プログラミング学習における誤り訂正問題の自動生成に関する研究

2008MI176 小川 和輝　2008MI211 佐藤 雄基
指導教員：蜂巣 吉成

1　はじめに

プログラミング教育では，プログラムに接する機会を多くすることが重要である．多くのプログラムの作成をおこなうことのほかに，他人が書いたプログラムを読んで内容を理解するコードリーディングや，プログラム中の誤りを発見し，修正するデバッギングをおこなうことも必要である．

プログラミング学習では一般的に与えられた課題に則してプログラムすべてを記述する全文記述問題や，プログラムの一部に空欄を設けた空欄補充問題などが利用される場合が多い．全文記述問題では，プログラムを作成する過程においてデバッギングもおこなうが学習者ごとにプログラムの内容が異なり，一般的な学習において効果的なデバッギングをおこなえるとは限らない．コードリーディングの面では，問題に解答する過程で他人のプログラムを読む機会は少なく，コードリーディングに効果的な学習方法とは言えない．全文記述問題を多数解答することは，解答者だけでなく出題者にとっても，その解答の正誤判定をおこなうことになり負担となる．

空欄補充問題では，他人のプログラムを読み理解する必要があるので，コードリーディングのスキルを養成する助けになる．解答者の負担も全文記述問題より軽く，出題者にとっては問題作成の負担は増えるが，moodle[1] などの出題システムを利用することで，自動採点も可能であり，負担を軽減することができる．しかし，デバッギングの面においては，プログラムの誤りがどこにあるかを見つける作業をおこなわないので，デバッギングスキルの養成には向いていない．

誤り訂正問題は，空欄補充問題と同様に他人のプログラムを読み理解する必要があり，プログラムの誤りのある位置を探す作業も伴うのでコードリーディングとデバッギングのスキルを養成するのに役立つ．解答者の負担も全文記述問題と比較して少ない．しかし，既存のシステムではプログラムの誤り訂正問題を出題し，その正誤判定を自動でおこなうものは少ない．これは次のような点が難しいからである．

1. 誤りを含んだプログラムを効率よく作成する方法が知られていない
2. 訂正した問題の正誤判定を自動化することが難しい
   a. プログラムの変更を一切自由にすると，基のプログラムとは大きく異なる場合がある
   b. 訂正の方法が複数存在する場合がある

本研究では，これらの問題を解決し，プログラムの誤り訂正問題を生成し，その正誤判定を自動で行う方法を提案する．

問題点 1 を解決するために，よくあるプログラムの誤りを分析し，字句誤り，文法誤り，意味誤りに整理した．

問題点 2a を解決するために，整理した誤りごとに，解答時に訂正可能な箇所を制限する方法を提案する．しかし，誤りを設けた場所のみを変更可能とするとプログラムの誤りを探すのではなく，訂正可能箇所を探すことになり空欄補充問題と実質的に同様の問題となるので，類似した記述の箇所も変更可能とする．

問題点 2b を解決するために，整理した誤りごとに，複数解答の可能性を整理し，複数解答が可能な場合にはそれらを判定して正誤判定する方法を提案する．問題点 2a の解決で利用した，変更可能箇所の制限を利用することで，複数解答の組み合わせを限定し，対応するべき複数解答の種類を限定することができる．

2　関連研究

moodle は，PHP で動作する，問題に記述された選択肢から正答を選ぶ多肢選択問題，問題文の正誤を解答する正誤問題，記述問題，テキスト内に問題を挿入できる穴埋め問題などを出題可能なシステムである．誤り訂正問題は記述問題の応用例として作成可能ではあるが，解答の限定方法は，任意の文字を表す記号であるワイルドカードや，正規表現の利用のみである．本研究で用いている方法は moodle と比較して，字句によらない誤りに対応しやすい．

AEGIS[2] は，出題文，出題箇所を指定するタグをもちいて製作された XML ドキュメントから，システムのもつ基準に従って問題を作成し，正誤判定を自動でおこなうシステムである．しかし，問題ごとに XML のタグを埋め込む必要があり，その作業の自動化はされていない．AEGIS では，英文を対象とした出題が可能だが，本研究ではプログラミング言語を対象とする．

