Vol.2013-CE-122 No.1
Vol.2013-CLE-11 No.1
2013/12/14

タグ付き解答例プログラムからの プログラミング問題コンテンツの自動生成

喜多村　和誠[1,a)]　玉木　久夫[2,b)]

概要：本研究では，プログラミング問題の解答例プログラムにタグを用いて情報付与を行い、それに基づいて問題文，配布プログラム，解答検査プログラムなどの問題コンテンツを自動生成する方式を検討する．Java 言語によるプログラミング問題を対象としてこの方式の実装を行い，個々のコンテンツを手作業で作成した場合との入力文字数の比較によりコンテンツ作成効率の向上効果を評価した．

1. はじめに

1.1 研究の背景

プログラミングの習得のために不可欠なプログラム作成演習においては，作成されたプログラムの正誤判定や，アルゴリズムやスタイルについてのアドバイスなど、評価や支援の自動化の余地が大きい．実際，その目的で開発されたシステムやそれらの教育現場での運用についての多数の報告がある [1], [2], [3], [4], [5], [6], [8], [9]．それらの支援システムやツールは学習効果の向上に効果があることは疑いないものの，広く一般の教育現場で普及しているとは言い難い．多くは開発を行った大学等の教育機関やその周辺で使用されるに留まっている．また，研究として報告される以外に，プログラミング教育の担当者が必要に迫られて独自のツールやシステムを開発して使用している場合も少なくない．

これは，教育のために費やされる資源の有効利用の観点から極めて残念な状況と言わざるを得ない．比較的少数の洗練されたシステムが世界中の教育機関で使用されるようになり，共通の基盤のもとで多くの人によって開発された莫大なコンテンツが共有されるようになれば，プログラミング教育の質の向上に大いに資するものと期待される．この望ましい状況を現実のものとするためには，支援の方法やコンテンツそのものだけでなく，コンテンツの効率的な開発方法についても研究を行う必要がある．

発表者らのグループは，その所属する情報科学科のプログラミング入門授業で用いる学習支援システムを開発し [6]，その後継システムを目指した様々な研究を行って来た [7], [10], [13]．現在は，Java 言語によるプログラミングの学習支援システムの設計と開発を行っている．このシステムは，学習者のモデルに基づき個別化され，かつ高度に対話的な支援を目指している．このように「知的」とも呼べる機能を，我々は人工知能的なアプローチではなく，徹底的な作りこみによって実現しようと考えている．そのためには，説明や演習問題等のコンテンツに多量の付加情報を与える必要があり，コンテンツの記述法と開発法は益々重要な課題となっている．

1.2 研究の概要

この研究では，プログラミング教育・学習において中心的な役割を果たすプログラミング問題のコンテンツ開発法に焦点を当てる．ここでいうコンテンツは，問題文や解答検査プログラム，そして必要に応じて事前に配布されるテンプレートプログラムや事後に提示される解答例プログラムなどを含む．

我々は，これらのコンテンツのなかで，解答例プログラムが問題についての情報を最も多く含むことに着目し，解答例プログラムにタグを用いて情報を付与したものから，すべてのコンテンツを生成する方式を検討している．この方式には次のような利点が考えられる．

(1) 解答例プログラムの主役化．　通常のプログラミング問題の準備は解答例プログラムの作成を伴う場合と伴わない場合があるが，前者の場合でも事後の配布を前提としない限り，解答例プログラムがコンテンツとして意識されない場合が多いであろう．しかしながら，問題作成者が実際に解答のプログラムを作成してみる

[1] 明治大学大学大学院理工学研究科
Meiji University, Graduate School of Science and Technology
[2] 明治大学大学理工学部
Meiji University, School of Science and Technology
[a)] kazuaki@cs.meiji.ac.jp
[b)] tamaki@cs.meiji.ac.jp

ことは，問題が意図した通りの練習内容を含んだものであるかを確認するために重要であり，常に推奨されるべきだと考える．解答例プログラムを中心に問題を作成することにより，この望ましい実践が必然となる．

(2) 重複する情報の一元化．種々のコンテンツには，同一の記述や，一方が他方から容易に導くことができる記述が重複して現れる場合がある．これらの重複する記述をひとつにまとめることにより，開発の労力を低減することができる．

