コンパイルエラーメッセージの分類に基づく初学者に対するプログラミング学習支援手法

2019SE025 近藤亮太
指導教員：名倉正剛

1 はじめに

プログラミング初学者の学習言語として，C 言語を利用することが一般的である [1]．コンパイル言語では，構文ミスによりコンパイルエラーが生じた場合に，学習者はそのエラーメッセージを拠りどころに構文ミスの除去作業を実施する．熟練者であれば複数のエラーメッセージのうちどのメッセージが原因を示すかを経験的に理解しているが，初学者には容易に理解できないことがある．

2 初学者によるコンパイルエラーメッセージの理解への課題

エラーメッセージが構文ミスを含む箇所を直接指し示す場合，プログラミングを学習する初学者は文法に誤りがある箇所やその種類を認識できる．初学者がそのエラーメッセージに対応するエラーを解決し，再度コンパイルを実施してその解消を確認することにより，対象言語の文法への理解を深めることができる．

例えば，初学者がソースコード 1 に示すコードを記述したとする．このコードではコメントに示すように，5 行目と 7 行目の 2 箇所に構文ミスを含む．このソースコードをコンパイルすると，画面 1 に示すように， 2 箇所の構文ミスに起因して “error:” の文字列を含む 3 つのエラーメッセージが表示される．

この例では 5 行目で for ループの “)” を “]” に間違えていることにより，画面 1 では 5 行目に関して 2 つのエラーメッセージを生じている[*1]．1 つ目のエラーメッセージは for ループの開始の “(” に対応する終了の “)” が存在せずに，for ループの “(…)” に文法的に現れない文字 “]” が出現していることを示している．また，2 つ目のエラーメッセージは “]” について解釈した結果，“]” の前には文法的に変数や式などの何かしらのステートメントが現れるべきなのに，出現していないことを示している．

このように 1 箇所の構文ミスが複数の文法の制約に違反している場合，複数のエラーメッセージ群を生じることがある．それらが同一の構文ミスに起因するものかどうかを初学者は容易に判断できない．

ソースコード 1: 構文ミスを含むプログラムの例

```
1 #include <stdio.h>
2
3 int main( void ) {
4     int i, sum = 0;
5     for (i = 1; i <= 100; i ++]   // ) が ] に間違え
6         sum += i ;
7     printf("合計=%d\n", sum)     // ; が無い
8 }
```

```
total.c: In function ‘main’:
total.c:5:31: error: expected ‘)’ before ‘]’ token
    5 |     for (i = 1; i <= 100; i ++] // ) が ] に間違え
      |                               ^
      |                               )
total.c:5:31: error: expected statement before ‘]’ token
total.c:7:31: error: expected ‘;’ before ‘}’ token
    7 |     printf("合計=%d\n", sum)     // ; が無い
      |                               ^
      |                               ;
    8 | }
      | ~
```

画面 1: 出力されるエラーメッセージ

3 提案

3.1 概要

本研究では，コンパイルエラーが発生した場合に発生したエラーメッセージがどの構文ミスに起因しているのかを特定し，その原因ごとにエラーメッセージを分類して提示することで，初学者のプログラミング学習を支援する手法を提案する．

3.2 処理の流れ

提案手法の構成と処理の流れを，図 1 に示す．提案手法ではエラーメッセージと対応する構文ミスを修正する手順をあらかじめルール化しておく．対象とするエラーメッセージを条件部に，それに対する修正方法を結論部に記述する．

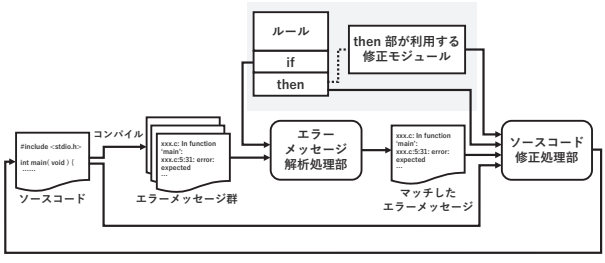

図 1: 提案手法の処理の流れ

処理の流れは，次のとおりである．

手順 1) ソースコードをコンパイルする．

手順 2) エラーメッセージ解析処理部は，手順 1 で出力されたエラーメッセージを解析し，ルールの条件部に

[*1] ファイル名 “total.c” の直後の数値が，コンパイラの解析の結果，構文ミスとして解釈されたソースコード位置を示し，1 つ目の数値が行番号を，2 つ目の数値が行内の先頭からの文字位置を表す．

ソースコード 2: ルールの記述例

```
1 "Missing specified token":{
2  "IF":"(\\w+.c):(\\d+):(\\d+): error: \\w+
3      ‘(\\S+)’ before ‘(\\S+)’ token",
4  "THEN":"replace($1, $2, $3, $5, $4)"
5 }
```

