Vol.2015-CE-131 No.9
2015/10/10

プログラミング穴埋め問題における コメントと問題難易度の関連分析

村田　美友紀[1,a)]　掛下 哲郎[2]

概要： 我々は，プログラミング教育ツール pgtracer の開発を行っている．本ツールが出題する問題はプログラムとトレース表からなる穴埋め問題である．これまでの研究により，穴抜きの種類やプログラムに付加されるコメント等が，問題の難易度に影響を与えることが分かった．本稿ではコメントに注目し，その種類による問題難易度の違いを定量的に評価する．分析の結果，コードの意図（アルゴリズム）を説明するコメントが最も難易度に関係することが分かった．さまざまな種類のコメントを使い分けることで問題の難易度を制御できるため，学生の習熟度に応じた問題の提供や解答結果を用いた学生の習熟度の推定など，本ツールを用いたプログラミング教育の改善が期待できる．

Relationship Analysis between Difficulty Level and Comment Type in Fill-in-the-Blank Programming Questions

Miyuki Murata[1,a)]　Tetsuro Kakeshita[2]

Abstract: We are developing a programing education support tool pgtracer utilizing fill-in-the-blank question. The tool gives a fill-in-the-blank question that consists of a program and a trace table to a student. From our previous reseach, we have found that difficulty level of question is affected by the blank type and comment. In this paper, we quantitatively evaluate relationship between difficulty level of questions and comment type. Through the analysis we found that the comment which explains the intension of a program affects the difficulty level more than other type of comments. Because we can control the difficulty level of a question by using various types of comments, we can expect to improve programing education, such as providing questions with appropriate difficulty level and proper evaluation of achievement by analyzing student's answer, by utilizing pgtracer.

1. はじめに

プログラミング教育は理工系の大学や高専において重要性が高い．しかし，学生の学力低下に関する懸念やプログラミング実習時に教員や TA だけでは十分な指導が行えないなどの課題がある．そこで，我々は穴埋め問題を用いたプログラミング教育支援ツール pgtracer の開発を行っている [1], [2]．本ツールは大学等で広く普及している e-Learning システム Moodle[3] のプラグインモジュールとして動作し，プログラムやトレース表に対する穴埋め問題を出題する．

プログラム全体を記述する問題に比べ，穴埋め問題は学生にとって取り組みやすく，プログラミング初学者の学習支援としても有効である．また，学生の穴埋め行動を学習ログとして収集し分析することによって，学生の理解度や不得意箇所の特定など教育改善および個々の学生の学習支援に役立てることもできる．pgtracer はプログラミング言語に対する汎用性を持つが，本稿では C，C++を対象に議論する．

pgtracer が出題する問題は，XML で記述されたプログラム，トレース表，プログラム用マスク，トレース表用マスクの 4 種類のファイルから構成される．問題は教員が作成する．pgtracer は教員の負担を軽減するため問題作成機能として，XML ファイルへの自動変換機能，グラフィカ

