K-023

階層構造化による C 言語ソースコードの自動採点
Automatic Scoring of C Programming Source Code Using Hierarchy-structure

小山　昂紘‡　　原田　史子†　　島川　博光†
Takahiro Koyama　　Fumiko Harada　　Hiromitsu Shimakawa

1. はじめに

大学の情報教育は，学生にコンピュータに関する基礎知識や概念を理解させ，活用できる能力を身に付けさせることを目標としている．プログラミング教育の初期段階では，学生にプログラムを作成する能力を習得させるために，プログラミングに関する講義と実際にソースコードを記述するプログラミング演習を設けている場合が多い．プログラミング演習では，学生は個人ごとで演習課題を解くためにソースコードの作成に取り組む．

現在のプログラミング演習では，学生の作成したプログラムが与えられた課題に対して正しいデータを出力するかで課題内容を理解できたかを評価する．しかし，プログラムの実行結果からわかるのは学生のプログラミングに対する総合的な理解度である．演習課題で作成するプログラムは変数の型や条件分岐による処理の流れなど，複数の分野の知識を結合して構成される．そのため，学生のプログラミングに対する理解度を正しく評価するには，これらの分野の知識を個々に評価する必要がある．学生が個々の分野を理解できているか評価するには，プログラムの出力結果ではなく，ソースコードの記述内容を詳細に評価する必要がある．

2. 詳細採点に対する取り組みと問題点

2.1 ソースコードの詳細評価における負荷

著者らは，学生がプログラミングに対する知識や理解度を評価するために手法 [1] を提案した．手法 [1] ではひとつの演習課題に対してソースコードを評価するポイントとなる複数のチェック項目を設ける．次に，各チェック項目をプログラミングの理解度を表現する分野に関連付ける．学生の各チェック項目に対する正誤情報を分野ごとに抽出し，関連する分野ごとに分ける．チェック項目の正誤情報を分野ごとに調べることで各分野に対する学生の理解度を評価した．実際に 2010 年度の演習において 189 名の学生の理解度を評価する実験を実施した．

演習では学生は 4 つのクラスに分かれ演習に取り組む．演習課題は全部で 12 週にわたり出題され，合計で 92 問あった．ひとつの課題につき 9 項目程度のチェック項目を設けた．各クラスには 10 名程度の評価者を配属し，チェック項目にしたがって学生のソースコードを採点した．

実験の結果，学生の理解度を分野ごとに評価することができた．しかし，評価者の負担が大きいことが問題としてあげられた．そのため，採点者の負荷を軽減し，ソースコードを詳細に採点する手法が必要となる．

2.2 既存研究

ソースコードの評価者の負荷を軽減する手法として自動採点が挙げられる．文献 [2] では作成したプログラムの実行結果を比較することで作成されたソースコードの採点をしている．この手法ならば，システム上でプログラムを実行し，出力結果を解答と比較するだけなので評価者に負担がかからない．しかし，この手法ではソースコードの内容までは確認することはできない．プログラミング演習では指導者に指定された技法を用いてプログラムを作成する必要がある．例えば，階乗を求めるプログラムを作成する場合，ループを用いる技法と再帰関数を用いる技法が考えられる．実行結果を調べるだけでは指導者が指定した技法を用いているかは確認できない．

ソースコードから処理の制御を調べて，ソースコードの中から特定の処理を見つける手法として文献 [3] が挙げられる．文献 [3] ではコードクローンと呼ばれるソースコード中の類似または一致した部分を見つける手法について述べられている．しかし，この手法はソースコードから特定の処理を見つけるだけで処理の順序は考慮されない．そのため，作成されたソースコードが課題の意図に即した正しいものかは判断できない．

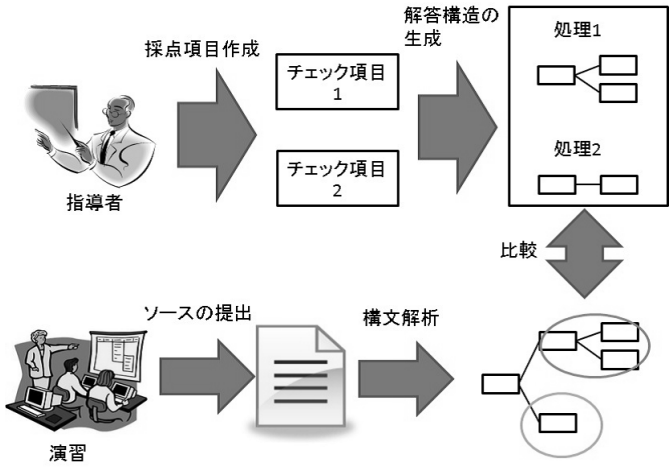

図 1: 手法の概要図

3. ソースコードの木構造化による詳細採点

3.1 構文木を用いた採点

本論文ではソースコードを木構造化し，ノードの組み合わせを調べることで詳細に評価する手法を提案する．本手法では，指導者は課題ごとに評価するポイントを定義し，解答となる木構造を作成する．学生が提出したソースコードを木構造化し，解答となる木構造が存在か調べることで採点をする．以上の手順を踏むことで学生のソースコードが評価するポイントを記述できているか確認できる．これらの処理を自動化することでソースコードの詳細な自動採点が可能となる．