3　誤り訂正問題の分析

3.1　誤りの分類

プログラミング課題の，過去の学習者の解答における誤りを分類すると，次のような種別に分類される．

(1) 字句 1 語の誤り

例 printf，scanf の’f’ 忘れ
’&’ と’&&’，’%’ と’%%’ の混同
エスケープ文字の’\’ の入力ミス
– ’\’ を入力するとき’\\’ と書かれていない
文字列の’\0’ や break 文の入れ忘れ
条件の誤り
– ’&&’ と’||’，’==’ と’!=’，不等号の混同
– 変数の誤り

(2) 文法の誤り

例 scanf の’&’ 忘れ

else if 文の else 忘れ

’=’ と’==’ の混同

(3) 意味の誤り

例 初期化忘れ，配列の範囲の誤り

繰り返し文の継続回数および条件の誤り

字句，文法の誤りを設けるときには対象とする字句，文法および，そこに用いる誤りを形式化する．意味の誤りは複数の正答をもつ可能性が他の誤りより高いので，誤りを設ける前後の字句の形式化に加えて，解答範囲の制限，制御文のループ回数の検査などをおこなう．

図 1 は分類した誤りのうち，意味の誤りの例を示したものである．

1. 初期化忘れ

```
#include <stdio.h>

int main(void)   □:誤りを設ける箇所
{
  int i, x, n, ans;
  □
  printf("x? ");
  scanf("%d", &x);
  printf("n? ");
  scanf("%d", &n);

  for(i = 1; i <= n; i++){
    ans = ans * x;
  }
  printf("%d ^ %d = %d¥n", x, n, ans);

  return 0;
}
```

2. 配列の範囲

```
#include <stdio.h>

int main(void)   □:誤りを設ける箇所
{
  int i, n, array[5];
  for(i = □1; i <□= 5; i++){
    printf("array[%d]? ", i);
    scanf("%d", &array[i]);
  }
  printf("何番目の要素を表示しますか? ", n);
  if(n >□= 0 && n <□= 5){
    scanf("%d", n);
    printf("array[%d] = %d¥n", n, array[n]);
  }else{
    printf("不正な値です¥n");
  }
  return 0;
}
```

図 1 意味の誤りの例

これらの誤りを正答のパターン別に分類すると以下のようになる．

(a) 別解のない誤り

- printf 文，scanf 文の’f’ の入力漏れ
- scanf 文の引数の’&’ の入れ忘れ
- else if 文の else 忘れ
- ’&’ と’&&’，’=’ と’==’，’%’ と’%%’ の混同
- エスケープ文字’\’ の入力ミス
- 条件の誤り

(b) 複数解答のある誤り

- 繰り返し文の継続回数，開始条件と終了条件
- 配列の範囲

(c) 正答の字句は 1 通りでも正答となる位置が複数通りある誤り

- 初期化忘れ
- 文字列の終端記号’\0’ や break 文の入れ忘れ

3.2 編集可能箇所

(a)，(b) については該当字句，および誤りを設けた字句に一致する場所など，それらの字句に類似した字句の箇所を編集可能とする．(c) については正答の他，いくつかの文の前後を編集可能とする．なお，(c) において行単位で編集可能箇所を設けたい場合は，基のソースコードに存在しなかった，編集可能な空行を新たに設ける．

図 2 は (a) における scanf 文の’&’ 忘れの誤りを含むソースコードで，類似した字句の例を示す．scanf 文と printf 文はそれぞれ引数をもち，scanf 文の引数から’&’ を削除したものと printf 文の引数は形式が類似しているので，編集可能箇所とする．

設問の例

```
#include <stdio.h>

int main(void)   □:類似するパターンをもつ字句
{
  int i, x, n, ans;
  ans = 1;
  printf("x? ");
  scanf("%d", □x);
  printf("n? ");
  scanf("%d", □n);

  for(i = 1; i <= n; i++){
    ans = ans * x;
  }
  printf("%d ^ %d = %d¥n",□x,□n,□ans);

  return 0;
}
```

図 2 類似した字句の例

図 3 は (b) における for 文の開始，終了条件のときの正答の制限例で，for 文における最初の編集可能箇所を初期値に限定し，継続条件の’n’ を固定した．図 4 は (c) における空行の追加例である．