(3) 同一の題材に基づく問題変種の複数生成．同一の解答例プログラムから，学習者のレベルなどに応じて種々の問い方をする問題変種の生成が可能になる．

(4) 複数の枠組みへの対応．解答検査プログラムは，多くの場合単独で動作するわけではなく，学習支援システムの用意した枠組みのなかで動作する．解答検査プログラムを陽に記述せずに，解答例プログラムとその付加情報から生成する方式では，複数のシステムや，同一システムのなかでの枠組みの改訂に容易に対応できる．

本研究では，Java 言語によるプログラミング問題を対象としてこの方式を試験的に実装し，上の項目（2）の利点を評価するための実験を行った．ソースファイル（タグ付き解答例プログラム）の文字数と，生成されたファイルの文字数を比較し，次の結果を得た．

(1) 生成されたファイルの総文字数の合計は，ソースファイルのそれを大きく上回る．

(2) 生成されたファイルを手作業で作成する場合には，テンプレートが用意されることを想定し，テンプレートを除いて純粋に手入力が必要と思われる文字数だけを数えた場合，その合計は典型的な問題の場合ソースファイルの文字数とほぼ等しい．

(3) 直訳問題と呼ばれる形式の問題（日本語によりステップごとに詳細に記述されたプログラムを Java プログラムに直訳する問題で，山下 [13] により提案され初心者クラスでのアンケートで，場面によって有効であるとして支持を得たことが報告されている）では，生成ファイルの手入力が必要な文字数の合計は，ソースファイルの文字数を大きく上回る．

これらの実験結果をもとに，この方式の潜在的効果と今後の研究の方向について考察した．

2. 実装内容

この節では，前節で述べた方式に基づくコンテンツ生成の具体的な実装内容について述べる．

2.1 プログラミング問題コンテンツ

この研究では，Java 言語によるプログラミング学習支援システム MAX/Java[12] を対象として，提案方式の実装と実験を行う．MAX/Java におけるプログラミング問題のコンテンツは HTML で記述された問題文と，Java のクラスとして記述された提出プログラム検査コードからなる．後者をテストクラスと呼ぶ．テストクラスを実行する枠組みは，教育用 Java プログラム検査システム Jast[11] から流用されたものである．この枠組みにおいては，2 種類のテストクラスが扱われる．1 番目は，提出クラスの任意のメソッドの単体テストを行うものであり，Junit に準拠した記法で記述する．2 番目は，入出力の検査に特化したものであり，OutputTest インターフェイスを実装する．OutputTest インターフェイスでは次のメソッドが定義されている．

- public String[ ] getInputs() テストに使用する入力値を文字列として返す．配列の長さはテストを行う回数になる．
- public String getMainClassName() main メソッドがあるクラス名を返す．特に指定しない場合は null を返す．値が null の場合はシステム側で提出プログラム内から main メソッドのあるクラスを探し出す．
- public boolean isValidOutput(String output, int index, String input, List<String>messages) 提出プログラムを実行した結果が正しい出力だったかを判定する．output は提出プログラムの出力，index はテストケースの番号，input は用いたテストケースの入力，messages はテスト結果から学習者に伝えるメッセージを記録するためのリストである．

本研究では，部分的な実装を求めるプログラミング問題の場合に問題文に付属する配布プログラムも問題コンテンツに含めて考える．また，問題文には，入出力の例を標準的に含めることにした．これは，学習者がまず自分で十分なテストを行うことを奨励するためである．さらに，問題文には，実装するクラスのメソッドの仕様（Javadoc 形式）を標準的に含めることとした．これは，部分的な実装の問題と全実装の問題を統一的に扱うためおよび，学習者にドキュメントの習慣と Javadoc 表現への親しみを身に付けさせる教育的配慮の二つの理由による．

2.2 アノテーションタグ

解答例プログラムに対する情報付与は，XML タグのような形式により行う．これらのタグをアノテーションタグと呼ぶ．アノテーションタグは，Java ソースプログラム中のコメントの中に記述される．開始タグは<tagname>の形式であり，終了タグは</tagname>の形式である．検討しているアノテーションタグのなかには，解答中の部分的コードの変種を併記するための variant タグや解答プログラム中のある点で成り立つべき条件を記述するための assert タグなど，高度に対話的なフィードバックに必要なものが含まれるが，本研究では，2.1 節で述べたコンテン

ツを生成するために必要な次の基本的なタグのみを実装している．

2.2.1 problem タグ

学習者に提示する問題文を記述するためのタグ．開始タグと終了タグの間に，コメントとして問題文を記述する．記述された問題文は，問題内容となる HTML ページ中の div タグ内に埋め込まれるため，div タグ内で使用可能なタグ（p タグや br タグなど）が使用可能である．記述内容を用いて問題文を作成する．直訳問題の場合には，問題文は解答例のコードから逆翻訳によって生成されるので，このタグは必要ない．

