修士論文

# 学生のプログラミングの素養を調査する手法に関する研究

平成 26 年度修了

三重大学大学院工学研究科

博士前期課程　電気電子工学専攻

小林　史生

# 目次

## 1章　はじめに

コンピュータプログラムは現代社会において多くの様々なシステムを動かすために使われており，その仕組みを知ることは技術者だけでなくそれ以外の多くの人にとっても大切である．そのためプログラミングは理系大学ではもちろんのこと，中央大学の文学部や日本女子大学の文学部・家政学部などの文系学科でも必修科目として教えられている[1-2]．しかし，プログラミングは学生の生まれつきの素養によってよくできるグループとなかなか理解することができないグループに分かれることが知られている[3]．そのような状況の中，プログラミングを教える講師はプログラミングが苦手な学生を指導することに時間を取られてしまい，その間得意な学生は授業時間を持て余してしまうということが起こっている．その上，プログラミングが苦手な学生は講師がどれだけ時間をかけて指導しても最後までよく理解できないというケースも多い[3]．このように，現状のプログラミングの授業では，学生のプログラム作成にかかる時間の差に十分に対応しきれていない．

この問題に対して，授業を実施するにあたり事前に学生のプログラミングの素養を適切に評価することができれば，その結果によって学生をグループ分けして，それぞれに合った教育を行うことができるようになると考えられる．

学生の素養を測る手法に関する先行研究として，Saeed Dehnadi[3-4]らの研究がある．彼らはプログラミング学習前の学生にプログラムの代入の表現に関する問題を推測で解かせ，その際に学生が用いた代入記号の意味に対する解釈の一貫性と，その後のプログラミングの授業の成績との間に一貫性があることを示した．しかし，この手法ではプログラミングそのものを題材として扱っており，事前にプログラミングの知識を得たことのある学生に関しては適切な結果が得られない等の問題点が考えられる[5-6]．

そこで，本研究ではプログラミングで用いる概念を抽象化した問題によって学生のプログラミングの素養を評価することでプログラミングの素養を測る手法を考案した．この手法を用いることで，過去のプログラミング経験の有無に関わらず，学生たちのプログラミングの素養を評価してグル

ープ分けしてそれぞれのレベルに合った指導を行うことで，教育の質を改善することができるようになると期待できる．

本論文の構成は以下のとおりである．2 章で本研究の目的について，3 章では関連する手法について述べる．4 章でプログラミングの素養テストを提案し，5 章でその検証実験と結果について述べる．6 章では本手法の利用案について述べ，7 章で今後の課題について述べる．8 章で本論文のまとめを述べる．

## 2 章　研究の目的

### 2.1　背景

現在のプログラミング教育では，素養のある学生とそうでない学生が同じ授業を受けているが，両者のプログラム作成にかかる時間には大きな差が生まれる[7]．そんな中，講師はプログラミングが苦手でプログラム作成に時間のかかっている学生を指導するのに時間を取られてしまう．このためプログラミングが得意で早く演習を終わらせてしまった学生は講師からのスキルアップのための指導を受けることができず，時間を持て余してしまうという問題がある．実際に 2014 年 6 月に三重大学で「プログラミング演習Ⅱ」の授業での演習の時間に学生の様子を観察した結果を図 1 に示す．この日の演習問題は教科書の例題を参考にすれば容易な問題であったが，それでも学生たちの間で演習にかかる時間に差が出ていることが分かる．

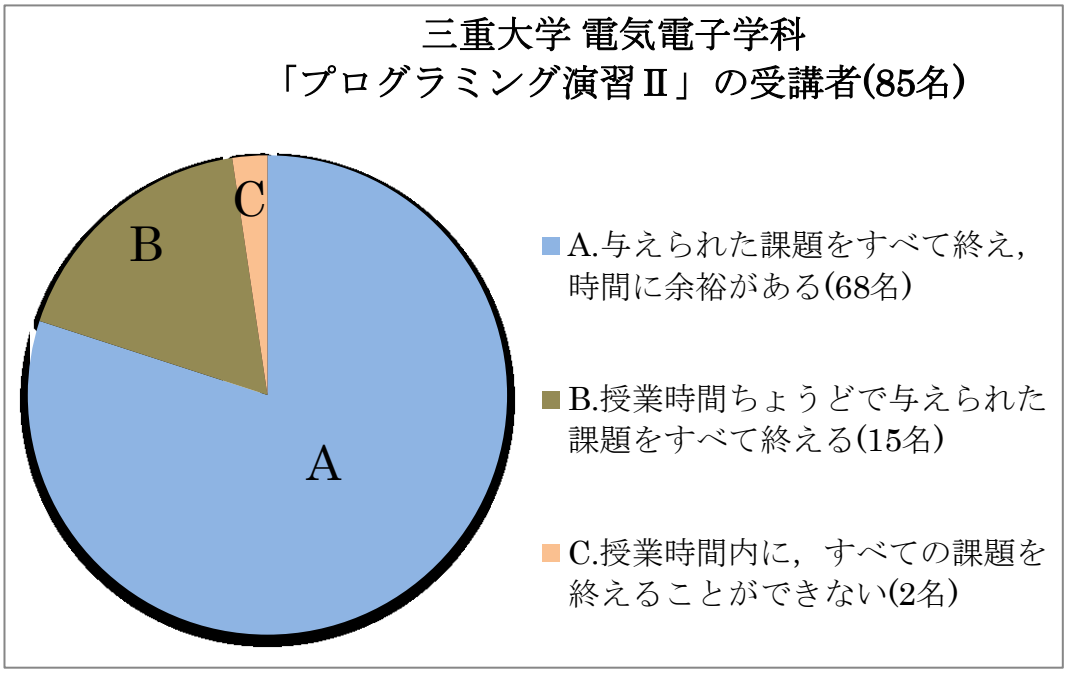



図 1　「プログラミング演習Ⅱ」の授業の観察結果

## 2.2　問題の解決策

この問題を解決する策としては，学生が教師の指導なしでもスキルアップのための自己学習を行うことができるような手法を確立することや，学生達の演習にかかる時間の差を縮小することが考えられる．なお，本研究室ではスキルアップのための自己学習システムに関する研究として，ソフトウェアテストの学習システムに関する研究（高桑，2014）[7]や，プログラミングスタイルの自己学習環境に関する研究（伊藤，2013）[8]などが行われている．

本研究では，クラス内で生じる学生のプログラム作成時間の差を縮小するという解決策に焦点を当てる．その具体的な手段として，学生の素養によってクラスを分けて，各クラスのレベルに合った指導を行うことが挙げられる．学生たちのプログラム作成時間に大きな差が生まれる原因は素養のある学生とそうでない学生が同じクラスで授業を受けている点にあるためである．

## 2.3　研究の目的

本研究では，前節で述べた解決策のうち学生のプログラミング作成時間の差を緩和するという解決策に注目し，授業前に学生を素養によってグループ分けすることを可能にするために学生のプログラミングの素養を事前に調査することができるような手法を確立することを目的とする．

# 3 章　関連手法

この章では，プログラミングの素養を測ることに関連するいくつかの手法を紹介する．

## 3.1　先行研究　Saeed Dehnadi らの手法

### 3.1.1　Dehnadi らの手法の概要

学生のプログラミングの素養を評価する手法に関する先行研究として，Saeed Dehnadi[3-4]らによる手法が存在する．この手法では，プログラミング受講前の学生に対して図 2 に示すようなプログラミングの代入文に関する問題を出題する．

