K-019

初心者向けエラーメッセージの明瞭化とプログラミングスキルに基づく視線の比較

Clarification of Error Messages for Beginners and Comparing Gaze Based on Programmer's Skill

藤坂 直輝† 山田 光穂† 石井 英里子‡ 星野 祐子†
Naoki Fujisaka Mitsuho Yamada Eriko Ishii Yuko Hoshino

1. はじめに

プログラミング経験の浅い初心者にとって，コンパイルエラーメッセージの内容を正確に理解し，ソースコードを修正することは難しい．特に，コンパイラが明瞭に提示できないエラーに対して，プログラマは何を修正すればよいか混乱することも多い．このようなエラーは，ある程度のプログラミングスキルを持った中級者，上級者でも，修正に時間を要することも少なくない．初心者にとって，わかりにくいエラーは学習の障害になり，挫折する原因にもなりうる．そこで，これらのエラーメッセージを初心者にわかりやすく提示し，学習効率の向上を目指すシステムの開発を行っている．これまでのシステム[1]は，C 言語のプログラムをコンパイルし，プログラムを実行する機能，入力されたプログラムの全角文字，複数言語の誤りを検知する機能，文法を確認し，正規の文法を表示させる機能を有した Web アプリケーションである．今回新たに，include 文の記述忘れによる問題への支援機能を実装した．また，支援機能を増やすために，動的に Web ページを変化させることとした．加えて，学習者がより誤りに気づきやすいインタフェースを開発するために，コンパイルエラー修正時の視線計測を行った．

2. 関連研究

ソフトウェア開発において，エラーの理解と修正は初心者だけでなく，全プログラマにとって，課題となっており，これまでにさまざまな支援方法が提案されてきた．西村ら[2]は，英語のエラーメッセージを日本語に対応させた和訳データベースを作成，コンパイルエラーから修正案を提案するシステムを提案した．後藤ら[3]はコンパイルエラーに補助メッセージを追加，レーベンシュタイン距離を使用した綴りミスの支援を行っている．近藤ら[4]は一つの構文ミスによる複数のコンパイルエラーメッセージを分類するシステムを開発している．これらの研究はコンパイルエラーメッセージをもとに支援するため，コンパイルエラーメッセージの内容が明確ではないエラーに対しては支援することができない．また，コンパイラによって出力されるエラーメッセージが異なるため，コンパイラ変更の対応が難しい．

コンパイラが明示できないエラーである複数言語の混在誤りを支援する研究には，蜂巣ら[5]によって提案された自動修正ツールがある．これは，複数のプログラミング言語の共通モデルを作成し，ある言語のコーディング中に，他言語の構文が出現した際に，自動修正を提案する．このツールは，統合開発環境である Visual Studio Code 向けのプラグインとして開発されている．しかし，混在させてしまった他言語の文法誤りがあれば支援ができないことやプラグインを入れる手間がある．

Assiri ら[6]は実行時エラーの理解を支援する JENL Tool を開発した．このツールは，自然言語処理を使用して，Java の実行時エラーである例外処理のメッセージをユーザが理解しやすい文に変換する．

プログラミング学習に視線情報を活用した研究には，酒井ら[7]のアイトラッカーを用いたプログラミング学習改善の検討がある．この研究では，プログラミング問題解答時の参加者の視線情報を測定し，k-means 法や k-medoids 法を使用して，特徴量を抽出することで，効果的な学習支援を目指している．Turner ら[8]は C++と Python のプログラムに存在するバグを発見するプロセスの視線情報を測定する実験を行った．その結果，非初心者の方が初心者より 1.4 倍ほど，バグ発見の精度が高いと示した．

本研究では，既存システムへの機能追加とプログラム修正中の視線計測及び視線データの比較を行った．

3. 不明瞭なエラーメッセージ

プログラムは，コンパイラによって機械語に翻訳され，実行可能な状態になる．しかし，プログラム中に誤りが存在した場合に，コンパイラは実行ファイルを作ることができず，エラーを出力する．そのエラーはメッセージとして開発者に提示され，プログラムの誤りを発見し修正することに役立つ．一般的なコンパイラは誤りが発生している箇所と誤りの内容を出力するため，開発においてプログラムを修正する手掛かりになる．また，プログラミング学習においても，自身が作成したプログラムの誤りを発見し，何が誤りかを判断し，修正を行うことで学習することができる．そのため，エラーメッセージの誤り内容が明瞭でなければ，修正や学習を行うことは困難である．

