構文要素の差分を用いたプログラム作成問題の難易度分析

Difficulty Analysis of making Programming Exams using differences in Syntax Elements

中里 耕作[*1], 内田 眞司[*1], 本間 啓道[*1]
Kosaku NAKAZATO[*1], Shinji UCHIDA[*1], Yoshimichi HONMA[*1]
[*1] 奈良工業高等専門学校　情報工学科
[*1]Department of Information Engineering, Nara College, National Institute of Technology
Email: i10354@nara.kosen-ac.jp

あらまし：プログラミングの試験では，仕様に応じたプログラムを実装するプログラム作成問題が採用されることがある．しかし，試験問題の難易度を定量的に評価する指標がないため，作成者の経験により難易度を推測して作問されることが多い．本稿では講義中に提示されたプログラムと問題の正解プログラムの構文要素の差分と試験成績との関係を分析し，プログラム作成問題の難易度定量化の可能性を考察する．
キーワード：構文要素，作問，プログラミング教育

1.　研究背景・目的

プログラミングの講義では，受講者の理解度を評価するためにプログラム作成試験を実施する場合がある．プログラム作成試験では与えられた仕様に応じたプログラムの実装が求められる．ただし，プログラム作成問題の難易度を定量的に評価する指標が確立されていないために，問題の難易度は問題作成者の経験に依存する場合が多い．

一般的にプログラミング講義では，教材や課題などでサンプルプログラムが提示される．学習者は与えられたサンプルプログラムを動作確認などすることで文法などの学習内容を習得する．筆者らはこの講義で学んだ学習内容がプログラム作成問題にどの程度含まれているのかが，試験問題の難易度に影響を与えると考える．

本研究では講義で与えられたサンプルプログラムと試験問題間の差と試験の成績との関係を分析する．

2.　研究方法

De-gapper は，プログラムのソースコードを構文解析し，新しい構文要素（それ以前には出てこなかった構文要素）を検出して出力するツールである[1][2][3]．検出された構文要素は，「major な構文要素」と「minor な構文要素」に分類される．major な構文要素とは，今までのプログラムに出現していない新しい構文を表す，新たな学習概念の構文要素である．minor な構文要素とはそのような構文が書けるということを，学習者が他の major な構文要素から「容易に類推することができる」構文要素である．筆者らはこの 2 種類の構文要素がプログラム作成問題の難易度に影響を与えると考えた．

```
//プログラム 1
int n=10;
printf("%d", n+100);
```

```
//プログラム 2
int n=10+20;
printf("%d", n+100);
```

図 1　プログラム例

図 1 で示す 2 つのプログラムをプログラム 1，プログラム 2 の順で学習したとする．De-gapper で分析するとプログラム 1 では printf 文において，関数呼び出しの引数に加法式が書ける，加法式の中に識別子，演算子及び数値が書ける，という構文要素が検出される．プログラム 2 では，変数宣言の中に加法式が書ける，変数宣言の中の加法式に数値及び演算子が書ける，という構文要素が検出される．このときプログラム 2 が示す内容の本質は「加法式の中に数値及び演算子が書ける」こと，つまりプログラム 1 で検出されたものと同様であるので，プログラム 1 を理解していればプログラム 2 は「容易に類推することができる」と考える．

試験問題の解答プログラムに，講義や演習等で与えられるサンプルプログラムから「容易に類推することができる」部分（minor な構文要素）が多く含まれていれば，その問題の難易度は容易であると考えた．他方，新たな学習概念の構文要素は，従前に学習した構文要素から「容易に類推することができる」とは言えない．つまり，major な構文要素が多く含まれていればその問題の難易度は至難であると考えた．

以上より，以下の 2 つの仮説を立てる．

仮説 1 ：minor な構文要素を多く含むプログラム作成問題の難易度は低い

仮説 2 ：major な構文要素を多く含むプログラム作成問題の難易度は高い

3.　分析方法

分析対象講義は 2021 年度に奈良高専情報工学科 4 年生で実施された講義「プログラミングⅢ（通年）」である．受講者は３８名で講義では C 言語が採用され，与えられた要求仕様にしたがって要求仕様に従って，いずれかの手法により動作するプログラムを実装することが求められた．学生の理解度を評価す

