Vol.2016-CE-136 No.14
2016/10/16

プログラミング課題・作品評価補助システムの提案

島袋 舞子[1,a)]　小林 史弥[1]　林 康平[1]　兼宗 進[1]

概要：課題プログラムと作品プログラムを対象とした採点補助システムを提案する。小学校からのプログラミングの必修化により、小学校から大学までの幅広い年齢層を対象としたプログラミング教育が行われる予定である。本発表では、学習者の作成したプログラムを客観的に評価する仕組みを検討し、必ずしもプログラミングのエキスパートでない教員が生徒のプログラムを評価する際の手助けをするシステムを提案する。

キーワード：プログラミング教育, 自動採点, 作品評価支援, C, ドリトル

A Programming Evaluation System for School Lessons

Shimabuku Maiko[1,a)]　Kobayashi Fumiya[1]　Hayashi Kohei[1]　Kanemune Susumu[1]

Keywords: Programming Education, Auto Rating, Evaluation, C, Dolittle

1. はじめに

初等中等教育でプログラミング教育を行う取り組みには，科目として義務化する国も現れる [1] など，世界的な流れとなっている．国内でも，総務省がプログラミング人材育成に関する調査を実施する [2]，文部科学省がプログラミング教育の実践ガイドを公開する [3] など，プログラミング教育の推進が活発化しており，効果的なプログラミング教育の検討は，重要な課題となっている．

授業の中でプログラミングを扱うにあたっては、成績評価が必要になる。関心・意欲・態度などの授業内評価やペーパーテスト等の理解度評価は通常の授業と同等に扱うことができるが、提出された課題プログラムの評価や、プログラム作品の評価はプログラミングの経験のない教員にとっては負担となる可能性がある。

そこで本発表では、生徒のプログラムを 2 種類に分類し、それぞれの評価について、システム的にどのような補助が可能かを検討する。

[1] 大阪電気通信大学
Osaka Electro-Communication University, Neyagawa, Osaka 572–8530, Japan
[a)] shimabuku.m@gmail.com

2. 評価すべきプログラムの種類

生徒が作成したプログラムを教員が評価する場合について、大きく次の 2 通りを想定した。

(1) 課題プログラム: 与えられた仕様を満たす課題プログラムを作成する
(2) 作品プログラム: アニメーションやゲーム等の自由な作品プログラムを作成する

2.1 課題プログラム

課題プログラムは、与えられた課題の仕様を満たすプログラムを作成する。プログラムの内容は生徒ごとに異なって構わないが、動作は仕様のとおりである必要がある。課題プログラムは、計算や文字を出力する場合が多いが、グラフィックスなどが対象となる場合もある。工業高校や大学の専門課程など、専門的な授業において多く使われる。

仕様のとおりであることを確認するためには、一般には複数の入力で動作を確認する必要がある。たとえば「与えられた 2 つの整数値に対して、乗算の結果を表示せよ。2,4 の入力では 8 が出力される」という課題の場合、入力値によらず常に 8 が出力されるプログラムを正解から排除する

ために、「2,4」という入力に加えて、「3,5」などの他のパタンでの動作を確認する必要がある。

したがって、課題プログラムの採点にあたっては、一般に複数の入力を与えた上で、その出力が想定される値かどうかを確認することが必要である。3 章では、課題プログラムの自動採点について検討を行う。

2.2 作品プログラム

作品プログラムは、生徒が自分で作りたいプログラムを考えて、それを実現するプログラムを作成する。課題プログラムとは異なり、正解が存在しないことが特徴である。作品プログラムは、アニメーションやゲームなど、グラフィックスを利用する場合が多い。小学校から大学の一般共通教育など、情報の専門でない授業において多く使われる。

作品プログラムの評価としては、プログラムの動作を観察することで、アイデアや完成度などを評価することが一般的である。教員が実行して評価することもあれば、発表会の形で授業の中で生徒自身が実行し、相互評価を行うこともある。

このような実行評価を行う場合には、プログラムを読まずに実行結果だけを観察するため、プログラミングの授業であるにもかかわらず、プログラム自体の評価を行えていないという問題がある。そこで 4 章では、作品プログラムの評価に利用するためのプログラム分析情報について検討を行う。

3. 課題プログラムの自動採点

3.1 システムの提案

課題プログラムを自動で採点し、プログラムの評価を補助するシステムを提案する。課題プログラムを自動で採点するためには、次の点を考慮する必要があった。

(1) 入力値と出力値の定義付け

課題プログラムを採点するには、入力を与えた上で、その出力が想定される値かどうかを確認する必要がある。そのため、あらかじめ与える入力と想定される出力の値を定義しておく必要がある。

