DEIM Forum 2009 D9-5

学習者に合わせたC言語演習穴埋め問題の自動生成

有安　浩平† 　池田　絵里†† 　岡本　辰夫† 　國島　丈生†† 　横田　一正††

† 岡山県立大学大学院 情報系工学研究科　〒 719–1197 岡山県総社市窪木 111
†† 岡山県立大学 情報工学部 情報通信工学科　〒 719–1197 岡山県総社市窪木 111
E-mail: †{ariyasu,eikeda,okamoto,kunishi,yokota}@c.oka-pu.ac.jp

あらまし　大学の情報系学科の多くはC言語の教育を重視していて，C言語についての授業が科目としある．C言語を代表とするプログラミング言語では学習する上で，講義による学習と同等以上に，実際にプログラミングをさせる演習が重要になっている．このプログラム演習では多くの演習問題を解き，たくさんのプログラムに触れることにより学習効果が高められる．しかし，問題作成者の意図に合わせて，たくさんの演習問題を作るのは大変な手間がかかる．そこで，本研究では問題作成者の意図に合わせて自動で穴埋め問題を生成する機構の開発を行った．この機能では解答のプログラムコードと出題傾向を記述したものを入力として，穴埋め問題とその解答を生成する．生成する問題は変数，型宣言，関数，条件判定などを問題作成者の意図に合わせて穴をあけるようにする．また，その生成した問題を提示する方法，学習者に解かせた解答を自動的に評価する方法についても述べる．

キーワード　e ラーニング，C 言語学習支援

Automatic Generation of Fill-in-the-Blank Exercises in Adaptive C Language Learning System

Kouhei ARIYASU†, Eri IKEDA††, Tatsuo OKAMOTO†, Takeo KUNISHIMA††, and Kazumasa YOKOTA††

† Okayama Prefectural University, Graduate School of Systems Engineering　111, Kuboki, Soja, Okayama, 719–1197 Japan
†† Okayama Prefectural University, Faculty of Computer Science and System Engineering　111, Kuboki, Soja, Okayama, 719–1197 Japan
E-mail: †{ariyasu,eikeda,okamoto,kunishi,yokota}@c.oka-pu.ac.jp

Abstract　C language is the necessary subject for computer engineering education. On C language learning exercise is as important as lecture. C programming exercise can provide learning effect for student by solving many problems. But,It is difficult to make many problems to problem making intention. Therefore,We developed the automatic Fill-in-the-Blank question generate system for the C language exercise problem. And in this paper, we report on the method to evaluate the answer and a method to show the poroblem that we generated. The problem that this system generated reflects problem making intention.

Key words　e-Learning, C language learning support

1. はじめに

大学の情報系学科では，プログラミング言語の教育のために，C 言語の教育を重視していて，C 言語の科目を必修としているところが多い．C 言語の学習では講義により文法事項の学習を行い，その後，演習で実際にプログラムを作成させ動作の確認や，プログラムの組み方の学習を行う．プログラムの学習では実際にプログラムを作ることによりプログラミングの作法やエラーへの対処法などを学ぶことができ，大きな学習効果があり，演習は重要な位置づけになっている．このプログラミング演習では多くの問題を解き，たくさんのプログラムに触れることにより学習効果が高められる．しかし，問題作成者の意図に合わせて，たくさんの演習問題を作るのは多くの負荷が問題作成者にかかる．問題作成において，教員の負荷を軽減させるためにソースコードを用いてランダムに穴をあけることにより自動的に問題を作成するシステムの研究が取り組まれている [1]．だが，このシステムでは，ランダムでソースコード中の穴をあけるので，問題作成者の作成意図に合わせた出題ができるとは限

らない.

