Vol.2015-CE-129 No.18
2015/3/22

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

村田　美友紀[1,a)]　掛下 哲郎[2]

概要：　我々は，穴埋め問題を用いたプログラミング教育ツール pgtracer の開発を行っている．本ツールはプログラムとトレース表からなる穴埋め問題を出題し，学生が解答する．また，穴埋めが完了した時刻，答，正誤などの解答履歴を収集する．解答ログのうち穴の種類と解答時刻の分析により，解答時間が異なることを利用して穴の種類による難易度の違いを定量的に決定できる．本稿では，問題難易度が変化する要因を明らかにするため，評価実験を行なった．学生の主観的難易度と pgtracer が収集する解答時間や正解率の比較した結果，穴の種類とコメントが問題難易度に影響しており，これらを利用することによって問題難易度を制御できることが分かった．

Evaluation of Difficulty of the Problem for A Programming Education Tool pgtracer utilizing Fill-in-the-Blank Questions

Murata Miyuki[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 program and trace table to a student. When a student fills in a blank, pgtracer automatically collects log data such as time, correct answer and evaluation result. We then can evaluate the difficulty of a blank using answer time and the type of the blank. In this paper, we perform an experiment in order to clarify the factor which affects the difficulty of the problem. Through the analysis of the answer time, log record and the subjective difficulty of the student and teacher, we found that the blank type and the existence of comment affect the difficulty of the question so that we can control the difficulty by using the blank type and comment.

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]．本ツールは大学等で広く普及している e-Learning システム Moodle[2] 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．

[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

プログラム全体を記述させるような問題に比べ，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にも有効である．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ができる．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

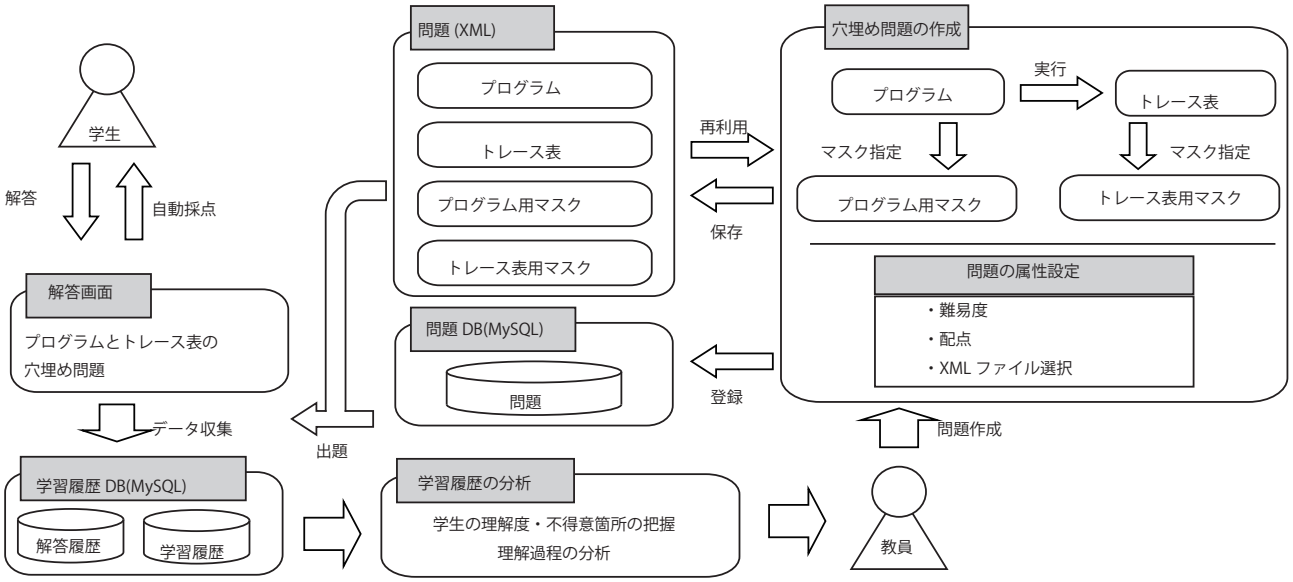


図 1 pgtracer を活用したプログラミング教育

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する難易度の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するために重要である．

pgtracer は，穴埋めが完了した時刻，答，正誤などの解答ログを収集する．pgtracer の運用実験の結果，解答ログのうち穴の種類と解答時刻の分析をすることで，穴の種類に応じた難易度の違いを定量的に決定できることが分かった [3]．

本稿では，この分析結果をもとに決定された穴の種類による難易度を用いて，穴埋め問題の学生の主観的難易度を評価するための実験を行った．プログラムを理解するのにコメントが影響することから，コメントの有無についても検討の材料とする．また，実験結果を分析することで難易度を制御する手法について検討する．

