修士論文

プログラミング演習における効率のよい指導のための間違いの早期指摘に関する研究

平成 23 年度修了

三重大学大学院工学研究科

博士前期課程 電気電子工学専攻

小島　佑介

目次

第1章 はじめに

情報処理技術者を育成するため，あるいは情報化社会における教養を身につけさせるために，プログラミング教育が行われている．しかし，大学で行われているプログラミング演習の講義では，学生がプログラム中の間違いを特定できずプログラム作成が止まってしまうといった問題や，講師は学生の理解度に合わせて一人ひとりに直接アドバイスをするため時間がかかると言った問題などが多く存在する．そのため，限られた時間の中で全ての学生に効率良く指導をするということは困難である．そのため，プログラミング演習を支援するための様々な研究が行われている[1-10].

著者が所属する研究室でも，講師が学生に適切かつ早い時点でアドバイスができるようにすることを目的として，プログラミング演習システム PROPEL(PROgramming Practice Easy for Learners)の開発を行っている[1-5]．PROPEL では学生の作成途中のソースコードを 30 秒間隔で定期的に保存しており，5 分間または 15 分間入力の更新が止まっている学生を講師が分かるようにしている．

しかし，PROPEL によって見つけられたプログラムの更新が止まっている学生は，プログラミングを苦手としており，プログラムには，字句の間違いのような軽微な間違いが，指摘されずに多数残っている状態になっていた．講師は間違い及び修正方法を直接教えるのではなく本人に気がつかせるようにアドバイスを行うために，このような学生に対してはアドバイスに多くの時間がかかる．結果として，講師がアドバイスを行うことができる学生数が少なくなるという問題が生じていた．

本研究では，講師が効率良くアドバイスできるようにすることを目的として，講師以外の者が指摘できない間違いは講師が指摘し，それ以外の間違いについてはシステムで指摘するという方針に基づいて，PROPEL に組み込む機能の開発を行う．これにより，講師は一人の学生にアドバイスするのにかかる時間が短くなり，多くの学生に対して効率よくアドバイスができるようになることが期待できる．

第2章　プログラミング演習システム PROPEL

三重大学の電気電子工学科では C 言語を学ぶためにプログラミング演習の講義が行われている．この講義の受講者は毎年約 80 名いる．それに対して，指導する側は講師が 1 名と TA(Teaching Assistant)が 1 名の合計 2 名である．この講義は，与えられた課題をそれぞれの学生に割り当てられたパソコンを用いて作成するという形式で行われている．

しかし，プログラミング演習では講師の数が学生の数に対して少ないため，講師からすぐにアドバイスがもらえず，プログラム作成をあきらめる学生が出てくるということがある．このようなプログラムの作成ができなくなる学生に対しては早くアドバイスをする必要がある．

そこで，著者の研究室では，講師が学生に適切かつ早い時点でアドバイスを行うために，プログラミング演習システム PROPEL の開発を行ってきた．PROPEL は，学生がプログラム作成をするための学生用画面と，学生のプログラム作成状況を把握するための講師用画面を提供している．PROPEL のシステム構成図を図 1 に示す．

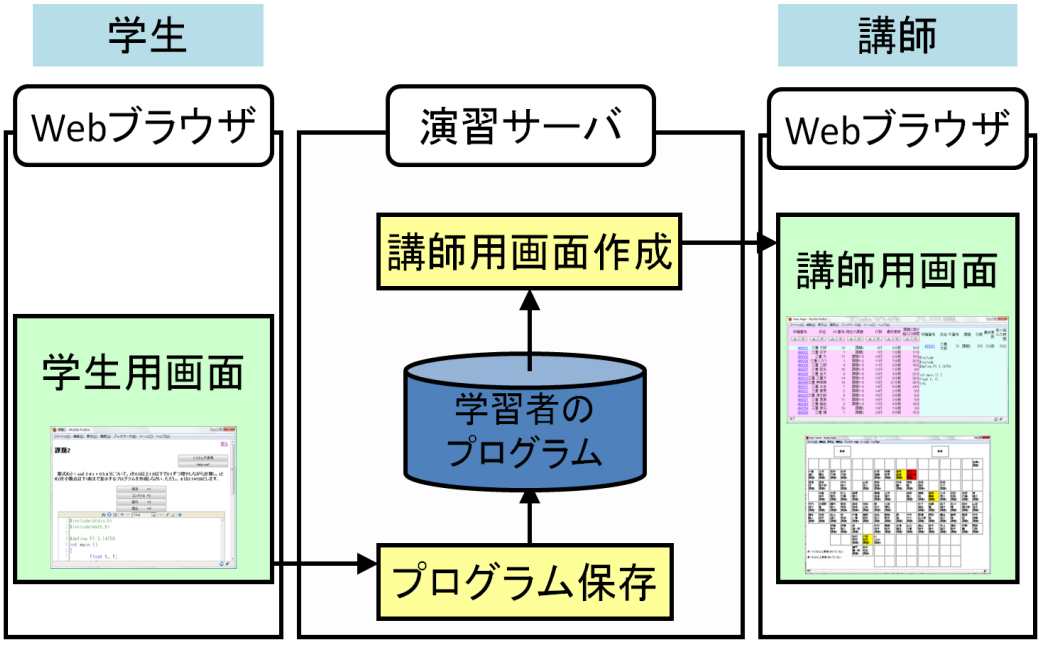

図 1 PROPEL のシステム構成図

学生用画面と講師用画面は，Web ブラウザを用いて開くように設計されている．学生用画面ではプログラムの作成ができる．そこで作成しているプログラムはコンパイル時，もしくは 30 秒毎に自動的にサーバに送信，保存される．講師用画面では，サーバに蓄積されたプログラムを基に学生のプログラム作成状況を把握できるようになっている．次に，学生用画面と講師用画面についての詳しい説明をしていく．

2.1. 学生用画面

学生用画面にはプログラムを作成するためのエディタが用意されている．このエディタで作成しているプログラムは 30 秒毎に自動的に演習サーバに送信される．そして，演習サーバでは送られてきたプログラムを全て保存している．このほかに学生用画面では，プログラムのコンパイル，実行，提出ができるようになっている．講師のアドバイスがほしい時のために，講師を呼び出すための呼び出しボタンも用意されている．この学生用画面を図 2 に示す．