そこで，本研究では，ソースコードを解析し，そこに問題作成者の作成意図を合わせて作問することにより，問題作成者の意志に合わせた演習問題の自動生成を行う．本研究で作成する問題は，あるソースコードをもとにコード中の一部分を穴へと変換し，その穴になった部分を答えさせる穴埋め問題を対象とする．また，その作成した問題を出題を行うシステムについての提案も行う．この出題システムでは学習者の解答結果により，学習者の理解が不十分なところを見つけ，次の出題時に苦手な部分の問題を重点的に学習を行わせるフィードバックを行う．

2. 現状の問題点

2.1 演 習 問 題

プログラミング演習ではテーマに合わせたプログラム作成課題が与えられ，学習者はその課題に合わせたプログラムの作成を行う．この作成課題は，その授業で満たしたい学習目標に合わせ，意図が与えられ，それをもとに問題が作成される．演習では学習内容ごとに学習意図が変わり，学習の進行状況にともない，学習目標はより複雑なものへと変化していく．そのため，問題作成者は授業の進行に合わせて新しい問題を作成していかなければならない．そのため，問題作成者には問題作成のために多くの手間がかかっているということが 1 つ目の問題点として挙げられる．

2.2 関 連 研 究

このような問題作成の手間を軽減するために，自動で作問するシステムの研究開発がおこなわれている [1]．このシステムでは Java の学習を対象にしていて，問題作成者が設問とソースコードをシステムに登録すると，キーワードや識別子を自動的に抽出し，その部分に穴をあけることにより，穴埋め問題を作成する．問題中の穴になる部分は乱数により毎回異なる場所に出現するようになっており，同じソースで反復練習を行えるようになっている．しかし，乱数を用いて穴をあける場所を決定したのでは，問題を作成する上での意図が盛り込まれないため，問題作成者が作りたい問題にならない可能性が高い．また，このシステムを使い反復学習を行う場合，ランダムで作問されるため，以前間違えた問題にを考慮した出題が出来ないといった問題点があり，2 つ目の問題点として挙げられる．

2.3 問題作成意図とは

問題作成意図とは，問題作成者が「こんな目的で問題を作りたい」と考えている内容であり，ある学習目標に対し,「正解できれば，これが理解できている」という指針になるものである．この問題作成意図は，満たしたい学習目標ごとに様々なものがある．

例えば，構文について学習させたい時，問題作成意図としては「出力文が分かっているのか確認したい」,「for 文の引数が分かっているのか確認したい」,「処理の流れが分かっているのか確認したい」等さまざまな問題作成意図があげられる．

これらは,「処理の流れが分かっているか確認したい」などの抽象的な意図から,「for 文の引数が分かっているのか確認したい」といった具体的な意図まで様々な抽象度の問題作成意図に対応しなければならない．

よって，問題作成意図を考慮した問題の生成を行うことが求められ，3 つ目の問題点として挙げられる．

3. アプローチ

2. で述べた問題点を踏まえた上で，解決のためのアプローチを述べる．

3.1 問題作成意図を考慮した問題の自動生成

まず，1 つ目の問題点と 3 つ目の問題点に対応するアプローチとして，ソースコードと問題作成意図を記述したものから，自動的に穴埋め問題を生成する．自動生成には，問題の基となるソースコードと問題作成意図を記述したファイルが必要になる．これら 2 つのファイルをもとに，ソースコードの構文解析を行い，その解析結果に対し問題作成意図を適応し，問題の自動生成を行う．作成する問題は，穴埋め問題とする．穴埋め問題を使う理由としては，穴埋め問題の穴を広くとれば自由記述の問題として出題できるからである．また，選択式の問題のように解答例がないため，勘で答えることが難しいので学習効果が高いためである．

3.2 誤答によるフィードバック

2 つ目の問題点に対するアプローチとして，学習者の誤答結果から次に出題する問題の制御を行う．学習者の解答の中の誤答を解析し，理解していない分野を推測する．そして，その推測をもとに次の出題時に，理解していない分野の問題を優先的に出すように出題の制御を行う．また，これらの誤答の解析を統計処理し，学生がつまづきやすい分野を見つけ出し教師に提示するのにも役立てる．

4. 機 能 詳 細

この章では 3. 章で述べたアプローチについて，より具体的な機能の実現方法を述べる．

4.1 出　　題

出題は構文解析・問題リソースの生成・問題生成・問題出題の流れで行われる．

4.1.1 構 文 解 析

構文解析機を用いて抽象構文木を自動生成する．本研究で用いた構文解析機は，コンパイラコンパイラの ANTLR [2] を用いて作成し, 実装は JAVA で行われている．同時に記号表も作っておき, 後の採点時に変数の対応を取れるようにしておく．出来上がる構文木の仕様は以下のようになる．