2.2.2 implement タグ

学習者に未実装部分が含まれているプログラムを配布し，正しく実装させる事を題意とする形式の練習問題に使用される．implement タグは学習者に実装を要求する部分を示す．解答例プログラムのうち，このタグで挟まれた部分を実装要求のコメントで置き換えたものが配布プログラムとなる．

2.2.3 testcase タグ

提出プログラムに対して与えるテスト入力を記述するためのタグである．テストケース 1 回の入力をひとつの文字列として記述する．テストケースの個数だけ，それらの文字列をカンマで区切って記述する．

2.3 タグの使用例

これらのタグをコメントとして埋め込んだ簡単な例を図 1 に示す．問題は入力として三角形の底辺と高さを与え，面積を計算して出力するというものである．このタグ付き解答例ソースから生成されたテストクラスと問題文をそれぞれ図 2 および図 3 に示す．また，この問題を直訳問題として処理した場合の問題文を図 4 に示す．図 3 と 4 の問題文において Javadoc の詳細部分は割愛する．

3. 解析と生成の手順

この節ではアノテーションタグ付き解答例プログラム解析し必要なファイルを生成するための方法と手順について述べる．

3.1 構文解析

構文解析には，Eclipse の JDT(Java Development Tools) プラグインを用いた．構文解析の結果は AST（抽象構文木）よって表現される．この構文木を Visitor のデザインパターンに基づいて走査し処理を行う．しかしながら，コメント情報は AST に含まれないため，AST 中にある行番号の情報をソーステキストを対応させてタグの処理を行う必要があった．

```
1  import java.util.Scanner;
2
3  /**
4   * 三角形の面積を求めるプログラム。</br>
5   * 三角形の底辺の長さと高さが与えられたときに面積値を出力します。
6   */
7  public class Triangle {
8      // <problem>
9      // 底辺と高の入力が与えられるので，
10     // 三角形の面積を求めて出力してください．</br>
11     // 底辺と高さが０は入力の終わりを示します．
12     // 結果を出力せずにプログラムを終了してください．</br>
13     // 入力は１行に底辺の長さと高さが間にスペースを入れて与えられます．
14     // </problem>
15
16     // <testcase>
17     // "1 2
18     // 4 1
19     // 5 10
20     // 1 11
21     // 0 0",
22     // "0 0",
23     // "19 17
24     // 11 13
25     // 199 1999
26     // 2013 12
27     // 0 0"
28     // </testcase>
29     /**
30      * 三角形の入力処理を行い、
31      * getArea メソッドを用いて面積値を求めて出力します。</br>
32      * 数値の読み込みにはScanner クラスを用います。
33      * Scanner クラスには nextInt メソッドがあり、
34      * int 型の値を読み込むことができます。</br>
35      * 出力はSystem.out.println メソッドを用います。
36      * @param args[] コマンドライン引数(使用しません)
37      */
38     public static void main(String args[]) {
39         int a,b;
40         Scanner in = new Scanner(System.in);
41         while(true) {
42             a = in.nextInt();
43             b = in.nextInt();
44             if(a==0&&b==0) {
45                 break;
46             }
47             System.out.println(getArea(a,b));
48         }
49     }
50
51     // 正しく動くようにgetArea メソッドを実装してください
52     /**
53      * 底辺の長さと高さから三角形の面積値を返します。</br>
54      * 三角形の面積の公式は底辺×高さ/2です。
55      * 注意はこのままの計算だと誤差がでます。
56      * @param base 三角形の底辺の長さ
57      * @param height 三角形の高さ
58      * @return 引数の三角形の面積値
59      */
60     // <implement>
61     public static double getArea(int base, int height) {
62         double area = base*height;
63         return(area/2);
64     }
65     // </implement>
66 }
```