エラー内容を明確に指摘できない例として，複数のプログラミング言語や全角文字の混在誤りがある．図 1 は C 言語のプログラム中に Python の elif 文及び全角の「｜」を記述してしまった例である．これを GNU Compiler Collection for Windows (GCC)でコンパイルした結果を図 2 に示す．全角の「｜」を使用した場合は，「error: stray '¥357'」と文字コードのエラー，Python の elif 文を混在させた場合は「implicit declaration of function 'elif'」(関数 elif の暗黙的宣言)と「error: expected ~」(次に予期する記号を示す)エラー

```
1   #include<stdio.h>
2   int main(void){
3           int a = 1;
4           if(a == 0 ｜｜ a == 1){
5                   printf("aは0か1です");
6           }
7           elif(a < 0){
8                   printf("aはマイナスの値");
9           }
10  }
```

図 1　誤りのある C 言語のプログラム

† 東海大学大学院情報通信学研究科 Graduate School of Information and Telecommunication Engineering, Tokai University
‡ 鹿児島県立短期大学　Kagoshima Prefectural College

が出力される．これらのエラーメッセージは誤りの内容を直接的に指摘していないため，プログラミング初心者は混乱する恐れがある．また，複数言語の混在誤りはプログラマが適切なプログラムを記述したと思い込むことがあるため，誤りに気づかないことがある．

```
test.c: In function 'main':
test.c:4:13: error: stray '¥357' in program
    4 |   if(a >= 0 |• <= 10){
      |              ^

test.c:4:14: error: stray '¥275' in program
    4 |   if(a >= 0 |• <= 10){
      |               ^

test.c:4:15: error: stray '¥234' in program
    4 |   if(a >= 0 |• m a <= 10){
      |                ^

test.c:7:2: warning: implicit declaration of function 'elif' [-Wimplicit-function-declaration]
    7 |  elif(a < 0){
      |  ^~~~

test.c:7:13: error: expected ';' before '[' token
    7 |  elif(a < 0)[
      |             ^
      |             ;
```

図 2　ソースコード 1 のコンパイルエラー(GCC)

図 3 は冒頭で#include<math.h>を記述し忘れた例である．このプログラムを Microsoft Visual Studio にて，コンパイル及び実行しようとすると，「関数 'sin' は定義されていません．int 型の値を返す外部関数と見なします．」と警告が表示される．しかし，実行ファイルは作成されており，実行すると，予期していない値が代入される．また，Tiny C Complier でコンパイルした際は，警告文が出力されず，予期していない値が代入される．このような動作は，標準ライブラリのヘッダファイルが暗黙的にインクルードされる GCC では問題ないが，インクルードされないコンパイラでは予期しない結果になることがある．そのため，標準ライブラリなどの関数を理解していない初心者にとって，難解なエラーであると推測される．

これらの問題に対して効果的な支援をすることで，学習効率が向上すると考え，プログラミング学習支援システムの開発を行っている．

```
1  int main(void) {
2          double sinValue = sin(30 * (3.14 / 180));
3  }
```

図 3　#include 忘れのプログラム例

4. 本システムの概要

本システムは，プログラムを入力して，記述誤りを確認し，修正を繰り返して学習することを想定している．したがって，現時点では問題を出題して演習をしたり教科書のような解説を載せたりする学習用コンテンツは持っていない．

4.1 システムの仕様

システムの概要を図 4 に示す．本システムの開発環境には，フロントエンドに JavaScript ライブラリの React，バックエンドに Apache サーバー上で動作する PHP，データベース管理システムに MySQL を使用している．Web ブラウザ上でユーザがプログラムを入力し，Submit ボタンを押すことで動作する．その後，React から PHP に HTTP 通信を行い，プログラムと入力値を送信する．PHP は GCC を用いてプログラムをコンパイルし，コンパイルエラーがなければ入力値を踏まえて実行をする．エラーが発生した場合は，PHP のスクリプトにプログラムを送り，字句分割，字句解析，文法確認により，エラーを分析する．以下，各機能について説明する．

