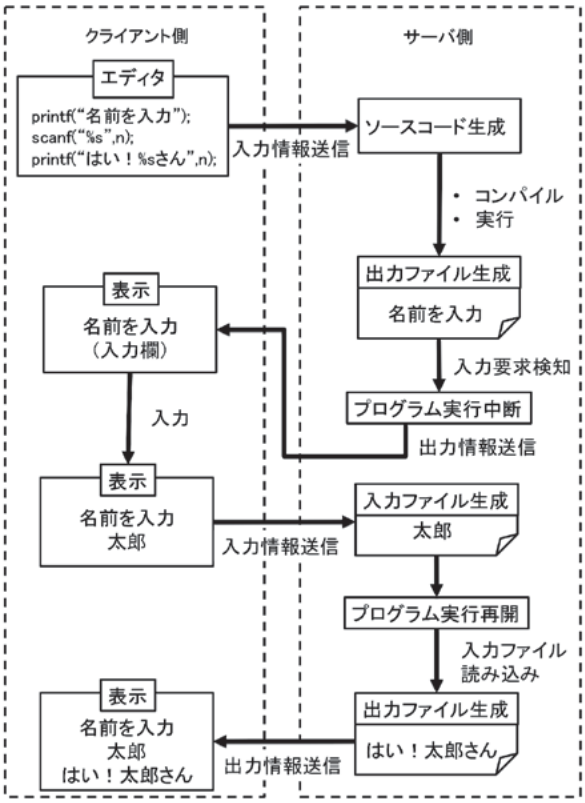

図 7 標準入出力における処理の手順

関数といった標準入力を扱う関数を用いると、実行状態から入力要求状態に変化し、プログラムを一時停止する。入力要求状態を検知すると、クライアント側に入力要求を通知する。クライアント側から入力情報が送信されたらプログラムが再開され、入力情報が記録されたファイルを読み込み、実行状態に戻る。

プログラムが標準入力を要求しているかどうかの検知方法について、あらかじめコンパイル前にソースコードを追記し、scanf 関数の機能の拡張を行う。新たに独自に定義した scanf 関数の処理では、「入力要求」状態を示すファイルの作成を行う。プログラム実行と同時に、その状態ファイルを定期的に確認する処理を並列に実行する。

プログラムを一時停止する方法として、kill コマンドの一時停止シグナルを用いる手法で実装した。プログラム実行時、そのプログラムのプロセス ID をファイルに記録しておく。プログラムを一時停止するタイミングでそのプロセス ID をもとに、kill コマンドを実行する。クライアント側から入力情報が送信されプログラムを再開するタイミングでは、同じように kill コマンドの一時停止からの再開シグナルを用いて再開を行う。

クライアント側からの入力情報プログラムが読み込む方法として、コンパイル前にソースコードを追記する手法を用いた。クライアント側から標準入力として送られてきた情報は、サーバ側のファイルに記録される。プログラムが入力待ち状態の一時停止から再開された後、標準入力に用いる文字列が記録されたファイルの内容を読み込む。

これらの手法を用いることで、ユーザ視点では通常の C 言語学習で扱うような本来の scanf 関数と同じ使い方ができ、内部の仕様に気にすること無く対話的な入出力のできるプログラミングに取り組むことができる。

入力関数について、scanf 関数の他にも fgets 関数など様々存在するが、本システムでは仕様として scanf 関数のみ扱えるものとなっている。しかし、この手法を応用することによって様々な出力関数に対応させることが可能である。

4. 実験

本研究では、142 名の学生に ECLAT を用いた C 言語プログラミング演習を行い、システム評価を行った。評価方法として、アンケートの集計や、システムの応答速度（ターンアラウンドタイム）の測定を行い、ECLAT のプログラムツールとしての優位性を検証する。

4.1 ターンアラウンドタイムの検証

ターンアラウンドタイムとは、プログラムを記述したのち、処理を開始してから次の要求の受け入れが可能になるまでの時間のことである。本研究におけるターンアラウンドタイムの定義は、ユーザによって「実行ボタン」が押下されてから実行結果が表示し始めるまでの時間とする。また、scanf 関数の使用によって標準入力が要求される場合のプログラムのターンアラウンドタイムについては、「実行ボタン」が押下されてから最初の入力欄が表示され始めるまでの時間とする。時間を計測するため、クライアント側の処理としてターンアラウンドタイムを計測したデータをサーバ側に送信する機能を実装した。この機能によって、ユーザがプログラムを実行するたびに実行結果の表示にどれだけの時間がかかったのかを確認することができる。