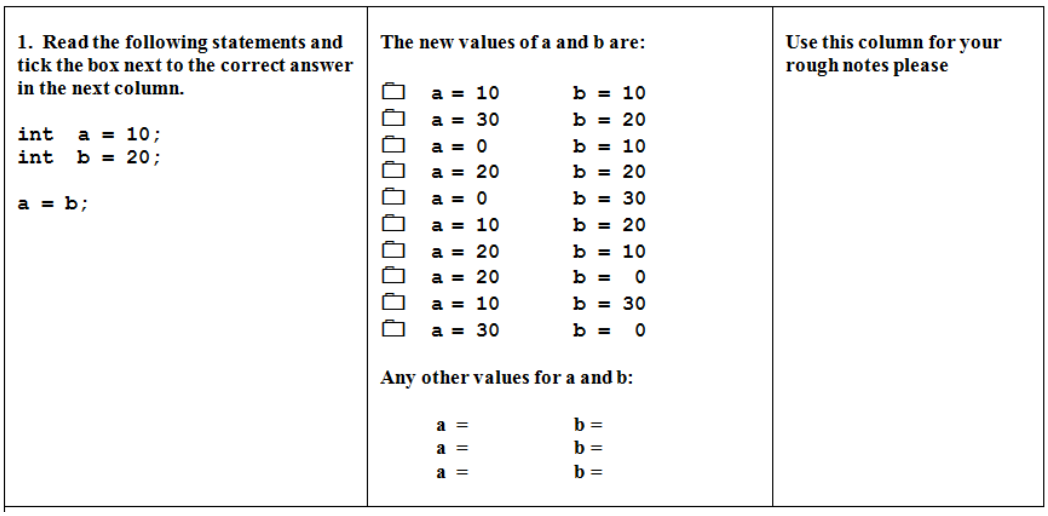



図 2　Dehnadi の手法で用いられた問題例

この問題の受験者はプログラミングの授業を受ける前の学生であるため，当然この中に出てくる代入記号 ”=” が何を意味しているか教わったことがない者が多く，この意味を推測して選択肢の中から正しいと思う解答を選ぶことになる．この時，学生が代入記号に対して用いるメンタルモデル，つまり学生が代入記号に対してどのような意味があると推測するかは，いくつかの種類に分かれる．このメンタルモデルの種類として Dehnadi らが挙げたものを表 1 に示す．右の値を左へ移す，右の値を左へコピーする，左右の値を入れかえる等，11 種類のメンタルモデルが示されている．

表 1　Dehnadi らの挙げた代入記号に対するメンタルモデルの種類

| |
|---|
| 1. Value moves from right to left. |
| 2. Value copied from right to left. |
| 3. Value moves from left to right. |
| 4. Value copied from left to right. |
| 5. Right-hand value added to left. |
| 6. Right-hand value extracted and added to left. |
| 7. Left-hand value added to right. |
| 8. Left-hand value extracted and added to right. |
| 9. Nothing happens. |
| 10. A test of equality: nothing happens. |
| 11. Variables swap values. |

Dehnadi らは，図 2 の問題と似たような代入文に関する問題を複数出題した場合に学生がそれぞれの問題において用いるメンタルモデルの一貫性に着目して，学生を多くの問題で一貫したメンタルモデルを適用して解答した学生とそうでない学生にわけ，一貫性のあるグループは一貫性のないグループに比べてプログラミングの授業でよい成績を修める傾向が強いことを示した．図 3 に彼らの手法により一貫性のあるグループとされた学生（黒），および一貫性のないグループとされた学生（白）それぞれの，授業での期末テストの成績の分布を示す．この図より，一貫性のあるグループの学生の方が一貫性のないグループの学生よりもテストの得点が高い傾向にあることが見て取れる．

### 3.1.2　Dehnadi らの手法の問題点

Dehnadi らはプログラミングを学んだことがない学生が代入記号に対して用いるメンタルモデルの一貫性がプログラミングの成績に関係があることを示したが，本研究で提案しているような授業前の学生のグループ分けの手段として用いるには，彼らの手法には二つの問題がある．

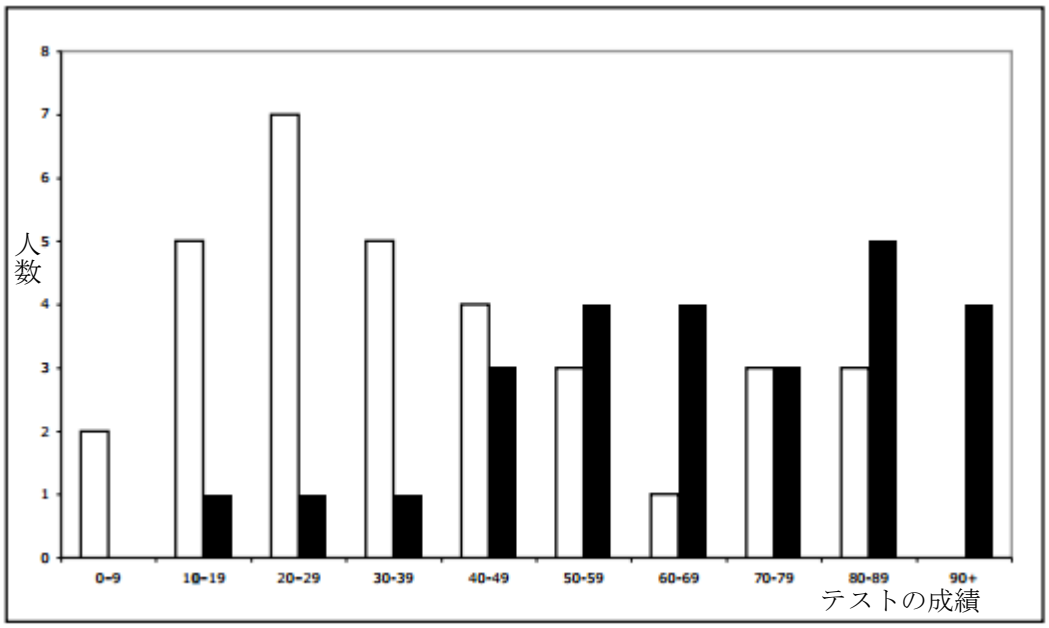



図 3　dehnadi らの手法と学生の授業でのテストの成績の関係
（黒：一貫性のあるグループ，白：一貫性のないグループ）

まず一つは，この手法はプログラミングそのものを題材としており，過去にプログラミングについての知識を学んだことがある場合は C 言語や Java 等のプログラミング言語における代入記号の正しい意味を知っている可能性が高いため，その場合は当然すべての問題で一貫したメンタルモデル，すなわちプログラミング言語で実際に使われているのと同じメンタルモデルを用いて解答できる可能性があるということである．

二つ目の問題点は，学生が今までに教わったことがないものに関する問題を推測だけで解くことを要求されるという点にある．ここで見ているのは学生が正しい答えを選んだかではなく一貫したメンタルモデルを用いているかであるが，図 1 のような問題を与えられた学生は問題の答えとして正しいものを選ぼうとするであろう．しかし，事前に学んだことがなければ，プログラミング言語における代入文の正しい意味はどれだけ考えても根拠をもって答えられるものではない．そのため，結局は推測で解答するしかなく，考えることに意味を見出せずに真剣に取り組むことができない学生が出てくる可能性がある．

### 3.1.3　三重大学での追試実験

前節で指摘した問題点の検証のため，三重大学の学生を対象に Dehnadi らの手法の追試実験を行った[5-6]．実験対象者は，「2013 年度電気電子設計（ソフトウェア設計)」の受講者のうち希望者 14 名と，「2014 年度計算機基礎」の受講者のうち希望者 32 名である．「電気電子設計」は大学 3 年次に開講され，この授業の受講者はすべてそれ以前にプログラミングに関する授業を履修している．一方，「計算機基礎」は大学 1 年次に行われ，この授業の受講者は大学でのプログラミングの授業はまだ受けていない状態である．

まず，「2013 年度電気電子設計」の受講者 14 名に対して Dehnadi らの手法を実施したところ，14 名全員が，全 12 題ある問題を全て C 言語で用いられる代入記号と同じメンタルモデルを用いて解答した（C 言語のプログラムとして正しい解答をした)．このことから，やはりプログラミング経験者に対しては適切な結果が得られないと言える．

次に，「2014 年度計算機基礎」の受講者 32 名に対して実施したところ，32 名のうち 5 名が，全 12 題を 1 分～2 分程度という極端に短い時間で解き終えるという結果が得られた．この時間は，12 問全ての問題文と選択肢に目を通して解答を選択するにはあまりに短すぎる時間であり，これらの学生は問題を読まずにいい加減に解答していた可能性が高い．これにより，やはりこの手法では真剣に取り組めない学生が存在することが確認された．

