N_003

# コード・イディオムの構造解析を用いたソースコードの自動評価

## The Automatic Evaluation of Source Codes Using Structural Analysis of Code Idioms

小柳 順一 †　　島川 博光 †
Junichi Koyanagi　Hiromitsu Shimakawa

## 1. はじめに

著者らはCプログラミング演習科目において，個人の理解状況に合わせたプログラミング演習支援[1][2]を目指している．一般的にプログラミング演習科目では，課題ごとに設定した評価項目に沿って，教員がソースコードの出来具合を目で確認することによって，評価を行う．教員は提出された全てのソースコードに対して，この作業を繰り返し行わなければならない．そのため，評価に多大な時間を要し，負荷が高い．

本論文では，この問題点を解決するために，1 つの処理を実現する数行単位のコード集合ごとにその構文正誤を自動評価する手法を提案する．本手法では，まず学習者が作成したソースコードから自動評価可能なコード集合を抽出し，教員が作成したモデルプログラムに組み込む．そして，そのモデルプログラムをコンパイルし，テストデータを用いて正しく実行されるかを判定する．処理が正しく機能した場合は，さらに抽出したコード集合のエレガントさを評価する．これにより，コード集合を構文正誤，意味正誤，プログラミング・スタイルの 3 つの観点で自動評価することができる．

## 2. ソースコード評価の現状

大学などで行われるプログラミング演習のクラスは，1 名の教員に対し，多数の学習者で編成される．そのため，教員は多数のソースコードを受理して評価を行うことになる．現在，本学のプログラミング演習科目における評価は，まず教員が図 1 に示すような実現されるべき技術的要素を評価する項目を課題ごとに 10～20 個ずつ設定する．そして，教員は学生が作成したソースコードに対して，それらの要素の出来具合を目で確認する．さらに教員は提出されたソースコードに対して，この評価作業を繰り返し行わなければならない．特に，複雑な処理を実現しているソースコードの評価を行うさいは，正しく実現できているかどうかを判断するために，その処理の流れを頭の中で思い描く．そのため，教員は評価に多大な時間を要し，負担が非常に大きい．

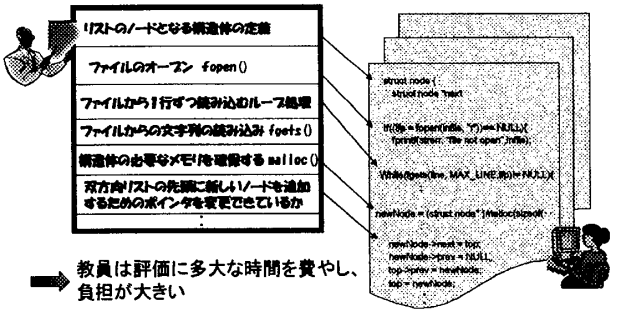


図 1：ソースコードの評価方法

†立命館大学大学院　理工学研究科

## 3. コード・イディオムを用いた評価支援

### 3.1 コード・イディオム

一般的なソースコードの主な構成要素は，様々な処理を実現するコード群である．何らかの処理を実現する一連のコードには，コード間の順序制約や依存関係，またコードが持つデータ構造そのものにパターンが存在するものもある．たとえば，図 2 に示すようなC言語プログラミングで「双方向リストの先頭にノードを追加する」処理を実現するコードには，順序制約や依存関係が存在する．本論文では，このような一つの処理を実現する数行単位のコード集合をコード・イディオムと捉える．このコード・イディオムの自動評価手法を提案し，ソースコード評価時における教員の負荷軽減を目指す．

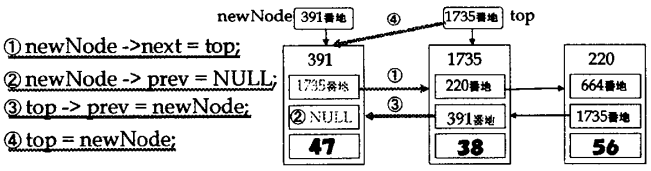


図 2：コード・イディオムの例

### 3.2 コード・イディオムの生成と選出

大学などのプログラミング演習において，教員は理解しがたいコードを記述している学習者を発見することがある．そのさい，そのような学習者に対して，教員や TA は理解の容易なコードを記述するように指導を行いながら，演習を進めている．このことからも，ソースコード作成者は，他人にも理解が容易なエレガントさを兼ね備えたソースコードを作成することが望まれている．特に，複雑な処理を実現するコード・イディオムは，簡潔に記述するべきである．したがって，教員はソースコード中のコード・イディオムのエレガントさにも重点を置いて評価をしなければならない．

そのために，本手法では，まず教員がエレガントなコード・イディオム・モデルの生成と選出をしておく必要がある．図 3 に「双方向リストの先頭にノードを追加する」処理を実現するエレガントなコード・イディオム・モデルの選出過程を示す．まず教員は課題の解となるモデルプログラムを生成し，そのモデルプログラム内の「双方向リストの先頭にノードを追加する」処理を実現するコード・イディオムを手動で抽出する．①のコードは②のコードの後にあっても，この処理は正しく機能する．一方，④のコードは，先頭ノードのアドレスを指すポインタ変数topを，追加したいノードのアドレスに変更するコードである．そのため，④のコードが①や③のコードよりも順序が先である場合，この処理は正しく機能しない．このように，コード・イディオム中の各コードには，コード間に順序制約が