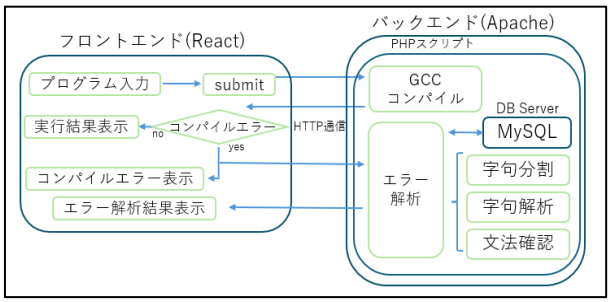

図 4　システムの概要

4.2 字句分割

字句分割では，プログラムを意味の最小単位である字句に分割をする．アルファベット，数字，アンダースコア(_)，ドット(.)，シャープ (#)は連結して一つの文字列とする．クォーテーションで囲まれた範囲，コメントアウト(「//」と「\n」，「/*」と「*/」)の該当範囲は一つの字句とする．上記とは異なる文字である，全角文字などは文字列を連結せず，ひとつの字句として設定される．

4.3 字句解析

字句解析では字句分割で分割された字句にタイプ付けを行う．字句は C 言語の予約語，他言語の予約語，演算子，記号，数値，変数名や関数名，C 言語にない字句に分類される．分類中に，登録された他言語の予約語，全角文字が存在した場合は，エラーを追加する．

4.4 文法確認

文法確認では，タイプ付けされた字句の並びと登録した文法データベースを照合し，正誤の判定を行う．複数の文法が存在する予約語は 1 つでも合致していれば正とする．正と判断された文は括弧の中身とその予約語の条件と合っているかを照合する．また，#include が検出された際に標準ライブラリを登録し，インクルードされていないライブラリ関数が使われていないか判断する．

4.5 エラーメッセージの出力

本システムの各機能で発見されたエラーは，内容とその発生した行番号，エラーの出力タイプ，文法誤りの場合は正しい文法をエラーメッセージ配列に格納する．エラーのタイプには A, B を設定し，A の場合は，エラー内容をそのまま出力するように指示し，B の場合は，正しい文法での書き方を追加する．図 1，図 3 の誤りのあるプログラムをこのシステムで実行した例を図 5，図 6 に示す．これらのエラーメッセージは行番号をもとにソートされ，プログラム中の誤りがある該当箇所の右側に出力する．エラータイプ B である文法に対するエラーメッセージでは，予約語を青色で表示し，マウスカーソルを予約語の上に乗せることで正規の文法が表示される．また，本システムのエラー出力と同時に，GCC のコンパイルエラーも出力されるため，一般的なコンパイルエラーメッセージの読み方の学習になると期待できる．また，GCC は暗黙的に標準ライブラリが記述されているヘッダファイルをインクルードしてしまうため，コンパイルエラーが検出されない時でも，「コードを解析する」ボタンを取り付け include 忘れのエラーも確認できるようにした．

図 5　実行結果(図 1 のソースコードのエラー解析)

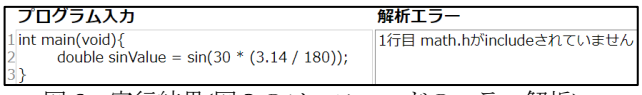

図 6　実行結果(図 3 のソースコードのエラー解析)

5. エラーメッセージ表示インタフェースの検討

わかりやすいエラーメッセージを提示しても，読まれなければ効果は薄いと考え，インタフェースの改善を検討している．初級プログラマがエラーを発見しやすいインタフェースの開発をするために，プログラミングスキルにもとづく，コンパイルエラー修正時の視線の違いを分析する．初心者と上級者では注視する箇所や時間が異なると仮定し，コンパイルエラー修正時の視線情報を測定する実験を行った．以下に実験条件，結果，考察を示す．

5.1 実験条件

参加者に C 言語のプログラムとコンパイルエラーメッセージを提示し，プログラム修正中の視線情報を測定する実験を実施した．以下に，参加者に提示されたコンパイルエラー修正問題の一覧を示す．

- 問題 1: セミコロンの欠如
- 問題 2: Python の for 文が使用されている
- 問題 3: 全角スペースが含まれている
- 問題 4: Python の "elif "が C 言語で使用されている
- 問題 5: 全角の"|"が含まれる
- 問題 6: 引用符がない

実験参加者は，学部 1 年生 3 名と，学部 4 年生 1 名，大学院生 3 名の計 7 名である．参加者には C 言語のエラー（問題1～問題6）を修正させ，その際の視線情報を記録した．視線分析には，Tobii 社製の Pro Spark[9]と Pro Lab[10]を使用した．図 7 は実験機材の配置写真である．実験中の視線を正確に測定するため，顎台を使用し参加者頭部の動きを抑制した．参加者は，図 8 に示すように，Microsoft Visual Studio を使用して提示されたプログラムとコンパイルエラーメッセージを閲覧し，プログラムを修正した．参加者はプログラムの修正及び実行を繰り返し，正解に達するか途中で断念すると，次の問題に進む．参加者が作業をしている間，視線は測定され続け，記録される．実験終了後にはアンケートを実施し，参加者に問題修正のプロセスについて回答を求めた．

