カスタマイズ可能なプログラミング学習用プルーフリーダの提案

2012SE227 清水祐輔　2013SE065 加賀弘晃　2013SE110 松井敦紀
指導教員：蜂巣吉成

1　はじめに

大学で行われているプログラミング演習では，学習者は教育者に与えられた課題内容に沿ったプログラムを作成することで学習を進めている．しかし，学習者は学ぶべき要点は何かを意識することなくプログラムを作成し，実行結果は課題に合致していても教育者の考える教育意図から外れたコードを書く場合がある．このとき，教育者が学習者のコードを確認して，意図と異なる記述を指摘する方法があるが，教育者の負担が大きい．教育者が模範解答を開示し，学習者が自身のコードと比較するという方法もあるが，なぜ修正しなければならないのか理解できないまま学習を進めてしまう場合がある．

2015 年度の卒業研究 “教育意図を利用したプログラムのプルーフリーダの提案”[1] では，教育意図を学習項目ごとに分類し，学習者の解答が教育意図を満たしているかを判定するツールを提案している．この研究では，教育意図が記述されている箇所を抽出パターンを用いて評価コードとして抽出する．模範解答と学習者の解答の評価コードを比較することで判定を行っている．しかし，このツールには 2 つの問題がある．1 つ目は，プログラムの制御構文を扱う学習項目のみを評価していることである．2 つ目は，学習項目ごとに抽出範囲をあらかじめ決めているので，その範囲から外れたコードの細かい部分を判定基準に加えることができないことである．

本研究の目的は，学習者の解答に教育者の考える教育意図が含まれているかを確認，指摘を行うプルーフリーダの提案である．本研究では，課題ごとに固有の教育意図が存在すると考え，教育者が自由に抽出パターンを追加できるカスタマイズ機能と特定の字句列の出現回数の計測方法を提案し，[1] の問題点を解決する．本研究では，教育者が学習者に学ばせたい C 言語の学習項目を明解 C 言語入門編 [7] を参考に設定した．カスタマイズ機能によって，教育者が抽出パターンを追加することで，従来の抽出パターンに含まれない部分の不適切な記述を検出することができる．本研究で提案するツールによって，学習効果の向上と教育者の負担軽減が期待される．

本研究で提案するツールは，プログラミング演習において，学習者がプログラムが実行できたことを確認した上で教育意図と一致しているかチェックすることを目的としている．学習者のコードは，コンパイルが成功し，課題の実行例と同等な結果が得られたプログラムを対象とする．

2　関連研究

コーディングチェッカーや学習者向きのフィードバックを行うツールはいくつか提案されている．

Truong ら [2] は，制御構造の観点から模範解答と学習者の解答を比較し，両コードの差を埋めるようにフィードバックを与えるツールを提案している．比較する部分が文の構造のみなので，より細かい部分で教育意図と合致したプログラムであるかは判定することができない．

C-Helper [3] は，初学者の陥りやすいミスを指摘することができるが，課題ごとに込められた教育意図を踏まえた判定はできない．

CX-Checker[4] は，カスタマイズ機能を備えた C 言語プログラムのコーディングチェッカーである．XPath を用いて，ルールを追加することができ，独自のチェックをすることができる．しかし，課題ごとに一からルールを追加していくことを考えた場合，教育者の負担が大きくなる．

proGrep[5] は学習者のプログラムを収集して，その履歴に基づいたパターンを作成し，学習者に対して自動的にアドバイスを行う．課題に対する解答が教育者の教育意図に沿っているかを判定することはできない．

昨年度の研究 [1] では，教育意図が含まれている重要箇所を模範解答と学習者の解答から自動抽出した評価コードと呼ばれるコードを出力して，その差異から教育意図が含まれているかを確認するプルーフリーダを提案している．教育意図が含まれる重要箇所はマーキングパターンとよばれる重要箇所を抽出するためのパターンを用いることで自動抽出される．しかし，昨年度のプルーフリーダには 2 つの問題があった．1 つ目は判定できる学習項目が制御構文を対象にしたものであり，制御構文に含まれないコードの細かい部分が関係する教育意図が含まれる課題を評価することができないということである．2 つ目は抽出からは判定できない課題を評価できないことである．これらの問題から評価できる課題が狭くなっていた．