(2) 想定される出力値の曖昧さ

課題によっては、数値のみ出力や数値と単位を出力するなどの表記のゆれが現れる場合も想定される。**図 1** に具体例を示す。図 1 の例は、入力された値段から 2 割引した値段を表示する課題である。100 を入力値として与えたとき、80 と出力されることが想定されるが、「80 円」や「2 割引きの値段は 80 です」と出力するプログラムが存在する場合がある。このように、想定される出力には数値だけでなく、他の文字列が含まれる場合がある。

(3) 複数の出力値に対する正答割合

条件分岐を使用した課題プログラムを採点する場合は、条件によって出力が変化するため、複数の入力を与え、それぞれの出力が想定される値かを調べる必要がある。**図 2** に具体例を示す。この場合、入力に 0〜12 までの値を与え、想定される値が出力されるかを確認する。すべての入力で想定される値が出力されることが望ましいが、部分的に出力される場合でも一定の評価をおこないたい場合があると想定される。どの程度、入力に対して想定される出力を得られたかをわかるようにする必要がある。

課題 1. 入力の 2 割引の値段を表示するプログラムを作れ。
・与えた入力値： 100
・想定される出力値： 80
・実際の出力値： 80 円，2 割引の値段は 80 です

図 1 出力にゆれが含まれた課題例

【課題例】
月を入力し、季節を出力するプログラムを作れ。「12,1,2 は冬、3,4,5 は春、6,7,8 は夏、9,10,11 は秋」とする。

【解答例】

```
#include <stdio.h>
int main(void) {
    int x;
    scanf("%d", &x);
    if(x == 1 || x == 2 || x == 12) {
        printf("冬");
    } else if(x == 3 || x == 4 || x == 5) {
        printf("春");
    } else if(x == 6 || x == 7 || x == 8) {
        printf("夏");
    } else if(x == 9 || x == 10 || x == 11) {
        printf("秋");
    } else {
        printf("12 以外が入力されました");
    }
        return 0;
}
```

図 2 条件分岐を含む課題とプログラム例

(4) 特定の命令を使用したプログラム

特定の命令を指定する課題プログラムも存在する。ある処理を繰り返し実行させる際に、while 文のみを使用させたい場合や for 文のみを使用して記述させたい場合がある。そのため、特定の命令を指定できるようにする必要がある。

以上の点を考慮し、課題プログラムを自動で採点、プログラムの評価を補助するシステムを設計する。

3.2 設計

提案システムでは、プログラム実行時に何らかの入力を与え、得られる出力結果を比較し、採点をおこなう。採点

者はあらかじめ与える入力値と想定される出力値をテキストファイルに定義する。特定の命令を使用する課題が出題される場合もあるため、定義は入出力値だけでなく、ソースコード内に含まれる命令を指定できるようにする。想定される出力値の表記のゆれは、テキストファイルに定義するのではなく、システム内で処理をおこなうことで解決する。採点結果は、表形式のファイルに出力する。このとき、複数の出力値に対する正答割合を把握できるようにした。入出力値と同じように定義する方法も考えられたが、記述の手間を考え、今回は表形式のファイルに出力し、採点者が目視で確認できるようにした。

提案システムを使用し課題プログラムを自動で採点するには、(1) 教員が課題ごとに与える入力と想定される出力などをテキスト形式で独自のルールに従い定義する（以下、定義ファイル）。(2) 提案システムで提出された課題プログラムをコンパイル、定義ファイルに記述された入力値で実行し、出力結果と定義された出力値を比較する。(3) 比較結果を集計し、表形式のファイルで出力する。提案システムを利用し、プログラムを自動評価する流れを**図 3** に示す。

3.2.1 定義ファイルの作成

本システムでは、採点者が入力と想定される出力を独自のルールに従いテキストファイルに定義する必要がある。定義をする際の基本的なルールは以下のとおりである。

(1) 1 行目に問題番号を記述する
(2) 2 行目以降に書式に従い入力値と想定される出力値を記述する

1 行目に問題番号を記述し、2 行目以降に入力値と想定される出力値を記述していく。もし、ソースコード内で使用する命令を指定する場合は、問題番号の次の行に書式に従い記述する。入力と出力は半角のコロン「:」で区別する。入力値は「:」の左辺に記述し、出力値は「:」の右辺に記述する。複数の入力値から 1 つの出力値が得られる場合など、特殊な場合は書式に従って記述する必要がある。書式の一覧を**表 1** に示す。

複数の課題プログラムがある場合は、次の記述を行うときに 1 行空白を入れることで、1 つのテキストファイルに続けて記述する。記述したファイルは、テキストファイルとして保存する。

