Vol.2012-CE-114 No.6
2012/3/16

プログラミング学習のための ソースコード読解問題類題生成システムの実現と評価

吉田拓己[†1]　立岩佑一郎[†1]
山本大介[†1]　高橋直久[†1]

本研究では読解問題の類題を生成するシステムを実現し, 読解問題の作成の負担を減らすことを目標にしている. ソースコードの解析結果によって読解問題の出題方針を決めることができる. この出題方針を用いることで類題を作成することが可能になった. 評価結果から類題作成時間と手間の軽減ができたことを確認した.

Implementation and Evaluation of a System for Generating Similar Problems of Reading from a Program for Programming Learning

Takumi Yoshida,[†1] Yuichiro Tateiwa,[†1]
Daisuke Yamamoto [†1] and Naohisa Takahashi[†1]

In this research, the system which generates the similar problems of a reading is implemented , and it aims at reducing the burden of creation of a problems of reading. The policy of generating problems can be decided by the analysis result of a source code. It became possible to create similar problems by using this policy. As a result of evaluation we confirmed this system can reduce creation time and burden.

†1 名古屋工業大学大学院
Graduate School of Engineering,Nagoya Institute of Technology

1. はじめに

本稿では C 言語のプログラムのソースコードを読んで, それに対する問題を解く形式の読解問題の類題生成システムを提案する. 当研究室ではプログラミングの学習を支援するシステム CAPES[1)] の研究を行っている.CAPES ではプログラミング技術を受講者に身につけさせるため, 演習を半自動化することで繰り返し学習を支援している. 一般的なプログラミング演習や CAPES では課題の要求するプログラムを作成させる記述形式の問題を扱っている. 記述形式の問題が解けない原因として受講者（特に初学者）がプログラムのソースをよく読まない, 新たに習った文法を理解できていない, 正解プログラムのアルゴリズムを考えられない, といったことがあげられる.

プログラミング学習の初学者は記述問題に取り組む前段階として文法やアルゴリズムを理解する必要があるため, 参考書などに書かれたプログラムのソースをよく読むといった方法をとる. しかし読むだけでは実際に理解したかどうか確認しにくい. そのためプログラムをよく読んだか確認するには, 以下のようにそのプログラムの内容に関した設問を出題すればいい.

- 文法の理解の確認
  文法を学ぶにはその文法が使われているプログラムの読解問題を出題
  例）
  for 文を学びはじめた受講者に対して,for 文を含むプログラムの例を提示して, その for 文の本体が何回繰り返されるかを問う. 受講者がこの問いに正しい答えを返した場合には, その受講者はプログラムを読解して for 文の繰り返し動作を正しく理解していると判断できる.
- アルゴリズムの理解の確認
  アルゴリズムを学ぶにはそのアルゴリズムが使われているプログラムの読解問題を出題
  例）
  バブルソートのアルゴリズムを学び始めた受講者に対して, バブルソートのアルゴリズムを含むプログラムの例を提示して, ソート途中の状態やソートを完了するまでに何回スワップが行われたかを問う. 受講者がこの問いに正しい答えを返した場合には, その受講者はプログラムを読解してバブルソートのアルゴリズムを正しく理解していると判断できる.

また読解形式の問題は記述形式の問題よりも比較的容易である. そのため一回の解答におけ

る負担が少なく, 短時間で解くことができるため繰り返し学習に適していると言える. しかし全く同じ読解問題を繰り返し解いても正解が分かっていると問題を読まなくても解けてしまうことがあるなどあまり効果的ではなく, それを解決するために多数の読解問題を作成するのはとても手間がかかる.

一般的に演習は指導者が問題を出し, 受講者がそれに対する解答を提出し, 指導者が採点を行う. これらを全受講者に対し一人の指導者で行うと, 特に採点で大きな負担が生じることになる. これを解決するためには答案の自動評価を行わなければならないが, 自動評価をするには受講者の提出した答案が正解かどうか比較し判断するための正解例が必要となる.

以上のように読解問題を演習で用いる場合には, 読解問題の作成の手間と課題取得から答案評価までの負担を減らさなければならない.

