Vol.2013-CE-121 No.7
2013/10/12

プログラミング学習の理解度とソースコードタイピングに関する考察

中田 豊久[1,a)]

概要：キーボード操作を覚えるために，画面に表示される日本語を打ち込むというタイピング練習ソフトウェアというものがある．本研究では，表示される文字を日本語ではなくコンピュータプログラムにして，プログラミングの学習に役立てることを提案する．このようなプログラミング学習のためのタイピングをソースコードタイピングと本研究では呼ぶことにする．大学で初めてプログラミングを学ぶ学生に対し，週 1 回の授業の開始前にこのソースコードタイピングを課した．その結果，成績の向上は見られなかったが，タイピングの速度とプログラミングの理解度にある一定の相関があることが明らかとなった．

キーワード：プログラミング学習，タイピング

Source Code Typing for Learning Programming for Beginners

Nakada Toyohisa[1,a)]

Abstract: In order to learn programming for beginners I propose souce code typing that is a training to type correctly key by tracing displayed source code. I performed an experiment to evaluate the typing system in my programming lecture in my university. From the experiments it is founded that a speed of the typing is influenced to understanding of computer program in some condition. We concluded that there is an association between typing speed and understanding of computer program, but there is not a cause association of them.

Keywords: Learning Computer Programming, Keyboard Typing

1. はじめに

プログラミング教育には，条件式や繰り返しなどの構文を理解することや，それらをどのように組み合わせて問題を解くか，そしてプログラムとしてコンピュータの中で表現したい世界をどのようにモデル化するかなどの段階がある．本研究ではこれらの中で特に，初めてプログラムを学ぶ初心者を対象としたプログラミング教育について論じる．

Dehnadi ら [1] によると，どの学力レベルの学生であっても約 60%の人はプログラミングに向いていないと述べている．残り約 40%の人は，その 60%の人に比べて何が違うかというと，何か混沌とした状況から法則を導き出し，そしてその法則を新しいデータに対して正確に適用できる能力を持っている人たちだという．プログラミング技術の習得は，学力のレベル，文系／理系にかかわらず，誰にでもできることではないことを示唆している．

このような容易ではないプログラミング技術をできるだけ多くの人に享受してもらうために，様々な試行がなされている．まず大きな流れとしては，Problem-Based Learning (PBL)[2] という方法が注目されている．従来は，教師が条件式などを 1 つずつ説明し，学生は多くの時間をその説明を聞くことに費やす授業方法であった．一方，PBL ではまず問題を提示し，それを解くために必要なこと逆算して学んでいく．PBL は主に医学教育の分野で発達し，様々な教育現場に波及している．プログラミング学習においても

[1] 新潟国際情報大学
Niigata University of International and Information Studies
[a)] nakada@nuis.ac.jp

ゲームなどを利用し，何のプログラムを作成するのかをより明確として学生の興味を喚起してきた研究例がある [3].

また PBL の考え方をより細部までわたらせたような学習方法もある．プログラムには様々なサンプルコードがあるため，まずは内容を理解していなくても兎に角そのサンプルコードを打ち込み，実行することから始める学習方法がある．そのような学習方法を喜多ら [4] は，写経型学習と呼んでいる．また写経プログラミングと呼ぶこともある．

本研究では，数あるプログラミング教育の支援の中でも，この「まずは理解する前にプログラムを打ちこむ」というところに着目し，その学習効果を検証する．写経プログラミングと異なる点は，写経プログラミングはまずプログラムを入力して実行してから，プログラムの内部を例えば少し変更しながら理解するものである．一方，本研究での「まず理解する前にプログラムを打ちこむ」とは，頭でプログラムのロジックを学ぶだけでなく，とにかく何度も一定のプログラムを打ちこむことにより，指の動作だけではあるのだが「体感的」にプログラミングを会得するということである．この体感的に学習するとは，例えば漢字や英単語の綴りを覚えるためにノートに何度も書き取りをすることと似ている．そのために我々は，C 言語用のタイピングソフトを開発した．

本研究で知りたいことは，タイピングソフトがプログラミング学習に役立つのか，またそもそもタイピングの速さや正確性といった数値が，C 言語の理解をある程度数値化した授業の成績と相関があるのかである．つまり以下の疑問に対する回答について示すことを目的とする．

(1) C 言語タイピングを行うことによって C 言語授業のテストの成績はよくなるのか?
(2) C 言語タイピングと C 言語授業のテストの成績に何かしらの関係があるのか?