## 3.2　Cab（日本エス・エイチ・エル株式会社）

プログラミングの素養を含むシステムエンジニア職としての素質を測る適性検査として，日本エス・エイチ・エル株式会社の製品である Cab というテスト[9]や，IBM 社[10]による適性検査などがある．いずれも特徴が似通っているため，ここでは Cab について紹介する．

### 3.2.1　Cab の概要

Cab は, 主に IT 企業による採用活動において受験者のコンピュータ職としての適性を測るのに用いられ，以下の 5 つの測定科目から構成される．

1. 暗算
2. 法則性
3. 命令表
4. 暗号
5. パーソナリティ

### 3.2.2　Cab の問題点

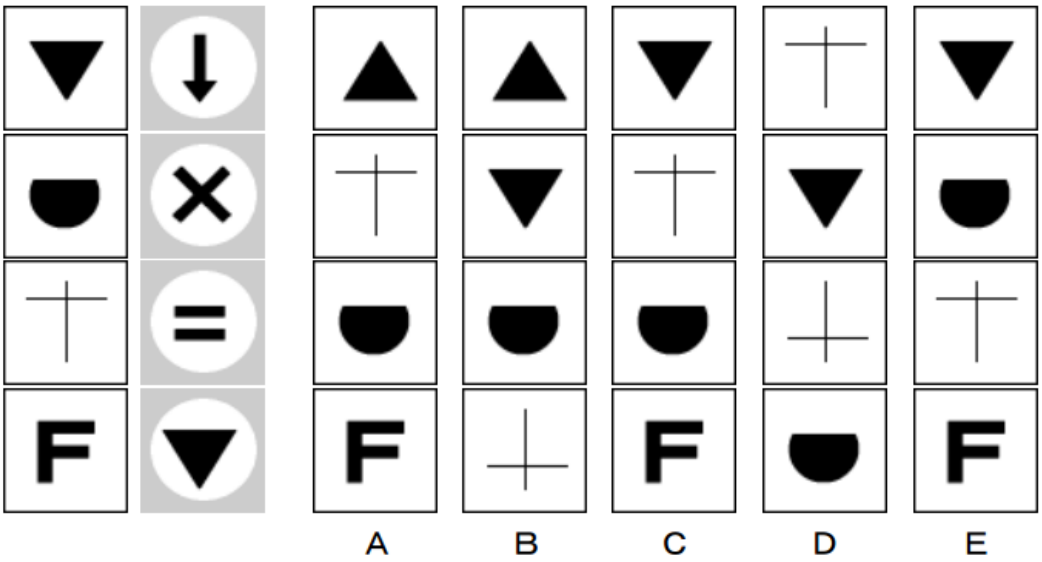
このテストはプログラマーの素養だけでなく，システムエンジニアとして働く際に重要となる判断力やヴァイタリティ，チームワーク等，様々な能力を測ることができるとされている．しかし，本研究の目的はプログラミングの教育の質の改善であり，プログラミングの素養を測ることができればよく，これらの全ての能力を測定する必要はない．また，このテストを行うには費用だけでなく，72～95分の時間がかかり学生への負担も少なくないため，今回の目的のために用いるには適切ではない．

### 3.2.3　プログラミングに関係の深い測定科目

このテストの測定科目の中で，プログラミングの素養にもっとも関係が深いと考えられるのは命令表のテストである. このテストの例題を図4に示す．はじめにいくつかの命令記号が示され，与えられた図形に指示通りの命令を上から順に実行するとどうなるかを選択肢から選ぶという問題である．このテストは，命令記号の定義を見て，それを正しく理解して適用する能力を測ることができるが，プログラミングの学習にも，初めて見る構文などの定義を理解し，それに従って記述する能力が必要であるため，このテストはプログラミングの学習段階に必要な能力に関係の深いものである．ただし，ここで扱っているのは図形の変化であり，図形を素早く正確に把握する能力も必要で，このテストでもプログラミングとはまた別の能力も同時に測っている．

| | | | |
|---|---|---|---|
| ↓ | 上下をさかさまにする | → | 左右をさかさまにする |
| ☆ | 次の命令を取り消す | × | 前の命令を取り消す |
| ▼ | 何も変えない | = | 前の図形と入れ替える |

(a) 命令表の例




(b) 問題例

図 4　命令表の問題例

# 4章　プログラミングの素養を測る手法の提案

## 4.1　評価すべき素養の検討

プログラミングの素養を評価する手法を考案するために，まず，プログラミングを習得する上で学習者に必要とされる素養について検討した．今回は，プログラミングの習得には以下に示すような素養が必要であると考えた．

1. 新しく学んだ構文や意味を理解してそれを正しく使用する能力

   プログラミング言語は，自然言語とは異なるが言語の一種であるため，定められた文法が存在する．プログラミングを学習する上では，例えば C 言語での if 文や for 文のような厳密に決められた構文の形やそれが表す意味を理解し，正しく使用できるようにならなければならない．

   例：C 言語の for 文

   ```
   for( 前処理; 制御式; 後処理 ) {
       繰り返す処理;
   }
   ```

2. 目的の動作を実現するために表現を組み合わせる論理的思考能力

   if 文や for 文のような構文単体を理解しても，それだけでプログラムのあらゆる動作を実現することはできない．ほとんどの場合，プログラムの動作は if 文や for 文，配列，ポインタといった様々な表現を多様に組み合わせて実現されている．このため，実用的なプログラムを作成することができるようになるためには多くの表現を自在に組み合わせて意図した動作を実現させるための論理的な思考能力が必要とされる．

例：学生全員分のスコアの合格，不合格を判定するプログラム

```
for( 学生全員分繰り返す記述 ) {
    if( 合否の判定式 ) {
        合格の場合の処理;
    } else {
        不合格の場合の処理:
    }
}
```

この他にもプログラミングの習得のために必要な素養がある可能性はあるが，今回は特に上記の 2 点に着目して，これらの素養を測ることができるような手法を考案する．これらの素養を測るために，何らかの処理を表す表現を定義して対象となる学生に示し，その表現を利用した問題を出題するという形式の素養テストを作成する．具体的にどのような処理を定義するかについては，次節で検討する．

### 4.2　出題する問題の検討

素養テストの作成にあたり，どのような処理を定義すればよりプログラミングの素養を測ることができるか検討する．プログラミングの素養テストであるので，実際のプログラミングで用いる概念に近い処理を表現するのが望ましく，特に，プログラミングの素養がある者とない者で差がつきやすいようなものがよいと考えた．

プログラミング学習者がつまずきやすいポイントとして，以下に示すような点が挙げられることが J H Benedict と B Du Boulay の研究[11]によってわかっている．

1. 代入とシーケンス実行
2. 繰り返し実行
3. 再起実行
4. 並列実行

今回の目的は，大学での授業でよくできるグループと苦手なグループにグループ分けできる手法を考案することであるため，特にプログラミングの基礎の授業で学ぶものについての素養を評価する必要がある．これらの中で，プログラミングの基礎の授業で扱うのは代入とシーケンス実行，繰り返し実行の 2 つである．今回はこの 2 つに加え，同じく基礎の授業で取り扱う主な項目の 1 つである分岐実行[12-13]を加えた下記 3 点に関する表現を定義することで，プログラミングの授業で扱うような概念を理解することができるかどうかを評価する素養テストを提案する．

- 代入とシーケンス実行
- 分岐実行
- 繰り返し実行

### 4.3　素養テストの提案

前節では，代入とシーケンス実行・分岐実行・繰り返し実行に関する問題を出題する素養テストを考案することを検討した．本節では実際にどのような表現の定義と問題を作成したかについて述べる．

