Vol.2014-CE-124 No.6
2014/3/14

穴埋め問題を用いたプログラミング教育支援ツール pgtracer における教員用機能の実装

柳田崚[†1]　太田康介[†1]　大月美佳[†1]　掛下哲郎[†1]

我々は，プログラミングに必要な概念を系統的に学習する教育支援ツールを開発している．本ツールは，Moodle 上で動作しプログラムとトレース表の組に対して，いくつかの個所をマスクした穴埋め問題を学習者に出題する．本ツールで出題す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の 4 つから構成しており，各ファイルは XML で表現する．XML で定義された問題を教員が容易に作成・編集できるように，本ツールは教員用機能としてプログラム及びトレース表の XML ファイルを自動生成する機能，プログラムやトレース表用のマスクを生成・編集する機能，問題登録機能を提供している．また，学生の成績データや学習ログを取集・分析するために，成績及び学生履歴閲覧機能，学習ログ管理機能を持つ．本稿では穴埋め問題を構成する各 XML 文書の設計および教員用機能の実装について述べる．

A Programming Education Support Tool pgtracer utilizing Fill-in-the-Blank Questions: Teacher Function

RYO YANAGITA[†1]　KOSUKE OHTA[†1]
MIKA OHTSUKI[†1]　TETSURO KAKESHITA[†1]

We are developing a programming education support tool pgtracer utilizing fill-in-the-blank questions.pgtracer operates on Moodle, and make questions that some parts are masked in a pair of program and trace table. A question consists of a program, a trace table and two masks for program and trace table. The four files are expressed using XML.pgtracer is equipped with question generation function, question registration function, question editing and management functions as functions for teacher. pgtracer also provides student's records and history inspection function, study log management function, in order to collect and analyze student's records and study log. We describe design of each XML document which constitutes fill-in-the-blank questions and implementation of pgtracer function for a teacher in this paper.

1. はじめに

我々は，プログラミングに必要な概念を系統的に学習する教育支援ツール pgtracer を開発している[1]. 本ツールは，Moodle 上で動作しプログラムとトレース表の組に対して，いくつかの個所をマスクした穴埋め問題を学習者に出題する. 本ツールは, 学部 1~2 年次で学習する変数, データ型, 制御構造などのプログラミングの基本的な概念を主な教育対象としている．テーマごとに学習できる問題を作成することで，プログラミングに必要な概念を利用者に系統的に学習させることができる．これらの問題はプログラム，トレース表，プログラム用マスク，トレース表用マスクの 4 つから構成され，XML ファイルで表現する．

本ツールは，学生用機能と教員用機能を提供している．学生用機能は，主に問題を選択，解答し採点を受ける機能である[2]．一方，教員用機能では，学生に対して出題する問題の作成機能や採点結果を管理する機能を提供する．また，学生の解答過程のログデータを収集することで，個々の学生やクラス全体の理解度や不得意個所を特定し教育に役立てることが期待できるほか，ログデータから，学生が解答した順序や，それぞれの穴を解くのに要した時間を求めることができる. そこで, 本ツールは教員用機能として，問題の XML 文書を自動生成する問題生成機能，問題登録機能，問題編集・管理機能を持つ．また，学生の成績データや学習ログを取集・分析するために，成績及び学生履歴閲覧機能，学習ログ管理機能を持つ．

本稿では，2 節で出題する穴埋め問題がどのように構成されているか述べ，問題を構成する各 XML ファイルの構造を述べる．3 節では，本ツールの教員用機能となる各機能について述べる. 4 節では, 3 節で述べた教員用機能から，今回実装した機能について具体的に述べる．そして，5 節で，実装した機能に関する試用者からのコメントと検討を述べる．最後に 6 節でまとめと今度の課題とする．

2. 穴埋め問題の構成と XML 設計

本ツールで出題する問題は，プログラムとトレース表に対する穴埋め問題である．本ツールは，直接的にプログラムとトレース表に穴を空けて出題するのではなく，プログラムとトレース表および，穴埋め箇所を定義したプログラム用マスク及びトレース表用マスクで問題を構成して出題する．このように，プログラム本体と穴埋めのマスク情報のように分離して記述することで，再利用や加工を容易に行うことができる．

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 つの構造と相互関係を記述するために，これら 4 つは独立した XML ファイルとして記述する．各

†1 佐賀大学
Saga University

XML ファイルは，ID を持たせることで，それを参照し相互関係を把握できる．これらの要求を満たすため，プログラム問題を記述するためのXML形式のDTDを定義した[3].