本論文は次のように構成されている．2 章では実験対象とするプログラミング授業の技術レベルについて示す．3 章で実験で用いた C 言語タイピングソフトについて説明する．4 章で分析するために使用したデータについて述べ，そして 5 章で実験の概要および結果を示す．6 章では関連研究について触れ，7 章でまとめる．

2. 対象とする初心者向けプログラミング授業の技術レベル

本研究における初心者向けプログラミング授業のレベルは，次のような課題に適合するプログラムを作成できるレベルの学生を育成することである．授業で取り扱うプログラミング言語は，C++である．

2 つの数字をキーボードから入力し，その合計と平均を出力しなさい．また，その入力された 2 つの数字の掛け算と同じ結果になる 1 桁の数字の掛け算のすべての組み合わせを出力しなさい．例えば，1，6 と入力された場合，合計 7，平均 3.5，1*6 と同じ結果になる掛け算は，2*3，3*2，6*1 と出力されれば良いです．

```
#include <stdio.h>
int main()
{
  printf("数字を 2 つ入力してください。\n");
  int value1;
  scanf("%d",&value1);
  int value2;
  scanf("%d",&value2);
  printf("\n");

  printf("合計 %d，平均 %lf\n\n",
    value1+value2,(value1+value2)/2.0);

  printf("%d * %d と同じ掛け算は，\n",
    value1,value2);
  int p = value1*value2;
  for(int cnt1=1;cnt1<=9;cnt1++){
    for(int cnt2=1;cnt2<=9;cnt2++){
      if(cnt1*cnt2==p && cnt1!=value1){
        printf(" %d * %d\n",cnt1,cnt2);
      }
    }
  }
  return 0;
}
```

この課題は，実際に授業でテストとして用いたものの一部である．テストは持ち込み不可で教科書だけでなくインターネットなどを一切見ることができずに，課題のプログラムを作成する．ただし#include などのスペルを度忘れしても大丈夫なように，最低限必要のことだけが書かれたサンプルプログラムを問題文に載せている．

3. C 言語タイピングソフト

キーボード操作を練習するために，画面に表示される日本語をタイプしていくソフトウェアがある．それらを総称してタイピングソフトと呼んだりする．そのタイピングソフトは，通常は日本語が表示されるわけであるが，そこにプログラミング言語を表示させるものが，本研究で提案するタイピングソフトである．日本語のタイピングソフトと同様に，打ち終わるまでの時間や正確さを競う．その外観

を図 1 に示す．

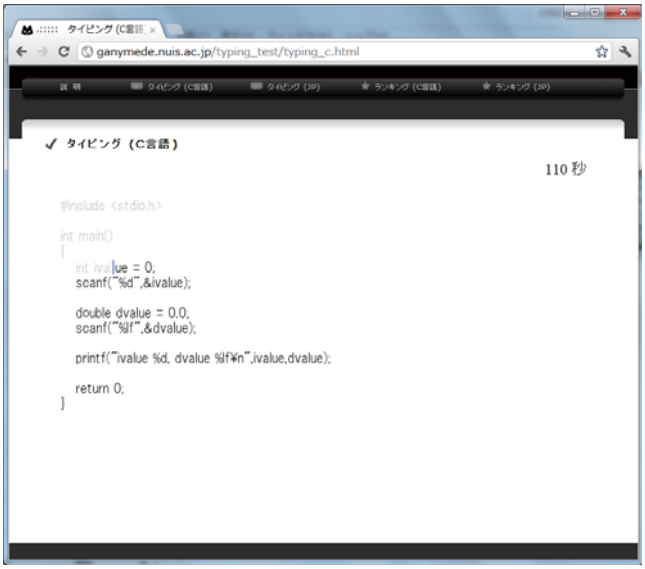

図 1 プログラミング言語学習用のタイピングソフト．

このタイピングソフトでは，ユーザは画面に表示される C 言語のプログラムを入力する．タイピングを開始する前にユーザ ID を入力し，そして開始ボタンを押すと C 言語のプログラムが表示される．その後は，ユーザはそのプログラムをひたすら打ち続ける．入力をミスをした場合のペナルティは特にない．用意されたプログラムを全て入力すると，そこまでにかかった時間，ミスをしたキー数が保存されて終了する．

