R-2

プログラムの論理エラーに対する問題解決能力向上の手法

橋浦研究室　　1175114　佐藤 翔太　　1175129 杉山 大地

1. はじめに

現在の大学におけるプログラミング学習では，指導者が学習者に対し演習課題を与え，学習者がその課題に取り組むという形が一般的である[1]. プログラミング問題の解決能力を育成するためには，このような演習課題に取り組む場合に，学習者が解決手順の作成やその試行を行い，エラーがあれば内省を行うというアクティビティを繰り返すことが効果的である[2]. 一方で，現状のプログラミング学習においては，学習者の問題解決能力の欠如という問題が存在している[3]. 具体的には，プログラム中のエラーが発見できずその有効な解決方法を考えることが困難になることが挙げられる. 本研究では，プログラミング学習における問題解決能力について，学習者がプログラムに問題を引き起こしている原因を見つけ，自らが有効な解決策を考えたり，それを具体的なアルゴリズムとして表現する能力と定義し，この改善を目指す.

2. 研究目的

本研究の目的は，学習者による論理エラー発生原因の発見を支援することで，その発見を促進し論理エラーに対する問題解決能力を向上させることである.

2.1 論理エラーとは

ここで本研究が取り扱うプログラムのエラーについて述べる. 学習者がプログラミング学習の際に直面するプログラムのエラーについては，以下の 3 種類に分類することができる.

1. コンパイルエラー
   - コンパイラから出力されるエラー
2. 実行時エラー
   - 実行時に OS より出力されるエラー
3. 論理エラー
   - 上記 2 つのエラーを除いた，実行結果が仕様の意図とは異なる場合のエラー

このうち，論理エラーは開発環境で検知されずコンパイルエラーと実行時エラーに比べて発見が困難である.

例えば，「2 つの整数の入力値 a, b を受け取り，a が b 以上であれば a を出力し，そうでなければ b を出力するプログラムの作成」というプログラム問題があるとする. その問題の解答として図 1 左側のプログラムが作成された場合を考えると，a と b が等しい場合に出力が無くなる. この欠陥は開発環境では検知されないため論理エラーと区分できる. このような問題を修正するためには，プログラムが辿り得る分岐を正しく知っておく必要があるため. 学習者にとって解決が困難な問題である. このため，本研究ではプログラムの分岐網羅に着目し，この網羅度を向上させることで，論理エラー発見を促進することを目指す.

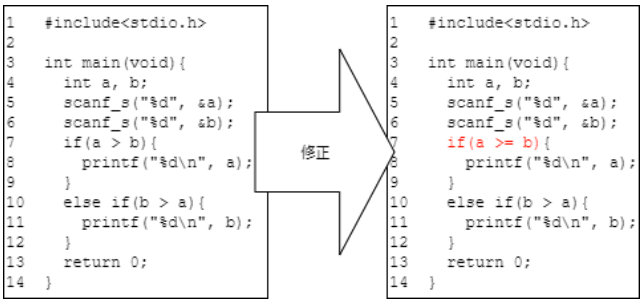

図 1 論理エラーの発生原因を修正する流れ

3. 提案手法

本研究では，学習者による論理エラー発生原因の発見能力を促進するために，プログラム問題演習を通してツールによりプログラムの論理エラーを検出し，その発見のヒントとなる情報を提供する手法を提案する.

論理エラーの検出は，模範となるプログラムとの比較により行う. 論理エラー発見につながるヒントの提供は，論理エラーの発生原因となる入力と，その結果の出力をメッセージとして示すことで行う.

4. 実装

本研究では，論理エラーの検知及びその発見のヒントとなる情報を提供するツールを以下の 3 つに分けて実装する. これらのツールはすべて WEB サーバ上に配置する.