[1] 熊本高等専門学校　生物化学システム工学科
[2] 佐賀大学 知能情報システム学科
[a)] m-murata@kumamoto-nct.ac.jp

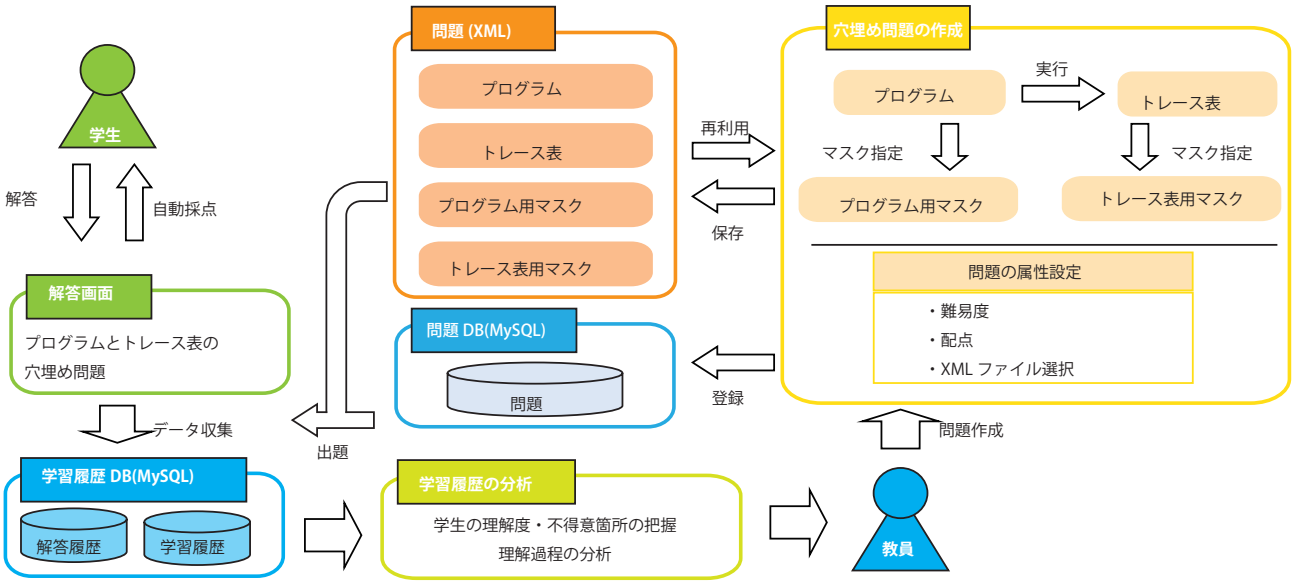

図 1　pgtracer を活用したプログラミング教育

ルなマスク指定ユーザーインターフェース，4 種類のファイルや問題難易度を指定するための問題定義機能を備えている．

同じプログラムであっても入力データを変更することによって，異なるトレース表が作成できる．また，プログラムやトレース表の穴埋めの箇所やコメントの表示の有無を変えることで複数のマスクファイルが作成でき，同じプログラムに対しても難易度が異なる問題を作成できる．問題の難易度を適切に設定することは，学生の理解度を正しく推定し，学習効果を高めるために重要である．

先の研究では，穴の種類とコメントの有無の異なる問題を用いて評価実験を行い，学生の得点と解答時間に基づいて，問題の難易度を定量的に評価した [4]．この結果，コメントが問題難易度に影響を与える重要な要因であることが分かった．また，[5], [6] により，適切なコメントがプログラムの可読性を向上させることが分かっている．プログラムの可読性が向上すると問題プログラムの理解が進むことから，コメントの付け方により問題難易度が変化すると考えられる．このため，コメントを用いて問題の難易度を制御できると考える．

本稿ではコメントに注目して問題難易度との関連を考察するため，評価実験を行い，その結果を定量的に分析する．コメントと問題難易度の関連が明確になれば，コメントによる問題難易度の制御が可能になる．問題難易度の制御が可能になれば，学生の習熟度に応じた問題の提供や解答結果による習熟度の推定が可能になり本ツールを用いたプログラミング教育の改善が期待できる．

本稿は以下のように構成されている．2 節では，pgtracer の概要を述べる．3 節では，C プログラムにおけるコメントを分類する．4 節では．学習ログ収集のために行った実験について述べ，5 節以降で分析を行う．

2. プログラミング教育支援ツール pgtracer の概要

pgtracer は，Moodle のプラグインモジュールとして開発している．**図 1** に pgtracer を活用したプログラミング教育の概念図を示す．

pgtracer が出題する問題は教員が作成するが，pgtracer は教員の問題作成を支援するための問題生成機能を備えている．問題は，プログラム，トレース表，プログラム用マスク，トレース表用マスクといった 4 種類の XML ファイルから構成されている．プログラム，トレース表と穴抜き部分を指定するマスクを分離することによって，同じプログラムに対して異なるマスクを定義できる．また，プログラムやトレース表の各行について表示/非表示を設定できるため，コメントの表示/非表示を制御することもできる．4 種類の XML ファイルを組み合わせることで，多様なプログラミング穴埋め問題を定義できる．

登録された問題は DB に格納される．pgtracer は登録された問題情報を元に問題一覧を生成し，表示する．学生が問題一覧から解答する問題を選択すると，pgtracer は問題を表示する．

学生の解答画面では，穴抜きされたプログラムとトレース表が並んで表示される．学生が入力した解答は自動採点される．自動採点の際に，プログラムまたはトレース表の XML ファイルに記載された値との文字列比較に加え，学生

表 1　コメントの分類

目的・用途	主な記述位置
著作権，作者，バージョン情報	ソースファイルの冒頭
変更履歴，変更理由	ソースファイルの冒頭
目的使用方法の説明	ソースファイルの冒頭 関数・メソッドの宣言位置
引数，戻り値の説明	関数・メソッドの宣言位置
変数，データ構造の説明	関数・メソッドの内側
内容の局所的な説明	関数・メソッドの内側
コメントアウト	不特定
その他	不特定

が解答したプログラムをコンパイル・実行し，その実行結果と正解プログラムの実行結果の比較も行う．たとえば，正解文字列が「ans += i」であったとき，「ans = ans + i」も正解とする．

