コンピュータと教育 58－5
（2000．12．15）

手書きのインターフェイスによるコンピュータ上でのプログラム採点

浅見 美紀、 佐合 尚子、 水野 慈子、 竹田 尚彦
愛知教育大学 情報科学選修

学習者の進度に合わせてプログラム課題を出題する演習問題サーバでは、進度を把握するために細かな成績データを入力しなければならない。一方、プログラム課題の採点作業は紙ベースで手書きで採点しているため、成績データの入力の負荷と採点されたプログラムの管理が問題となっていた。本研究では、教師の採点作業の流れをできるだけ変えることなく、かつ手書きコメントの良さを失うことなく、教師用インターフェイスの改善を試みた。そこで、採点入力に手書き入力インターフェイスCrossPadを用い、採点結果の分析を行い成績の自動的な判断を行うツールを作成した。また、採点結果の一覧性を確保するために、演習問題サーバの教師用インターフェイスも改善した。

Marking of Programs on a Computer With Handwritten Interface

Miki Asami, Naoko Sagou, Chikako Mizuno, Naohiko Takeda
Aichi University of Education Information and Computer Science

The Exercise Server adjusts giving exercise in computer programming to the rate of each student. It has need of detailed report to comprehend the rate. As of now, teachers mark programs by handwritting. So it has been problems that the teacher has a load of input detailed reports and how manage to marked programs. In this study, we attempt to improve the interface for teachers that can mark as current way and keeps goodness of handwritten comment. Therefor, we use CrossPad which has handwritten interface when input. And we construct the tool that analyze the marking result and estimate the results mechanize. We also improve the interface to keep browsability of marking results.

1．はじめに

現在、本大学の情報科学コースでは、プログラミングの演習授業において「演習問題サーバ」[1][2]を使用している。このサーバは、学習者のプログラミング能力に合わせて個別に演習問題を出題する。学習者のレベルに合った課題に取り組むことで、特に、プログラミング能力の低い学生の学習負担が軽減される等の効果が上がっている。

しかしその一方で、サーバを利用する際の教師側のインターフェイスに問題があることも分かった。具体的には、演習問題サーバへの採点結果入力に時間がかかること、採点時における他のプログラム参照が煩雑なこと、の2点である。

現在行っている演習問題の採点は、1）提出されたレポート（紙）に直に手書きでコメントを加える、2）書き加えたコメントに基づき成績データを演習問題サーバに入力する、3）成績の判断に迷ったときは他のレポートを参照して相対的に判断し、データの入力を行う、という3つの作業から成る。

まず、採点入力の手間に関する問題について考える。この問題は、演習問題サーバに入力する成績に関するデータ項目数が多いことに起因する。演習問題サーバでは、学習者に新たな演習問題を出題するために、個人の成績データ（習得項目と呼

ぶ)をあらかじめ入力しておく必要がある。習得項目は各単元ごとに用意されており、「合格」「不合格」「保留」といった判断を下して演習問題サーバに入力する。そのため、各単元ごとに入力すべき習得項目数が増えるので、採点結果の入力に手間がかかることになる。

次に、他人のレポートを参照する手間について考える。採点にあたって、採点結果に一貫性を持たせるために他人のプログラムを再び参照することがある。現在、この作業は印刷された紙ベースで行われているので、検索性が悪く、管理も煩雑である。

このような非効率を改善するためには、レポートを電子メール等の電子的な提出物にし、採点の(半)自動化や効率化をはかることも一案として考えられる。しかし、一方で紙に手書きされたコメントを学生が見ることで、レポートが評価されていることの安心感が得られたり、教師とのコミュニケーションが得られるなどの利点がある。

本研究では、現状の採点の手順1)～3)までを大きく変更することなく、手書きの特性を活かした上で、教師の採点入力時の負担を軽減することを考える。具体的には、レポート上にかかれた手書きコメントを電子的に読み取ることにより、サーバへの入力を容易にする。また、手書きコメントとレポートを重ね合わせ表示することにより、サーバ内で手書きコメントつきのレポートが参照できるようにして、参照性の向上をはかることにした。

