様式Ｃ－１９

科学研究費助成事業（科学研究費補助金）研究成果報告書

平成 24 年 5 月 18 日現在

機関番号：３２６０１
研究種目：基盤研究（C）
研究期間：2009～2011
課題番号：21500905
研究課題名（和文）学生のプログラミング能力の向上を目的としたプログラム可読性チェックシステムの開発
研究課題名（英文）Development of a System for Evaluating Program Readability designed to Improve Students' Programming Competence
研究代表者
M・J Duerst（M・J DUERST）
青山学院大学・理工学部・教授
研究者番号：30383920

研究成果の概要（和文）：現在，大学におけるプログラミング教育では可動性が優先され可読性は軽視されている．だが可読性が軽視されたプログラミングは動作への理解を阻害し，学習効果を低下させる可能性がある．そこで本研究では，「読みやすいソースコードを書かなければいけない」と学生に意識させるために，可読性が低いコードに対して修正指示を出すシステムを設計・開発した．さらに，システムによって適切な指摘が行われているかを検証するための評価実験を行った．また，本システムによる教育効果を明らかにするため，大学のプログラミング教育における可読性の影響の調査・分析を行った．

研究成果の概要（英文）：In an university setting, programming education tends to focus on correct program execution, neglecting program readability. This in turn can inhibit program understanding. We therefore proposed, designed, and implemented a system that encourages students to write readable programs by indicating difficult-to-read locations. We also verified the correct functioning of the system. In addition, to evaluate the effectiveness of this system, we experimentally analyzed the relationship between program readability and program understanding.

交付決定額

（金額単位：円）

	直接経費	間接経費	合　計
２００９年度	1,000,000	300,000	1,300,000
２０１０年度	600,000	180,000	780,000
２０１１年度	700,000	210,000	910,000
年度			
年度			
総　計	2,300,000	690,000	2,990,000

研究分野：総合領域
科研費の分科・細目：科学教育･教育工学・教育工学
キーワード：プログラミング教育・可読性・C 言語

１．研究開始当初の背景

申請者らは，大学で学生のプログラミング教育に携わっている．情報テクノロジー学科 2 年生を対象とした，C 言語の基礎を教える授業を担当しており，毎年 100 名ほどの学生が履修している．この授業では，毎週，数種類の課題を出題しているため，教員のもとには毎週数百のプログラムが提出されることになり，すべてを目視で詳細にチェックすることは非常に大変である．そこで現在は，提出された課題プログラムを自動チェックするシステム（図 1，図 2）を構築し，平成 20 年度より運用している．これは，プログラムが正しく動くことを確認するもので，学生が

プログラムをウェブブラウザ上からアップロードすると，システムがコンパイルおよび動作確認のテストを行う．実行結果が正しくない場合，学生は再提出を求められるので，受理されるまでプログラムの修正とアップロードを繰り返す．このシステムを導入した結果，以下の効果が得られた．

- これまで学生は，コンパイルエラーや実行時エラーの対処を行うことに注力し，動作確認を怠る傾向にあった．それが（教員が目視で動作確認を行う場合と比較して）多様なテストケースに対し瞬時に動作確認が実施されるようになったことにより，テストの重要性に対する意識が向上した．
- 教員がプログラムを機械的にチェックする手間が大幅に軽減され，その分，人間でなければできない指導に割く時間が増えた．