学生が解答を入力するたびに，学生の入力文字列，正解文字列，正誤，入力終了時刻が解答ログとして収集される．また，問題 ID や採点結果，学習開始時刻などが学習ログとして DB に保存される．これらログを解析することによって，学生の理解度や不得意箇所の特定など教育改善および個々の学生の学習支援に役立てることができる．

3. コメントの分類

プログラムに記述されるコメントにはいくつかの種類があるが，[7] によって **表 1** のように分類されている．コメントアウトは，コードの変更などため不要になったコードを削除せずにコメントとして残しているものである．

評価実験の被験者を務めた学生は，実験時において関数を学習していない．このため，本稿では関数・メソッドは取り扱わない．また，配列や構造体などのデータ構造も未学習であるため，変数は基本データ型の変数のみを扱う．また，プログラミング能力を評価するための問題プログラムとしては，著作権や作者，変更履歴などの情報は必要ない．また，同様にコメントアウトも不要と考える．よって，本実験で扱うコメントの種類としては，大きくプログラム全体の説明，変数の説明，内容の局所的な説明の 3 種類とする．

一般的なプログラムにおいては，内容の局所的な説明としてそのコードの意図（アルゴリズム）についての補足的な説明が記述される．しかし，プログラミング教育を目的とした場合は，学習者の理解を助けるために，そのコードが実行する処理を直訳的に説明した文章がコメントとして記述されることがある．例えば，**図 2** の 8 行目のコードを考えると，コードの意図を説明するコメントとしては，「変数 ans の初期化」となり，コードを文章化するコメントとしては，「変数 ans に 0 を代入する」となる．本稿ではこのようにコードの処理内容を直訳的な文章で表現することを「コードの文章化」と呼ぶ．

```
1  //1から10までの整数の和を計算する
2  #include<stdio.h>
3  int main(){
4          int i;     //forの制御変数
5          int ans;   //計算結果
6
7          //変数ansの初期化
8          ans = 0;
9          //1から10までをansに加える
10         for（i=1; i<=10; i++){
11                 ans += i;
12         }
13         return 0;
14 }
```

図 2　コメント付プログラムの例

表 2　本稿で対象とするコメントとその記述場所

コメントの種類	記述位置
プログラム全体の説明	ソースファイルの冒頭
変数の説明	変数定義の直後
コードの意図の説明	コードの直前
コードの文章化	コードの直前

表 3　プログラミング科目の平均点

グループ	人数	科目の平均点
V	3	72.0
W	3	72.0
X	4	74.3
Y	4	75.5
Z	4	73.0

以上の理由により，本稿では**表 2** に示す 4 種類のコメントを用いる．

4. 評価実験

4.1 評価実験の目的と方法

平成 27 年 7 月下旬から 8 月上旬にかけて評価実験を行なった．実験の目的は，コメントによる問題難易度への影響を定量的に評価することである．

被験者は熊本高等専門学校八代キャンパスの 2 年生 18 名である．本キャンパスでは 2 年時からプログラミング教育が始まり，評価実験を行なった平成 27 年 8 月までに，定数，変数，選択構造，反復構造を修得済みであった．

実験は 2 回に分けて行なった．1 回目は 7 月下旬に行い，pgtracer のユーザー登録と使用方法について説明した．2 回目は，8 月上旬に行い，評価実験用の問題への解答とアンケートへの回答をしてもらった．2 回目の本実験の前に学生に pgtracer に慣れてもらうため，各自でシステムを使用する期間を設けたが，1 回目と 2 回目の間に期末試験があったため，この期間の使用はほとんど見られなかった．

2 回目の実験では，学生を 5 つのグループ（V グループ〜

表 4　コメント表示/非表示による問題のバージョン

バージョン	コメントの種類			
	プログラム全体の説明	変数の説明	コードの意図の説明	コードの文章化
ver.1	○	○	○	×
ver.2	×	○	○	×
ver.3	○	×	○	×
ver.4	○	○	×	○
ver.5	○	○	×	×

Z グループ）に分け，それぞれ異なる 5 つの問題を解答してもらった．ここで，学生のグループ分けには，2 年生が受講するプログラミング系科目の前期中間試験の成績を用い，グループ内の成績がほぼ同じになるようにした（**表 3**）．

また，問題解答時に学生の主観的難易度を評価するためのアンケートも実施した．これは，各問題ごとに難易度を 4 段階（とても易しい，易しい，難しい，とても難しい）から選んで回答するものである．また，各自が解答した 5 つの問題を易しい順に順位付けをしてもらった．