図 1 タグ付き解答例プログラムの例

3.2 問題文の作成

問題文の作成に必要な情報は，問題文, サンプル入力，サンプル入力に対する出力，Javadoc の 4 種類である．問題

```
1  package triangle;
2
3  import java.util.List;
4  import jp.ac.meiji.cs.th.sakai.wasabi.logic.program.max
       .java.OutputTest;
5
6  public class Test implements OutputTest {
7      private String[] answer = {
8              "1.0\r\n2.0\r\n25.0\r\n5.5\r\n",
9              "",
10             "161.5\r\n71.5\r\n198900.5\r\n12078.0\r\n"};
11
12     public String[] getInputs() {
13         String ret[] = {
14             "1␣2\r\n4␣1\r\n5␣10\r\n1␣11\r\n0␣0\r\n",
15             "0␣0\r\n",
16             "19␣17\r\n11␣13\r\n199␣1999\r\n2013␣12\r\n0␣
                   0\r\n"};
17         return ret;
18     }
19
20     public String getMainClassName() {
21         return "Triangle";
22     }
23
24     public boolean isValidOutput(String output,
25     int index,String input,  List<String> messages){
26         String ln = System.getProperty("line.separator"
            );
27       String newAnswer = answer[index].replaceAll("\r\n"
             ,"\n").replaceAll("[\n|\r]",ln);
28       if(!newAnswer.equals(output)) {
29         messages.add("正しい答えが出力されませんでした。");
30         return false;
31       }
32       return true;
33     }
34 }
```

図 2 Triangle のテストクラス

文は problem タグの記述から取得するか，あるいは直訳問題の場合には，解答例のコードから逆翻訳によって生成する．サンプル入力は，testcase タグで記述した 1 つ目のテストケースを用いる．このサンプル入力に対応する出力は，解答例プログラムをリフレクションによって実行することによって求める．最後の Javadoc は Java の標準的なツールを用いて解答例プログラムの Javadoc コメントから生成する．解答例プログラムに Javadoc の記述が無い項目については，空欄を表示するのではなく，欄そのものを削除するようにした．

3.3 テストクラスの作成

生成されるテストクラスは，2.1 節で述べたふたつのタイプのうち 2 番目のタイプであり，OutputTest インターフェイスを実装する．テスト入力に対する正しい出力をテストクラス中で定義された文字列定数によって表現し，isValidOutput メソッドでは提出プログラムの出力をこれと比較することによって正しさの判定を行う．テストクラス中に埋め込む，この正しい出力は，問題文のサンプル出力と同様に解答例プログラムの実行によって得る．

3.4 解答例プログラムと部分実装問題の場合の配布プログラム作成

これらは，タグ付き解答例プログラムソースから不要な部分を削除することで生成できるので，作成は容易である．

4. 実験

この節では前節で述べた実装を用いて行った実験の方法と結果について述べる．

4.1 実験方法

例題をいくつか考え，それぞれの問題に対してタグ付き解答例プログラムを作成し，問題文，テストクラスおよび解答例プログラムあるいは配布プログラムを生成する．生成されたファイルの文字数を数え，ソースプログラム（タグ付き解答例プログラム）の文字数と比較する．生成されたコンテンツを手作業によって作成する場合には，テンプレートなどを用いて入力の手間の減少を図ることを想定し，生成ファイルについては単純な文字数の合計だけではなく，テンプレートを用いた作成においても手入力が必要と考えられる文字数の合計を数える．さらに，作業効率の向上に結び付く要因を分析するためのデータとして，生成ファイルのなかの文字をその由来によって分類して数える．

4.2 実験に使用した問題

実験では，上で例とした挙げた Triangle（三角形の面積計算）と次に示す 5 題の計 6 題，およびそれらのうちで比較的小規模な 4 題の直訳問題版を使用した．

4.2.1 Emirp

入力で与えられた 1 0 進整数が素数で，なおかつその整数を反転して読んだ（1 の位から順に見た）際の整数も素数であるとき，その数値を Emirp と呼ぶ．与えられた数値 N が Emirp かを判定する．入力 N は $0 \leq N \leq 50000$ で，N が Emirp ならば “N is Emirp.”，そうでなければ “N isn’t Emirp.” と 1 行に出力する．

4.2.2 Circle Count

与えられた文字列内にいくつのOが含まれているかを数える．ここで言うOとは丸い部分の事ではなく, 文字の閉じている部分の事を言い，A は 1，B ならば 2 つの閉じている部分がある．文字列には大文字，小文字の英字と数字しか現れない．EOF までにいくつのOがあったかのカウントを 1 行に出力する．

4.2.3 Supermarket

客が商品を購入したときのレシートを作成する．商品には魚（fish），肉（meat），野菜（vegetable），飲み物（drink），菓子（snack），酒（alcohol）の種類があり，菓子と酒には贅沢品としての税が課せられる．またこのスーパーには同じ商品名で値段の違う商品は無い．通常商品の税と贅沢品の税と購入品（名前，種類，値段）が与えられたとき，以

