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

2019SE044 小川愛弥　2019SE060 戸谷剛士

指導教員: 蜂巣吉成

1　はじめに

プログラミング学習において演習問題を解くことはプログラムを理解し作成，利用するために必要不可欠である．学習者は演習問題を一度解くだけではなく，その類似問題を繰り返し解き，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．類似問題を繰り返し解くためには多くの問題を作成する必要がある．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，問題作成において整合性を取ることが手間である．教育者は既存のプログラミング問題を書き換えて新たな演習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，既存のプログラミング問題を書き換えて新たな演習問題を作成するときに，書き換え忘れがあった場合など整合性を保つことは難しいという問題がある．

本研究は，模範解答プログラムから問題文を生成し，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．長野，寺本ら [1] が扱う事ができなかった問題も扱い，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．本研究では，長野，寺本ら [1] の課題である，テンプレートを作成できない問題が扱えないことやパラメータの記述内容によっては予期せぬエラーが発生し，そのエラーの対応が難しい課題を解消することができる．

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) を同時生成している．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．1 つのファイルに書くことによって，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を提案する．模範解答プログラムから問題文に必要な情報を抽出するため，整合性を保ちやすい．

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．2020 年度の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．プログラミング応用の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

繰返し，配列，関数，文字 (列) の単元で，標準入力から入力し，計算結果を出力する問題を対象とする．

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．多次元配列はプログラミング応用の演習問題，新明解の演習問題の配列単元で，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 5.3 節で考察する．

3.3　問題分析の結果

基本型を扱う問題は入力内容で，プログラミング応用は 130 問 (98%) あった．エラー処理を行う問題はプログラミング応用は 17 問 (13%) あった．計算結果を出力する問題数はプログラミング応用は 80 問 (60%) あった．出力桁数を用いる問題数はプログラミング応用は 38 問 (29%) あった．繰り返し出力を行う問題数はプログラミング応用は 40 問 (30%) あった．関数を作成する問題数はプログラミング応用は 90 問 (68%) あった．配列を用いる問題数は計算，作成方法で，プログラミング応用は 47 問 (35%) あった．対象となる問題は，プログラミング応用では 48 問 (36%) あり，対象外となる問題は，プログラミング応用では 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，それぞれツールが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．また，模範解答プログラムはその場でコンパイル実行でき，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作成する．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 1，2 の形で記述する．

ソースコード 1　入力の記述ルール

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

ソースコード 2　出力の記述ルール

```
1 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．

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．

5.1 節で考察する．入力の繰り返しが必要な配列は 4.5.1 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 3 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 3　出力桁数の記述ルール

```
1 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 5.2 節にて考察する．

4.4　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．

4.5　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．

4.5.1　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 4 のように記述する．

ソースコード 4　配列の記述ルール (固定)

```
1 for (添字 = 0; 添字 <入力回数; 添字 ++) {
2     printf("配列の説明 [%d]", 式);
3     scanf("フォーマット指定子", & 配列名 [添字]);
4 }
```

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 5 のように記述する．

ソースコード 5 配列の記述ルール (実行時入力)

```
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]);
}
```

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．

4.5.2 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.

4.6 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．

4.6.1 記述ルール

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 6 のような記述ルールを作成した．

ソースコード 6 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 作成結果例

配列の記述ルールを TEBA[3] のパターンで記述し，ソースコード 7 から作成できる問題文をソースコード 8 に示す．

ソースコード 7 模範解答プログラム

```
#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 8 問題文

```
データ数を入力後,身長 (cm)をデータ数の分だけ繰り返し
    入力し,平均を出力するプログラムの作成
制約
 出力は小数点以下第 1位まで出力する
 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
```

4.9 評価

生成可能な問題について分析を行ったが，本節では本研究で提案した方法で扱うことができる問題数について検証する．プログラミング応用演習問題において，扱うことができる問題数を表 1 に示す．ただし，プログラミング応用の第 12 回はファイル操作を扱うため割愛している．

表 1 生成可能な問題数

	生成可能な問題数		生成可能な問題数		生成可能な問題数		生成可能な問題数
第 1 回	8	第 2 回	1	第 3 回	2	第 4 回	0
第 5 回	4	第 6 回	0	第 7 回	5	第 8 回	1
第 9 回	0	第 10 回	0	第 11 回	0	第 13 回	0
第 14 回	1	第 15 回	0				
合計	22 問						

表 1 より扱うことができる問題数は 22 問となった．3 節で述べた対象となる問題数 48 問の約 46% となった．

5 考察

基本的な入出力の問題に加えて出力桁数や配列，関数を扱う方法を提案してきたが，制約文の拡張として入力値の範囲，パターンによる拡張，構造体の扱いが存在する．

5.1 入力値の範囲

入力値の範囲について，入力値のエラー処理を参照することで情報を抽出することができると考察した．入力値のエラー処理の例をソースコード 9，10 のような形である．

ソースコード 9 エラー処理 (例 1)

```
1 if (!(入力 <= 100))
2     printf("エラー\n");
```

ソースコード 10 エラー処理 (例 2)

```
1 if (!((0 < 入力) && (入力 < 100)))
2     printf("エラー\n");
```

否定を用いることで，if 文内の条件式と入力値の範囲が等しくなり，情報の抽出が行えると考察した．この拡張により単純な入力値の範囲だけでなく，配列の要素数で範囲が決定する問題などを対象にできることが見込まれる．

5.2 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．例えば，配列の全要素を出力する処理はソースコード 11 のようにパターン化できる．

ソースコード 11 繰り返し出力の記述ルール

```
1 for (i = 0; i < 入力回数; i++) {
2     printf("出力する配列の説明 [添字]:フォーマット
          指定子\n", 変数, 配列名 [添字]);
3 }
```

標準入力から 1 文字ずつ入力する処理は，ソースコード 12 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る．

ソースコード 12 文字の入力

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

5.3 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 13 のような形の記述ルールを提案する．

ソースコード 13 構造体の記述ルール

```
1 /***
2 ** 構造体の説明
3 ** @メンバ 型 1 メンバ 1 メンバ 1の説明
4 **/
5 struct 構造体名 {
6     型 1 メンバ 1
7 };
```

'*' 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．また，入出力の繰り返しについても入出力の種類が任意であるので，考察の余地がある．

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 を作成することなどが挙げられる．

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).

[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.