N-033

プログラミング演習における模範解答プログラムを用いた演習問題の自動生成
Automatic Programming Exercise Generation from A Model Answer Program

戸谷剛士* 小川 愛弥† 蜂巣 吉成‡ 吉田 敦 § 桑原 寛明¶
Tsuyoshi Toya Manaya Ogawa Yoshinari Hachisu Atsushi Yoshida Hiroaki Kuwabara

1. はじめに

プログラミング学習においてプログラムを作成する演習問題を解くことは必要不可欠である．学習者は演習問題を一度解くだけではなく，多くの問題を解き，様々なプログラムに触れてプログラミング言語の概念や文法などを学んでいく．多くの問題を解くためには多くの問題が必要である．

一般に演習問題は，問題文と模範解答プログラムで構成されるが，問題作成においてそれらの整合性を取ることが手間である．教育者は既存の演習問題を書き換えて新たな問題を作成することで，問題作成の負担を軽減している．しかし，既存の演習問題を書き換えて新たな問題を作成するときに，書き換え忘れがあると整合性を保つことはできない．

本研究は，模範解答プログラムから演習問題の問題文を生成する方法を提案する．問題文に必要な入出力の内容，制約を模範解答プログラムから抽出することで，問題文を生成する．入力のプロンプトや結果出力を示す printf，scanf 文に記述ルールを設けることで問題文生成に必要な情報の取得を行う．

2. 関連研究

長野，寺本ら[1]の研究では，C 言語学習における条件分岐と繰り返し入力を対象とした演習問題のテンプレートを用いた自動生成方法を提案している．テンプレートファイルと書き換え記述ファイルを用いて，模範解答プログラム，問題例(問題文と実行例)を生成している．本研究では入力以外の繰り返しや配列を用いた計算などのテンプレートが作成できない問題を対象にしている．また，テンプレートファイルや XML のタグを模した形式を用いず，模範解答プログラムや問題文に文章として自然な記述方法を提案する．

喜多村ら[2]の研究では，Java の問題文,配布プログラム，解答検査プログラムを 1 つのファイルから生成する手法を提案している．この研究では Java 言語によるプログラミング問題を対象として問題の解答例プログラムに，XML タグのような形式で情報付与を行い，このタグ付き解答例ソースから 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と問題文が生成される．Java プログラムにおいて，プログラムとコメントで記述された問題文の整合性をとる必要がある．本研究では XML のタグを模した形式を用いず，模範解答プログラムから問題文を生成する方法を提案する.プログラムの入出力文などから問題文に必要な情報を抽出するので，整合性を保ちやすい．

* 南山大学 Nanzan University
† 株式会社ビーネックスソリューションズ BeNEXT Solutions Inc.
‡ 南山大学 Nanzan University
§ 南山大学 Nanzan University
¶ 南山大学 Nanzan University

3. 問題分析

模範解答プログラムは，一般に入力，計算，出力の 3 つで構成されていることが多い．問題文を作成する上で必要な入出力内容，その制約，プログラムの作成方法を見つけ，模範解答から抽出する内容を決めるために問題分析を行う．南山大学理工学部の 1 年生向け授業科目「プログラミング応用」の演習問題を参考に，入力，出力，計算の 3 つについて問題分析を行った．演習問題の問題数は 133 問(ソースコードを打ち込む問題，文法のルールを確認する問題は除く)ある．

3.1 対象とする問題

繰り返し，配列，関数，文字(列)の単元で，作成するプログラムがターミナル上で対話的に実行する問題を対象にする．すなわち，入力を促すプロンプトが存在し，結果表示に計算方法が明記されている問題である．問題文と実行結果の例をソースコード 1，ソースコード 2 に示す．

ソースコード 1 対象問題の問題文例

```
整数 x,整数 y を入力し、平均を出力するプログラムの作成
```

ソースコード 2 対象問題の実行結果例

```
整数 x? 10
整数 y? 6
平均:8
```

3.2 対象外とする問題

入出力がない問題，出力結果が図や表形式になる問題は，対象外とする.多次元配列は演習問題で扱っている問題数が少ないので対象外とする．条件分岐は長野，寺本ら[1]の研究で問題を生成できるので対象外とする．ファイル処理の問題は入力が標準入力ではない問題が多いので対象外とする．構造体，ポインタの単元は，問題文にあるプログラムを作成するときの制約として書かれていることが多いので，対象外とする．構造体を扱う方法は 5.4 節で考察する．