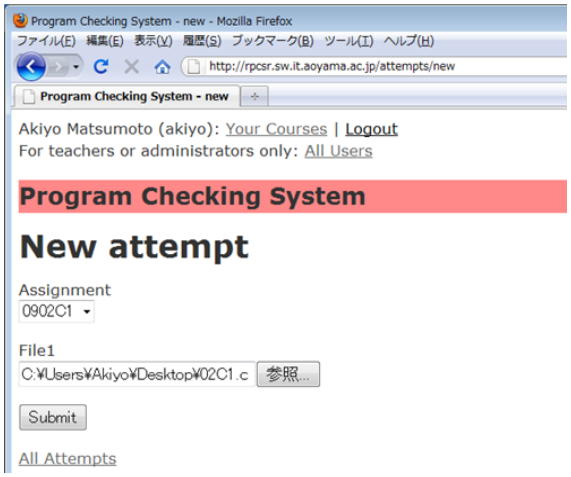


図 1：プログラム提出システム（提出画面）

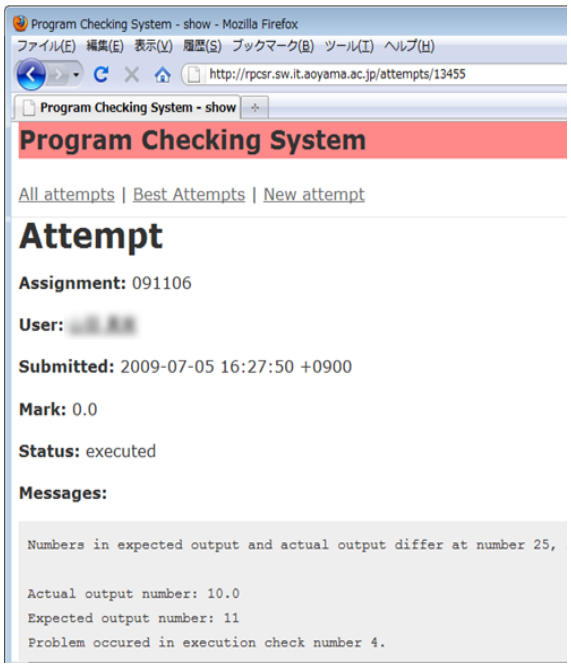


図 2：プログラム提出システム（結果画面）

しかしながら，この動作チェックシステムには限界がある．「美しいソースコードを記述する」ことに関しては強制力が働かないため，学生は「正しく動きさえすればよい」という観点でプログラムを作りがちとなる．教員がいくら「他人（もしくは数ヶ月後の本人）が読みやすいプログラムを書かなければいけない」と注意しても，なかなか徹底できないのが現状である．しかしながら，「可読性が高いソースコードを記述すること」はプログラムの構造を正しく理解することにつながる．また，大規模なプログラムを大人数で書く場面においても，可読性はプログラムの修正・機能拡張を容易にし，再利用性を高める意味で大変重要である．そこで，我々は「読みやすいソースコードを書かなければいけない」と学生に意識させるシステムを開発する．

２．研究の目的

学生のプログラミング能力を養うことを目的として，プログラム課題の提出システムに，可読性の低いコードに対して修正指示を出す機能を組み込む．たとえば，

- 崩れたインデント
- 空白・空行の不足
- コメント量の不足
- 長すぎる関数やブロック
- 関数化すべきコピー＆ペースト
- 不適切な変数・関数名
- 深すぎるネスト構造

など，構文エラーでもなければ実行結果にも影響を及ぼさないが可読性が低いと判定できる項目について，検出できるようにする．

ここで大切なことは，システムが「親切に教えすぎない」ことである．「どのように修正すればよいか」については学生自身に考えさせることにより，学生のプログラミング能力向上を図る．教師にとっては正解（具体的な修正案）を教えてしまった方が，その場では時間も手間も少なくて済む．しかし，その指導内容がなかなか学生の身につかず，学生は同じ誤りを繰り返してしまう．これでは学生にとってプラスにならないのは勿論のこと，同じ指導を繰り返し行わなければならない教員にとっても不幸である．そこで我々は，プログラミング能力の育成に結びつくようなシステムの設計・開発を目指す．

本システムは，プログラミングの演習授業において，可読性について各学生に対し均質な指導を行うことが可能であり，有意義である．さらに教員の余計な負担を軽減し，教員はシステムでは対応できない本質的な指導や個別対応等に，より多くの時間を費やせる，という効果も期待できる．学生，教員の双方にとって役立つシステムであるといえる．

３．研究の方法

(1) プログラミング教材の実態調査

実際に SE・プログラマを育成している企業・学校において利用されているプログラミングの教材を調査する．なお，大学でのプログラミング教育と，即戦力としてプログラマを育成する場合の教育は，位置づけが異なるものであり，内容も同等である必要性は無いが，実態調査としては幅広く行っておく方が望ましいと考えている．