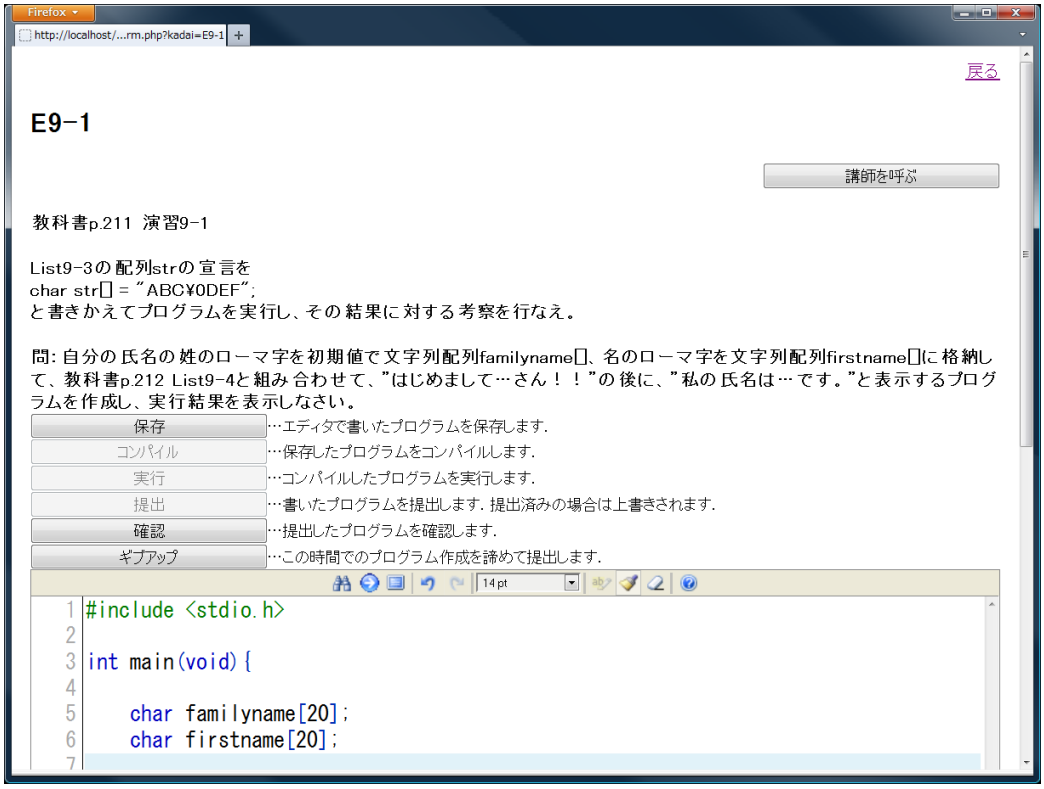

図 2 学生用画面

2.2. 講師用画面

講師用画面は二つ用意されている．一つ目がプログラムの入力が止まっている学生を把握できる演習室状況画面で，二つ目が学生の一覧と選択した学生が作成しているプログラムが表示される学生状況画面である．

2.2.1. 演習室状況画面

演習室状況は，プログラミング演習が行われている演習室の座席表となっている．この画面では，演習サーバに蓄えられたプログラムを基に個々の学生のプログラム作成状況が分かるようになっている．例えばプログラムが 5 分間更新されていない学生の座席は黄色，15 分間更新されていない学生の座席は赤色で表示される．学生用画面で講師の呼び出しボタンを押した学生は紫色で表示される．演習室状況画面を図 3 に示す．これらの講師用画面は，講師が教室を巡回しながら iPad でも見られるように画面設計されている[5]．図 4 に iPad を用いて演習室状況画面を表示している様子を示す．