2. 演習問題サーバにおける採点

まず、演習問題サーバを用いて行われている従来の採点方法について述べる。

ここで採点とは、学生のプログラム提出から、教師が採点結果をサーバへ入力するまでの一連の作業を指すものとする。

演習問題サーバは、1年次後期「プログラミング入門」、2年次前期「情報処理実習II」の2つの講義で使われている。受講者は、1998年度以降入学の情報科学選修の学生それぞれ約40名である。

2.1 プログラムの提出・採点

学生は、まず出題された演習問題を理解し、仕様書作成や設計を行い、プログラミングする。そして、プログラムが動くことを確認し、ソースプログラムと実行結果を印刷して提出する。教師は、提出されたソースプログラムや実行結果について、採点を行う。このとき、プログラミングスタイルや文法上の間違いなどについての細かな指摘を行い、必要に応じてコメントを付加していく。採点の結果、各課題には5段階の総合評価点が付けられる。評価が低かった課題については、何度でも再提出することができ、総合評価点を上げることが可能である。

また、採点基準を統一するため、教師は適宜採点済みのプログラムを参照する。プログラムの参照を行うために、現在はすべての課題について、学生へ返却する前にプログラムのコピーを取り保存している。教師は、採点時に必要があれば、コピーの中から目的のプログラムを見つけ、採点中のプログラムと比較する。

2.2 演習問題サーバへの入力

教師は採点されたソースプログラムをもとに、演習問題サーバに採点結果を入力する。

図1に、現在使用している演習問題サーバでの採点入力画面を示す。画面左部は、出題された演習問題の内容と、学習者が提出したソースプログラムを表示するようになっている。なお、現時点ではソースプログラムのオンライン提出を実施していないため、ソースプログラムの表示は行っていない。

右上部には、採点項目として、その時点で獲得可能な項目が表示される。「合格」、「保留」、「不合格」のボタンを押すと、選択されている項目が下の各リストに入れられるようになっている。教師は、ソースプログラムを見ながら該当する項目について合否を判定し、リストへの振り分けを行う。