(2) 指摘項目の検討

先述の指摘項目（研究開始前段階の案）に加え，他機関のプログラミング教育の実態調査や，他の教育者からの聞き取りレビューなどを踏まえ，指摘項目を改めて検討する．

(3) システムの実装

検討した指摘項目を検出するシステムを実装する．システムは「課題提出」としての機能も兼ね備えたものとし，学生の自宅からも利用できるよう利便性を考慮してウェブアプリケーションとして構築する．

(4) 評価実験

システムの実装後には動作検証を行い，実装したルールが正しく検出できることを確認する．その結果を踏まえ，必要に応じてシステムの改善を行う．

(5) システムの運用

構築したシステムを授業で実際に運用する．

(6) 可読性の教育効果の検証

「プログラムの可読性を重視することが，他にどのような良い影響をもたらすか」などの教育的観点から本システムの有効性を明らかにする．

４．研究成果

(1) システムの実装

学生がプログラム課題を提出するウェブアプリケーションシステムに，可読性が低いコードに対して修正指示を出す機能を追加実装した．本システムの可読性に関する検出項目は，①崩れたインデント，②関数化すべきコピー＆ペースト，③不適切な変数・関数名，④空白の不足，⑤長すぎる関数，の5項目である．構築後には第三者による動作検証を行い，実装したルールが正しく検出できることを確認した．

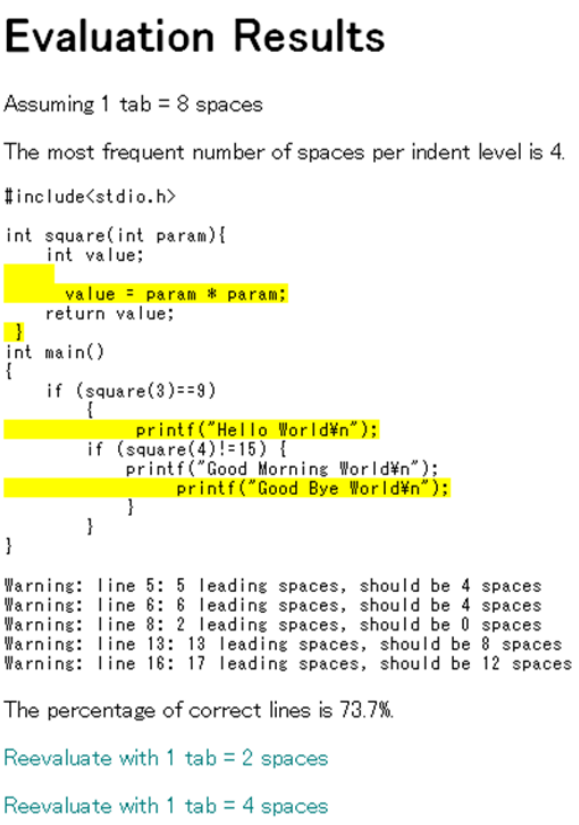
① 崩れたインデント

以下の手法で崩れたインデントを検出し出力する．

まず，プログラム中のインデント量を取りだす．その際に，空白とタブのどちらがどれだけ使用されているかを評価する．タブが環境によってインデント量に変化を起こすことを考慮に入れて，複数の量の空白に変換させる．本研究では昨今の主流である，1タブが8空白のときと4空白のときと，2空白のときの計算を行う．タブをそれぞれの量の空白に変換させて，インデント量とする．そして，それぞれのインデントにおいて評価を行う．1タブにおける空白の量の違いにより，複数の結果が出力される．そしてその中で最もインデントの統一性が高かったものを選択し表示する．また，そのほかの結果がどのようなものであったかも表示できる．

続いて，関数や制御構造などの字下げが行われる個所を発見し，その深さを評価する（以降，字下げの深さをレベルという）．そして，1レベルごとのインデント量の統一性を評価する．その後，他の行とインデント量が異なり，その統一性を乱していると判断された行は注意が必要と評価される．最後にプログラムの中で，注意の必要がない行が含まれる割合を計算する．この割合が高いほど統一性のあるプログラムだと評価できる．

出力例を図3に示す．まず，1タブが何空白の場合に最も評価が高かったかを表示する．プログラム中の1レベルにおける空白の値の最頻値を表示する．次に評価されたプログラムの本文を表示する．この際，注意された行の背景色を変化させて直感的に読み取れる



