プログラミング課題評価補助システムの提案

An Evaluation System for Programming Education

小林 史弥 † 林 康平 † 島袋 舞子 † 兼宗 進 †
Fumiya Kobayashi Kohei Hayashi Maiko Shimabuku Susumu Kanemune

1. はじめに

大学等の授業で利用可能な、プログラミング課題採点補助システムを提案する。同様の研究は数多く行われている [1][2]。対象は、結果を文字で出力するプログラミング課題である [3]。教員は課題ごとに、与える入力と、結果の想定される出力を定義する。システムは提出された課題プログラムをコンパイル・実行し、「提出の有無」「エラーの有無」「結果の正誤」を集計して出力する。今回は C 言語を対象に試作と評価を行った。

2. 対象とする課題の例

入力した整数が偶数かどうかを判定するプログラム例の一部を図 1 に示す。

```
int n;
scanf("%d", &n);
if (n % 2 == 0) {
  printf("偶数\n");
} else {
  printf("奇数\n");
}
```

図 1: 正解プログラム例

このプログラムは整数値を 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。プログラムが正しく作られているかどうかを調べるためには、入力として偶数を与えたときの結果と、奇数を与えたときの結果が正しく出力されていることを調べる必要がある。

この問題に対する正誤判定の定義例を図 2 に示す。1 行目の「A160718_1」は問題番号である。2 行目は入力値として「6」を与えたときに、「偶数」という文字が出力されることを判定する。3 行目は入力値として「5」を与えたときに、「奇数」という文字が出力されることを判定する。

```
A160718_1
6:偶数
5:奇数
```

図 2: 定義ファイル記述例

† 大阪電気通信大学, Osaka Electro-Communication University

3. 正誤判定用の定義ファイル

図 2 は入力値が 1 個だけの単純な例だったが、実際には複数の入力を使用したり、値の範囲を指定したりといった複雑な記述が必要な場合も存在する。現在対応している定義ファイルの書式を表 1 に示す。

表 1: 定義ファイルの書式

書式	意味
5:10	「5」の入力で「10」が出力される
5,6:10	「5」または「6」の入力で「10」が出力される
[1,2]:10	「1 と 2」の入力で「10」が出力される
5:[1,2]	「5」の入力で「1 と 2」が出力される

4. 実装

4.1 動作環境

今回はバッチ形式の処理を行う形で実装した。データはデータフォルダに学習者ごとのフォルダを作り、その中に課題番号の名前で課題ファイルを格納しておく。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形とした。使用した OS は Mac OS X である。

4.2 処理の流れ

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. 正誤判定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。

表形式の結果ファイルの例を図 3 に示す。

5. システムの評価

5.1 評価手法

C 言語の入門授業の第 9 回（for 文）と第 12 回（配列）の課題 10 個を対象として評価を行なった。人数は 99 人

	正解数	未提出	要確認	A160516_1	A160516_2	A160516_3
正解者数				87	67	79
正解率				66.666	33.333	66.666
eh10a023	2	0	0	1	0	1
eh11a002	3	0	0	1	1	1
eh12a002	0	3	0	未	未	未

図 3: 表形式の結果ファイルの例

であり、課題プログラム数は 990 件である。第 9 回の課題の一部を図 4 に示し、それに対応する定義ファイルの記述を図 5 に示す。

評価実験はプログラミング系授業の TA 経験のある学生 1 名が行なった。行なった評価は 3 種類である。

(1) ソースコードを目視で評価

(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

それぞれの評価実験について、所要時間と精度を測定した。精度は各評価の結果を照合し、判定結果が異なる場合にどちらが正しいかを確認した結果である。

A160516_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。
A160516_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。
A160516_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)

図 4: 第 9 回課題例

```
A160516_1
7:*******
5:*****
10:**********

A160516_2
7:**#**#*
5:**#**
10:**#**#**#*

A160516_3
5:120
4:24
2:2
```

図 5: 第 9 回課題例に対応した定義ファイル記述例

5.2 評価結果

評価結果を表 2 に示す。ソースコードの目視によるチェック (1) は最も多くの作業時間を要し、最も精度も低かった。誤判定の原因としては、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。表への記載ミスも存在した。

手動でのコンパイル実行 (2) は高い精度で判定を行えているが、作業時間が多く効率が悪いことがわかった。誤判定の原因としては、表への記載ミスが存在した。

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%

6. まとめ

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行い、試作したシステムと評価実験の結果を報告した。その結果、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を確認した。

今後は C 言語のバージョンの違いに対応したり、処理系ごとの結果の差異を調査したいと考えている。

評価については、今回は学生 1 名の作業と比較したが、今後は教員を含めた複数人での調査を行うことで改善などを行なっていきたい。

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] 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.

[2] 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, 2015.

[3] 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.