Vol.2013-CE-122 No.1
Vol.2013-CLE-11 No.1
2013/12/14

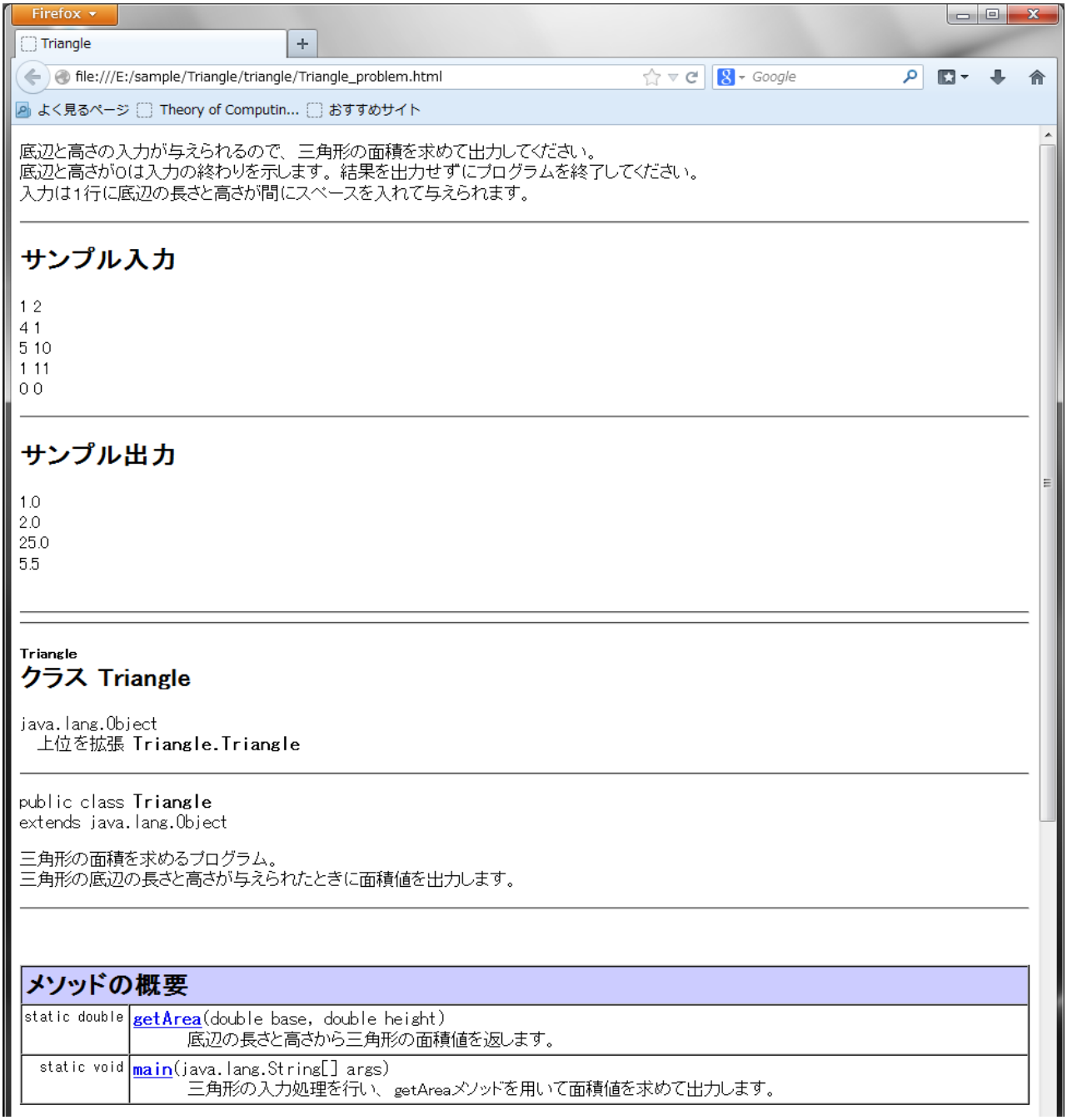

図 3 Triangle の問題文

Vol.2013-CE-122 No.1
Vol.2013-CLE-11 No.1
2013/12/14

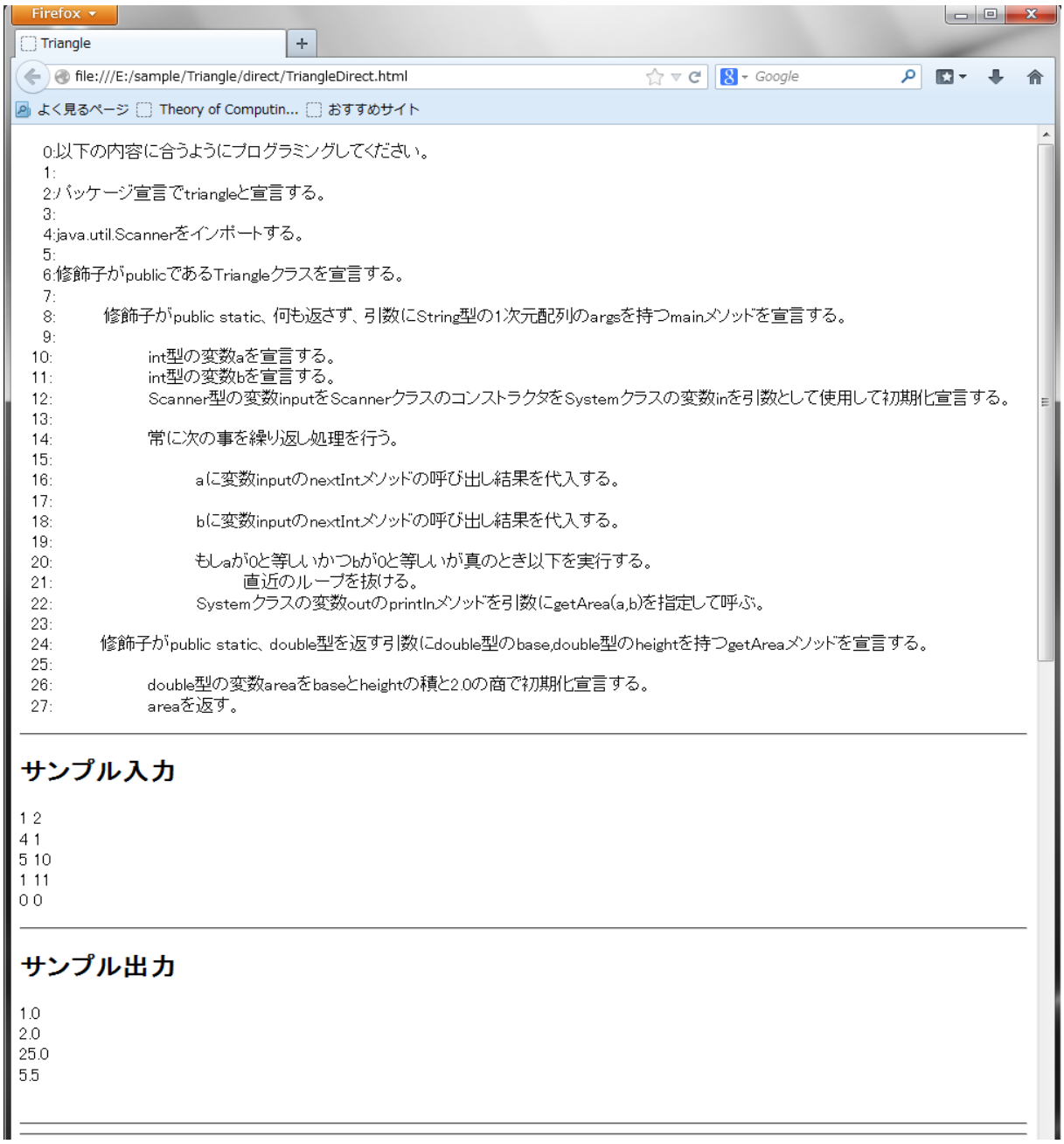

図 4 Triangle(直訳) の問題文

下の形式で出力する．
商品名 1　個数　値段
.....
商品名 m　個数　値段
通常商品の消費税合計　贅沢品の消費税合計　合計支払金額

4.2.4 Time Required

電車の始点から終点までの各駅の間の所要時間を求める．入力は駅の数と終点から各駅への所要時間が与えられる．

4.2.5 Matrix

２つの行列 A と B が行と列の大きさと要素値と共に与えられ，行列 A，B の和と差と積を計算して出力する．もしも、行列の寸法が一致しないときには“miss match”と出力する．行と列の個数はそれぞれ 1 以上 5 以下であり，行列の各要素は-10000 以上 10000 である．また各要素を出力する際は 5 ケタで右詰めで出力する．