根　出来上がる抽象構文木の根は SOURCE となる．そして，この根は子要素に GLOBAL 要素と FUNC 要素しか持つことが出来ない．

節　節になる要素は以下の表 1 に示されるものがある．表の左に示された名前の節を根とする部分木を取り出すと，取り出された部分木の内容は表の右側のようになる．

葉　葉には変数名，型名，関数名または定数が格納される．また，式の省略要素の NONE が来ることもある．

解析例　簡単なプログラムを構文解析器で解析した例を図 1 に示す.

節の名前	部分木の内容
DECLARATION	宣言
TYPEDEF	型宣言
TYPE	型名
FUNC	関数
GLOBAL	グローバル領域
BLOCK	{} で囲まれた部分
STATEMENT	文
EXPRESSION	式
FORMAT	フォーマット指定子
ARGUMENTS	引数群
ASSIGNMENT	代入
LASSIGNMENT	代入の左辺
RASSIGNMENT	代入の右辺
RVALUE	右辺値
FOR,DO,WHILE,IF,SWITCH	各構文
ARRAY	配列
STRUCT	構造体
MEMBER	構造体のメンバ
NONE	式の省略

表 1　構文解析木の節

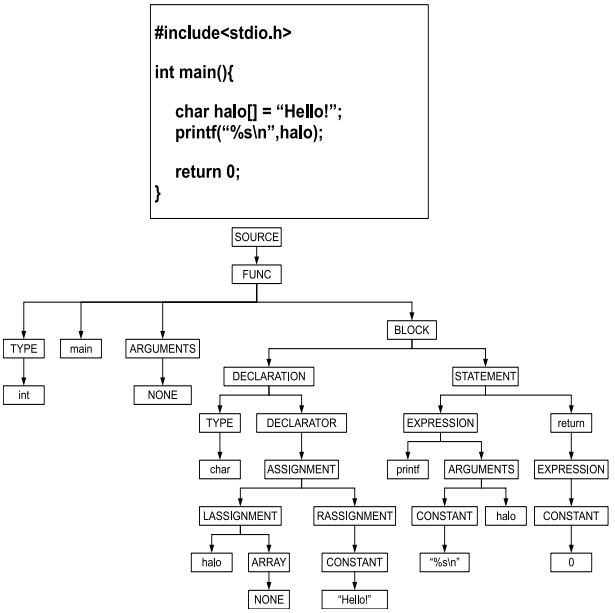

図 1　プログラムの解析例

4.1.2　問題作成意図

問題作成を行うための意図は様々な種類のものがある．今回はプログラミングの学習初期の内容についての考えられる学習意図について述べる．表 2 は問題作成意図の例である．表 2 で表された意図は抽象的で、表現が多く記述のバリエーションが多くなる傾向がある。そこで、この問題作成意図は、単純化・具体化するために、以降に説明する問題作成ルールを組み合わせることによって表現する。

入力関数の引数が分かっているか
制御構文の引数が分かっているか
出力関数の引数が分かっているか
関数を使うのに必要なヘッダが分かっているか
必要となる引数が分かっているのか
所定の演算子が使えるか
処理の流れを理解しているか
配列の定義のしかたが分かっているのか
フォーマット指定子が分かっているか
演算子の使い方が分かっているか
どこでどの変数を持ちいらなければならないか分かっているか
返り値が分かっているか
必要な関数が分かっているか
処理内容が分かっているか
自分でプログラムを作れるか
出力結果のイメージができるか
出力関数の動きが分かっているか

表 2　問題作成意図の例

4.1.3　問題作成ルール

問題作成ルールは以下のような記述を持つものになる．この記述を複数個組み合わせある問題作成意図を表現する。

createblank(枝名, 指定 (, 指定内容))