打ち込むプログラムは，1 つの画面に収まる小さなプログラムを 3 つの打つ．1 つのプログラムの打ち込みがすべて終了すると，次のプログラムが画面に表示される．1 つのプログラムは 15 行程度のプログラムで，3 つのプログラムを通して条件式や繰り返し文などが一通り入っているようになっている．3 つのすべてのプログラムを打ち終わるためには，530 回程度のキーボードを打つ必要がある．キーボードを打つ回数は，プログラム中の空白を省略したりすることができるため，一定ではない．そして 3 つのプログラムをすべて打ち終わるまでの時間は，平均で 4 分 22 秒，速い人で 2 分弱，遅い人で 7 分以上かかる．

日本語タイピングのソフトの場合には，そのゲーム性を高めるために，表示される日本語はランダムに表示されることが多い．しかし，本タイピングソフトは学習を第一の目的としているため，いつも同じプログラムが同じ順序で表示される．

4. 分析するデータ

タイピングの速さなどの属性と，C 言語の理解を表 1 のデータによって測定する．タイピングソフトから取得するデータとして，タイピングの速さ（最大，最小，平均），速さのブレ，不正確さ，積極性を測定する．それぞれの値の

表 1 タイピングの速さなどの属性と C 言語の理解の測定方法．

データ項目	測定するデータ
タイピングの速さ (最大)(max time)	全て打ち終わるまでの時間の実験期間内の最大
タイピングの速さ (最小)(min time)	全て打ち終わるまでの時間の実験期間内の最小
タイピングの速さ (平均)(ave time)	全て打ち終わるまでの時間の実験期間内の平均
タイピングの速さのブレ (sd)	全て打ち終わるまでの時間の標準偏差
タイピングの不正確さ (error)	入力ミスの回数
タイピングに対する積極性 (count)	タイピングを実施した回数
学習開始時のタイム向上率（$r0$）	タイピングを実施した日時を横軸，タイピング速度を縦軸としたグラフを二次関数で近似し，学習開始時の二次関数の傾きを向上率 ($r0$) とする．タイムが急激に向上するとより大きな負の値となる．
学習終了時のタイム向上率（$r1$）	上記の二次関数の学習終了時の傾きとする．学習終了時までタイムが向上している場合には，負の数または 0 に近い数字になる．正の数の場合には，学習終了時にはタイムが悪くなっていることを表す．
C 言語の理解 (score)	C 言語授業のテストの点数

具体的な意味は，表 1 のとおりである．タイピングの速さの最大，最小，平均とは，実験期間内に実施する複数回のタイピングの中で打ち終わるまでの時間が最もかかったものを最大，最も速かったものを最小，そしてそれらすべての平均を平均としている．そして C 言語の理解については，C 言語の授業におけるテストの成績とする．さらにこれらに加え，授業開始時から終了時までのタイピングの変化を表す指標として，学習開始時のタイム向上率（$r0$）と，学習終了時のタイム向上率（$r1$）を導入した．これは，図 2，図 3 のように横軸をタイピングを実施した日時，縦軸をタイピング速度（すべてのタイプを終わるまでの時間）としてプロットして二次関数によって近似し，その二次関数の開始時点と終了時点の傾きを求めたものである．$r0$ の負の値が大きいものは図 2 のように実験期間内に大きくタイピング速度が向上したことを，$r0$ の負の値が小さいものは図 3 のように実験期間内のタイピング速度の変化が比較的に安定していることを表す．学習終了時のタイム向上率 $r1$ が正の値になるものは，学習終了時にはタイピング速度が落ちていることを示す．一方 $r1$ が 0 に近いまたは負の