これらの問題を解決するため, 以下の要件を満たす類題生成システムを実現することを目標としている.

要件 1 プログラムの内容を深く理解しているか確認するため，変数の値の変化やプログラムの動作の追跡を伴うような読解問題を生成する.

要件 2 演習効果を高めるため，類似しているが少し異なるような多数なプログラムの読解問題を生成する.

要件 3 課題取得，答案提出，答案評価の QA サイクルからなる繰り返し学習を可能にするため，答案の自動評価を実現する.

要件を満たすためタグにより出題方針指定可能な読解問題類題生成システム[2] を実現した. プログラムの書き換えたい箇所や出題したい設問の種類などの情報を含んだ出題方針をタグとしてプログラムのソースコードに記述することで問題プログラムの書き換えや, その正解例を自動で生成し, それを用いることで答案の自動評価を可能にした. しかしこのシステムにはどのタグをどの個所に付けられるか把握しないと問題を作成できないといった問題がある. タグにはプログラムの書き換えを行うものや, 設問を出題するものがあるがプログラムのどの個所に書くことができるのかといったタグに関する知識が無いとどうやって問題を作成すればわからない. また, 中途半端にタグ付けを理解していると誤った箇所にタグを付けてしまう可能性がある. 誤った箇所にタグ付けを行ってしまうと類題が正しく作成されない.

これらの問題を解決するシステムとして, プログラミング学習のためのソースコード読解問題類題生成システムを提案する.

提案システムは類題設定ファイル作成機能を持つ. 類題設定ファイルとは, ソースコードの中にどの個所を書き換え, どの設問を出題するかといった類題を作成するのに必要な情報（出題方針）を書き込んだファイルである. システムがソースコードを解析し, 出題方針として設定可能な箇所を提示することでどの個所にどんなタグを付けられるのかといった知識が無くても問題を作成できるようにする. 問題追加者はシステムによって提示された個所を選択し内容を入力することで, その内容に応じたタグ付けを行った類題設定ファイルをシステムが自動で作成する.

本システムは以下のユーザを対象にしている.

対象者 1 読解問題の類題を作成したい問題追加者

読解問題を受講者に出題する際に演習効果を高めるため, 多くの類題を作成する必要がある. このシステムを使うことで類題の作成の手間を軽減することができる.

対象者 2 記述問題が理解できなかった受講者

記述問題が理解できない受講者に読解問題を提示し問題を解くことで新しく習う文法やアルゴリズムを習得し, 記述問題も解けるようにする. 本システムでは設問の正解例を表示させることでプログラムの途中結果や出力結果を知ることができる. これらをアルゴリズムの習得に役立てることができる.

対象者 3 理解度の確認や復習のために繰り返し問題を解きたい受講者

読解問題を繰り返し解くことで理解度の確認や復習を行う. 生成される問題は毎回異なるようにするため, まったく同じ問題を解くのと違って考える必要がある.

2. 提案システムの実現法

2.1 本研究で対象とする読解問題とその類題

読解問題とはソースコードを読まないと解くことができない問題で, 学ばせたい文法やアルゴリズムを含んだソースコードに対する問題を出題することで理解の確認をすることができる. 本システムで扱う読解問題は, **設問**, **表示用ソースコード**, **正解例**によって構成されている. 設問はプログラム中のある部分に対する問題で, 例えば printf での出力を問う設問がある. 表示用ソースコードは受講者に表示されるプログラムのソースコードである. このソースコードには設問の出題箇所が分かるように設問対象行に対して設問番号が付与されている. 正解例は各設問の正解となるもので, 本システムではこれを用いることで正誤判定を自動で行うことができるような問題を扱っている.

類題は読ませたいソースコードに対して出題する設問の形式とソースコードの書き換えによって作られる少し異なる問題である. 設問の形式を変える場合, たとえばあるソースコードの出力を問う問題から実行過程を問う問題や入力を問う問題に換えることで, 同一ソースコードから複数の問題を作成することができる. また, ソースコードを書き換えることによってプログラムの動作が変わるため, 同一の設問から複数の類題を作成することができる. 本システムで扱う類題は設問の種類と表示用ソースコードの書き換えの組み合わせによって決定する. 設問の種類を変えることで受講者に問う内容を変更し, プログラムの変数の初期値を書き換えることでプログラムの動作を変える.