図 8 に、ユーザから集計したターンアラウンドタイムを新しいものから順に 2500 個抽出したものをヒストグラムにして示す。この結果について、最頻値として 100 ミリ秒から 200 ミリ秒のターンアラウンドタイムが 901 回として現れていることが分かった。

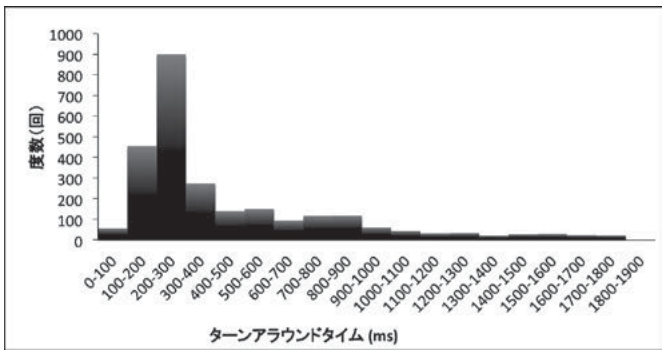

図 8 ターンアラウンドタイムの結果

また次に多く計測されたされたターンアラウンドタイムとして、先程よりも速い値である 100 ミリ秒から 200 ミリ秒で 456 回であった。なお、全体の平均は 560 ミリ秒であった。

4.2 ユーザアンケート

本実験では、ECLAT を用いた講義を受けた学生に対し、システムの利便性や学習意欲についての調査としてアンケートの集計を行った。対象とした学生は、福井大学物質生命化学科の 1 年生 102 名と、福井工業大学電気電子工学科の 1 年生 40 名の計 142 名である。アンケートは学生が ECLAT を用いた学習についてどのように感じたかについて、選択肢を用意した。また、自由に意見を記述できる項目も用意した。表 1 にアンケートの内容を示し、集計結果を表 2 に示す。質問 1 以外の質問については 4 段階の評価で回答を行う。

質問 1 について、本システムを利用した経験があるかどうかの確認を行った。結果、全ての学生が「はい」と回答したことから、利用率の高さが伺える。

質問 2 では本システムのインターフェイスについて、適切なサイズや色に設定されているかどうかの検証を行った。結果は 8 割もの学生が「見やすい」と答えたことからおおむね視覚的な問題はないものと考えられる。

質問 3 は本システムを利用する上で的確に扱えスムーズに学習を進められるかどうかを検証した。「あった」と答えた学生がそれぞれ 3 割を示していることから、多くの学生がどう操作すればいいのか分からないという場合があることが分かった。

質問 4 では本システムの持つエラー行の強調機能およびコンパイルエラーの出力に関して、学生が適切にエラーに対処できているかどうかを確認した。自信を持って簡単だったと回答した学生はわずか 3 割程度であった。プログラミングにおいて、コンパイルエラーは基本的に初学者ほど修正する作業が大変なので、そのような原因が結果に反映されていると考えられる。

質問 5 では、プログラムが実行されてから実行結果を表示するまでの時間、つまりシステムのターンアラウンドタイムについて、学生がどの程度快適に感じるかを検証している。結果は「早い」と感じた学

表 1 アンケート内容

質問 1　「ECLAT」を用いたプログラミング学習の経験があるか
（以下、「ある」と答えた者を対象に回答）
質問 2　ボタンやエディタ部、実行結果表示部のパーツについての全体的な見やすさ
質問 3　使い方が分からなかったり操作を間違えたりすることがあるか
質問 4　コンパイルエラーが発生した際、間違いの箇所がどこにあるかを見つけることができたか
質問 5　プログラムを実行してから結果が表示されるまでの時間について、どのように感じたか
質問 6　「ECLAT」を用いて学習したことで、C 言語をもっと学びたいと思うようになったか
質問 7　プログラミングの学習について、これからも「ECLAT」を使って学習したいと思うか
質問 8　その他、自由記入

表 2 アンケート結果（142 名回答）

質問 1	ある			ない
	100%			0%
質問 2	見やすい			見辛い
	19%	64%	14%	3%
質問 3	無かった			あった
	29%	36%	31%	4%
質問 4	簡単			難しい
	24%	46%	25%	5%
質問 5	早い			遅い
	46%	39%	13%	3%
質問 6	思う			思わない
	12%	57%	19%	11%
質問 7	思う			思わない
	16%	53%	19%	12%