3.3 問題分析の結果

エラー処理を行う演習問題は 17 問(13%)あった．計算結果を出力する演習問題は 80 問(60%)あった．出力桁数を用いる演習問題は 38 問(29%)あった．繰り返し出力を行う演習問題は 40 問(30%)あった．関数を作成する演習問題は 90 問(68%)あった．配列を用いる演習問題は 47 問(35%)あった．対象となる演習問題は 48 問(36%)あり，対象外となる演習問題は，85 問(64%)あった．

4. 演習問題自動生成方法の提案

3 節の分析結果よりプログラミング演習の問題文はプログラムの入出力を明記する本文とプログラムの入出力の制

約や作成方法を明記する制約文で構成することができる．本研究では模範解答プログラムをツールに与え，そのプログラムから本文と制約文に必要な情報を抽出し，それぞれツールに事前に用意した本文と制約文の雛形に挿入する．この方法では，ツール利用者が用意するファイルは模範解答プログラムのみでよい．問題の妥当性を確認するためには模範解答プログラムを作成し，実行結果を確認する必要がある．動作確認したプログラムから情報を抽出するので，問題文の整合性を保ちやすい．任意のプログラムから問題作成に必要な情報を抽出することは難しいので，模範解答プログラムを作成する際の記述ルールを作成し，情報の抽出を図る．本文は入出力パートと計算方法パートで構成し，制約文は入出力，作成方法で構成した．

4.1 本文と入出力の記述ルール

本文は演習問題において学習者が作成すべきプログラムの入出力について説明するための文である．この文はあらかじめツールが用意した雛形に教育者が作成した模範解答プログラムから抽出した入出力の情報を挿入することで生成する．模範解答プログラムから抽出する必要がある情報は入出力とそれらの型である．これらの情報について，入力はプログラム実行時にターミナル上に表示されるプロンプトを出力する printf 文から参照できる．同様に，出力は計算結果を出力する printf 文から参照できる．そして，それらの型はそれぞれのフォーマット指定子を参照することで補完できる．以上を用いて模範解答プログラムの入出力の記述ルールはソースコード 3，ソースコード 4 形で記述する．

ソースコード 3 入力の記述ルール

```
printf("入力?");
scanf("フォーマット指定子",&変数名);
```

ソースコード 4 出力の記述ルール

```
printf("出力:フォーマット指定子\n", 変数名);
```

入力は入力を示す printf 文内の入力の末尾に?を記述することで入力を識別し，次の行の scanf 文内の変換指定子から型を補完する．出力は出力を示す printf 文内の出力の末尾に:を記述することで出力を識別し，その printf 文内のフォーマット指定子から型を補完する．抽出したそれらの情報はツールに格納する．これらの抽出した情報をあらかじめツールが用意する雛形に挿入する際にそれぞれの情報を挿入する箇所の目印が必要である．そこで次に示す雛形を用いる．

・()を入力し、「」を出力するプログラムの作成.

この雛形では入力を挿入する箇所には()を記述し，出力を挿入する箇所には「」を記述する．挿入の際，目印として用いた()と「」は削除する．

これらの記述ルールと雛形を用いて生成できる問題をソースコード 5 ソースコード 6 に示す.

ソースコード 5
生成できる問題(模範解答プログラム)

```
#include<stdio.h>
int main(void)
{
    int sum;
    int jpn,math,eng;
    double ave;

    printf("国語の点数?");
    scanf("%d",&jpn);
    printf("数学の点数?");
    scanf("%d", &math);
    printf("英語の点数?");
    scanf("%d", &eng);

    sum = jpn + math + eng;

    ave = sum / 3.0;

    printf("全科目の平均:%f\n", ave);
    return 0;
}
```

ソースコード 6 生成できる問題(問題文)

国語の点数(整数),数学の点数(整数),英語の点数(整数)を入力し，全科目の平均(実数)を出力するプログラムの作成.

4.2 入力の制約

入力の制約とは，入力値の範囲と入力の繰り返しである．入力値の範囲は次のような形である．

・入力は 100 以下である.

これについては 5.3 節で考察する．入力の繰り返しが必要な配列は 4.5 節で説明する.

4.3 出力の制約

出力の制約とは，出力桁数についてと出力の繰り返しである．出力桁数は次のような形である.

・出力は小数点第一位まで出力する.

これをプログラムに実装するためには，ソースコード 7 のようにフォーマット指定子を変更する必要がある.