4.3 実験結果

表 1 は，各問題に対するソースファイル（タグ付き解答例プログラム）と生成されたファイルの文字数をリストしている．生成されたファイルは，テストクラス（提出プログラムの検査プログラム），問題文（HTML 表記），およびタグ情報を削除した解答例プログラムである．合計の欄は，この三つの生成ファイルの合計を示している．　部分実装問題の場合の配布プログラムは，解答例プログラムから実装要求部分をその指示で置き換えたもので，解答例プログラムとの重複が大きいため，その文字数は表に含めていない．

表中の文字数において，空白の連続は，ひとつの空白として数えている．各生成ファイルについての内訳で，「総数」はファイル中の文字の総数を表す．「コピー」はソースファイルから抜き取ってコピーした文字列に属す文字数を，「解析」は，ソースファイルの解析により生成ツールによって生成された文字列に属す文字数を表している．「総数」から「コピー」と「解析」を引いた値は，生成ツールであらかじめ用意された文字列を挿入した部分の文字数になる．「手作業」はそのファイルを手作業で作成した場合に，手入力が必要と考えられる文字数を表している．すなわち，共通のテンプレートとして用意できるとみなせる部分の文字数を総文字数から引いたものになっている．生成された解答例プログラムは，ソースファイルから不要な部分を削除したものなので，「総数」と「コピー」は等しく，また手作業でもほぼすべての文字の入力が必要である．最右欄の合計手作業文字数から最左欄のソースファイル総文字数を引いたものが，実装した方式を用いた場合の手入力文字数の削減数と考えられ，作業効率化効果のひとつの目安を表す．

扱った問題は，上で挙げた 6 問とそのうち 4 問の直訳問題版である．コード量の比較的大きい Super Market と Matrix については，直訳問題には適さないと考えて除外した．

以下の分析では，タグ付きの解答例プログラムからコンテンツを生成する方式を「生成方式」，手作業によって個々のコンテンツを作成する方式を「手作業方式」と呼ぶ．生成ファイルの総文字数の合計は，どの場合もはるかにソースファイルの文字数を上回る．生成ファイルのうち，手作業で入力することが必要と思われる文字数の合計を，ソースファイルの文字数と比べた場合，最初の 6 問では拮抗している．すなわち，上で定義した入力文字数削減効果は全くないか，あってもわずかである．実装方式によって手入力文字数削減が期待される理由はは次のふたつである．

(1) ソースファイル中の文字列が，複数の生成ファイルにコピーされる．
(2) ソースファイルの情報を解析して生成ツールが生成した文字列が生成ファイルに含まれる．

表の「コピー」および「解析」の欄から，これらの効果がある程度あることが読みとれるが，結果的に手入力文字数の削減につながっていないのは，ソースファイルにおけるタグ記述のオーバーヘッドがあるためであろう．

これらの最初の 6 問については，解析によって生成される文字列は，テストクラスにおいても問題文においても，テスト入力に対する出力に限られている．これが，生成方式の手入力文字数削減効果を限定的にする一番の原因と考えれらる．しかし，出力の長さが長くなる問題 Matrix では「解析」欄の文字数が大きくなり，結果として，ある程度の手入力文字数削減効果を挙げている．

一方直訳問題では，ソースファイルの Java コードから問題文の主な部分が生成されるため，「解析」欄の文字数が大きくなり，手入力文字数の削減効果はどの 4 問においても大きくなっている．

5. 考察と今後の課題

今回のファイル文字数のみによる評価によって断定的な結論を下すことはできないが，問題文と解答検査プログラムのみが必要とされるような従来型の問題に対しては，タグ付き解答例プログラムを通じたコンテンツ開発の能率向上効果は限定的と思われる．直訳問題では，問題文の自動生成による能率向上効果は大きいが，すべての学習内容と学習者に対してこの問題形式が有効なわけではない．また，直訳問題では問題文の生成において，タグが役割を果たしていない．

今後の方向を考える上で注目すべきは，テスト入力に対する出力の自動生成の効果である．タグ記述のオーバーヘッドのために，手入力文字数の削減には必ずしもつながっていないが，部分的には確かな削減効果をあげており，特に出力が長くなる問題ではその効果が大きく結果として

