情報処理学会論文誌　教育とコンピュータ　Vol.2　No.2　20–36 (Oct. 2016)

論　文

穴埋め問題を用いたプログラミング教育支援ツール pgtracerの開発と評価

掛下 哲郎[1]　柳田 崚[1]　太田 康介[1]

受付日 2015年12月17日, 採録日 2016年7月9日

概要：プログラミング教育は理工系の大学等の教育機関において重要性が高いが，学生の学力低下に関する懸念や，プログラミング実習時の人的支援体制が十分には確保できない等の課題をかかえている．この問題に対処するために，本研究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 を提案する．本ツールは，プログラムやトレース表に対する穴埋め問題を学生に出題し，学生の答案を自動採点する．pgtracer を用いることで学習時間を短縮し，学習効率を高めることができる．穴抜き個所の指定や，プログラム中のコメントの表示・非表示を切り替えることにより，教員は様々な難易度の穴埋め問題を柔軟に作成できる．また，本ツールは，教員が学生の理解状況や解答過程を把握するために必要な解答履歴や解答時間，正誤等の学習履歴を収集する．教員は，把握した情報をもとに問題を改善し学生に出題する PDCA サイクルを繰り返すことで，効果的なプログラミング教育を実現できる．我々は pgtracer の初期バージョンに対する評価実験を行い学生や教員のコメントを収集した．コメントに基づいて改良した pgtracer に対する評価実験を行ったところ，操作性に不満のある学生は 3 割から 1 割にまで減少し，pgtracer が学習に役立ったとの評価を多くの学生から得た．

キーワード：プログラミング教育，穴埋め問題，教育支援ツール，ラーニング・アナリティクス

Development and Evaluation of Programming Education Support Tool pgtracer Utilizing Fill-in-the-Blank Question

Tetsuro Kakeshita[1]　Ryo Yanagita[1]　Kosuke Ohta[1]

Received: December 17, 2015, Accepted: July 9, 2016

Abstract: Programming education is important at universities majored in science and engineering. However there are obstacles due to declining student's ability and lack of support staff for programming education. In this paper, we propose a programming education support tool pgtracer utilizing fill-in-the-blank questions for a program and a trace table in order to cope with this problem. Student's answers are automatically scored by pgtracer. The learning time is reduced so that learning efficiency is improved. A teacher can create fill-in-the-blank questions with various difficulty levels by selecting various types and places of blanks, and by hiding all or part of the comments within the program. The tool also collects student's learning history. Then a teacher can analyze achievement level and learning process of each student and those of the entire class. The teacher can thus utilize the information to realize PDCA cycle for effective programming education. We conducted an evaluation experiment of the initial version of pgtracer and collected comments of students and teachers. We improve pgtracer considering these comments and conducted another evaluation experiment of pgtracer after the improvement. As a result, unsatisfied students are reduced from 30% to 10%. We also received positive evaluation from many of the students.

Keywords: programming education, fill-in-the-blank question, education support tool, learning analytics

1. はじめに

プログラミング教育は理工系の大学等において重要性が

[1] 佐賀大学
Saga University, Saga 840–8502, Japan

図 1　穴埋め問題の例

Fig. 1　An example of a fill-in-the-blank question.

高いが，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 等だけでは十分な指導が行えない等の課題をかかえている．学生は，授業や教科書をもとに新しい知識を学習するが，復習する際には自身の能力にあった学習をすべきだろう．しかし，学生自らが自身の能力を理解し，それに合わせて学習することは難しい．教員が各学生にあった課題を出題できればよいが，教員にとっても学生ごとの理解状況を詳細に把握するには大きな手間がかかる．

この問題に対処するためのプラットフォームとして，本論文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 を提案する．pgtracer は，教員が穴埋め問題を柔軟に作成・編集する機能，学生の答案を自動採点する機能，学生の学習履歴や解答履歴を収集する機能等を提供する．これらの機能を活用することで，効果的なプログラミング教育支援環境を構築できる．また，運用実験を通じて pgtracer の機能や操作性を評価し，学生および教員から収集したコメントに基づき pgtracer を改良する．

pgtracer の問題形式はプログラムとトレース表を用いた穴埋め問題である（**図 1**）．トレース表はプログラムの実行過程を表す表である．トレース表の各行はプログラム中のルーチン名とステップ番号の組に対応しており，行の並びはプログラム中のステップの実行順序を表す．また，各列はプログラム中の変数（大域変数，局所変数，仮引数）または出力値にそれぞれ対応しており，各ステップで変数等が保持する値を表す．

pgtracer は，プログラムとトレース表が一貫した状態になるように，設定された穴を埋める作業を学生に課する．これにより，一からプログラムを記述させるよりも短時間で学習でき，プログラム中で教育内容に対応した個所を重視した指導が効率良く行える．

問題中の穴抜き個所としては，プログラム中のトークン，トークン列，文，およびトレース表中の変数値，ステップ番号，変数名，ルーチン名等から教員が自由に指定できる．これによって穴埋め問題の難易度を柔軟に制御できる．また，プログラム中には様々なコメントを記述できるが，コメントの一部または全部を隠すことによっても穴埋め問題の難易度を制御できる．

pgtracer はオープンソースの学習管理システム Moodle [13] 上で動作し，学生の答案を自動採点する．そのため，インターネット環境があれば，時間や場所を選ばない学習が可能になる．また，学生が穴埋めを行うたびに，学習データを収集できる．収集したデータを分析することで，個別の学生やクラス全体の理解状況を教員が把握できる．

上述した様々な機能を通じて，pgtracer は学生の学習意欲を高めることができる．

本論文は以下のように構成されている．まず 2 章では関連研究について述べる．3 章では，pgtracer の概要を紹介し，4 章および 5 章で pgtracer の教員用機能と学生用機能について詳しく述べる．6 章では，pgtracer の初期バージョンを用いた第 1 回運用実験の概要と運用実験後のアンケート結果に基づいた操作性評価について述べる．我々は第 1 回運用実験の際に収集したレビューコメントや分析データをもとに pgtracer を改善した．7 章では，具体的な改善内容を示す．8 章では第 2 回の運用実験を実施することで，改良した各機能の操作性および有用性を評価する．最後に 9 章でまとめと今後の課題について説明する．

なお，付録では穴埋め問題を記述している XML ファイルの定義，pgtracer が用いているテーブルの設計および，学生が穴を埋めるごとに実行される自動採点アルゴリズムを具体的に説明する．

2.　関連研究

2.1　トレース表を用いた学習の有用性

トレース表を通じて，学生はプログラム中の文の実行順序や変数値の変遷等のプログラム実行過程を具体的に理解できる．これは，プログラムを理解するうえで不可欠である．また，プログラムの理解は，プログラム作成の前提である．そのため，トレース表を用いた学習はプログラミングを学習する者にとって重要性が高い．

江木らの研究 [1] ではプログラム理解におけるトレース学習の重要性が検討されており，プログラムをトレースする作業が学生のプログラミングの行き詰まり状態の打開につながるとされている．また，プログラムの実行過程をアニメーションにより可視化して教育支援を行う研究 [2] もある．これらにより，プログラミング初学者にとってはプログラムの実行過程を可視化して示すことが重要であり，トレース表を活用した学習はプログラミング教育において有効な学習手段になることが期待される．

2.2　プログラミング教育の様々なアプローチ

プログラミング教育を行う研究については，様々なアプローチから多くの議論がされている．pgtracer は穴埋め問題という出題形式を採用しているが，プログラミング教育を行う手段としてプログラムを一から書かせることに価値を置く研究もある．

西田ら [3] はプログラミング初学者のための学習環境 PEN を開発している．PEN はプログラム理解の促進に重きを置いており，学習者は日本語ベースの手順記述言語を用いて，実際に一からプログラミングをしながらステップ実行機能や変数表示機能による支援を受けて学習できる．これに対して，pgtracer は穴埋め問題によって，ピンポイントで個別の要素を問うことに重きを置いている．そうすることで，難易度の調整も容易にできるほか，学習時間を短縮できるため授業への導入も容易になる．

初学者向けのビジュアルプログラミング教育環境の研究 [4] もあるが，学生の学習過程の分析やプログラミング能力の把握には踏み込んでいない．また，教育用のロボットプログラミング言語を用いているため，実用的なプログラミングを行うためには，余分な学習が必要になる．

pgtracer と同様に穴埋め問題を用いたプログラミング教育システムとしては，Funabiki ら [5] による研究がある．この研究では，Java プログラムの予約語を対象とする穴埋め問題の自動生成を行っているが，pgtracer では，はるかに多様な種類の穴抜きを定義できる．

2.3　学生の理解状況の把握と適切な問題の自動選択

pgtracer と同様に学生の理解状況を把握し，それに応じた問題を出題しようとする研究もいくつか存在する．田口ら [6] は個々の学習者の理解状況や学習意欲に応じて演習課題を出題する研究を行っている．学習者の理解状況は協調フィルタリングを用いることで，過去の他の学習者の演習履歴に基づき，各演習課題の得点を推測する．そして，教員が過去の演習への取り組み姿勢から判定した学習意欲に応じて演習問題を課する．ほかにも，学生の問題ごとの習得度から各問題の理解度を把握する研究 [7] や，学生自身の理解度を過去に解いた全問題の成績データから動的に評価する研究 [8] 等様々なアプローチの研究がある．

これらの研究は，問題の出題を自動的に行うことに重点を置いている．これに対して，pgtracer は学生や教員から見た使いやすさと，様々な学習情報の収集に重点を置いており，プログラミング問題の個別の穴の単位まで細分化した要素について，学生の詳細な理解状況を把握することを目指している．そのため，pgtracer で収集した情報を活用することで，上にあげた関連研究の技術を適用することも可能と考えられる．