例えば，「配列の合計と平均を求めるプログラムを作成しなさい」という課題を例に挙げる．この課題の教育意図は「配列の 0 番目の要素から順に処理を行い，合計を求める」，「for 文内で冗長な計算をしない」である．このとき，模範解答のコード断片であるソースコード 1 に対し，学習者の解答としてソースコード 2 のようなコード断片が考えられる．

ソースコード 1　模範解答のコード断片

```
1 sum = 0;
2 for (i = 0; i < n; i++){
3    sum += a[i];
4 }
5 ave = sum / n;
```

ソースコード 2 学習者の解答のコード断片

```
1 sum = 0;
2 for (i = 1; i <= n; i++){
3    sum += a[i-1];
4    ave = sum / n;
5 }
```

ソースコード 3 ソースコード 1 の評価コード

```
1 $var=0;
2 for(=0;<;++){
3 $var=;
4 }
```

ソースコード 4 ソースコード 2 コード断片

```
1 $var=0;
2 for(=1;<=;++){
3 $var=;
4 }
```

この課題では，ソースコード 1 のように，繰返しを 0 から始めて合計の計算を行うことと平均の計算を for 文の繰返し後に行うことが適切である．しかし，ソースコード 2 では繰返しを 1 から始めて配列の添字を-1 することで合計の計算を行い，for 文内で繰返し平均を計算しているので，教育意図を満たさない不適切な解答であるといえる．昨年度研究で提案されたプルーフリーダでソースコード 1 と 2 の重要箇所を自動抽出して出力される評価コードはソースコード 3，4 となる．これらを比較することにより，繰返しを 1 から始めていることを指摘することはできるが，平均の計算については判定基準に含まれていない．このように昨年度研究のプルーフリーダはコードの細かい部分にある課題特有の教育意図を判定基準とすることができない．

3 カスタマイズ可能なプルーフリーダの提案

3.1 概略

2 節で [1] の問題点は，制御構文のみを対象とした評価範囲の狭さであると述べた．この問題から，制御構造だけをみる粒度の粗い抽出だけでなく，式などの細かい粒度の重要箇所を自由に抽出できる機能と抽出による判定では評価できない学習項目への対応が必要であると考えた．細粒度のコード片を抽出対象とする場合，課題ごとに抽出範囲が異なるので，教育者が範囲を自由に追加，削除できる必要がある．我々は細粒度のコード片を抽出範囲として加えることができるカスタマイズ機能を考えた．抽出による評価ができないユーザーが自由に定義する識別子は，呼び出された回数を確認することで評価できると考え，特定の字句列をカウントする機能を考えた．この 2 つの機能を昨年度研究のプルーフリーダに拡張することで，明解 C 言語入門編 [7] で取り扱われている課題を網羅することができる．

3.2 学習項目の追加

本研究では，明解 C 言語入門編 [7] の章立てを基に昨年度研究 [1] で対象であった条件分岐（if，switch），繰り返し（for，while，do-while），配列では判定できない課題を学習項目「関数」,「構造体」,「文字列」,「ポインタ」,「マクロ」の 5 つに分類した．また，5 つの学習項目の教育意図を文献 [7] の課題を基に考えた．その結果，関数とマクロでは「適切に呼び出されているか」，構造体では「適切に定義されているか」，文字列では「ナル文字を意識しているか」，ポインタでは「適切な場面でポインタを使用しているか」といった箇所が判定基準になると考えた．文字列やポインタの判定基準は，制御文以外の箇所も評価をする必要があり，従来の学習項目で行われていた制御構造からの比較判定では確認することができない．よって，後述する 3.3 節のカスタマイズ機能を使用して教育者が判定基準として加えるものとした．関数，マクロ，構造体は後述する 3.4 節の字句列の出現回数での判定で評価を行う．