図 1 に手法の流れを示す．本手法では，はじめに指導者は課題ごとにソースコードを評価するポイントとなるチェック項目を作成する．次にチェック項目ごとに解答

†立命館大学情報理工学部
‡立命館大学大学院理工学研究科

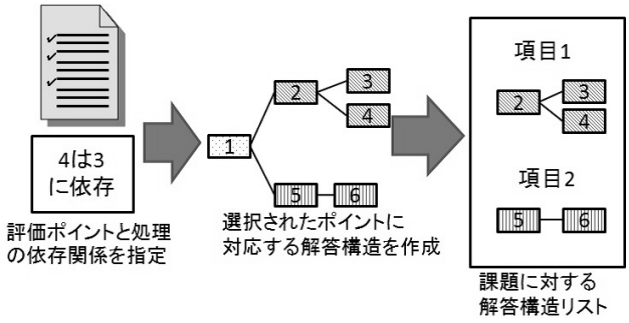

図 2: 解答構造の作成

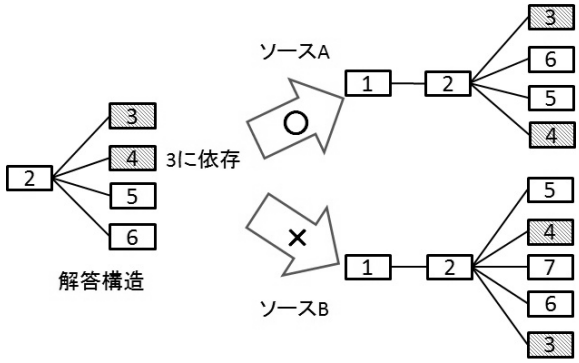

図 3: 解答構造による正誤判定

となる構文木を生成する．この解答となる構文木を解答構造と定義する．学生は課題の解答となるソースコードを作成し，演習用のサーバに提出する．次に提出されたソースコードを構文解析し，構文木を作成する．作成した構文木の中から解答構造と同一の構造を検索する．同一の解答構造があれば，解答構造に対応したチェック項目を正解とする．

3.2 解答となる構文木の生成

解答構造の作成の流れを図 2 に示す．はじめに，指導者は解答となるソースコードを作成する．次に解答となるソースコードからチェック項目の評価ポイントとなるコードを指定する．最後にシステムがソースコードから構文木を生成し，指定されたコードに対応したノードを抽出する．以上の流れで指導者は解答構造を作成する．

プログラムには処理の順序に依存関係が存在する．例えば，キーボードから標準入力された整数値を用いて計算するプログラムの場合，計算の処理の前に標準入力の処理が無くてはならない．そのため，指導者は評価ポイントとなるコードを抜き出す際に処理の依存関係についても考慮する必要がある．指導者はチェック項目の評価ポイントとなるコードを指定した後，それぞれのコードが表す処理の依存関係についても記述する．解答構造だけでなく処理の依存関係についても指定しておくことで採点のときに処理の順序が解答構造と異なるソースコードが提出されても自動採点が可能となる．

3.3 構文木の比較

ソースコードから作成した構文木から解答構造のルートと同一の処理を検索する．ルートと同一の処理を見つけたならば，ルート以下の処理を比較していく．比較した結果，学生が解答構造と同一の処理を作成していることが分かれば，解答構造に対応するチェック項目を正解とする．このとき，木構造を比較する方法として編集距離を用いる．編集距離は木構造の違いを，処理の挿入，削除，置換の 3 つの観点から調べて比較する手法である．処理の違いを表す 3 つの観点にそれぞれコストを与えて比較対象の違いの度合いを求める．編集距離を用いて比較することで処理が解答構造と処理の順序が同一でない場合でも採点することができる．編集距離の計算のアルゴリズムとしては動的計画法を用いる．

図 3 を用いて学生が作成したソースコードの正誤判定について説明する．評価ポイントとなる解答構造では処理 3 と処理 4 は依存関係にある．解答構造のルートとなる処理 2 のノードを見つけて子ノードを調べる．ソース A の木構造は処理 5 と処理 6 の位置が解答構造とは異なる．しかし，処理 5 と処理 6 には依存関係がないので順序は入れ替わっても良い．また，依存関係である処理 3 と処理 4 の順序関係は変化していない．よって，ソース A は解答構造と同一の処理が記述できていると判定できる．ソース B は処理 4 が処理 3 の前に存在する．処理 3 と処理 4 は依存関係にあるため，処理 3 と処理 4 の順序は逆になってはいけない．そのため，ソース B を提出した学生は解答構造に対応するチェック項目を満たすことができていないと判定できる．また，解答構造には存在しない処理 7 が追加されている．処理 7 の用に解答構造にない処理が追加された場合は処理の内容が他の処理に影響を与えないか調べる必要がある．

4. おわりに

本論文では学生が提出したソースコードを木構造に変換し，指導者が指定した処理を記述できているか評価する手法を提案した．本手法を用いることにより，評価者に負担をかけること無く学生のソースコードを詳細に評価することができる．また，手法 [1] と組み合わせることで評価者に負担をかけること無く，学生のプログラミングに対する理解度を評価することが可能となる．

今後の課題として指導者が解答構造を作成するためのツール，木構造を比較して自動採点をするツールの開発があげられる．

参考文献

[1] 小山昂紘, 谷川紘平, 原田史子, 島川博光，”絶対評価を用いたプ ログラミング演習における理解度測定 ”，電子情報通信学会/情報処理学会，情報科学技術レターズ，Vol.10，No.3，pp．67-70，Sep.，2011.

[2] 倉田 英和，富永 浩之，林 敏浩，垂水 浩幸，”実行テストによるプログラム判定を用いた初級 C プログラミング演習支援と授業実践 ”，情報処理学会研究報告．コンピュータと教育研究会報告 2007(101)，11-18，2007-10-05

[3] 神谷 年洋，”コードクローンをテンプレートとして用いて識別子のバリエーションを調べる手法の提案 ”，電子情報通信学会技術研究報告．KBSE，知能ソフトウェア工学 108(65)， 59-64， 2008-05-22