Evaluation Results

Assuming 1 tab = 8 spaces

The most frequent number of spaces per indent level is 4.

```
#include<stdio.h>

int square(int param){
    int value;

     value = param * param;
    return value;
}
int main()
{
    if (square(3)==9)
        {
            printf("Hello World¥n");
        if (square(4)!=15) {
            printf("Good Morning World¥n");
                printf("Good Bye World¥n");
            }
        }
}
```

```
Warning: line 5: 5 leading spaces, should be 4 spaces
Warning: line 6: 6 leading spaces, should be 4 spaces
Warning: line 8: 2 leading spaces, should be 0 spaces
Warning: line 13: 13 leading spaces, should be 8 spaces
Warning: line 16: 17 leading spaces, should be 12 spaces
```

The percentage of correct lines is 73.7%.

Reevaluate with 1 tab = 2 spaces

Reevaluate with 1 tab = 4 spaces

図3：インデントの評価出力例

ようにする．さらに，各行に対して理想とするインデント量を表示する．

② 関数化すべきコピー＆ペースト

類似処理を繰り返し行うならば，本来関数にすべきであるが，初心者はコピー＆ペーストを行って変数名などの相違点を修正する，といった安易な作り方をしてしまいがちである．そこで，“Don't Repeat Yourself”の考え方に基づいたプログラムの作成を促すため，このような「関数化すべきコピー＆ペースト（図4）」を検出する．

研究分担者の松本は，ウェブページの中から反復構造を検出する研究に携わっている．そこで用いている配列アラインメントという手法を本システムでも利用し，ソースプログラムの中から関数化すべきコピー＆ペースト，すなわち反復構造を抽出する．

③ 不適切な変数・関数名

変数や関数を宣言する際，プログラムの内容を見なくても変数名だけで想像できることが理想である．そのため，1 文字のみ，または同一文字を繰り返しただけの変数名・関数名など，不適切だと言えるものには警告を出す．ただし，例外として座標として用いられている可能性の高い`x, y, z`や，ループ変数として用いられている場合の`i, j, k`を除く．

④ 空白の不足

空白の入れ方のルールは，K&R（プログラミング言語C）に準ずる．詳細は，以下のとおりとする．

- カンマ・セミコロンの後には空白または改行，前には空白を入れない
  例）`int x, y;`
- 演算子の前後には原則として空白
  例）`x = a + b;`
- 【例外】符号の直後には空白を入れない
  例）`a = -b;`
- 【例外】インクリメント・デクリメント演算子と変数の境界には空白を入れない
  例）`i++; ++j;`
- 中括弧・小括弧は外側に空白を置き，内側には空白を入れない
  例）`for (i = 0; i < 10; i++) {`
- 【例外】関数名の直後の小括弧は外側に空白を入れない
  例）`printf("¥n");`
- 配列の角括弧は外側・内側とも空白を入れない
  例）`int x[10];`
- `#include`と`<`の間には空白
  例）`#include <stdio.h>`

これらのルールに違反しているものを検出し，警告を出す．

```
12      for (i=0; i<NUM; i++) {
13          sum[0] = Eng[i];
14          sum[1] = Math[i];
15          total = 0;
16          for (j=0; j<2; j++) {
17              total = total + sum[j];
18          }
19          personalAve[i] = total / 2;
20      }
21      total = 0;
22      for (i=0; i<NUM; i++) {
23          total = total + Eng[i];
24      }
25      engAve  = total / NUM;
26
27      total = 0;
28      for (i=0; i<NUM; i++) {
29          total = total + Math[i];
30      }
31      mathAve  = total / NUM;
```

個人ごとの平均点

英語の平均点

数学の平均点

図4：関数化すべきコピー＆ペーストの例

⑤ 長すぎる関数

1つの関数の長さは，画面に収まる程度，もしくはA4用紙1枚で収まる程度で，60行程度以内が理想，限界でも100行以内とされている．そこで，長さが60行を超える関数がある場合には，分割するように警告を出す．

(2) 可読性の教育効果の検証

本システムによる教育効果を明らかにするため，大学のプログラミング教育における可読性の影響の調査・分析を行った．我々は，プログラムの可読性に関する様々な要素が学習そのものと密接な関係があるという仮説を立てた．検証のために大学のプログラミング実習の2年分・約7000本のプログラムから，学生の習熟度と可読性に関する 25の要素を抽出した．具体的には，習熟度に関しては，