1. 問題出題部
   - 学習者へのプログラミング問題の出題機能や，解答をプログラム解析部へ送信する機能，解答をサーバへ提出する機能を持つ WEB ページ
2. プログラム解析部
   - 学習者の解答プログラムの解析機能を持

つ WEB サーバサイドのプログラム

3. 解析結果出力部

➢ 解析結果を学習者へ提示する機能を持つ WEB ページ

学習者が問題出力部から出題される問題を基に作成した解答プログラムは，問題出題部に備えられた機能からプログラム解析部へ送信される．

プログラム解析部は，送信された解答プログラムへの解析で論理エラー発生原因の有無を確認し，その結果を解析結果出力部へ送信する．プログラム解析部から解析結果出力部へ送信される内容は，論理エラー発生原因となる実際の入力内容と，その結果として出力される内容である．

図 2 に解析結果出力部の実際の出力例を示す．図 2 の表中の「入力」が論理エラーを引き起こす実際の入力内容，「出力」がその入力の結果として出力される内容をそれぞれ表している．

チェックが完了しました
不具合を引き起こす入力値を発見しました。
チェックしたプログラムに、不具合が存在していると見られます。

表中の「パターン」=> 不具合の原因と結果を示す通し番号
表中の「入力」 => 不具合の原因となるscanf()での入力内容
表中の「出力」 => プログラムを実行して出力される全ての内容

表中の「入力」の内容を実際に入力したり、「出力」の内容を基に
エラーの原因を推測するなどして、実際にプログラムを修正して下さい。

パターン	入力	出力
1	0000000003	入れ替え間隔を入力してください(1~3まで) 1~3までの数字を入力してください
2	0000000002	入れ替え間隔を入力してください(1~3まで) one two three four five six

問題文に戻る

図 2 解析結果の提示画面の例

5. 実験

ツールの有無による論理エラー発生原因の発見率の差を確認するための実験を行った．具体的には，日本工業大学情報メディア工学科に所属する学生 9 名を対象に，プログラム中の論理エラーを発見するテストと，ツールを用いた論理エラーの修正に特化した問題演習を行うものである．

実験の手順を以下に示す．

i. 演習前テスト(20 分)
   ① プログラムの各経路に対応する入出力を問う問題
   ② 論理エラーの原因となる経路の選択
ii. 問題演習(30 分)
   論理エラーを含んだプログラムの修正
iii. 演習後テスト(20 分)
   i と同様の内容
iv. アンケート(5 分)
   ① i から iii までの問題の理解度
   ② ツールの使いやすさ

演習前後のテストで出題する問題では，フローチャートを見て，設問に解答する形式である．これらの問題で用いるフローチャートは論理エラーを含んだプログラムを基に作成されている．被験者にはフローチャートの各分岐を網羅するための入力変数の内容と，分岐を辿った結果の出力内容を回答してもらう．

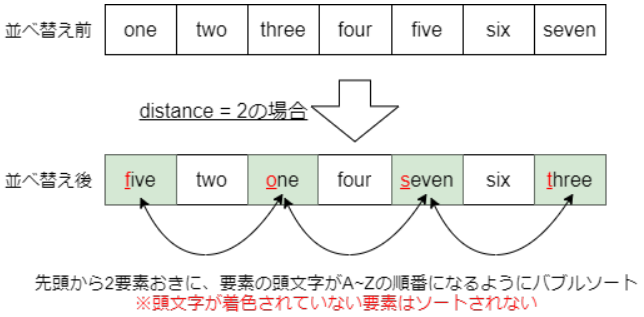

図 3 問題演習で出題した問題の仕様

```
1  #define _CRT_NO_SECURE_WARNINGS
2  #include <stdio.h>
3  #include<stdlib.h>
4  #include <string.h>
5  #define N 7
6
7  int main(void)
8  {
9    const char* p[N]={
10          "one", "two", "three",
11          "four", "five", "six",
12          "seven"
13   };
14   const char* temp;
15   int i, j, distance;
16   printf("入れ替え間隔を入力してください(1~3まで)¥n");
17   scanf("%d", &distance);
18
19   if(distance<=1||distance>=3){
20     printf("1~3までの数字を入力してください¥n");
21     return 0;
22   }
23
24   for(i=0; i+distance<N; i+=distance){
25     for(j=0; j+distance<N; j+=distance) {
26       /* 文字列の比較 */
27       if(strcmp(p[distance], p[j])==0) {
28         temp=p[j+distance];
29         p[j+distance]=p[j];
30         p[j+distance]=temp;
31       }
32     }
33   }
34
35   for(i=0; i<N-1; i++) puts(p[i]);
36
37   return 0;
38 }
```

赤字： 仕込んだ論理エラーの発生原因

図 4 問題演習で使用したプログラム

問題演習では被験者をツールを使用する実験群(5 人)と，使用しない統制群(4 人)にランダムに分けて実施した．問題の内容は論理エラーを含んだプログラムを，正しく動作する場合の入出力を手掛かりに修正するものである．