生がほとんど占めている。この結果から、プログラムの実行確認がテンポよく行われ、スムーズな学習が実現されていることが分かる。

質問 6 および質問 7 では、本システムについての満足度を評価している。結果はどちらもほぼ同じ割合となっており、「とても思う」「やや思う」と答えた学生が 7 割を占めている。また、ほどんどの学生がこの 2 つの質問に対して同じ回答を選択していることから、多くの学生はこれからも ECLAT を用いて C 言語プログラミングをやっていきたいと感じていることが分かる。

5. 考察

本研究で開発した Web 上で C 言語プログラミングが実行可能なアプリケーション「ECLAT」の評価を行った。評価方法として、本システム ECLAT を学生に利用してもらうことで、システムの応答速度を表すターンアラウンドタイムの計測を行う。また、ECLAT を用いた学習を通して感じたことをアンケートによって集計を行った。結果、ターンアラウンドタイムやページの見やすさについて、学生にとって快適であるという評価が得られた。

また、本システムを利用したことによって C 言語プログラミングについての学習意欲の向上が見られた。これは、特別な準備や操作を省くことで学生にかかる負担を軽減することで、プログラミング本来の面白さに触れることが可能になったことが一つの要素であると考えられる。

ECLAT を使っていて、よく操作を間違えてしまう学生が全体の 7 割ほど現れた原因として、ECLAT の使い方を解説したページを用意しなかったことが原因となっていると考えられる。操作方法は、プログラミング演習の際に口頭で行ったが、聞き逃してしたり忘れたりしてしまった学生のために、丁寧な操作方法を示したページの用意が重要であることが分かった。

また、コンパイルエラーの発生の際、エラーの原因を見つけることに対して難しいと答えた学生が 7 割以上占めていた。プログラミング初学者はエラーを発見してそれを訂正することはとても難しいので、ECLAT にはコンパイルエラーの出力を解析してクライアント側のソースコードエディタにエラーと思われる行を赤く強調する機能を実装した。しかし、コンパイラの出力するエラー行の出力リストはプログラムの根本的なエラーを示さない場合があるので、本来修正すべき行を赤く強調できない場合がある。このことが原因となり、コンパイルエラーを修正する作業を難しく思った学生が多く占めたのではないかと考えられる。

6. まとめと今後の課題

本システムは Web ブラウザ上で動作するので、開発環境を整える負担が少なくスムーズに学習に取り組むことが可能となる。また、プログラムのコンパイルや実行の操作を簡単に行うことができるため、プログラミング学習に対して集中的に学ぶことができる。

今後の展望について、ECLAT はユーザアンケートの結果から、ECLAT の使い方について分からない点やつまずいてしまった点が見られた。これは ECLAT の使い方を口頭のみで伝えていたことが原因であると分析し、今後は使い方を丁寧に示したページを実装することを目標とする。またコンパイルエラーが発生した場合に表示されるエラーリストとして、出力される文字列が全て英語であるので、英語に不慣れなプログラミング初学者にとっては混乱を招く原因になっている要因があると考える。そのため、英語のエラー文を日本語で分かりやすく表示するための機能の実装が必要になると考えられる。

本研究で扱うプログラミング言語について、コンパイラ言語である C 言語を対象にしている。C#や Java といった他の言語を扱いたい場合、本システムを応用することで簡単に実装可能である。そのため他の言語についてのプログラミング演習で ECLAT を用いることを考えた場合についても、対応可能であると考えられる。

参考文献

[1] 田口浩. 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. 48-2, pp. 958–968, Feb 2007.

[2] A. Oram. Making Software: エビデンスが変えるソフトウェア開発, pp. 107–120. Theory in practice series. オライリー・ジャパン, 2011.

[3] 耕大山本, 将寿春原, 克紀大金, 勝一中村, 節雄横山, 庸造宮寺. エラー要因事例ベースの動的学習手法を導入した c 言語教育システムの開発と基礎的評価. 電子情報通信学会技術研究報告. ET, 教育工学, Vol. 108, No. 146, pp. 67–72, jul 2008.

[4] ideone. https://ideone.com.

[5] codepad. http://codepad.org.