提案システムは問題追加者向けの読解問題作成支援システムと受講者向けの読解問題学習支援システムの２つにより構成されている. 問題追加者は読解問題作成支援システムを用いて類題設定ファイルを作成し, それにより生成された類題から受講者に出題するものを選ぶ. 受講者が問題を選択するとシステムがその問題の類題をランダムに取得し問題を作成する. 受講者が入力した解答は正誤判定機能で正しいか判定され, その結果を受講者に表示する. また正解率などの演習履歴を DB に格納し, 問題取得機能で参照することで受講者の達成度に応じた類題を出題することも可能となっている.

大まかな流れは図 1 のようになる.

2.2 類題設定ファイル作成機能の実現法

プログラムの解析機能, 設定箇所提示機能, 出題内容取得機能, タグ付け機能を提供することで本システムのメイン機能である類題設定ファイル作成機能を実現する. この機能の大まかなながれは図 2 のようになる.

STEP1 プログラムの解析

解析するプログラムは自分で用意したものを入力する他にも, あらかじめ登録されたプログラムを選んで解析することもできる. 汎用的なプログラム静的解析ツールを用い, プログラムに含まれる関数定義情報, 関数参照情報, 変数定義情報, 制御構造情報を取得する. 取得した情報にはそれぞれ名前, 出現行といった情報が含まれるのでそれぞれに ID を割り振る. 出題箇所と内容はこの ID で管理する.

STEP2 設定箇所提示

この解析結果を用いてタグを付けることが可能な箇所を抜き出し, 問題追加者に提示する. 例えば print が含まれる箇所は出力を問う問題が出題され, 変数が定義されている箇所は初期値の書き換えが可能であることを表示する.

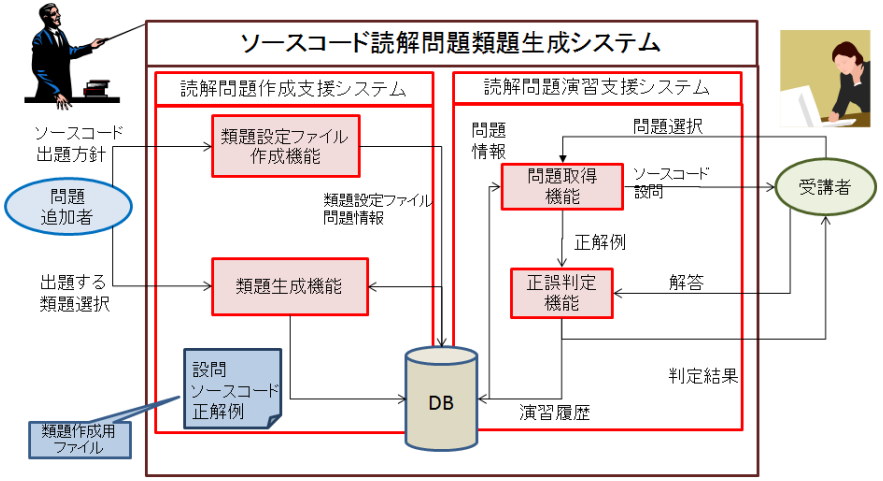

図 1 システム概要

STEP3 出題内容内容取得

問題追加者は表示された内容から出題したい箇所を選択し, どのような設問, 書き換えを行いたいかといった問題内容に関わるものや, 類題生成数, 類題設定ファイルの名前といった問題情報を入力する.

STEP4 タグ付け

システムはその内容に合うタグ付けをソースコードに行い類題設定ファイルを生成する.

2.3 プログラム解析機能の実現法

入力されたプログラムに対して C コンパイラと同様に字句解析と構文解析を行う. プログラムの解析は, プログラミング演習支援システムにおける実行履歴の構造化方式[3] で用いられているものを本システムで扱えるよう解析する情報を追加した. この解析を行うことで入力されたプログラム中の関数や変数の定義されている行番地や名前, 制御構造情報などを構文解析データとして抽出する.