枝名には表 1 で示した構文木の節が来る．この指定した節以下を穴埋めの穴と置き換える．
指定には”all”または”select”が入り，”all”の場合は枝名で指定した枝の下はすべて穴になる．また，”select”を入れた場合は，後の指定内容で指定したとおりの穴のあき方になる．
指定内容では，枝名で指定した部分の中でも一部分にだけ穴を開けたい場合に指定するのに用いる．しかし，この部分指定では構文解析で分割された単位以下の指定はできない．ここで用いれるものは，rootonly(枝名となっている部分のみを書き換える)，nameonly(子要素の変数名のみ書き換える)，numberonly(変数の値のみ書き換える)，conditiononly(条件式のみ書き換える)などがある。
問題作成意図の記述の例を以下に示す.

createblank(printf,all)
createblank(char,select,rootonly)

4.1.4　問題の生成

問題は 4.1.1 節で述べた構文解析の結果と 4.1.2 節で述べた問題作成意図から作られる．まず，構文解析した結果の変数となる葉に id を付け，変数名から変数名と id の対応表を作る．この対応表では，同じ変数は同一の id で定義される．

そこに選ばれた問題作成ルールを順に適用し，穴をあけていく．穴の開け方は，まず問題作成ルールの枝名を読み，構文木から枝名と一致する部分木を見つける．次に指定のところで記述された内容に従ってその部分木を穴に置き換える．そして，今回出題には ADEL のテスト機構 [3] を用いたのでテスト機構で扱える XML の形式に成型する．このとき，生成する問題には問題ごとに問題 id がふられる．そして、適応した問題作成

ルールの種類によりグループ分けされ, 問題データベースに格納される．

テスト機構の記述は以下の項目を持つ XML である．1 つの問題が item 要素 1 つに対応していて，問題を記述した question 要素や採点情報を記述した evaluate 要素を持っている．また，似た内容の問題を持つ item 要素を集めて問題グループとし program_set 要素でくくることもできる．

program_set　問題のグループを持つ要素で，問題グループを識別する id 属性を持つ．子要素には複数の item 要素を含む．

item　問題候補のルート要素であり，固有の識別子である id 属性と問題の形式を表す type 属性をもつ．item1 つが問題 1 問に相当する．子要素に question 要素，response 要素，hints 要素，explanation 要素，evaluate 要素をもつ．

question 要素　問題の内容を記述する要素で，穴が開けられたソースコードはここに記述される．

response 要素　提示の際に解答の候補として記述され，穴埋め問題の解答欄の数に比例する．

hints 要素　問題に対するヒントが記述されている．

evaluate 要素　採点に関する事柄を記述する要素．子要素に function,correct,score,weight,point を持つ．

correct 要素　正解となる文字列を記述する．この要素と解答を比較し正解判定を行う．

point 要素　この問題を正解すると加算される点数を記述する．

explanation 要素　採点後に表示される文字列を記述する．

そして，問題記述の例を図 2 に示す．

```
<program_set id=”1”>
 <item id="1" type="fill_in_the_blank">
      <question>
     <p>[ A ]と[ B ]にあてはまる語句を入力しなさい</p>
     <p>#include<stdio.h><br/>
          <br/>
          int main(){<br/>
          [ A ] halo[] = “Hello!”;<br/>
          printf(“[ B ]”,halo);<br/>
          return 0;<br/>
          }</question>
     <response id=”1”>=[ A ]</response>
     <response id=”2”>=[ B ]</response>
     <hints>フォーマット指定子について</hints>
     <evaluate>
          <function></function>
          <correct id=”1”>char</correct>
          <correct id=”2”>%c\n</correct>
          <score>1</score>
          <weight correct="1" incorrect="-1" />
          <point>10</point>
     </evaluate>
     <explanation>文字を表示するときは%cである。</explanation>
   </item>
</program_set>
```

図 2　問題記述の例

4.1.5　出　　題

ADEL のテスト機構では先ほど述べた記述を読み込み問題を提示する．出題時の問題の選択は, 問題データベース中に存在するグループから出題意図を基にシステムが選択し, そのグループ内からランダムで問題が出題される．出題する問題のイメージを図 3 に示す．この例は，4.1.2 で示した問題作成ルールの記述例と，図 1 の構文木から問題を作成した例である．

```
●問題1
[ A ]と[ B ]に当てはまる語句を入力しなさい。

#include<stdio.h>

int main(){
      [ A ] halo[] = “Hello!”;
      printf(“[ B ]”,halo);
      return 0;
}
```