表 1 ソースと生成ファイルの文字数

問題	ソース	テストクラス				問題文				解答例	生成ファイル合計	
		総数	コピー	解析	手作業	総数	コピー	解析	手作業	総=コ〜手	総数	手作業
Triangle	1219	810	48	113	124	3489	163	33	608	447	4746	1179
Emirp	2539	1260	82	535	286	5614	371	98	1061	1314	8188	2661
Circle Count	2578	815	124	44	164	5855	470	7	1202	1235	7905	2601
Super Market	5155	2498	944	885	1364	5206	1358	193	1831	1961	9665	5156
Time Required	1931	1126	249	212	297	3291	646	16	923	573	4990	1793
Matrix	5495	1983	243	1087	532	11054	495	96	2065	3252	16289	5849
Triangle（直訳）	1017	810	48	113	124	4176	163	553	1295	447	5433	1866
Emirp（直訳）	2083	1260	82	535	286	7182	371	1394	2725	1329	9711	4340
Circle Count（直訳）	2122	815	124	44	164	7517	470	1340	2864	1260	9592	4288
Time Required（直訳）	1128	1126	249	212	297	4054	646	655	1686	573	5753	2556

全体の手入力文字数が削減されている．

テスト入力に対する出力は，タグによって付与された情報（この場合テスト入力）と解答例プログラムから自動的に引き出すことのできる情報の自明な例である．タグ付き解答例プログラムを通じたコンテンツ開発の有効性が最大限に発現するのは，人手による記述が困難あるいは多大な手間を要するような情報が，入力例プログラムとタグ情報の組み合わせから自動的に生成できるような場面である．現在発表者らのグループで設計・開発を行っている Java プログラミング学習支援システムでは，高度な対話性を持った問題形式を導入する予定であり，対話性を実現するためのコンテンツの多くはこの状況に該当すると考えている．この枠組みのなかで，生成されるべきコンテンツの具体的な仕様とそれを生成するためのタグの仕様を詰めて行くことが今後の課題である．

謝辞

本研究の方式について、熱心に御議論いただいた藤井聡一朗氏と山下美穂氏に感謝いたします。

参考文献

[1] Al Shamsi, F. and Elnagar, A.: An Intelligent Assessment Tool for Students' Java Submissions in Introductory Programming Courses. Journal of Intelligent Learning Systems and Applications, Scientific Research, 4(1) 59–69, 2012.

[2] Arnow, D. and Barshay, O.: WebToTeach: An interactive focused programming exercise system. Proc. 29th Annual Frontiers in Education Conference (FIE'99), 12A9/39–44 (vol. 1), 1999.

[3] Brusilovsky, P. and Sosnovsky, S.: Individualized exercises for self-assessment of programming knowledge: An evaluation of QuizPACK. Journal on Educational Resources in Computing, ACM, 5(3)Artcle6 (22pages), 2005.

[4] Cheang, B., Kurnia, A., Lim ,A.,and Oon, W.-C.: On automated grading of programming assignments in an academic institution.
Computers & Education, Elsevier, 41(2) 121–131, 2003.

[5] Daly, C. and Horgan, J.M.: An automated learning system for Java programming. IEEE Transactions on Education, IEEE, 47(1) 10–17, 2004.

[6] Fujii, S., Ohkubo, K., and Tamaki, H.: MAX/C on Sakai - A Web-based C-Programming Course. Proc. 2nd International Conference on Computer Supported Education, 196-201 (vol.1), 2010.

[7] Fujii, S., Ohkubo, K., and Tamaki, H.: Tracesheets - Spreadsheets of Program Executions as a Common Ground between Learners and Instructors, Proc. 4th International Conference on Computer Supported Education, 158-163 (vol.1), 2012.

[8] Parsons, D. and Haden, P.: Parson's programming puzzles: a fun and effective learning tool for first programming courses.
Proc. 8th Australasian Conference on Computing Education, Volume 52, 157–163, 2006,

[9] Saikkonen, R. Malmi, L. and Korhonen, A.: Fully automatic assessment of programming exercises. ACM Sigcse Bulletin, ACM, 33(3) 133-136, 2001.

[10] 小芦勇介，玉木久夫:理解度モデルに基づく C 言語学習支援システムの設計と実装および試用，情報処理学会研究報告　Vol.2013-CE-119 No.1 1–8.

[11] 大久保　和則：　プログラミング教育用 Web アプリケーション構築基盤の開発，明治大学大学院理工学研究科２０１０年度修士論文，２０１０年３月.

[12] 山下　美穂:　Web 上の Java プログラミング学習支援システム MAX/Java の構築，明治大学理工学部情報科学科 2010 年度卒業研究，http://www.th.cs.meiji.ac.jp/researches/2010/yamashita/

[13] 山下　美穂: プログラミング学習のための練習問題の多様化と効率的な開発に向けて，明治大学大学院理工学研究科２０１２年度修士論文，２０１３年３月.