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 XML で記述された 4 種類のファイルから構成されている．これらのファイルのうち，プログラムとトレース表については，元となるソースファイルや入力ファイルを与えることで pgtracer が自動的に生成する．穴埋め部分を定義するためのマスクファイルについては，教員が表示されたプログラムやトレース表に対し，ドラッグ操作によって穴抜き部分を指定することで，pgtracer が XML ファイルを生成する．よって，教員は XML を直接編集することなく穴埋め問題を作成できる．

また，4 種類のファイルを指定して問題を登録，編集する機能も備えており，容易に問題を作成できる．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ため，難易度の異なる問題を複数作成できる．

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．学生は問題一覧から解答する問題を選択する．問題が選択されると，pgtracer はその問題を構成する 4 種類のファイルを用いて問題を作成し，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．学生は穴抜き部分に解答を入力する．学生が入力した解答は自動採点機能によって採点される．正誤は，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．

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，学習開始時間などは学習ログとして 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 [4]．本稿

表 1　穴の種類による解答時間

大区分	中区分	小区分	平　均（秒）	標準偏差
プログラム	トークン単体		9.75	27.0
	トークン列		19.45	49.7
	文全体		22.24	50.6
トレース表	変数値	変更なし	5.34	11.7
		変更あり	9.14	28.7
	ステップ番号	連続	5.04	9.88
		不連続	7.41	10.0
	変数名		11.55	77.7

表 2　穴の種類による問題の難易度

設定難易度	穴の種類				
	ステップ番号	変数値	変数名	トークン	文
1	○	○	×	×	×
2	×	○	○	×	×
3	×	○	×	○	×
4	×	×	×	○	○
5	×	×	×	×	○

（問 **1**）　解答した 10 個の問題を難易度別に分類してください．レベル 1 が最も易しく，レベル順に難易度が高くなるとし，各レベルの問題が 2 個〜3 個になるように分類してください．
（問 **2**）　（問 1）で回答したレベルについて，皆さんが感じた主観的な難易度を下記から選んでください．
A.　非常に易しい
B.　易しい
C.　やや易しい
D.　どちらでもない
E.　やや難しい
F.　難しい
G.　非常に難しい
（問 **3**）　pgtracer はプログラミング学習のサポートに有効だと思いますか．（5 段階評価））
（問 **4**）　今後も pgtracer を使用していきたいと思いますか．（5 段階評価））
（問 **5**）　pgtracer およびそれを活用したプログラミング学習について，感想，コメント，pgtracer の改善点などがあれば書いてください．

図 **2**　アンケート項目

においても，学生の解答行動を分析するために活用する．

3.　穴抜きの種類に応じた難易度分析

過去に行なった pgtracer の運用実験を通じて収集した学習ログから，穴の種類と解答時間に着目して分析した [5]．穴の種類については，プログラムに対してはトークン単体，トークン列，文全体，トレース表に対しては変数値，ステップ番号，変数名の 6 種類に分類した．また，ステップ番号，変数値については，直前のステップにおける値と連続であるか否かによって難易度が変化すると予想して 2 つに分類し，計 8 種類とした．

穴の種類ごとに解答時間の平均と標準偏差を**表 1** に示す．解答時間が長いほど，学生にとって難しい問題であると考えられることから，解答時間が長い順に以下のように難易度を定義した．

レベル 1　ステップ番号（連続），変数値（変更なし）
レベル 2　ステップ番号（不連続）
レベル 3　トークン単体，変数値（変更あり）
レベル 4　変数名
レベル 5　トークン列，文全体

t 検定を用いてそれぞれの穴の種類について比較したところ，変数以外の穴に関しては，別レベルの穴に対しては有意差が見られ，同レベルの穴に対しては有意差がないことが確認できた．変数名については有意差を得ることができなかった．これは，穴抜きのサンプル数が少なかったためと考えられる．

4.　評価実験

4.1　評価実験の目的と方法

我々は，平成 26 年 12 月から翌 1 月にかけて，熊本高等専門学校八代キャンパス（以下，熊本高専八代 C とよぶ）の 2 年生全員 128 名と生物化学システム工学科 4 年生 42 名を対象に pgtracer を用いた評価実験を行った．評価実験の目的は，学生や教員の感じる主観的難易度と pgtracer で収集した学習ログに基づく解答時間や正解率の比較を通じて，難易度が変化する要因を明らかにすることである．熊本高専八代 C には，機械電気系，土木建築系，生物化学系の 3 つの学科があり，全学科共通したプログラミング教育として 2 年次に情報工学基礎，3 年次にプログラミング基礎が開講されている．情報工学基礎では，変数，変数，選択構造，反復構造，関数，ポインタ，ファイル操作といった C 言語の基本文法を学習し，プログラミング基礎では，簡単なゲームやシミュレーションを扱ったプログラミング実習を行っている．