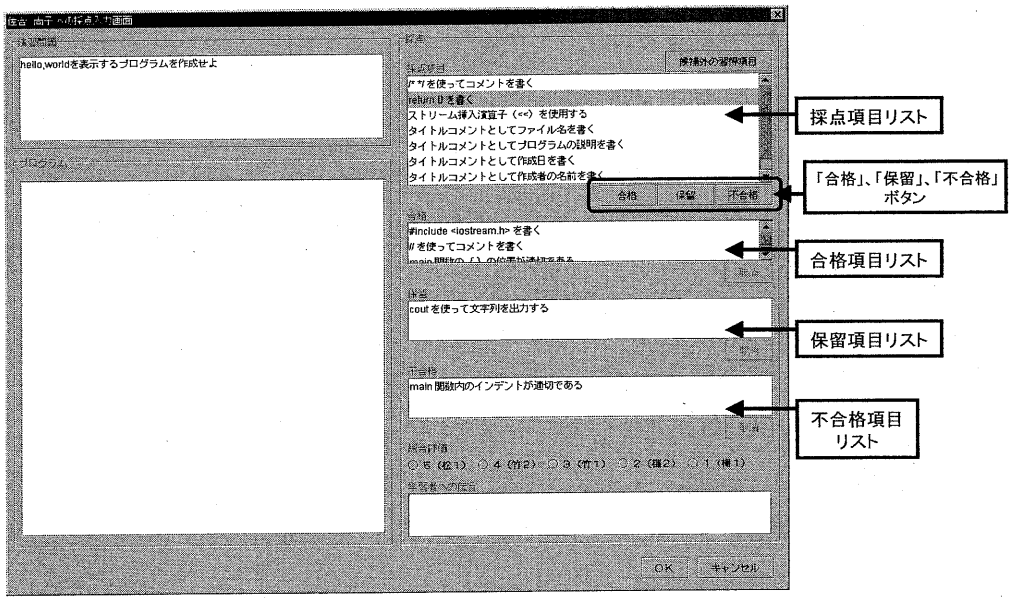

図1： 現在の採点入力画面

なお、本論文では、以下「採点項目」と「習得項目」を同義のものとして扱う。

2.3 問題点

採点作業では、現在教師の負担となっている2点の問題がある。

1つ目は、採点結果をサーバに入力する作業には時間がかかるという点である。

採点項目は、単元が進むたびに新しく追加されていく。また、未達成の項目は次の単元へ持ち越される。そのため、未達成項目が多い学生ほど、学習が進むにつれて採点項目が増えていくということになる。これまでのサーバ使用経験から、1つのプログラム課題について確認しなければならない採点項目の数は、20〜30である。現在教師は、採点入力を手作業で行っているが、1授業40名分の処理を行うには、慣れた者が行った場合でも1時間はかかる。演習問題サーバを使用することによって、教師は1つの授業につき毎週約1時間割かなくてはならないである。

これまで、「採点項目を文法的な項目別に分けておく」などの、表示上の工夫を行ったが、根本的な解決になっていない。

2つ目は、採点する際に他のプログラムを相互参照することが、紙ベースでは煩雑になるという点である。

現在は、過去に採点したソースプログラムをコピーをとり保存しているが、参照のたびに多量のコピーの中から目的のプログラムを探し出さなくてはならない。

3．採点支援機能

本研究では、教師の負担を軽減するための採点支援を行うツールの開発とユーザインターフェイスの改善を試みた。

これまで教師が手作業で行ってきたことをコンピュータ上で支援するためには、採点結果をデジタル化する必要がある。この時教師が行う作業は、従来の採点と比べて変化の少ないことが望ましい。また、コンピュータ上参照するレポートにも、従来通り手書きでコメントが付されているとよい。そこで、従来通り紙の上でペンを用いて採点しつ

つ、採点の結果をデータとしてコンピュータに取り込む方法を考える。

従来と変わらぬ採点作業を実現するために、本研究では手書き入力インターフェイスとしてIBM 社の CrossPad[3]を使用する。教師はCrossPad の上にソースプログラムを乗せ、これまで通りの採点を行えばよい。採点した結果は、手書き部分のみをデータとしてコンピュータに取り込むことができる。また、元のソースプログラムをgif形式の画像として用意しておくことで、手書きのデータと重ね合わせ、教師が行った採点の結果をそのまま再現することができる。 教師はこれまでと同じ作業をしながらにして、必要なデータをコンピュータ内に蓄積することができるのである。

さらに、採点した手書きのデータは、後から一括してコンピュータに取り込むことができる。採点場所についても、コンピュータの前で行う必要がなく、従来以上の制約を受けることはない。

CrossPad からの入力を利用し、本研究ではまず、手書きの採点データを処理するツールを作成する。このツールでは、1）入力された採点データを解析し、不合格と判定される採点項目に該当する記述があるかどうか調べる、2）手書きデータとソースプログラムを重ねあわせ、手書き採点済みのプログラムを作成する、という処理を行う。

そして、ツールにより出力されたデータを利用して、最終的には演習問題サーバ上での採点支援を行う。第 1 の問題解決として、採点項目の事前振り分け、第 2 の問題解決として、コンピュータ上での採点結果閲覧・検索を実現する。