値のものは，学習期間の最後までタイムが向上し続けたことを示す．

また，ユーザが初めてこのタイピングを実施したときのデータは，システムになれるまでに要した時間が含まれていると考えて，分析の対象としていない．

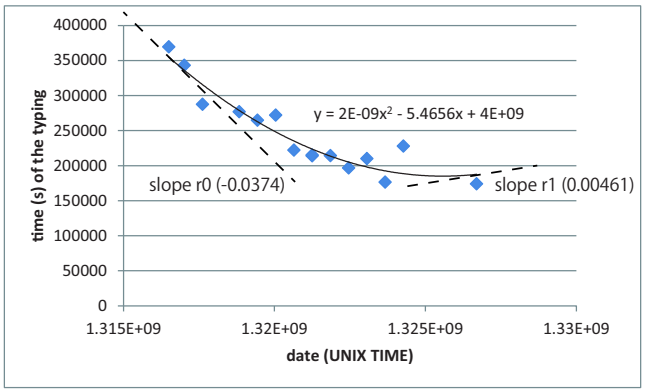

図 2 タイピング速さの変化例 1: 横軸はタイピングを実施した時間 (UNIX Time)，縦軸は全ての文字を打ち終わる時間 (秒)．

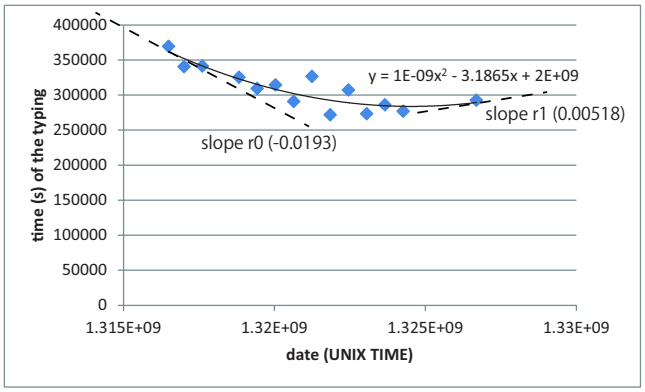

図 3 タイピング速さの変化例 2: 横軸はタイピングを実施した時間 (UNIX Time)，縦軸は全ての文字を打ち終わる時間 (秒)．

5. 実験概要と結果

5.1 実験の概要

表 2 に示す 2 つのプログラミング授業において授業の開始前に毎回のタイピングを義務付てデータを収集した．

プログラミングの授業は 1 週間に 1 回，4 か月間で計 14 回の講義・演習と，1 回の試験で構成されている．1 回の授業・演習は 90 分を 1 コマとして 2 コマ連続で行われる．タイピングは，授業開始前に必ず 1 回を行うように義務付けている．ただし，速くタイピングを完了した人は，もう一度行うことをしてもよいことにしている．タイピング以外の要素として，授業自体のスタイルは，学生に課題を与え，その課題を解くための技術的な内容について説明するという方法である．学生は全ての課題を解くとその日の授業が終わるというものである．

テストでは，教科書やインターネットなどを一切見ることができない状態で，与えられた課題に沿ったプログラムを作成する．

表 2 科目名と学習期間

科目名	C1	C2
受講者	プログラミング言語の未経験者向け	C1 を終了した学生の初心者向け
期間	第 1 回 2011 年 4 月 5 日〜 2011 年 7 月 19 日 第 2 回 2012 年 4 月 10 日〜 2012 年 7 月 17 日 第 3 回 2013 年 4 月 9 日〜 2013 年 7 月 23 日	第 1 回 2011 年 9 月 20 日〜 2012 年 1 月 23 日 第 2 回 2012 年 9 月 24 日〜 2013 年 1 月 21 日
内容	変数，配列，if 文，while 文，for 文	C1 の復習，関数，構造体，ポインタ
学生数	第 1 回 34 第 2 回 20 第 3 回 33	第 1 回 38 第 2 回 38
授業回数	14 回 1 回 180 分	14 回 1 回 180 分
タイピングの実施	授業開始前に最低 1 回の実施を義務付ける．	

5.2 実験結果

タイピングを実施した年度と，実施しなかった年度の授業におけるテストの成績を表 3 に示す．タイピングを実施するようになって成績が向上することは見られない．

表 3 タイピングを実施した年度と実施しなかった年度のテストの成績．テストは 30 点満点であり，与えられた課題のプログラムを作成するものである．

	C1			C2			
年度	平均点	標準偏差	受講者数	平均点	標準偏差	受講者数	タイピング
2008	24.5	7.7	31	21.3	5.9	31	なし
2009	21.3	9.2	38	15.7	9.6	35	なし
2010	26.5	7.2	19	16.7	9.5	27	なし
2011	19.1	6.2	34	16.2	10.5	38	実施
2012	19.9	6.6	20	13.2	7.4	38	実施
2013	19.8	6.9	33				実施

表 4 にそれぞれのデータ項目と C 言語理解との相関関係を示す．*は 5%有意水準，**は 1%有意水準を表すものとする．未経験者向けである C1 と初心者向けの C2 の両方を足し合わせたデータで相関がみられた箇所は，タイピングを実施した回数とテストの成績のみである．C1 と C2 を比較した場合，より初歩の授業である C1 の方が，タイピングとテストの成績に相関があることが分かる．

表 4 実験で得られたすべてのデータを対象とした，それぞれのデータ項目と C 言語理解との相関関係.