定義ファイルの記述例を**図 4** に示す。図 4 では、課題 1 と課題 2、課題 3 について定義した。課題 1 は、整数値を 1 個入力するとそれが偶数かどうかを判定し、結果を表示するプログラムである。2 行目では、入力値として「6」を与えたとき、「偶数」と出力される。3 行目では入力値として「5」を与えたとき、「奇数」と出力されると定義した。

課題 2 は、2 つの数を入力すると、べき乗を表示するプログラムである。6 行目では、入力値として「3」と「4」を与えたとき、「81」と出力されると定義した。

課題 3 は、for を使用し、数値を入力すると、その数の階乗を出力するプログラムである。2 行目で for 文をソースコード内で使用している。3 行目では入力値として「5」を与えたとき、「120」と出力されると定義した。

課題 1 のプログラム例を**図 5**、課題 2 のプログラム例を**図 6**、課題 3 のプログラム例を**図 7** に示す。

表 1 定義ファイルの書式

書式	意味
入力 A:出力 A	「入力 A」で「出力 A」を出力
入力 A, 入力 B:出力 B	「入力 A」または「入力 B」の入力
[入力 A, 入力 B]:出力 C	「入力 A と入力 B」の入力
入力 A:[出力 D, 出力 E]	「出力 D と出力 E」を出力
use(関数 A)	ソースコード内で「関数 A」を使用

```
課題 1
6:偶数
5:奇数

課題 2
[3,4]:81

課題 3
use(while)
5:120
```

図 4 定義ファイル記述例

```
#include <stdio.h>
int main(void) {
    int n;
    scanf("%d", &n);
    if (n % 2 == 0) {
        printf("偶数\n");
    } else {
        printf("奇数\n");
    }
    return 0;
}
```

図 5 課題 1 のプログラム例

3.2.2 採点の実行

採点を行う際は、data ディレクトリの下に生徒ごとのディレクトリを作成し、その中に採点するプログラムを置く。**図 8** にシステムのディレクトリ構造を示す。その後、提案システムにより課題プログラムをコンパイルし、定義ファイルに記述された入力値により実行をおこなう。実行結果が、定義ファイルに記述された出力と同じかどうかを比較し、同じであれば正解、異なっていれば不正解と判定し、結果を閲覧するための表形式のファイルを生成する。

3.2.3 採点結果ファイル

課題プログラムの採点結果は、表形式のファイルで閲覧

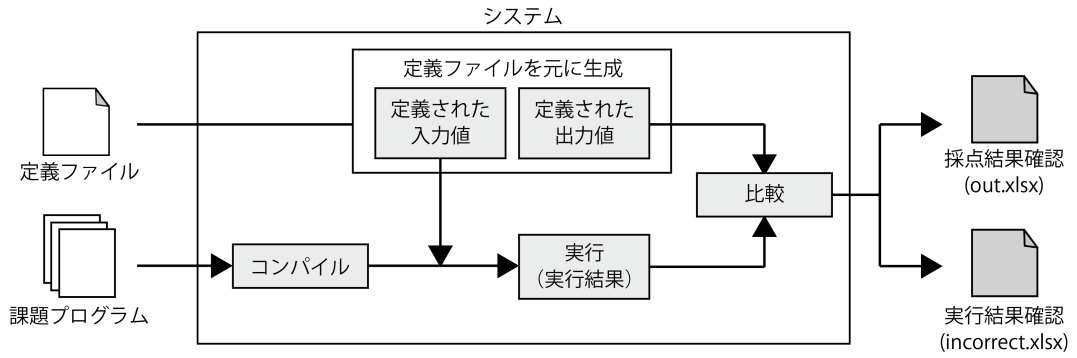

図 3　システム概要図

```
#include <stdio.h>
int main(void) {
    int i = 0,j,k,s = 1;
    scanf("%d",&j);
    scanf("%d",&k);
    while (i < k) {
        s = s * j;
        i++;
    }
    printf("%d",s);
    return 0;
}
```

図 6　課題 2 のプログラム例

```
#include <stdio.h>
int main(void) {
    int i,m,s;
    s = 1;
    scanf("%d",&m);
    for (i = 1; i < m+1; i++) {
        s = s * i;
        if(i == m) {
            printf("%d",s);
        }
    }
    return 0;
}
```

図 7　課題 3 のプログラム例

することができる。

表形式のファイルは「採点結果を確認するファイル (out.xlsx)」と「実行結果を確認するファイル (incorrect.xlsx)」の 2 つが生成される。採点結果を確認するファイルでは「評価一覧」と「問題別テスト通過数」の 2 つのシートが用意されている。

図 9 に評価一覧シートを示す。評価一覧シートでは、全体の正解数と問題別の正誤などを確認できる。問題ごとの判定結果は、「1」が正解、「0」が不正解、「–」は未提出を