以下で、演習問題サーバで行う処理について述べる。

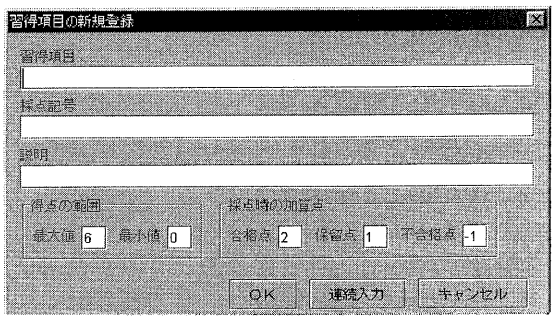

図 2：採点記号入力画面

3.1 習得項目の振り分け

前述したように、で見たように、採点項目を「合格」、「保留」、「不合格」に分けるためには、ボタンを押すという作業が必要である。現在この作業は、教師の手作業で項目ごとに行われており、多くの時間がかかっている。

そこで、あらかじめ採点支援ツールで手書きされたコメントを抜き出しておく。この時、コメントは「不合格」に対応する習得項目のものだけしかないものとする。そして、ツールで抽出した「不合格」に該当する項目を、採点入力画面を開いた時点で下の不合格リストに入れておくという処理を行う。

このようにすると、教師は採点項目のリストに残っている項目についての合否判断を行うことになる。不合格項目については、確認を行うだけでよい。作業の対象となる項目数が減るので、教師が採点入力を行うのに要する時間や負担は軽減される。ここで、処理の対象を不合格項目にしぼったのは、採点時に書き込みを行う場合マイナス要因、つまり減点の対象となるものが大半を占めるためである。

手書きされたコメントが不合格項目に該当するかどうかの判断には、あらかじめ定めておいた採点記号を用いる。手書きの採点データの中にその記号が認識された場合は、対応する習得項目が不合格だということができる。

この採点記号は、演習問題サーバに習得項目を登録する際に、同時に登録を行う（図 2）。対応する採点記号を持たない場合は、空白のままでよい。習得項目とそれに対応する採点記号の例を、表 1 に示す。

3.2 採点結果の閲覧

次に、コピーを取って保存しておいた過去の採点結果をコンピュータ上で閲覧できるようにする。

	習得項目	採点記号
5	プログラムのファイル名をコメントする	File
6	プログラムの説明をコメントする	Content
7	プログラムの作成者氏名をコメントする	Name
8	プログラムの作成日をコメントする	Date
13	main 関数内のインデントが適切である	← or →
14	main 関数の中に return 0 を書く	return
19	1文が適切な長さで書かれている	Long
20	タイトルコメント、プリプロッセッサ文、関数のまとまりを明確にする	>

表1：採点記号

CrossPad 上で採点したデータは、手書き部分のみ画像データとしてコンピュータに取り込まれる。これを、もとのソースプログラムと重ねることで、画面上でも紙面と同じ手書きの結果を見ることができる。今までサーバ上に保存されていなかった採点結果を保存できるため、過去の採点結果の検索を容易に行うことができるようになる。

3.2.1 教師の結果閲覧

採点時には、他学生のプログラムと比較したり、同じ学生の過去のプログラムと比べて改善された点を確認したりという作業が頻繁に行われる。そこで、採点結果ファイルを使って履歴を一覧参照できる機能を、演習問題サーバに追加する(図3)。

参照の対象となるのは、同じ課題が出題された他学生のプログラム、再提出であった場合は、その学生が以前提出したプログラムである。

他学生のプログラムと比較は、同一単元内での採点基準を統一するために必要である。また、再提出されたプログラムを前回のものと比べることは、その学生の理解度がどれだけ進んだかを知る上で役立つ。このように、採点時に他のプログラムを参照することは重要な役割を持つ。

そこで、参照には2つのパターンを用意する。すなわち、1）同一単元での他学生の採点履歴の参照、2）個人別の過去の履歴の参照、である。履歴プログラムを参照しながら採点を行ったり、総合評価を行ったりすることで、採点時の基準統一を計ることができる。

