研究ノート

プログラミング実習における学習効果について

――自作プログラミング対模写プログラミングの比較――

河　村　一　樹

東京国際大学論叢　人間科学・複合領域研究　第9号　抜刷
2024年（令和6年）3月20日

研究ノート

プログラミング実習における学習効果について
――自作プログラミング対模写プログラミングの比較――

河　村　一　樹

Learning Effectiveness in Programming Practice
—Homebrew Programming vs. Copycat Programming—

KAWAMURA, Kazuki

Abstract

The author has been in charge of programming education since 2013. Until FY2022, the practice has been based on self-programming. However, the problem of students copying and pasting was discovered, and it became necessary to take some countermeasures. In order to prevent them from copying and pasting, the author came up with the idea of moving to copying programming (referred to as copying programming in this paper), which assumes copying. To this end, the author decided to conduct a demonstration experiment in FY2023 to compare the learning effects of home-made programming and imitation programming.

目　次

はじめに

大学におけるプログラミング教育は，以前から学部学科を問わず，幅広く行われてきている。これには，一般教育の一環としての一般情報教育があげられる。これに合わせて，情報処理学会では一般情報教育（GE: General Education）委員会が設置されており，筆者は幹事（2003-2007）そして委員長（2007-2016）さらに委員（2016-）として研究調査活動を継続している。

一般情報教育委員会では，2007年にGEBOK（GE Body Of Knowledge）を策定しており，その中の「GE-ALP アルゴリズムとプログラミング」においてプログラミング教育に関する知識体系を明らかにした。その基本的な方針としては，特定の実用言語の習得を目指すのではなく，コンピュータサイエンスの頻出概念[1]を習得すると同時に身につけていく技能としてプログラミングをとらえることとしている。この方針に準拠したプログラミング教育の事例として，東大における全学プログラミング教育があげられる。

さて，商学部でもプログラミング教育を実施しており[2]，筆者は2013年度から科目「プログラミング基礎」（旧名称「プログラミング実習」）を担当している。当初は，商学部の学生の履修が主だったが，数年前からは他学部の学生の履修の方が多くなっている。これは，他学部ではこのような科目がなく，プログラミングに興味関心のある学生が履修するようになったからではないかと思われる。このことから，商学部独自の科目というよりも，一般情報教育の科目という位置づけに変わったといえる。

「プログラミング基礎」は，当初2コマ連続での開講だったが，2016年度からはペアコマ（月・木1コマずつ）に変更になった。また，2014年度からは講義をせずに個別指導だけの授業形態として，現在に至っている。

講義をせずに個別指導だけに切り替えた理由は，プログラミングスキルの習得には個人差があり一斉授業ではミスマッチが生じること，個別に質問を受けることでわからないことをクリアにして実習を進めてもらうこと，これによってコピペを防ぐことなどがあげられる。

その個別指導において，ある問題があることに気づいた。それは，自学自習がうまくできていない学生が全く質問しない（できない）こと，その結果他の学生のソースコードをそのままコピー・アンド・ペースト（以降，コピペと略す）して提出しているという疑惑であった。半期の間，個別指導をしているので，学生の顔と名前が一致するようになり，最終アンケート（実名記入）でコピペの有無を聞いたところ，疑惑のある学生からはコピペしたことがあるとの回答を得た。このため，プログラミングのコピペに対して新たな策を講じる必要が出てきた。

本来，コピペは他人のソースコードをそのまま写し取り自分のプログラムとすることである。実は，このことをうまく利用したものに写経プログラミングがある。ただし，本稿では，模写プログラミングとする。模写プログラミングは，写経プログラミングと同じことを表しているが，「写経」というと邪念を払拭し無心でという意味合いが含まれるが，プログラミングでは何らかの意思（理解しようとする行為）がある。このため，「写経」とせずに，「模写」という言葉を使っている。

実習に模写プログラミングを取り入れることによって，コピペの必要がなくなるとともに，学生のプログラミングスキルを向上させるための工夫を取り入れることを考えついた。

本稿では，今までやってきた自作プログラミングと模写プログラミングのプログラミングスキル評価を比較するための実証実験について論じる。なお，実証実験については2023年に実施するため，秋学期終了後に取得した学習データを比較検討する予定である。