以下に問題演習に用いたプログラムの例述べる．この問題は「7 つの文字列を，入力する 1 から 3 までの整数の間隔で頭文字がアルファベットの昇順になるように入れ替え，最後にすべて出力する」という振る舞いを仕様として，プログラムを修正していくものである．仕様を図示したものと，問題演習として掲載したソースコードを図 3, 図 4 に示す．なお，図 2 で示した解析結果の提示画面の例は図 4 のプログラムを解析した結果を表している．

問題演習において，実験群と統制群がそれぞれ変更を加えた論理エラーの発生原因の数を評価し，χ^2検定を用いて有意差を確認する．有意水準αは 0.05 とする．

6. 結果と考察

図 4 に存在するすべての論理エラー発生原因に対して，被験者が変更を加えたか否かを集計した結果を表 1 に示す．さらに，表 1 を基にχ^2検定を行った結果を表 2 に示す．ここでは帰無仮説H_0を「ツールの有無により論理エラー発生原因に変更を加えた数に差はない」とし対立仮説H_1を「ツールの使用により論理エラー発生原因に変更を加えた数に差が生じる」とした．

表 1 実験・統制群が論理エラー発生原因に変更を加えた・加えなかった数のクロス集計表

#		変更有り	変更無し	合計
1	実験群	10	25	35
2	統制群	7	21	8
3	合計	17	46	63

表 2 表 1 を基に行ったχ^2検定の結果

χ^2値	0.101
P 値	0.751

表 2 より P 値が 0.05 を大きく上回った為，帰無仮説H_0を棄却することができなかった．よって，ツールを利用することで複数の種類の論理エラー発生原因の発見を促進することは出来ないという結果となった．

ここで，ツールの利用により特定の種類の論理エラー発生原因を発見しやすくなったかどうかに着目する．図 4 の 19 行目にある 2 つの論理エラー発生原因「distance<=1」「distance>=3」(以下，条件式の誤り①②)に変更を加えた被験者は実験群では 3 人，統制群では 1 人であった．この部分のみに関するデータのみを取り出したものを表 3 に示す．表 3 のクロス集計表を基にχ^2検定を行った結果を表 4 に示す．帰無仮説H_2を「ツールの有無により，条件文の誤り①②変更を加えた人数に差はない」とし対立仮説H_3を「ツールの使用により条件式の誤り①②変更を加えた人数に差が生じる」とした．

表 3 実験・統制群が条件文の誤り①②に変更を加えた・加えなかった数のクロス集計表

#		変更有り	変更無し	合計
1	実験群	6	4	10
2	統制群	2	6	8
3	合計	8	10	18

表 4 表 3 に対するχ^2検定の結果

χ^2値	2.205
Yates 補正	1.015
P 値(補正有り)	0.314

表 4 より P 値が 0.05 を上回ったため，帰無仮説H_2を棄却することができなかった．

最後に，アンケートの結果に着目する．アンケートにて,実験群の内 3 人から「ツールが何を示しているか分からなかった」「ツールが示す内容の見方について事前学習したかった」といった，ツールの使用に困難さを示す意見があった．ツールが示すメッセージをユーザがより理解しやすいように変更するなど，ツールの出力内容の見方の解説をより詳細化する事が課題として挙げられる．

7. まとめ

本研究の目的は論理エラー発生原因の発見を促進し，それに対する問題解決能力を向上させることであった．実験結果から，提案した手法が特定の論理エラーの発見を支援できる可能性が示唆された．

文　献

[1] 渡辺 博芳, 荒井 正之, 武井 恵雄, “初等アセンブラプログラミング評価支援のための事例ベース構築法,”情報処理学会論文誌, Vol.44, No.2, pp.496-506, Feb. 2003.

[2] 宮田 仁, 大隈 紀和, 林 徳治, "プログラミングの教育方法と問題解決能力育成との関連," 教育情報研究, Vol. 12, No. 4, pp.3-13, Dec. 1997.

[3] Siti Rosminah Md Derus, Ahmad Zamzuri Mohamad Ali, "Difficulties in Learning Programming: Views of Students," Proc. of International Conference on Current Issues in Education (ICCIE2012), pp.74-78, Sep. 2012.