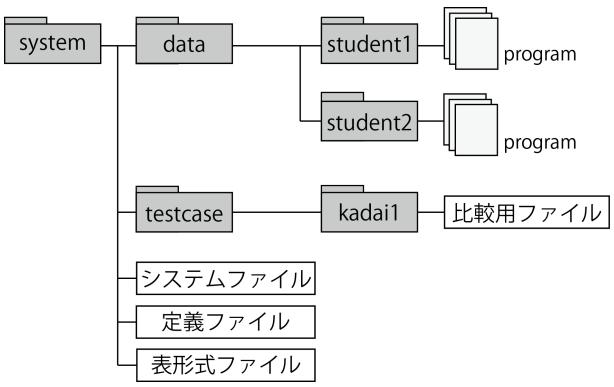

図 8　ディレクトリ構造

	正解数	未提出	要確認	kadai1	kadai2	kadai3
正解者数				87	67	79
正解率				86.2	67.4	76.3
student01	2	0	1	0	1	1
student02	3	0	0	1	1	1
student03	1	1	0	1	0	-

図 9　評価一覧シートの例

表している。不正解の場合でも、部分的な正解がある場合はセルの背景色を強調する形で表示される。評価一覧シートで確認できる項目を以下に示す。

- 生徒ごとの正解数合計
- 生徒ごとの未提出合計
- 生徒ごとの要確認合計
- 問題ごとの正解者数
- 問題ごとの正解率

問題別テスト通過数シートでは、実行結果と定義ファイルで定義した出力が一致した数を確認することができる。評価一覧シートで要確認と表示された項目に関して、実行結果と定義した出力がいくつ一致しているかを確認する際に使用する。不正解が含まれる場合でも、部分的な正解がある場合はセルの背景色を強調する形で表示される。以下は、問題別テスト通過数シートで確認できる項目である。

- 定義した出力数（テストケース）
- 問題ごとの不正解率
- 生徒ごとの実行結果と定義した出力が一致した数

	kadai1	kadai2	kadai3	kadai4	kadai5
テストケース	3	3	1	3	4
不正解率	13.8	32.6	23.7	25.7	29.7
student01	2/3	1	1	0	1/3
student02	1	1	1	1	1
student03	1	1/3	0	1	1

図 10　問題別テスト通過数シート例

kadai1	テスト番号			テスト番号		
入力	1	100	解答	1	80	
入力	2	200	解答	2	160	
入力	3	50	解答	3	40	
誤答率	13.8		誤答者数	25.7	提出者数	97
学生番号	入力	解答	出力値			
student01	100	80	73832 円です。			
student04	100	80	73832			
student07	100	80	値段は？ 73832			

図 11　実行結果確認ファイルの出力例

問題別テスト通過数シートの例を**図 10** に示す。

実行結果を確認するファイルでは、システムが誤答と判定した実行結果を確認することができる。**図 11** にファイルの出力例を示す。シートは問題別に分けられており、問題ごとに定義した入出力値と実行結果の値を確認できる。このファイルは、要確認の項目について、定義した出力値と実行結果を見比べての確認やどの入力値の部分で誤答と判断されているのかといった確認等に使用することを想定している。以下は、このファイルで確認できる項目である。

- 定義した入力と出力値
- 実行結果の値
- 提出者数
- 誤答率
- 誤答者数

3.3　実装

今回はバッチ形式の処理を行う形で実装した。処理を行うスクリプトは Python 言語で記述し、複数のスクリプトをシェルスクリプトで組み合わせて実行する。

処理は正誤判定定義ファイルに記述された問題ごとに、利用者フォルダの課題ファイルに対して処理を行う。

(1) 定義ファイルの問題ごとに定義を解釈し、実行時に入力に与える値と出力されるべき値を実行用のフォルダに格納する。
(2) 提出課題プログラムを gcc でコンパイルする。コンパイルの成功/エラーをログファイルに記録する。
(3) コンパイルしたファイルを実行し、結果の正誤判定結果をログファイルに出力する。
(4) ログファイルの結果を集計し、表形式のファイルを作成する。

(1) では、定義ファイルに記述された内容を正規表現を用いて 1 行ずつ解釈する。この結果をもとにし、各問題ごとにテストの入力、出力の内容が記述されたファイルを出力する。またそれと同時に比較をおこなう問題番号のリストも生成する。

(2) では、data ディレクトリ以下を全探索し問題番号に一致するファイルを対象に gcc でコンパイルする。このとき、コンパイルエラーが発生した場合はそのファイルをログファイルに書き出す。