図 7　実験機材の配置写真

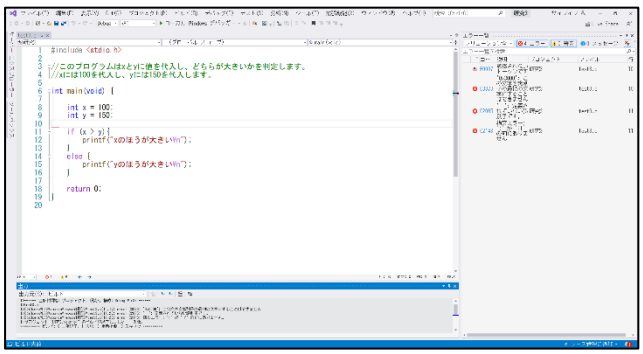

図 8　実験中に表示される画面

5.2 実験結果

実験結果を表 1 に示す．各問題の正答を「○」，誤答を「×」で表し，数値データは解答時間(分:秒)で示す．表 2 は各問題の解答時間におけるコンパイルエラーメッセージ注視時間の割合を示したものである．

5.3 実験の考察

表 1 より，問題 1，6 は，全ての参加者が比較的素早く解答していることから，コンパイルエラーメッセージの内容

表 1　各問題の正誤と解答時間

参加者	問題1		問題2		問題3		問題4		問題5		問題6		合計
1	○	0:41	×	2:32	×	8:05	○	1:36	○	5:32	○	1:04	19:30
2	○	0:20	×	4:49	×	3:58	×	6:07	○	4:40	○	1:36	21:30
3	○	0:33	×	2:20	×	3:52	×	4:49	×	3:29	○	1:42	16:45
4	○	1:00	×	4:22	×	6:20	○	1:13	○	4:57	○	2:51	20:43
5	○	1:25	×	4:02	○	2:15	○	1:24	×	1:42	○	1:47	12:35
6	○	0:32	○	2:15	○	0:35	○	0:42	○	2:27	○	0:43	7:14
7	○	2:05	×	2:44	×	3:57	×	2:04	×	1:42	○	2:51	15:23
各問題の正答数と平均解答時間	7	0:57	1	3:18	2	4:09	4	2:34	4	3:30	7	1:48	

表 2　各問題の解答時間における

コンパイルエラー注視時間の割合

参加者	問題1	問題2	問題3	問題4	問題5	問題6	平均	正答数
1	24%	17%	27%	25%	18%	16%	22%	4
2	20%	30%	26%	16%	13%	11%	20%	3
3	0%	23%	9%	2%	5%	7%	8%	2
4	30%	29%	34%	22%	29%	21%	29%	4
5	22%	34%	36%	7%	25%	14%	26%	4
6	44%	24%	20%	17%	27%	7%	24%	6
7	14%	12%	20%	10%	5%	2%	11%	2

が明確で適切であると考えられる．一方，問題 2，3 は比較的，正答者が少なかったことから，エラーメッセージの内容から誤りを特定することが困難であると考えられる．問題 2 の誤答者のアンケートには，「エラーの内容がわからず，Python に慣れているから混ざってしまった」と回答された．このことから，Python に精通している者は，記述されたプログラムが適切だと思い込んでしまい，正答できなかったと考えられる．また，問題 2 を正答した，C 言語と Python 経験者である参加者 6 は，アンケートに「for 文を拝見し，自分の知らないものだったので書き直しました」と回答していた．これは，for 文を見た際に，C 言語の for 文とは異なると気づいたため，修正できたと考えられる．問題 3 の誤答者のアンケートには「文字コードを知っていないと対応ができない」と回答があり，コンパイルエラーがわかりにくいことが示された．また，正答者である参加者 5 は，プログラムの括弧や改行の整形を行っている際に，エラーが修正されたため偶然であると考える．さらに，正答できた参加者 6 のアンケートには「エラー内容を見てもわからなかったため，エラーが起きた行にカーソルを合わせてみたらスペースがあったため消しました．」とあり，エラー内容が分からなくても，発生箇所から問題を特定し修正することが可能であることが示唆された．加えて表 1 より，問題 2，3，5 の参加者全員の平均修正時間は 3 分以上であり，問題 1，6 と比べると，修正に時間がかかる問題であることが考えられる．このことは，それらの問題のエラーに対して支援が必要であることを意味する．

