デバッグ支援のためのフローチャートによる プログラムの動作の可視化

下山 晃生[*1]・北 英彦[*1]・高瀬治彦[*1]
Email: 421m220@m.mie-u.ac.jp

*1：三重大学工学研究科電気電子工学専攻

◎Key Words プログラムの可視化，デバッグ支援，プログラム動作

1. はじめに

情報社会の発展とともに，プログラミング学習の重要性が高まっており，多くの教育機関でプログラミング教育がなされている。著者らの大学でもプログラミング能力を身に着けるために，著者らの研究室が開発しているプログラミング演習システム PROPEL を用いて，実際にプログラムを作成しプログラムに対する理解を深めるプログラミング演習が行われている[1]。

学習者がプログラム作成の演習課題に取り組む際，作成したプログラムを実行したときに，動作がおかしいと気付いても，その原因を取り除けないことがある。原因を取り除くためには，プログラムのどこで・どうして正しい動作をしなくなったのかを知る必要がある。小さく，簡単なプログラムであればプログラムの構造を容易に把握することができるが，プログラムがより大きく，複雑になるとプログラムの構造を容易に把握することが難しく原因の特定が困難になる。

この問題を解決する方法の一つは，プログラムの挙動を視覚化がある。本研究の目的は，プログラミング初心者を対象とし，学習者が作成したプログラムの挙動を，フローチャートを用いて視覚化することで，プログラミング学習におけるデバッグを支援する機能の提案である。

2. 研究の目的

本研究の目的は，プログラムの流れを可視化することで論理エラーの原因の発見を支援し，学習者の問題解決能力とプログラムの構造に対する理解を向上させることである。

2.1 論理エラーとは

プログラミングにおいて発生するエラーは以下の 3 種類に分類される。

1. コンパイルエラー
   コンパイルする際に，文法の誤りや変数の未宣言など，プログラムが正しい構造ではない場合に発生する。このエラーが発生すると，プログラムはコンパイルできず，実行ファイルが生成されない。
2. 実行時エラー
   プログラム実行時に，ゼロでの割り算や存在しないファイルへのアクセスなど，プログラムが実行される過程で予測できない条件やエラーが発生した場合に発生する。このエラーが発生すると，プログラムは中断され，エラーメッセージが表示されることがあり，対処しないとプログラムは異常終了する。
3. 論理エラー
   プログラムはエラーなく実行されるが，計算式やアルゴリズムの誤りなどといったプログラムの設計や論理の誤りによって，期待される結果が得られないか，意図しない動作をする状態。

これらエラーのうち，論理エラーはシステム的なエラーではないため，開発環境で検知されないため，他のエラーに比べて発見が困難である。

3. 現状の問題点

現状におけるプログラミング初心者がデバッグする際に発生する論理エラーが解消できない要因として，プログラムが大きくなると論理エラーの原因を特定しにくくなることがある。

この原因の１つとしてプログラムの全体像の理解不足があげられる。プログラムが大きくなると，ソースコードの可読性が低くなり，プログラム全体の流れや構造を理解することが難しくなる。プログラムの全体像を把握していないとプログラムのどの部分で何が行われているかの理解が不足していると，エラーがどこで発生しているか特定するのが難しくなる。

また，相互作用の複雑化も初心者がデバッグする際に困難に陥りやすい要因であると考える。

プログラムが大きくなると，多くの条件分岐，ループ，例外処理などが含まれることがあり，これらの制御フローが相互に絡み合い，プログラムの実行パスを形成している。そのため制御フローがどのように影響しあうのかを理解せずに特定の部分を修正すると，別の部分に予期せぬ影響を及ぼし，結果的にエラーが解決しないといったことが起きうる。

これらの問題を解決するには，学習者がプログラムの全体像を把握し，各部分の役割と処理の流れを理解する必要がある。

3.1 Tung の可視化機能

演習で使用している PROPEL には，論理エラーの解決を支援する機能として Tung の視覚化機能が実装されている。ここでは，その機能について紹介する。Tung の可視化機能は，既存のプログラムの動作の可視化機能である Online Python Tutor の拡張である伊藤の可視化機能を更に発展させたものである[2][3]。

機能は，変数や配列の値が命令の実行前後で変化した領域を色で強調表示し，変更を視覚化エリアで実行前後の状態を表示することで示し，ユーザーはプログラムコードの１行の実行前後の挙動を確認できる。