3.2.2 学習者の結果閲覧

学習者に対しては、サーバへのログイン後のメ

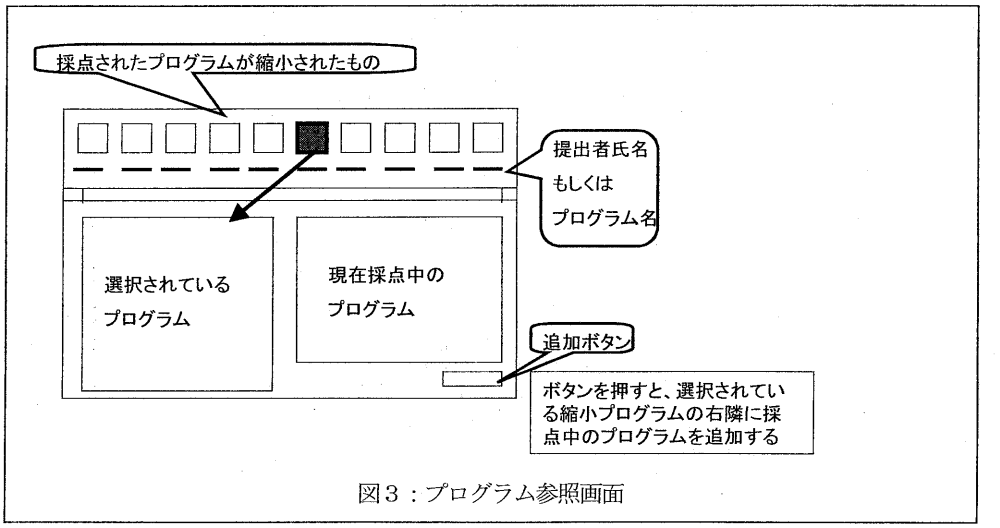

図3：プログラム参照画面

ニューの中に結果に関する項目を追加する。このメニューから、学習者は採点の終わったプログラムを順次、コンピュータの画面上で確認することができる。授業時に一斉返却していた従来の方法に比べ、学生は早いフードバックを得ることができ、印刷物のようにかさばることもない。加えて、画面上で従来どおりの手書きの採点結果を見られるというメリットがある。

4．採点支援ツール

4.1 手書き文字入力システム CrossPad

CrossPad には、開発用のツールキット IBM InkSDK[4] が提供されており、CrossPad からの入力データ（ink データと呼ぶ）を解析、処理するツールを作成することが可能である。ここでは、ink データをもとに、先に述べた演習問題サーバでの採点支援機能に必要なファイルを作成するツールについて述べる。

4.2 開発環境

IBM Ink SDK は、Java 版と C++ 版が用意されているが、ここでは Java 版の SDK を用いる。この、Java 版 SDK は、JDK1.1 版が動作していることが前提となっている。一方演習問題サーバは、Java2 SDK Ver.1.2 を用いて開発されている。