[　　] = [A]

[　　] = [B]

図 3　問題イメージ

4.2　解答との照合

4.2.1　照 合 方 法

学習者が解答してできたソースを構文解析し構文木の作成を行う．そして, 問題作成時に作成した構文木と比較を行う．また，変数の位置をすべて穴をあけた場合は問題リソース作成時につけた id をもとに，記号表により，学習者が入力した文字が問題作成者の意図した文字列でなくても，必要な場所にすべて同じ文字列が入っていた場合正解とする．

4.3　フィードバック

正誤判定を行って，誤りと判定された問題を対象に行う．出題時に振り分けたグループをもとに問題出題意図を特定する．今回，フィードバックには ADEL の ECA ルールを用いる [4]. ECA ルールは Event,Condition,Action を用いたルールの記述である．本研究では，問題を解答するという Event 時に，間違えた問題の問題作成意図を Condition にし，問題を切り替えるという Action を起こす．そうすることにより，次に学習を行うときに前回間違えた問題のグループを重点的に学習できるようになる．

4.4　システムの流れ

システムの流れは図 4 になる．まず，システムはソースコードを解析機を使い構文解析を行い，解析木を作成する．その出来上がった解析木と問題作成意図を用いて，問題を生成する．続いて，生成された問題をグループ分けし，グループ分けした問題の中から, 前回からのフィードバックなどをもとに出題する問題を決定する．そして，問題を学習者に解かせる．学習者の解答の後，採点を行い，間違えた問題を解析しフィードバック情報を生成するといった流れで処理が行われる．

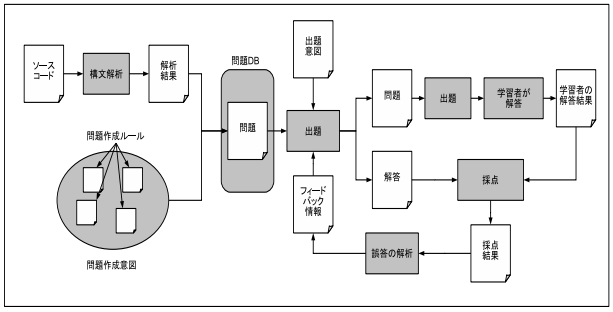

図 4　システムの流れ

5.　結論と今後の課題

本研究では，教材作成者の問題作成意図を踏まえた上での問題の自動生成の手法を提案した．また，フィードバックにより学習者の理解度に合わせた問題の自動生成と提示法の提案も行った．これにより，問題作成者の意図にあった演習問題の作

成時の手間を軽減する事が可能となった．今後の課題としては，以下のようなことがあげられる．

未実装部分の実装とシステムの評価　今回時間の都合上行うことのできなかった部分の実装を行い，実証実験を行いシステムの有用性を示す必要がある．

問題作成意図・採点・フィードバック機能のさらなる考察　問題作成意図・採点・フィードバック機能には問題が残されているため，更なる考察をする必要がある．

オーサリング機能の付加　解答に複数の記述方法があった場合，複数のソースコードをアップロードしておいたり，選択問題を出題するため間違いの選択肢をもんだ作成者に入力してもらうことが考えられるため，オーサリング機能の付加が必要である．

多言語への応用　現在のシステムでは,C 言語にしか対応できていないため,JAVA や COBOL など他言語学習への応用を考察して行く．

文　　献

[1] 内田保雄. 初級プログラミング学習のための自動作問システム. 情報処理学会研究報告, Vol.2007, No.123(20071207) pp. 109-113 2007-CE-92-(16).

[2] ANTLR. ANTLR Parser Generator. http://www.antlr.org.

[3] 西輝之, 延原哲也, 劉渤江, 横田一正. ” 適応型 e ラーニングに必要な診断的テスト機構”. DEWS2006.

[4] 小西裕, 延原哲也, 横田一正. ” 適応型 e ラーニングシステム ADEL における ECA ルール制御機構の実現”. 平成 19 年度電気・情報関連学会中国支部第 58 回連合大会，pp.473-474，Oct.20，2007.