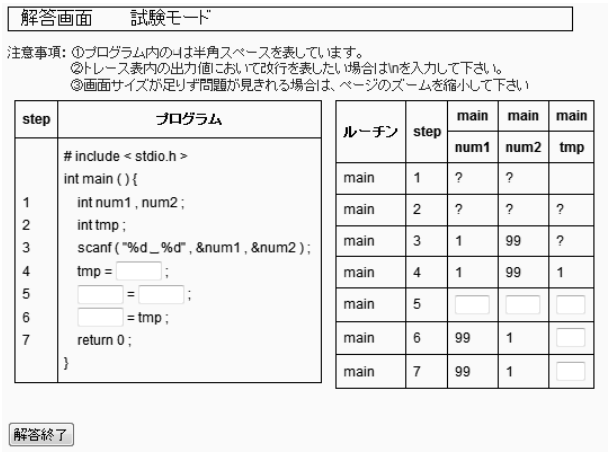
実験時において，2 年生は関数までを学習済みであった．また，熊本高専八代 C の学生は，情報工学を専門としていないため，プログラミングに苦手意識を持つ学生が多い．また，情報工学基礎を担当する 4 名の教員にも評価実験に参加していただいた．

評価実験は以下のように行った．穴の種類とコメントの有無によって設定した難易度の異なる 10 個の問題を，難易度を伏せて解答してもらった．その後，**図 2** に示す項目でアンケートを実施した．（問 1）では，10 個の問題を難易度別に 4 つのグループに分類してもらい，（問 2）では，それぞれのグループに対する主観的難易度を問う．

各問題について（問 2）のような主観的難易度を直接問わずにグループ分けを行ったことには，学生の主観的難易度の違いをより明確に表現させる目的がある．

表 3　出題した問題の穴抜きの分布

設定難易度	コメント	ステップ番号	変数値	変数名	トークン	文	説明
1	×	2	5				入力した実数のうちの最小の数を求める
1	○	3	4				入力した実数のうちの最大の数を求める
2	×		17	3			規則に従って○または●を表示し，○の数を求める
2	○		5	2			入力した整数について，正数と負数の数を求める
3	×		5		4		入力した 2 つの整数の値を入れ替える
3	○		15		5		入力した 2 つの整数の商余を引き算のみを使って求める
4	×				2	1	合計が 100 を超えるまで，入力された整数の合計を求める
4	○				4	2	身長と体重を入力し，標準体重と体重との差を求める
5	×				3	2	演算子と 2 つの実数を入力し，答えを求める
5	○					3	2 次方程式の係数を入力し，実数解の個数を求める


解答画面　試験モード

注意事項: ①プログラム内の□は半角スペースを表しています。
②トレース表内の出力値において改行を表したい場合は\nを入力して下さい。
③画面サイズが足りず問題が見きれる場合は、ページのズームを縮小して下さい

step	プログラム
	# include < stdio.h >
	int main ( ){
1	int num1 , num2 ;
2	int tmp ;
3	scanf ("%d _ %d" , &num1 , &num2) ;
4	tmp = [ ] ;
5	[ ] = [ ] ;
6	[ ] = tmp ;
7	return 0 ;
	}

ルーチン	step	main num1	main num2	main tmp
main	1	?	?	
main	2	?	?	?
main	3	1	99	?
main	4	1	99	1
main	5			
main	6	99	1	
main	7	99	1	

解答終了

図 3　問題の例（設定難易度 3，コメントなし）

4.2　評価実験用問題の作成

評価実験に用いる問題難易度の設定について述べる．3 節で述べた穴の種類による難易度を考慮し，問題に用いる穴の種類によって 5 つの難易度を設定した (**表 2**)．また，それぞれの難易度においてコメントの有無による 2 種類の問題を用意し，合計 10 個の問題をランダムに並べ替えて出題した．以下，実験問題作成用に設定した難易度を設定難易度と呼ぶ．出題した問題の穴抜きの分布を**表 3** に示す．

変数値を穴とする場合は，前後に表示された変数値がヒントとなってしまうため，出題意図に応じて穴の前後も合わせて穴に設定するため，穴の数が多くなる傾向がある．また，文全体を穴とする場合は，答が推測できる程度にするために穴の数が少なくなる．例として，設定難易度 3，コメント無の問題を**図 3** に示す．