3.3 カスタマイズ機能

本研究では，パターンに基づいたコード変換が可能である TEBA[6] を利用してカスタマイズ機能を実現することで，昨年度研究の問題点を解決する．カスタマイズ機能とは，教育者がプルーフリーダにマーキングパターンを自由に追加できる機能のことを指す．マーキングパターンの追加には TEBA[6] の構文変換機能を用いる．例として 2 節の課題で，for 文内で冗長な計算を行っていないかを確認したい場合に教育者が作成するマーキングパターンの記述をソースコード 5 に示す．マーキングパターンは抽出したい字句を**${名前:種別}**といった書式で表す．種別とは TEBA で字句を識別するために決められた要素の種類のことを指す．評価コードとして抽出したい箇所を**#**で囲む．例では，for 文内で平均の計算をしていないかを判定基準としたいので割り算の計算を抽出するマーキングパターンを記述している．このとき，評価コードで割り算が行われた箇所が分かるようにしたいので，演算子の/を**#**で囲む．

ソースコード 5 教育者が追加するマーキングパターン

```
1 ${e:EXPR}= ${e1:EXPR} #/# ${e2:EXPR} ;
```

マーキングパターンを作成することで教育意図が含まれる細粒度のコード片を抽出することができる．学習項目「繰り返し（for）」とカスタマイズ機能で追加したマーキングパターンを併用した 1 と 2 の評価コードはソースコード 6，7 のようになる．それぞれの評価コードを比較した際に平均の計算が行われている箇所が異なるので，学習者が for 文内で冗長な計算をしていることを判定することができる．

教育者がカスタマイズ機能を用いてマーキングパターンを追加することで，コードの細かい部分に記述された課題特有の教育意図を評価することができる．

ソースコード 6 カスタマイズ機能を併用したソースコード 1 の評価コード

```
1 $var=0;
2 for(=0;<;++){
3 $var=;
4 }
5 /
```

ソースコード 7　カスタマイズ機能を併用したソースコード 2 の評価コード

```
1 $var=0;
2 for(=1;<=;++){
3 $var=;
4 /
5 }
```

3.4　字句列の出現回数での判定

本研究では，特定の字句列の出現回数をコードから判定して模範解答と学習者の解答の比較を行う．特定の字句列とは，判定対象として教育者がツールに入力した関数名，マクロ名のことを指す．字句列の出現回数をカウントする評価方法と抽出を用いた重要箇所の比較による評価方法を組み合わせることで，従来のプルーフリーダで評価ができなかった課題を評価できる．

「階乗の計算をする関数 fact を定義して組合わせの計算を行う関数 comb を作成しなさい」という課題を例に考える．この課題の教育意図は，「定義した fact を適切な回数呼び出し comb を作成する」である．この課題に対して関数の利用方法を理解せずに，同じ処理をする関数を別々に定義してしまう学習者の解答があると想定される．このとき，模範解答と同じ関数名を使用しているかを学習者に確認をした後，適切に使用されているか判定したい関数 fact を特定の字句列として呼び出された回数を調べると，学習者の解答では定義した関数を各 1 回ずつしか呼び出していないので，不適切な解答であると判定できる．

「マクロを定義して 10 人の生徒のテストの点数の合計点と平均点を出力するプログラムを作成しなさい」という課題を例に挙げる．この課題の教育意図は二つあり，「for 文を利用して繰り返し点数を入力，計算を行う」と「マクロを使用して人数の変化に対応させる」である．この課題に対して，定義したマクロを使用できる箇所なのに使用していない場面がある学習者の解答が想定される．昨年度の研究 [1] では変数や定数を判定対象としていないので，for 文の構造が模範解答と一致していればマクロを使用していない学習者の解答も適切な解答であると判定される．抽出での判定に加え，マクロ名を特定の字句列として使用回数を調べることによって，マクロを使用していない学習者の解答を不適切な解答であると判定することができる．

4　プルーフリーダの設計と実現

本研究では，教育者が抽出範囲をカスタマイズでき，関数名，マクロ名の出現回数をカウントして評価コードを出力する評価コード生成ツールを実現した．プルーフリーダは模範解答と学習者のソースコードから評価コードを抽出して比較を行う．評価コード生成ツールの処理の概要を図 1 に示す．