1. 正答の制限前

```
#include <stdio.h>

int main(void)   □:編集可能とする場所の候補
{                ⬚:対象とする設問
  int i, x, n, ans;

  printf("x? ");
  scanf("%d", &x);
  printf("n? ");
  scanf("%d", &n);
  ans = 1;
  for(i □= 1; i □<= n; i++){
    ans = ans * x;
  }
  printf("%d ^ %d = %d¥n", x, n, ans);

  return 0;
}
```

2. 正答の制限後

```
#include <stdio.h>

int main(void)   □:編集可能とした箇所
{                ⬚:対象とする設問
  int i, x, n, ans;

  printf("x? ");
  scanf("%d", &x);
  printf("n? ");                  制限した部分
  scanf("%d", &n);
  ans = 1;
  for(i □= 1; i □<= n; i++){
    ans = ans * x;
  }
  printf("%d ^ %d = %d¥n", x, n, ans);

  return 0;
}
```

図 3 正答を制限する例

3.3 複数解答

(b) については複数解答がある誤りの場合，複数解答の位置が同一ならば図 5 で行っているように，適用するパターンに複数ある解答を出題者が予め記述することで対応できる．

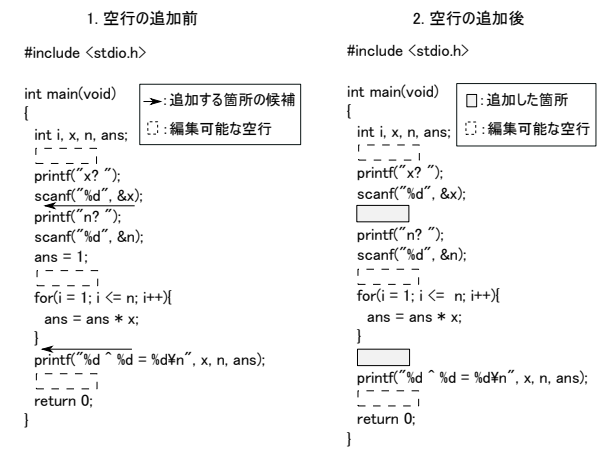

図 4 編集可能な空行の追加例

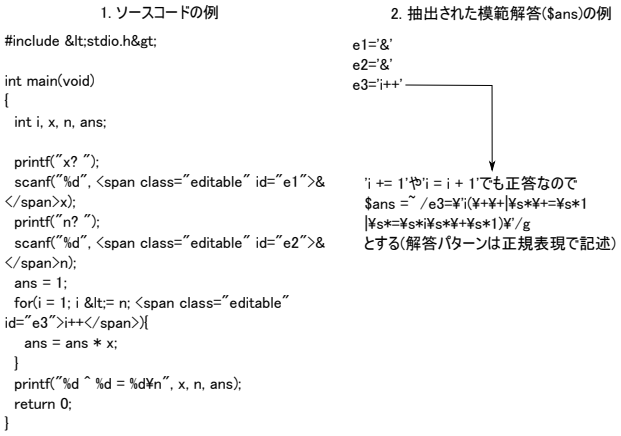

図 5 複数解答における正誤判定の例

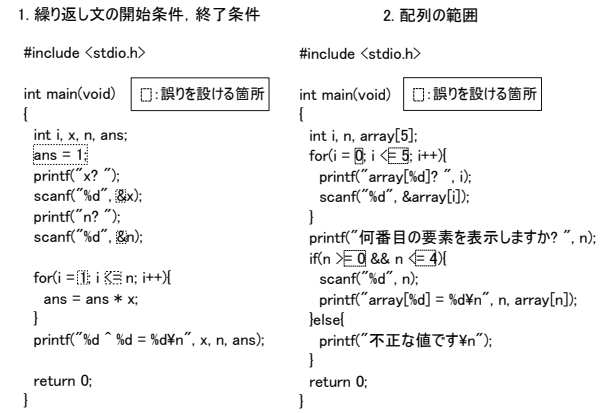

図 6 複数解答の例

図 6 は繰り返し文における開始条件および継続条件，配列の範囲をあつかう．図 6 の繰り返し文における 1. の開始条件と終了条件の例では，'(i = 1; i <= n;' でも，'(i = 0; i < n;' でも正答となる．2. の for 文も同様である．2. の配列の範囲は if 文中が，'n >= 0' か'n > -1' と'n <= 4' か'n < 5' の組み合わせなら正答となる．

