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
―問題文と模範解答プログラムと実行例の自動生成―

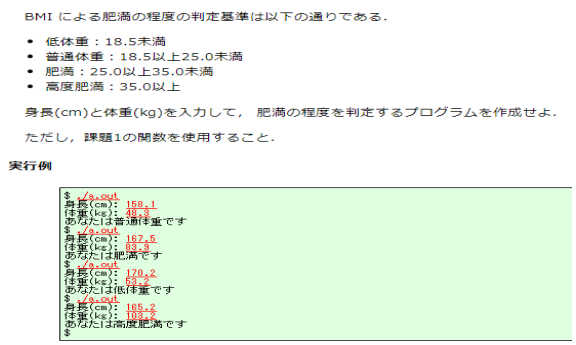
2018SE054 長野翔瑠　2018SE092 寺本圭佑
指導教員：蜂巣吉成

1　はじめに

プログラミング学習は，一度構文を学んでも，その一度きりでは完全に理解することは難しい．学習者は，構文を用いた類似問題を繰り返し解いていくことが必要である．学習意図に沿った類似問題を多く出題することが，プログラミング学習支援につながる．プログラミング学習に用いる演習問題は，図 1 のように問題文，実行例で構成されていることが多い．加えて問題作成者は模範解答プログラムを用意する必要がある．問題作成者は，これら 3 つの整合性を保ちながら問題を作成する必要があり労力がかかる．

本研究の目的は，問題作成者の負担軽減のために，整合性を保持した演習問題の自動生成方法を提案することである．整合性が保たれるというのは，問題作成において必要な要素間の問題の意味の一致と，問題作成において必要な要素内での構文エラーや記述ミスがない状態である．

教科書や演習問題の調査から，条件分岐や繰り返し入力の問題には類似問題が多く自動生成に適していることがわかった．複数の問題に共通な部分と異なる部分を整理した．共通部分をテンプレートファイル，異なる部分を書き換え記述ファイルとして定義した．テンプレートファイルには問題文，模範解答プログラムの共通部分と実行例の入力を記述し，書き換え記述ファイルには問題文と模範解答プログラムで整合性が取りやすいように記述をする．


BMI による肥満の程度の判定基準は以下の通りである．

- 低体重：18.5未満
- 普通体重：18.5以上25.0未満
- 肥満：25.0以上35.0未満
- 高度肥満：35.0以上

身長(cm)と体重(kg)を入力して，肥満の程度を判定するプログラムを作成せよ．

ただし，課題1の関数を使用すること．

実行例

```
$ ./a.out
身長(cm): 158.1
体重(kg): 48.3
あなたは普通体重です
$ ./a.out
身長(cm): 167.6
体重(kg): 69.9
あなたは肥満です
$ ./a.out
身長(cm): 170.2
体重(kg): 60.2
あなたは低体重です
$ ./a.out
身長(cm): 165.6
体重(kg): 103.2
あなたは高度肥満です
$
```

図 1　問題文と実行例[*1]

2　関連研究

堀井ら [1] では，問題テンプレートを用いたプログラミング学習における空欄補充問題の自動生成に関する研究がされている．この研究は，空欄補充問題の問題文とプログラム生成を目的とし，類似問題の共通部分をテンプレートとして考える手法が本研究と類似している．問題生成のために問題記述言語を提案しており，記述方法を覚えてプログラムを記述する必要があり手間がかかる．

喜多村ら [2] では，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に関する研究がされている．この研究は，Java を対象とした問題文，配布プログラム，解答検査プログラムを一つのファイルから生成する手法を提案している．本研究は，この研究の独自に定義したタグを用いて情報付与を行い，必要なプログラミング問題コンテンツを自動生成することを参考にしている．

掛下ら [3] はプログラムやトレース表に対する穴埋め問題を出題する，pgtracer という教育支援ツールを提案している．ツールを使用することで学習時間を短縮し，学習効率を高めることができる．この研究では穴埋め問題を生成しているが，本研究は C 言語の演習問題となる問題文，模範解答プログラム，実行例の同時生成を目的としている．