この機能は以下の手順で行われる.

STEP1 解析するプログラム取得

問題追加者によって入力, もしくはあらかじめ登録されたプログラムの内容を取得する.

STEP2 解析

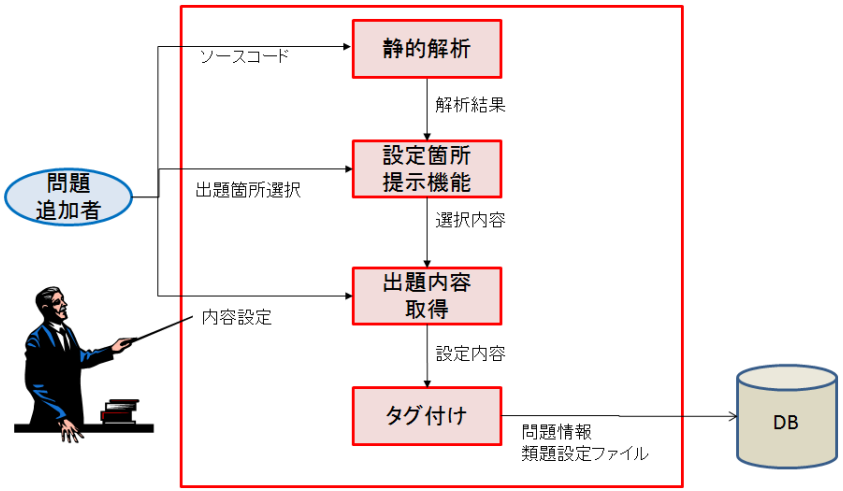

図 2 類題設定ファイル作成機能

表 1 関数定義情報

項目	説明	例
関数名	定義されている関数の関数名	maxmin
開始行	関数本体行の開始行番号	3
終了行	関数本体部の終了行番号	6
戻り値の型	戻り値の変数の型	int
仮引数の数	仮引数として使用される変数の数	2
仮引数の型	仮引数として使用される変数の型名	int,int
仮引数の名前	仮引数として使われる変数の変数名	a,b

取得したプログラムに対し字句解析と構文解析を行う.

STEP3 解析結果の変形

解析結果を種類に応じて配列に格納し関連付けする.

解析結果として用いるデータを表 1 から表 4 に例とともに示す.

2.4 設定箇所提示機能の実現法

プログラム解析機能によって得たプログラム中の変数や関数の定義場所を用いて, 問題追加者に設定箇所を提示する. 問題追加者は提示された個所を指定することでシステムに出題する個所を伝える. この機能は以下の手順で行われる.

表 2 関数参照情報

項目	説明	例
関数名	参照されている関数の関数名	printf
参照場所	関数呼び出しがされている関数名	main
行番号	関数呼び出しがされている行番号	13
実引数の数	実引数として使用される変数の数	2
実引数の名前	実引数として使われる変数の変数名	” n= % d \ n”,n
文字列のみ	実引数が文字列のみかどうか	no

表 3 変数定義情報

項目	説明	例
変数名	定義されている変数の名前	a
定義場所	変数が定義されている関数名	main
行番号	変数定義がされている行番号	3
変数型	定義されている変数の型	int
仮引数	関数の仮引数となっているか	yes
入力	プログラムの入力として扱う変数かどうか？	no

表 4 制御構造情報

項目	説明	例
種類	制御構造の種類	for
制御変数	制御構造を制御する変数名	i
配置場所	制御構造が配置されている関数名	main
開始行	制御構造を開始している行番号	12
終了行	制御構造を終了している行番号	15
条件式	制御構造の条件式	i < 6
初期化式	for 文の初期化式	i=1
再初期化式	for 文の再初期化式	i++

STEP1　プログラムの解析結果を取得

解析結果は関数定義情報, 関数参照情報, 変数定義情報, 制御構造情報の４種類に分けられ配列に格納されている.