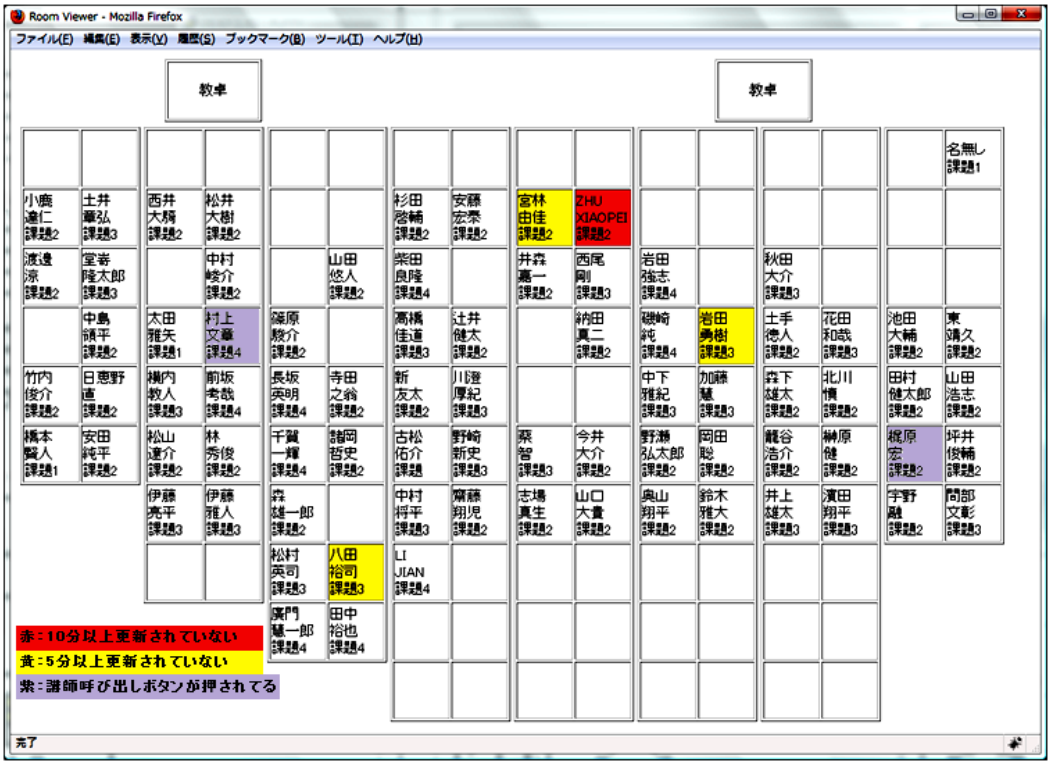

図 3 二つ目の講師用画面

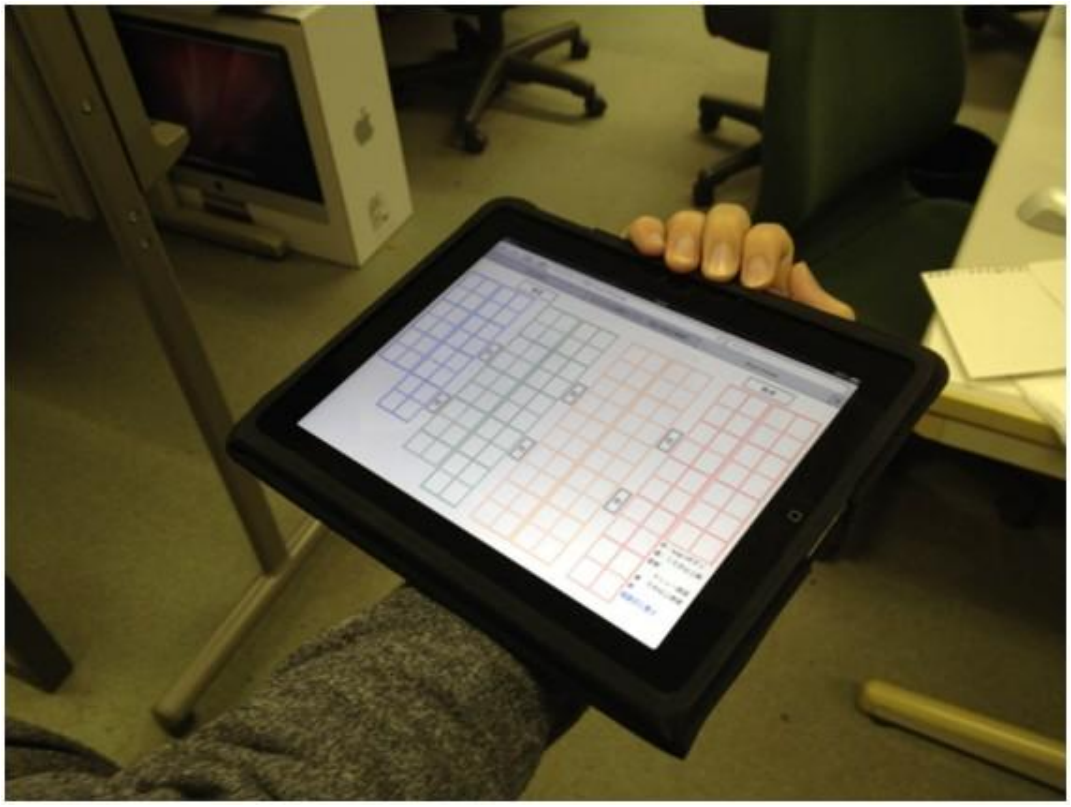

図 4 iPad で表示した演習室状況画面

2.2.2. 学生状況画面

学生状況画面を図 5 に示す．学生状況画面の左側にはプログラミング演習の講義を受講している学生の一覧が表示される．そこでは学籍番号，氏名，使用している PC の番号，取り組んでいる課題名，作成しているプログラムの行数，プログラムの更新が止まっている時間，課題に取り組んでいる時間が分かるようになっている．画面の右側では画面の左側で選択した学生が作成しているプログラムを閲覧することができる．

また，現在，プログラム中にどのような間違いがどの程度残っているのかグラフにして可視化する研究も行っている[4].

図 5 学生状況画面

第3章 PROPEL の現状の問題点

PROPEL により，講師はプログラムの入力が止まっている学生が分かるようになった．しかし，このような学生は，プログラミングを苦手としており，プログラムには多数の間違いが残っていることが多い．講師は間違い及び修正方法を直接教えるのではなく本人に気がつかせるようにアドバイスを行うために，このような学生に対してはアドバイスに多くの時間がかかる．結果として，講師がアドバイスを行うことができる学生数が少なくなるという問題が生じていた．

この原因の一つとして，プログラムを提出する時点でコンパイルを行い，構文間違いの指摘を行っていたことが考えられる．学生は，この時点で初めて，構文間違いの修正を行っていた． そのため，図 6 に示すように，コンパイル前の段階ではプログラムに多くの間違いが残っていた．

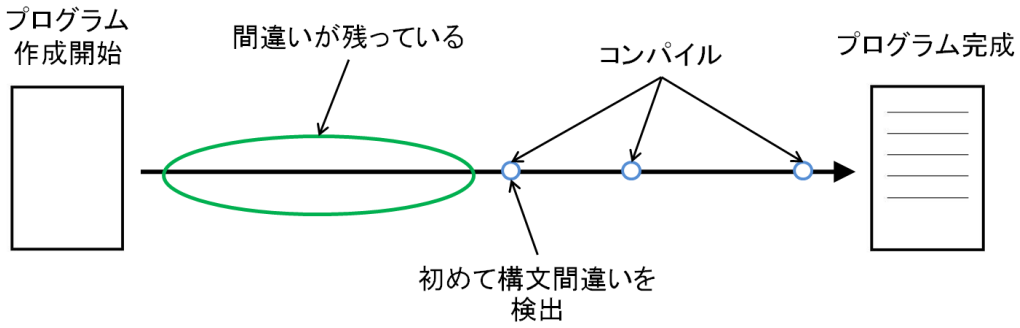

図 6 従来のプログラム作成の流れ

実際に，プログラム中に間違いが残っていた例として図 7 に示すプログラムがある．

行	プログラム
1	#inclde　<stdio.h>
2	
3	int　　year;
4	
5	printf("西暦を入力して下さい."); 　　scanf("%d",&year);
6	
7	if(year>=1896&&year<=1923,year%4==0)
8	puts("夏季オリンピックが開催されました.");
9	
10	else if(year>=1924&&year<=1993,year%4==0);
11	puts("両方のオリンピックが開催されました.");
12	
13	else if(year>=1994&&year<=2008,year%4==0)
14	puts("夏季オリンピックが開催されました.");
15	
16	else if((year>=1994)&&(year<=2008),year%4==2)
17	puts("冬季オリンピックが開催されました.");
18	
19	else
20	puts("オリンピックが開催されませんでした.");
21	return(0);
22	}