3　問題分析

3.1　問題作成における問題点

問題を書き換えて別種の問題を作成するとき問題文，模範解答プログラムを書き換え，プログラムを複数回実行する必要がある．実行例を表示する場合，実行結果をリダイレクトして自動で表示することもできるが，その方法では入力例を表示することができない．加えて，実行例の表示には入力例の文字を色付けする必要があり手間がかかる．この過程を問題文，模範解答プログラム，実行例の整合性を保ちながら行うことは手間のかかる作業である．

3.2　単元と項目

本研究では，様々な単元で用いることが多い条件分岐と繰り返し，型の変更を対象とした．条件分岐は if・if-else・多分岐・入れ子・switch・演算子，繰り返しは for・while・do-while・入れ子・繰り返し入力，型の単元では宣言・演算・入力・出力を学ばせたい．本研究は，柴田 [4] のプログラム例，本学理工学部の 2020 年度のプログラミング基礎，プログラミング応用の課題を参考に分析を行った．

3.3　問題数と分類

3.2 の項目の問題を分析した結果，模範解答プログラムの構造が似ている問題は，条件分岐は if，if-else，多分岐の問題が多く，繰り返しは繰り返し入力の問題が多く，それに関する問題は 127 問中 52 問 (41%) であった．

[*1] https://www-p.st.nanzan-u.ac.jp/classes/2020/50A09/02/

抽出した類似問題で行われる処理は，制御構文 (条件分岐・繰り返し) から作られている．抽出した問題でどのような変更があるのか分析し，次のようにまとめた．問題例の問題文は変更部分を最小限にするように工夫している．

1. 制御構造が同じで制御式が異なる問題
   二つの整数値で，(後者が前者の約数なら”B は A の約数”/差が 10 以下なら”差は 10 以下”) と表示し，それ以外は (”B は A の約数ではない”/”差は 11 以上”) と表示させる問題がある．模範解答プログラムは制御式と出力が変わり，それに応じて問題文の出力も変わる．
2. 条件分岐で分岐数や条件が異なる問題
   これは，判定内容次第で条件分岐の個数，条件式，出力が変わる．例として，(点数/月) を読み込み表のように判定させる問題がある．条件分岐の問題は，問題文に条件式と出力が対応した表を記述することで編集が容易となる．
3. 計算処理は同じで入力の方法が異なる問題
   問題分析から，繰り返し入力の学習として 4 つの例が挙げられる．例として，(10 個の/n 個の/Ctrl+D が押されるまで入力された/負の数が入力されるまで入力された) 値の和を求める問題がある．入力方法に関する模範解答プログラムの部分は，入力変数を読み取る scanf 関数のみ共通部分となっており，それに応じて問題文の出力も変わる．
4. 計算処理は同じで型が異なる問題
   これは，プログラム内で宣言を変えるだけでなく，入出力時のフォーマット指定子，関数の引数を考えたときの型まで変更する必要がある．それらを変更することで模範解答プログラムの生成が可能となる．
5. 出力が類似した問題 (図形を描く)
   これは，プログラムの制御構文の位置，個数が一致している問題が多く，制御式の書き換えで，別の図形を出力させることができる．

3.4 対象とする問題

本研究では 3.3 節の 1，2，3，4 の問題を扱う．1 と 2 の 2 つの問題群は変数名，if-else 構文の個数，if-else 構文内の処理を変えることで問題を生成する．3 は入力方法のみを変えることによって 4 種の問題を生成する．4 は変数の型，printf 関数や scanf 関数のフォーマット指定子を変えることで問題を生成する．5 は，図形の種類に限界があり，多くの問題を生成することが難しいので対象外とする．

4 テンプレートを用いた自動生成方法の提案

4.1 演習問題自動生成ツールの設計

3 章の分析結果から，同じ分類の中の問題で模範解答プログラムと問題掲示用 HTML ファイルの記述は，それぞれ共通した構造を持つことが分かった．問題を書き換えることで別の問題を生成する場合，共通する部分を書き換える必要はなく，その部分を誤って書き換えると，問題が成り立たなくなり整合性が保てなくなる．