1. 大学における一般情報教育としてのプログラミング教育

前述したGEBOKは，2007年に公開され[3][4]，その後，2017年に改訂された[5]。最新版の「GE-ALP アルゴリズムとプログラミング」では，次のようなエリアを設定している。

『コンピュータによる具体的な問題解決方法を、適切なアルゴリズムとして表現できるようになることを学ばせる。さらに、実際にコンピュータでそのアルゴリズムに基づいて処理を行わせるのに必要なプログラミング能力を修得させる。』

プログラミング教育で，よく指摘されることとして，言語のシンタックスや文法だけを教えることに終始していることがあげられる。これは，教える側にとって教えやすいことや実習課題も作りやすいことによる。しかし，このような教育では，汎用性や応用力が身につかないことから他の言語に対応できなかったり，アルゴリズムの理解が不足しているため問題を分析して解法を表現することもできないといった問題が生じる。

情報処理学会一般情報教育委員会では，こういった指摘に対して，コンピュータサイエンス寄りの知識体系をプログラミング教育に取り込むことを提案している。コンピュータサイエンス寄りということは，頻出概念で取り上げている大規模問題の複雑さ・形式的モデル・効率・抽象化レベルといったものをプログラミング教育に取り込むということである。また，GE-ALPに準拠した教科書[6]を，一般情報教育委員会推奨という形で発刊している。大学毎に，これらを取り込んだ形でのプログラミング教育を提案しているわけである。

この事例として，東京大学の全学プログラミング教育があげられる[7]。東京大学では，1・2年次に理系も文系も全員教養学部に所属して全学的なリベラル・アーツ教育を受けることになっている。その中で，科目「情報」「アルゴリズム入門」といった科目が開講されており，プログラミングを学ぶ意義については次のように指摘している。

『まず，問題を分析し解法を表現するための，問題解決のための言葉としての役割です。もう一つは，コンピュータの動きを体感するためのツールとしてです。これらは，コンピュータや情報システムの動作を理解し活用するには欠くことのできないものでしょう。』

また，授業の設計思想については，

『プログラミングができるようになることは主たる目的ではありません。あくまで学習のための手段と位置付けています。試験や課題でも，プログラミングスキルよりは，情報科学の基礎を理解しているかどうかに重きを置いています。』

としている。

本稿で取り上げる科目「プログラミング基礎」も，前述したように一般情報教育としての位置づけに変わってきていることから参考になる。

2. 科目「プログラミング基礎」の実施経緯

ここでは，2013年度から担当してきた科目「プログラミング基礎（旧プログラミング実習）」の授業概要について取り上げる。この中で，特記すべき事項として，新型コロナウィルス

（COVID-19）感染症があげられる。これによって，大学教育でも一時期多大な影響を受けることになった。そこで，コロナ禍に応じた変容として記述する。

2.1　コロナ禍以前（2013年度から2019年度まで）

（1）2013年度

前任者から引き継いだ科目「プログラミング実習」は，半期2コマ/週，2単位，選択，1年次から履修可という実習を含む授業であった。教養としてのプログラミング教育という位置づけで，プログラマの育成ではなく問題解決力や論理的思考力の育成，それらを通してコンピュータをある程度ホワイトボックス化することを目指すものであった。授業の実施については，多くの大学で実施されてきた伝統的な形態[8]である1コマ目は講義で2コマ目は実習というパターンを踏襲した。

プログラミング言語は，JavaScriptとした。これは，スクリプト言語であることから初心者にとっては敷居が高くないこと，構文規約がきつくない（変数宣言が自由，文末のセミコロンがなくても動く，完全なフリーフォーマット，…）こと，テキストエディタとブラウザさえあれば動くこと，プログラミングの結果を即画面で確認できること，ブラウザ（Google Chrome）によってはデバッガーがあること，などの理由からである[9]。

これに合わせて，教科書[10]を用意した。その章構成は，**図1**のようになる。なお，第7章については2017年度から変更した。これは，（旧）第7章はHTMLがバージョン4に対応したJavaScriptのプログラミングを例題にしていたが，バージョン5に移行するのに合わせて改変したためである。また，JavaScriptを用いたゲームプログラミングの方が学生の関心も高いということもあった。