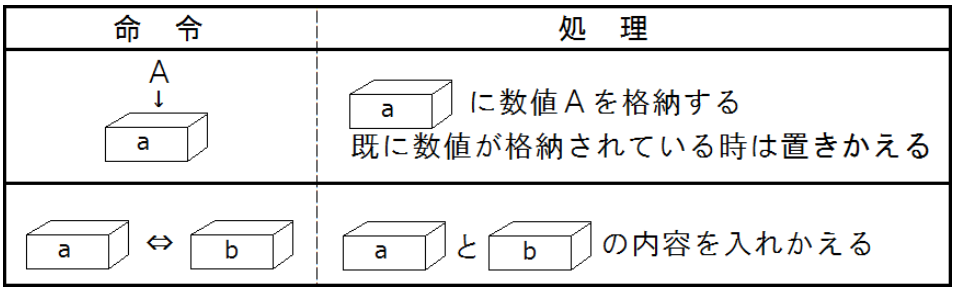
#### 4.3.1　代入とシーケンス実行

代入とシーケンス実行のテストでは，図 5 (a) に示すように，箱に数値を格納する命令と，2 つの箱に格納されている数値を入れ替える命令の表現を定義し，図 5 (b) に示すような問題を出題する．この問題は，箱に数値を格納する命令が代入を，上から順番に 1 つずつ実行する点がシーケンス実行を表現している．

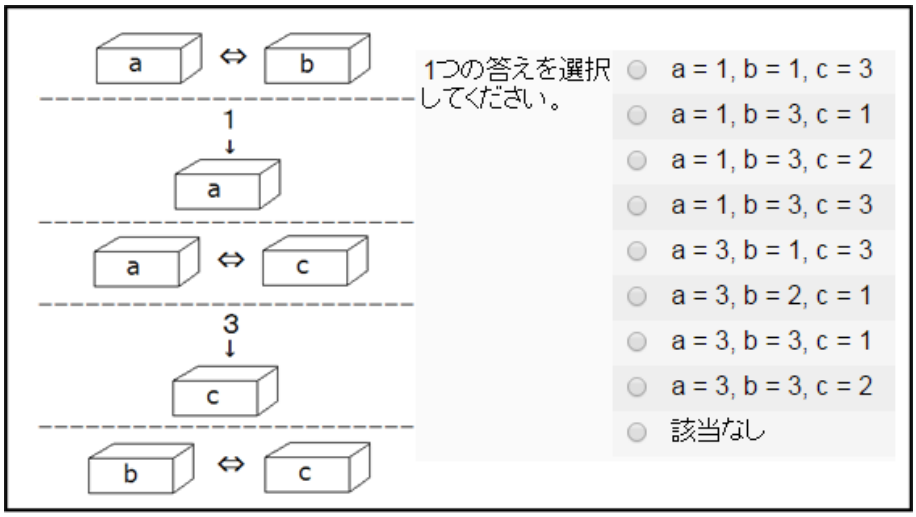
#### 4.3.2　分岐実行

分岐実行を理解する能力があるかのテストとして，図 6 (a) に示すような命令を図 6 (b) の中で分岐実行として示される順序に従って処理する問題を出題する．分岐実行の問題例を図 7 に示す．このテストは，分岐実行の命令の直前の演算結果が偶数か奇数かによって次の命令が分かれるという処理でプログラミングにおける if 文に近い処理を表現している．これを

正しく理解できればプログラミングの分岐処理も理解することができると考えられる．




(a)　命令の定義




(b)　問題例

図５　代入とシーケンス実行のテスト

| |
|---|
| **A ◎ B : A, B の和を表す**<br>**例 : 3 ◎ 5 = 8, 7 ◎ 5 ◎ 10 = 22** |
| **A ◉ B : A, B の差の絶対値を表す**<br>**例 : 3 ◉ 5 = 2, 7 ◉ 5 ◉ 10 = 8** |
| **A ○ B : A, B の差のうちの最大の値を表す**<br>**例 : 3 ○ 5 = 5, 7 ○ 5 ○ 10 = 10** |

(a) 命令の定義

| 処理順序：基本は左から順に計算 |
| --- |
| 分岐実行<br>［｜］：直前の結果が偶数のときは｜の左を，奇数のときは右を実行<br>例：2［◎ 2 ｜ ◉ 2］= 4，2 ○ 3［◎ 1 ｜ ◉ 1］= 2 |
| 繰り返し実行<br>【｜】:【】内の|の左の演算を|の右で指定された回数繰り返す<br>例：3【◎ 2 ｜ ③】= 3 ◎ 2 ◎ 2 ◎ 2 = 9 |

(b) 処理順序

図 6　分岐実行・繰り返し実行の命令と処理順序

**上記の命令を用いると以下の演算結果がどうなるか答えよ**

4 ◎ 1[○ 7 ｜ ◎ 3 ] = （ ？ ）

2 [◎ 2 ｜ ◉ 1] [○ 3 ｜ ◉ 2] = （ ？ ）

図 7　分岐実行の問題例

### 4.3.3　繰り返し実行

繰り返し実行のテストは分岐実行のテストは，図 6 (a) の命令を，図 6 (b) の中の繰り返し実行として示される順序に従って実行する問題を出題する．繰り返し実行の例題を図 8 に示す．

このテストは，【 】内の命令を指定された回数繰り返すという処理を正しく行えるかどうかを見るもので，プログラミングにおける繰り返し実行を表現している．実際のプログラミングの繰り返し表現は条件判定式によって繰り返しを制御するが，この問題では最も基本的な一定回数の繰り返し処理に近い処理となっている．

**上記の命令を用いると以下の演算結果がどうなるか答えよ**

2 ◎ 4 【◉ 10 | ②】= （ ？ ）

7【◉ 2 ○ 4 | ③】= （ ？ ）

図 8　繰り返し実行の問題例

また，4.1 節で述べた評価すべき素養のうち，複数の表現を組み合わせる時に必要な論理的思考能力を測る必要があるが，これについては，上記の分岐実行と繰り返し実行の命令を組み合わせた形の問題も出題する．

以上が今回考案した素養テストの概要である．関連手法と比較した際の利点としては，プログラミングの知識を得たことがある者に対しても効果が期待できる点と，プログラミングで使う概念に特化しており，Cab 等よりも受験者への負担が軽い点が挙げられる．

なお，図 6(a) の数値の演算命令の定義において，例として項が 3 つある演算を挙げているが，演算の定義の図だけでは処理順序の定義が曖昧であるためこのような例は適切ではなく，項が 3 つ以上の例を与えるのは処理順序の定義後でなければならない．後述の実験ではこの点に気付かずこの図を用いたが，今後この素養テストを行う場合は命令の定義の図では 2 項の例のみを与えるべきである．

# 5 章　実験

## 5.1　検証実験

今回提案した手法の効果を検証するため，実験を行った．本来は授業履修前に素養を評価することを目的としているためプログラミング未学習の学生を対象にして検証するのが望ましいが，プログラミングの成績が出るまでに時間がかかってしまうため，素養テストとプログラミングの素養との関係を検証するのが難しい．そこで，今回はすでにプログラミングの授業を受け，その成績が出ている学生を対象として検証実験を行った．

### 5.1.1　実験対象

三重大学 工学部 電気電子工学科の電気電子設計の授業のうち，2014 年度電気電子設計（ソフトウェア設計）の受講者を対象にして実験を行った．この授業は，「プログラミング演習Ⅰ・Ⅱ」，「プログラミング言語」という電気電子工学科のプログラミングに関するすべての授業の後に履修されるもので，今回の対象はすでに全員プログラミングに関する知識を教わっている．ただし，「プログラミング言語」に関しては選択科目であり，今回の対象のうち 2 名は履修していない．なお，実験時にこれらの学生に対して各授業の成績を研究に利用してもよいかアンケートを実施し，本論文では対象 41 名のうち成績の利用の許可が得られた 36 名の結果を報告する．