4.2　実験用問題

評価実験用の問題を作成するにあたって，コメントが与える影響を検証するために**表 4** に示すように，コメントの表示/非表示の異なる 5 つのバージョンを定義した．バージョン 1 とバージョン 2 の違いは，「プログラム全体の説明」コメントの有無である．よって，バージョン 1 とバージョン 2 を比較すると，「プログラム全体の説明」コメントが与える影響を検討できる．同様に，バージョン 1 とバージョン 3 を比較すると「変数の説明」コメントについて，バージョン 1 とバージョン 5 を比較すると「コードの意図の説明」コメントについて検討できる．また，バージョン 4 とバージョン 5 を比較することで，「コードの文章化」コメントについて検討できる．

評価実験用に**表 5** に示す 5 つの問（問 A〜問 E）を用意し，3 節で設定した 4 種類のコメントを付加した．穴の種類による問題の難易度を同程度とするため，穴はプログラムのトークンを 4 つ，トレース表の変数値（直前の値と変更があるもの）を 2 つの計 6 個に統一した．また，トークンの穴のうち，1 つは変数名とした．問題は 100 点満点とし，それぞれの穴の配点は等しくした．

よって，問が 5 問，バージョンが 5 つなので，25 通りの問題が作成できる．各グループが解答した問題を**表 6** に示す．ここで，A1 は問 A バージョン 1 の問題であることを示す．コメントが与える影響は，問 A〜問 E のそれぞれについてバージョン間の違いを比較することで評価できる．例として問題 C1 を**図 3** に示す．

5.　学習ログを用いた分析

実験では，18 人の学生がそれぞれ 5 問の問題を解答した．このうち，グループ X の学生 1 名が解答した問題 B1 は，正常にログが収集できなかったため，分析の対象から除外した．また，同じ問題を何度も解答した学生がいるが，この場合は最初の解答ログを分析の対象とした．

表 5　実験用に作成した問

問	説明
問 A	入力した得点によって，優/可/不可を判定して表示する．
問 B	数当てゲーム，入力した値が答より大きいか，小さいか，当たりかを表示．正解になるまで繰り返し，最後に当たった数を表示する．
問 C	入力した値（整数）の階乗を計算する．
問 D	入力した 10 個の実数について正数と負数の個数を数える．0 は正数とする．
問 E	正方形の一辺の長さを入力し，その正方形に内接する円の面積と円周の長さを計算する．

表 6　各グループが解答した問題

学生グループ	問題				
V グループ	A1	B2	C3	D4	E5
W グループ	A2	B3	C4	D5	E1
X グループ	A3	B4	C5	D1	E2
Y グループ	A4	B5	C1	D2	E3
Z グループ	A5	B1	C2	D3	E4

図 3　問題の例 (問題 C1)

表 7　グループごとの得点と解答時間，最初の穴埋めに要した時間

グループ	平均得点（点）	平均解答時間（秒）	最初の穴埋めに要した時間（秒）
グループ V	77.6	210.6	40.1
グループ W	84.1	161.9	39.4
グループ X	68.9	155.7	45.4
グループ Y	76.4	195.8	60.5
グループ Z	72.4	155.6	31.9

表 8　問ごとの平均得点と平均解答時，最初の穴埋めに要した時間

問	平均得点（点）	平均解答時間（秒）	最初の穴埋めに要した時間（秒）
問 A	80.3	170.8	37.5
問 B	70.4	186.9	50.4
問 C	76.6	145.2	32.9
問 D	76.5	140.6	28.5
問 E	72.8	232.1	69.1

5.1　グループごとの得点と解答時間

問ごとの平均得点と平均解答時間，学生が最初に解答した穴の入力が確定するまでに要した時間の平均を**表 7** に示す．これより，科目の成績ではグループ間のプログラミング科目の平均は同程度であったが，実験結果を見ると，得点の平均が最も高いグループ W と最も低いグループ X では 15.3 点の差があった．グループ X には平均点が 39.6 点の学生がいたため，平均点が低くなっている．また，解答時間を見ると，最も速いグループ Z と最も遅いグループ V では 55 秒の差があった．最初の穴埋めに要した時間については，グループ Y で最も時間がかかっているが，平均 337.2 秒を要した学生がいたため，平均時間も長くなっている．

よって，全体的には得点，解答時間ともにグループ間で平均化できているが，局所的に点数が低い，または解答時間が長い学生がいたことが分かる．

5.2　問ごとの分析

問ごとの平均得点と平均解答時間，最初の穴埋めに要した時間の平均を**表 8** に示す．

全体の平均得点は 75.8 点である．すべての問は平均の ±5 以内に収まっている．各問題は 100 点満点で，穴の数は 6 個であることから，穴 1 個あたりの得点が 16.7 点であることを考慮すると，問についての難易度の差はないと言える．また，全体の平均点が高く，解答時間も短いことから，問自体が易しかったと考えられる．