教育者は模範解答のファイル名，課題を識別するための課題番号，評価する学習項目を評価コード生成ツールに入力する．評価コード生成ツールは，入力された学習項目にあわせて重要箇所の抽出，字句列の出現回数のカウントを

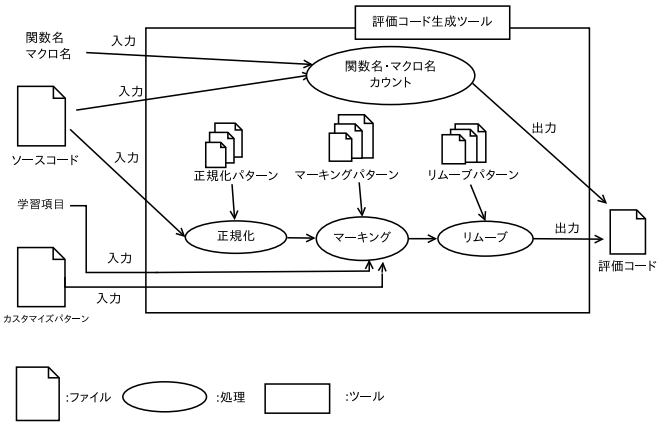

図 1　評価コード生成ツールの処理の概要

行う．コード内の重要箇所は 3 つのパターンをマッチングさせることで抽出される．正規化パターンでは，初期化式を含む宣言を宣言のみと代入文に分離させるなどのプログラムの正規化を行う．マーキングパターンでは，教育意図が含まれている箇所をコードから抽出するためのマーキングを行う．リムーブパターンでは，マーキングパターンでマーキングされた箇所以外の記述を削除する．カスタマイズ機能を利用する場合は，作成したパターンをマーキングパターンに追加する．学習項目「関数」,「マクロ」を選択した場合，教育者はツールに関数名，マクロ名を入力する．ツールは，入力された関数名，マクロ名の出現回数をカウントする．最後に，ツールは抽出，カウント結果を評価コードとして出力する．ツールの実装には，パターンに基づいたコード変換が可能な TEBA[6] を利用した．

5　プルーフリーダの評価

南山大学情報理工学部 2 年生を対象に開講された 2014 年度春学期プログラミング応用実習の課題 (59 問) とこれらの課題に解答をした 7 人の学生の解答を対象に本研究で提案したプルーフリーダの評価を行った．対象とした課題 59 問中 28 問は模範解答と異なる解答をした学生がいなかったので集計の対象外とした．本ツールを利用したところ 31 問中 20 問で不適切な学習者の解答を正しく判定することができた. 正しく判定できた課題として課題内容「ベクトルの要素と内積を求めるプログラムを作成しなさい」，教育意図「配列の要素を入力する関数と内積を求める関数を定義して，適切な場面で利用する」という課題があった．この課題に対して，配列の要素を入力する場面で定義した関数「initializeVector」を呼び出さず，関数と同じ処理を再度記述している学習者の解答があった．この解答と模範解答のコード内にある関数名「initializeVector」の出現回数をカウントしてソースコード 8，9 を作成し，比較することで，不適切な解答であると判定することができた．

ソースコード 8　模範解答の評価コード

```
1 initializeVector : 3
```

ソースコード 9 学習者の解答の評価コード

```
1 initializeVector : 2
```

また，判定ができなかった問題は 11 問あり，課題に対して模範解答が複数記述できることが原因であった. この問題の対処法に関しては，6.2 節で考察する.

6 考察

6.1 カスタマイズパターンの記述能力

カスタマイズパターンは，式や演算子などの細粒度のコードをマーキングして評価コードとして抽出可能である.

昨年度では文単位での抽出が可能であり，それにカスタマイズ機能を加えることで任意の構文要素を抽出できるようになった.