存在する場合があるので処理が正しく機能するコード順のコード・イディオム・モデルを生成しなければならない．そのために，まず抽出した 4 行のコードを並び替えた全 24 通りのコード・イディオム・モデルを生成する．そして，24 種類のコード・イディオム・モデルをモデルプログラムに組み込み，組み込んだモデルプログラムに対して，それぞれコンパイルとテストデータを与えた実行を行う．この実行結果より，処理が正しく機能するコード・イディオム・モデル候補が明らかになる．さらに教員はコード・イディオム・モデルの候補から，エレガントさを兼ね備えたコード・イディオム・モデルの選出を主観で行う．この結果，エレガントなコード・イディオム・モデルが生成できる．

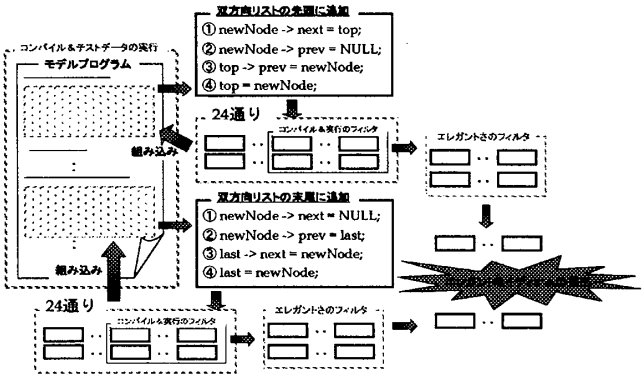


図 3：コード・イディオムの選出過程

### 3.3 コード・イディオムの自動評価

本手法では，教員によって選出されたエレガントなコード・イディオム・モデルと同様のコード・イディオムが学習者のソースコードに存在するかどうかを調べることで，ソースコードの評価を可能とする．教員は学習者のソースコードから自動評価したいコード・イディオムを手動で抽出する．抽出したコード・イディオム内の変数は，モデルプログラムで使用可能な変数に置換して，モデルプログラムに組み込む．このモデルプログラムをコンパイルすることで，組み込んだコード・イディオムの構文正誤を評価できる．コンパイルが成功した場合は，その実行プログラムにテストデータを与える．この実行結果からコード・イディオムが持つ意味の正誤が評価できる．

さらに，処理が正しく機能した学習者のコード・イディオムのエレガントさも評価する．学習者のコード・イディオムをコード・イディオム・モデルの候補と照合する．学習者のコード・イディオムがコード・イディオム・モデルの候補と一致しない場合，学習者のコード・イディオムがエレガントな記述でないことを教員に通知する．

## 4. ソースコード評価支援システム

本論文で提案したコード・イディオムの自動評価を実現するシステムを図 4 に示す．本システムはユーザに負荷をかけない単純なマウス操作のみで使用可能なユーザインタフェースを想定している．本システムは主に３つのフィールドから構成される．ユーザはソースコード表示フィールドから自動評価したコード・イディオムをマウスで選択する．変数置換フィールドでは，選択したコード・イディオム内の変数をモデルプログラムで使用可能な変数に置換する．評価結果フィールドでは，モデルプログラムのコンパイル，テストデータを用いた実行結果やコード・イディオムのエレガントさの評価結果が表示される．

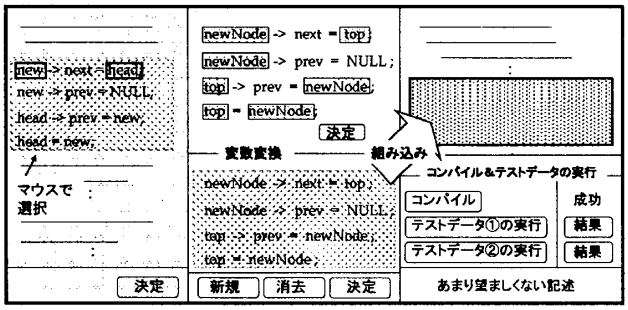


図 4：ソースコード評価支援システム

## 5. 関連研究との比較

これまでにも，ソースコードの評価を支援する研究は活発に行われている．文献[3]は，学習者のソースコードの構造とモデルの構造を比較する静的分析を提案している．また文献[4]は，ソースコードの実行形式ファイルにテストデータを与える動的解析を提案している．これらの手法では，web ベースのプログラミング環境を使用した個人の技量を検証する試験などで用いられる穴埋め形式の課題が対象である．しかし，講義で得た知識の実践やその理解を目標としているプログラミング演習のような科目では，穴埋め形式の課題は出題されないため，適用することができない．

## 6. おわりに

本論文では，C 言語プログラミングにおけるコード・イディオムの構造解析を用いたソースコードの自動評価手法を提案した．本手法を用いると，複雑な機能を実現するソースコードの評価にかかる教員の負荷が軽減される．今後は，提案手法を実現する評価支援システムの実装を進め，提案手法の有効性を検証する予定である．

## 参考文献

[1] 田口 浩, 島川 博光, 石井 篤, “個人の理解度に合わせた C 言語プログラミング教育支援環境”, *Proc. of the 2nd International Conference on Creating, Connecting and Collaborating through Computing*, pp.60-63, 2004.

[2] 田口 浩, 島川 博光, “プログラミング教育における個人の理解度に応じた学習順序の決定”, 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.3, pp. 343-345, 2004.

[3] Nghi Truong, Paul Roe, Peter Bancroft, “Static Analysis of Student’s Java Programs”, ACM International Conference Proceeding Series; Vol. 57, pp.35-41, 2001.

[4] Nghi Truong,Paul Roe, PeterBancroft, “Automated Feedback for Fill in the Gap Programming Exercises”, *proc.of* Australasian Computing Education Conference, pp.117-126, 2005.