さて，期の後半から，落ちこぼれた学生によるプログラムのコピペ問題が発覚した。このため，最終回において個別に面接試験を実施した際に，コピペについて調べた結果，半数弱の学生が部分的にでも経験していたことが明らかになった。その理由について学生に聞いたところ，わから

第1章　情報[処理]教育	第5章　JavaScriptの基本編	4．フォーム
1．情報処理教育と情報教育	1．画面への出力	⬇（途中から）
2．情報処理教育におけるプログラミング	2．変数の扱い	
3．情報教育におけるプログラミング	3．演算式の扱い	第7章(新)　CSSを利用したWebサイトへの 実用的適用
第2章　プログラム	4．画面からの入力	1．Webサイトの基本的構造とCSS
1．プログラムとは	5．選択文	2．CSSについて
2．プログラミングパラダイム	6．繰返し文	3．CSSの記述
3．プログラム言語	7．配列の扱い	4．CSSを外側に記述する
4．プログラム動作環境	8．関数の扱い	5．JavaScriptを使ってCSSをコントロールする
第3章　アルゴリズム	第6章　JavaScriptの応用編	6．文字列を挿入する
1．アルゴリズムとは	1．整列のアルゴリズム	7．アニメーション
2．アルゴリズムの記述	2．探索のアルゴリズム	8．簡易ゲームの制作
3．アルゴリズムの評価	3．再帰のアルゴリズム	
第4章　JavaScript	第7章(旧)　JavaScriptの実用編	
1．JavaScriptとは	1．ウィンドウの扱い	
2．JavaScriptの動作環境	2．文字の編集	
3．JavaScriptの書き方	3．画面の編集	

図1　教科書の章立て

ないことがわからないままになってしまう，講義のペースについていけなくなる，提出が遅れるとまずいのでついコピペしてしまうと語った。

この原因として，わからないことをわからないままにした結果，独力でプログラミングができなくなったことがあげられる。プログラミングは，自分でキー入力しデバッグすることで，構文の意味やロジックの組み立てなどがわかってくるという特徴がある。このためコピペに頼っていては，学習効果がゼロとなり，教員・学生ともに不幸といえる。

(2) 2014年度

前年度発覚したコピペ問題を解決するには，わからないことをわからないままにしないように，また，自分でプログラミングを行うという学習姿勢を維持できるように，講義は一切せずに個別指導を中心に実習を進めるような授業を試みることにした[11]。

講義を行うと，学習の進捗は学生ではなく教員に依存することになりやすい。教員はシラバス通りの授業進捗を守るために先に進んでいく。その結果，授業についていけずに遅れた学生は，落ちこぼれたままとなり易い。それだけでなく，一斉授業であるが故に，教員にとっても学生個々人の理解度を把握することが難しくなる。さらに，プログラミングスキルは，個々人による差が生じやすいだけでなく，本科目の履修前にプログラミングの経験があるかないかがすでに差になる。

これに対して，個別指導では，教員はチュータあるいはメンターとしての役割を担うことになる。学生のわからないことについて懇切丁寧に説明し，学生がわかるまで指導を行う。必要であれば，学生のソースコードを直接見たり，プログラムを動かしながら間違えの箇所を指摘することもある。進捗が遅れ気味の学生に対しては，自学自習の状況を把握してどのように改善すればよいかのアドバイスをする。さらには，もう少し頑張るようにと声がけすることで心理的な応援を図ることもできる。これらによって，学生も徐々にプログラミングの面白さに気づき，次の実習課題に取り組むモチベーションを高めることができるようになる。

(3) 2015年度

2014年度は個別指導を中心に進めたが，学生の進捗がまちまちになるだけでなく，クラス全体の進捗が当初想定していたものより遅れるという事態に陥った。これは，実習課題の結果（プログラムのソースコードと実行結果の画面キャプチャ）を印刷したものを毎週実習室で受取り，チェックした結果を次週に戻すというやり方に問題があった。つまり，学生からみると，早く実習を終えても最大7日間待たされることになり，その分実習が遅れていくことになる。

そこで，2015年度からは，新たにグループウェアであるサイボウズLive[12][1)]を導入した。これによって，学生はいつでも実習課題を提出できるようになり，かつ，提出時刻が学習ログとして記録されるので学生の進捗状況を正確に把握できるようになった。また，掲示板機能を学生のQ&Aとして利用することで，学生からの質問にも答えることができるようになった[13]。