そこで本研究では、演習問題サーバとは別に、採点支援に必要なデータを作成するツールを JDK1.1 を用いて開発する。

4.3 データ

本ツールで扱うデータは次の通りである。また、ツールの構成を図 4 に示す。

＜入力データ＞

① プログラムソースファイル（gif 形式）

② 手書き採点ファイル（ink 形式）

＜出力データ＞

③ 採点結果ファイル（bmp 形式）

④習得項目合否ファイル（txt 形式）

⑤採点結果ファイル（gif 形式）

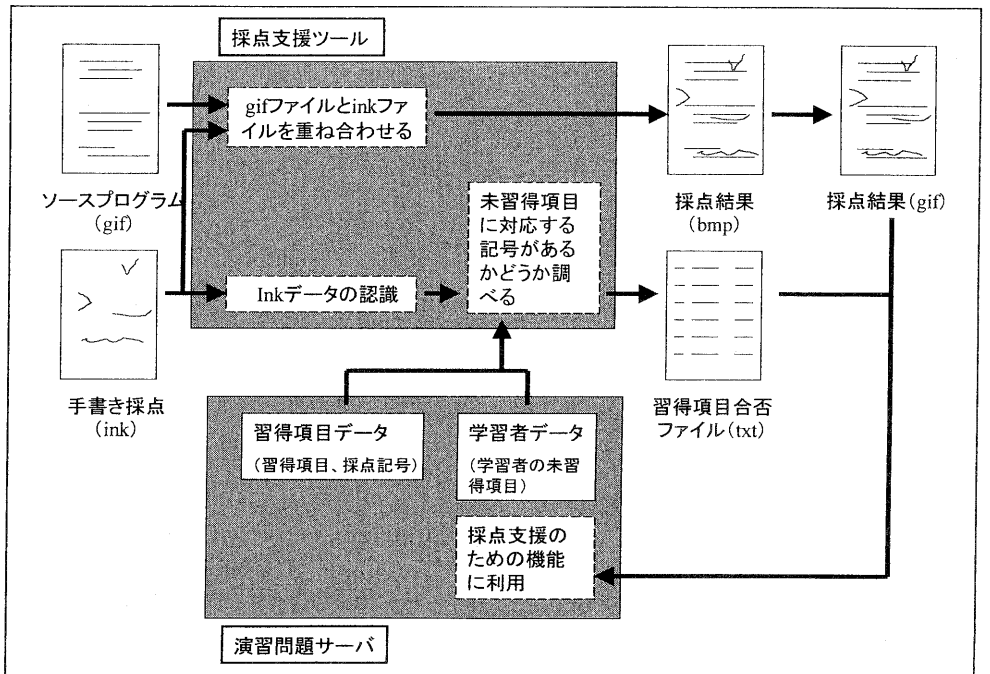

図４：ツールの構成

①プログラムソースを、gif形式の画像ファイルとして用意する。各学生、各課題についてスキャナで取り込む作業を行うことは大変手間がかかるので、txt 形式のソースをオンライン提出してもらい、gifに一括変換できるとよい。

② CrossPad から入力された、手書き文字データ。

③ IBM Ink SDK には、ink ファイルの背景として gif画像をロードする機能がある。これを利用して①、②を重ねあわせる。
Ink SDK では、ファイルを PostScript、Bitmap、Tiff の形式で保存することが可能である。本研究では、重ね合わせたファイルは Bitmap 形式で保存することとした。

④②のファイルをもとに文字認識を行い、採点項目の合否をファイルに保存する。

⑤採点結果のファイルは、後に演習問題サーバで使用するため、Java で扱うことのできる gif形式に変換しておく。

4.4 採点支援機能

4.4.1 授業名登録

出力ファイルの作成は、授業ごとに一括して行うものとする。よって、本ツールは授業名の登録、削除といった機能を持つ。

授業名が新たに登録されると、本ツールパッケージ、Ink Manager、演習問題サーバパッケージのしかるべき場所に、同一名のフォルダを作成する。各データは以後、このフォルダ内に収められることになる。

4.4.2 採点結果ファイルの作成

ソースプログラムと手書きの採点データを重ね合わせた採点結果ファイルを得るには、IBM Ink SDK の Background 機能を使用する。この機能は、選択された ink ファイルの背景として、gif ファイルをロードするものである。ソースプログラムと手書きの採点データを重ねるので、教師が採点したものがそのまま、データとしてコンピュータに保存されることになる。作成されたファイルは、bmp 形式で保存する。ソースプログラムと手書きの採点データを重ね合わせたものの一部を、図５に示す。

4.3.3 習得項目合否ファイルの作成

CrossPad から入力された採点データの中に、採点記号が含まれていたかどうかの結果を、ファイルに記録しておく。このファイルは、提出されたプログラムごとに用意する。

採点項目ファイルは、3種類のデータから成る。習得（採点）項目番号と対応する採点記号、その項目に対する合否判定値である。

習得項目番号は、演習問題サーバに保存されている習得項目に付けられている番号に対応している。

合否の判定値は、初期値を０とする。手書き採点データである ink ファイルを認識し、その結果あらかじめ定めておいた「採点記号」が含まれていれば不合格とみなし、判定値を－１に書き換える。