る試験は年間 4 回（前期中間，前期末，後期中間，後期末）実施され，いずれの試験でもプログラム作成問題が 4 問出題されていた．図 2 は試験毎に講義で与えられたサンプルプログラム数とその概要である．De-gapper で分析する際には，分析対象試験からさかのぼったサンプルプログラムとの差分を計測する．例えば，前期期末試験「問題２」に含まれる構文要素を分析する際は，前期中間，前期末の講義で与えられたサンプルプログラム１６個との差分を計測する．

分析方法は以下の通りである．

1. サンプルプログラムと試験問題の解答プログラムを De-gapper[(2)]で構文解析し，major な構文要素と minor な構文要素を計測する．
2. 計測した構文要素と各試験の平均点，中央点との相関を分析する．

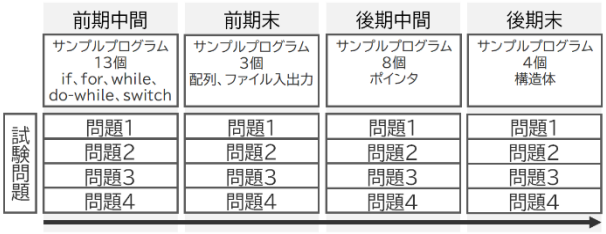

図 2　講義概要

4.　分析結果と考察

図 3 に各試験問題の平均点とプログラム 1 行あたりの major な構文要素の数を示す．割合が大きいほどプログラム 1 行あたりの major な構文要素が多いことを表している．またグラフ中の点線は回帰直線を表している．図から前期末試験が major な構文要素と平均点に強い負の相関がみられた．このことから，プログラム作成試験問題の難易度を構文要素の差から計測することができる可能性が示唆された．

最も平均点が低かった後期中間試験問題 4 はポインタによる文字列操作の問題であった．ポインタを示すソースコードの構文要素は minor な構文要素として分類されていた．C 言語を学ぶ上で、ポインタの学習は最大の難関といわれている[4]．一方でポインタを示す構文要素は複雑なものではない．このことから構文要素の差のみで難易度を計測することは困難であることがわかった．

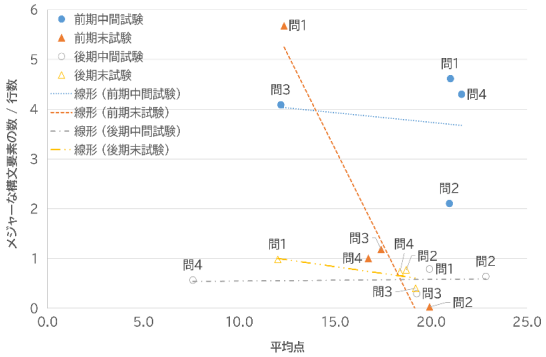

図 3　分析結果

5.　まとめと今後の課題

本稿では講義中に提示されたプログラムと問題の正解プログラムの構文要素の差分と試験成績との関係について仮説をたて、実際に実施された講義のデータを用いて分析した．その結果，プログラム作成問題の難易度定量化の可能性が示唆された．

今後の課題として，サンプル数を増やして分析を行なうこと，平均点が高い，もしくは低い正解プログラムに頻出する構文要素の特徴分析などがある．

参考文献

(1) 長慎也，保福やよい，西田知博，兼宗進:”De-gapper—プログラミング初学者の段階的な理解を支援するツール”，情報処理学会論文誌，Vol. 55, No. 1, pp. 45–56, 2014.
(2) 保福やよい，長慎也，西田知博，兼宗進:”プログラム差分検出ツール「De-Gapper」による授業分析”，研究報告コンピュータと教育（CE），2013-CE-121, 8,pp.1-6,2013.
(3) hoge1e3. De-gapper. https://github.com/hoge1e3/degapper, 2016. 参照: 2023-01-30.
(4) 前橋和弥：新・標準プログラマーズライブラリ C 言語ポインタ完全制覇，技術評論社, 2017.