### 5.1.2　素養テストの実施

まず，対象の学生に対して，今回提案した素養テストを三重大学 Moodle の小テスト機能を用いて実施した．

実施内容は，代入とシーケンス実行の問題 4 題，分岐実行の問題 4 題，繰り返し実行の問題 4 題，分岐実行と繰り返し実行を組み合わせた問題 2 題の計 14 題を制限時間 15 分として実施した．各問題の配点に関しては，問題ごとに得点の重みを変えた方がよい可能性はあるが，この点については問題ごとの結果を見てから検討することとし，ここではすべて 1 問 1 点の 14 点満点として出題した．

### 5.1.3　素養テスト結果とプログラミングの成績の比較

素養テストを学生に対して実施した後，その結果とその学生のプログラミング関係の授業（プログラミング演習Ⅰ，プログラミング演習Ⅱ，プログラミング言語）の成績を比較し，相関関係を考察した．なお，これらの授業のうち，「プログラミング演習Ⅰ・Ⅱ」は必修でプログラミングの基礎を学ぶ科目であり，それぞれ 1 年次後期，2 年次前期に履修している．「プログラミング言語」は，ポインタなどもう少し発展的なプログラムを学ぶ選択科目であり 2 年次後期に開講されている．なお，不合格により同じ科目を再履修している学生に関しては，1 回目の履修時の成績を用いて比較した．

## 5.2　実験結果

素養テストを実施した学生の得点の分布を図 9 に示す．プログラミングの成績の分布は，よくできる集団とあまり得意でない集団の 2 つのグループに分かれることが知られている[1]．今回の素養テストの結果も，11 点と 12 点の間を境界に，2 つの集団に分かれるという結果が得られた．

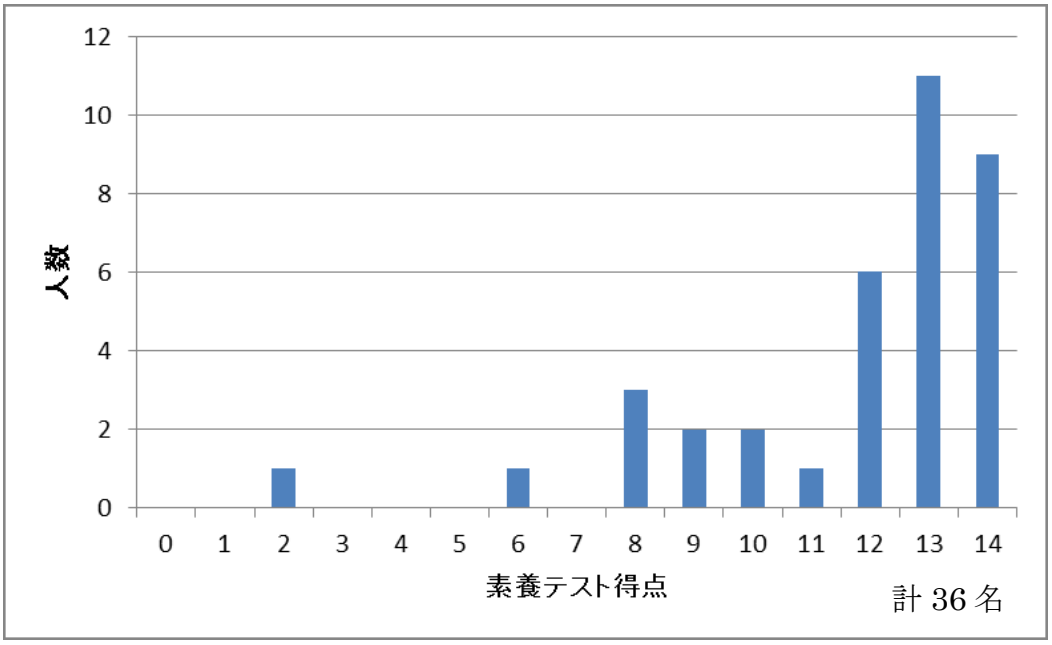



図 9　素養テストの得点の分布

また，学生が素養テストに解答するのにかかった所要時間のグラフを図 10 に示す．今回の素養テストは 15 分の制限時間を設けて実施したが，その時間内に問題を解ききれなかった学生はおらず，平均解答時間は約 9 分で，多くの学生が 7～10 分程度で解答を終えて提出した．もっとも所要時間の短い学生は 2 分 36 秒で解答を終えているが，この学生は得点も満点であり，早い学生であればこの程度の時間で解答できることが分かった．