(4) 2016年度

2016年度には，大学全体として教務制度に大幅な改変が行われた。一つはクォータ制によるペアコマ（2コマ授業については別曜日に1コマずつ実施）の採用，もう一つはLMSとしてのMoodleの運用である。

クォータ制については，他大学と同様に，授業を短期間に集中することで教育効果を向上させること，履修を工夫することで短期留学・ボランティア活動・インターンシップ活動に参加できるようにようになることを目指したといえる。これによって，2015年度までは月曜日4・5コマの2コマ連続であったが，2016年度からは月曜日3コマ・木曜日3コマと別曜日での開講に変わった。

実習が中心となる授業では，1週間も空けずに実習を繰り返すことができるので，学生も集中できて捗ることがわかった[14]。

Moodleについては，大学全体として教材管理や授業運営のICT化を進めるという方針が打ち出されたことにより導入された。そこで，本科目において，サイボウズLiveの代わりに，Moodleを利用することにした。サイボウズLiveの掲示板機能についてはMoodleの[フォーラム][2]を，サイボウズLiveの共有フォルダ機能についてはMoodleの[課題]を，それぞれ使うことにした。

このように，2016年度以降は全面的にMoodleを用いた授業に切り替えた。教科書に相当するテキストは章節毎にMoodleの[ファイル]，実習課題はMoodleの[課題]，実習課題毎のヒントと実行結果はMoodleの[ファイル]，質疑応答（全員閲覧可）はMoodleの[フォーラム]，実習課題の評点はMoodleの[評定]として，それぞれアップロードすることにした。これによって，学生は授業中だけでなく授業外（放課後）でも，Moodleにアクセスできる環境さえあれば自学自習できるようになった[15][16]。