5.　アンケート回答を用いた分析

5.1　有効なアンケート回答の抽出

アンケートの回答を集計するにあたって，**表 4** の条件を満足する回答を無効な回答とした．

（問 1）は，難易度別に 4 つにグループ分けするものであるが，「すべて 3 である」，「1 と 2 しかない」など，4 つのグループに分類していないものを無効とした．また，4 つのグループには分かれているものの，問題 1 から順に「1, 1, 1, 2, 2, 2, 3, 3, 3, 4, 4, 4, 4」というようにレベルが単純な昇順となっているものがあった．問題は難易度順ではなくランダムに出題されており，その主観的難易度が単純な昇順になるとは考えにくい．このため，信頼性が低いと判断し，回答を無効とした．

表 4　無効な回答の条件

問	無効とする条件
1	すべて同じ数字である 1 から 4 までのすべての数字がない 数字が単純な昇順となっている
2	数字が昇順でない 数字が 1 種類である

（問 2）は，分類した 4 つのグループについて，主観的な難易度を答えてもらうものである．（問 1）において，最も易しい問題のグループをレベル 1 として順に難しくなるように分類するよう指示しているため，（問 2）の回答は単純な昇順となるはずである．よって，単純な昇順でないもの，回答がすべて同じ数字であるものは無効とした．

表 4 に該当する回答を除外したところ，有効回答数は 2 年生が 34，4 年生が 34 であった．教員については，条件を満足する回答が 1 つあったが，サンプル数が少ないことから未回答項目を含む回答 1 つを除外し，残り 3 つはすべて有効回答とした．

5.2　学生の主観的難易度の分析

主観的難易度は，（問 1）で分類したレベル 1 から 4 に（問 2）で回答した主観的な難易度を割り当てたものである．**図 4**，**図 5** に問題ごとの学生および教員の主観的難易度の平均を示す．

まず，学生の主観的難易度について述べる．図 4 より，コメント無の問題については設定した難易度と学生の主観的難易度に一致が見られなかった．設定難易度 1 の問題は，3 つの入力した整数から最も小さい数を選ぶ問題である．本来は変数 min に小さい値を代入するところが，大き

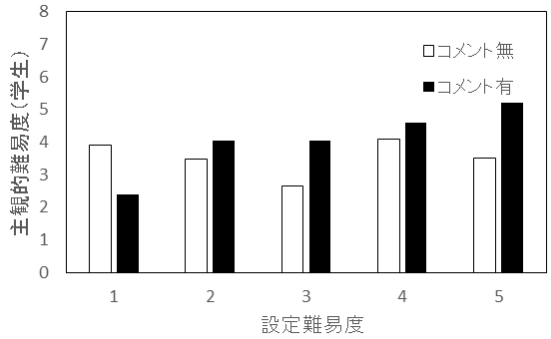


図 4 学生の主観的難易度

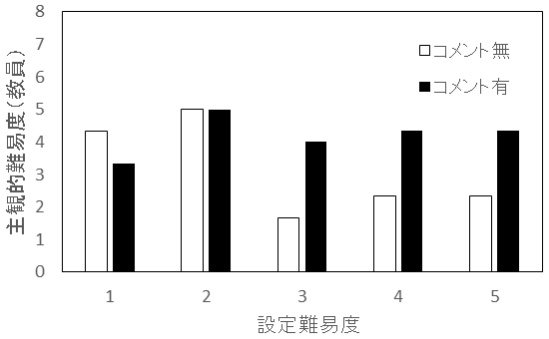


図 5 教員の主観的難易度

い値を代入するように誤ったコードが記述されていた．このため，変数名から推測される動作と異なる動作をしており，学生の解答を困難にし，学生の主観的難易度を高くしたと考えられる．

設定難易度 3 の問題は，2 つの値の入れ替えを行う問題である．教科書にも載っており，学生にとって既知の問題であったため，主観的難易度が低くなったと考えられる．また，設定難易度 5 の問題は，switch 文を用いた 2 つの変数の四則演算を行うプログラムであり，学生がどのようなプログラムかを比較的推測し易かったと考えられる．

このことから，変数名が学生のプログラムの理解を助けており，名前から推測される動作をしていない場合，学生の主観的難易度が高くなる．また，値の入れ替えなどプログラミングにおける定石のような問題や既知の問題であれば，学生の主観的難易度は低くなると考えられる．よって，コメント無の場合は，学生が問題プログラムの意図を理解できるかどうかが学生の主観的難易度の決定に依存していることが分かる．

一方，コメント有の問題については設定した難易度と学生の主観的難易度がほぼ一致することが分かる．これは，コメントによって学生がプログラムの意図を理解できているため，穴の種類による難易度が主観的難易度を制御できると考えることができる．

学生の成績による主観的難易度の違いを見るために，4 年生の回答について情報系科目の成績順に上位，下位の 2 グループ，上位 5 名分の主観的難易度を調べたところ同様の分布が観測された．下位 5 名の主観的難易度（**図 6**）については，他のグループよりも主観的難易度が高く，特に設定問題 2 は成績上位者に比べて 3.03 ほど高かった．成績下位の学生は習熟度が低いため，他の学生が問題にしないような基本的な事項について，つまづいている可能性がある．