ソースコード 7 出力桁数の記述ルール

```
printf("出力:%.1f\n",変数名);
```

これより，模範解答プログラムのフォーマット指定子を参照することで出力桁数を取得できる．このとき，ツールは出力の printf 文であることのみ識別すれば良いので，新たに記述ルールを用意する必要がない．出力の繰り返しは配列の要素や'A'から'Z'までなどの連続した文字コードを出力する際に用いられる．このときの配列の要素とは配列の要素とその説明を同時に出力するものが該当する．これについては 5.3 節にて考察する.

4.4 配列の扱い

入力の繰り返しに配列を用いる場合がある．また，その配列の入力回数についても固定のものとプログラム実行時に入力する 2 通りある.

4.4.1 配列の記述ルール

配列の入力個数が固定のものについてはソースコード 8 のように記述する.

ソースコード 8 配列の記述ルール(固定)

```
for (添字 = 0; 添字 <入力回数; 添字++) {
   printf("配列の説明[%d]?", 式);
   scanf("フォーマット指定子", & 配列名[添字]);
}
```

この記述ルールでは for 文，printf 文，scanf 文を連続して記述した際に入力の繰り返しであることを識別する．次に配列の入力関数がプログラム実行時に入力するものについてはソースコード 9 のように記述する．実行時に入力するプログラムでは入力の繰り返しを行う for 文の直前に入力回数を入力するための printf 文と scanf 文を記述する．この記述ルールでは先ほどの記述ルールに加えて，scanf 文の有無で入力の繰り返しがプログラム実行時入力であることを識別する．

ソースコード 9 配列の記述ルール(実行時入力)

```
printf(データ数の説明);
scanf(フォーマット指定子, &入力回数変数);
for (添字 = 0; 添字 < 入力回変数数; 添字++) {
   printf("配列の説明[%d]?", 式);
   scanf("フォーマット指定子", & 配列名[添字]);
}
```

4.4.2 本文の雛形

入力回数が固定である場合の雛形は次の形である．

・()を N 回入力し，「」を出力するプログラムの作成.

そして，N を入力回数を置き換え，目印として用いた()と「」は削除する．入力回数が実行時入力である場合の雛形は次の形である．

・データ数を入力後，()をデータ数の分だけ繰り返し入力し，「」を出力するプログラムの作成.

4.5 作成方法の制約

作成方法の制約とは，作成する関数が指定されることである．すなわち作成する関数の関数名，引数、返り値を模範解答プログラムから抽出する必要がある．これについては 4.6 節で説明する．

4.6 関数の扱い

指定した関数を作る問題では模範解答プログラムから関数の関数名，関数の説明，引数，引数の説明，返り値の説明を抽出する必要がある．しかし，関数の説明や引数の説明，返り値の説明はプログラム上には出現しない．よって，コメントを用いて情報を付与することで模範解答プログラムから情報を抽出する．

本研究では javadoc のコメント記述方法を参考にソースコード 10 のような記述ルールを作成した．

ソースコード 10 関数の記述ルール

```
/**
* 関数の説明
* 引数 変数名 1 引数の説明 1
* 引数 変数名 2 引数の説明 2
* 返り値 返り値の説明
*/
型 関数名(型 1 変数名 1, 型 2 変数名 2)
{
   ～処理～
   return 返り値;
}
```

関数の仕様のコメントは関数宣言の直前に記述する．"/**"は関数の仕様のコメントの開始を示し，”*/”でコメントの終了を示す．関数の仕様のコメントは関数の説明，引数，返り値の順番で記述する．また，これらの記述ルールと別に関数宣言時と関数内の return から情報を抽出する方法を検討した．この方法では模範解答プログラムの記述は容易であるが，先述した関数の説明や引数の説明，返り値の説明を取得することができないので採用しなかった．

4.7 情報の抽出順序

本文や制約文の記述ルールには重複する箇所があるので，次の順序で情報を適切に抽出する．

1. 入力の繰り返し(実行時入力)
2. 入力の繰り返し(固定)
3. 本文の入出力
4. 出力桁数
5. 作成する関数

4.8 ツールの実現

問題生成ツールを Perl と属性付き字句系列に基づく書換え処理系である TEBA[3]を用いて実現した。配列の記述ルールは TEBA のパターンで記述した。ソースコード 9 の記述ルールはソースコード 11 のように記述できる．

ソースコード 11
TEBA のパターン記述(実行時入力)

```
printf(${:EXPR});
scanf(${scstrn:EXPR}, &${n:ID_VF});
for (${i:EXPR} = 0; ${i} < ${n}; ${:EXPR}) {
  printf(${prstr:EXPR}, ${:EXPR});
  scanf(${scstr:EXPR}, &${ar:ID_VF}[${i}]);
}
```