(5) 2017年度

2017年度は，実習課題をMoodleの[課題]から[小テスト]に変更した。[小テスト]では，図2のような構成になる。

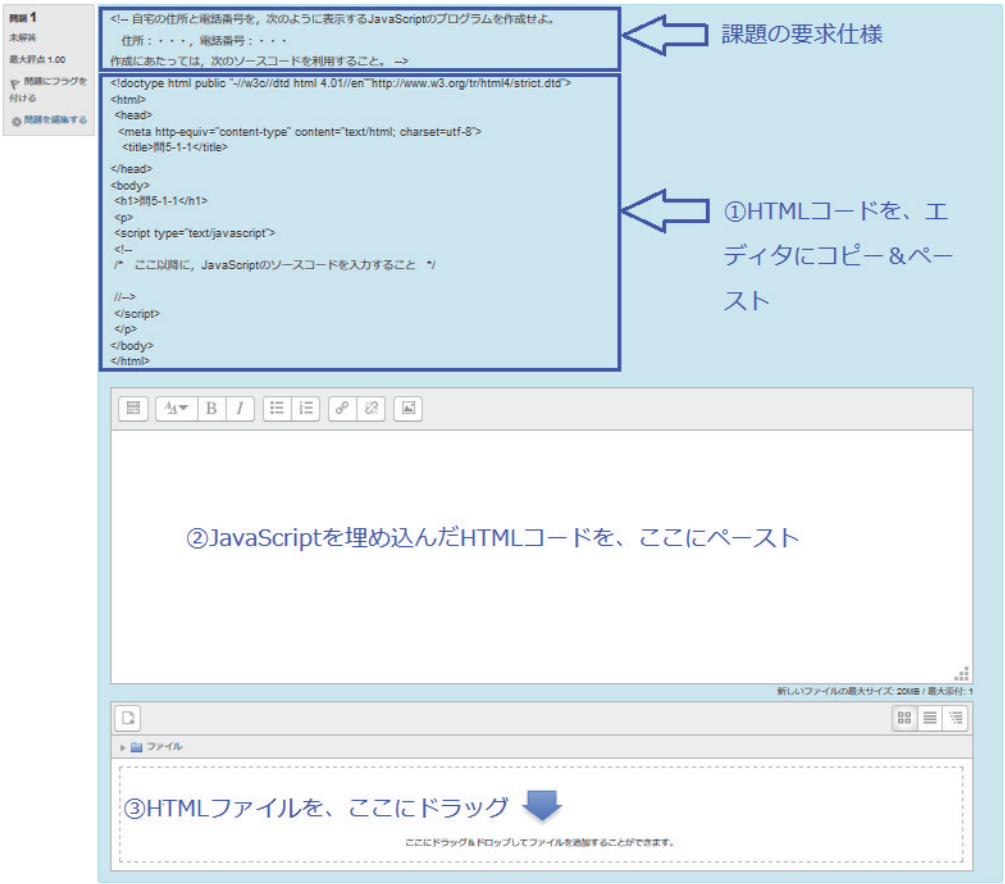


図２　Moodle[小テスト]の構成

[小テスト]の問題タイプは[作文問題]としており，これによって**図2**にあるように，上段に問題文の表示と，下段に回答の自由記述とさらに必要であればファイルをアップロードできる欄が用意される。具体的には，上段において，実習課題の問題となるプログラムの日本語仕様をHTMLのコメント（<-- から --> まで）とし，その下にHTMLのソースコードを載せている。JavaScriptの文は，HTMLのscriptタグ内で入力するようにしてある。中段の欄は出来上がったHTMLのソースコードをコピペする場所とし，下段の欄にはそのhtmlファイルをドラッグする。

学生は，[小テスト]にあるHTMLのソースコード（①に該当）をコピーしてテキストエディタ（Visual Studio Codeを推奨）にペーストする。そして，実習課題に則したJavaScriptのコードを入力し，文字コードをUTF-8としてhtmlファイルを保存する。その上で，デバッグを行うことによって，構文エラーやロジックエラーをつぶしてプログラムを完成させる。プログラムができたかどうかは，[ヒントと実行結果]の画面と照合し一致しているかどうかで判断する。出来上がったら，**図2**の中段（②に該当）にソースコードをペーストするとともに，下段（③に該当）にhtmlファイルをドラッグしてテストを完了する。

以上の動作に対して，学生が実習課題をコピペした時刻は[開始日時]，アップロードを終えた時刻は[受験完了日時]，そして，その間の時間は[所要時間]という形で学習ログに記録される。これによって，実習課題毎に，学生の学習時間の傾向をつかむことができるようになった。

（6）2018年度

2018年度は，[フォーラム]を履修者全員が閲覧可から，履修者一人と教員だけの閲覧に切り替えた。もともと[フォーラム]は電子掲示板であり，誰でもが閲覧できるコミュニケーションツールである。それまでは，わからないことがある学生が質問をスプレッドとして挙げると，教員の方で回答を返信するという使い方をしていた。しかし，学生の中には自分のプログラミングしたソースコードをそのままコピペして質問してくることがあり，それを閲覧した他の学生が真似てしまうという問題が発覚した。

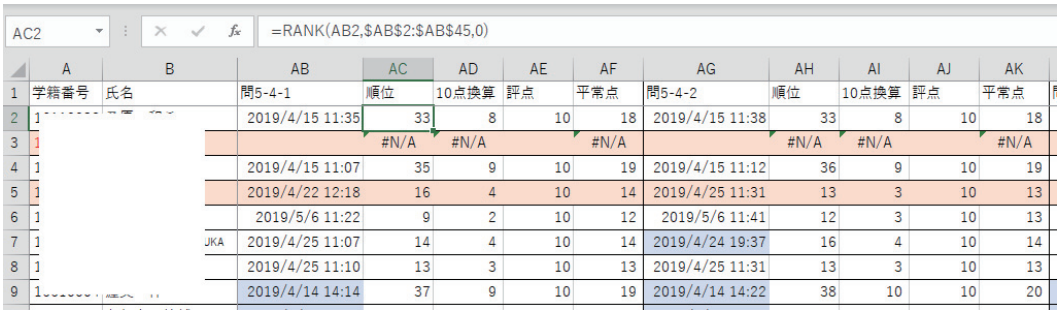
そこで，[フォーラム]について，**図3**のように制限（[利用制限]の[アクセス制限]）を与え，教員と学生の1対1の構成とし，履修者全員分の[フォーラム]を事前に登録した。