2.4　学生の解答過程の分析

Deperlioglu ら [9] は，プログラミングにおけるブレンド型学習を提案している．e-learning を組み合わせることで，学生の学習状況をシステムが把握でき，学生に応じたアドバイスや問題を提供できる．しかし，この研究では通常の学習管理システムが提供している問題ごとの得点や時刻のデータのみを収集しており，pgtracer のように個別の問題における学習過程や穴ごとの分析ができるようなデータ収集は行っていない．

教育用のゲームプログラミングを題材として，学習過程の分析を目指す研究 [10] もあるが，収集しているデータの詳細は明らかにされていない．

同様の研究としては前田らによるプログラム作成過程の分析 [11] や，C 言語のプログラミング過程の分析 [12] がある．前田らの研究では，学生のプログラミング過程を演習中のキー入力状況から分析している．これにより，個別の学生がプログラムを作っていく過程を把握することはできるが，クラス全体の理解度を把握するのは難しい．一方，pgtracer では，穴ごとの学習データを記録し，それを穴ごとに集計することで，クラス全体の理解度を把握できる．また，学生がトレース表を埋めていく過程も把握できる．

3.　プログラミング教育支援ツール pgtracer

3.1　pgtracer の構成

pgtracer は授業の一環，低学力の学生を対象とした補習，学生の自習等で活用することを主目的とし，C++（構造化プログラミング）および Java（オブジェクト指向プログラミング）の初学者向けの学習支援ツールとして企画している．本論文執筆時点では，C++プログラミングに対応した部分が完成している．

本ツールは，学習管理システム Moodle のプラグインモジュールとして開発している．学習管理システムに組み込むことで，他の教育コンテンツとも統合したワンストップサービスを学生や教員に提供できる．また，24 時間 365 日の運用も可能である．Moodle では各種データを MySQL で管理しており，MySQL のレコードの追加，削除，更新，取得を容易に行う関数も用意されている．本ツールでも，問題の情報や学習履歴等を MySQL テーブルに保持する．これらのデータは，pgtracer の穴埋め問題を定義するために必要な情報を漏れなく記述できるように設計されている．テーブル間の関連および詳細な設計については付録 A.2 に記述している．

pgtracer では，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の 4 つのファイルを用いて穴埋め問題を定義する．プログラムやトレース表の本体と穴抜きのマスク情報を分離して記述することで，プログラムやマスク等の再利用や加工を容易に行うことができる．これら 4 つのファイルは独立した XML ファイルとして記述

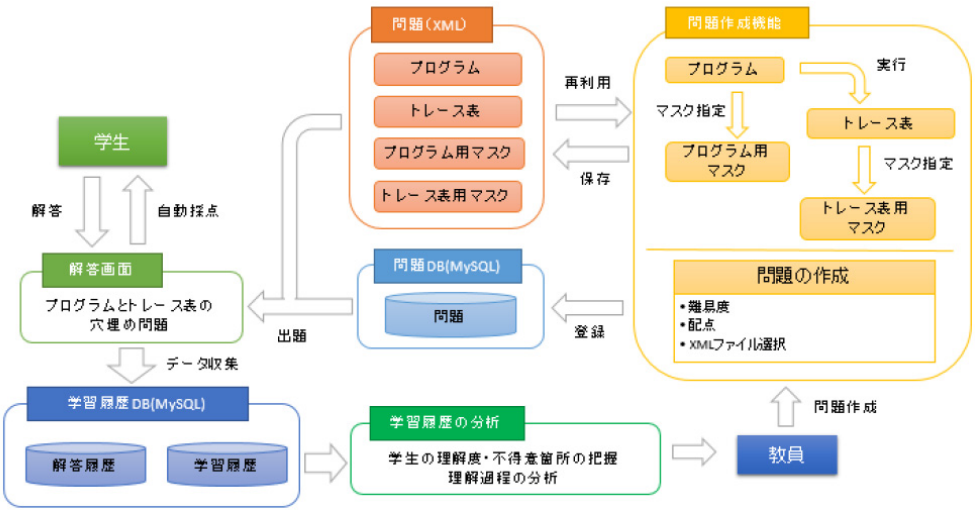


図 2　プログラミング教育支援ツール pgtracer を用いたプログラミング教育の概念図
Fig. 2　Concept of programming education utilizing programming education support tool pgtracer.

するが，各 XML ファイルは，ID を通じて相互に参照している．各ファイルの構造を定義する DTD の詳細を付録 A.1 に示す．

3.2　pgtracer を用いたプログラミング教育

図 2 は本ツールを用いたプログラミング教育の概念図である．

教員はまず，問題作成機能を用いて穴埋め問題を作成する．作成された XML ファイルや問題の情報は Moodle のディレクトリ内や MySQL テーブル（付録 A.2）に保持される．学生には MySQL テーブルの情報をもとに問題が表示される．

学生が問題に解答するとツールは自動採点を行い，結果を学生に返す．答案の採点結果は学習履歴として，解答過程は解答履歴として MySQL テーブルに格納される．

教員は学習履歴や解答履歴から個々の学生やクラス全体の理解度・不得意箇所および学生のプログラム解答過程を分析できる．教員は，分析結果を考慮して新たな問題を作成し，学習データを収集する PDCA サイクルを繰り返すことで，出題する問題を学生にとって適切な難易度の問題へと改善し，学生の不得意箇所に対する重点的な教育を実現できる．

適切な難易度の穴埋め問題を作成する際に重要な点は，pgtracer が様々な難易度の多様な穴埋め問題を記述できるか，pgtracer が収集する学習履歴等から学生やクラス全体の理解状況を詳細に把握できるかの 2 点である．これらの観点については，以下で詳しく述べる．

3.3　pgtracer による多様な穴埋め問題の記述

穴埋め問題の難易度は，問題を構成するプログラムの難易度に影響される．このほかに，pgtracer は，穴抜きの種類やプログラムに付加するコメントの有無，トレース表の一部行の非表示によって，同じプログラムとトレース表の組に対しても難易度の異なる多様な問題を定義できる．教員は，学生の正解率や学習時間を確認しつつ難易度を調整することにより，問題の難易度を適切なレベルに設定できる．

3.3.1　穴抜きの種類

pgtracer が提供する穴抜きの種類としては，以下のものがあげられる．これらの穴抜きを組み合わせることで，多様な難易度の穴埋め問題を定義できる．

- プログラムに対する穴抜き
  - 単一トークン
  - トークン列
  - 文全体
  - 文の並び
- トレース表に対する穴抜き
  - 変数名
  - 変数値
  - 出力値
  - ステップ番号
  - ルーチン名

pgtracer ではプログラムとプログラム用マスクを組み合わせることで，プログラムに対する穴抜きを定義する．プログラムは付録 A.1.1 に示す DTD に合致する XML ファイルで記述される．これにより，プログラムはトークン列にまで分解される．また，プログラム用マスクは，付録 A.1.3 に示す DTD に合致する XML ファイルで定義される．プログラムに対する穴抜きは，プログラムの XML 文書に対する XPath 式で定義するため，任意のトークン列に対して穴抜きを定義できる．

一方，トレース表とトレース表用マスクを組み合わせる

ことで，トレース表に対する穴抜きを定義する．トレース表は付録 A.1.2 に示す DTD に合致する XML ファイルで記述される．これにより，トレース表は表を構成するセルの並びにまで分解される．また，トレース表用マスクは，付録 A.1.4 に示す DTD に合致する XML ファイルで記述される．トレース表に対する穴抜きも，トレース表の XML 文書に対する XPath 式で定義するため，任意のセルに対して穴抜きを定義できる．

穴抜きの種類により穴埋め問題の難易度は変化する [14]．また，トレース表の穴抜きにおいて，変数値の穴抜きは，当該ステップにおける変数値の変更がある場合とない場合とで穴埋め問題の難易度が変化する．ステップ番号の穴抜きについても，直前の行との連続番号になる場合と不連続になる場合（条件分岐，ループ，ルーチン呼び出し等）とで難易度が変化する．

pgtracer では，プログラムとトレース表の双方に自由に穴抜きを行って問題を定義できるため，トレース表の穴埋めによるプログラム理解と，プログラムの穴埋めによるプログラム記述能力の双方について問う穴埋め問題を定義できる．これによって，以下に列挙するように穴を設定した要素に応じて，様々なレベルの学生の理解度を把握できる．

- トレース表の変数値に穴抜きすることで，変数値の変化を正しく理解していることを確認する．
- トレース表のステップ番号や変数名を穴抜きすることで，文の実行順序や対応する変数を正しく理解していることを確認する．
- プログラムの変数名，演算子，予約語等の個別トークンに穴を設定することで，初歩のプログラミング能力を確認する．
- 式，文，複合文，ルーチン等のトークン列に穴抜きすることで，より高度なプログラミング能力を確認する．

たとえばプログラム全体を明示した状態で，トレース表の変数値とステップ番号のすべてを穴抜きすれば，与えられたプログラムの動作を学生が正しく理解していることを確認できる．逆に，トレース表の全体を明示した状態で，プログラム中のすべての文を個別に穴抜きすれば，一からプログラミングを行う能力を評価できる．このように，1 つのプログラムに対しても穴抜きの種類の異なる様々な問題を作成できることが，プログラムとトレース表，2 種類のマスクファイルを用いる利点である．

3.3.2　プログラムのコメントの非表示

プログラム中には様々な種類のコメントが記述されている．大別すると以下の 4 種類があげられる．

- プログラム全体に対するコメントや問題を説明するためのコメント
- 個別のルーチンの機能を説明するためのコメント
- 変数が保持する値を説明するためのコメント
- アルゴリズムの各ステップを説明するためのコメント