(3) では、定義ファイルより得られた入力値を用いて課題プログラムを実行する。課題プログラムの実行は、問題ごとに全学生ファイル分おこない、定義ファイルに書かれた入力と出力の組み合わせの分だけ、実行される。もし、無限ループを含む課題プログラムが実行されてしまった時は、500 ミリ秒待って実行を取り消し、誤答として扱う。実行結果と定義された出力値を grep を用いて比較、正誤判定する。正誤判定の結果をログファイルに書き出す。前述した出力値の表記のゆれは、正誤判定を完全一致ではなく部分一致とすることで対応した。また大文字と小文字の区別もしないこととした。

(4) では、正誤判定のログファイルを基に、Excel のスプレッドシート形式のファイルを生成する。ファイル生成には、Python3 の Xlsxwriter ライブラリを用いた。

3.4　システムの評価

3.4.1　評価実験

提案システムにより課題プログラムの自動採点と評価の補助が可能かを、採点にかかった時間と正しくプログラムを採点した割合（精度）から明らかにする。

今回対象としたのは C 言語で記述された課題プログラムである。課題には、ループや配列の要素が含まれる。出題した課題を**図 12** に示す。先頭に付く英数字は問題番号を表している。評価対象者は 99 人で、提出された課題プログラムは 990 件である。

プログラムの採点は、以下の 3 つの方法で検証をおこない、採点にかかった時間と正しくプログラムを採点した割合を比較する。

(1) ソースコードを目視で採点
(2) 手動でコンパイルし実行結果を目視確認
(3) 表形式の結果ファイルを確認

(1) では、提出された課題プログラムのソースコードを 1 件ずつ目視で確認し、採点結果を表形式のファイルに記入する。(2) では、提出された課題プログラムを gcc でコンパイルし、実行結果を目視で確認、採点結果を表形式のファイルに記入する。(3) では、提案システムを利用する。事前に定義ファイルの記述方法を説明した後、提出された課題プログラムの定義ファイルを作成し、提案システムにより自動採点をおこなう。得られた表形式の結果ファイル

課題 1: 数を入力すると、その数だけ「*」を表示するプログラムを作れ。

課題 2: 数を入力すると、その数だけ「*」を表示するプログラムを作れ。ただし、3 の倍数番目は「#」を表示せよ。

課題 3: 数を入力すると、その数の階乗を表示するプログラムを作れ。(n! = 1 × 2 × ... × n)

課題 4: 2 つの数を入力すると、べき乗を表示するプログラムを作れ。(4^3 = 4 × 4 × 4)

課題 5: scanf で数を 5 回入力すると、それらの最大値を表示するプログラムを作れ。(for の中で 5 回入力すること)

課題 6: scanf で数を 5 回入力すると、それらの最小値、最大値、合計、平均を表示するプログラムを作れ。(for の中で 5 回入力すること)

課題 7: 配列の値の平均値を表示するプログラムを作れ。

課題 8: 配列の値の最小値と最大値を表示するプログラムを作れ。

課題 9: 配列の値が奇数のものは、その値を 2 倍の値で上書きし、それらの合計を表示するプログラムを作れ。

図 12 出題した課題

を確認する。

評価者はプログラミング系科目のティーチングアシスタント経験のある学部生 1 人である。3 種類の採点方法でそれぞれ採点にかかった時間と正しくプログラムを採点できた割合を比較する。

3.4.2 結果

評価実験の結果を**表 2** に示す。時間は採点にかかった時間を示し、精度は(正しく判定された結果数)/(全設問数)で求めている。課題プログラムのソースコードを目視によってチェックする (1) は最も多くの作業時間を要し、最も精度も低かった。誤採点の原因としては、実行結果を確認していないために、条件分岐の経路などの抜けをケースを見落としている場合が多かった。また、表への記載ミスも存在した。

手動でコンパイルし、実行結果を確認する (2) は高い精度で判定をおこなえているが、作業時間が多くかかっている。誤判定の原因としては、表への記載ミスが存在した。

提案システムを使用した (3) は、比較的短時間で高い精度の正誤判定を行えていることがわかった。作業時間の内訳は、「定義ファイルの作成に 13 分」「システムによる結果ファイルの生成に 4 分」「表全体に目を通すのに 2 分」である。誤判定の原因としては、整数値の平均を計算する問題で、想定していなかった別解が定義ファイルに記載されていなかった場合などが存在した。

表 2 評価結果

評価方法	時間 (分)	精度
(1) 目視チェック	555	93%
(2) 手動コンパイル実行	203	96%
(3) 提案システム	19	99%

3.5 考察

3 章ではプログラミング課題の採点を補助するシステムの提案をおこない、試作したシステムと評価実験をおこなった。実験の結果から、本システムを用いることで、作業時間を短縮することができ、採点の精度を高めることができることがわかった。