（7）2019年度

2019年度は，実習課題の評価方法についての変更を行った[17]。今までは，実習課題毎に，プログラムの仕様通り（「正確性」に相当）であれば10点，軽微なエラーであれば5点，重度なエラーであれば0点として平常点をつけた。

これに対して，実習課題を[課題]から[小テスト]に変えたことにより，実習課題毎に[受験終了日時]を学習ログとして取得できることに着目した。この[受験完了日時]によって，学生がどれだけ早く実習課題を終えたかの指標（「迅速性」に相当）となり得る。そこで，学習ログをExcelのスプレッドシートにエクスポートし，**図4**のように実習課題毎に平常点を算出した。

図３　Moodle[フォーラム] の設定



AC2　=RANK(AB2,$AB$2:$AB$45,0)

	A	B	AB	AC	AD	AE	AF	AG	AH	AI	AJ	AK
1	学籍番号	氏名	問5-4-1	順位	10点換算	評点	平常点	問5-4-2	順位	10点換算	評点	平常点
2	1		2019/4/15 11:35	33	8	10	18	2019/4/15 11:38	33	8	10	18
3	1			#N/A	#N/A		#N/A		#N/A	#N/A		#N/A
4	1		2019/4/15 11:07	35	9	10	19	2019/4/15 11:12	36	9	10	19
5	1		2019/4/22 12:18	16	4	10	14	2019/4/25 11:31	13	3	10	13
6	1		2019/5/6 11:22	9	2	10	12	2019/5/6 11:41	12	3	10	13
7	1	JKA	2019/4/25 11:07	14	4	10	14	2019/4/24 19:37	16	4	10	14
8	1		2019/4/25 11:10	13	3	10	13	2019/4/25 11:31	13	3	10	13
9	1		2019/4/14 14:14	37	9	10	19	2019/4/14 14:22	38	10	10	20

図4　平常点の算出

図4の[問5-4-1]は，学生が実習課題「問5-4-1」を終えてアップロードした時刻であり，学習ログから[受験完了日時]を取り出したものである。[順位]は，{全提出者数-(n-1)}とし，nは提出した順番（早い順に，1, 2, …，全課題提出者数とする。なお，同時刻の場合は同順となる。）とした。[10点換算]は，[順位]を10点から0点に換算（「迅速性」とする）し，[評点]は課題合格のときのみ10点（「正確性」とする）を与えた。そして，[平常点]は，[10点換算]+[評点]とした。以上によって，実習課題の評価は，「正確性」に「迅速性」を加味した形となる。

2.2　コロナ禍（2020年度から現在まで）

2019年12月に，中国で新型コロナウィルスの感染が発生し，世界に流行していくことになった。我が国でも，2020年1月16日に初の新型コロナ患者が報告され，一気に全国に感染が広がった。これに合わせて，本学でも感染対策として，対面授業からオンライン授業やオンデマンド授業に切り替える時期があった。なお，現在（第8波後）は感染状況も収まったこともあり，対面授業に戻っている。

（1）2020年度

2020年4月8日に，始めて緊急事態宣言が発出され，不要不急の外出自粛が求められるようになった。これにともない，全学的にZoom（有償版）が導入され，オンライン授業に向けての準備が進められた。2020年度の春学期は，約2週間遅れの4月16日から開始となった。すべての科目は，Zoomによるオンライン授業となったが，実習を伴う科目についてはいずれも開講せずとなった。これは，実習室では教員と学生のディスタンスが取りづらいこと，PC毎にアクリル板を設置する必要があること，室内の換気や機器（キーボード，マウス，プリンターなど）の消毒が毎回必要になること，などによるものと思われる。

秋学期からは，春学期未開講分の科目を含め，すべてオンライン授業での開催となった。このため，この半期は週9コマ分の授業を担当することになった。なお，「初年次演習」だけは1年生で少人数制の対話型授業でもあるため，対面授業（ただし，マスク着用，授業前の消毒，室内換気）となった。

本科目におけるZoomによるオンライン授業は，次のように進めた[18]–[20]。あらかじめミーティングルーム（ペアコマなので，月4コマ目と木4コマ目）を半期単位に設定しておく。これを，Moodleの「プログラミング基礎」コースのトップに貼りつけておく。これによって，学生は，自宅等のパソコン（ノートパソコンを推奨）で，Moodleを起動して，ミーティングルームのURLか，ミーティングIDとパスコードを指定することで入室する。