2.1 プログラム本体

プログラム本体(program)は問題の基盤となる部分である．ひと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極めて多様なため，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，図 1 のプログラム本体記述用 XML 形式の DTD である．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上で，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為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，トレース表における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．

2.2 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，クラスやルーチンなど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（図 2）.

2.3 プログラム用マスク

pgtracer では，一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

```
<!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>
]>
```

図 1 プログラム記述用 DTD

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．例えば，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．

```
<!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)>
]>
```

図 2 トレース表記述用 DTD

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（図 3）．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．最終的に，問題中の穴埋め全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かけられる．

2.4 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は同じであり，該当するマスを XPath で指定し，そこに配点の重みが指定できるようにする．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及び出力値を指定することができる. また，

```
<!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 3 プログラム用マスク記述用 DTD

プログラム用マスクの hidden 要素と同様，表示したい行の XPath を指定することで，一部のステップを中略することもできるほか，表示したい列の XPath の順番を入れ替えて指定することで，列の非表示及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（図 4）．

```
<!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">
]>
```

図 4 トレース表用マスク記述用 DTD

3. 教員用機能

本ツールにおいて, 教員はプログラミング問題を作成し，学生に出題する必要がある．また，プログラミング教育を支援する上では，学生の学習状況や理解度を的確に把握することが重要である．そこで，本ツールは，教員用機能として, 問題作成機能, テーマ・問題登録/編集機能, テーマ・問題管理機能，成績及び学生履歴閲覧機能，学習ログ管理機能を持つ．以下で各機能について述べる．

3.1 問題生成機能

本ツールで出題するプログラミング問題は XML 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負担が大きい．そこで，本ツールはプログラミング問題を容易に作成できるように，問題を構成するプログラム及びトレース表を自動生成する機能を実装する．また，プログラム用マスクとトレース表用マスクを生成・編集する機能を実装する．これによって，作成の手間を省くほかにも，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．

問題生成機能で作成した XML ファイルは，モジュールフォルダ下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）毎に分類されたフォルダに保存される．

3.1.1 プログラムの XML ファイルの自動生成機能