${scstrn:EXPR} は式に，${n:ID_VF}は識別子にマッチし，マッチしたトークンを取得できる．

4.9 生成結果例

定義した記述ルールを用いて作成した模範解答プログラムソースコード 12，ソースコード 14 から生成できる問題文をソースコード 13，ソースコード 15 に示す．

ソースコード 12
模範解答プログラム(入力回数固定)

```
#include <stdio.h>

int main(void)
{
  int a[10];
  int sum, i;


  for (i=0;i<10;i++) {
   printf("配列[%d]? ", i);
```

```
        scanf("%d", &a[i]);
    }

    sum = 0;
    for (i=0;i<10;i++) {
        sum += a[i];
    }
    printf("合計: %d\n", sum);

    return 0;
}
```

ソースコード 13 問題文(入力回数固定)

```
問題
配列(整数)を 10 回入力し，合計(整数)を出力するプログラムの作成.
```

ソースコード 14
模範解答プログラム(入力回数実行時入力)

```
#include <stdio.h>
/**
* 要素数 size の実数配列 a の合計を求める
* 引数 a 実数配列
* 引数 size 配列の大きさ
* 返り値 平均
*/
double avr_array(double a[], int size)
{
    double sum = 0;
    int i;
    for (i = 0; i < size; i++){
        sum += a[i];
    return sum/size;
}

int main(void)
{
    int num, i;
    double heights[128], avr;

    printf("人数? ");
    scanf("%d", &num);
    for (i = 0; i < num; i++) {
        printf("身長(cm)[%d]? ", i);
        scanf("%lf", &heights[i]);
    }
    avr = avr_array(heights, num);
    printf("平均： %.1f\n", avr);
    return 0;
}
```

ソースコード 15 問題文(入力回数実行時入力)

```
問題
データ数を入力後，身長(cm)(実数)をデータ数の分だけ繰り返し入力し，平均(実数)を出力するプログラムの作成.
制約
double avr_array(double a[], int size)
要素数 size の実数配列 a の合計を求める
引数 a 実数配列
引数 size 配列の大きさ
返り値 平均

平均は小数点以下第 1 位まで出力する
```

4.10 評価

3 節で対象とした演習問題 48 問のうち，22 問(約 46%)が生成できた．生成できなかった問題は次の 4 つであった．

1. 出力の繰り返しを扱う問題
2. 標準入力を扱う問題
3. マクロを扱う問題
4. 計算内容を簡潔に説明することが難しい問題

これらについては 5 節にて考察する．

5. 考察

5.1 必要な制約

問題作成において模範解答プログラムから情報を抽出するための記述ルールは大きく分けて次の 2 つである．

・特定の処理をパターンに基づいて記述する方法
・コメントを用いて情報を付与する方法

基本的には 1 つ目の方法で対応し，扱えない場合のみ 2 つ目の方法で対応する．

5.2 生成できなかった問題

出力の繰り返し，標準入力を扱う問題は特定の処理をパターンに基づいて記述する方法で対応することができる．マクロを扱う問題はマクロの値を取得する必要がある．よって，マクロの定義をパターンとして処理することで対応することができる．計算内容を簡潔に説明することが難しい問題は本文の雛形に情報を挿入する型では対応することができず，その計算内容ごとに雛形を用意する必要がある．よって，main 関数にコメントを用いて情報を付与することで対応する．

5.3 パターンの拡張

4.5 節では，典型的な配列の入力処理をパターンで表し，パターンにマッチした模範解答プログラムから問題生成に必要な情報を抽出した．この方法を一般化することで，典型的な処理をパターンで表して，必要な情報を抽出して問題生成できると考える．例として

・標準入力
・マクロ定義
・エラー処理

について考える．

標準入力から 1 文字ずつ入力する処理は，ソースコード 16 のようにパターン化し，問題文の雛形を

・標準入力から 1 文字ずつ入力し，「」を出力するプログラムの作成.

として問題生成が可能であり，同様に出力の繰り返しについてもパターンで記述可能である.

ソースコード 16 文字の入力

```
while ((変数 = getchar()) != EOF) 文
```

標準入力を扱う問題の例をソースコード 17，ソースコード 18 に示す.