5.3 教員の主観的難易度の分析

教員の主観的難易度について述べる．学生と同様にコメント有の問題については，設定難易度と教員の主観的難易度がほぼ一致する結果となった．

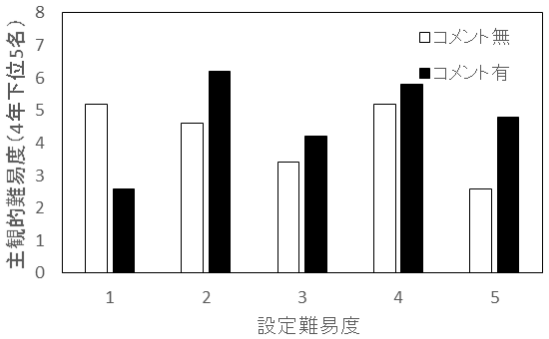
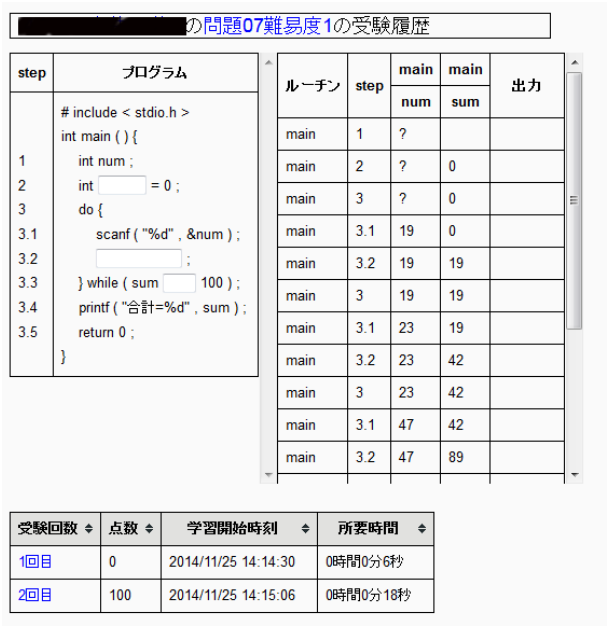


図 6 学生の主観的難易度（4 年，下位 5 名）

コメント無の問題については，設定難易度 1, 2 の問題に対する主観的難易度が高く，設定難易度 3，4，5 の問題が低かった．設定難易度 1，2 の問題は，トレース表の穴埋め問題であり，設定難易度 3，4，5 の問題はプログラムの穴埋め問題となる．

実験に協力いただいた教員へのインタビューでも，トレース表の変数値に対する穴埋めが難しかったとの意見を頂いた．教員がプログラムを理解するときもプログラムのコードから変数値の変化をトレースするが，1 行 1 行のコードをトレースするわけではなく，値の変化について大まかな傾向を把握する程度である．教員にとってトレース表の変数値に対する穴埋めは客観的な難易度としては高くないが，各ステップに変数値を当てはめる作業を面倒に感じたと考えられる．このため，主観的難易度を評価するときに，「難しい」と「面倒」が混同された可能性がある．

設定難易度 3, 4, 5 について，コメントの有無に注目して比較すると，コメント有の方が主観的難易度が高くなっている．教員へのインタビューにより，設定難易度 3 の問題は通常の算術計算を用いないで剰余を求めるアルゴリズムの理解が必要であったこと，設定難易度 4 では，身長の値について単位変換が必要なことが主観的難易度が高くなった理由との回答であった．また，設定難易度 5 の問題では 2 次方程式の判別式を計算するためのコードが穴抜き


[redacted]の問題07難易度1の受験履歴

step	プログラム
	# include < stdio.h >
	int main ( ) {
1	int num ;
2	int [ ] = 0 ;
3	do {
3.1	scanf ("%d" , &num) ;
3.2	[ ] ;
3.3	} while (sum [ ] 100) ;
3.4	printf ("合計=%d" , sum) ;
3.5	return 0 ;
	}

ルーチン	step	main num	main sum	出力
main	1	?		
main	2	?	0	
main	3	?	0	
main	3.1	19	0	
main	3.2	19	19	
main	3	19	19	
main	3.1	23	19	
main	3.2	23	42	
main	3	23	42	
main	3.1	47	42	
main	3.2	47	89	

受験回数	点数	学習開始時刻	所要時間
1回目	0	2014/11/25 14:14:30	0時間0分6秒
2回目	100	2014/11/25 14:15:06	0時間0分18秒