図 7 間違いが多く残っているプログラム例

これは，キーボードから 4 ケタの整数を入力し，その年に夏季オリンピックもしくは冬季オリンピックが開催されたかどうかを表示するプログラムである．コンパイルをした時点では以下の 4 種類の間違いが含まれていた．

① #include のタイプミス．

② 「int main(void)の書き忘れ．

③ if 文の条件式を囲む括弧の後ろにセミコロンが書かれている．

④ if 文の条件式内で「&&」を書くべき所にコンマを書いている．

このように，コンパイルをするまで，プログラム中には間違いが修正されずに残されていることがある．講師が，これらの間違いを学生に気がつかせるようにアドバイスしていたのでは，それだけで長い時間がかかってしまい，他の学生にアドバイスをする時間がなくなることになる．

第4章 提案

本研究では，第 3 章で述べた問題を改善するために，講師が効率良くアドバイスできるようにすることを目的として，PROPEL に組み込む機能の開発を行う．具体的には，講師以外の者が指摘できない間違いは講師が指摘し，それ以外の間違いについてはシステムが指摘するという方針に基づき，システムによる間違いの早期に指摘機能を提案する．システムが指摘する間違いの例としては，文末のセミコロンの書き忘れが挙げられる．そのほかにも，条件式内で比較演算子「==」を用いるべきところで代入演算子「=」を用いている間違いのような，文法は正しい間違いも指摘の対象にすることを考えている．間違いの検出は，プログラムの提出時ではなく，システムがサーバにソースコードを送信したときに行う．この様子を図 8 に示す．これにより，学生はプログラムの作成の早い段階でシステムから間違いの指摘を受けることができる．

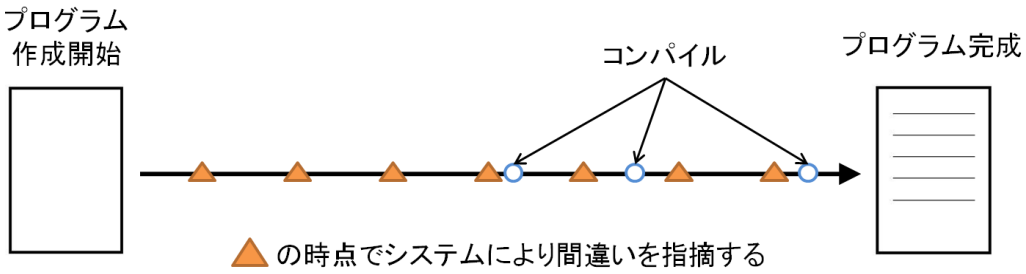

図 8 システムによる間違いの早期指摘

第5章 指摘するべきではない間違い

作成途中でありプログラムとして完結していないため，プログラム中の構文間違いには学生の間違いによるものと作成途中によるものがある．二つの間違いの中で，作成途中によるものは学生に伝えると余計な混乱を招くことになりうる．したがって，これは学生に伝えるべきではない間違いである．しかし，単にコンパイルをしただけでは，検出された間違いがどちらの間違いなのか判断はできない．

例えば, 作成途中のプログラムとして図 9 のプログラムが考えられる．7 行目が作成途中の行である．このプログラムをコンパイルして出てくるエラーメッセージが図 10 である．

行	プログラム
1	#include <stdio.h>
2	int main(void)
3	{
4	int i
5	int x[5];
6	
7	printf("正の整数を /*作成途中の行*/
8	
9	for(i = 0; i<5; i = i + 1){
10	printf("x[%d]=", a);
11	

図 9 作成途中のプログラム

エラーメッセージ
test.c:5:2: error: expected '=', ',', ';', 'asm' or '__attribute__' before 'printf'
test.c:7:9: warning: missing terminating " character [enabled by default]
test.c:7:2: error: missing terminating " character
test.c:9:2: error: expected expression before 'for'
test.c:11:2: error: expected declaration or statement at end of input

図 10 図 9 のプログラムのコンパイル結果

図 10 のメッセージでは，4 行目のセミコロンを忘れていること，7 行目のダブルクォーテーションが閉じていないこと，11 行目で main 関数の括弧が閉じていないことを指摘している．この中で，7 行目に関するエラーが学生に伝えるべきでないエラーである．図 10 のメッセージでは，2 行目と 3 行目のメッセージがそれに当たる．しかし，このメッセージからは学生に伝えるべきではないと判断することはできない．

本研究では，作成途中のプログラムから間違いを検出するために，プログラムの文脈は見ずに，一行単位で間違いを見つける方法をとることにした．そこで，一行単位で見つけられる間違い，もしくは間違いの可能性があるものについて，どのようなものがあるか調査を行った．

第6章 システムで指摘できる間違いの調査

作成途中のプログラムから間違いを検出するために，一行単位で見つけられる間違い，もしくは間違いの可能性があるものについて，どのようなものがあるか調査を行った．

6.1. 調査対象

三重大学の工学部電気電子工学科では，プログラミング演習 I という科目で C 言語のプログラミングについて教育を行っている．この科目は一年生向けの必須科目であり，初めてプログラミングについて学ぶものが多い．学生は毎年約 80 名いる．今回の調査では，PROPEL に保存されている，2010 年に行われたプログラミング演習 I の講義で作成された 5 つの課題の作成履歴を調査した．