データ項目	C1+C2	C1	C2
タイピングの速さ最大 (max time)	相関なし	-0.289 **	相関なし
タイピングの速さ最小 (min time)	相関なし	-0.313 **	-0.247 *
タイピングの速さ平均 (ave time)	相関なし	-0.298 **	相関なし
タイピングの速さのブレ (sd)	相関なし		
タイピングの不正確さ (error)	相関なし		
タイピングに対する積極性 (count)	0.172 *	0.249 *	0.247 *
学習開始時のタイム向上率（$r0$）	相関なし		
学習終了時のタイム向上率（$r1$）	相関なし		

次にそれぞれの相関があったところについての散布図を示す．図 4 は，未経験者向け授業である C1 と初心者向け授業の C2 の両者のデータを足し合わせたデータにおけるタイピングを実施した回数とテストの成績の関係である．この図から分かるように，相関は 5%有意水準において統計的には認められるが，これら 2 つのデータ間に何かしらの関係があるとは思えない．ごく一部のデータが他のデータに比べて大きく逸脱しているため，相関が計測されてしまったのだと思われる．

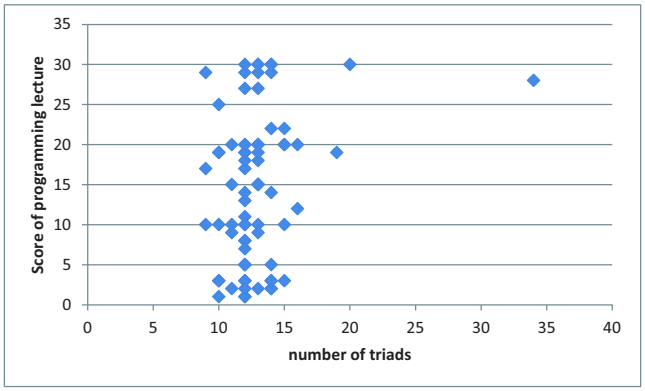

図 4 未経験者および初心者向け授業のすべての受講生を対象とした，タイピングを実施した回数とテストの成績の関係．相関係数 0.172(5%有意水準).

一方，C1，C2 におけるタイピング速度とテストの成績には一定の相関があると考えられる．図 5 は C1 における，図 6 は C2 におけるタイピング速度（最小）とテストの成績の関係である．特に図 5 の C1 におけるデータでは，右下がりの負の相関があるように見える．この場合の負の相関とは，タイピングが速いほど，テストの成績が良いことを示す．

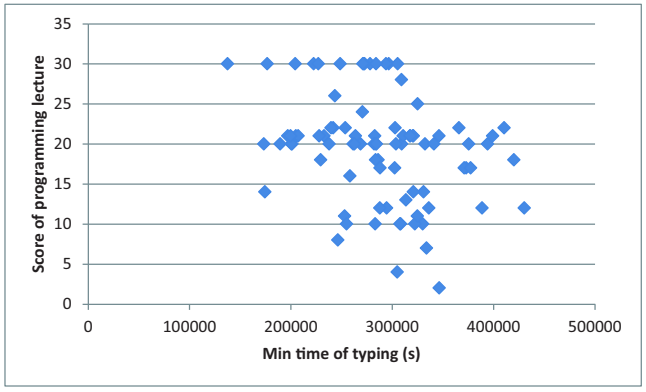

図 5 未経験者向け授業 C1 の受講生を対象とした，タイピングの速度（最小）とテストの成績の関係．相関係数-0.313(1%有意水準).

