プログラミングの授業におけるアクティブラーニング

A trial of the active learning in the subject of “Programming”

森　文彦

Fumihiko Mori

玉川大学工学部情報通信工学科，　194-8610　東京都町田市玉川学園6-1-1

Department of Information & Communication Technology, College of Engineering, Tamagawa University,

6-1-1 Tamagawagakuen Machida-shi Tokyo 194-8610

Abstract

Computer programing is one of subjects which is easy to stumble because it does not work flexibility. To overcome this difficulty we introduced an active learning process of 1) programing and execution of the simple fundamental program , 2) iterative writing the program , and 3) confirming by a short test. These successful experiences look like leading the students to a level of above the pass in the first stage of programing technology.

Keywords: active learning, programing, fundamental program,

1．はじめに

コンピュータが出現してから約70年．今では，あらゆる所で使われており，コンピュータ無しでは生活できないほど必須ものとなっている．このコンピュータを動かしているものが「プログラム」である．この「プログラム」を作ることが「プログラミング」である．許された厳格なルールに従って書かれたプログラム以外はコンピュータを正しく動かすこと（有益な仕事をしてもらうこと）はできない．全く融通の効かない堅物なのである．記号“;”を１つ欠けても全く動かない．単純かつ簡単なものほど人間には気づきにくいのが人間の特性である．そのため，プログラマーを目指す専門家の卵であっても，適切な指導がないと初期の段階では，つまずき易く，プログラマーになることを諦める人も少なくないといわれている．

このような難行を克服するための１つの対処法のいくつかを以下に述べる．

① 優しい基本パターンを身に着け自信をつけること．

・単純なプログラムの作成と実行の体験

・この時の手順（基本パターン）を正しく記録して覚える（半年繰り返すと自然に見につく）

・命令文・宣言文などプログラムの構成要素も実際に動くプログラムの中で覚える

② 命令文や配列なども動作の仕組みとともに実プログラムの中で覚えること．

③ プログラムが正しく動作することの確認．

まとめるとプログラミング言語を初めて学習する場合は，確実に理解すべき部分を「確実に理

解する」必要がある．これを順番に確実に積み上げることによって，より発展的な学習が可能となる．プログラミングの教育において，論理的思考の訓練は必須の要素である．この訓練のために，「暗記」は不要，もしくは弊害があるとする意見があるが，「確実に理解する」ためには大変有効である．

プログラミングで躓く要因の一つは「一文字でも誤ると動かない」ということである．確実に理解し，ミスを減らすことや，ミスを発見するためには，「正解の確実な理解と記憶」が不可欠である．この過程を欠くとプログラミングを苦手と感じて，プログラミングにアレルギーやトラウマを感じるようになっていくようになる．

プログラミングの学習に理解の伴わない「暗記」のみを適用すると，プログラミング技術や論理的思考の訓練に悪影響を及ぼすことが考えられるが，初歩の段階では，「確実に理解する」必要がある部分について，「何度もプログラム書く」という筋肉的運動を伴う「暗記」（体で覚える）は非常に有効であり，重要である．

ここでは，プログラミングの最も初歩を学ぶ授業において，「暗記」の要素に重点をおいたアクティブラーニングを紹介する．

2. 授業内容

工学部機械情報システム学科の1年生はプログラミングの入門の授業として，「プログラミングⅠ」を受講する．その授業内容を表１に示す．

プログラミング言語はC#を用いる．授業では，MyPC（各学生の所有するノートPC）とテキストを使用する．講義資料やテキストは確実に理解すべき部分の必要最低限の内容とした（文献1）．

授業の流れを図１に示す．その概要を以下に述べる．①説明を聴いた後，「重要な例題プログラム」を授業中に確実に動作確認をして理解し，演習問題を解く．②授業外の課題で「重要な例題プログラム」を3回程度手書きで書く（できる限り解答を見ない）．③授業開始2分で小テストの範囲を確認し，内容を思い出す．④小テストの内容は前回の授業の「重要な例題プログラム」である．

授業，課題，小テストを通じて一貫して同一のプログラムを理解し，確実に身に着けることを各回の授業で積み重ねる．勉強面で暗記は基本的に苦痛を伴うことが多いが，授業でシステム化することによって，受講生全員がある程度効率よく，プログラムを記憶し理解するようになる．