6.2. 課題内容

学生が作成しているプログラムに現れる間違いは，その課題の内容により異なってくる．調査に用いた課題の特徴を表 1 に示す．

表 1 課題の特徴

課題	特徴
課題 1	if 文，printf 文
課題 2	if 文，printf 文
課題 3	配列，文字列，printf 文
課題 4	配列，scanf 文，for 文，printf 文
課題 5	配列，for 文，printf 文

6.3. 調査結果

調査の結果，一つの課題毎に，間違いは 15～20 種類存在した．この中には，文末に書かれるべきセミコロンの書き忘れのような典型的な間違いから，あまり見られない間違いまであった．一部ではあるが，調査から得られた間違いと，課題に取り組んだ学生の中でその間違いをしていた人数を表 2 に示す．ただし，課題 1 と課題 2 については，補習で出された課題のため，調査人数が他より少なくなっている．

表 2 調査結果

間違い	間違っていた人数/全学生数					軽微な間違い
	課題 1	課題 2	課題 3	課題 4	課題 5	
セミコロンの忘れ	2/5	2/4	8/79	4/65	11/76	✔
「#include <stdio.h>」の書き忘れ，またはタイプミス	2/5	1/4	4/79	2/65	8/76	✔
等価演算子「==」と代入演算子「=」の間違い. 正：if(a == b)　誤 if(a = b)	2/5	0/4	0/79	0/65	0/76	✔
ライブラリ関数，または変数名のタイプミス	0/5	0/4	5/79	5/65	6/76	✔
「,」と論理積「&&」の間違い 正：if(a<=0 && b<=0 && c<=0) 誤：if(a<=0, b<=0, c<=0)	1/5	1/4	0/79	0/65	2/76	✔
if 文の開始部にセミコロンがある 例：if (・・・) ;	0/5	1/4	0/79	0/65	076	✔
「int main(void)」の書き忘れ	0/5	2/4	2/79	0/65	1/76	
"(" と "{" のタイプミス	0/5	0/4	0/79	1/65	2/76	
括弧の閉じ忘れ	0/5	0/4	0/79	4/65	0/76	

今回調査した結果得られた間違いは以下のように分けることができた.

- 字句の間違いのような軽微な間違い
- プログラム全体を見なければわからない間違い

一つ目の軽微な間違いとは，システムで指摘できる間違いであり，セミコロンの書き忘れやライブラリ関数のタイプミスのような間違いが例として挙げられる．二つ目の間違いには，「int main(void)」の書き忘れが例として挙げられる．この間違いのうち，軽微な間違いについてはシステムで指摘できる可能性があることが分かった.

第7章 間違いの早期指摘機能

調査の結果得られた一行単位で検出できる間違いを対象に，間違いの早期指摘機能の開発を行う．この章で，間違いの早期指摘機能について具体的な説明をしていく．

7.1. 間違い検出の対象とするプログラム

学生用画面で作成しているプログラムは演習サーバに30秒間隔で自動的に送信，保存されている．間違いの検出は学生用画面からプログラムが送信されるごとに，その最新のプログラムを用いて行う．

7.2. 間違い検出の対象とする行

演習サーバには作成途中のプログラムが送られてくる．これを考慮して間違いの検出は一行単位で行うこととした．ただし，作成途中の行は間違い検出の対象から外すこととする．

作成途中の行の判別には，プログラム内にコメントとして記録された数値を用いる．この数値は，作成用画面からサーバにプログラムを送信する際，エディタ内のカーソルがある行を記録したものである．

例えば図11の作成途中のプログラムが演習サーバに送られてくることを考える．

行	プログラム
1	#include <stdio.h>
2	int main(void){
3	
4	char name[50];
5	
6	printf(名前を入力してください。") /*作成途中*/
7	scanf("%s", mame);

図11　一行単位の間違い検出

このプログラムの6行目のprintf文にはダブルクォーテーションとセミコロンが抜けているが作成途中の行である．また，7行目では変数 name

のタイプミスにより出てきた，宣言されていない変数 mame が使用されている．

このような場合, 1 行目から 5 行目と 7 行目のそれぞれについて間違いの検出を行う．6 行目には間違いが含まれているが，作成途中の行であるため，間違いの検出は行わない．

7.3. 指摘対象とする間違い

本研究では，一行単位で見つけられる典型的な間違いを指摘の対象とする．調査の結果得られた間違いの中で指摘の対象とする間違いを表 3 に示す．

表 3 指摘する間違い

番号	間違い
1	文末のセミコロンの書き忘れ．
2	「#include <stdio.h>」の書き忘れ，またはタイプミス．
3	ライブラリ関数，または変数名のタイプミス．
4	printf 文などの「"」の書き忘れ． 例：printf(・・・");
5	scanf 文の「&」の書き忘れ． 例：scanf("%d", a);
6	不等号の書く順番の間違い．　例：if(a =< b)
7	全角文字の使用．
8	for 文の条件式での余分なセミコロン． 例：for(…; …; …;)
9	等価演算子「==」と代入演算子「=」の間違い． 正：if(a == b)　誤 if(a = b)
10	比較の仕方の間違い． 正：　if (a == b && b == c){ }　誤：if (a == b == c){
11	「, 」と論理積「&&」の間違い． 正：if(a <= 0 && b <= 0 && c <= 0)　誤：if(a <= 0, b <= 0, c <= 0)
12	if 文の丸括弧の後ろにセミコロン．　例：if(・・・) ;

7.4. シンボルテーブル

間違いの検出には，宣言されている変数や定義されている関数やマクロの情報を必要とするものがある．そこで，間違いの検出の第一段階としてプログラムからシンボルテーブルを作成する．シンボルテーブルとはプログラム中で宣言されている変数や関数や定義されているマクロに関する情報を記録したものである．今回，変数に関しては，識別子と有効範囲，型，宣言，次元，要素，宣言が正しいかどうか（セミコロンの

書き忘れにより宣言が失敗しているときは，そのことを記録する）の情報を記録する．関数に関しては識別子，型，宣言を記録する．マクロに関してはマクロ名，形式の情報を記録する．