文献 [2] では問題文，実行例，模範解答プログラムをすべて同一ファイルに記述することで記述忘れを起こしにくいようにしている．その方法を参考に，書き換え不要の共通箇所をテンプレート，変更箇所をパラメータとし，それぞれテンプレートファイル，書き換え記述ファイルにまとめる．書き換え記述ファイルにまとめられた最低限の情報から問題文，実行例，模範解答プログラムを生成するのでこの 3 つの整合性を保ちやすくなる．テンプレートファイルや書き換え記述ファイルに必要な情報を，XML のタグを模した形で記述する．タグ内のデータは共通のデータの対応で広く普及しており，表計算ソフトで編集が容易な CSV 形式を用いる．

入力ファイルはテンプレートファイルと書き換え記述ファイルである．書き換え記述ファイルは提案する記述方法でパラメータに必要な情報を記述する．演習問題自動生成ツールはこれらを入力とし，問題文と実行例が書かれた HTML ファイルと，ソースコードが書かれた模範解答プログラムを出力する．条件分岐の表は問題提示ファイルに記述される．問題文，実行例，表を HTML ファイルのテンプレートに当てはめて問題掲示用の HTML ファイルを作成する．

テンプレートファイルと書き換え記述ファイルに分け，問題の変更部分を 1 つのファイル内で行い演習問題を生成するように工夫することで整合性を保ちやすくした．

4.2 タグの種類

4.2.1 共通のタグ

条件分岐と繰り返し入力に関する問題の生成に用いる共通のタグは 3 つある．

<SENTENCE>タグは，開始タグ<SENTENCE>と終了タグ</SENTENCE>で囲まれた部分に問題文のテンプレートとパラメータを記述する．問題文は一部を変更するだけで成立するような問題文にする必要がある．

<CODE>タグは，模範解答プログラムを生成するためのタグである．開始タグ<CODE>と終了タグ</CODE>で囲まれた部分に模範解答プログラムのテンプレートと，書き換え記述ファイルで記述を補完するパラメータを記述する．

<INPUTEX>タグは，開始タグ<INPUTEX>と終了タグ</INPUTEX>で囲まれた部分に実行例の入力データを記述する．端末での実行例の表示を参考に 1 行ごとに 1 回の実行で必要な値を記述する．変数が複数の場合はカンマで区切る．実行例の 1 つの入力例が複数ある場合も同様に同じ情報が必要である．CSV 形式で記述する．

- 値 1, 値 2, 値 3

記述された値は実行に必要な入力として用いられる．

4.2.2 条件分岐のタグ

条件分岐に関する問題の生成に用いるタグは 3 つある．

<INPUTVARS>タグは書き換え記述ファイルに入力変数の情報を記述することで，問題文の生成，模範解答プログラムの入力変数に関わる部分を補完する．模範解答プログラムの変数宣言には型と変数名，printf 関数には変数の説明，scanf 関数には型と変数名が必要である．問題文には変数の説明が必要である．書き換え記述ファイルの<INPUTVARS>タグは，CSV 形式で次のように記述する．

- 型, 変数名, 説明

<CALCVARS>タグは書き換え記述ファイルにパラメータの情報を記述することで，問題文の生成，模範解答プログラムの計算変数に関わる部分を補完する．例として，3 教科の合計をもとに判定を行う問題のように合計の計算を格納するための計算変数を必要とするときこのタグを用いる．模範解答プログラムの変数宣言には型と変数名，代入式には変数名と計算式が必要である．問題文には変数の説明が必要である．書き換え記述ファイルの<CALCVARS>タグは，<INPUTVARS>タグと同様に CSV 形式を用いて，次のような形で記述する．

- 型, 変数名, 説明, 計算式

<BRANCH>タグは書き換え記述ファイルにパラメータの情報を記述することで，問題の表，模範解答プログラムの分岐に関わる部分を補完する．模範解答プログラムの分岐を補完するためには条件式と文の情報が必要である．問題の表には文と条件式の説明が必要である．書き換え記述ファイルの<BRANCH>タグは，CSV 形式を用いて，次のような形で記述する．