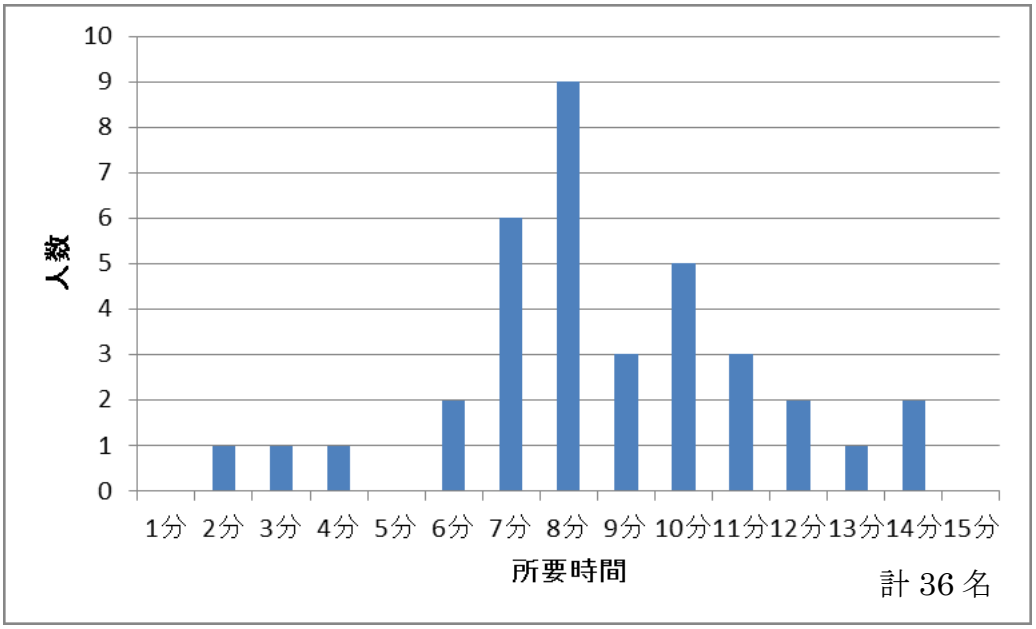


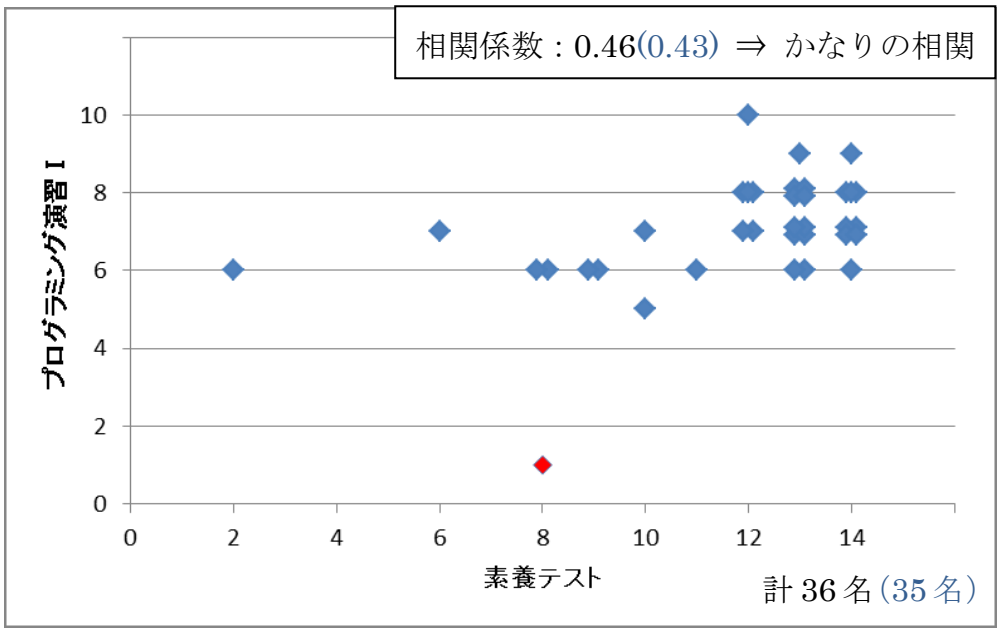

図 10　学生が素養テストを解くのにかかった時間

### 5.2.1　素養テストの結果とプログラミングの成績の比較

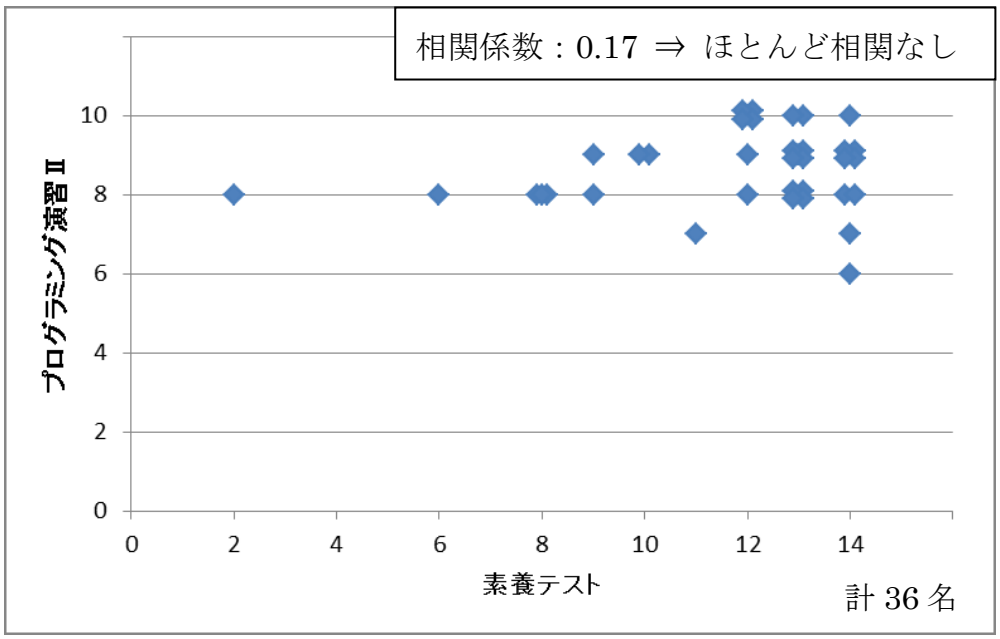
次に，素養テストの結果と対象の学生のプログラミング関係の授業の成績（プログラミング演習Ⅰ・Ⅱ，プログラミング言語）との相関関係を図 11 に示す．「プログラミング演習Ⅰ」と「プログラミング言語」の初回の授業では評定が 0 や 1 の学生が存在するが，これらの学生は出席日数不足等により不合格になっており試験によるプログラミング能力の評価がなされていない．そのため例外点と考え図中では赤いプロットで示している．

素養テストの結果と「プログラミング演習Ⅰ・Ⅱ」，および「プログラミング言語」の成績との相関係数はそれぞれ，0.43，0.17，0.39 となっており，「プログラミング演習Ⅰ」，「プログラミング言語」とは相関があり，「プ

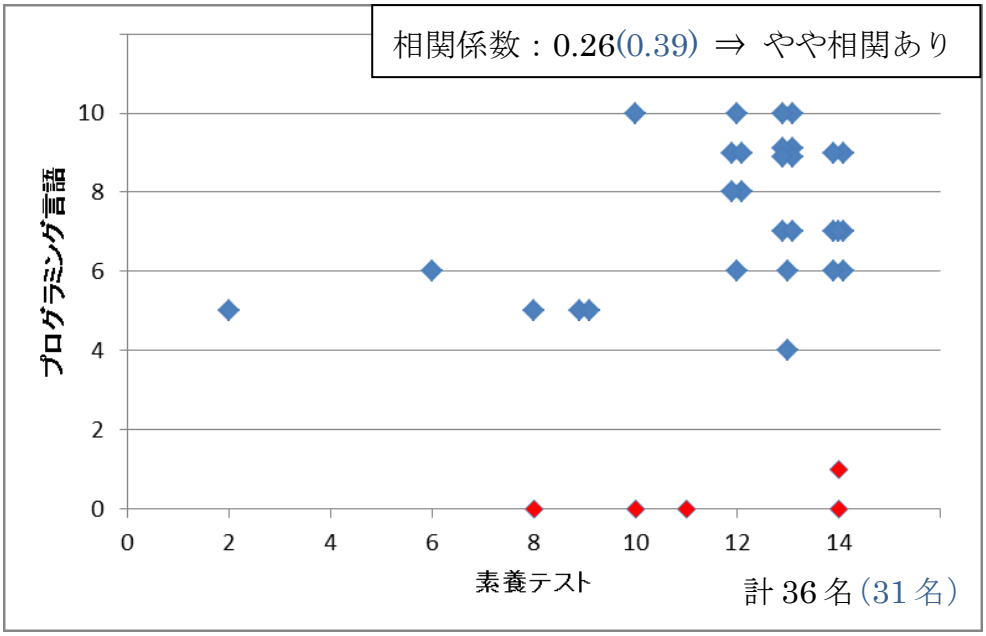
ログラミング演習Ⅱ」とはほとんど相関がないという結果が得られた．なお，ここでの相関係数に関する表現は「図解でわかる統計解析」（日本実業出版社）[14]に基づいている．




(a) 素養テストと「プログラミング演習Ⅰ」




(b) 素養テストと「プログラミング演習Ⅱ」




(c) 素養テストと「プログラミング言語」

図 11　素養テストと各授業の成績との相関関係

### 5.2.2　素養テストの各形式とプログラミングの成績の比較

素養テストの結果とプログラミングの成績の関係についてより詳しく考察するため，素養テストに含まれる問題形式ごとにプログラミングの成績との関係を検証した．各問題形式と各授業の成績との間の相関係数を表 2 に示す．この中で，相関係数が 0.4 以上あるものを，かなりの相関があるとして青色の背景で示し，0.3 以上 0.4 未満のものを，やや相関があるとして黄色の背景で，0.2 以上 0.3 未満のものを弱い相関があるとして[14]赤色の背景で示した．この表より，素養テストの各問題形式のうち，代入とシーケンス実行に関する問題については学生のプログラミングの成績との間に相関関係は見られなかった．分岐および繰り返し実行に関する問題は，分岐実行のみ，繰り返し実行のみの問題については学生の成績との相関関係が見られたが，分岐実行と繰り返し実行の両方を含む問題（分岐&繰り返し）に関しては相関関係が見られたものの，分岐・繰り返し単体の結果と比べると相関係数は低いという結果となった．この理由については次節で考察する．

これらの結果のうち，明らかに相関係数が低くプログラミングの成績との間に相関関係が全く見られなかったのは代入とシーケンス実行に関する問題である．ここで，素養テストのうち代入とシーケンス実行の問題を外して，分岐・繰り返し実行に関する問題の合計得点と各授業の成績との相関関係の分布図を図 12 に示す．