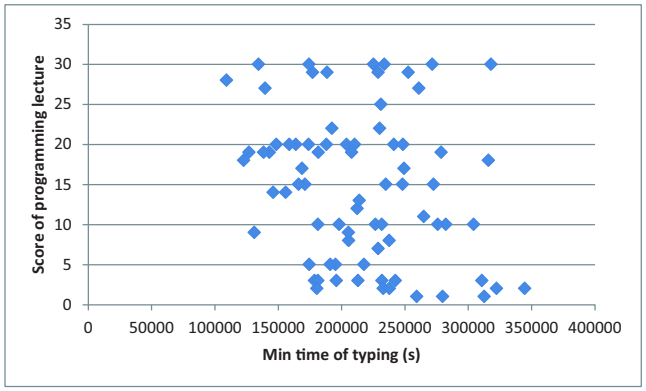

図 6 初心者向け授業 C2 の受講生を対象とした，タイピングの速度（最小）とテストの成績の関係．相関係数-0.247(5%有意水準).

6. 関連研究

プログラミング初心者に対する教育方法の研究は数多くある．例えば Robins ら [5] は，初心者に対する学習を，Knowledge，Strategy，Model と分類し，教師はそれを意識した授業を展開するとよいと述べている．Knowledge とは，条件式や繰り返し文といったプログラミングの基本的な知識である．Strategy は，それらをどのように組み合わせてプログラムを作成するのかを考えるところ，そして Model は，さらに上位のプログラム全体の構造を設計することである．我々の取り組んでいるプログラミング学習は，この分類に従うと Knowledge と Strategy を教授していることになる．Model の部分は，本研究では対象外としている．また初心者に対する学習の分類は，Ali ら [6] の Scaffolding，Concept Mapping，Constructivism などもある．

プログラミング学習だけに限らず，学習全般に言えることだと考えられるが，学習者にきめ細かいケアをすることによって学習を促進させる研究報告がある．Vihavainen ら [7] は，熟達者との共同作業によりプログラミング学習の初心者は必要な時に必要な情報をすぐに得ることがで

き，そして自分自身が作成するプログラムに対するフィードバックを常に得られることが重要であると述べている．おそらくきめ細かいケアは，学習にとって最も重要なことである．しかしそれは同時に，学習者が教育を受けるために多くのコストを必要とすることでもある．

黒板やホワイトボードに説明を書きながら教師がプログラムの構文を説明し，学生は主にその説明を聞いているという従来のタイプの授業は効果がないと報告されている [6]．また同じように，プレゼンテーションソフトを用いてあらかじめ用意されたプログラムコードをスクリーンなどに表示して説明することも効果がないと言われている．そのような背景から Live Coding と呼ばれる，教師が学生の目の前で実際にプログラムをタイプする説明方法が提唱されている [8]．そしてその評価 [9] より，その可能性が示されている．この Live Coding が良いという理由には，授業中の緊張感が増す等の様々な要因が考えられるが，1 つにはプログラムのコードを学生が集中して 1 つずつ見ることができることだと思われる．プレゼンテーションソフトなどを用いて一度に表示されるプログラムコードは，学生はざっと全体を見渡すだけで 1 行 1 行を克明に見るわけではない．一方，Live Coding によってプログラムコードが生成される過程を見せると，学生は 1 つずつのコードをより意識しやすくなるのだと思われる．このような効果は，本研究におけるソースコードのタイピングによってさらに促進されると考える．

「はじめに」で述べたように Dehnadi ら [1] は，60%の人間がプログラミングに向いていないと主張している．プログラミングに向いているということを決める要素は，混沌の中から法則性を発見し，新しい状況にその法則を正しく適用できる能力であるという．このようなプログラミングの素質に関する研究は，他にもある．Simon ら [10] は，簡単な検索アルゴリズムを言葉で分かりやすく説明できるかどうかがプログラミングの授業の成績に関係していると主張する．また数学や抽象化の能力がプログラム学習やコンピュータ科学の理解に必要であると述べる研究は多くある．Bennedsen ら [11] は，高校数学がコンピュータ科学の素養として重要であると述べている．抽象化能力については，Perrenet ら [12], [13] は，アルゴリズムをどの程度抽象化して考えているかによってコンピュータ系科目の成績が決まってくると述べている．しかし抽象化能力については，同じく Bennedsen ら [14], [15] の研究によって，プログラミングの習得およびコンピュータ科学の学位取得に相関が無いとも報告している．ソースコードのタイピングは，数学的思考や抽象化能力を促進させるためのものではない．よって抽象化能力や法則性を見つけ出す能力がプログラミング技術の修得に必須であるならば，タイピングによる鍛錬だけではプログラミング技術を多くの人に享受するという目的は達成できないのかもしれない．しかしもしも抽象化能力のようなものが必須で無いならば，タイピングの鍛錬によってプログラミングに挫折する人を減らすことが不可能であるとは言い切れない．