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，プログラムの XML ファイルを自動生成する．自動生成に対応するプログラムは，本ツールが定めている C++/C 言語における開発範囲に該当する概念が対象となる．

- 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)

画面上でプログラムファイルを選択し作成ボタンを押すと，ファイルがシステムにアップロードされる．この際に，アップロードされたファイルが cpp ファイルであるか確認する．アップロードしたファイルのコンパイルが成功した場合には，XML 文書及びプレビュー画面を表示する（図 5）．コンパイルに失敗した場合にはエラー文を表示する．

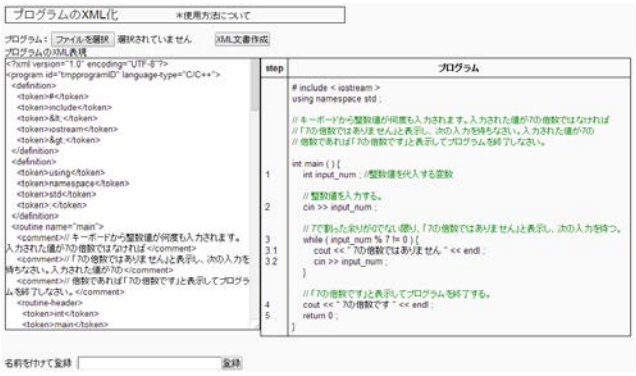

図 5 プログラムの XML 自動生成結果

3.1.2 トレース表の自動生成機能

プログラムの XML ファイルから，トレース表の XML ファイルを自動生成する．本機能では，前節で述べたプログラムの XML 自動生成で作成した XML ファイルのみを対象としている．トレース表は，プログラムの標準入力やそれ以外の入力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する．図 6 では，プログラムに対しての標準入力および標準入力以外のファイルからの入力を事前に指定することで，プログラムを実行した結果を得ることができる（図 7）

標準入力からの入力を指定する際には，入力値が記述されたテキストファイルを選択する．入力値はテキストファイル 1 行につき 1 データ記述する．これによって，教員が望む入力値でトレース表を生成できる．XML 文書の生成に成功すると，プログラムと同様に XML 文書及びプレビュー画面を表示する（図 8）.

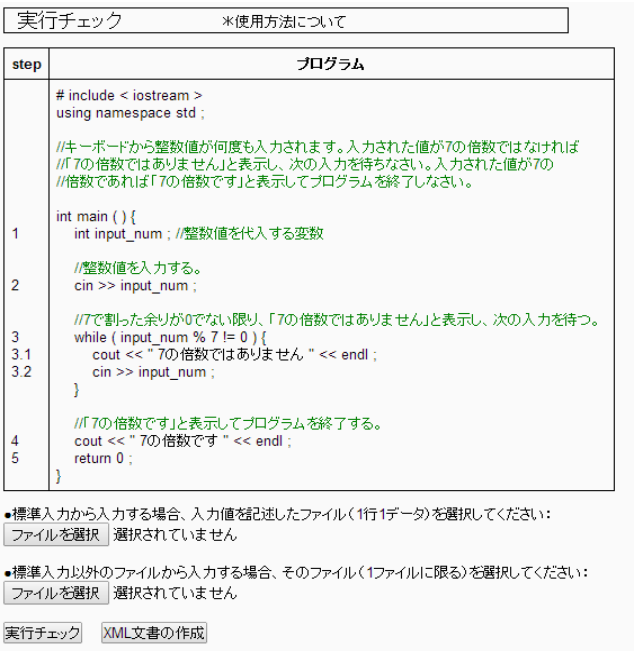

図 6 実行結果確認画面

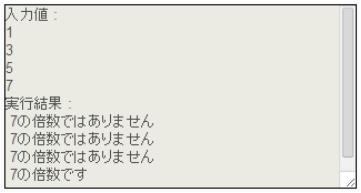

図 7 実行結果の例

図 8 トレース表の XML 自動生成結果

3.1.3 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部分及び穴埋めとする部分と配点の重みである．これを教員が効率よ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，マスク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 9）.

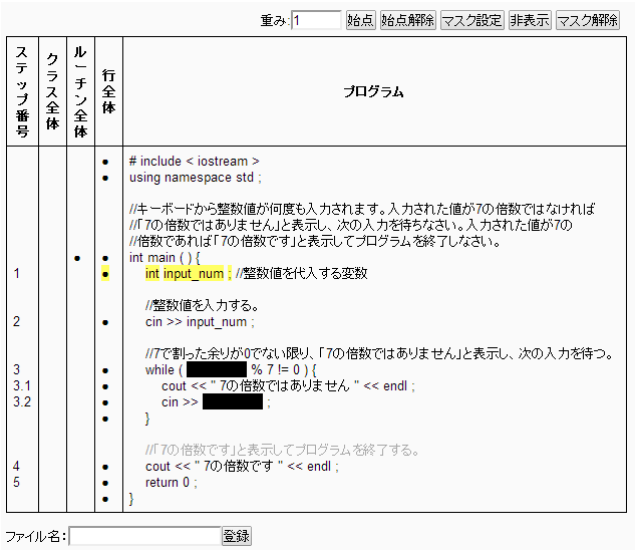

図 9 プログラム用マスク編集画面

本機能で対象となるプログラムは，3.1.1 節で述べたプログラムの XML 自動生成で作成した XML ファイルとなる．図 9 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，トークン単位での選択が可能となっている．また，穴埋め箇所を設定する際には連続したトークンに対して穴埋め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の選択が可能となっている．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埋め個所として設定される．また，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．その他に，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，スムーズなトークン選択が可能となる．

3.1.4 トレース表用マスク編集機能

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・列及び穴埋めとする部分と配点の重みである．これを教員が効率よ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 10）．

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埋め箇所を設定する．始点及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全てにマスクを設定する機能も実装した．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．また，列の入れ替え機能も実装する予定である．

図 10 トレース表用マスク編集画面

3.2 テーマ・問題登録/編集機能

教師は「テーマ登録/編集画面」でテーマを作成する（図 11）．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/非表示の選択が可能である．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となる．

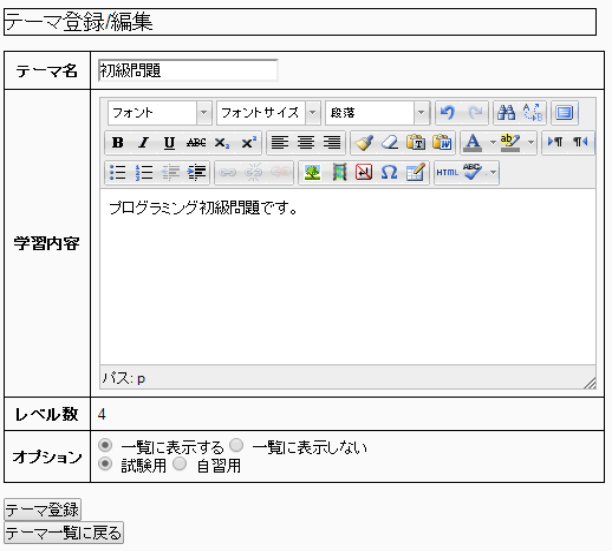