7.5. 間違いの検出方法

本研究では，コンパイラを用いずに一行単位で間違いを検出することとした．そこで，表 3 のそれぞれの間違いを検出する方法を考えた．

最初に，間違いを検出する準備として，対象とする行に書かれているコメントと文字列を削除する．次に，残ったソースコードをトークンに分ける．トークンとは文法上の基本単位であり，予約語，ラベル，識別子，特殊記号，数，文字列からなる．例えば「if (a = b){」は if，(，a，=，b，)，{ の 7 つに分けられる．

対象とする行をトークンに分けたら，そのトークンに対してシンボルテーブルと表 4 の間違いの検出方法により間違いの検出を行う．

表 4 間違いの検出方法

番号	間違いの検出方法
1	文末のセミコロンの書き忘れ． ① ライブラリ関数を探す． ② 丸括弧が閉じているか確認する． ③ 丸括弧の後ろにセミコロンがなければ間違い． ④ ③までで間違いがなかった場合，変数宣言をしているか確認する． ⑤ シンボルテーブルを使い宣言か失敗していることを確認する． ⑥ その原因がセミコロンの書き忘れなら間違い． ⑦ ⑥までで間違いがなかった場合，代入文のセミコロンの書き忘れを確認する． ⑧ トークンの先頭が宣言されている文字列であることを確認する． ⑨ その変数に代入が行われていることを確認する． ⑩ 最後にセミコロンが書かれていなければ間違い．

2	「#include <stdio.h>」の書き忘れ，またはタイプミス． プログラム中で#, include, <, stdio.h, >が連続して書かれているか確認する．
3	ライブラリ関数，または変数名のタイプミス． ① 文字列があるか確認する． ② 文字列があった場合，それが予約語と一致しないことを確認する． ③ インクルードされているヘッダファイル内で定義されているライブラリ関数，シンボールテーブル内の変数，マクロと比較して，一致しなければ間違い．
4	printf 文などの「"」の書き忘れ．　例：printf(・・・"); ① printf, puts, scanf を探す． ② 丸括弧が閉じているか確認する． ③ 丸括弧内にセミコロンが 1 つだけある場合は間違い．
5	scanf 文の「&」の書き忘れ．　例：scanf("%d", a); ① scanf を探す． ② 丸括弧が閉じているか確認する． ③ 待つ括弧内に「,」があることを確認する． ④ 丸括弧内で「,」の後ろに「&」がないことを確認． ⑤ 「,」の後ろに配列，もしくはポインタがなければ間違い．
6	不等号の書く順番の間違い．　例：if(a =< b) ① 「=」を探す． ② 「=」の次に「<」「>」「+」「-」「*」「?」「%」「!」があれば間違い．
7	全角文字の使用． ① ソースコードのバイト数と文字数を比較し一致しなければ間違い．
8	for 文の条件式での余分なセミコロン． ① for を探す． ② 丸括弧が閉じているか確認する． ③ 丸括弧内に「;」が 3 つあれば間違い．
9	等価演算子「==」と代入演算子「=」の間違い． ① if, while,for を探す． ② 丸括弧が閉じているか確認する． ③ 丸括弧内に getchar()が書かれていないことを確認する． ④ 丸括弧内に「=」が単独であれば間違い．

10	比較の仕方の間違い.　正： if (a == b && b == c){ } 誤：if (a == b == c){ ① if, while, for を探す. ② 丸括弧が閉じているか確認する. ③ 丸括弧内で「==」が 2 つ以上出てくることを確認する. ④ それぞれの「==」の間で「&&」か「||」がなければ間違い.
11	「&&」の代わりに「,」の使用. 正：if(a <= 0 && b <= 0 && c <= 0) 誤：if(a <= 0, b <= 0, c <= 0) ⑤ if, while, for を探す. ⑥ 丸括弧が閉じているか確認する. 条件式内で「, 」が出てきたら間違い.
12	if 文などの丸括弧の後ろにセミコロン. 例：if(・・・) ; ① if, while, for を探す. ② 丸括弧が閉じているか確認する. ③ 丸括弧の後ろにセミコロンがあれば間違い.

7.6. 間違いの指摘方法

間違いの指摘は，コンパイラのように英文を用いて指摘を行うのではなく，日本語の文を用いて行う．その際，「エラー」と「注意」という単語を用いて間違いの程度を表す．「エラー」は明らかな間違いであることを意味する．表 3 の 1～8 番がこの間違いに当たる．「注意」は間違いの可能性があることを意味する．表 3 の 9～12 番がこの間違いに当たる．