表 2　素養テストの各問題形式とプログラミングの成績との間の相関係数

| | 演習Ⅰ（1年次後期） | 演習Ⅱ（2年次前期） | 言語（2年次後期） |
|---|---|---|---|
| 全体 | 0.43 | 0.17 | 0.39 |
| *代入とシーケンス | 0.025 | −0.085 | 0.01 |
| *分岐・繰り返し合計 | 0.42 | 0.23 | 0.44 |
| ・分岐 | 0.42 | 0.22 | 0.47 |
| ・繰り返し | 0.47 | 0.23 | 0.44 |
| ・分岐＆繰り返し | 0.36 | 0.18 | 0.2 |

| かなりの相関 | やや相関 | 弱い相関 |
|---|---|---|

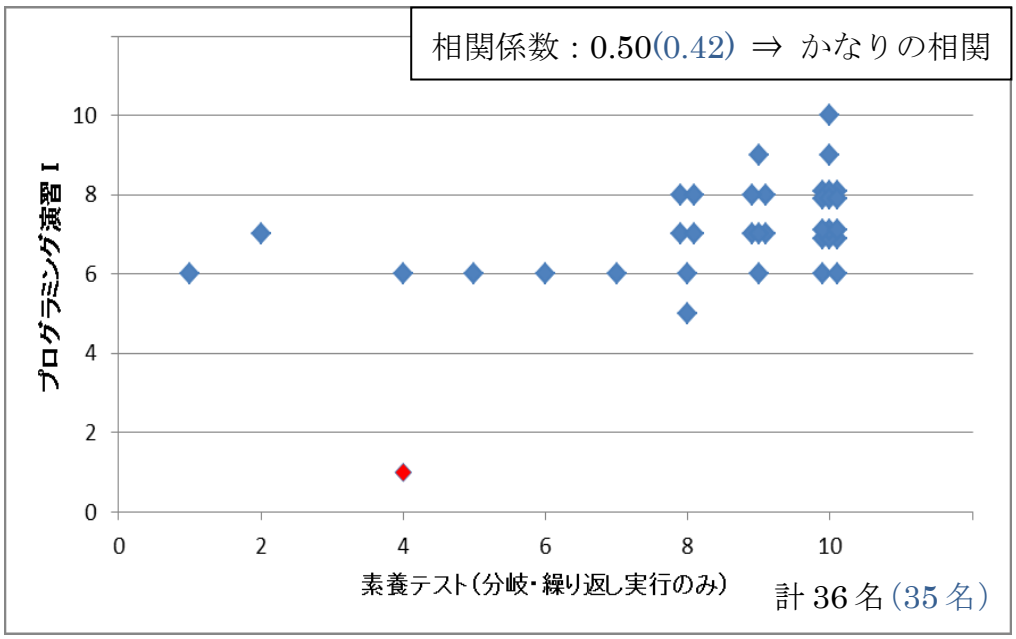



図 12　分岐・繰り返し実行のテストと「プログラミング演習Ⅰ」の関係

### 5.3　考察

今回提案した素養テストを実際に学生に対して使用した結果を前節で示したが，この節ではそれらの結果について考察する．

#### 5.3.1　素養テスト結果と各授業の成績との関係

前節の図 9 で示した受験者の得点の分布より，受験者の得点は 11 点と 12 点の間を境に二つの集団に分かれるという結果となった．また，対象のプログラミングの成績との比較を示す図 11 より，12 点以上を取った学生のグループは 11 点以下の学生のグループに比べて授業の評定も高い傾向にあることがわかる．特に，「プログラミング演習Ⅰ」と「プログラミング言語」の成績ではその傾向が強く，それらの授業で評定が 8 以上という高い成績を得ている学生はいずれも今回の素養テストで 12 点以上のグループに入っている．「プログラミング演習Ⅱ」に関しては，他の 2 つの授業と比べると低い相関係数になっている．この授業の成績評価ではレポートなど，プログラミング能力以外の部分も重視されており[15]，その分プログラミングが苦手な学生でも好成績を取りやすかったという理由が考えられる．それでも分岐・繰り返し実行との間の相関係数は 0.2 を超えており，他の 2 つの授業よりは弱いとはいえ相関関係は見られるという結果となった．

#### 5.3.2　素養テストの所要時間

対象の学生が素養テストを解答するのに要した時間はもっとも早く解答を終えた学生で 2 分 36 秒，7 分～11 分程度の学生が大半で，もっとも長い時間をかけた学生は 15 分の制限時間を一杯まで使った 14 分 56 秒であった．また，平均解答時間は約 9 分となっている．すべての問題を解答し切れないまま提出した学生はおらず，このことから，この素養テストはおおよそ 10 分程度，長くても 15 分あれば実施でき，学生に大きな負担をかけることなく行うことができることが確認できた．

### 5.3.3　代入とシーケンス実行

今回実施した素養テストの問題形式のうち，代入とシーケンス実行に関する問題の得点は，学生のプログラミングの成績との間に相関関係が見られなかった．代入とシーケンス実行はプログラミングにおいてもっとも基本的な概念であり，学習者がプログラミングを身に付ける際に最初に覚えるものであるが，これが理解できない場合プログラムを作成することが非常に困難になる．今回の代入とシーケンス実行に関する問題はこれを理解する能力を評価することを目的として作成したものであるが，今回の対象は電気電子設計の中でプログラミングを取り扱うソフトウェア設計を自ら選択した学生がほとんどで，ここでつまずく学生はいなかった可能性が高いと考えられる．よって，今回の代入とシーケンス実行に関する問題の得点は学生のケアレスミスによってしか差が付かず，学生のプログラミングの成績との間に相関関係が見られなかったと考えられる．この問題に関しては，今回のようにソフトウェアに関する科目を進んで履修している学生の集団だけでなく，このような問題が得意か不得意かという点で偏りのない集団に対して実験を行い，結果によっては素養テストからこの問題を外したり，得点の重みを調整したりする必要性も検討していく必要がある．

### 5.3.4　分岐・繰り返し実行

分岐・繰り返し実行の問題については，プログラミングの成績との間に 0.4 前後の高い相関係数が得られた．この中で分岐実行と繰り返し実行を組み合わせた問題については，分岐実行および繰り返し実行単体で出題した問題よりも相関係数が低くなっている．この理由として，素養テストの問題数が少なかったことが考えられる．分岐実行と繰り返し実行単体の問題はそれぞれ 4 題ずつ出題したのに対し，分岐実行と繰り返し実行を組み合わせた問題は 2 題しか出題しなかった．このため，学生の得点に差が付きにくく，相関関係が表れにくくなってしまった可能性がある．この問題も問題数を 4 題にすることで，相関係数を向上させることができると考えられる．これにより今回平均 9 分程だった所要時間が増えてしまうが，制限時間として設定した 15 分で解き切れないことは少ないと思われる．

### 5.3.5　プログラミング経験の素養テストの得点への影響

今回の検証実験の対象としたのは三重大学工学部 3 年次の学生であり，対象全員が素養テストの実施より先に大学でのプログラミングに関する授業を履修してきている．今回の素養テストは，プログラミングにおける代入とシーケンス実行，分岐実行および繰り返し実行の概念を表現した形の問題となっているが，対象がそれまでにすでにこれらの概念について学習していた場合，その知識が今回の問題を解く際の助けになり，素養テストの結果に影響を与えてしまうという可能性が否定できない．本手法の本来の目的はプログラミングの授業の受講前の学生に対して実施してプログラミングの素養を評価することであり，この目的のために使用できることを確認するためにはやはりプログラミング未経験の学生を対象にして検証実験を行う必要がある．

# 6 章　本手法の利用案

本手法には課題はまだ残っているが，今回の検証実験の対象に関してはプログラミングの成績との相関関係を確認することができており，プログラミングの素養を評価する手法としての効果を期待できる．そこで本章では，この手法をプログラミング教育の質を改善するためにどのように利用できるかについて，ひとつの具体案を示す．

## 6.1　学生のグループ分け

まず，プログラミングの授業を受ける前の学生に対してこの素養テストを実施し，その結果によって学生たちを 2 つのグループに分類する．図 13 に示した今回の実験結果では，授業で好成績を修める学生は素養テストで 12 点以上獲得している傾向が強いため，ここでは分類する基準は「12 点以上獲得しているかそうでないか」とする．