STEP2　プログラム中から解析結果で見つけた箇所を抽出

関数定義情報から出現行と名前のデータを取り出し, プログラム中の一致した行の同じ名前の部分を見つける. 他の情報についても同様.

STEP3　抽出した個所を問題追加者に提示

見つけた箇所は解析結果の種類 (関数参照情報や変数定義情報など) によって色分けするなど異なった形で問題追加者に表示される. 色分けなどをして問題追加者にどの種類の情報によって提示されたものなのかわかるようにする.

STEP4　設問出題可能箇所を問題追加者に提示

出題するかどうかの選択を行うインタフェースを問題追加者に提示する. 出力を問う問題は一部を問う場合は printf を, 全部を問う場合は main 関数を選択することができるようプログラム中の出現箇所に埋め込む. 出題するか出題しないかの 2 種類の選択肢を用意する.

実行過程を問う問題はプログラムとともに表示される行番号に対して変数の値を問う問題, 実行回数を問う問題, 出題しない, の 3 種類の選択肢を用意する.

STEP5　プログラム書き換え可能箇所を問題追加者に提示

変数の書き換え可能箇所に対して書き換えるかどうか選択させるインタフェースを埋め込み, どのような書き換えをするか指定させる. このとき変数の型の情報を解析結果から取得し, 変数の型によって選択肢が変化する.

- 数字を扱う変数の場合
  1,2,3 のようにコンマで区切られた数字の中から一つをランダムに選ぶ【選択型】と 1 から 3 のように数字の範囲を決める【範囲型】の２種類の方法で変数の初期化を行うことができる.
- 文字や文字列を扱う変数の場合
  a,b,c や aa,bb,cc のようにコンマで区切られた文字 (文字列) の中から一つをランダムに選ぶ【選択型】で初期化を行うことができる.
- 配列を扱う場合
  1,2,3 を 2,3,1 にするなど配列の要素を並べ替えることで初期化する【要素型】を行うことができる.

2.5　出題内容取得機能の実現法

問題追加者が選択した個所を取得し, その箇所に応じた内容の入力フォームを用意する. このフォームに入力することでどのような問題を出題するのかをシステムに伝えることができる. この機能は以下の手順で行う.

STEP1　設定箇所を取得

設定箇所提示機能で出題するように選択したものを取得する. 解析結果を用いてどの個所が出題されるのか決定する.

STEP2　設定箇所に対する内容を取得する入力欄を問題追加者に表示

設定箇所の選択内容によって問題追加者に表示するための入力フォームは異なる.

- 変数書き換えの内容を入力
  選択された書き換えの種類に応じて内容を入力する欄を作成する.
  【範囲型】が選択されていた場合は最小値と最大値を入力する欄が用意され, それ以外は候補となる要素をコンマで区切って入力する欄が用意される. この入力欄の名前も解析の種類＋出現番号となっており, これにより書き換え箇所と書き換えの種類と書き換え内容を管理する.
- 出力を問う問題の内容入力
  選択された個所が main 関数だった場合は特に何も入力させない. 選択箇所が printf だった場合は, その printf の出力を問う条件を設定するための条件を入力する欄が表示される.
- 実行過程を問う問題の内容入力
  選択された設問の種類が実行回数を問うものだった場合, ある条件で何回実行されるかといった条件を付けることができる. また, 変数の値を問う問題の場合は問う変数の名前と, 設問の条件を設定することができる. 問うことができる変数は指定した行を含む関数の中で宣言されていないといけない. そのため関数定義情報の開始行と終了行のデータを用いて指定された行がどの関数に含まれているか判断する. つぎに変数定義情報の定義場所を取得し指定された行を含む関数で定義されている変数を取得する. これらを候補として選択できるようにすることで, どの変数を出題できるか即座にわかるため問題追加者の負担を減らすことができる. また問う変数によっては小数・以下どれくらいを問うか設定したり, 配列の何番目の要素かを問う入力欄を表示する.

