C言語のプログラミングスタイル評価システムの構築

松本 翔太，松本 章代，Martin J. Dürst
青山学院大学理工学部情報テクノロジー学科

1　はじめに

大学におけるプログラミングの教育現場では，可動性が重要視されるあまり，可読性が軽視されがちな傾向が見られる。課題で作成されたプログラムを再利用したり，複数名で協力してプログラミングを行うこともめったにない。しかし実際の社会におけるプログラミングの現場では，複数名で一つのプログラムを組んだり，一度作成したプログラムを後で拡張したりすることが多い。そうなると，プログラムを見たときにその構造や機能がすぐに理解できなければならない。よって可読性の高い，見やすいプログラムを作成することは，作業の効率化の面からみても非常に重要といえる。プログラミング教育において，その可読性の重要性を身につけさせることは大事なことである。

本研究室はこれまで，学生が作成したプログラムの可読性に関する評価および可読性に対する意識の向上を目的とした研究を行ってきた。スペースの入れかたや改行の入れかたなどを解析する研究 [1] や，オープンソースと学生が記述したソースファイルのコメント文や空白類について実態調査を行う研究 [2] である。今回本研究はインデントの統一性に注目している。どのようなスタイルに対しても，そのインデントの変化を測定して，統一性の評価を行うシステムを構築する。このシステムを使用することにより，インデントの統一性を意識したプログラム作成の促進を目的としている。

2　関連研究

2.1　プログラミング教育の支援に関する研究

関本らのプログラミングスタイルの診断システム [3] では，条件文に代入文を含めるなどの式やぶら下がりelse問題など，構文に関するガイドラインに焦点を当て，プログラムの中からこのようなまぎらわしいパターンの検出を行った。プログラミングスタイルの様々な悪形パターンを検出することにより，初心者がしがちな顕在化していない間違いを見つけることもできる。本研究は構文に関する可読性よりも，インデントなどによる見た目の可読性に焦点を置いて，その統一性を評価している。

2.2　パターン認識によるバグ検出に関する研究

小田らのパターンマッチングに基づいたCプログラムの落し穴検出法 [4] では，構文上のささいな相違によって発生する，コンパイラによって見つけられないバグを落とし穴と名付け，正規表現を用いたパターンマッチングで検出を行った。また，発見した落とし穴を即座に修復できるソフトウェアツールを開発した。本研究は，バグの検出ではなく可読性の向上によりバグの発生しにくいプログラム作りを目指す。またバグが発生したとしても，可読性の高いプログラムなら発生個所の特定もそれほど難しいことではないであろう。

3　プログラム中のインデント

3.1　スタイル

インデントの使用は可読性に関する要素の中で最も基本的な技法である。空白を用いてプログラムの構造を分かりやすくするために行う。特に条件分岐や繰り返し処理などの制御構造の範囲を明示させるために使用される。基本的な字下げのスタイルにはK&Rスタイルや BSD/オールマン スタイルなどがある（図1参照）。C言語には実に多くのスタイルが存在し，特に規定されたスタイルは無い。しかし，同じプログラム，もしくは同じプロジェクト内ではスタイルを統一することが望ましい。

K&R スタイル

```
void style (...)
{
    if (a > b) {
        c = a;
    } else {
        c = b;
    }
}
```

BSD/オールマンスタイル

```
void style (...)
{
    if (a > b)
    {
        c = a;
    }
    else
    {
        c = b;
    }
}
```

図 1: C言語の基本的なスタイル

Construction of a Programming-Style Evaluation System for the C Language
Shota Matsumoto, Akiyo Matsumoto and Martin J. Dürst
Department of Integrated Information Technology, College of Science and Engineering, Aoyama Gakuin University
5-10-1 Fuchinobe, Sagamihara, Kanagawa 229-8558, Japan
matsu@sw.it.aoyama.ac.jp, {akiyo, duerst}@it.aoyama.ac.jp

3.2 スペースとタブ

字下げには基本的にスペースとタブが使用される。それぞれに長所と短所がある。タブは環境によって 1 タブのインデントの量が変化してしまう危険性を含んでいる。使用するときは開発環境に気をつけるべきである。インデント量が多いときはスペースのみの使用は手間と言える。しかし細かい調整はスペースのほうが向いている。

4 プログラミングスタイル評価システム

4.1 概要

本システムは，まずプログラム中のインデント量を取りだす。その際に，スペース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，複数の量のスペースに変換させる。本研究では昨今の主流である，1 タブが 8 スペースのときと 4 スペースのときと，2 スペースのときの計算を行う。タブをそれぞれの量のスペースに変換させて，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 タブにおけるスペースの量の違いにより，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可能にする。

続いて，評価方法について述べる。まず，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。

4.2 結果表示形式

結果の表示は HTML 形式である（図 2）。まず，1 タブが何スペース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 レベルにおけるスペース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。

5 教育効果

学習者はこの評価システムを利用して，評価結果からプログラムの可読性の高さを判断し，また，自らの環境にあったインデントが行われているかを判断できる。その結果を参考に，より可読性の高いプログラムにすべく修正作業を行える。それにより，学習者はプログラムを作成する段階で，可読性を意識した作業を行うことができると思われる。

Evaluation Results

Assuming 1 tab = 4 spaces

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

```
#include<stdio.h>

int calc(int a,int b){¥n

    for (a = 1; a <= 9; a++)
   {
              for (b = 1; b <= 9; b++) {
        printf("%d x %d = %d¥n", a, b, a*b);
      }
    }
}
```

```
Warning: line 4: 1 leading space, should be 4 spaces
Warning: line 6: 3 leading spaces, should be 8 spaces
Warning: line 8: 10 leading spaces, should be 12 spaces
Warning: line 9: 6 leading spaces, should be 8 spaces
```

The percentage of correct lines is 63%.

Reevaluate with 1 tab = 2 spaces

Reevaluate with 1 tab = 8 spaces

図 2: 結果表示

6 おわりに

本研究ではプログラムのインデントの統一性を評価するシステムを構築した。今後はこのシステムを使用することにより，可読性に対する意識は向上するか，さらにその意識の向上は，プログラミング学習の効率の上昇につながるかを調査する予定である。

謝辞

本研究は文部科学省科学研究費補助金（基盤 C，課題番号 21500905）の交付を受けている。

参考文献

[1] 中島正登, 伊藤一成, Martin J. Dürst. C 言語プログラミングにおけるフォーマット特製の抽出. 情報処理学会第 69 回全国大会, 2007.

[2] 宮島明寛, 松本章代, Martin J. Dürst. オープンソースを用いた C 言語の記述スタイルの統計分析の試み. 情報処理学会第 71 回全国大会, 2009.

[3] 関本理佳, 海尻賢二. プログラミングスタイルの診断システムの構築. 教育システム情報学会誌, Vol. 17, No. 1, pp. 21–29, 2000.

[4] 小田まり子, 掛下哲郎. パターンマッチングに基づいた C プログラムの落し穴検出法. 情報処理学会論文誌, Vol. 35, No. 11, pp. 2427–2436, 1994.