- 課題の得点
- 課題の提出時間
- 試験の成績

可読性に関しては，

- インデント
- 行数・バイト数
- 関数名・変数名
- 空白
- 空行
- コメント

といった項目である．得られたデータを解析し，プログラミング教育における可読性の影響を検討した．

① 解析手法

収集したプログラムのデータは学生ごとにまとめて，可読性に関するデータをプログラムあたりの平均に変換する．そのうえで学習の習熟度に関する様々な要素の多寡によって学生のグループ分けを行う．そして優秀な集団とそうでない集団に分けて，その間にある可読性に関する要素の差を検定する．その結果から優秀な集団に見られる特徴を抽

出すれば，それが可読性に関する要素と学習の習熟度の関係性を導き出せる．

② 解析結果および考察

解析結果から，優秀な成績の学生のプログラムには，可読性に関するいくつかの特徴が確認された．変数名には英単語を用いる，十分な空白・空行で余裕をもった記述を心掛ける，などのスタイルが習熟度には良い影響を与えることがわかった．十分な量の空白類がプログラムを読みやすくし，構造の理解に有効であるためだと考えられる．また，インデントの統一性については特徴的な強い関係性を検出した．この要素は課題の得点とは負の相関を持ち，最終試験の得点や実習の成績とは正の相関を持っていた．課題の得点は，正しく動作するものが提出期限内に提出されたかどうかに大きく依存する．つまり，「正しく動作するプログラムを作成するだけならインデントに気を使うことは重要ではないが，それではプログラミングの知識・能力に結びつかない．時間をかけてでもインデントの統一性に気を配ってプログラムを作成したほうが力が身につく．」と考えられる．

以上の解析結果より，変数名，空白類，インデントといった可読性に気を配る学生は，優秀な成績を収めていることが判明し，本システムの有効性が実証された．

５．主な発表論文等

（研究代表者、研究分担者及び連携研究者には下線）

〔学会発表〕（計７件）

① 松本翔太，松原俊一，Martin J. Dürst：
大学のプログラミング教育における可読性の影響の調査，
情報処理学会第 74 回全国大会，2012 年 3 月 8 日，名古屋工業大学

② 新開 崇弘，松本章代：
プログラム課題提出システムにおける可読性チェック機能の開発，
平成 23 年度 第 6 回情報処理学会東北支部研究会，2012 年 2 月 14 日，東北学院大学

③ 浅倉 一哉，松本章代：
プログラム課題提出システムにおける剽窃検出機能の開発，
平成 23 年度 第 6 回情報処理学会東北支部研究会，2012 年 2 月 14 日，東北学院大学

④ 松本章代，Martin J. Dürst：
可読性の指摘を行うプログラミング教育システムの開発 ― 反復構造の自動検出による関数化の指摘 ―，
情報処理学会 第 73 回全国大会，2011 年 3 月 3 日，東京工業大学

⑤ 本間皇成，松本翔太，松本章代，松原俊一，Martin J. Dürst：
チームプログラミングを可能とした教育支援アプリケーションの開発，
情報処理学会 第 73 回全国大会，2011 年 3 月 3 日，東京工業大学

⑥ 中村一星，松本章代，Martin J. Dürst：
教育支援を目標とした遠隔ペアプログラミング環境の構築，
情報処理学会 第 72 回全国大会，2010 年 3 月 11 日，東京大学

⑦ 松本翔太，松本章代，Martin J. Dürst：
C 言語用のプログラミングスタイル評価システムの構築，
情報処理学会 第 72 回全国大会，2010 年 3 月 11 日，東京大学

〔その他〕

① 研究内容紹介用 Web ページ
http://mmt1.cs.tohoku-gakuin.ac.jp/research/readability.html

② 開発したシステム
http://rpcsr.sw.it.aoyama.ac.jp/

６．研究組織

(1)研究代表者

M・J Duerst（M・J DUERST）
青山学院大学・理工学部・教授
研究者番号：30383920

(2)研究分担者

松本　章代（MATSUMOTO AKIYO）
東北学院大学・教養学部・講師
研究者番号：40413752