表 5 問題情報 DB

項目	説明	例
id	類題設定ファイルを識別するための番号	1
unit_id	問題を登録する単元番号	1
program_id	問題を登録する課題番号	1
count	類題生成数	20
name	類題設定ファイルの名前	文字列の分割
visible	受講者に表示するかどうか	yes
out_size	出力を問う設問の数	2
in_Asize	入力を問う問題の正解の数	2
in_Qsize	入力を問う問題の選択肢の最大数	5
exe_size	f 実行過程を問う設問の数	3
i_size	プログラムを書き換える箇所の数	4
er	実行結果を問う問題を出題するかどうか	yes

- 問題情報の入力
  単元番号, 課題番号, 類題設定ファイルの名前, 類題生成数などを入力する. 単元番号と課題番号は課題一覧取得機能で用いる. また表示する課題名として類題設定ファイルの名前を用いる. 類題生成数は作成した類題設定ファイルでいくつの類題を作成するか決めるものである.

STEP3 設定内容の確認

STEP2 で設定した内容の一覧を表示して問題追加者に確認をしてもらう. ここで確認を押すとタグ付け機能が実行され, 類題設定ファイルとして問題情報とともにデータベースに格納される.

2.6 タグ付け機能の実現法

出題内容取得機能で取得した出題箇所と内容を用いて元のプログラムにタグ付けを行う. このタグ付けを行ったファイルを類題設定ファイルとしてデータベースに登録する. 同時に問題情報も登録する. 問題情報のデータベースは表 5 のようになっている.

この機能は以下の手順で行われる.

STEP1 設定番号を決定する

データベースに登録されている設定番号の最大数を取得しその番号+1をした番号を今回登録する設定ファイルの番号にする.

STEP2 出題方針の取得

プログラム解析結果と出題箇所と出題内容を取得する. 解析結果の各種類の情報から出現行と名前のみを取得する.

STEP3 タグ付け

プログラムを一行ごとに読み込み, その行に対して各出題箇所から出現行の一致したものを取り出してタグ付けを行う. 各種類に対するタグ付けは以下のようになる.

- 変数の書き換えによるタグ付け
  選択された書き換えの方法によって付けるタグの種類が決まる. タグの内容は出題内容で取得したものを用いる. タグが決定されると, その変数の初期値が書かれている部分をタグで置き換える.
- 出力を問う設問によるタグ付け
  出力の一部を問う問題が出題された場合は出現箇所の printf の一文を囲むように設問タグを付ける.
- 実行過程を問う設問によるタグ付け.
  実行過程を問う設問が出題された場合は出題箇所の行を囲むように設問タグを付ける. 実行回数を問う場合は設問タグの属性を実行過程に設定する. 変数の値を問う場合はタグの属性を変数の値に設定し, 出題する変数と, それに対する変換指定文字列を与える.

STEP4 データベースに登録

以上の STEP で取得したデータを問題情報としてデータベースに登録し, 類題設定ファイルをフォルダに格納する.

3. プロトタイプの実装

類題設定ファイル作成機能ではプログラムの解析によってどの個所に問題を設定できるか提示する. プログラム中からどの個所に, どんな内容の問題を出すかといった出題方針を入力することでタグ付けを自動で行い類題設定ファイルと問題情報をデータベースに登録する.

3.1 プログラム解析機能の実装

プログラム解析機能の実現法に基づいてプログラム解析機能を実装していく. 解析するプログラムの他にも単元番号や課題番号を入力する. システムは入力されたプログラム中から

関数定義情報などの情報を取り出して情報の種類ごとに配列に格納する. この配列の内容を関連付けし以降の処理で必要に応じて用いる.

3.2 設定箇所提示機能の実装

字句解析や構文解析による結果からプログラム中の設問出題が可能な箇所や書き換えが可能な箇所を抽出して図 3 のように問題追加者に提示する. 関数定義情報は青, 関数参照情報は緑, 変数定義情報は赤, 制御構造情報は黄で色分けし, どの個所に何があるかわかりやすく表示する. 出題箇所の決定や内容の選択はセレクトボックスを用い, この内容を出題内容取得機能に送る.

実装した提示箇所を以下に示す.