7.7. 間違いの指摘画面

間違いの指摘は，図 2 の学生用画面の下側に間違いを指摘するメッセージを表示する専用のスペースを設けて，そこで行う．図 12 に間違いの指摘をする画面を示す．

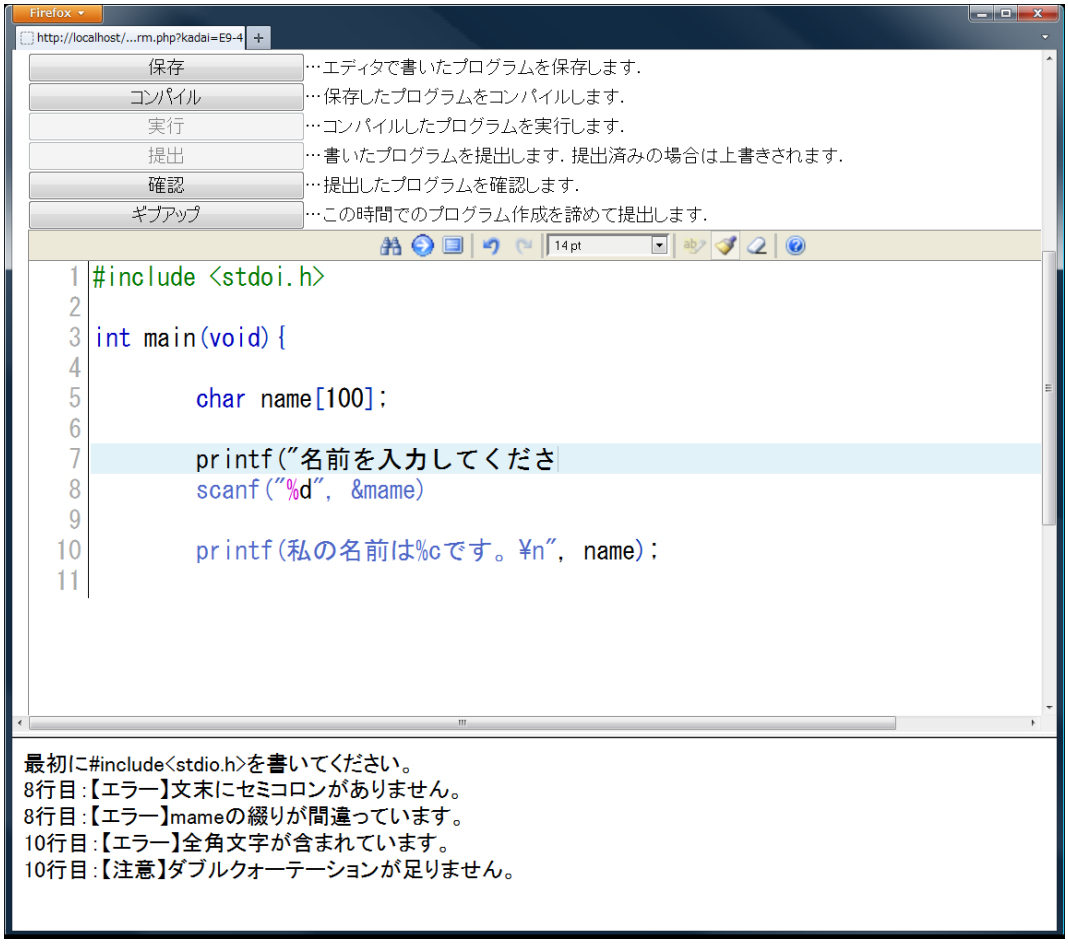

図 12 間違いの指摘画面

7.8. システム構成

図 13 に間違いの早期指摘機能を組み込んだ場合のシステム構成図を示す.

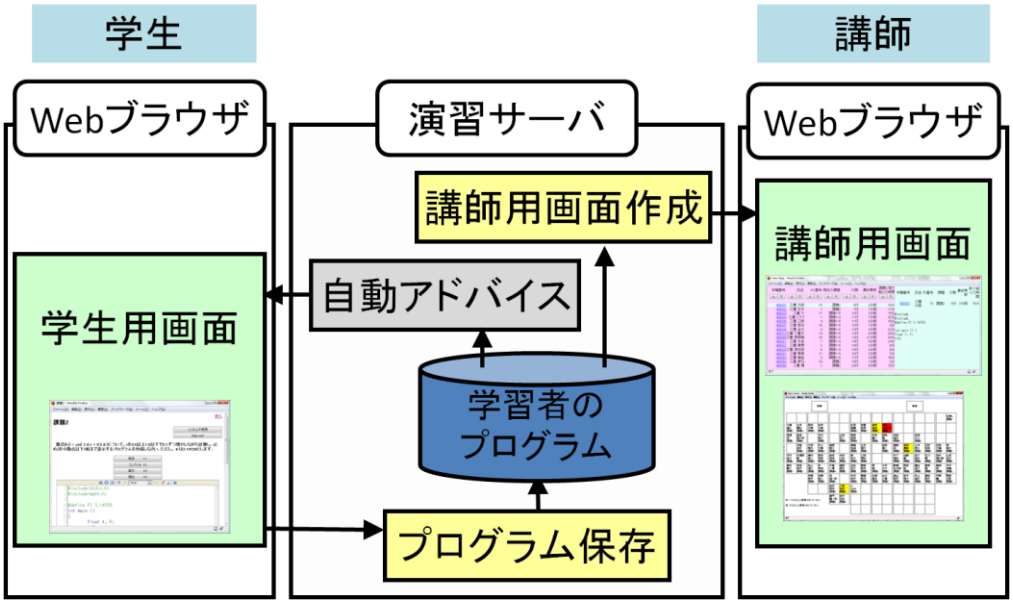

図 13 間違いの早期指摘機能を組み込んだ時のシステム構成図

学生用画面で作成しているプログラムは 30 秒間隔で自動的にサーバへ送信される．サーバでは，受け取ったプログラムを対象に，一行単位で間違いの検出を行う．そして，検出した間違いを，学生用画面に設けた専用のスペースを使って指摘する．

第8章 検証

最後に，本提案により間違いが正確に検出できるのかと，有効性について検証を行った．検証にはPROPELに保存されている以下の課題で作成されたプログラムを用いた．

(1) 2008年度後期のプログラミング演習Ⅰの補講で作成された合計5個のプログラム作成履歴

(2) 2009 年度後期のプログラミング演習Ⅰの講義で作成された合計67個のプログラム作成履歴

(3) 2011 年度前期のプログラミング演習Ⅱの講義で作成された合計31個のプログラム作成履歴

検証では(1)～(3)のプログラム作成履歴の中で出てきた間違いを正確に検出できるかを確かめた．全ての出現した間違いの検出回数と出現回数を表5に示す．

表5 検証結果

番号	間違い	検出回数	出現回数
1	文末のセミコロンの書き忘れ.	18	18
2	「#include <stdio.h>」の書き忘れ，またはタイプミス.	3	3
3	ライブラリ関数，または変数名のタイプミス.	22	22
4	printf文などの「"」の書き忘れ. 例：printf(・・・");	9	9
5	scanf文の「&」の書き忘れ. 例：scanf("%d", a);	2	2
6	不等号の書く順番の間違い.　例：if(a =< b)	3	3
7	全角文字の使用.	4	4
8	for文の条件式での余分なセミコロン. 例：for(…; …; …;)	1	1
9	等価演算子「==」と代入演算子「=」の間違い 正：if(a == b)　誤 if(a = b)	4	4
10	比較の仕方の間違い. 正：　if (a == b && b == c){ }　誤：if (a == b == c){	4	4