マッチするエラーメッセージのうち，一番最初に現れるエラーメッセージを抽出する．

手順 3) ソースコード修正処理部は，手順 2 でエラーメッセージの抽出に利用したルールの結論部が示す手順により，ソースコードを変更する．

手順 4) 手順 1 から 手順 3 を，該当するルールにより解析できるエラーメッセージが発生しなくなるまで繰り返し実施する．エラーメッセージ解析処理部は，手順 2 において，前回のコンパイル結果と比較し，出力されなくなるエラーメッセージを分類する．これをすべてのルールに対して実施する．

手順 4 の分類に基づきメッセージをグループ化し，コンパイル結果としてユーザに提示する．ユーザである初学者は，グループ化されたエラーメッセージを確認することで，どのエラーメッセージとどのエラーメッセージが関連しているかを理解することが可能になる．

3.3 ルールに基づく構文ミスの検出・修正

提案手法では，エラーメッセージの解析とその原因に対する修正をルールベースで実施する．ルールは，エラーメッセージを表す条件部（IF 部）と，そのエラーに対する修正方法を表す結論部（THEN 部）から構成する．条件部には表示されるエラーメッセージをマッチングするために利用する文字列を，正規表現で記述する．エラーメッセージ解析処理部は，手順 2 においてこの記述にマッチしたエラーメッセージを抽出する．そして抽出したエラーメッセージに対して，ソースコード修正処理部は結論部に記述する方法で修正を行う．

ルールの記述例を，ソースコード 2 に示す．このルールは，ソースコード 1 に示すようなソースコードをコンパイルした際に生じる画面 1 に表示されるエラーのうち，最初に表示されるエラー（ソースコード 1 の 5 行目に起因して発生するエラー）を検出し，該当するソースコードを修正するためのルールである．

4 評価実験

提案手法の効果を確認するため，大学初年度のプログラミング演習で発生した構文ミスとその際のコンパイルエラーメッセージを利用し，提案手法で分類できるかを確認した．対象とした演習は南山大学理工学部 1 年生の C 言語の実習形式の授業である．C プログラミング演習の最初期の 1 週分の演習での課題に対するソースコードとコンパイルエラーメッセージを取得し，提案手法により発生したコンパイルエラーメッセージを分類したところ，250 件中 96 件のエラーメッセージ群を分類できた．

表 1: エラーメッセージに対する分類の状況

①	②	③	④	⑤
1	1.0	32	1.0	0.0
2 ~ 5	2.0	51	1.6	0.0
6 ~ 10	9.0	7	1.0	0.0
11 ~ 15	11.5	4	1.0	5.5
16 ~	35.5	2	2.0	0.5

※ 表中の丸囲み数字は次を示す
①：発生したエラーメッセージ数の範囲　②：①の中央値
③：エラーメッセージ群の件数（合計=96 件）
④：分類グループ数（平均）
⑤：分類できなかったエラーメッセージ数（中央値）

5 考察

提案手法はコンパイルエラーメッセージを構文ミスごとに分類することを目的にしている．そこで，エラーメッセージ群ごとに，分類グループ数と分類できずに残ったエラーメッセージ数を算出した．その結果を表 1 に示す．

表 1 を参照すると，提案手法により分類した 96 件のエラーメッセージ群のうち 32 件には 1 件のエラーメッセージしか含まれていなかった．これらについては提案手法により分類することによる効果を得られていない．表 1 の ④ の列を参照すると，エラーメッセージ数の範囲によらず 1 ~ 2 件程度の構文ミスに起因し，複数のエラーメッセージを発生していたことがわかる．エラーメッセージが 2 ~ 5 件の範囲では中央値で 2.0 件のエラーメッセージを生じていた．仮にそれらが 2 件だったとしても，一度のコンパイルで同時に発生すると，それらが同じ構文ミスによって生じたものか，別個の構文ミスによって生じたものか，初学者に判断が難しい場合がある．そのようなエラーメッセージ群を構文ミスにより分類できたことにより，初学者の理解を支援できたといえる．

6 結論

本研究では，初学者によるコンパイルエラーへの除去作業を支援することを目的に，コンパイルエラー発生時にコンパイルエラーメッセージをその原因により分類して提示する手法を提案した．構文ミスの配置によっては適切に分類できない場合があり，ルールの適用方法を改善することが，今後の課題である．

参考文献

[1] Brown, N. and Altadmri, A.: Investigating Novice Programming Mistakes: Educator Beliefs vs. Stu dent Data, in Proc. of the Tenth Annual Confer ence on International Computing Education Re search (ICER ’14), 2014, pp. 43 – 50.

[2] N. Brown, A. Altadmri: Novice Java Programming Mistakes: Large-Scale Data vs. Educator Beliefs, ACM Trans. on Computing Education, Vol.17, 2017, pp.1-21.