- 出力を問う問題の出題箇所
  出力を問う問題は主に printf の出現箇所に出題できる. ただし実引数が文字列のみの場合は出力を問う問題に適さないため出題可能箇所として提示しない. また main 関数を選択することでプログラムの出力全てを問う問題を出題することもできる.
- 実行過程を問う問題の出題箇所
  実行過程を問う問題は任意の箇所に出題できる. ただし出題箇所によっては意味のないものになるため, 主に for の中など制御ブロック内に出題するのが望ましい. しかし今回のプロトタイプシステムでは任意の行を指定する形式をとった. プログラムを実行したときのある行が実行される回数や, ある時点での任意の変数の値を問うことができる.
- 変数の書き換え可能箇所
  変数の書き換えが可能な箇所は変数が定義されている場所かつ初期化も行われているところに制限した. これは解析結果が変数の定義の部分だけしか抽出しないためである. また, プログラムの入力として扱われている変数のみ変更できる.

3.3 出題内容取得機能の実装

設定可能箇所提示機能により提示した個所から, 問題追加者がセレクトボックスで指定したものに対する詳細な内容を取得する.

選択した各設問や書き換えに対する入力フォームが用意され, この入力フォームに出題内容を入力することでその出題内容に合ったタグをタグ付け機能で行う. 今回実装したプロトタイプシステムでの出題内容は以下の 3 つに分類される.

出題内容１ 書き換えの内容
選択された箇所をどのように書き換えるか入力する. 例えば【範囲型】だった場合は最小値と最大値を入力する.

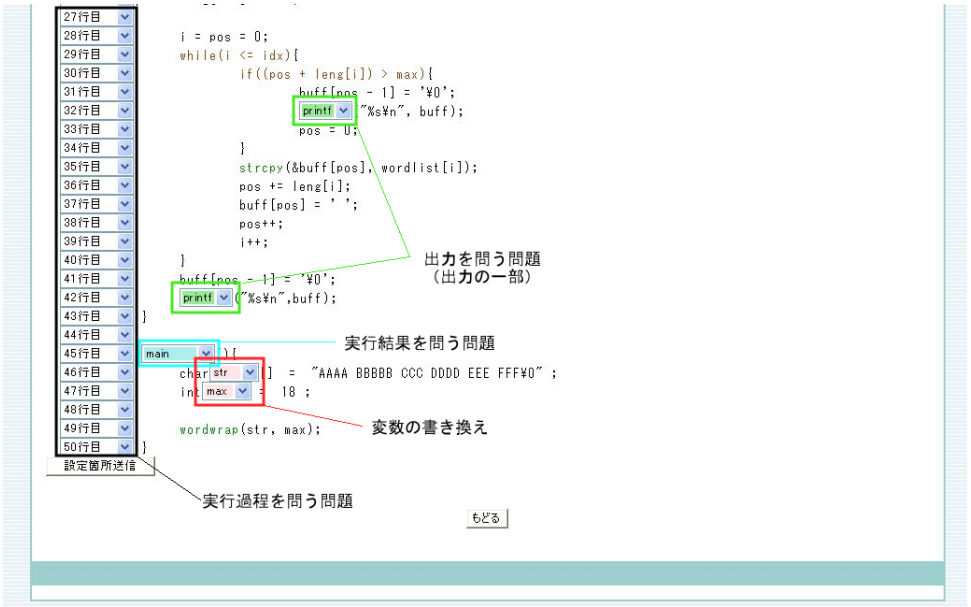

図 3 設定箇所提示画面

出題内容２ 設問の内容
設問の種類に応じて内容を入力していく, 変数を問う問題は問いたい変数を入力し必要に応じて条件や配列の番号などを入力する.

出題内容３ 問題情報の内容
設定ファイルの名前や類題をいくつ生成するかといった類題生成数を入力する.

3.4 タグ付け機能の実装

タグ付け機能の実現方法に基づいてタグ付け機能を実装した. 出題内容取得機能によって習得された内容を用いてタグを作成し, 設問箇所提示機能によって選択された個所に対して対応したタグを自動で書き込む. その後入力した問題情報を DB に格納する. この機能は出題内容の確認がされた後自動で行われ, 登録完了画面に移行する. この画面では作成した類題設定ファイルを用いて類題を作成するか, 一旦メインメニューに戻るか選択することができる.