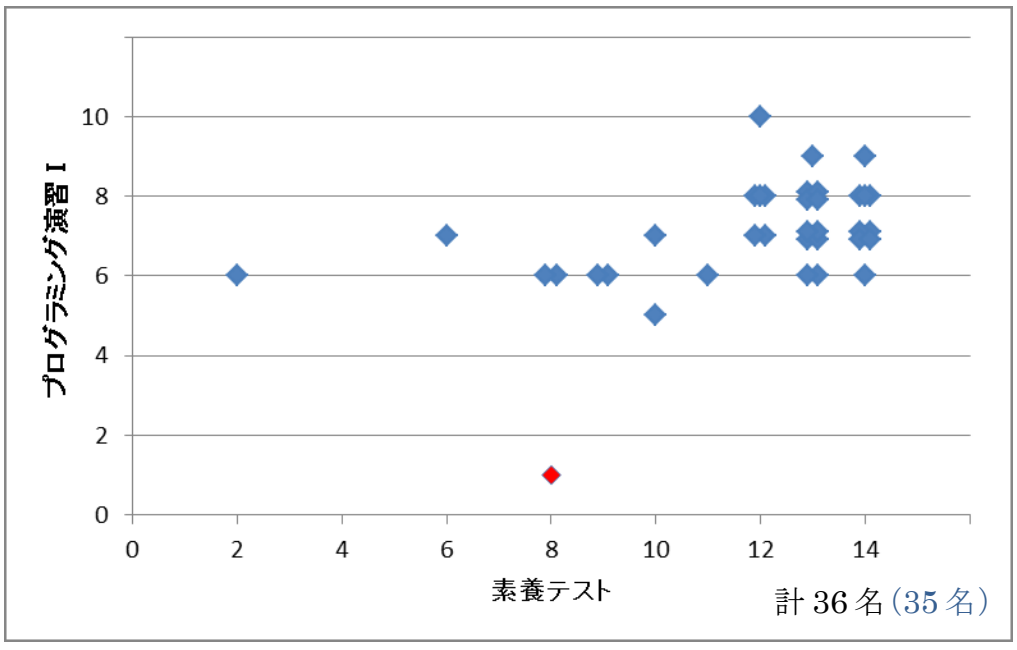



図 13　素養テストと「プログラミング演習 I 」の相関関係

## 6.2　各グループへの適した授業の実施

前節で述べたように学生をグループ分けした後，グループごとに異なるクラスで授業を行う．11 点以下のグループではこれまで通りプログラム作成に時間を使い，12 点以上のグループは全体的に早くプログラムが作れるためその分さらに応用的なプログラミングの演習や発展的な内容の学習に時間を割くことができる．このように，素養のある学生とそうでない学生が同じクラスで授業を受ける場合に比べて学生のプログラム作成にかかる時間の差が緩和され，時間を持て余す学生を減らすことで教育の質を改善することができる．

# 7章　今後の課題

本研究ではプログラミング学習済みの学生に対して素養テストを実施し，素養テストで出題した問題のうち分岐実行および繰り返し実行の問題の得点と学生が過去に受けたプログラミングの授業の成績との間に相関関係があることが確認できた．しかし，今回は学生の成績との間に相関関係が得られなかった代入とシーケンス実行に関する問題の必要性の検討と，本来使用を想定しているプログラミング学習前の学生への効果の確認が，本手法の今後の課題として残っている．このため，実際にプログラミング学習前の学生に対しても本手法を実施し，これらの課題について検証する必要がある．

# 8章　まとめ

プログラミングの授業において，プログラミングの素養がある学生とそうでない学生では，プログラム作成にかかる時間に大きな差ができる．教師はプログラミングの苦手な学生を中心的に指導するため，プログラムを早く作り終えた学生はスキルアップのための指導が受けられず，時間を持て余してしまうという現状がある．この問題は，授業を受ける前に学生のプログラミングの素養を評価することができれば，素養のある学生とそうでない学生にクラス分けをすることで改善することができる．そこで，本研究では，プログラミングの教育の質の改善に役立てることを目的として，学生のプログラミングの素養を調査するための素養テストを提案した．

この素養テストは，プログラミングを習得する際に必要と考えられる素養を考慮して考案しており，代入とシーケンス実行，分岐実行，繰り返し実行という問題形式で構成されている．

この素養テストの検証実験を三重大学の 3 年次の学生 36 名に対して実施し，彼らが過去に受けたプログラミング関係の授業の成績との相関関係を調べたところ，素養テストのうち分岐実行・繰り返し実行に関する問題の得点と「プログラミング演習Ⅰ」，および「プログラミング言語」の成績との間には 0.4 を超える高い相関係数が得られ，これらの間に相関があることが確認できた．また，「プログラミング演習Ⅱ」との相関に関しても，相関係数は少し低くなるが弱い相関がみられた．このことから，この素養テストはプログラミングの素養を評価する手法として効果があると期待できる．

ただし，今回実験を行った対象は本来の用途として想定している対象とは異なり，全員プログラミング経験者で，かつプログラミングに関する選択科目をすすんで履修している学生が多い．このため，今回は学生の成績との相関が得られなかった代入とシーケンス実行に関する問題の必要性や，過去にプログラミング授業を受けたことによって素養テストの得点に影響が出た可能性についてはさらに検証を行っていく必要がある．

## 参考文献


[1] 「中央大学文学部 カリキュラム」
<http://www.chuo-u.ac.jp/academics/faculties/letters/guide/curriculum/>
（2015/1/30 アクセス）

[2] 「日本女子大学 基礎科目詳細」
<www.jwu.ac.jp/unv/faculty_department/curriculum_gakubu/curriculum_mejiro/kiso/>（2015/1/30 アクセス）

[3] Saeed Dehnadi and Richard Bornat：The camel has two humps (2006)

[4] Saeed Dehnadi, Richard Bornat, Ray Adams：Meta-analysis of the effect of consistency on success in early learning of programming (2009)

[5] 小林史生, 北英彦:学生のプログラミングの素養を調査する手法, CIEC コンピュータ利用教育学会，PC カンファレンス 2014 (2014)

[6] Fumio Kobayashi and Hidehiko Kita：A Method to Ascertain Whether Students Have Natural Aptitude for Programming, The 4th International Symposium for Sustainability by Engineering at Mie University (2014)

[7] 高桑稔，北英彦：プログラミング能力向上を目的としたプログラムテストの学習システム，CIEC コンビュータ利用教育学会， PC カンファレンス 2014 (2014)

[8] 伊藤雅人，杉山宏太，北英彦：プログラムの書式チェックを行うコンピュータシステムの提案，CIEC コンビュータ利用教育学会， PC カンファレンス 2013 (2013)

[9] 「日本エス・エイチ・エル」<http://www.shl.co.jp/>（2014/6/2 アクセス）

[10] 「日本 IBM」<http://www.ibm.com/jp/ja/> (2015/2/13 アクセス)

[11] J H Benedict and B Du Boulay：Some difficulties of learning to program (1986)

[12] 柴田望洋：解きながら学ぶ C 言語 入門編, SB Creative（2004）

[13] 柴田望洋：解きながら学ぶ C 言語 実践編, SB Creative（2004）

[14] 前野昌弘，三國彰：図解でわかる 統計解析，日本実業出版社（2000）

[15] 「三重大学ウェブシラバス プログラミング演習Ⅱ」
<http://syllabus.mie-u.ac.jp/?action=display&id=472>（2014/2/2 アクセス）

## 実績一覧

[A] 小林史生, 北英彦:学生のプログラミングの素養を調査する手法, CIEC コンピュータ利用教育学会，PC カンファレンス 2014，2014/8/9

[B] Fumio Kobayashi and Hidehiko Kita : A Method to Ascertain Whether Students Have Natural Aptitude for Programming, The 4th International Symposium for Sustainability by Engineering at Mie University, 2014/9/29

[C] 小林史生，北英彦：学生のプログラミングの素養を調査する手法, CIEC 春季研究会 2015 研究報告，2015/3/28

## 謝辞

本論文は，著者が三重大学大学院工学研究科博士前期課程に在籍中に行った研究をまとめたものである．本研究を進めるにあたり，懇切丁寧な御指導と御督励を賜った三重大学大学院工学研究科の北英彦准教授に深く感謝いたします．

また，貴重な時間をさいて本論文を査読して頂いた，三重大学大学院工学研究科の森香津夫教授，地域イノベーション学研究科の鶴岡信治教授に深く感謝致します．最後に，日頃熱心に討論していただいた計算機工学研究室の皆様方にお礼申し上げます．