- 条件式, 文, 説明

条件式や文が複数ある場合，分岐数は増加していく．2 行目以降の条件式は else-if の構文，条件式に記述がない場合は else の構文が補完される．

他にも JSON を模した記述方法と YAML を模した記述方法を検討した．JSON を模した記述方法は，一つの情報がまとまっており見やすい利点もあるが，”[]”やダブルクォーテーションの記述が増えてしまう．YAML を模した記述方法は，それぞれの情報の対応が明確であるが，分岐の数が増加するにつれてソースコードが長くなる恐れがあり見づらくなる恐れがある．以上の理由から，表計算ソフトで編集が容易な CSV 形式で記述する方法に決定した．

条件分岐ではソースコード 1 とソースコード 2 のような記述方法を提案する．

ソースコード 1 テンプレートファイル (条件分岐)

```
1  <SENTENCE>
2  <INPUTVARS/>を読み込み,次の表のように<CALCVARS/>
       判定するプログラムを作成しなさい.
3  </SENTENCE>
4  <CODE>
5  #include<stdio.h>
6  int main(void){
7      <INPUTVARS/>
8      <CALCVARS/>
9      <BRANCH/>
10     return 0;
11 }
12 </CODE>
```

ソースコード 2 書き換え記述ファイル (条件分岐)

```
1  <INPUTVARS>
2  int,jap,国語
3  int,eng,英語
4  int,math,数学
5  </INPUTVARS>
6  <CALCVARS>
7  int,score,合計,jap+eng+math
8  </CALCVARS>
9  <BRANCH>
10 score>=270,A+,270以上 300以下
11 score>=240,A,240以上 270未満
12 score>=210,B,210以上 240未満
13 score>=180,C,180以上 210未満
14 ,不合格です。,180未満
15 </BRANCH>
16 <INPUTEX>
17 90,78,89
18 100,100,95
19 59,59,61
20 </INPUTEX>
```

4.2.3 繰り返し入力のタグ

入力方法の変更を行うことで類似問題を生成する．その中で，型の変更や入力方法の部分で入力変数を読み取る scanf 関数の生成を行う．条件分岐と同様に検討したが，計算式や scanf 関数が繰り返し入力内にあることから実現が困難だったので，計算式はテンプレートファイルに記述した．書き換え記述ファイルに型と変数名を記述し，変数名を参照することで型を定める方法を検討したが，2 つのファイルで整合性をとる必要がある．整合性を保ちやすくするために，あらかじめ入力変数をある変数で固定する方法を検討したが，変数名が制限され，自由度がなくなると考察した．繰り返し入力は CSV 形式で記述する条件分岐とは異なる XML の属性を用いた記述方法を提案する．

繰り返しの入力方法に関する問題の生成に用いる共通のタグは 2 つある．

<VAR>タグは問題文，模範解答プログラムの変数宣言に関わる部分を補完する．模範解答プログラムの変数宣言を補完するためには，型と変数名，場合によっては初期化子が必要であり属性 type，name，init に記述する．入力変数を読み取る scanf 関数の生成に入力変数であるという情報 (input) が必要であり属性 iotype に記述する．それぞれの変数で型を指定したい場合は，指定するための目印が必要であり属性 tgr に記述する．問題文の生成には，入力変数の型が必要である．

<INPUTLOOP>タグは問題文，模範解答プログラムの入力方法に関する部分を補完する．計算式はあらかじめテンプレートファイルに記述するが，繰り返し入力内にあることが多いので開始タグ<INPUTLOOP>と終了タグ</INPUTLOOP>で囲み繰り返し部分を明示する．模範解答プログラムの入力方法を補完するためには，入力方法の種類が必要であり属性 looptype に記述するが，種類によって必要な情報は異なる．プログラムの生成には，あらかじめプログラムでデータ数が指定されている場合，データ数の情報が必要であり属性 num に記述する．実行時に入力データ数を指定する場合，入力データ数を格納するための変数名の情報が必要であり属性 varname に記述する．

何らかの条件を満たすまで入力を続ける場合，満たす条件式の情報が必要であり属性 expr に記述する．問題文の生成には，入力方法の種類が必要である．

繰り返し入力ではソースコード 3 とソースコード 4 のような記述方法を提案する．

ソースコード 3 テンプレートファイル (繰り返し入力)

```
<SENTENCE>
<VAR/>型の<INPUTLOOP/>値の合計を計算するプログラムを作成しなさい．
</SENTENCE>
<CODE>
#include <stdio.h>
int main(void){
    <VAR name="total" init="0"/>
    <VAR iotype="input" name="x"/>
    <INPUTLOOP>
    total = total + x;
    </INPUTLOOP>
    printf("合計は %$です。\n",total);
    return 0;
}
</CODE>
```