図 11 テーマ登録/編集画面

1 つの問題は，問題タイトル，レベル，配点，言語の種類，属するテーマ，各 XML 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．

図 12 問題登録/編集画面

問題を構成する各 XML ファイルは，作成したユーザ毎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．

3.3 テーマ・問題管理機能

本ツールに登録されたテーマ，問題，XML ファイルの管理を行う．テーマは「テーマ一覧画面」によって新規登録及び編集/削除/コピーができる（図 13）．

図 13 テーマ一覧画面

テーマに登録されている各問題は，テーマ一覧のテーマ毎にある問題一覧によって表示される（図 14）．問題一覧では，各問題の編集/削除/コピーができる．問題のコピーは，同一テーマへはもちろん，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 ファイルを管理（コピー，リネーム，削除）する機能も提供する．

3.4 成績及び学生履歴閲覧機能

学生ごと、問題ごとの成績一覧を表示する。絞り込みやソートが可能である。学習履歴閲覧は、Moodle の機能を利用して実現する。

図 14 問題一覧画面

3.5 学習ログ管理機能

学生の成績データや学習ログを収集し，それを様々な観点から分析することで，学生やクラスの不得意箇所を特定し，教育改善に役立つことが期待される．また，プログラムの理解過程を分析する際や，問題の難易度を定量的に評価する際にも役立つことが期待される。具体的な収集データとしては，学生のユーザ ID，問題 ID，マスの ID，答案，採点結果，日付，時刻等が挙げられる。

4. 教員用機能の実装

教員用機能は表 1 に示すようなファイル群（php ファイル）で構成される．この中で，今回実装を行った機能を下線で示す．以下で，これらの機能の実装に関する詳細を述べる．

表 1 教員用機能を構成するファイル

画面・機能	ファイル名
テーマ・問題管理	
• テーマ一覧	thema_list
• 問題一覧	problem_list
• XML ファイルの検索	retrieve_xml_file
• XML ファイルの管理機能	manage_xml_file
問題生成	
• プログラム XML の自動生成	generate_program
• トレース表の自動生成	generate_trace_table
• 実行チェック	run_check
• プログラム用マスクの編集	generate_mask_for_program
• トレース表用マスクの編集	generate_mask_for_trace_table
テーマ・問題登録/編集	
• 問題の定義	define_problem
• プレビュー	preview_problem
成績の閲覧	
• 成績一覧	show_result
• 学生毎表示	show_result_of_a_student
ログの管理機能	
• 学生の学習ログの収集設定	manage_log_setting

4.1 プログラムの XML ファイルの自動生成

cpp ファイルから 2 節で述べた DTD に従って XML 文書を自動生成する．プログラムを XML 文書に変換するには，プログラムの各ステートメントが DTD のどの要素と一致するのか解析し，解析結果に応じて XML ノードを作成/追加していく必要がある．そこで，本機能ではまずプログラムを行毎にトークンに分割する．そして，行毎のトークン

から対応する要素を導き出す.

例えば，変数を定義した文「int a = 0;」であれば，「int」「a」「=」「0」「;」のように分割される．この文が DTD においてどの要素に対応しているのかを判定する際には表 2 に示す条件を用いる.

表 2 要素の判定条件

条件	対応する要素
文中の'//'と一致したトークンから末尾トークンまで（ただし，末尾トークンが';'でない）	comment
文の構造が「データ型 ルーチン名(パラメータ){」と一致する	routine
routine ではなく，文の先頭トークンがデータ型である	definition
上記条件のどれにも当てはまらない場合	
文の末尾トークンが'{'である	compound-statement
文の末尾トークンが';'である	simple-statement

トークンで分割された文を上記の条件と比較し，該当した条件の要素として XML 文書を構築していく．上の例でいえば，「int a = 0;」は definition 要素に該当する.

上表の条件の他にも，本機能は様々な記述方法に対応するために追加条件を設けている．例えば，文の末尾トークンが'{'でも';'でもない場合には次の行まで文が続いていることが考えられる．また，routine 要素の記述方法として末尾トークンの'{'を次行で記述することがある．しかし，全ての記述パターンに対応しようとした場合，更に複雑なアルゴリズムが要求される．そのため，現状ではプログラムの記述に対して，ある程度の制約を設けることで対応している.