これらのコメントの全部または一部を非表示とすることにより，問題の難易度を制御できる．実際に，pgtracer の問題難易度の設定や学生の解答行動において，コメントの有無の影響が大きいことが分かっている [15]．元となるプログラム自体の難易度が高い場合でもコメントを表示すれば難易度を下げることができ，逆に易しい問題の場合でもコメントを表示しないことで難易度を高くできる．

プログラム中の非表示部分は，付録 A.1.3 に示すプログラム用マスクのうち，hidden 要素を用いて指定する．hidden 要素は，プログラムの XML 文書に対する任意の XPath 式を指定できるため，任意のトークン列を非表示指定できる．そのため，穴埋め問題を学生に提示する際には，コメント以外の部分も非表示指定できる．この機能は，プログラム中の未履修部分を学生から隠す場合等に用いることもできる．

3.3.3　トレース表の一部の非表示

トレース表の一部の行を非表示とすることで，繰り返して実行される個所を単純化して表示できる．これにより，学生はプログラム実行の規則性を把握するのが難しくなる．また，トレース表の列の一部を非表示にすることで，主要な変数値のみに着目したトレース表を作成できる．トレース表の列のうち，変数に対応する列同士を入れ替えることで，学生が変数名を推測するのを難しくすることもできる．これらの設定を通じて，穴埋め問題の難易度を制御できる [15]．

トレース表中の表示部分は，付録 A.1.4 に示すトレース表用マスクを用いて，行および列の単位で指定する．逆にいえば，トレース表用マスク中で指定されなかった行または列は，学生には表示されない．

3.4　穴埋め問題を用いた学生やクラス全体の理解度把握

pgtracer は学生の学習履歴や解答履歴を収集する．これらのデータを分析することで，学生やクラス全体の理解状況を把握するための情報を様々な詳細度で求められる [16]．

たとえば，学生が穴埋め問題に解答する過程を再現することで，穴埋めの順序や，個別の穴を埋めるためにかかった時間，一度埋めた穴を修正する過程等を把握できる．成績の良い学生と，つまずいている学生の解答過程を比較することで，指導のためのヒントを発見することもできる．また，個別の穴埋め問題について，穴ごとに学生の答案や正誤，所要時間の分布等を表示することで，クラス全体の理解状況や学生が犯しやすい誤りを，教員が容易に把握できる．穴埋め問題ごとに平均点や平均所要時間を計算することで，クラス全体の理解度の概要も把握できる．

4.　教員用機能

本ツールは教員用機能として問題編集機能，テーマ・問題の登録・管理機能，学習履歴および解答履歴の収集機能

を持つ．以下で各機能について述べる．これらの機能により，教員が穴埋め問題の編集を容易に行えるように支援する．

4.1　問題編集機能

本ツールで出題するプログラミング問題は XML 形式で表現される．XML ファイルの編集および DTD との整合性検査は XMLEDITOR.NET 等でも行えるが，手作業で XML ファイルを作成するのは，一般的な教員には難しい．そこで，pgtracer は問題を構成するプログラムおよびトレース表を，教員が作成したプログラムと入力データに基づいて自動生成する機能を提供する．また，プログラム用マスクとトレース表用マスクを編集する機能を提供する．これによって，作成の手間を省くだけでなく，人手による編集に起因する誤りの混入を防ぐこともできる．

問題編集機能で作成した XML ファイルは，Moodle モジュールの管理用フォルダ内のプログラミング言語，ユーザ，種類（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）ごとに分類されたフォルダに保存される．

4.1.1　プログラムの XML ファイルの自動生成機能

本ツールはアップロードされたプログラムのファイルをコンパイルし，文法的に正しいことを確認したうえで，プログラムの XML ファイルを自動生成する．変換対象は，以下で列挙する言語機能（構文およびライブラリ関数）のみを含む C/C++プログラムである．

- 変数（大域変数，局所変数），定数
- データ型（整数，実数，文字，文字列，論理型，ポインタ）
- データ構造（配列，構造体）
- 式（四則演算，論理演算，比較，代入）
- 文字列の操作（strcat，strcmp，strlen，strcpy）
- 入出力文（cin，cout，printf，scanf）
- ループ文（while，for，do-while）
- ルーチン定義，仮引数，戻り値，ルーチン呼び出し（再帰含む）
- ファイル操作（ofstream，ifstream）

変換された XML ファイルとプログラムのプレビュー画面を**図 3** に示す．自動生成された XML ファイルには，教員が任意の名前を付けてシステムに登録できる．

4.1.2　トレース表の自動生成機能

教員がプログラムの XML ファイルを指定し，プログラムに対する入力データを与えると，pgtracer はトレース表の XML ファイルを自動生成する．本機能では，前項で述べた機能で pgtracer が自動生成した XML ファイルのみを対象としている．トレース表は，プログラムに対する入力データによって変数の値や結果が変化する．そのため，本機能では自動生成前に，実行結果を確認できる機能を提供

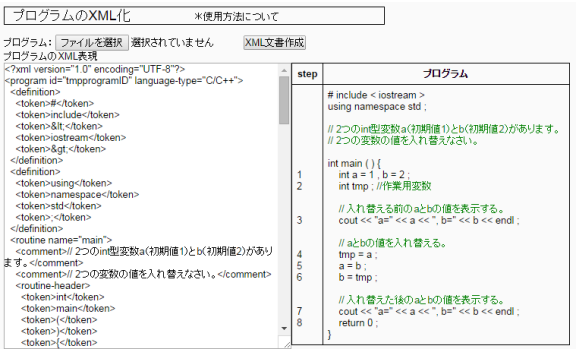


図 3　プログラムの XML 自動生成結果

Fig. 3　Automatic generation of XML file representing a program.

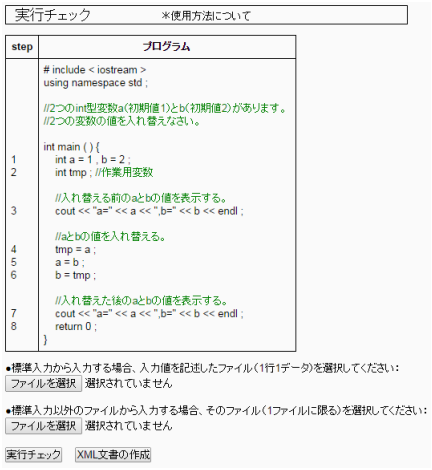


図 4　実行結果確認画面

Fig. 4　Execution result confirmation of a program.

図 5　実行結果の例

Fig. 5　An example of a program execution result.

する（**図 4**，**図 5**）．

教員がプログラムの XML ファイルを指定すると，pgtracer は，それを C++プログラムに戻し，トレース表に含まれるルーチン名，ステップ番号，変数名，各ステップにおける変数値，プログラムの出力値を出力するための文を C++プログラムに挿入する．入力ファイルには，入力値がテキストファイル 1 行につき 1 データの形式で記述されている．出力文が挿入された C++プログラムは pgtracer によってコンパイル・実行され，入力データを読みながらトレース表を生成する．これによって，教員が望む入力値でトレース表を自動生成できる．XML ファイルの生成に成功すると，プログラムと同様に XML ファイルおよびトレース表のプレビュー画面を表示する（**図 6**）．

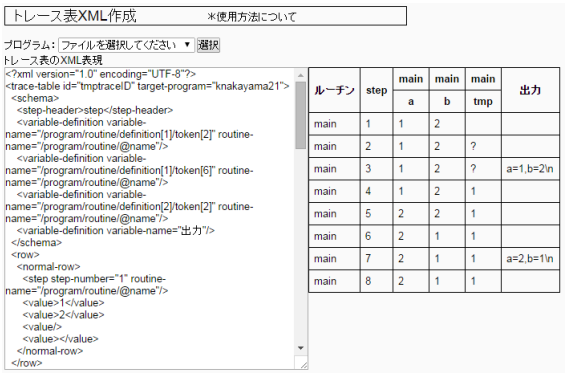


図 6　トレース表の XML 自動生成結果

Fig. 6　Automatic generation of XML file representing a trace table.

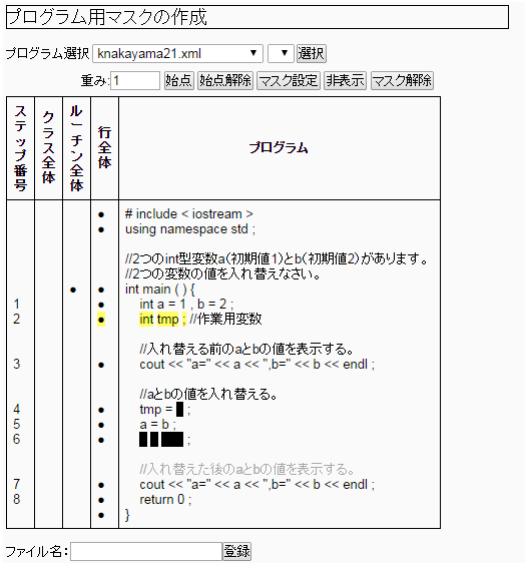


図 7　プログラム用マスク編集画面

Fig. 7　Edit function for program mask.

4.1.3　プログラム用マスクの編集機能

プログラム用マスクで表現するのは，プログラムの非表示箇所および穴抜きを設定する箇所と配点の重みである．これを教員が効率良く設定できるように，本ツールでは教員が指定したプログラムを画面上に表示し，非表示個所や穴抜きを設定したい箇所を指定することでプログラム用マスクの XML ファイルを生成・編集する機能を提供する（**図 7**）．

本機能で対象となるプログラムは，pgtracer が自動生成した XML ファイルのみである．本機能は，3.3 節で述べた様々な穴抜きや非表示が柔軟に定義できるように設計している．

図 7 に表示しているプログラムは，マウスでクリックすることで，単一トークン単位での選択が可能となっている．また，穴抜き箇所を設定する際には連続したトークンに対して穴抜きを行うことも想定される．そのため，トークンの始点と終点を指定することで，連続した複数トークン（トークン列）を選択できる．選択されたトークンは背景が黄色で表される．この状態で，マスク設定ボタンを選択することで選択部分が穴抜き個所として設定される．また，トークンの選択を簡易化するために，プログラム左列に 3 つの項目（行全体，ルーチン全体，クラス全体）を設置する．プログラム横の黒丸を選択することで，該当するトークン列を一括して選択できる．たとえば，行全体に該当する黒丸をクリックすることで，3.3.1 項でいう ‘文全体’ の穴抜きが可能である．

図 8　トレース表用マスク編集画面

Fig. 8　Edit function for trace table mask.