図 7 は正誤判定の一例である．設問から学習者が編集した箇所の字句と，そこに対応する模範解答中の字句を比較する．その結果を基に，for 文のループ回数のように異なる字句が検出されても正答と判定されるように，ループ回数など適切な処理をもちいて判定する．

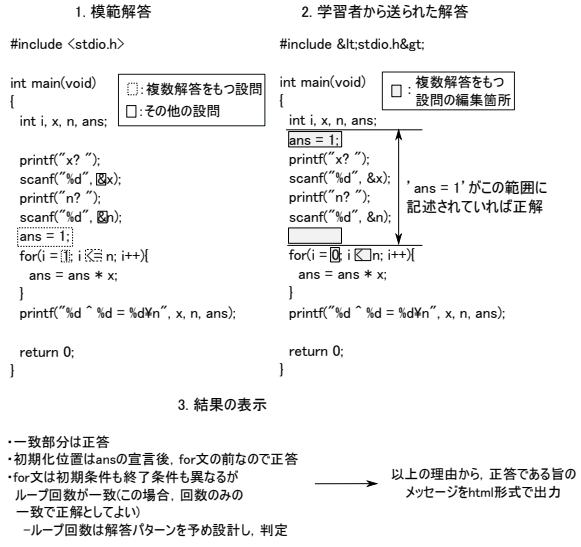

図 7 正誤判定の実行の概略

4 誤り訂正問題の自動生成方法

この章では，本研究で誤り訂正問題をどのように生成するかを扱う．問題作成に必要な誤りを生成するパターンの作成，字句書き換えによる誤り訂正問題の生成，解答および正誤判定の各部分について説明する．本研究では，字句の書き換えおよび構文解析における，字句書き換えのパターン化を目的とし，TEBA をもちいる．誤りおよび編集可能箇所の設定は字句書き換えパターンで記述する．

4.1 誤り訂正問題の出題システム

本研究では perl，TEBA[3] および CGI をもちいて，誤り訂正問題の生成および正誤判定を行うシステムを実現する．出題者は模範解答となるソースコードを作成し，問題作成ツールを通して誤りを含むソースコードを作成する．ツールでは教員が，設けたい誤りを設定するパターンを選択し，そのパターンを用いてソースコードの書き換えを行う．書き換え後のソースコードに教員が誤りに対して，編集可能化など任意の操作を行った後，html 化することにより，問題を作成する．

解答者は設問となる HTML ファイルを Web ブラウザで表示させ，誤りを訂正したものを CGI に送信する．CGI は解答者が送信した設問字句と模範解答となる対応部分を比較して正誤判定をおこなう．解答環境で利用する技術

として，Web ブラウザ上のテキストを編集する JEIP[4] がある．JEIP とは，前もって span タグで囲まれたテキストを編集可能とする JavaScript プログラムである．

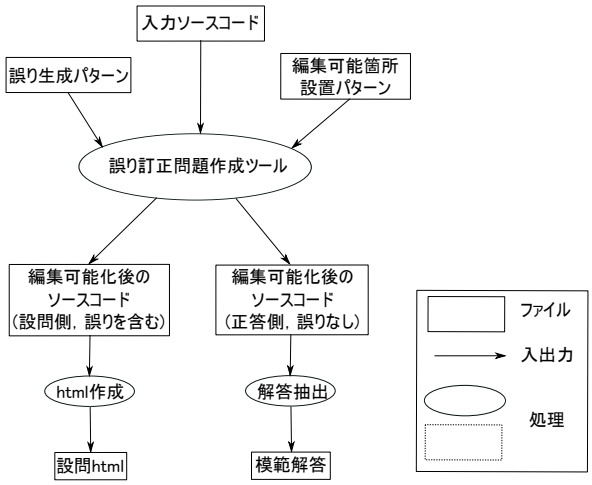

図 8 誤り訂正問題作成の概略

4.2 誤りを生成するパターンの作成

ここではパターンファイルの追加手順を説明する．ソースコード中にある誤りを設けたい字句系列と，どのように誤りを設けたいかという字句系列，そして誤りを説明する文章を入力し，ファイルとして保存する．設ける誤りの内容を指定する字句系列に，編集可能箇所をどのように設けるか指定する字句を記述する．