一方で、定義ファイルに記述ミスがあると、正誤判定の結果がすべて間違ってしまう可能性がある。今回の評価実験では見られなかったが、システムの動作確認をしている段階で、何度かこのようなミスが発生した。また、適切なインデントやコメントがあるかなど、ソースコードの整形は教員が目視するときに着目するポイントであるが、本システムでは評価に含めることはできない。

今回、学部生 1 名を対象に評価実験をおこなったが、今後は教員を含めた複数人での検証をおこない、上記の問題点を解決、改善することを検討する必要があると考える。

4. 作品プログラムの評価補助

4.1 作品評価分析の考え方

アニメーションやゲームなどの作品プログラムの評価は、プログラムを実行し、その動作を観察する形の評価が一般的である。小学校からのプログラミング必修化により、このような主観的な評価が広く行われることが予想される。プログラムを実行する形の評価では、アイデアや完成度などを確認することができ、基本的には妥当が期待できる。

しかし、実行だけの評価では「プログラミングの学習でありながらプログラムそのものを評価対象にしていない」という問題があることに加えて、全部の作品プログラムを読むことには労力がかかるという問題と、プログラムの質を評価することは教員にとって大きな負担になる可能性がある。

そこで、「ソースコードを読んでプログラムの工夫点を確認したほうがよい」と思われるプログラムの候補とその特徴点を提示するシステムを試作した。

4.2 作品分析システム

作品プログラムでは、作品ごとに異なるプログラムが書かれており、実行結果も作品ごとに異なるため、課題プロ

表 3　分析対象としたプログラム要素

要素	意味
文字, 行	プログラムのサイズ
分岐, 反復	制御構造の利用
関数	関数定義
乱数, 数学	数値関数の利用
複文	複数の文の反復など
インデント	字下げ、文ごとの改行など
コメント	コメントの記入
タイマー	アニメーション
グラフィックス	タートルグラフィックス
衝突	衝突イベント
色	三原色による色の合成
組図形	描いた図形をグループ化
GUI	ボタン、ラベル等の GUI 部品
音楽	音楽演奏

```
【作品プログラムの一部】
かめた＝タートル！作る。
色１＝色！１００　２００　１００　作る。
色２＝色！１２０　１２０　１２０　作る。

【作品プログラムの抽出結果】
student1.dtl:  色 (2) 数学 (24) 関数 (1)
```

図 13　作品プログラムと抽出結果の出力例

グラムと違い、「入力に対する出力の正誤判定」を行う形の評価は行えない。

そこで、ドリトル言語を対象に、プログラムから要素（行数などの統計量とオブジェクトと命令のグループ）の使用頻度を分析し、クラス全体の平均と比較して特徴的な要素を持つプログラムを抽出することにした。

表 3 に分析対象とした要素を、**図 13** に作品プログラムの一部と、それに対応する抽出結果の出力例を示す。この例では RGB の三原色でオリジナルの色を作っていること、三角関数などの数学関数を使っていること、自分で関数を定義して使っていることなどの情報が出力されている。

システムは Perl で記述した。program という名前のフォルダに作品プログラムを置いて実行すると、結果が out.txt に出力される。

4.3　授業作品の評価

2016 年度前期に大学で行われた 3 つの授業について、作品分析を行った。対象は医療系と機械系の学生であり情報は専門ではない。システムと比較するために、次の 3 種類の学生または教員による評価と比較した。**表 4** に実施した授業を示す。

(1) 授業 1 は学生の相互評価を行った。これはプログラムが専門でない教員と同等の、実行のみによる評価と考えることができる。

表 4　分析した授業

授業	受講生	人数	時間 (学習+作品)	事前授業
授業 1	機械系	66 人	2 時間+2 時間	C 言語
授業 2	医療系	35 人	12 時間+3 時間	なし
授業 3	医療系	41 人	12 時間+3 時間	なし

表 5　学生による相互評価とシステムでの抽出

評価	学生相互評価 (件)	システム抽出 (件)
上位	33	17
下位	33	4

(2) 授業 1 と授業 2 では、情報系でない学部を卒業した新任教員によるプログラム評価を行った。これはソースコードを見ることのできる一般の教員によるプログラム評価と考えることができる。

(3) 授業 3 では職業としてのプログラミング経験のある教員によるプログラム評価を行った。これはソースコードの質を正しく判定できる教員によるプログラム評価と考えることができる。

4.3.1　学生による相互評価への適用

授業 1 では、ドリトルのプログラミングを 2 時間学習した後、作品制作を 2 時間行い、授業の最後に発表会を実施した。学生は同級生の作品を見て、よいと思ったものに投票した。投票する件数等に制限は設けていない。

表 5 に、学生評価の高かった上位半数の作品と、下位半数の作品に対するシステムによる抽出結果を示す。システムが抽出した 21 件のうち、学生評価の上位には 17 件が、下位には 4 件が含まれていた。