マスク設定ボタンでなく非表示ボタンを押した場合には，選択部分は非表示部分として設定される．非表示指定は，3.3.2 項で説明したように，コメント部分に適用することを想定した機能である．非表示指定された部分は，穴埋め問題には表示されない．

4.1.4　トレース表用マスクの編集機能

トレース表用マスクが表現するのは，トレース表で表示する行・列および穴抜きとする部分と配点の重みである．これを教員が効率良く設定できるように，本ツールでは教員が指定したトレース表を画面上に表示し，マスクしたい箇所を指定することでトレース表用マスクの XML ファイルを生成・編集する機能を提供する（**図 8**）．本機能も 3.3 節で示した様々な穴抜き等を柔軟に定義できるように設計した．

本機能はプログラム用マスクと同様に，トークンを選択することで穴抜き箇所を設定する．始点および終点となるトークンを指定することにより，それらによって囲まれた矩形内のトークンすべてにマスクを設定する機能も実装した．これにより，マスクを設定する手間が減少する．プログラム用マスクの編集機能と異なるのは，トークンの非表示がない点である．その代わりに，トレース表は各行に付けられたチェックボックスによって各行の表示・非表示を指定する．非表示指定された行は，プログラミング問題の中では「中略」と表示される．また，変数に対応する列の入れ替え機能も実装した．

図 9　テーマ一覧画面
Fig. 9　Management function of themes.

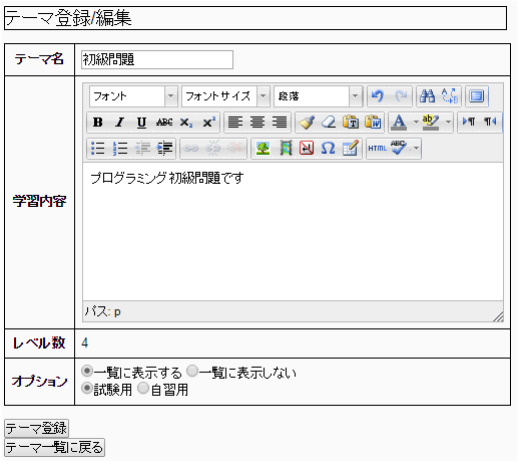


図 10　テーマ登録/編集画面
Fig. 10　Registration and editing functions of a theme.

4.2　テーマの登録・管理機能

教員は複数のテーマ（穴埋め問題のグループ）を定義することができ，1 つのテーマには複数の問題を登録できる．テーマは「テーマ一覧画面」によって新規登録および編集/削除/コピーができる（図 9）．

教員は「テーマ登録/編集画面」でテーマを作成・編集できる（図 10）．ここでは，テーマ名とレベル数の設定，自習/試験用の選択，問題一覧への表示/非表示を選択できる．テーマを登録すると，テーマ一覧からテーマへの問題登録が可能になる．ここで，「試験用」を選択すると，当該テーマの問題は試験モードに設定され，学生が穴埋めを済ませて「解答終了」ボタンを押した時点で採点が行われる．一方，「自習用」を選択すると，当該テーマの問題は自習モードに設定され，個別の穴を埋めるごとに採点が行われる．

4.3　問題の登録・管理機能

各テーマに登録されている各問題は，テーマ一覧の問題一覧ボタンを押すことで表示される（図 11）．問題一覧画面では，問題の新規登録のほか，各問題の編集/削除/コピーが行える．各問題は，同一テーマだけでなく，他テーマへコピーすることもできる．また，自動生成によって作成した XML ファイルをコピー，リネーム，削除する機能も提供している．

1 つの問題は，問題タイトル，レベル，配点，各 XML

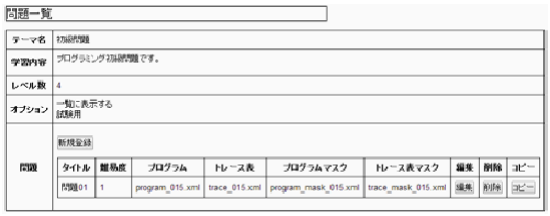


図 11　問題一覧画面
Fig. 11　Management function of questions of a theme.

図 12　問題登録/編集画面
Fig. 12　Registration and editing functions of a question.

ファイルの情報を持つ．これらの情報は，「問題登録/編集画面」から指定し登録できる（図 12）．プログラムの XML ファイルを選択すると，それに対応するプログラム用マスクやトレース表の XML ファイルのみが選択可能になる．また，トレース表を選択すると，それに対応するトレース表用マスクのみが選択可能になる．これにより，誤った問題の定義を防ぐ．なお，レベルの値は 1 以上，当該問題のテーマで設定された値以下の範囲で，教員が任意に指定できる．pgtracer の穴埋め問題では，プログラムとトレース表の組合せが同一でも，マスクの設定により難易度が変化するため，その点を考慮してレベルを決定する．

問題を構成する各 XML ファイルは，作成したユーザごとに分けられたディレクトリに登録されている．そのため，問題登録の際に XML ファイルを選択する際には，ファイル作成者のユーザ名を指定したうえで，そのユーザが作成した XML ファイルを指定することになる．また，編集した問題が解答画面でどのように表示されるか確認できるようにプレビュー機能も提供する．

4.4　学習履歴および解答履歴の収集機能

本ツールでは，学生が穴を埋める際や解答を終える際に，学習履歴および解答履歴を収集できる．学生が穴を埋めた

際には，対象の穴を示す XPath 式，学生の入力文字列，正解文字列，正誤，入力終了時刻が解答履歴テーブルに保存される．また，学生が問題の解答を終えた際には，ユーザ ID，問題 ID，得点，解答開始時間，解答終了時間が学習履歴テーブルに保存される．こうして収集した履歴データを活用することで 3.4 節で述べた分析が可能になる．

5. 学生用機能

pgtracer は，学生用機能として問題の選択機能および自動採点機能を提供している．以下で各機能について述べる．

5.1 問題の選択機能

問題の登録・編集機能で作成された問題は，問題の選択画面（図 13）でテーマ，タイトル，難易度（レベル）ごとに分類して一覧表示されており，学生は自分の学力に合った，もしくは教員に指示された問題を選択する．それぞれの問題は学生が一度も受験したことのない問題ならば「未受験」と表示され，受験したことのある問題ならば「得点/満点」の形式で表示される．これらの文字列は解答画面へのリンクとなっており，解答したい問題のリンクをクリックすれば受験できる．

5.2 自動採点機能

pgtracer では，問題は自習モードと試験モードに分類されている．自習モードは C++プログラムの文法や動作の確認のための機能として，試験モードは学生の C++プログラミング能力やプログラム理解度の確認・評価を行うために開発した．したがって，本ツールの自動採点は，学生が穴埋めを済ませて解答を終了した時点だけでなく，自習モードにおいては個別の穴を埋めた直後にも実行される．

自習モードの場合は，穴埋めを行った直後に正誤を判定し，穴の色で正誤を表現する．正解は黄緑で，不正解は赤で表す．不正解の穴にマウスカーソルを合わせると，吹き出しを用いて正解を表示する（図 14）．穴埋め直後の自動採点では，模範解答との完全一致により判定を行う．これは，自習中には模範的な解答を示すことで，合理的なプログラミング技術を身につけてもらうためである．

自習・試験モードともに，学生がひととおり解答を終え解答終了ボタンを押すと，採点結果画面（図 15）に遷移する．採点結果画面では，最終的な解答画面の状態をもとに採点を行う．合計点を計算する際には正解した穴の重みの合計を求め，問題全体の重みの合計で割り，問題の配点に乗じて求める．重みは，その問題における穴の重要度であり，マスク作成時に設定される．学生が不正解の穴にカーソルを合わせると，吹き出しを用いて正解を表示するのは，図 14 の場合と同様である．

解答終了時点での自動採点においては，まず模範解答と答案の文字列比較を穴ごとに行い，完全に一致した場合に

問題一覧

テーマ	タイトル	難易度			
		1	2	3	4
四則演算	足し算・引き算	未受験	100/100	100/100	90/100
	掛け算・割り算	未受験	100/100	100/100	未登録
奇数・偶数	奇数偶数の判断	未受験	未登録	80/100	未登録
if文	問題が未登録です。				
for文	問題が未登録です。				

図 13　問題の選択画面
Fig. 13　Student view of question list.

```
   //input_numが奇数の場合、「奇数」と表示する。
3  if ( input_num % 2 ==  1 ) {
3.1   cout << " 奇数 " << endl ;
   }
                    正解:0
   //上記以外でinput_numが3の倍数の場合、「偶数かつ3の倍数」と表示する。
4  else if ( input_num % 3 == 1 ) {
4.1   cout << " 偶数かつ3の倍数 " << endl ;
   }
```

図 14　自動採点および正しい答えの表示
Fig. 14　Automatic scoring function with the right answer.