出題するプログラミング問題では，プログラムとトレース表の相互関係を表す情報としてステップ番号が用いられている. 本機能では, このステップ番号の自動生成も行う. ステップ番号は 2.1 節で説明した correspondence 要素で表す．プログラム内でステップとなる部分は，definition 要素及び simple-statement 要素である．構文解析時には，これらの要素を検出するごとに新たなステップ番号を割り振る.

以上，述べたように XML ファイルの生成を行うが，生成対象のプログラムが正しく動作しなければ，生成した XML ファイルも用いることができない．そのため，XML 文書に変換する際には，プログラムのコンパイルを行い，コンパイルが成功するかどうかの確認を行っている．Moodle 上でプログラムをコンパイルする際には，php の関数 proc_open を用い，外部コマンドとして g++コマンドを実行している.

4.2 トレース表の自動生成

自動生成されたプログラムの XML ファイルと，与えられる入力値を元にトレース表の XML ファイルを自動生成する．トレース表は主にスキーマ部分とステップ部分に分けることができる．スキーマ部分にはプログラム内の変数名やトレース表の各列項目名が記述される．このスキーマ部分（schema 要素）は，プログラムの XML ファイルから definition 要素を参照すれば作成できる．トレース表の自動生成において，多くの割合を占めているのは，各ステップ部分（step 要素）の生成である.

トレース表の各ステップでは，当該ステップにおける各変数の値や出力値，ステップ番号が表される．そのため，ステップ部分を生成するには, 実際にプログラムを実行し，各ステップの変数値，出力値を取得する必要がある．プログラムの実行は，4.1 節で述べたコンパイルと同様の方法で，外部コマンドとして実行できるが，各ステップの変数値，出力値の取得はプログラムを実行しただけでは取得することができない. そこで, 実行するプログラムに対して，以下の手順で修正を加えることで，ステップ毎の実行結果を取得する.

① プログラムの XML ファイルから correspondence 要素を取得する.
② correspondence 要素から，プログラム用 XML ファイルにおける，ステップ番号と対応する definition 要素または simple-statement 要素を確認する.
③ ②で該当した要素に, 当該プログラムで定義されている各変数値を出力する文を追記する.
④ 出力文が追記されたプログラム用 XML ファイルから, 実行可能なプログラムファイルを生成する.

①及び②で挙げた要素の定義に関しては，2.2 節で述べている．③において，プログラム内で定義されている変数を取得する際には, プログラム用 XML ファイルを参照し，変数の定義（definition 要素）を取得すればよい．出力文を追記する際には，基本的にはステップに該当する要素（definition 要素または simple-statement 要素）の子要素末尾として追加するが，例外として if 文や for 文，switch 文などはカンマ演算子を用いて，判定条件を評価する前に変数値を出力する．これは，トレース表内で，ステップとして記述する必要があるためである．④については，元の XML ファイルからタグを削除することで, 実行可能なプログラムファイルを生成する．また，③で利用する出力文には std::cout 文が用いられているため，IOStream ライブラリが必須となる．そこで，プログラムファイルを生成する過程で, プログラム先頭に IOStream ライブラリをインクルードする文を必ず追記することにしている.

以上の修正を加えたプログラムを実行し，ステップ毎の変数値を取得すば，それを基に DTD に従ってノードを生成することで，トレース表の各ステップ部分の XML 文書を生成できる.

今回実装したトレース表 XML 自動生成機能は，ステップ，ルーチン，変数，出力といった列項目で構成された基

本的なトレース表を生成するものであり，本ツールの C++/C 言語の開発範囲全てに対応できるものではない．現時点での未実装項目は以下となる．

- 構造体，ポインタを含むプログラム
- ファイル入出力を含むプログラム
- 再帰表現を含むプログラム

上記を含むプログラムは，トレース表の列項目として独自の項目（入力ファイル名，出力ファイル名等）を追加する必要があるため, 現時点では自動生成できないが, 今後，順次実装していく予定である．

4.3 プログラム用マスクの編集

画面上のプログラムのトークンを選択し，「マスク設定」や「非表示」を押すことでマスクを設定する．本機能では，不具合の発生を防ぐために，「マスク設定」や「非表示」を押すたびに XML を更新するのではなく，画面表示のみ更新を行う．マスクの情報は，「マスク設定」や「非表示」を押すたびに, 図 15 のようにフォーム内に input 要素の hidden 属性でマスクの情報として XPath や重みをもたせた要素を追加する．そして，遷移先でその値を元に XML を作成するように設計している．