演習問題サーバにおいて、採点項目とは、採点時に確認しなければならない事項のことをいい、１）出題された演習問題を解くことによって獲得できるような習得項目、２）単元を学習することで獲得できる習得項目、３）達成できていない習得項目、のいずれかに該当する習得項目すべてを指す[５]。

このため、学生がその時点で必要とする採点項

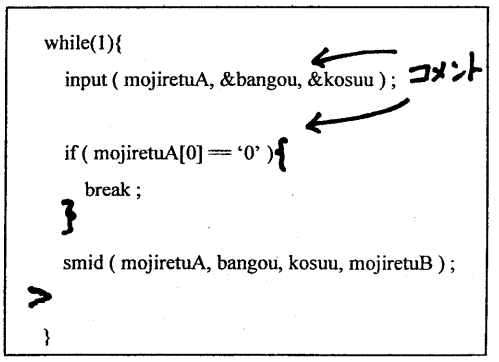

図５：採点結果ファイル

目はそれぞれ異なっており、採点項目ファイルに含まれるべき習得項目の内容や数も一つ一つ違ったものになる。そこで、ファイルを作成するときは、その学生が現時点で必要としている採点項目の情報を演習問題サーバからもって来る必要がある。

5．採点記号認識率

今回は、CrossPad から入力したデータの中から採点記号を正しく認識する頻度を調べた。

認識の対象は、キーボード中の主だった記号に加え、プログラム採点時に使用できそうな英単語である。1クラス分約40名の採点を仮定し、各記号、単語について 40 個分の認識を行った。ただし、過去の採点経験から、1つのプログラムで複数回記入されてきたものについては、認識を 80 個分行っている。

CrossPad に付属のソフトには、手書き文字の認識率を向上させるためのトレーニングセットが用意されている。トレーニングを行うことで、専用の辞書が使用者の手書き文字の癖を学習し、認識の正確さが増すのである。

表2に、トレーニングを1度行った場合と行っていない場合の認識結果を並べて示す。数値の高いものほど、採点記号として実際に利用することが可能である。逆に、数値の低いものは使用することができない。従って、「¥」のように、認識率の極度に低いものについては、採点記号の候補から外していく。また、トレーニングを行った後の認識率は、行っていない時と比べ高くなっているといえる。現在は一度のトレーニングしか行っていないが、回数を増やすことでさらに認識率が向上することが期待できる

今後は、トレーニングを繰り返し行い、認識率を高めていくことができるかどうかを調べる必要がある。

採点記号	1 回のトレーニング	トレーニングなし
File	67.5	50.0
Date	90.0	87.5
Name	80.0	47.5
Content	100.0	92.5
Long	95.0	82.5
> (*)	71.3	47.5
? (*)	75.0	77.5
&	85.0	92.5
{ (*)	87.5	92.5
} (*)	77.5	86.3
¥ (*)	0.0	0.0

数値は正しく認識された割合 (%) を示す。認識は、各記号 40 個分 (ただし(*)のついているものについては、80 個分) 行った。

表 2：認識結果

6．まとめ

現在使用中の演習問題サーバでの教師側インターフェイスに注目し、教師の採点を支援する機能について述べた。現在は手書きデータを処理するツールを作成中である。

今後は、採点記号を決定し、実際の運用を通して負担の軽減を確認することを目標とする。

[1]竹田尚彦、浅見美紀、川口清志：『演習を重視したプログラミングの個別学習』、2000 PC Conference 論文集、pp.151-152、(2000, 7)
[2]川口清志、竹田尚彦：『プログラミング教育における演習問題サーバの開発』、電子情報通信学会信学技法、pp.41-48、(1998,12)
[3]http://www.jp.ibm.com/pc/vlp/vhcrp8b/vhcrp8ba.html
[4]http://www.jp.ibm.com/pc/companion/cp/index.html
[5]川口清志：『個別学習を可能にするプログラミング演習問題サーバの開発』、平成 11 年度　愛知教育大学教育学研究科修士論文、(2000,3)