採点結果画面

step	プログラム
	# include < iostream >
	using namespace std ;
	//2つのint型変数a(初期値1)とb(初期値2)があります。
	//2つの変数の値を入れ替えなさい。
	int main ( ) {
1	int a = 1 , b = 2 ;
2	int tmp ;
3	cout << " a = " << a << " , b = " << b << endl ;
4	tmp = a;
5	a = b ;
6	b = a;
7	cout << " a = " << a << " , b = " << b << endl ;
8	return 0 ;
	}

ルーチン	step	main a	main b	main tmp	出力
main	1	1	2		
main	2	1	2	?	
main	3	1	2	?	a = 1 , b = 2
main	4	1	2	1	
main	5	2	2	1	
main	6	2	1	1	
main	7	2	1	1	a = 2 , b = 1
main	8	2	1	1	

得点 : 90/100
問題一覧に戻る

図 15　採点結果画面
Fig. 15　Result of automatic scoring.

は正解とする．しかし，pgtracer の穴埋め問題は自由記述式なので，プログラム中の穴には複数の正解が存在する場合がある．そのため，プログラムの穴埋めを採点する際には，模範解答との文字列比較が失敗した場合，学生の答案を用いて出題されたプログラムを書き換え，それをコンパイル・実行したうえでトレース表と実行結果を比較し，両者が一致した場合には正解と判定する．

したがって，解答終了時点の自動採点機能は，穴に入力された答案がどのような場合でも，実行結果が一致していれば正解と判定する．たとえば「n を 1 増やす」という処理に穴抜きがあった場合，「n=n+1」，「n=1+n」，「n++」，「n+=1」等の様々な正解のバリエーションに対応する．

プログラムの実行結果とトレース表を比較するために，pgtracer は，ステップ番号や変数値等を表示するための出力文をプログラムに自動挿入している．

6. pgtracer の第 1 回運用実験

本章では，協力していただいた教員から収集したコメントや，学生に対して行ったアンケート調査の結果に基づいて，初期版 pgtracer の操作性を評価する．

6.1 運用実験の概要

pgtracer の各機能の操作性を評価するための運用実験では，本学・知能情報システム学科で実施しているプログラミング教育（1 年次後期～2 年次前期開講．座学・演習各 2 科目）のうち，2 年次科目を履修中の学生 62 名を対象とし，自習モードと試験モードで合計 84 問（21 タイトル × 4 レベル）の C++プログラミング問題を出題し，2 週間の期限を設けて取り組んでもらった．運用実験に取り組んだ学生を対象にアンケート調査を行うとともに，コメントを収集した．また，pgtracer の履歴収集機能を用いて学習履歴を収集した．

84 の穴埋め問題は，プログラムとトレース表の組を 21 種類用意し，それぞれに対して，穴抜き個所や非表示個所が異なる 4 つの問題を定義した．また，プログラムとトレース表が共通の 4 つの問題に対しては，穴抜き個所等による難易度の差を考慮して 1～4 の 4 段階のレベルをそれぞれ設定した．

運用実験で用いた問題は，本学および熊本高専の教員数名に問題生成機能を用いて作成していただいた．その際に，各教員からレビューコメントを収集した．

6.2 学習状況の分析

学生の利用時間は 1～2 時間のケースが最も多く，1 つの問題に対して 4～5 分程度の時間を要することが分かった（**図 16**）．このように pgtracer ではプログラムを一から記述する学習方法と比較して短時間で学習できる．

穴埋め問題を用いることで，授業で教えた内容に即して，ピンポイントで学生に穴埋めさせる問題を作成できる．これにより学習効率を高めることができる．

pgtracer の学習履歴を確認したところ，最終的に満点を獲得した答案（延べ数）が 90%を超えていた（**図 17**）．これは，pgtracer を用いることで学生の学習意欲が引き出されたことを示す傍証でもある．

6.3 pgtracer の操作性の評価

今回の運用実験では，問題作成機能について，おおむね良い評価が教員から得られた．一方でいくつかの指摘・要望もいただいた．以下にいただいたコメントを紹介する．

(1) プログラムマスクやトレース表マスクの作成画面で，既存の XML ファイルを編集できるようにしてほしい．
(2) トレース表用マスク XML の作成で，複数のマスを同時にマスクしたい．

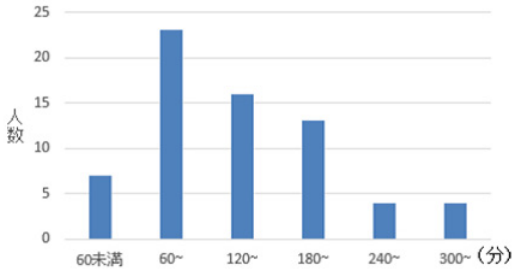


図 16　学習時間の分布
Fig. 16　Learning time distribution.

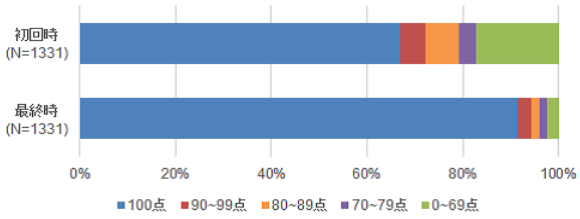


図 17　学生の得点分布
Fig. 17　Distribution of student scores.

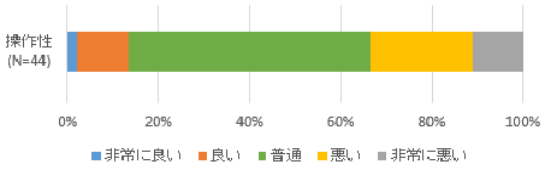


図 18　解答時の操作性評価
Fig. 18　Usability evaluation of the answering process.

(3) トレース表の出力値で，水平タブや改行を表したい．
(4) プログラム中で，文字配列を初期化した際には，トレース表では終端文字を表示してほしい．

学生向け機能の操作性については，入力の容易さや分かりやすさ，応答時間等をふまえて 5 段階評価でアンケート調査した．アンケート回答率は 71.0%である．

集計結果を見ると，6 割強の学生は pgtracer の操作性はおおむね問題ないと回答している（**図 18**）．一方でおよそ 3 割の学生が pgtracer の操作性に不満を感じていることが分かった．学生のコメントを見ると，操作性に関する多くの要望が寄せられている．それらのうち主要な意見を以下に示す．

(1) Enter キーを押すと採点が行われるため，押し間違えたとき面倒．
(2) プログラム中の半角スペースの有無が分かりにくい．
(3) トレース表にたくさん穴が空いている場合，直前の行と同じ値を簡単に入れられるようにしてほしい．
(4) 自習モードについて，マウスを重ねることで，ただちに正解が表示されるようになっているので，表示するかしないかを学習者が選択できるようにしてみてはどうか．
(5) 自習モードにおける自動採点について，模範解答とは

一致しないが意味的には正しい場合に不正解となることがある．

7. レビューに基づいた pgtracer の改善

本章では，6.3 節で収集したコメントをふまえた pgtracer の改善内容を示す．

7.1 教員用機能の改善

穴埋め問題を作成した教員からいただいたコメントをもとに機能の修正および改善を行った．

マスクの編集機能および複数マスの同時マスク機能については，コメントをいただいた際に修正を行っている．現在の解答画面では，Tab を押すと次の穴へ移動し，また Enter を押すと Web ブラウザに対する submit 命令が優先され次の画面に遷移してしまうため，水平タブや改行等を入力できない．そこで，水平タブや改行については文字列 '\t', '\n' をトレース表の XML ファイルに記述し，解答入力時には，'\t' や '\n' を学生に入力させることで正誤判定を行うこととした．

プログラム中の文字配列については，初期化された際に，これまではトレース表の変数値には空白で表されていた．しかし，文字配列を学習する際には，終端文字等の知識を学ぶ必要があるため，'\0' を表示するよう修正した．

7.2 半角スペースの代替文字による表示

プログラム中の出力文字に半角スペースが含まれているか分かりにくく，誤りと判定される答案が多く見られた．そこで，文字列内の半角スペースについては代替文字として空白記号␣を用いて表現するように修正した．

7.3 トレース表への入力補助機能

トレース表の変数値の列には同じ値が続くことが多いが，トレース表の変数値の列に穴が連続して空いていた場合，同じ値を何度も入力する手間がかかる．学生からも「入力に無駄な手間がかかる」との指摘があり，学生の意欲低下も懸念される．そこで以下の入力補助機能を実装した．

- 学生がトレース表の変数値に対応する穴に入力を行った場合，入力した穴以降のすべての穴に，pgtracer がその値を自動入力する．
- ただし，学生自身が以前に値を入力した穴およびそれ以降の穴については，自動入力の対象外とする．

図 19 は実際に入力補助機能を用いて学生が赤枠部分に答案を入力した際の例である．学生が赤枠部分に入力すると，pgtracer は青枠部分を自動入力する．図 19 右では，以前に学生が値を入力した穴の直前までを書き換えることにより，学生自身による入力を優先している．この機能を用いると，トレース表の変数値については，値が変化した穴のみに答案を入力すれば穴埋めが完成する．

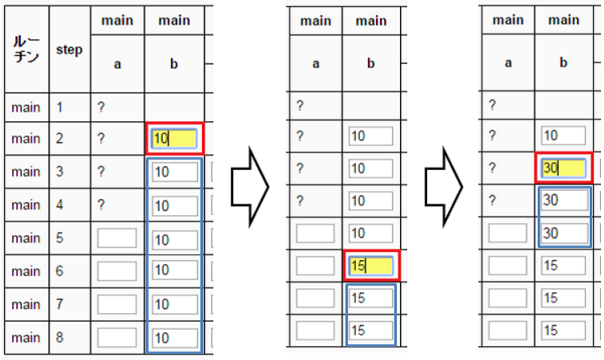


図 19　入力補助を用いた解答例
Fig. 19　The assist function to fill trace table.

図 20　自習モード時の解答画面
Fig. 20　The self study mode for a fill-in-the-blank question.