```
<form…>
<input type=”hidden” name=”hiddens[]” value=”o/o/comment”>
<input type=”hidden” name=”hiddens[]” value=”o/o/comment”>
<input type=”hidden” name=”questions[]” value=”o/o/token,1”>
</form>
```

図 15 プログラム用マスクの生成

4.4 トレース表用マスクの編集

本機能は，基本的にはプログラム用マスク XML の作成機能と同様の方法で実装している．相違点は，トレース表の各ステップの表示・非表示が設定できる点である．これは，各ステップ行に対応するチェックボックスを配置することで行う．また，本機能には列の入れ替え機能も存在するが，現時点では未実装である．

4.5 テーマ・問題登録/編集機能

本機能は，学生に出題するテーマや問題を設定して登録することで，テーマや問題一覧として表示できる機能である．これらテーマや問題の情報は，MySQL テーブルに保持するように設計する（表 3，表 4）．これは，Moodle が各種のデータを MySQL で管理しており，MySQL テーブルに対するアクセスルーチンも提供しているためである．

表 3：テーマテーブル(pgtracer_thema)

主キー	名前	データ型	説明
○	id	int	テーマ ID
	tracer_id	int	モジュール ID
	name	varchar	テーマの名前
	qview	int	学生への表示・非表示
	qmode	int	試験/自習モード

表 4：問題テーブル(pgtracer_question)

主キー	名前	データ型	説明
○	id	int	問題 ID
	title	varchar	問題のタイトル
	tracer_id	int	モジュール ID
	thema_id	int	テーマ ID
	qlevel	int	問題の難易度
	point	int	問題の配点
	program	varchar	プログラムの XML
	trace_table	varchar	トレース表の XML
	mask_for_program	varchar	プログラム用マスクの XML
	mask_for_trace_table	varchar	トレース表用マスクの XML

5. 試用者のコメント

今回，4 節で実装した各機能を，数名の方に試用してもらい，いくつかのコメントを得ることができた．実際に，このコメントを元に機能を改良したものもある．以下に試用者のコメントの一部を示す．

- 問題一覧画面にて, 問題の表示順を変更できるようにしたい.
- 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。
- 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.

これらのコメントは，教員の作業効率の向上が期待できるものが多い．今回得たコメントを元に，機能を修正し改良していくことが重要である．

6. まとめと今後の課題

本稿では，プログラミング教育支援ツール pgtracer の穴埋め問題を構成する各 XML 文書の設計および教員用機能の実装について述べた．今後は，教員用機能の試用を通じて収集したコメントを検討し，機能改善に役立てていきたい．また，実際に学生に使用してもらうテスト段階まで到達することを目標に，未実装機能の実装を進める．

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ム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 11-2P-01
[2]太田，柳田，大月，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 の概要と学生用機能の実装，情報処理学会第 124 回コンピュータと教育研究会，2014
[3]大月，太田，柳田，掛下，XML を用いた穴埋め式プログラミング問題の記述，平成 25 年度電気関係学会九州支部連合大会 11-2P-03