図 7 1 回目の学習を除外するパターン

箇所となっている．このコードに対応するコメントは「判別式 D を求める」であり，判別式を求めるための式は記載していない．このため，学生が判別式を知っているかどうかが解答に影響すると考えて難易度を高くしたとの回答があった．

設定難易度 3，4，5 はプログラムの穴埋め問題であるが，コメントを使用せずに学生がコードを推測できるようにするため，正解のプログラム自体がコメント有のものよりも易しいものになったと考えられる．そのため，プログラミングの習熟度が高い教員にとっては，主観的難易度が低くなったと考えられる．これは，正解のプログラムが難しいものであったとしても，コメントによって主観的難易度を低くできることを示す．

6. 学習ログを用いた分析

6.1 学習ログの活用

アンケートを記名式で行った 4 年生について，主観的難易度と問題解答について検討する．問題解答に関する情報は学習ログから収集する．一つ問題に対して一人の学生が複数回解答することがあるが，最初の解答を検討の対象とする．ただし，得点が 0 点であり，かつ解答時間が 20 秒未満または 1000 秒以上の解答は除外する．教員の解答の中で最も早いものは，解答時間 21 秒，得点 100 であった．このため，20 秒未満の解答は，問題を解くのに十分な時間経過がないと判断した．

また，pgtracer の学習ログ分析機能を用いて学生の解答行動を確認すると，**図 7** に示すように，短時間の学習の直後に高得点の学習を行うパターンが多く見られた．pgtracer では，採点後に問題の正答を確認できる．この機能を使っていったん解答を始めるがすぐに諦めて，正答を確認することで高得点を得ようとしたと考えられる．解答時間が 20 秒未満かつ得点が 0 点を条件とすることで，このような短時間の学習を分析対象から除外できる．直後の学習ログについては分析の対象とした．

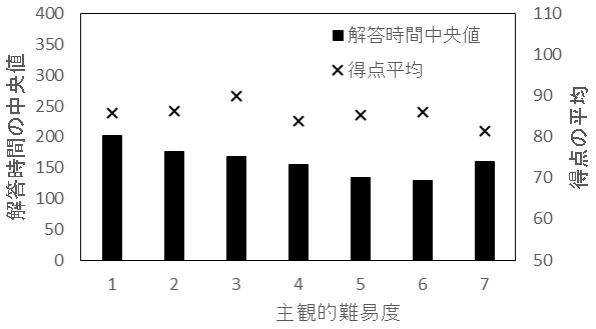


図 8 主観的難易度と得点平均および解答時間の比較

表 5 問題ごとの穴数

コメント無					
レベル	1	2	3	4	5
穴数	4	4	6	7	7
コメント有					
レベル	1	2	3	4	5
穴数	9	3	6	5	3

学習時間が 1000 秒を超える学習について解答行動を確認すると，他の穴に比べて非常に時間がかかる穴が含まれおり，途中で解答することを中断していることが分かる．このため，解答時間に解答行動が正確に反映されていないと判断した．

6.2 主観的難易度と学習ログ

図 8 に，主観的難易度別に得点の平均と解答時間の中央値を示す．主観的難易度が上がるにしたがって得点，解答時間とも下がっている．25 名の学生が主観的難易度が 7 であると解答した問題（設定難易度 5，コメント有，穴数 3）について学習ログを確認したところ，すべての穴について解答できていない学習ログが多く見られた．このように，主観的難易度が高い問題は，すべての穴埋めを完了せずに解答を中断する学生が増えるため，解答時間が少なくなったと考えられる．

6.3 主観的難易度と解答時間の関係

本節では解答に要した時間について，一つの穴あたりの解答時間数，問題ごとの解答時間数の度数分布を検討する．問題の穴の数は，表 3 に示す通りである．pgtracer は変数値の穴埋めをサポートするために，変数値の穴を埋めると入力した値をその後の穴にも自動的に入力する．そこで，本節では穴あたりの解答時間を求めるために，変数値が前の穴から変化する穴のみをカウントする (**表 5**)．