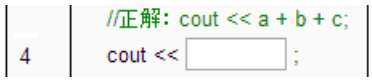


図 21　1 つのマスの問題例
Fig. 21　A step with single blank.

図 22　複数マスの問題例
Fig. 22　A step with multiple blanks.

7.4 自習モードの改良

自習モードにおいて，正解の表示・非表示を学生が選択できる仕組みを追加し，学生がより自習しやすい環境を実装した（**図 20**）．この機能を実装することで自習モード時の学生の解答過程にも影響が出ると予想される．そこで，学生が正答の表示・非表示を変更した場合，それを解答履歴テーブルに記録する．

7.5 自習モードにおける自動採点機能の改良

図 21 のように「a + b + c」を 1 つの穴として穴抜きする場合には，学生の解答がどのような場合でも実行結果が一致すれば正解となる．

しかし，**図 22** のように，それぞれが 1 つのトークンに対応する複数の穴を用いてマスクを設定する場合，学生の解答が①a，②c，③b ならば，①は模範解答と完全一致するため正解となるが②，③は不正解となる．

このようなケースはほかにも起こることが考えられるため，今回，総当たり法を用いて採点手順の改良を行った．変更した採点アルゴリズムの詳細は付録 A.3 に示す．

これにより，解答の順序が入れ替わったとしても答案を正しく採点できる．なお，総当たり法を用いることで，計算量の爆発が起きる可能性もあるが，現実的な穴埋め問題では，1 文中の穴抜きの個数は数個程度にとどまるため，実用上の問題は少ないと考えられる．

8. pgtracer の第 2 回運用実験

本節では，第 1 回運用実験で収集したコメントやアンケート結果に基づいて改良した各機能の操作性，および改良版 pgtracer の有用性を評価する．

8.1 運用実験の概要

pgtracer の改良後，再度各機能の操作性および有用性を評価するために運用実験を行った．本学科 1 年生 57 名（全員がプログラミング教育を履修中）を対象とし，3 週間の期限を設けて自習・試験モードの計 44 問を出題した．また，運用実験終了後には学生に対してアンケート調査を行った．ただし，アンケートについては，学生 57 名の中で指示した計 44 問のすべてに解答した学生 37 名のアンケート結果に基づいて考察する．

8.2 pgtracer の操作性の評価

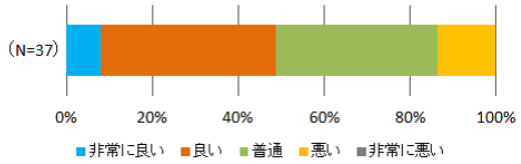
pgtracer の操作性については，第 1 回運用実験と同様に 5 段階評価で意見を収集した（**図 23** (1)）．集計結果を見ると，‘非常に悪い’ と回答した学生はいなかった．また，‘悪い’ と回答した学生，いわゆる操作性に不満を感じている学生は 37 名中 5 名であり，第 1 回運用実験では 3 割程度であったのに対して 1 割程度まで減少した．この結果から，pgtracer の操作性は第 1 回と比較して向上したと考えられる．

また，7 章で改良した pgtracer の各機能が学生にとって役に立ったかどうかを評価した（図 23 (2)，(3)）．集計結果を見ると，トレース表の入力補助機能については 6 割強，自習モードの正答を切り替える機能は 5 割強の学生が役に立ったと回答した．

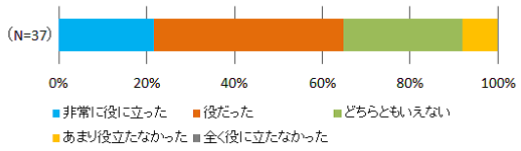
自動採点機能の改良については，第 1 回運用実験の際，採点結果に不満のあるコメントが 4 件あったが，第 2 回運用実験ではそのようなコメントは寄せられなかった．

第 2 回運用実験において，学生の穴抜きへの解答は 46,332 個あり，その中で完全一致せず自動実行による採点で不正解となった解答は 1,789 個である．その 1,789 個の解答について，目視により採点結果を確認したところ，正解になるべき解答はなかった．このことから，改良後の自動採点機能では，正しく採点が行われていることが確認できた．

(1) 解答時の操作性評価



(2) トレース表の入力補助機能の評価



(3) 自習モードで正答の表示・非表示を切り替える機能の評価

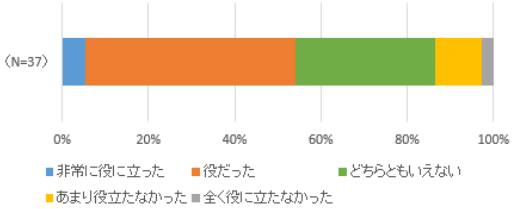


図 23　pgtracer の操作性評価
Fig. 23　Usability evaluation of pgtracer.

8.3 pgtracer の有用性の評価

pgtracer の有用性は記述可能な穴埋め問題の範囲，教員や学生の評価等を通じて総合的に評価する必要がある．記述可能な問題の範囲については，3.3 節および付録 A.1 で議論し，多様な穴埋め問題を記述できることを示した．また，7.1 節の改善を通じて，pgtracer を試用した教員からも良い評価を得た．

さらに，学生にとって，pgtracer が復習や自身の能力評価の役に立つシステムであるか評価するため，我々は pgtracer の試験モードおよび自習モードについて**図 24** のアンケート調査を行った．

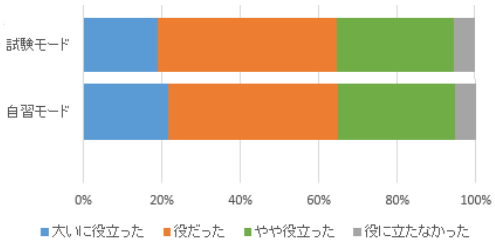
集計結果を見ると，文法や動作の確認・復習については，6 割強の学生から肯定的な回答を得ることができた．自習モードおよび試験モードともに大きな違いは見られない．

自分の能力の確認・評価については，自習モードではおよそ 6 割，試験モードではおよそ 5 割の学生から肯定的な回答を得ることができた．また，試験モードよりも自習モードの方が自身の能力の確認・評価において有用であることが確認できる．これは，自習モードにおいて正答の表示切替えが可能となったため，切り替えながら学習することで，試験モードよりも容易に採点結果を確認できるためだと考えられる．

この 2 つのアンケート結果では，‘役に立たなかった’ と回答した学生は 1 割に満たず少数だった．以下に，‘非常に役立った’ または ‘役立った’ と回答した学生から寄せられたコメントの一部を示す．

- 入力直後に採点してくれる自習モードの仕組みは非常

(1) pgtracer は C++プログラムの文法や動作の確認・復習に役立ったか



(2) pgtracer は自分の能力の確認・評価に役立ったか

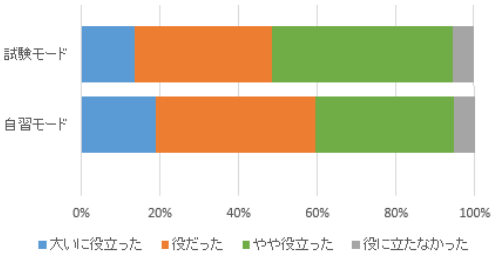


図 24　pgtracer の有用性評価

Fig. 24　Usefulness evaluation of pgtracer.

に勉強に役立つが，問題が簡単すぎたので改善してほしい（非常に役立った）．

- 動作が速くて，問題の難易度も良かったので活用していきたい（役立った）．
- プログラミング練習としていいシステムだと思う（役立った）．

一方で，‘やや役立った’と回答した学生の割合はかなり多い．しかし，‘やや役立った’と回答した学生からも，以下のような肯定的なコメントが得られた．

- 実際に冬季休業中に pgtracer を利用して同じような問題を何回も繰り返すことで，とてもいい反復練習になった．
- pgtracer は，分かりやすく使いやすいものでプログラミングの学習に生かせるものだと感じた．
- pgtracer はプログラミングが苦手な学生にはとてもありがたい．
- その場で間違いが分かるため，すぐに問題の解決法を導くことができ，とてもいいなと思った．

なお，‘やや役立った’と回答した学生から，「\n や; を忘れて解答をし直すことが何度もあった」との不満が 1 件寄せられたが，pgtracer の有用性に関するコメントとはいえない．

以上を総合的に判断すると，評価実験を通じて，学生からの不満はほとんどなく，肯定的な評価が十分多いことが分かる．

9.　おわりに

本論文では，プログラミング教育支援ツール pgtracer を提案し，本ツールがプログラミング教育に役立つ様々な機能を提供していることを示した．また，運用実験を通じて操作性等を評価し改善を図った．

pgtracer は，学習管理システム Moodle 上で動作し，自動採点機能（5.2 節および付録 A.3）を提供することで，時間や場所を選ばない学習を可能にする．穴埋め問題を用いることで学習効率を高めることもできる（6 章）．また，5 章で提案した学生用機能を，レビュー結果に基づいて改良（7 章）することで，操作性や有用性についても学生から良い評価を得た（8 章）．このように，穴埋め問題を用いた学習の効率化および pgtracer の機能充実や操作性向上等を通じてプログラミング学習に対する学生の学習意欲を高めることができた．

pgtracer は，教員に対して多様な穴埋め問題（3.3 節）の作成を支援するための柔軟な編集機能を提供している（4 章および 7.1 節）．さらに，pgtracer を用いることで，学生の詳細な学習履歴等を収集でき，それを分析することで教育上有用な情報を得ることもできる（3.4 節）．

本ツールを用いることで，学生のプログラム理解度やプログラミング能力を定量的に評価し，問題の作成，問題の出題，学習履歴等の分析，問題の改善からなるプログラミング教育の PDCA サイクルを実現できる．