平均解答時間を見ると，問 B と問 E について比較的解答時間が長くなっている．最初の穴を埋めるまでの時間も同様に長くなっていることから，他の問題に比べてプログラムの内容が単純ではなく，理解するのに時間がかかったと考えられる．

5.3　コメントの種類ごとの分析

コメントの種類による比較するためには，バージョン間の差について検討すればよい．「プログラム全体の説明」については，バージョン 1 とバージョン 2 の差を検討することで，プログラム全体の説明がなかったときに，得点や解答時間がどのように変化するかを考察する．

コメントの種類ごとの影響を検討するため，**表 9** にバージョン間の得点，解答時間，最初の穴埋めに要した時間のそれぞれの差を示す．表 9 より以下のことが分かる．

(1) 問題プログラムの説明を非表示にすると平均解答時間と最初の穴を埋めるまでの時間は短くなるが，平均得点は変化がない．
(2) 変数の説明を非表示にしても平均解答時間，平均得点，最初の穴を埋めるまでの時間とも大きな変化は見られない．
(3) コードの意図を非表示にすると解答時間は長くなるが，最初の穴を埋めるまでの時間は短くなる．得点は低くなる．
(4) コードの文章化を非表示にすると平均解答時間と最初の穴を埋めるまでの時間は長くなり，得点は低くなる．

（1）について，バージョン 1 では「プログラム全体の説明」と「コードの意図を説明」するコメントの両方が表示されている．このため，学生は 2 種類のコメントと見比べながらプログラム全体を理解しようとするのに対し，バージョン 2 では「プログラム全体の説明」がないため，2 種類のコメント見比べる時間が短縮されたと考えられる．また，得点に差がないことから，本実験で用いた問題については，プログラム全体の説明がなくとも「プログラムの意図」があれば，プログラム全体についての理解ができたと考えられる．

（2）について，今回問題に用いたプログラムでは，変数名はそれが格納する値を推測しやすいように命名した．このため，「変数を説明」するコメントがなくても変数の役割について理解できたため，得点や解答時間，最初の穴埋めに要した時間すべてにおいて，影響が少なかったと考えられる．

（3）について，バージョン 1 では「プログラム全体の説明」と「プログラムの意図を説明」するコメントと見比べながらプログラム全体を理解しようとしているのに対し，バージョン 5 では「コードの意図の説明」がないため，その時間が短縮されと考えられる．また，得点が低くなっていることから，「コードの意図の説明」がない場合にはプログラムの理解が困難になると考えられる．

（4）について，バージョン 4 では「コードの文章化」が表示されているため，プログラム全体を理解していなくても，直前のコメントを読んで解答することが可能である．一方バージョン 5 は，「コードの意図の説明」や「コードの文章化」などコードに関する説明がない．このため，「プログラム全体の説明」だけでは，解答が困難になったと考えられる．

以上より，問題を難しくする効果があるのは，「プログラムの意図の説明」と「コードの文章化」などプログラムの局所的な説明に関するコメントであることが分かる．しかし，今回実験用に作成した問自体が易しかったことで，「プログラム全体の説明」や「変数の説明」について，実験結果に有意な差が得られなかった可能性も考えられる．

表 9　コメントの種類を比較するためのバージョン間の平均の差

コメントの種類	バージョン間の平均の差 平均得点（点）	平均解答時間（秒）	最初の穴埋めに要した時間（秒）	備考
プログラム全体の説明	3.0	−29.0	−11.0	ver.2 − ver.1
変数の説明	2.2	14.0	0.5	ver.3 − ver.1
コードの意図の説明	−9.8	26.0	−7.8	ver.5 − ver.1
コードの文章化	−14.7	33.3	9.6	ver.5 − ver.4

表 10　コメントの種類を比較するためのバージョン間の平均の差 (問題ごと)

コメントの種類	平均得点 （点） 問 A	問 B	問 C	問 D	問 E	平均解答時間 （秒） 問 A	問 B	問 C	問 D	問 E
プログラム全体の説明	0.0	10.7	12.5	12.3	−22.4	−57.3	21.0	−39.3	−23.3	−61.6
変数の説明	−32.1	10.7	17.1	8.0	2.8	23.4	−16.3	71.3	7.5	−31.3
コードの意図の説明	−23.8	−8.4	8.5	10.8	−44.3	−1.8	55.8	−14.5	31.2	61.7
コードの文章化	−16.8	−16.5	−8.3	−11.3	−25.9	−88.5	115.5	24.3	23.0	133.3