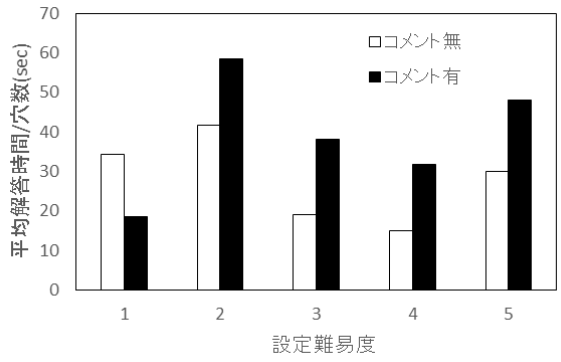


図 9　穴あたりの解答時間

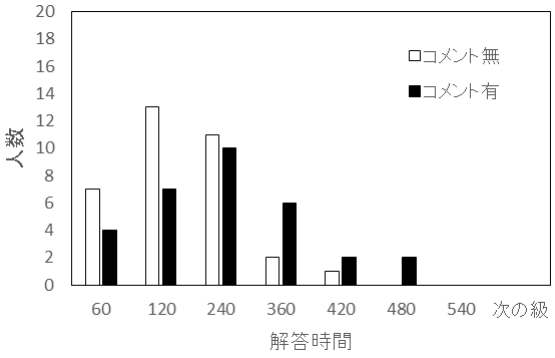


図 10　解答時間の度数分布（設定難易度 3）

図 9 より，コメント有の問題のほうが穴あたりの解答時間が長いことが分かる．また，コメント無の問題に比べ，コメント有の問題のほうが解答時間の度数分布のばらつきが大きい．例として，設定難易度 3 におけるコメント有，無の度数分布表を図 10 に示す．コメント無の問題について，解答ログを確認したところ，すべての問題を解かずに終了した学生が見られた．学生が問題を解答するのに必要な知識や問題意図を理解できなかったため，解答を諦めたと考えられる．また，コメント有の問題では，コメントをヒントにして試行錯誤しながら解答している様子が確認できる（図 11）．

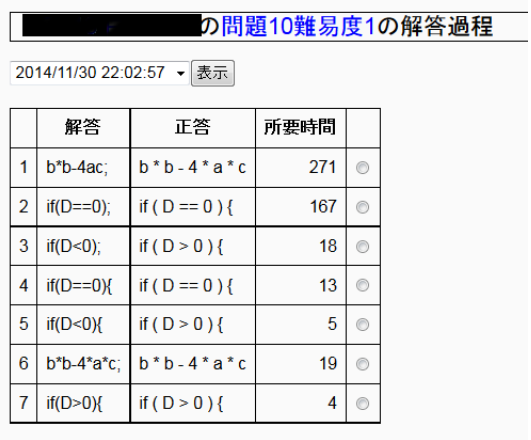
これより，コメント無の問題は，問題に対する知識や問題意図を学生が理解しているかを評価する上で役立つと考えられる．一方，コメント有の問題は，上記の知識や問題意図を学生に与えた上で，プログラムを書く能力や与えられたプログラムを理解する能力を評価する際に役立つと考えられる．

6.4　学生ごとの分析

本節では，学生ごとに求めた主観的難易度，解答時間，得点を用いて比較を行なう．

図 12 に解答時間の中央値と得点の平均の分布を示す．相関係数は 0.03 であり，相関は認められない．グラフ右下のはずれ値データ（5.33，502）が対応する学生の学習ログは表 6 のとおりであり，他の学生に比べて解答時間が長い．このデータを除外すると相関係数は 0.41 に向上し，解答時間の中央値と得点の平均値には中程度の相関が見られるようになる．図 13 は，主観的難易度と得点の分布を示す．相関係数は 0.21 であり，弱い相関がある．主観的難易度と得点の相関係数は −0.13 であり，相関はほとんどない．

次に主観的難易度と解答時間，主観的難易度と得点の相



	解答	正答	所要時間	
1	b*b-4ac;	b * b - 4 * a * c	271	
2	if(D==0);	if (D == 0) {	167	
3	if(D<0);	if (D > 0) {	18	
4	if(D==0){	if (D == 0) {	13	
5	if(D<0){	if (D > 0) {	5	
6	b*b-4*a*c;	b * b - 4 * a * c	19	
7	if(D>0){	if (D > 0) {	4	