後日、教員 A が全作品のソースコードを読むプログラム評価を行ったところ、下位で抽出された 4 件は優秀な評価に分類された。また、抽出されなかった下位のプログラムから優秀な評価に分類されたものは存在しなかった。

この結果から、実行による評価を行った場合でも、抽出された候補のソースコードを読むことで、実行だけではわからないプログラム上の工夫点を発見できる可能性を確認することができた。

4.3.2　新任教員による作品評価への適用

プログラミングの授業を初めて担当した教員 A が、授業 1 と授業 2 を対象に、次の評価作業を行った。

(1) 全作品について、プログラムを実行して作品を評価した。

(2) 全作品について、ソースコードを目視する形でプログラム評価を行った。

(3) システムからの抽出結果を参考に、再度ソースコードを確認した。

表 6 に授業 1 に関する評価結果を示す。教員 A は最初のソースコードの確認では、5 件の優秀な作品を判定した。続いてシステムで分析したところ、教員 A の判定した 4 件を含む 21 件の候補を抽出した。続いて教員 A が、出力さ

表 6　授業 1 に関する教員 A の評価とシステム抽出情報による修正

実行	コード	修正前	抽出	修正後
○	○	2	2	9
○	×	25	7	18
×	○	3	2	8
×	×	36	6	31

表 7　授業 2 に関する教員 A の評価とシステム抽出情報による修正

実行	コード	修正前	抽出	修正後
○	○	7	6	9
○	×	7	5	5
×	○	2	1	3
×	×	20	7	19

表 8　授業 3 に関する教員 B の評価とシステム抽出情報による修正

実行	コード	修正前	抽出
○	○	6	6
○	×	8	5
×	○	7	7
×	×	20	7

れた特徴要素を参考にソースコードを再度確認した結果、よいプログラムの判定について、次のような 13 件の追加と 1 件の削除を行うことができた。新たに抽出された 17 件のうち誤検出は 4 件であった。

- 抽出されなかった 1 件は、それほどよいプログラムではないことがわかった。
- 新たに抽出された 17 件のうち、13 件は工夫された箇所のあるよいプログラムであることがわかった。

表 7 に授業 2 に関する評価結果を示す。教員 A は最初のソースコードの確認では、9 件の優秀な作品を判定した。続いてシステムで分析したところ、教員 A の判定した 7 件を含む 19 件の候補を抽出した。続いて教員 A が、出力された特徴要素を参考にソースコードを再度確認した結果、よいプログラムの判定について、次のような 4 件の追加と 1 件の削除を行うことができた。新たに抽出された 12 件のうち誤検出は 8 件であり、教員 A が優秀と判定した作品のうち 1 件を検出することができなかった。

- 抽出されなかった 2 件のうち 1 件は、それほどよいプログラムではないことがわかった。
- 新たに抽出された 12 件のうち、4 件は工夫された箇所のあるよいプログラムであることがわかった。

4.3.3　プログラミング経験のある教員評価との比較

職業としてのプログラミング経験のある教員 B が、授業 3 を対象に、システムが抽出した結果を評価した。

表 8 に授業 3 に関する評価結果を示す。教員 B は全てのソースコードを読み込む作業を行い、13 件の優秀な作品を判定した。その結果をシステムの分析と比較したところ、候補として抽出した 25 件の中に、教員 B が優秀と判定した 13 件がすべて含まれていることを確認した。

4.4　作品評価の考察

3 つの授業に対する評価実験の結果を考察する。

学生による授業 1 の相互評価では、実行結果のみで評価を行った。ソースコードを見ずに評価を行うことから、小学校等の、普段プログラミングの授業を行わない教員の評価と近いことが予想される。多くのよい作品は実行結果から判断できると思われるが、表 5 に見るように、一部のよい作品は実行の評価が必ずしも高くない下位層にも存在することがわかった。このような作品をプログラミングの経験のない教員がソースコードから判別することは無理があるため、今回のようなシステムによるプログラムの抽出と、どこに特徴があるかというヒント情報を提供することは意味があると思われる。

新任の教員 A による授業評価では、実行結果とソースコードの両面から評価を行った。プログラミングを教えることはできるが他人のソースコードを読み慣れていないという意味では、中学校の技術科や高等学校の情報科などの教員の評価と近いことが予想される。実行による評価は的確に行えていると思われるが、ソースコードについては作品ごとの特徴に気づくことは難しく、授業 1 の評価では、システムの情報を参考にプログラムの評価を見なおした結果、1 件を対象から外し、13 件を加えることができた。このことから、システムの有効性が検証できたと考えている。一方、授業 2 の評価では、よいプログラムの抽出件数が授業 1 の 5 件（5/66=7.6%）から 9 件（9/36=25.0%）に増加した。これは授業 1 の再評価を通してソースコードを見る目が養われたと考えることができ、システムの提供する特徴要素の提示が役に立ったと考えている。