11	「,」の間違った使用. 正:if(a <= 0 && b <= 0 && c <= 0) 誤:if(a <= 0, b <= 0, c <= 0)	3	3
12	if 文の丸括弧の後ろにセミコロン. 例:if(・・・);	5	5
13	波括弧'{', '}'の閉じ忘れ.	0	13
14	丸括弧'(', ')'の閉じ忘れ.	0	2
15	int main の書き忘れ.	0	2
16	スペースの書き忘れ. 正:return 0; 誤:return0;	0	2
17	「¥」と「/」のタイプミス. 例:putchar('/n');	0	1
18	scanf 文のコンマの書き忘れ. 例:scanf("%d" &a);	0	1
19	シングルクォーテーション「'」とバッククォーテーション「`」の間違い	0	1

検証の結果，本提案で検出する 1〜12 番までの間違いを正確に検出できた．13〜18 番までの間違いは検出の対象外なので検出回数は 0 となっている．

また，プログラム作成履歴の中に現れた全ての間違い 100 個中 78 個（1〜12 番の間違い）を本提案により検出できた．これは全体の約 8 割である．この結果から，システムで指摘できる間違いはシステムで指摘出来ていると考えられる．これより，講師はシステムで指摘できない間違いについてアドバイスをすればよく，より多くの学生に対してアドバイスができるようになることが期待できる．

第9章 まとめ

著者が所属する研究室では，講師が学生に適切かつ早い時点でアドバイスができるようにすることを目的として，プログラミング演習システム PROPEL の開発を行っている．PROPEL の開発により，講師はアドバイスを必要としている学生が把握できるようになり，適切かつ早い時点でアドバイスができるようになった．しかし，アドバイスを必要としているような学生は，プログラミングを苦手としており，プログラム中には字句の間違いのような軽微な間違いが多く含まれていることがある．このような学生にアドバイスをするには多くの時間がかかる．結果，講師からアドバイスをもらえる学生の数が少なくなるという問題が生じていた．そこで，私はこの問題の改善のために，講師が効率よくアドバイスできるようにすることを目的として PROPEL に間違いの早期指摘機能を組み込むことを提案した．具体的には，PROPEL にコンパイル前の作成途中のプログラムから間違いを検出することを考えた．

検証の結果，本提案によりコンパイル前の段階の作成途中のプログラムからでも約 8 割の間違いを検出できることが分かった．

本研究により，講師は一人の学生にアドバイスするのにかかる時間が短くなり，多くの学生に対して効率よくアドバイスができるようになることが期待できる．

参考文献

[1] 伊冨昌幸，小島佑介，高橋功欣，北英彦：プログラムの作成状況を把握する機能を持つプログラミング演習システム，2010PC カンファレンス（2010）

[2] 望月将行，森田直樹，北英彦，高瀬治彦，林照峯：自動テスト機能を備えたプログラム提出システム，2003PC カンファレンス（2003）

[3] 南山幸紀，山家伊織，高瀬治彦，北英彦，林照峯：プログラミング初心者に間違いの原因に気付かせるためのプログラムの動作の分析結果を用いたアドバイス，電気関係学会東海支部連合大会（2008）

[4] 高橋功欣，小島佑介，北英彦：プログラミング演習における指導のための受講者のコーディング状況の可視化，2011PC カンファレンス（2011）

[5] 小川正，西口大亮，北英彦：プログラミング演習における iPad などの携帯デバイスの利用による指導の円滑化，2011PC カンファレンス（2011）

[6] 小田まり子，掛下哲郎：パターンマッチングに基づいた C プログラムの落し穴検出法，情報処理学会論文誌，Nov.1994 Vol.35 No.11

[7] 櫻井桂一：プログラミング実習を支援するための C プログラム誤り検出システムの開発，日本教育工業会誌/日本教育工業雑誌 25（Suppl.）, 67-70, 2001

[8] 高井健一，香川孝司，垂水浩幸：Web を用いたプログラミング学習ツールの作成，信学技報，IEICE Technical Report ET2003-127（2004-3）

[9] 史一華，徐海燕：プログラミング教育のための可視化ツール開発，信学技報，IEICE Technical Report ET2007-26（2007-9）

[10] 山本耕大，春原将寿，大金克紀，中村勝一，横山節雄，宮寺庸造：エラー要因事例ベースの動的学習手法を導入した C 言語教育システムの開発と基礎的評価，信学技報，IEICE Technical Report ET2008-29（2008-7）

謝辞

本論文は，著者が三重大学大学院工学研究科博士前期課程に在籍中に行った研究をまとめたものである．本研究を進めるにあたり，懇切丁寧な御指導と御督励を賜った三重大学大学院工学研究科の北英彦准教授に感謝いたします．

また，貴重な時間をさいて本研究論文を査読して頂いた，三重大学大学院工学研究科の森香津夫教授，高瀬治彦准教授に深く感謝いたします．

最後に，日頃熱心に討論していただいた計算機工学研究室の皆様方にお礼申し上げます．

実績一覧

[a] 小島佑介，北英彦，伊冨昌幸 : プログラミング演習システム発展のためのプログラム作成履歴閲覧ツール，2010PC カンファレンス（2010）

[b] 小島佑介，高橋功欣，北英彦 : プログラミング演習における効率のよい指導のためのエラー早期指摘, 2011PC カンファレンス 2011, 2011PC Conference 論文集，pp52-53

[c] 小島佑介，高橋功欣，北英彦 : プログラミング演習システムにおけるエラー早期指摘，電気関係学会東海支部連合大会（2011）

[d] 伊冨昌幸，小島佑介，高橋功欣，北英彦 : プログラムの作成状況を把握する機能を持つプログラミング演習システム，2009PC カンファレンス（2009）

[e] 高橋功欣，伊冨昌幸，小島佑介，北英彦 : プログラム作成履歴の分析支援ツールの開発，2010PC カンファレンス（2010）

[f] 高橋功欣，小島佑介，北英彦 : プログラミング演習における指導のための受講者のコーディング状況の可視化，2011PC カンファレンス（2011）

[g] 高橋功欣，小島佑介，北英彦 : プログラミング演習における指導のためのコーディング状況の可視化，電気関係学会東海支部連合大会（2011）