5.4　問題ごとの分析

問題ごとの平均解答時間と平均得点の平均を**表 10** に示す．表 9 と比較して，全体の傾向と大きく異なっている部分について検討する．

問 A についてみると，全体の傾向に比べて「変数の説明」が非表示の場合の得点の低下が大きい．これは，「変数の説明」を非表示にした問題 A3 について，得点が 16 点の学生がいたため，平均点が下がったと考えられる．また，「コードの文章化」を非表示にしたときの解答時間が短くなっている．これは，コードの文章化を表示した A4 について，解答に 601 秒を要した学生が 1 名いる．この学生は，解答中にプログラムやトレース表の見方について質問をしてきた学生であり，これに対する応答時間も含まれているため，解答時間が長くなったと考えられる．

問 B についてみると，「プログラム全体の説明」が非表示の場合に解答時間が長い．これは，「プログラム全体の説明」を表示した問題 B1 を解答したグループの平均解答時間が 157.2 秒，非表示の問題 B2 を解答したグループの平均解答時間が 210.6 秒であることが影響していると考えられる．

問 C についてみると，「コードの意図の説明」が非表示の場合の解答時間差が小さい．これは，「コードの意図の説明」を表示した問題 C1 を解答したグループの平均解答時間が 195.7 秒，非表示の問題 C5 を解答したグループの平均解答時間が 155.7 秒であることが影響していると考えられる．

問 D についてみると，「コードの意図の説明」が非表示の場合に得点の差が大きい．これは，「コードの意図の説明」を表示した問題 D1 を解答したグループの平均得点が 68.9 点，非表示の問題 D5 を解答したグループの平均得点が 84.1 点であることが影響していると考えられる．

問 E についてみると，「プログラム全体の説明」を非表示にした場合に，得点差が大きい．問 E は，四角形に内接する円の半径を求めるコードの一部を埋める穴で 50%の学生が不正解となっており，コードの意図の説明に加え，プログラム全体の説明がプログラムの理解を助けとなっていることが分かる．また，「コードの文章化」については，他の問よりも得点差が小さくなっている．コードの文章化を表示した問題 E5 で示したコメント中に変数名の記載ミスがあったため，75%の学生が不正解であったことが影響していると考える．このことから，プログラムやトレース表には正しい変数名が記載されているものの，学生が解答する際には，コードを文章化した直前のコメントを手がかりに解答を行なったと推測できる．

表 11　コメントの種類ごとの正解率の差

コメントの種類	変数	コード	変数値
プログラム全体の説明	−12.5	0.2	−4.6
変数の説明	−14.3	3.9	−4.6
コードの意図の説明	−11.1	−1.6	−15.7
コードの文章化	2.2	−16.7	−8.3

5.5　得点と解答時間の相関

得点と解答時間について，全体の相関係数は −0.18 であった．また，学生ごとに得点平均，解答時間の平均を用いた相関係数は −0.25，問題ごとに得点平均，解答時間の平均を用いた相関係数は −0.34 であった．

このことから，全体的には相関が見られないが，個人ごとの平均に弱い相関が見られるのは，前回の実験結果と同じ結果となった [4]．問題ごとの平均に弱い負の相関が見られることから，解答時間が長くなると得点が低くなる傾向があることが分かる．

5.6　穴の種類ごとの正解率

表 11 は，コメント，穴の種類による穴の正解率の差を示している．表 11 より，以下のことが分かる．

(1)「プログラム全体の説明」,「変数の説明」,「コードの

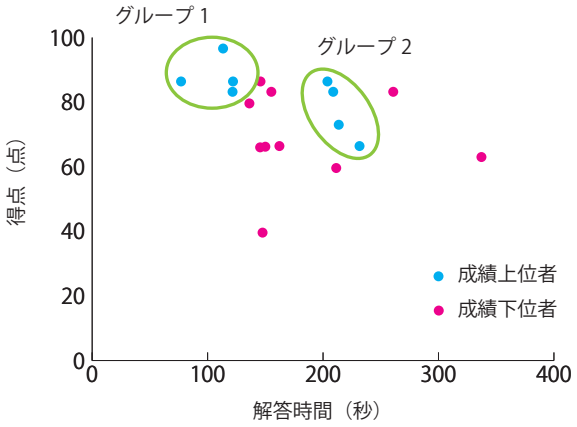

図 4　学生ごとの解答時間と得点の分布

意図の説明」をそれぞれ非表示にした場合，変数の正解率は同程度下がる．

(2)「コードの意図の説明」を非表示にすると，すべての種類の穴で正解率が低くなる．

(3)「コードの文章化」を非表示にするとコードの正解率が 16.7 ポイント低くなる．また，変数値の正解率も 8.3 ポイント低くなる．

「プログラム全体の説明」,「変数の説明」,「コードの意図の説明」の 3 つのコメントをすべて表示したバージョン 1 について変数に関わる穴の正解率は 100%である．(1) より，それらが変数解答に与える影響は同程度であることが分かる．

