

ソースコードの可視化によるプログラミング学習支援

下原 康貴[†] 松浦 佐江子[‡]

芝浦工業大学 システム工学部 電子情報システム学科^{†‡}

1. はじめに

本研究では、学部において初めてプログラミング言語（C 言語）を学習する演習「情報処理演習Ⅱ」の履修学生を対象とし、以下の目標を達成するためのツールを開発し、実験したので実験結果を報告する。

- (1) デバッグコードを何処に挿入すればいいか理解する事。
(デバッグコード: プログラムの処理遷移を確認するコード)
- (2) 変数の遷移, プログラムの処理順序を理解出来る事。
- (3) 実行結果からソースコードの何処に問題があるのか理解出来る事。
- (4) 情報処理演習Ⅱレベルの課題が自力で解ける事。

2. ツール

2. 1 ツール概要

C 言語のソースコードを解析し、その解析情報を基に変数への代入が行われた場所を発見し、自動でデバッグコードを埋め込んだソースコードを出力する。これをコンパイル実行すると、プログラムの処理順序に沿って、その時点での変数の値を図 1 のような形で標準出力から出力する。これを連続して出力する事で、学習者に変数の遷移, プログラムの処理順序を理解させる。この時、学習者はツールが提供する機能を選択し、実行時の情報の絞込みや、出力のタイミングの調整をするなどして、実行結果を変える事で理解を進める。

2. 2 学習者用ソースコード生成

ツールに入力するソースコード、挿入するソースコード、挿入箇所の例を図2に示す。図2の左側のソースコードがツールに入力するソースコードの例である。ツールはソースコードの開始時にデバッグコードが動作するために必要なソースコードを挿入する。次に、関数の開始部分に学習者が機能を選択するための機能選択スイッチを挿入する。変数の代入を読み取った場合に、その時点での変数の値を一覧で表示するためのデバッグコードを挿入する。

```
↓ 変数遷移 from line 18 ↓
i = 10
tmp = -2147483648
data[0] = 1.000000
data[1] = 2.000000
data[2] = 3.000000
data[3] = 4.000000
data[4] = 5.000000
data[5] = 6.000000
data[6] = 7.000000
data[7] = 8.000000
data[8] = 9.000000
data[9] = 10.000000
↑ 変数遷移 from line 18 ↑
```

図 1 コマンドラインからの出力

A Learning Support Tool for Programming Beginners based on Source Codes Visualization.

[†] Yasutaka Shimohara [‡] Sacko Matsuura

^{†‡} Shibaura Institute of Technology Department of Electronic Information Systems

```
#include <stdio.h>

int main( void )
{
    int i=0;
    float tmp;
    float data[10];

    for ( i=0; i<10; i++ ) {
        printf( "第%d 番目のデータを入力: ", i );
        scanf( "%f", &data[i] );
        printf( "データ出力\n" );
        for ( i=0; i<10; i++ ) {
            data[i]=data[10-i-1];
        }
        printf( "data[%d] is %f\n", i, data[i] );
    }
    return 0;
}
```

```
#define ON 1
#define OFF 0

/* 機能選択スイッチ */
int Array_Loop_Number;
int output_sw_all = ON;
int stop_sw = OFF;
int sleep_time = 0;
int output_sw_i = ON;
int output_sw_tmp = ON;
int output_sw_data = ON;

/* デバッグコード */
if(output_sw_all==ON) {
    printf( "\n\n ↓ 変数遷移 from line 14 ↓ \n\n" );
    if(output_sw_i==ON) {
        printf( "i = %d\n", i );
    }
    if(output_sw_tmp==ON) {
        printf( "tmp= %d\n", tmp );
    }
    if(output_sw_data==ON) {
        for(Array_Loop_Number=0; Array_Loop_Number<10; Array_Loop_Number++) {
            printf( "data[%d] = %f\n", Array_Loop_Number, data[Array_Loop_Number] );
        }
        printf( "\n\n ↑ 変数遷移 from line 14 ↑ \n\n" );
        sleep(sleep_time);
    }
}
```

図 2 学習者用ソースコード生成

2. 3 ツールの機能

ツールが提供する機能は、変数選択、スロー再生、一時停止の3つである。変数選択は、学習者が表示しなくても良いと判断した変数の表示スイッチをOFFにする事で、その変数の値が表示されなくなる。例えば、図2の機能選択で、output_sw_i=OFFとすると、図1の表示からiに関する表示が消える。スロー再生は、同じく機能選択にて、sleep_time=1 として、コンパイル実行すると、1つ1つの表示の間で1秒間待つようになる。一時停止は、stop_sw=ONとする事で、ENTERが入力されるまで次の表示に移らないようになる。

3. 実験結果

本実験では二人の被験者に対して、実験を行うが、その際に、以下の事に気を付ける。

- (1) 学習者が自発的に質問をした場合のみ答える。
- (2) 教える時は正しいプログラムを教えるのではなく、学習者の意図通りのプログラムを実現する方法を教える。
- (3) 極力言葉による解説は避け、ソースコードに `printf` 文を挿入し、どういうエラーが起きているのか理解させる。