6.2 複数解答例がある課題

プルーフリーダの評価を行った結果，複数記述例がある課題に対して，適切な学習者の解答を不適切であると判定してしまうという問題がみつかった．複数解答例がある課題として，教育意図を満たしているが模範解答と異なる記述をしているものが挙げられる．この問題は教育者が複数模範解答を用意することで解決ができるが，作成しなければならない模範解答が膨大になり教育者の負担が大きくなってしまう．解決方法として，この研究に関連する 2014 年度の卒業研究 “プログラミング演習における模範解答派生機能を備えた学習用校正ツールの提案” [8] で提案された模範解答派生機能を利用することが考えられる．[8] では，派生パターンというパターンを利用して複数記述例がある課題の模範解答を派生させることができる学習用校正ツールを提案している．[8] で提案されている「for 文の解答から while 文の解答を作成する」といった派生機能を用いることで複数解答例がある問題に対応できるのではないかと考えた．本研究で提案するツールでは模範解答と学習者の解答を一対一で判定をしている．これからの課題として，複数の模範解答と学習者の解答を比較できるようにプルーフリーダを改善していくことが挙げられる.

6.3 フィードバック方法

本研究のプルーフリーダでは，模範解答の評価コードと学習者の解答の評価コードの比較結果から判定されたコードの差異をフィードバックとして学習者に出力をする．しかし，このフィードバック方法では，学習者がなぜ直さなければならないのかを理解できず，修正することができない可能性がある．この問題の解決策として，評価コードとして抽出される重要箇所を要素ごとにわけて，それぞれの部分が不一致であった場合のコメントを用意しておき，コメントをフィードバックとして出力するという方法がある．例えば，教育意図が「配列の先頭要素から最後まで順に走査して合計を求める」となる課題に対して，学習者が初期化を使い，配列の 2 番目から繰り返し処理をしていたとする．このとき評価コード判定ツールが for 文内の初期値の差異をみて，学習者に「for 文の初期値は適切ですか？」とコメントを自動出力する．それぞれの学習項目にこのようなコメントを用意することで学習者が理解しやすいフィードバックができると考えている.

7 おわりに

本研究では，学習者の解答に教育者の考える教育意図が含まれているかを確認，指摘を行うことを目的とした，カスタマイズ可能なプログラミング学習用プルーフリーダを提案した．昨年度ツールの拡張をすることで，関数，構造体といった字句列の出現回数での評価，教育者がプルーフリーダに教育意図を自由に追加できるカスタマイズ機能を実現した．今後の課題は，学習者へのフィードバック方法の改善，複数の模範解答と学習者の解答を比較するためのプルーフリーダの改善である.

参考文献

[1] 後藤悠太，長谷優磨，田原寛隆: 教育意図を利用したプログラムのプルーフリーダの提案，南山大学情報理工学部 2015 年度卒業論文，(2016).

[2] N.Truong, P.Roe, P.Bancroft: Static Analysis of Student’s Java Programs, Proceedings of the Sixth Australasian Conference on Computing Education - Volume 30, ACE’04, pp. 317-325，(2004).

[3] 内田公太，権藤克彦: C-Helper:C 言語初学習者向けツール C-Helper の予備評価，ソフトウェア工学の基礎 XIX 日本ソフトウェア科学会 FOSE2012，近代科学社，pp. 231-232，(2012).

[4] 大須賀俊憲，小林隆志，ほか: CX-Checker：柔軟なカスタマイズが可能な C 言語コーディングルールチェッカー，情報処理学会論文誌， Vol. 53， No.2，pp. 590-600，(2012).

[5] 長慎也, 筧捷彦: proGrep ープログラミング学習履歴検索システム, 情報処理学会研究報告．コンピュータと教育研究会報告，vol. 2005，NO. 15，pp. 29-36，(2005).

[6] 吉田敦，蜂巣吉成，沢田篤史，張漢明，野呂昌満: 属性付き字句系列に基づくソースコード書き換え支援環境，情報処理学会論文誌， Vol. 53， No.7，pp. 1832-1849，(2012).

[7] 柴田望洋: [新版] 明解 C 言語入門編，ソフトバンククリエイティブ株式会社 (2004).

[8] 堀尾美貴，金崎真奈美，佐藤成: プログラミング演習における模範解答派生機能を備えた学習用校正ツールの提案，南山大学情報理工学部 2014 年度卒業論文，(2015).