表１　プログラミングの授業内容

授業回	テーマ	キーワード
第1回	ガイダンスと標準出力	標準出力
第2回	変数	定数と変数
第3回	演算子	演算子
第4回	標準入力	標準入力
第5回	判断文（１）	if文
第6回	繰り返し文（１）	for文
第7回	確認演習	確認演習
第8回	判断文（２）	switch文
第9回	繰り返し文（２）	while文, do-while文
第10回	メソッド（１）	引数と戻り値
第11回	メソッド（２）	メソッド演習
第12回	配列（１）	1次元配列
第13回	配列（２）	2次元配列と文字列
第14回	総合演習（１）	総合演習
第15回	総合演習（２）	総合演習

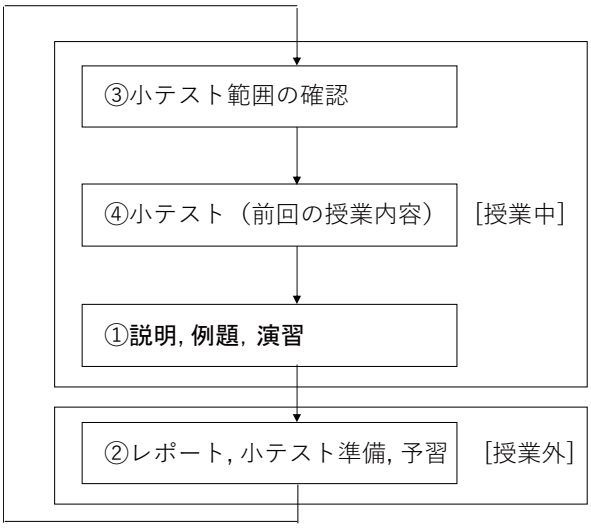

図１　プログラミングの授業の流れ

例えば，最初の授業では図２のプログラム例１のプログラムを「重要な例題プログラム」として，図３のようにプログラムの形式のイメージを確実に理解してもらう．ここでは「{}で囲まれた部分がブロックであること」，「ブロックの中の部分はプログラムが見やすくするためのインデント（字下げ）が必要であること」，「標準出力の処理はConsole.WriteLine();であり，” ”で囲まれたHello,Worldが画面に出力されること」を理解し，プログラムを確実に覚えてもらう．図１の授業の流れの通りに実施し，授業外課題と小テストを行うことで2回目の授業までにしっかり記憶に定着させる．

```
sing  System;
class  Ex01_1
{
   static  void  Main( )
   {
      Console.Write("Hello,World!");
   }
}
```

図２　プログラム例１

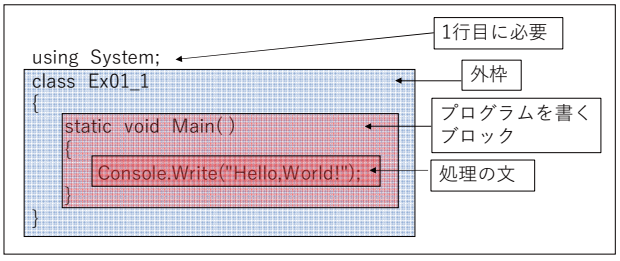

図３　プログラムの形式のイメージ

繰り返し文の重要プログラム例を図４に示す．プログラム例２は，１行ずつ，１つの処理ごとにどのような結果になるかを説明し，動かして確認してもらう．

```
using  System;
class  Ex06_2_1
{
    static  void  Main( )
    {
        int  sum, i;

        sum = 0;

        for( i=1; i<=10; i++ )
        {
            sum = sum + i;
        }

        Console.WriteLine( "合計は{0}", sum );
    }
}
```

図４　プログラム例２

図５のような簡単なプログラムの出力結果を予測する演習を行い，理解の確認とプログラムを読む訓練を行う．プログラムを読んでトレースできるようになることは，プログラミング技術において重要な要素である．図５のプログラムが分からない場合は，図４のプログラムなどに立ち返り丁寧に説明した後，逆に口頭でプログラムを実際に説明してもらう．

```
using  System;
class  Ren06_1
{
    static  void  Main( )
    {
        int i;
        for( i=0; i<=3; i++ )
        {
            Console.Write("xyz");
        }
    }
}
```