ソースコード 17 標準入力(模範解答プログラム)

```
int main(void)
{
  int ch, cnt;
  cnt = 0;
  while ((ch=getchar())!=EOF) {
    if ('a'<=ch&& ch<='z' || 'A' <=ch && ch<='Z')
      cnt++;
  }
  printf("アルファベットの個数: %d\n", cnt);
  return 0;
}
```

ソースコード 18 標準入力(問題文)

```
問題
標準入力から 1 文字ずつ入力し，アルファベットの個数(整数)を出力するプログラムの作成.
```

マクロについては，ソースコード 19 のようにマクロ定義時の#define を識別し，マクロ名とその値を抽出する．その後プログラム内のマクロを対応する値に置換することで対応する.

ソースコード 19 マクロ定義

```
#define マクロ名 データ値
```

エラー処理は，ソースコード 20 のように一般に if 文と変数が取りえない範囲，エラーメッセージを出力する fprintf 文，プログラムを終了させる exit 文で構成されるので，これらをパターン化することで抽出可能である.

ソースコード 20 エラー処理

```
if(入力変数 < 入力下限 || 入力上限 < 入力変数){
   fprintf(stderr , "不正なデータです.\n");
   exit(1);
}
```

5.4 構造体の扱い

構造体を扱う上で必要な情報として構造体名，その構造体の説明，メンバとその説明が必要となる．しかし，構造体の説明とメンバの説明はプログラム上に出現しない．すなわち，関数と同様にコメントを用いて与える必要がある．ソースコード 21 のような形の記述ルールを提案する.

ソースコード 21 構造体の記述ルール

```
/***
** 構造体の説明
** メンバ 型 1 メンバ 1 メンバ 1 の説明
**/
struct 構造体名 {
  型 1 メンバ 1
};
```

'*'を増やすことで関数の記述ルールとの差別化を図りつつ，類似した記述ルールにすることで新たに記述ルールを覚える手間を軽減が見込める．だが，この記述ルールでは typedef 句を用いた構造体宣言には対応していないので，考察の余地がある.

5.5 計算内容を簡潔に説明することが難しい問題

計算内容の説明が難しい問題は雛形に情報を挿入する方法では生成することが出来ない．ソースコード 22 のように main 関数にコメントで問題文の説明と実行時の入力例を記述することで問題を生成できると考察した．問題文の説明には問題文を記述し，実行時の入力例をコンパイルしたプログラムに与えて実行例を生成する.

ソースコード 22 実行例の記述ルール

```
/**
 * 問題文の説明
 * 実行例 1  実行時の入力例 1
 * 実行例 2  実行時の入力例 2
 */
int main(void){
  ～処理～
}
```

また，ソースコード 23，ソースコード 24 にある BIM のように一般的でなく計算方法がわからない可能性がある問題には計算方法を示す必要がある.

ソースコード 23 BIM 計算(模範解答プログラム)

```
#include<stdio.h>

int main(void)
{
 double height,weight,bmi;

 printf("身長?");
 scanf("%lf",&height);
 printf("体重?");
 scanf("%lf",&weight);

 bmi = weight / (height * height) * 10000;

 printf("BMI:%f\n",bmi);
 return 0;
}
```

ソースコード 24 BIM 計算(問題文)

```
問題
身長(実数),体重(実数)を入力し，BMI(実数)を出力するプログラムの作成.
```

ソースコード 25 のように main 関数にコメントで出力とその計算方法を記述することで計算方法を示すことができると考察した．

ソースコード 25 計算方法の記述例

```
  /**
   * BMI = 体重 km ÷ (身長 m)^2
   * 実行例 1 64.3  175.6
   * 実行例 2 112.5  184.5
   */
  int main(void){
    〜処理〜
}
```

6. おわりに

本研究では模範解答プログラムを用いて，問題文生成に必要な情報の分析や分析結果を基に抽出する方法を提案し，評価した．今後の課題として 5 節で考察した内容の拡張や構造体を扱うための更なる考察，演習問題の HTML を生成することなどが挙げられる．

謝辞

本研究の一部は JSPS 科研費 23K11359，2023 年度南山大学パッヘ奨励金 I-A-2 の助成を受けた．

参考文献

[1] 長野 翔瑠，寺本 圭佑:C 言語学習における条件分岐と繰り返し入力を対象とした演習問題のテンプレートを用いた自動生成方法の提案，南山大学理工学部 2021 年度卒業論文 (2022)．http://www.st.nanzan-u.ac.jp/info/gr-thesis/2021/hachisu/pdf/18se054.pdf

[2] 喜多村和誠，玉木久夫: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，情報処理学会研究報告，Vol.2013-CE-122 Vol.2013-CLE-11，No.1(2013).

[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ソースコード書換え支援環境，情報処理学会論文誌，Vol.53，No.7，pp.1832-1849，2012.