日本語環境ブロックプログラミングと連携した ソースコードの穴埋め選択問題生成システム

Source code hole-filling selection question generation system linked to Japanese language environment block programming

島岡 慎也
Shinya SHIMAOKA
神奈川工科大学情報学部情報工学科
Kanagawa Institude of Technology, Department of Information and Computer
SciencesEmail:s1821121s@gmail.com

あらまし：本項では，プログラミング言語のソースコードの穴埋め選択問題生成システムの問題の生成と，日本語環境ブロックプログラミングを連携させることによる学習支援のためのシステムについて検証します．

キーワード：ブロックプログラミング，問題生成，穴埋め，選択肢

1. はじめに

プログラミング初級者を対象として，ブロックプログラミングと呼ばれる，学習の導入に利用されるシステムが存在する．一方，システム開発では，プログラミング言語のコーディング力が必要とされる．学習のための穴埋め生成問題において，自動で選択肢を生成すると選択肢がワンパターンになりやすい．本研究では，ブロックプログラミングと連携したソースコードの穴埋め問題生成システムと，選択肢の生成方法を提案する．

2. 提案システム

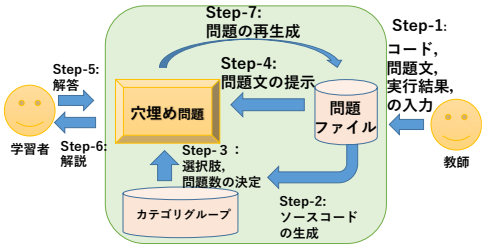

図 1 提案システム図

提案システム（図１）は，ブロックプログラミング環境で利用できる予約語や演算式記号などの要素を，要素が持つ意味や演算機能の類似性などを基準にしてグループ化しておき，ソースコード中に含まれる要素を穴埋め箇所とした場合に，グループ中の要素を選択肢として選択問題を生成する．提案システムの特徴は，穴埋め箇所数を変化させるだけでなく，異なるグループの要素を組み合わせることで，難易度の異なる穴埋め選択肢問題を生成する点にある．提案システムを用いて学習者は，出題されたブロックプログラミングを用いることで論理的思考力を身に着け，さらにソースコードの穴埋め選択問題を解くことでコーディング力を養成することができる．

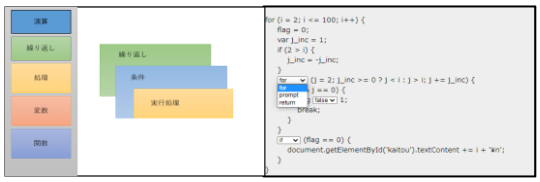

図 2 UI 例

図２の UI 例では，左側にブロックプログラミング環境，右側に対応したソースコードがあり，ソースコードの文中に問題が埋め込まれる形で問題が生成される．日本語のブロックプログラミングとソースコードを比較する，あるいはソースコードの前後を比較することで解答を推測が可能になり，初学者でも解答が推測でき，プログラミング言語の学習が行える．

3. 実験

本システムの問題自動生成機能によって生成した穴埋め選択問題は学習のために適切に問題を生成することができているのか，複数難易度における実際の出題内容をいくつかの

判断基準（表１）ごとに評価した全体の結果を表２，難易度毎の結果を表３～表５に示す．基準４の結果は分散の値となっている．また，実験で問題生成のために利用したカテゴリーグループの例を表 6 に，実験で実際に生成された問題の例を図３，図４に示す．図３は同じ要素からなる問題の生成例，図４は異なる要素からなる問題の生成例である．

表１ 実験評価基準

	基準	評価
1	正解となる解答が選択肢に入っている．	Yes / No
2	設問内で選択肢が複数かぶっていない．	Yes / No
3	ブロックプログラミングより解答が 推測できる．	Yes / No
4	出題内容の種類	予約語， 不等号， 四則演算， その他
5	選択肢のみから正解が推測できない．	Yes / No

表２ 実験結果（全体）

全問題			問題数	354
基準１	基準２	基準３	基準４	基準５
100%	100%	78%	0．806	89%

表３ 実験結果（簡単）

難易度：難しい			問題数	89
基準１	基準２	基準３	基準４	基準５
100%	100%	82%	0．740	100%

表４ 実験結果（普通）

難易度：難しい			問題数	176
基準１	基準２	基準３	基準４	基準５
100%	100%	77%	0．781	100%

表５ 実験結果（難しい）

難易度：難しい			問題数	89
基準１	基準２	基準３	基準４	基準５
100%	100%	78%	0．878	56%

表６ グループ例

グループ１	'for', 'while', 'do'
グループ２	'if', 'else', 'switch'
グループ３	'break‘, 'continue'

```
for (i = 2; i <= 100; i++) {
    flag = 0;
    var j_inc = 1;
    if (2 > i) {
        = -j_inc;
    for (j = 2; j_inc 0 ? j < i ? j > i; j += j_inc) {
        if (i % j 0) {
            flag 1;
            break ;
        }
    }
    if (flag == 0) {
        document.getElementById('kaitou').textContent += i + '¥n';
    }
```

（if / else / switch）

図３ 同じグループから生成された問題例

```
var a, b, c, goukei;


a = 1;
b = 2;
c = 0;
goukei round a + b;
document.getElementById('kaitou').textContent += goukei + '¥n';
goukei ;
document.getElementById('kaitou').textContent += goukei + '¥n';
goukei split b + c;
document.getElementById('kaitou').textContent += goukei while '¥n';
```

（round / if / =）

図４ 異なるグループから生成された問題例

4. まとめと今後の展望

実験結果より，提案システムの実現可能性を確認できた．今後はプログラミング学習の初学者が，プログラミングに関連して，論理的思考力からコーディング力への学習の移行の際に利用される学習環境となることが期待される．

参考文献

(1) 内田保雄：初級プログラミングのための自動作問システム，情報処理学会研究報告コンピュータと教育（CE），Vol.2007，123(2007-CE-092)，pp.109-113，(2007/12/08).

(2) 野上裕二，納富一宏：プログラミング学習支援における問題自動生成に関する基礎的検討， 情報処理学会 第 16 回情報科学技術フォーラム（FIT2017）講演論文集，第 3 分冊，K-022, pp.465-466, (2017.09).