図５　出力結果を予測する演習のプログラム例

繰り返し文と，配列を理解できると図６のようなプログラム例も容易に理解できるようになる．これは画像の左右反転処理を行う画像処理の考え方に繋がるものである．

```
using  System;
class  Ren12_3
{
        int i, j;
        int[,]   ar  =  {
            { 30,  45,  60,  90, 180 },
            { 60,  90, 120, 180, 360 },
            { 90, 135, 180, 270, 540 }
        };
        for(j=0; j<=2; j++)
        {
            for(i=0; i<=4; i++)
            {
                Console.Write("{0}", ar[j,4-i]);
            }
            Console.WriteLine();
        }
    }
}
```

図６　出力結果を予測する演習プログラム

3. 学生による授業評価

ここでは，学生による授業評価の結果を報告する．図６に授業評価の質問項目と評価方法を示す．

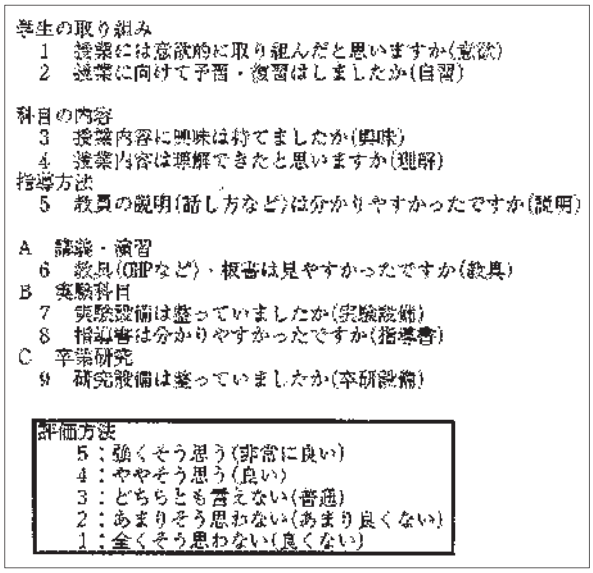

図６　学生による授業評価のアンケートの質問項目と評価方法

次に，2013年から2017年までの評価結果を表2に示す．2013年は，小テスト前後の予習・復習を徹底しては行っていなかったが，2014年以降は強化していった．その結果，成績・評価ともに良好になったことが分った．

表２　学生による授業評価のアンケート結果（文献2-6）

年度・学期(曜日・時限)	意欲	自習	興味	理解	説明	教具	平均	受講者数	回答者数
2013秋(水78)	3.85	3.65	3.85	3.75	3.95	3.90	3.83	21	20
2014秋(水78)	4.14	3.95	4.52	3.95	4.48	4.43	4.25	22	21
2015秋(火34)	4.39	3.93	4.14	3.89	4.50	4.46	4.22	31	28
2016秋(火12)	4.35	3.91	4.22	4.13	4.74	4.57	4.32	26	23
2016秋(水56)	4.30	3.95	4.40	3.90	4.70	4.65	4.32	21	20
2017秋(木56)	4.42	3.92	4.17	3.75	4.25	4.42	4.15	12	12

4. まとめ

プログラミングの学習では，「理解の伴わない「暗記」のみで適用すると，プログラミング技術や論理的思考の訓練に悪影響を及ぼすのではないか」という不安も考えられるが，初歩の段階では，「確実に理解する」必要がある文法については，「何度もプログラム書く」という筋肉的運動を伴う「基本パターンの暗記」（体で覚える）は非常に有効であり，重要であることが分かった．

参考文献

1）森文彦：プログラミング入門C# 2017年度版，玉川大学（2017）．

2）玉川大学工学部：学生による授業評価報告書27，玉川大学工学部教務担当会，p. 72（2014）．

3）玉川大学工学部：学生による授業評価報告書29，玉川大学工学部教務担当会，p. 73（2015）．

4）玉川大学工学部：学生による授業評価報告書31，玉川大学工学部教務担当会，p. 33（2016）．

5）玉川大学工学部：学生による授業評価報告書33，玉川大学工学部教務担当会，pp. 21-23（2017）．

6）玉川大学工学部：学生による授業評価報告書35，玉川大学工学部教務担当会（2018）．

2018年2月27日原稿受付，　2018年3月13日採録決定

Received, February 27, 2018; accepted, March 13, 2018