タイピングを何かしらの学習に応用する研究はあまり例がない．その中でも，Omori ら [16] は，発達障害の人に英単語のスペルを学習させることを計算機によって支援する研究を行っている．英単語のスペルは，ペンで紙に書いて覚えることが従来からよく行われている．しかし発達障害の人は，このペンによる学習が困難である．そこで計算機の画面に絵とその英単語を表示し，1 つは複数のスペルの正解候補から 1 つを選択するという練習，もう 1 つはそのスペルをキーボードで入力するという練習をした後で，スペルを理解しているかどうかのテストを実施した．その結果から，キーボードを用いた方がスペルの学習が良いことが分かった．これは，ただ全体を眺めているだけではなく自分自身で表示されている文字を再現するという行為が学習を促進していると考えることができる．ソースコードのタイピングも同じように，教科書に書かれたプログラムを眺めるだけでなく，打ち込んでみるというところに学習効果があると考える．

また，Thomas ら [17] はプログラムを作成している途中のタイピング速度とプログラミングに関する講義の成績には相関があることを示している．ここで言うタイピングの速度では，単純にプログラムを打ち終わるまでの時間ではなく，単語と単語の間のキーボードを打つまでの空き時間に注目している．キーボードを打つ速度は，単語の一番初めの文字で最も時間がかかると言われている．それは，単語を 1 つの塊として理解して，それから打ち出すという過程があると考えられているからである．その単語を 1 つのものとして理解する時間がかかる人ほど，タイピング時の単語と単語の間のキーボードを打つ待ち時間が長くなり，そしてそのような人は，プログラミングの理解がよくないという結果が出ている．我々の研究と同様にキーボードから得られる情報をプログラミングの理解に利用することを行っている．異なる点は，我々は鍛錬としてプログラムを打ちこむことを考えている．その鍛錬が進めば結果的にタイピング速度が速くなると考える．一方，Thomas らは，プログラミングをしている過程を詳細に見ることによって，出力物のプログラムからだけでなく学生のプログラムの理解度合を測ろうとしている．

7. おわりに

本研究は，ソースコードのタイピングによってプログラミングの習得が促進されるのか，また，ソースコードのタイピングの成績とプログラミングの授業のテストの成績には何かしらの相関があるのかを調べてきた．2 年間半の実験によりのべ 163 名の学生に対して実験を行い，テストの成績については向上しなかったが，ソースコードのタイピ

ング速度とテストの成績には一定の相関があることを発見した．

そこで「はじめに」で示した疑問の回答は，次のようになった．

---

(1) C 言語タイピングを行うことによって C 言語授業のテストの成績はよくなるのか?
- タイピングを実施したことによるテストの成績の向上は見られなかった．

(2) C 言語タイピングと C 言語授業のテストの成績に何かしらの関係があるのか?
- タイピングの速い人ほどテストの成績が良いことが分かった．

タイピングの速度とテストの成績に関する相関関係は，因果を示すものではない．因果関係とは，タイピング速度が向上すると C 言語の理解が高まる，ということである．もしもこのような因果関係が存在している場合，タイピングを練習することによって C 言語の理解を向上させることが可能になる．しかしこのような因果関係がない場合，すでに発見された相関関係は，次のようにして解釈することができる．

- 偶然に相関が計測された?
  - 引き続きデータを取り続けて検証する必要がある．
- 因果の流れが逆である?
  - 因果が逆とは，C 言語の理解が促進ことによってタイピングが速くなるということである．
  - この視点を考慮して，今後はプログラムのどの箇所でタイピングが遅いかなどの詳細な分析を行う．
- 因果の上流に共通する要因がある?
  - コンピュータ系の素養がある人は，タイピングも速いし，プログラミングもできるというような場合である．
  - 授業の開始時のプログラミング素養に関するテストの実施，また，他の成績との相関も分析したい．
- 分析したデータが選抜されてしまっている?
  - C1/C2 の履修は必修ではないため，そもそも最初からプログラミングを敬遠する学生のデータは取得できていない．そのような意味のバイアスはデータにかかっている．しかし，その前提の上でタイピングとテストの成績の相関，および因果を分析することができると考える．