表 6 評価結果

	類題作成時間		誤り率		手間	
	手動	提案	手動	提案	手動	提案
簡単	2:19.4	0:58.2	11.5	0.0	1.96	4.08
複雑	5:24.4	1:01.0	3.8	0.0	1.19	4.87

4. 評　　価

4.1 評価方法

提案システムにより読解問題の類題作成の時間と手間が軽減できたことを検証するために評価実験を行った. ソースコードの書き換えと正解例をともに手動で作成する方式に対して類題作成の時間と作成した類題が出題方針に従っていなかったり正解例が間違っている確率（誤り率）とアンケートによる手間の評価を行い比較した. 実験を開始する前に被験者である同研究室の 13 名に簡単なプログラムと複雑なプログラム, それぞれに対する出題方針を与え各方式で読解問題を１題作成してもらった. 読解問題の作成の方法をあらかじめ理解することで, 慣れによる評価結果への影響を無くすためである. その後各方式で類題を 10 題作成してもらい, 作成するまでの時間と誤り率を測定する. 最後にアンケートを行って各方式での類題作成がどれほど手間に感じたか調べた.

4.2 評価結果

評価結果は表 6 のようになった. 類題作成時間は類題を 10 題作成するまでの平均時間で, 誤り率は作成した類題の中に何%の誤った問題があるか表している. 手間はアンケートによる５段階評価の平均で 1 に近いほど手間がかかると感じ,5 に近いほど手間を感じていない. この表から問題の難易度が上がると手動での作成時間が上がるのに対し, 提案システムではさほど変わらなかった, これは手動での答案を作成する負担が増えたからだと考えられる. 答案の作成時間は手動だと１題作成する時間×類題作成数だけかかってしまうが, 提案システムだと出題方針を決める時間＋類題生成時間なため提案システムは類題作成数の影響をあまり受けないと言える. また誤り率が簡単な問題のほうが高いのは簡単な問題ほど答案や類題が出題方針に合っているかあまり確認しないためと予想される. 手間の数値から問題の難易度が上がるほど手間を軽減できたことが実感しやすいことが分かった.

4.3 まとめと今後の課題

本論文では読解問題を演習で使用する際の問題を解決するために, プログラミング学習のためのソースコード読解問題類題生成システムを提案し, そのプロトタイプシステムを作成した. 作成したプロトタイプシステムでは問題として出したいソースコードの解析を行い, 解析結果を用いて設定可能箇所を問題追加者に提示する. 問題追加者は提示された個所から出題する個所を選択し, その内容を入力することで類題設定ファイルを作成する. システムがその類題設定ファイルを解析することで類題を生成し, 受講者が指定した課題に関連付けされている類題をランダムに表示することで問題要求のたびに違った問題が表示されるようにした. 実装したプロトタイプシステムを用いて実際に類題設定ファイルの作成や類題の生成を行ったり, 既存システムと比較することでその有用性を確認した. 今後システムを実用化するために, より扱いやすくするようインタフェースを改善したり, 生成する類題の範囲を広げるため初期値を変える以外の書き換え方法を考案する必要がある. また, 提案システムを実際に演習で用いることで類題を用いた読解問題の演習効果や, 受講者の習得度に応じた問題を出せるようにシステムを改善していきたいと思う.

参　考　文　献

1) 中島秀樹, 高橋直久, 細川宣秀：プログラミング演習のための QA サイクル - 受講者の習得度に応じた問題提示メカニズム - . 電子情報通信学会論文誌,VOL.J88-D-I,NO.2, pp439-450(2005).

2) 吉田拓己，立岩佑一郎，山本大介，高橋直久:タグにより出題方針指定可能な読解問題類題生成システムの実現，平成 22 年度電気関係学会東海支部連合大会，教育支援 F3-4 (2010).

3) 岩間信介，立岩佑一郎，山本大介，高橋直久：プログラミング演習支援システムにおける実行履歴の構造化方式，第 1 回データ工学と情報マネジメントに関するフォーラム (DEIM2009)，2009.