この機能は制御構文の動作を 1 ずつ確認するのには適しているが，入力を必要とするプログラムや大きいプログラムには対応しておらず，また，プログラム全体の流れを把握するのには適していない。

4. 新たに提案する機能

新たに提案する機能を図 1 に示す。この機能では，図 1 内のエリア２とエリア３の任意の箇所をクリックするとその箇所に対応するコード，フローチャートの箇所に移動し赤枠で強調表現される。

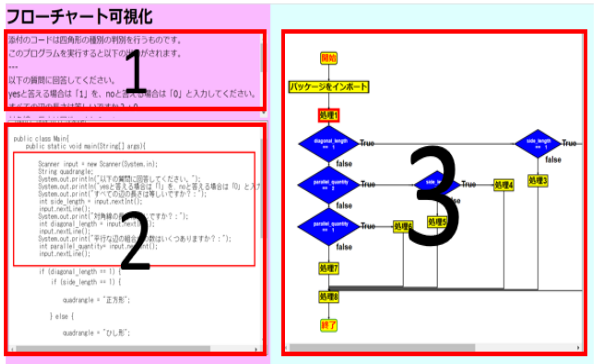
提案する学習者のデバッグを支援する機能では，プログラムの特定のセクションや条件分岐の中で何が起こっているかを追跡しやすくし，プログラムの全体像を把握しやすくすることでプログラムの構造の理解をうながすために，プログラムの流れ可視化する機能が必要である。

プログラムの挙動を説明する手段として，フローチャートと呼ばれる図表が使用されることがある。

フローチャートは，システムやコンピュータアルゴリズムの手順とプロセスの流れを示すために使用される図表で，プログラミング教育においては，アルゴリズムを学習者に説明するためによく使用されている。

長方形，円，ひし形など，様々な形状で手順を示し，形状同士を接続する矢印を使用して手順を示すため，プログラムの制御構文を学習するのに適している。

これらのことから，プログラミング学習者が自身で記述したコードをフローチャートとして可視化することで，どのように作成したプログラムが動くかの理解につながると考えた。



1．課題の内容　2. プログラムのコード
3. コードから作成したフローチャート
図 1　提案する機能の画面

5. 実験

今回，実際にフローチャートによるプログラムの可視化がプログラミング学習者のデバッグに役に立つかの検証を行うため，本大学のプログラミング演習の講義の受講者を対象に，四角形の判別を行うプログラムのデバッグを行う小テストを２回行った。

小テストの概要は以下のものである。

- 制限時間 15 分
- 論理エラーを含む四角形の判別を行うプログラムのコードを提示
- 論理エラーが起こった場合の入力と出力を提示
- 学習者は，プログラムの四角形の判別を行う条件分岐の部分を修正して提出する。

1 回目では，以上の条件でそのまま小テストを行い，2 回目では，上記の条件に加えて図 1 に示す機能を使い，小テストを行った。

6. 実験結果

小テストの結果を表１と表２に示す。表１からわかるようにフローチャートの機能を使わなかった場合の小テストの正答率は 65%，使った場合の正答率は 75%と全体を通しての正答率はあまり変わらなかったが，表２からわかるようにフローチャートの機能を使った場合，１回目不正解だった学習者のうち 57%の学習者が正解することができるようになった。

実験結果から，フローチャートによるプログラムの流れの可視化にはデバッグ補助する一定の効果があると考える。

表１　小テストの結果

	正解	不正解
フローチャートなし	42 人	23 人
フローチャートあり	49 人	16 人

表２　１回目不正解だった学習者の２回目の結果

正解	13 人
不正解	10 人

7. まとめ

今回，プログラミング学習において，学習者間で学習の進捗に差の原因である論理エラーを解決方法として，学習者が作成したプログラムの挙動をフローチャートで視覚化する機能を提案した。この機能を後期授業で活用し，その効果を確認するために調査が行う。

8. 参考文献

[1] 伊冨昌幸，小島佑介，高橋功欣，北英彦：プログラムの作成状況を把握する機能を持つプログラミング演習システム，2010PC カンファレンス（2010）

[2] 伊藤 福晃，北 英彦：“プログラミング演習における動作確認を支援するためのプログラム動作の可視化”，2018PC カンファレンス論文集，pp. 29–32（2018）.

[3] TRAN THANH TUNG，北英彦，高瀬治彦：プログラムの動作の可視化によるプログラムの動作理解の支援，2021PC カンファレンス論文集，pp. 274–277，（2021）