ソースコード 4 書き換え記述ファイル (繰り返し入力)

```
<VAR type="int"/>
<INPUTLOOP looptype="fix" num="10"/>
<INPUTEX>
3,2,3,4,2,3,9,2,1,10
</INPUTEX>
```

5 評価

本研究の目的は，問題の整合性を保ちながら，問題生成の手間を軽減し，対象の問題を十分に生成することである．そこで，問題生成の手間を軽減できているか，問題の整合性を保つことができるか評価を行う．

5.1 手間の軽減

問題生成の手間を減らすこと，整合性を保つことに関して，書き換え箇所，書き換え文字数，書き換えファイルの 3 点の基準で調査を行う．1 問目は問題生成に必要な情報を白紙の状態から作成し，2 問目以降は 1 問目の問題で記述したファイルを書き換えて生成を行う．1 問目と 2 問目以降で分けて調査を行い，評価基準に則り評価を行う．

条件分岐で”1 つの点数を入力させその成績を判定するプログラムを作成しなさい”という問題を 1 問目として，そこから 6 問の問題を生成したところ，表 1 の結果となった．表内の A，B，C，D はそれぞれ模範解答プログラム，問題掲示用 HTML ファイル，テンプレートファイル、書き換え記述ファイルを表す．ツールで問題生成をした方が記述量，編集箇所，編集ファイル数が減少したため，手間を軽減させることができている．

表 1 調査 (条件分岐)

		手動			ツール		
		A	B	合計	C	D	合計
1 問目	文字数	235	809	1044	159	175	334
6 問合計	文字数	1031	1341	2372	0	1122	1122
	箇所	112	139	251	0	93	93

5.2 整合性が保てない場合

本研究では分散したファイルの変更を一つのファイルにまとめることで整合性を保ちやすくしてきたが，どうしても整合性を保てない場合も存在し，次のような場合である．

- 入力変数の型と実行例の型が一致していない場合
- 異なるタグの記述で変数名が一致していない場合

1 点目のような問題作成者の誤りは整合性を保てない．2 点目は変数名から情報を得て，記述ファイル内の変数名が一致しているかどうか調べるような整合性を確認するツールの作成で防ぐことができる．

6 考察

入れ子以外の if 文を用いた条件分岐，繰り返し入力，型の変更についての問題の自動生成方法を提案してきたが，整合性を保ちながら生成可能な問題数を増加させるために，次の 2 点のような拡張方法が存在する．

- 条件分岐の問題で，分岐の外で出力を行う問題の生成
- 配列を用いた問題の生成や，出力文の自動生成

1 点目は，条件分岐の分岐を扱う<BRANCH>タグを用いる方法を考察した．例として，<BRANCH>タグの出力の部分でダブルクォーテーションを利用する手段が挙げられる．2 点目は，繰り返し入力の<VAR>タグを用いる方法を考察した．例として，新たに属性 element を加える方法や，属性 iotype に出力変数の目印として output を加えることでそれを利用する手段が挙げられる．

7 おわりに

本研究では，演習問題の生成に関わる問題文，模範解答プログラム，実行例の変更をすべて同一ファイルで行うことで，整合性を保持した問題の自動生成ツールを実現し，有効性を評価し，拡張についての考察も行った．

今後の課題として，6 節で考察した方法の拡張，問題数をさらに増加させるために配列，関数，ポインタ，構造体といった扱う単元の拡張を行うことが挙げられる．

参考文献

[1] 堀井和稀，西條広亮：問題テンプレートを用いたプログラミング学習における空欄補充問題の自動生成に関する研究，南山大学 2013 年度卒業論文 (2014).

[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] 掛下哲郎，柳田崚，太田康介：穴埋め問題を用いたプログラミング教育支援ツール pgtracer の開発と評価，情報処理学会論文誌教育とコンピュータ（TCE），Vol.2, No.2，20-36(2016).

[4] 柴田望洋：新・明解 C 言語　入門編，SB クリエイティブ (2014).