今後の研究課題としては，収集した学習履歴等を用いた学生やクラス全体の具体的な理解度や解答過程の分析，学生に対する分析結果のフィードバック，教育目標に応じた合理的な穴埋め問題の設計技術，および学生の理解度に応じた問題の自動出題機能に関する研究等があげられる．

謝辞　運用実験にご協力いただいた熊本高等専門学校および佐賀大学の教員および学生の皆さんに感謝します．本研究の一部は JSPS 科研費（課題番号 16K01022）の助成を受けています．

参考文献

[1] 江木鶴子，竹内　章：プログラミング初心者にトレースを指導するデバッグ支援システムの開発と評価，日本教育工学会論文誌，32(イ), pp.369–381 (2009).
[2] 黄　寧，丸山桃代，宮寺庸造，横山節雄：プログラム可視化によるプログラミング教育支援，電子情報通信学会技術研究報告，ET，教育工学，Vol.99, No.31, pp.1–6 (1999).
[3] 西田知博，原田　章，中村亮太，宮本友介，松浦敏雄：初学者用プログラミング学習環境 PEN の実装と評価，情報処理学会論文誌，Vol.48, No.8, pp.2736–2747 (2007).
[4] Hahul, R., Whitchurch, A. and Rao, M.: An open source graphical robot programming environment in introductory programming curriculum for undergraduates, *Proc. IEEE International Conference on MOOC, Innocation and Technology in Education* (*MITE*), pp.96–100 (2014).
[5] Funabiki, N., Korenaga, T., Nakanishi, T. and Watanabe, K.: An extension of fill-in-the-blank problem function in Java programming learning assistant system, *2013 IEEE Region 10 Humanitarian Technology Conference, R10-HTC 2013*, pp.85–90, (Aug. 2013).
[6] 田口　浩，糸賀裕也，毛利公一，山本哲男，島川博光：個々

の学習者の理解状況と学習意欲に合わせたプログラミング教育支援，情報処理学会論文誌，Vol.48, No.2, pp.958–968 (2007).
[7] 中島秀樹，高橋直久，細川宜秀：プログラミング学習のための QA サイクル：受講者の習得度に応じた問題自動提示メカニズム，電子情報通信学会論文誌，D-I，情報・システム，I-情報処理，Vol.J88-D-I, No.2, pp.439–450 (2005).
[8] 菅沼　明，峯　恒憲，正代隆義：学生の理解度と問題の難易度を動的に評価する練習問題自動生成システム AEGIS，情報処理学会研究報告，DD，[デジタル・ドキュメント]，Vol.2003, No.11, pp.25–32 (2003).
[9] Deperliogle, O. and Kose, U.: The Effectiveness and Experiences of Blended Learning Approaches to Computer Programming Education, *Computer Applications in Engineering Education*, Vol.21, Issue 2, pp.328–342 (2013).
[10] Malliarakis, C., Satratzemi, M. and Xinogalos, S.: Integrating learning analytics in an educational MMORPG for computer programming, *Proc. IEEE 14th International Conference on Advanced Learning Technologies*, pp.233–237 (2014).
[11] 前田恵三，中野靖夫：プログラム作成過程の分析，日本教育工学雑誌，Vol.19, No.3, pp.171–180 (1995).
[12] 前田恵三，中野靖夫：C 言語のプログラミング過程の分析，電子情報通信学会技術研究報告，ET，教育工学，Vol.94, No.260, pp.37–44 (1994).
[13] Moodle.org, available from ⟨https://moodle.org/⟩
[14] 柳田　崚，太田康介，大月美佳，掛下哲郎：穴埋め問題を用いたプログラミング教育支援ツール pgtracer の運用実験，情報処理学会情報教育シンポジウム SSS2014 (2014).
[15] 村田美友紀，掛下哲郎：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，情報処理学会コンピュータと教育研究会 (Mar. 2015).
[16] Kakeshita, T. and Ohta, K.: Student log analysis functions for web-based programming education support tool pgtracer, *17th International Conference on Information Integration and Web-based Applications & Services* (*iiWAS2015*), pp.120–128 (2015).

付　　録

A.1 問題の構成と XML ファイルの設計

A.1.1 プログラム本体

プログラムの XML ファイルは穴埋め問題の基盤となる部分である．1 つのプログラムに対して，穴埋め問題は複数存在しうる．また，問題の再利用性を高めるために，固定の言語ではなく複数の言語にも対応できるように一般化した形で記述されるのが望ましい．ただし，プログラミング言語の文法はきわめて多様なため，どのような言語に対応できるかは言語タイプ（language-type）として指定する．このような要求に基づいて設計したのが，プログラム本体記述用 XML 形式の DTD である．

```
<!DOCTYPE program [
  <!ELEMENT program (
     comment?, z80-statement*, definition*,
     class*, routine*, compound-statement*,
     correspondence*)>
    <!ATTLIST program
       id CDATA #REQUIRED
    language-type CDATA #REQUIRED>

  <!ELEMENT comment (#PCDATA)>
    <!ATTLIST comment id CDATA #IMPLIED>

  <!ELEMENT z80-statement
     address, machine-code, label,
     mnemonic-code, comment)>
    <!ATTLIST z80-statement
       id CDATA #IMPLIED>
  <!ELEMENT address (#PCDATA)>
  <!ELEMENT machine-code (token*)>
  <!ELEMENT label (#PCDATA)>
  <!ELEMENT mnemonic-code (token*)>

  <!ELEMENT definition (token*, comment?)>
    <!ATTLIST definition
       id CDATA #IMPLIED>

  <!ELEMENT class (
     comment?, class-header, definition*,
     routine*, class-footer)>
    <!ATTLIST class
       id CDATA #IMPLIED
       name CDATA #REQUIRED>
  <!ELEMENT class-header (token*)>
  <!ELEMENT class-footer (token*)>

  <!ELEMENT token (#PCDATA)>
    <!ATTLIST token id CDATA #IMPLIED>

  <!ELEMENT routine (
     comment?, routine-header, definition*,
     compound-statement*, routine-footer)>
    <!ATTLIST routine
       id CDATA #IMPLIED
       name CDATA #REQUIRED>
  <!ELEMENT routine-header (token*)>
  <!ELEMENT routine-footer (token*)>

  <!ELEMENT compound-statement (
     comment?,
     (simple-statement |
      compound-statement))>
    <!ATTLIST compound-statement
       id CDATA #IMPLIED>

  <!ELEMENT simple-statement (token*)>
    <!ATTLIST simple-statement
       id CDATA #IMPLIED>

  <!ELEMENT correspondence EMPTY>
    <!ATTLIST correspondence
       id CDATA #IMPLIED
       class-name CDATA #IMPLIED
       routine-name CDATA #REQUIRED
       step-number CDATA #REQUIRED
       target-path CDATA #REQUIRED>
]>
```

現時点の実装においては，本学科で教育されている Z80 アセンブリ言語，C/C++（構造化言語の機能のみ），Java を対象言語としている．Z80 と他の 2 つの間には文法的に大きな違いがあるので共通部分がほとんどないが，C/C++

と Java の間には共通部分があるため，定義部（definition）や関数（routine）等の要素が共有されている．

このようにある程度の言語的な構造を表現したうえで，トレース表においては行やステップ単位での指定を行うため，行単位での記述ができるようにした．この方針で最も影響を受けたのがクラス（class）やルーチン（関数/メソッド，routine）の定義部分で，開始と終了を行として明示するためにヘッダとフッタの要素を導入した．さらに，これらの行は最終的には穴埋めの単位であるトークン（token）にまで分解される．

また，プログラム本体の情報とは別に，各ステップとプログラムにおけるステートメントの対応情報（correspondence）を定義する．この対応情報は，トレース表の中で参照される．

A.1.2　トレース表

トレース表はプログラムに入力を与えることで決まる．基本形式は縦の要素をステップ，横の要素を変数とする表である．このため，HTML のテーブル記述を参考に行と列を記述する．

この記述において，最も複雑な部分は列の定義部分（schema）である．特に変数の定義部（variable-definition）では，クラスやルーチン等のネームスペースにより同名の変数が存在する可能性があるため，重複がおこらないようにそれらを区別する必要がある．そのため，それを区別するための属性をオプションとして指定できる．行におけるステップ記述でもクラスやルーチンを指定できるよう，同様の属性を定義した．

```
<!DOCTYPE trace-table [
  <!ELEMENT trace-table (schema, row*)>
    <!ATTLIST trace-table
       id CDATA #REQUIRED
       target-program CDATA #REQUIRED>

  <!ELEMENT schema (
     step-number-header,
     variable-definition*)>
  <!ELEMENT step-number-header (#PCDATA)>
  <!ELEMENT variable-definition (
     data-structure-name*)>
    <!ATTLIST variable-definition
       id CDATA #IMPLIED
       variable-name CDATA #REQUIRED
       routine-name CDATA #IMPLIED
       instance-name CDATA #IMPLIED
       class-name CDATA #IMPLIED
       storage-class #IMPLIED>

  <!ELEMENT data-structure-name (#PCDATA)>

  <!ELEMENT row (abbreviation-row | normal-row)>
  <!ELEMENT abbreviation-row EMPTY>
  <!ELEMENT normal-row (step, value*)>
  <!ELEMENT step EMPTY>
    <!ATTLIST step
       id CDATA #IMPLIED
       class-name CDATA #IMPLIED
       instance-name CDATA #IMPLIED
       routine-name CDATA #REQUIRED
       step-number CDATA #REQUIRED>
  <!ELEMENT value (#PCDATA)>
]>
```