図 9 は，誤り生成パターンの例である．図 9 のような形式で，説明文をコメントとし，検索字句，書き換え後の字句を本文とする．なお，図中の`<@eff-` と `@ect>`で囲われた部分が編集可能箇所となる．

1. 編集可能箇所設定パターンの例

print_tag.pt
```
// explanation
// printf文をprint～とする誤り
% before
printf(${st:STR}, ${id:ID_VAR});
% after
<@eff- printf ect@>(${st}, ${id});
% end
```

and_tag.pt
```
// explanation
// scanf文の引数で'&'を忘れる誤り
% before
scanf(${st:STMT}, &${id:ID_VAR});
% after
scanf(${st}, <@eff- & ect@>${id});
% end
```

2. 誤り作成パターンの例

err_print.pt
```
// explanation
// printf文をprint～とする誤り
% before
<@eff- printf ect@>(${st:STR}, ${id:ID_VAR});
% after
<@eff- print ect@>(${st}, ${id});
% end
```

rmv_and.pt
```
// explanation
// scanf文の引数で'&'を忘れる誤り
% before
scanf(${st:STMT}, <@eff- & ect@>${id:ID_VAR});
% after
scanf(${st}, <@eff-  ect@>${id});
% end
```

図 9 誤り訂正問題作成パターンの例

4.3 解答および正誤判定

解答環境には JEIP を利用し，学習者は JEIP により編集可能とした箇所を書き換えることによって解答をおこなう．正誤判定は模範解答と学習者の解答，それぞれのソースコードから解答字句を抽出し，それらを比較することでおこなう．

5 考察

5.1 評価

本研究では 3.1 節で分類した誤りを対象とする字句書き換えパターンを設計した．このうち，変数誤り以外には対応できる．変数誤りは適切な変数がパターンとして判断できないので対応できない．

5.2 考察

条件誤りにおける変数の誤りについては，ソースコードごとに個別にパターンを作成することで対応可能である．字句や構文の解析によって，ある程度他のソースコードに適用可能なパターンを記述できると考えられる．

本研究で作成した誤り訂正問題作成ツールは，誤りおよび編集可能箇所を設定するパターンにより設問の作成をおこなう．これらの字句書き換えパターンを記述することで，新たな設問の作成が可能となる．3.1 節の (b) のように複数の解答方法がある場合は正誤判定する際の，解答パターンも記述することによって新たな設問に対応できる．

6 おわりに

本研究は，教員の誤り訂正問題を作成する際の教員にかかる労力の減少を目的とし，C 言語における誤り訂正問題の生成および正誤判定の自動化と，その問題に利用する解答環境の提案を行った．

手法としては，まず模範解答となるソースコードを誤り生成パターンを利用して書き換え，学習者側で一部編集可能な Web ページとしてサーバにアップロードする．このページを学習者が編集し解答した後，その内容となるソースコードから span タグで囲まれた部分を抽出する．模範解答から同様にして抽出した部分と，学習者による解答を比較する．この結果から正答となる差分を検出して除外し，誤りの有無により正誤判定をおこない学習者に結果を表示する．

今後の課題として，過去の演習で蓄積された誤りを含むソースコードからの誤りパターンの生成が挙げられる．

参考文献

[1] Martin Dougiamas, “*Moodle*^{TM}, ”http://moodle.org/, 1999.

[2] 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，vol.2003, no.11，pp.25-32，Apr. 2003.

[3] 吉田 敦，蜂巣吉成，沢田篤史，張 漢明，野呂昌満：属性付き字句系列に基づくプログラム書換え支援環境，情報処理学会論文誌 Vol.53 No.7，pp.1832-1849, 2012.

[4] Joseph Scott, “*jQuery Edit In Place (JEIP)*, ” http://josephscott.org/code/javascript/jquery-edit-in-place/，2008.