(2) について，変数値の穴を正しく解答するためには，プログラムの動作の理解が必要である．「コードの意図の説明」を非表示にすると，プログラムの動作の理解が不十分になり，変数値の正解率が低くなったと考えられる．よって，コードに関する説明がプログラムの理解を助けると考えられる．

(3) は，「コードの文章化」が表示されるとコードの正解率が上がることを示している．プログラム全体の理解が不十分でも文章化されたプログラムをそのままコードに書き換えるだけで正解できるため，コードに関する穴の正解率が高くなると考えられる．変数値についても「コードの文章化」を表示すると，変数値の正解率が高くなる．しかし，「コードの意図の説明」を表示した場合の正解率が 15.7 ポイント向上することを考慮すると，「コードの意図の説明」を表示した場合に比べて正解率は低い．このことから，「コードの文章化」は「コードの意図の説明」に比べて，プログラム全体を理解するには効果が薄いと考えられる．

5.7　学生ごとの解答時間と得点の分布

図 4 に学生ごとの解答時間と得点の分布を示す．ここで，グループ分けに用いた科目成績の上位者，下位者ごとにマーカーの色を分けて示している．

得点に注目すると，成績上位，下位の学生とも得点にほとんど差がないことから問題が易しかったと考えられる．また，科目の成績と実験の得点平均の相関係数は 0.66 とやや高い正の相関が得られた．このことから，問題難易度を精査することによって，pgtracer は学生のプログラミング習熟度を測るために活用できると考えられる．

また，図 4 において成績上位者はグループ 1 とグループ 2 に分類できる．解答時間が短いグループ 1 の学生は，全員情報システム研究部の学生であった．ここで，情報システム研究部とは，各種プログラミングコンテストへの参加を目的に活動するクラブである．このため，グループ 2 の学生に比べて，プログラムに接する機会が多く習熟度が高いことから，短い解答時間で問題を解くことができたと考えられる．また，グループ 1 の近傍に，得点が高い 3 人の成績下位の学生がいるが，このうちの 2 名も情報システム研究部の学生であった．

表 12　コメントの種類ごとの主観的難易度

コメントの種類	主観的難易度の差
プログラム全体の説明	-0.29
変数の説明	-0.12
コードの意図の説明	0.10
コードの文章化	0.33

6.　アンケート回答を用いた分析

本節では，アンケートの回答を分析する．これによって，学生が主観的に感じる難易度と得点，解答時間などから定量的に推測される難易度について検討する．

6.1　コメントの種類と主観的難易度

表 12 に，コメントの種類ごとの主観的難易度を比較するために，バージョン間の主観的難易度の差を示す．表 12 より，学生が主観的に感じる問題の難易度は，コメントの種類に影響しなかったと考えられる．「プログラム全体の説明」がない場合でも，「コードの意図の説明」があったり，「コードの意図の説明」がなくとも「プログラム全体の説明」があるため，それぞれが学生が解答するうえでのヒントになったと考えられる．そのなかでも，「コード文章化」を表示しない場合には，難易度が高くなっており，プログラミング初学者の学生にとっては，「コードの文章化」が解答の助けとなると考えられる．

問ごとの主観的難易度の平均は，問 A が 1.67 と最も低く，その他の問は，2.11 から 2.24 の間であった．これより，問自体の主観的難易度はほぼ平均化されており，また，全体の平均が 2.07 であることから，学生の主観的にも今回用いた問が易しかったことが言える．

6.2　学生ごとの主観的難易度

学生には，5 つの問題を易しい順に順位付けしてもらった．学生の解答一覧を**表 13** に示す．表 13 より，同じグ

表 13　学生ごと主観的難易度順位表

グループ	ID	易しい	←		→	難しい
v	u01	A1	D4	E5	B2	C3
	u02	A1	C3	B2	D4	E5
	u03	A1	B2	C3	D4	E5
W	u04	A2	C4	B3	D5	E1
	u05	E1	D5	C4	B3	A2
	u06	A2	E1	B3	C4	D5
X	u07	E2	B4	C5	D1	A3
	u08	C5	A3	B4	D1	E2
	u09	A3	B4	C5	D1	E2
	u10	B4	A3	C5	D1	E2
Y	u11	A4	D2	E3	B5	C1
	u12	A4	D2	E3	B5	C1
	u13	A4	E3	C1	D2	B5
	u14	A4	B5	C1	D2	E3
Z	u15	A5	D3	C2	E4	B1
	u16	C2	D3	E4	B1	A5
	u17	B1	E4	A5	D3	C2
	u18	A5	B1	C2	D3	E4