A.1.3　プログラム用マスク

pgtracer では，1 つのプログラムに対して複数のプログラム用マスクを定義できるようにしている．マスク個所を変更することにより，問題の難易度を調節できる．

プログラムに対するマスクは，1 つ以上のトークンを伏せてそれを解答させるものである．このため，トークンを 1 つ以上伏せることが可能なようにする必要がある．そこで，トークンの指定には XPath を利用し，XML のノードをその木構造のパス表記で指定する．これによって，マスク対象としてトークン，トークンの列，行，行の列を柔軟に指定することが可能となる．また，hidden 要素を導入することで表示する場合に不要な部分を省略できるようにしている．たとえば，プログラム中のコメントが解答のヒントになることを防ぐために，コメントを hidden 要素として指定することが考えられる．

穴埋め問題（question）としては，属性 target-path にトークンとして XPath を 1 つ以上記述できるようにし，さらにその穴埋め 1 つごとに配点の重み（weight）を指定できる．この配点の重みは，1 以上の整数値であり既定値を 1 とする．学生の能力を正しく評価するためには，穴の重みを，難易度に比例するように設定することが重要だと考えられる．穴の難易度を評価する際には，解答履歴の分析を通じて得られる解答所要時間や正解率が参考になる．最終的に，問題中の穴埋めすべてについて集計した値で割った値が全体の配点に対して乗じられる．

```
<!DOCTYPE mask-for-program [
  <!ELEMENT mask-for-program (
     hidden*, question*)>
    <!ATTLIST mask-for-program
       id CDATA #REQUIRED
       target-program CDATA #REQUIRED>

  <!ELEMENT hidden EMPTY>
    <!ATTLIST hidden
       target-path CDATA #REQUIRED>

  <!ELEMENT question EMPTY>
    <!ATTLIST question
      target-path CDATA #REQUIRED
      weight CDATA "1">
]>
```

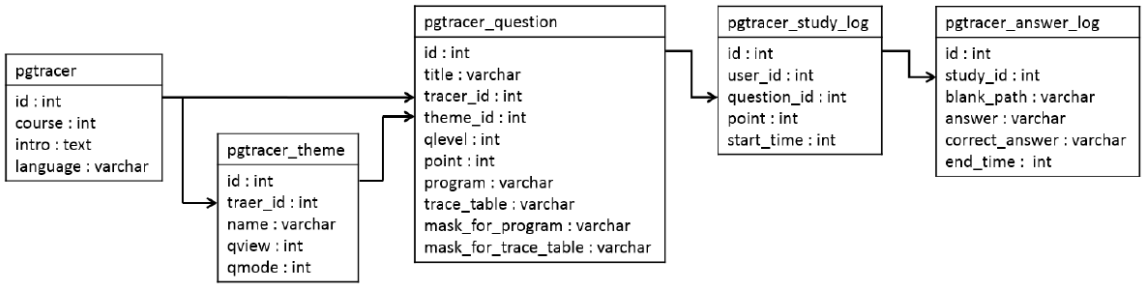


図 A·1　テーブル間関連図
Fig. A·1　Relationship among tables.

A.1.4　トレース表用マスク

トレース表の穴埋め問題もプログラム穴埋め問題と基本的に同様であり，該当する穴抜きを XPath 式で指定し，そこに配点の重みが指定できるようにしている．マスク対象としては，変数名，ステップ番号，当該ステップのルーチン名，各ステップの変数値および出力値を指定できる．

```
<!DOCTYPE mask-for-trace-table
  <!ELEMENT mask-for-trace-table (
     schema, row*, question*)>
    <!ATTLIST mask-for-trace-table
       id CDATA #REQUIRED
       target-trace-table CDATA #REQUIRED>

  <!ELEMENT schema (
     step-number-header, variable-definition*)>
  <!ELEMENT step-number-header (#PCDATA)>
  <!ELEMENT variable-definition EMPTY>
    <!ATTLIST variable-definition
       target-path CDATA #REQUIRED>

  <!ELEMENT row EMPTY>
    <!ATTLIST row target-path CDATA #REQUIRED>

  <!ELEMENT question EMPTY>
    <!ATTLIST question
       target-path CDATA #REQUIRED
       weight CDATA "1">
]>
```

また，プログラム用マスクの hidden 要素と同様，表示したい行の XPath 式を指定することで，一部のステップを中略とすることもできるほか，表示したい列の XPath 式の順番を入れ替えて指定することで，列の非表示および順番を変更できる．これによって，大きなトレース表の一部のみを学生に表示することや，同一パターン部分を学生から隠すことにより正解の推測を難しくすることができる．

A.2　テーブル設計

図 A·1 は pgtracer が用いる MySQL テーブルの名称，フィールド名，データ型およびテーブル間の関連を表している．pgtracer はモジュールの情報を保持しておくテーブルで，どのコースに登録されているかや，どの言語の学習を行うかの情報を保持している．Moodle の開発ガイドラインに従い，すべてのテーブルの主キーは id としている．

表 A·1　問題テーブル（pgtracer_question）
Table A·1　Question table (pgtracer_question).

フィールド名	説明
id（主キー）	問題 ID
title	問題のタイトル
tracer_id	pgtracer の ID
theme_id	pgtracer_theme の ID
qlevel	難易度
point	配点
program	プログラムのファイル名
trace_table	トレース表のファイル名
mask_for_program	プログラム用マスクのファイル名
mask_for_trace_table	トレース表用マスクのファイル名

pgtracer_theme は pgtracer に登録されたテーマのレコードを保持するテーブルである．テーマは，学習内容や作成者等のように教員が自由に設定することができ，学生に提示する問題か否かの設定や，試験モードなのか自習モードなのかの設定を行う．

pgtracer_question はツールで使用する問題の定義情報を保持しているテーブルである（**表 A·1**）．問題のタイトルやどのモジュールのどのテーマに登録されている問題なのかを保持する．また，問題作成ページで設定された問題の難易度，配点，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ファイル名を保持する．問題を表示する際にはこのテーブルから該当する問題のレコードを取得し，対応する XML ファイルを読み込む．

pgtacer_study_log は学習履歴を保持するテーブル（**表 A·2**），pgtracer_ans_log は解答履歴を保持するテーブル（**表 A·3**）である．これら 2 つのテーブルは学生の学習データを保持するために使用する．学習履歴を保持するテーブルでは，1 回の学習が 1 レコードに対応し，解答開始の段階でレコードが生成される．解答開始の段階ではユーザ ID，問題 ID，学習開始時間のみレコードに格納され，解答を終了した時点で採点結果と学習終了時間が追記される．解答履歴を保持するテーブルは，学習者が穴を 1

表 A·2　学習履歴テーブル（pgtracer_study_log）
Table A·2　Study log table (pgtracer_study_log).

フィールド名	説明
id（主キー）	学習履歴 ID
user_id	ユーザ ID
question_id	問題 ID
point	採点結果
start_time	学習開始時刻

表 A·3　解答履歴テーブル（pgtracer_answer_log）
Table A·3　Answer log table (pgtracer_answer_log).

フィールド名	説明
id（主キー）	解答履歴 ID
study_id	学習履歴 ID
blank_path	穴の XPath 式
answer	答案
correct_answer	正解の文字列
end_time	解答終了時刻

回埋めるたびに 1 つのレコードが追加される．学生が穴を埋めるたびに入力された文字列の正誤判定を行い，穴の位置情報や学生が入力した文字列，正解，入力した時間等を保持する．

A.3　自習モードにおける自動採点のアルゴリズム

プログラムの同一ステップ内に複数のマスが設定されている場合，7.5 節で解説したように，複数の答案の組合せが正解になる場合がある．自習モードでは，個別のマスに答案が入力されるたびに自動採点を行うが，解答画面におけるプログラムのステップごとに総当たり法を用いて自動採点を行うことで，複数マスの解答順序によらず正解判定を行うことができる．以下に，この場合の自動採点アルゴリズムの詳細を示す．

1. 当該ステップに含まれる各マスの完全一致による判定を行い，一致するものは正解，空白のものは不正解とする．
2. 1. で該当しないマスがある場合，当該ステップのすべてのマスについて学生の解答で書き換えたプログラムを作成する．
3. プログラムをコンパイル・実行し，実行結果が正解のトレース表と一致する場合には 2. で該当しなかったマスをすべて正解とする．
4. コンパイルエラーもしくは実行結果が正解のトレース表と一致しなかった場合，以下の処理を行う．
   4.1. 2. で該当しなかったマスが 1 つならば当該マスを不正解とする．
   4.2. 複数あるならば，すべて不一致マスとする．
5. 各不一致マスについて，解答が模範解答に含まれていなければ不正解とする．
6. 不一致マスが存在するならば以下の処理を行う．
   6.1. 模範解答から順列を求め，不一致マスが学生の解答と一致している候補をすべて取得する．
   6.2. 取得した各候補について，学生の解答と一致部分が多い候補から順に以下の処理を行う．
      6.2.1.当該候補でプログラムを書き換えて，コンパイルし実行する．
      6.2.2.もし実行結果が模範解答の実行結果と一致するならば，当該候補中の学生の解答に該当したマスを正解とし，残りを不正解とする．
   6.3. どの候補も実行結果と一致しなければすべてのマスを不正解とする．

掛下 哲郎　（正会員）

1989 年九州大学大学院工学研究科情報工学専攻博士後期課程修了．工学博士．現在，佐賀大学工学系研究科知能情報システム学専攻准教授．ソフトウェア工学，データベース，情報専門教育に関する研究に従事．2012 年本会優秀教育賞受賞．電子情報通信学会，IEEE，ACM 等各会員．

柳田 峻

2015 年佐賀大学大学院工学系研究科知能情報システム学専攻博士前期課程修了．現在，日本データスキル株式会社勤務．大学院時代，プログラミング教育支援ツール pgtracer の研究開発に従事．

太田 康介

2016 年佐賀大学大学院工学系研究科知能情報システム学専攻博士前期課程修了．現在，(株) EWM ファクトリー勤務．大学院時代，プログラミング教育支援ツール pgtracer の研究開発に従事．