表 2 より，正答数が 2 であった参加者 3，7 のコンパイルエラー注視時間の割合は 8%，11%と極端に少ないことが示された．したがって，参加者 3，7 はエラーの詳細を見ずにプログラムを修正している可能性が高い．一方，4 問以上正答していた参加者 1，4，5，6 はコンパイルエラー注視時間の割合が 20%~29%と、参加者 3，7 に比べ、エラーメッセージを頻繁に確認していた．このことは，エラーメッセージを見て理解することが，デバッグのプロセスにおいて重要なスキルであることを示唆しており，プログラミングスキルがまだ十分に身についていない者にエラーメッセージが提示する情報の見落としや，習慣がないことないことによるエラー内容の理解不足が懸念される．

以上のことから，わかりやすいエラーメッセージの提示が必要であること，そして初心者が重要なコンパイルエラーメッセージに注意を向けやすくするもしくは注意を向けることを習慣づけするインタフェースが必要であることが示唆された．

6. おわりに

本研究では，コンパイラの不明瞭なエラー内容を初心者にもわかりやすく表示し，正確な文法を表示することで，初心者プログラマの学習効率を向上させるプログラミング学習支援 Web システムの開発を行った．今回新たに，システム環境を変更し，include 忘れの支援機能を取り入れた．加えて，初心者がエラーに気づきやすいインタフェースを実現するために，コンパイルエラーを修正する際のプログラマの視線情報を測定する実験を行った．その結果，正答数が多い参加者ほど，エラーメッセージに目を向ける時間が長いことが分かった．今後は，さらにエラー修正時の視線情報を収集し，重要なエラーメッセージや，エラーがある箇所を強調するインタフェースの開発を行い，既存のインタフェースと改良したインタフェースの比較を行う予定である．また，機械学習を用いて，本システムでエラーを検知できる誤りを増やすことを検討している．

謝辞

本研究の一部は JSPS 科研費 JP23K11635 助成を受けた．視線情報収集にご協力いただいた皆様に感謝申し上げます．

参考文献

[1] Naoki Fujisaka, Mitsuho Yamada, Eriko Ishii, Yuko Hoshino, "Easy-to-Grasp Compiler Messages: Beginner-Friendly Web System Proposal", pp. 32–36, The 1st International Conference on ICT Application Research, September 9–12, (2023)
[2] 西村将広，橋浦弘明，古宮誠一，“プログラム初学者へのコンパイルエラー修正支援システム―エラーメッセージを用いた修正支援―”，情報システム学会第 6 回全国大会・研究発表大会論文集，pp.1-4，(2010)
[3] 後藤孔，藤中透，“プログラミング教育におけるデバッグ支援”，システム制御情報学会論文誌 Vol. 32， No. 6，pp.249-255，(2019)
[4] 近藤亮太，名倉正剛，“コンパイルエラーメッセージの分類に基づく初学者へのプログラミング学習支援手法”，第 9 回 実践的 IT 教育シンポジウム rePiT2023 in 函館，p.63-74，(2023)
[5] 蜂巣吉成，東直希，三上比呂，長野滉大，吉田敦，桑原寛明，“複数のプログラミング言語の文法知識に起因する制御文の誤りの自動修正方法の提案”，日本ソフトウェア科学会 39 巻 4 号 p.4_38-4_48，(2022)
[6] Fatmah Yousef Assiri and Hanan Elazhary "Automated Java exceptions explanation using natural language generation techniques", Computers & Applications in Engineering Education, 28, 626-644. Wiley Periodicals, Inc. DOI:10.1002/cae.22232. (2020)
[7] 酒井泰地, 浦野晶一，“アイトラッカーを用いたプログラミング学習改善の検討”，日本人工知能学会第 34 回年次大会論文集，セッション ID：4J2-GS-2-04．DOI: 10.11517/pjsai.JSAI2020.0_4J2GS204．(2020)
[8] Turner R, Falcone M, Sharif B, Lazar A "An eye-tracking study assessing the comprehension of C++ and Python source code" In: Proceedings of the Symposium on Eye Tracking Research & Applications, ACM, New York, ETRA ’14, pp 231-234，(2014)
[9] https://www.tobii.com/ja/products/eye-trackers/screen-based/tobii-pro-spark (2024-6-10)
[10] https://www.tobii.com/ja/products/software/behavior-research-software/tobii-pro-lab (2024-6-10)