さらに，C 言語ではない通常の日本語のタイピングの速度と，C 言語の授業の成績との相関については調査していない．今回の実験では，C 言語タイピングとテストの成績に相関があることが発見されたわけであるが，そもそも C 言語ではなく日本語のタイピングであっても同様の結果が得られた可能性がある．この点については，今後の課題としたい．

参考文献

[1] S. Dehnadi, R. Bornat: The camel has two humps, Middlesex University Working Paper (2006).
[2] Donald R. Woods: Problem-Based Learning: Resources to Gain the Most from PBL, Donald R. Woods, ISBN:9780969872528, (1997).
[3] O'Kelly, Jackie and Gibson, J. Paul: RoboCode & problem-based learning: a non-prescriptive approach to teaching programming, Proceedings of the 11th annual SIGCSE conference on Innovation and technology in computer science education, ITICSE '06, pp. 217-221 (2006).
[4] 喜多一, 岡本雅子, 藤岡健史, 吉川直人: 写経型学習による C 言語プログラミングワークブック, 共立出版 (2012).
[5] A. Robins, J. Rountree, and N. Rountree: Learning and teaching programming: A literature review. Computer Science Education, 13(2):137-172, (2003).
[6] Ali, S.: Effective Teaching Pedagogies for Undergraduate Computer Science. Mathematics and Computer Education, 39(3), 243-257, http://www.editlib.org/p/67913 (2005).
[7] A. Vihavainen, M. Pakula, and M. Luukkainen: Extreme apprentinceship method in teaching programmers for beginners. SIGCSE, pages 93-98, (2011).
[8] Paxton, John: Live programming as a lecture technique, J. Comput. Sci. Coll., vol.18 no.2, pp.51-56 (2002).
[9] Rubin, Marc J.: The effectiveness of live-coding to teach introductory programming, Proceeding of the 44th ACM technical symposium on Computer science education, SIGCSE '13, pp.651–656 (2013).
[10] Simon and Cutts, Quintin and Fincher, Sally and Haden, Patricia and Robins, Anthony and Sutton, Ken and Baker, Bob and Box, Ilona and de Raadt, Michael and Hamer, John and Hamilton, Margaret and Lister, Raymond and Petre, Marian and Tolhurst, Denise and Tutty, Jodi: The ability to articulate strategy as a predictor of programming skill, Proceedings of the 8th Australasian Conference on Computing Education - Volume 52, ACE '06, pp. 181-188, (2006).
[11] Bennedsen, J. and Caspersen, M. E., An investigation of potential success factors for an introductory model-driven programming course, First International Workshop on Computing Education Research, Seattle, WA, USA, pp. 155-163, (2005).
[12] Perrenet, J., Groote, J. F., and Kaasenbrood, E., Exploring students' understanding of the concept of algorithm: levels of abstraction, 10th Conference on Innovation and Technology in Computer Science Education, Lisbon, Portugal, pp.64-68, (2005).
[13] Perrenet, J. and Kaasenbrood, E., Levels of abstraction in students' understanding of the concept of algorithm: the qualitative perspective, 11th Conference on Innovation and Technology in Computer Science Education, pp. 270-274, (2006).
[14] Bennedsen, Jens and Caspersen, Michael E.: Abstraction ability as an indicator of success for learning object-oriented programming?, SIGCSE Bull. vol.38, no.2, pp.39-43, (2006).
[15] Bennedssen, Jens and Caspersen, Michael E.: Abstraction ability as an indicator of success for learning computing science?, Proceedings of the Fourth international Workshop on Computing Education Research, ICER '08,

pp. 15-26, (2008).
[16] Omori, M. , Sugasawara, H. & Yamamoto, J.: Acquisition and Transfer of English as a Second Language through the Constructional Response Matching-to-Sample Procedure for Students with Developmental Disabilities. Psychology, 2, 552-559. doi: 10.4236/psych.2011.26085, (2011).
[17] Thomas, Richard C. and Karahasanovic, Amela and Kennedy, Gregor E.: An investigation into keystroke latency metrics as an indicator of programming performance, Proceedings of the 7th Australasian conference on Computing education - Volume 42, ACE '05, pp. 127-134, (2005).