ループ内の学生が回答した難易度順には，いくつかの共通点が見られた．

最も易しいと答えた学生が最も多かったのは問 A（12 名）であった．中でも，すべてのコメントを表示したバージョン 1 とコードの文章化を表示したバージョン 4 では，これを解答したすべての学生が最も易しいと解答している．

一方，最も難しいと答えた学生が最も多かったのは問 E（8 名）であった．プログラム全体の説明を非表示にしたバージョン 2 では，解答した 4 名中 3 名の学生が，コードの内容に関する説明を非表示としたバージョン 5 では，解答した 3 名中 2 名が最も難しいと解答している．

また，グループ V では u02 と u03，グループ X では，u09 と u10 などグループ内での主観的難易度の順位が 3 問以上共通している学生の組が 4 組あった．

よって，問とバージョンを組み合わせると難易度の異なる問題が作成できるが，この問やバージョンの組合せを精査することで難易度の制御が可能となることを示唆している．

7. まとめ

本稿では，穴埋め問題を用いたプログラミング教育支援ツール pgtracer を使用して評価実験を行い，得られた学習ログとアンケートの回答を用いてコメントと問題難易度の関連を分析した．対象とするコメントは，被験者の学生の学習状況を考慮して，「プログラム全体の説明」，「変数の説明」，「コードの意図の説明」，「コードの文章化」とした．

学習ログを分析を分析した 5.3 節では，「プログラム全体の説明」，「変数の説明」，「コードの意図の説明」のうち，「コードの意図の説明」を非表示にすると最も問題難易度を高くできることが分かった．また，「コードの文章化」を表示することで最も問題難易度を易しくできる．一方，「プログラム全体の説明」や「変数の説明」は問題難易度に与える影響が少ないことが分かった．これにより，「コードの意図の説明」や「コードの文章化」などプログラムの局所的な内容を説明するコメントが問題難易度の制御に有効であることが分かった．

5.4 節では，「コードの文章化」を表示すると，プログラムやトレース表全体ではなく，穴の直前のコメントを手がかりに解答していることが分かった．また，5.6 節より「コードの文章化」は「コードの意図の説明」に比べてプログラム全体を理解する上では効果が薄いことが分かった．

5.5 節では，個人ごと，問題ごとの平均に弱い負の相関が見られることから，解答時間が長くなると得点が低くなる傾向があること，5.7 節では解答時間を利用してプログラミングの習熟度を評価できることが分かった．

アンケートの回答を分析した 6.2 節では，問とバージョンの組合せを精査することで，難易度の制御が可能となることが分かった．

本稿では，学生の学習状況を考慮して，対象としたコメントは 4 種類であったが，プログラムには関数やサブルーチンに関するものや変数が保持する情報など，他にもコメントの種類がある．また，変数や関数などの名前などもプログラムの理解に影響することから，これらと問題難易度の関係を検討するための評価実験を行い，問題難易度を制御する方法についてより具体的に検討していきたい．

参考文献

[1] Tetsuro Kakeshita, Ryo Yanagita, Kosuke Ohta, “A Programming Education Support Tool pgtracer utilizing Fill-in-the-Blank Questions: Overview and Student Functions”, Proc. 2nd International Conference on Education Reform and Modern Management (ERMM2015), Hong Kong, pp. 164-167，April 2015.

[2] Tetsuro Kakeshita, Kosuke Ohta, Ryo Yanagita, Mika Ohtsuki, “A Programming Education Support Tool pgtracer utilizing Fill-in-the-Blank Questions: Teacher Functions”, Proc. 2nd International Conference on Education Reform and Modern Management (ERMM2015), Hong Kong, pp.168-171, April 2015.

[3] Moodle.org, https://moodle.org/

[4] 村田，掛下：穴埋め問題を用いたプログラミング教育支援ツール pgtracer の問題難易度に関する考察, 情報処理学会 第 129 回コンピュータと教育研究会，2015.

[5] Tenny, T.:Program Readablity: Procedures Versus Comments, *IEEE Tran. Softw. Eng.a*, Vol14, No.9, pp.1271-1279，1988.

[6] Buse, R. P. and Weimer, W, R. :Metric for Software Readability, Proc 2008 Int’l Symp. Softw. Testing and Analysis, pp.121-130, 2008.

[7] 阿萬 裕久：小規模プログラムにおけるコメント行数とフォールト潜在性の関係に関する調査 (品質と測定), 電子情報通信学会技術研究報告. SS, ソフトウェアサイエンス Vol. 113, No. 24, pp.67-72, 2013.