図 11　解答ログ（設定難易度 5，コメント有）

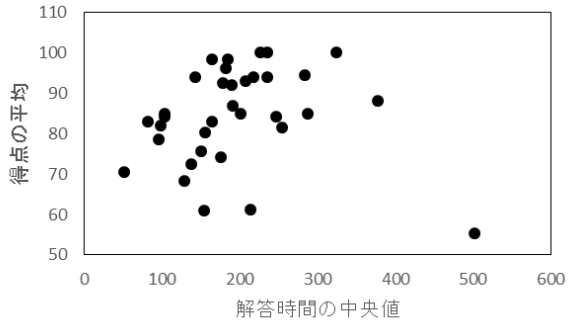


図 12　学生ごとの得点と解答時間の分布

表 6　図 12 のデータ（502，20）詳細

問題	解答時間の中央値	得点の平均値
1	385	42
2	740	80
3	381	44

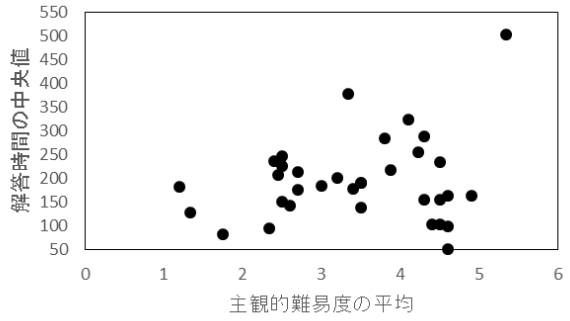


図 13　学生ごとの主観的難易度と解答時間の分布

表 7 学生ごとに求めた主観的難易度と解答時間，得点の相関係数

相関係数	解答時間	得点
−1 〜 −0.5	3	6
−0.5 〜 0	8	10
0 〜 0.5	14	13
0.5 〜 s	5	1

関を学生ごとに求め，検討する．問題ごとに含まれる穴抜きの数が異なるため，解答時間は表 5 に示す穴の数で割ったものを使用する．**表 7** に相関係数の頻度を示す．

解答時間は，主観的難易度が高くなると長くなり，正の相関になると予想できる．表 7 より，半数以上の学生について正の相関がみられるが，負の相関となる学生も 11 名存在する．そこで，相関係数が負の学生について，pgtracer の学習ログ分析機能を使用して確認を行った．その結果，7 名の学生について，短時間でいったん終了し，正当を確認してから直後の学習で高得点をとる学習パターンが見られた．直後の学習においては，通常の解答をする時間よりも短いことから，負の相関が得られたと考察できる．7 名のデータを除外すると，82.6%の学生が正の相関を持つ．

得点は，主観的難易度が高くなると低くなり，正の相関になると予想できる．表 7 より，半数以上の学生について負の相関がみられるが，正の相関となる学生も 14 名存在する．そこで，相関係数が負の学生について，学習ログ分析機能を使用して確認した結果，10 名の学生について，問題の半数以上が 100 点であった．このため，正の相関が得られたと考察できる．10 名のデータを除外すると，61.5%の学生が負の相関を持つ．

以上の考察から，学生の主観的難易度と問題の客観的な難易度の間には必ずしも相関は認められないものの，個別の学生単位で見ると，主観的難易度と得点・解答時間の間には相関が認められることが分かる．

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 の評価実験を行い．アンケートによる主観的難易度や学生の学習ログを分析することによって，pgtracer の問題難易度について考察した．

pgtracer の問題難易度の設定や学生の解答行動において，コメントの影響が高いことが分かった．5.2 節では，コメントによって，[3] で定義した穴の種類による難易度を用いて問題難易度を制御できることが分かった．また，5.3 節より，元となるプログラム自体の難易度が低くても，コメントを記載しないことで学生にとって難易度を高くできる．また，逆にプログラム自体の難易度が高くても，コメントを利用することによって難易度を低くできることから，1 つのプログラムから複数の異なる難易度の問題を作成するのにコメントの利用が有効であることが分かる．6.3 節により，コメント無の問題では，問題解答に必要な知識や問題意図の理解を評価するのに役立ち，コメント有の問題では，そのような知識を持った上でのプログラム作成能力や問題プログラムの理解を評価するのに役立つことが分かった．

5.2 節より，変数の変数の振る舞いが変数名から予測されるものと異なると学生の主観的難易度が高くなることが分かった．これにより，変数名を単なるアルファベットとする，もしくはわざと変数「min」を最大値の保存に用いるなど，変数の振る舞いが予測しにくいものにすることにより穴埋め問題の難易度を高くすることができる．わざと誤解を招くような変数の命名はプログラミング教育の教材としては好ましくないが，変数名を単なるアルファベットにすることで学生にヒントを与えないように工夫することは，プログラミング教育の観点からも妥当と考えられる．

プログラム中のコメントには，プログラム全体の意図，個別ルーチンの機能，変数が保持する情報，アルゴリズムを記述するといった様々な役割がある．今後は，コメントの種類と問題の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．

謝辞 本実験に協力いただきました熊本高等専門学校八代キャンパスの米沢教授，藤本教授，小島教授，学生達に心より感謝いたします．

参考文献

[1] 掛下，大月，嘉藤，村田：穴埋め問題を用いたプログラミング教育支援ツールの全体構想，平成 25 年度電気関係学会九州支部連合大会，(2013).
[2] Moodle.org, https://moodle.org/
[3] 大田，柳田，掛下：学生の解答履歴に基づいた穴埋め問題の難易度の定量的評価，平成 26 年度電気関係学会九州支部連合大会，(2014).
[4] 大田，掛下：穴埋め問題を用いたプログラミング教育ツール pgtracer のログデータ分析機能，情報処理学会 コンピュータと教育研究会，(2015).
[5] 柳田，大田，大月, 掛下：穴埋め問題を用いたプログラミング教育ツール pgtracer の運用実験，情報教育シンポジウム，(2014).