3. 1 実験手順

- (1) 中間試験 過去問題実施 + 同時に antlr[1]などツール実行環境インストール
- (2) ツールを用いながら情報処理Ⅱ演習問題復習(去年の課題を1から順番に解く)
- (3) 目標の個別評価テスト

1で挙げた目標について、個別にテストを行う。

目標1に対しては、エラーがあるソースコードを用意し、学習者に実行結果とソースコードを見せ、何処にデバッグコードを挿入すればバグの除去に役立つかを考察させる事で評価を行う。

目標2に対しては、学習者にソースコードを読ませ、実行結果を予想させる事で評価を行う。

目標3に対しては、エラーがあるソースコードを用意し、学習者に実行結果とソースコードを見せる。何処を直せば期待している結果が得られるかを考察させる事で評価を行う。

目標4に対しては、情報処理Ⅱ演習の予習を行い、自力で学習が出来るかどうか判定する。

- (4) 中間試験 過去問題、期末試験 過去問題実施
- (5) アンケート実施

表1. 実験所要時間

	被験者A	被験者B
課題3	3時間	4時間
課題4	3時間	6時間
課題5	2時間半	4時間
課題6	3時間	8時間経過
課題7	4時間半	
課題8	6時間	
課題9	4時間	
課題10	2時間経過	

表2. 実験所要時間(2回目)

	被験者A	被験者B
課題4		4時間

表3. 第一回中間試験 過去問題 結果

	被験者A	被験者B
問題1	4問中3問正解	4問中2問正解
問題2	3問中1問正解	未解答
問題3	未解答	未解答
問題4	未解答	未解答

3. 5 学習者の感想

被験者Aの意見

- ・ 今回の実験をやる前は問題を見ても手が動かなく、何もする事が出来なかったけど、まず出来るところからプログラムを書いていき、わからなかったとこのプロ

グラムを見直す事で徐々に力がついた。

被験者Bの意見

- ・ 課題3では数学の知識が必要などもあり、出来るのに時間がかかった。if構文を2回以上繰り返す時に頭が混乱した。(課題3)
- ・ ifやwhile構文を一緒に使う部分を考えるのに時間がかかった。(課題4)
- ・ 解くのは2回目だったけど、やったことを忘れていた。今回は自分力で出来るようになった。課題がどんどん進んでいっても、最初の方で使った構文を利用するのでだんだん慣れてきた(課題4 2回目)

4. 考察

実験を行っていく上で、ツールには多くの欠点が見つかった。第1に、ツールによって生成したソースコードの機能を選択するために、学習者はソースコードを読まなければならない。図1のソースコードを入力した結果出力されるファイルは140行になってしまい、その長さがやる気の喪失と、機能選択スイッチのを見つけ辛さを招いた。

この問題に対しては、最も使用頻度の高かった `sleep_time=1` をデフォルトの設定にしておき、ソースコードをツールにかけた時に、生成されたソースコードを読まずに済む確立を高くする。

第2の欠点として、変数が変化した部分のみに対してデバッグコードを挿入していたが、その結果出力される情報ではプログラムの実行経路が見えないという問題がある。この問題に対しては、デバッグコードは全ての行に対して埋め込み、変数が変化した部分のみを検出するためのスイッチを用意する事で、誤解を防ぎ、表示内容の絞り込みにも対応する。

第3の欠点として、ソースコード作成の際にツールを用いていたが、本ツールは変数の遷移や処理順序が正しいか確認する場合のみに有効である。学習者はコンパイルエラーの時に使用してしまうという問題が起こった。

この問題に対しては、変数の遷移や処理順序を確認し、プログラムを修正する問題を別途設定し、その問題に対してのみツールを用いて取り組む。

第4の欠点として、変数の変化を読み取る部分が十分ではなかった。再帰的处理の中で、`return` 文のみで実現するようなアルゴリズムのエラーを発見出来なかった。

この問題に対しては、代入が行われていなくても計算が行われていれば表示するという方法を検討する。

5. まとめ

最初に被験者Bが課題4に取り組んだ時は、6時間掛かったのに対して、課題6学習後、再度課題4に取り組んだ際には4時間で済んでいる。また、1回目よりも2回目の方が質問の回数も減っており、自力で解けているため、上達しているのは明らかである。このように、経験を積み重ね、プログラミングが上達する事が解った。

よってツールは彼らが経験を積むための支援が出来ればよい。今後は、ツールが学習者の疑問を解消する事が出来るか、ツールを用いた学習によって質問の数が減少するか、の2点について調査を行う。

また、4で挙げた内容で改善した状態で、改善前の状態でツールとしての使い勝手の違いについてアンケートを行う。

6. 参考文献

- [1] Antlr: <http://www.antlr.org/>