プログラミング経験の豊富な教員 B による授業評価は、大学や工業高校などの専門性の高い教員の評価と近いことが予想される。今回は、このような教員が選んだ作品をシステムが漏れなく検出できたことが成果と考えられる。

5.　関連研究

プログラムの自動採点は様々な手法を用いておこなわれている。与えられた課題の仕様を満たすプログラムを作成する課題プログラムの自動採点には、教員の模範解答プログラムと生徒のプログラムを main 関数で呼び出し、実行結果を比較、採点する手法 [4] やシステム上に用意された採点基準の項目を選択または数値を入力し、採点をおこなう手法 [5] が存在する。自動採点機能をプログラム開発環境の機能の一部とし、提供しているものも存在する [6][7]。また、プログラムの実行結果だけではなく、プログラムの処理をノード、データの参照関係をデータフローとしたグラフに変換し、正解と学習者のグラフを比較することで自動採点を実現しているものもある [8]。

作品プログラムの採点については、作品プログラムの中で使用した単語を抽出し、生徒同士を比較することで、作品

の独創性をはかる手法 [9] やあらかじめ決められた獲得概念のレベルに応じて作品を評価する手法 [10] が存在する。

6. おわりに

本研究では、課題プログラムと作品プログラムを対象とした採点補助システムを開発した。

決められたとおりに実装をおこなう課題プログラムでは、あらかじめ採点者が定義した入出力値などを利用し、システムで自動採点をおこなうことで採点の補助を試みた。採点結果を表形式のファイルに出力することで、目視での内容確認や部分点の評価を実現した。

2 種類の採点手法と比較実験をおこなった結果、採点作業時間を短縮することができ、採点の精度を高めることができることがわかった。一方で「定義ファイルを間違えて記述すると全ての採点結果が間違ってしまう」「ソースコードの整形を評価に含めることができない」といった課題があることがわかった。

作品プログラムでは、ソースコードを読んでプログラムの工夫点を確認したほうがよいと思われるプログラムの候補とその特徴を提示することで採点の補助をおこなう。今回はプログラムから要素の使用頻度を分析し、クラス全体の平均と比較し、特徴的な要素をもつプログラムを抽出し、採点者に提示した。評価実験をおこなった結果、システムの提供する特徴要素の提示が役に立ったことがわかった。また、プログラミング経験の豊富な教員が選んだ作品をシステムが漏れなく検出することができた。

謝辞 本研究は，科学研究費補助金（基盤研究（C）25350214）の補助を受けています．

参考文献

[1] Brown, N., Sentance, S., Crick, T., and Humphreys,S: Restart: The Resurgence of Computer Science in UK Schools, Trans. Comput. Educ. Vol.14, No.2, Article 9 (2014).
[2] 総務省: プログラミング人材育成の在り方に関する調査研究報告書 (2015).
[3] 文部科学省: 学校教育 – プログラミング教育実践ガイド, 入手先 <http://jouhouka.mext.go.jp/school/programming_zirei/> (参照 2016-02-06).
[4] 石原俊, 田口浩, 高田秀志, 島川博光: マークアップによる C 言語プログラミング試験採点システム. DEWS2007 (2007).
[5] 和田修平, 井上潮: 盗用発見と自動採点によるプログラミング演習課題の評価支援システム. DEIM Forum 2011 (2011).
[6] 板東慶一郎, 近藤大己, 長慎也, 松村真吾, 水越拓也: プログラム自動採点システム F-Java〜もっともっと演習を!〜, 情報教育シンポジウム 2006 論文集, Vol.2006, No.8, pp.119–124 (2006).
[7] 中村慎司, 筧捷彦: プログラミング授業支援システム WOJ の開発. 情報処理学会第 77 回全国大会講演論文集, Vol.77, No.4, pp.4939–4940 (2015).
[8] 大和雄一郎, 吉永泰甫, 竹内章. C プログラムの判断と重み付け採点を行う学習支援システム. 情報処理学会火の国シンポジウム 2012 論文集 (2012).
[9] 兼宗進, 長慎也: 文科系大学におけるサーバーサイドプログラミング授業の試み. 情報処理学会 コンピュータと教育研究会, CE(83), pp.141–148 (2006).
[10] 太田剛, 森本容介, 加藤浩: プログラム機能の自動分析機能とプログラム概念の自動評価機能を持つ Scratch 用プログラミング学習支援システムの開発. 情報教育シンポジウム 2016 論文集, Vol.2016, pp106–113 (2016).