授業開始後，教員（ホスト）はZoomのブレイクアウトルームを開設する。ブレイクアウトルー

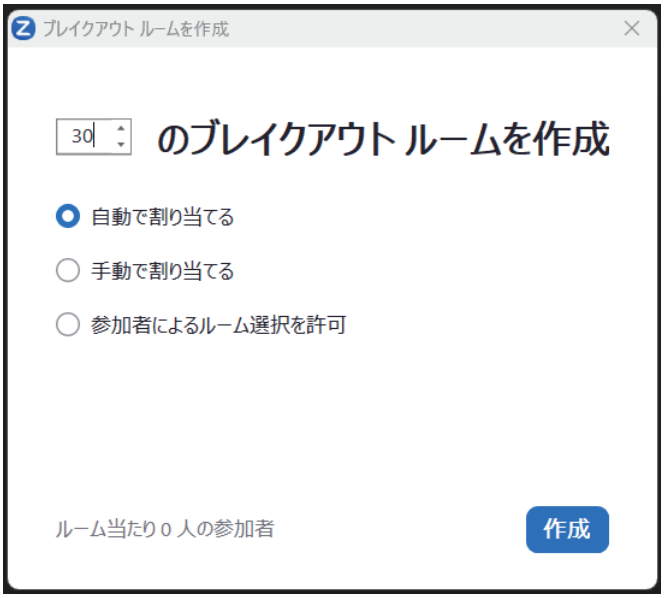


図５　ブレイクアウトルームの設定

ムは，図５のように設定する。

図５の「30」は，（遅刻を含めて）出席する学生数である。ここで学生分の数値にすることで，学生一人だけのルームとなる。「自動で割り当てる」にすることで，ランダムにルームが割り当てられることになる。これによって，個別指導の順番が毎回変わることになり，学生にも公平に扱われるという意識になる。

教員は，ルーム１から順番に入室する。そこで，学生と１対１での個別指導を行う。ノートパソコンであれば，標準でカメラとマイクが装着されているので，それらを用いて顔を見ながらアドバイスができる。それだけでなく，「画面の共有」を使うことで，学生のパソコンの画面を教員が見ることできる。これによって，学生のソースコードを，教員が直接見て問題点を指摘することができて，よりスムーズに問題解決を図ることができるようになる。

なお，遅刻した学生が途中からミーティングルームに入室したときには，ホスト側に通知がくる。そこで，まだ割り当てていないブレイクアウトルームに，その学生を入室させることができるようになっている。また，すべてのブレイクアウトルームを回った後に，再び学生が質問したい場合は，「ヘルプを求める」ボタンを押せばよい。すると，ホスト側に通知がくるので，教員は該当のブレイクアウトルームに再度入室して個別指導を行うこともできる。

(2) 2021年度

2021年度は依然としてコロナ禍ではあったが，対面授業とオンデマンド授業が併用して実施された。オンデマンド授業では，クラウド型動画コンテンツ管理ツールであるKaltura（Moodleとの連携機能あり）を用いてあらかじめ授業を録画し，それをMoodleで閲覧するという形をとった。

本科目については，実習室での対面授業に戻った。本来ならば，対面による個別指導（教卓に学生を来させて教員と１対１の対話）を行うことになるが，Zoomが実習室のすべてのパソコンにインストールされていたこともあり，2020年度と同様のZoomによる授業を進めた。授業アンケートによると，学生からも高評価であった。

(3) 2022年度

2022年度からは，1コマ目の開始時刻は午前10時から9時に戻り，授業時間は90分間から100分間に変わり半期14週となった。授業では，テキストエディタをTeraPadではなく，Visual Studio Code（以降，VS Codeと略す）に変更した。

TeraPadは専用のテキストエディタであり，ルーラー・行番号・改行・タブ・EOFなどが表示できること，半角/全角スペースを区別して表示できること，HTML・CSS・JavaScript・PHPなどに対応していること，などの機能がある。この中の半角/全角スペース表示については非常に有効である。というのも，全角モードのままスペースキーを押してしまい構文エラーを起こすといったケースを，事前に防ぐことができるからである。

ただし，文字コードについては，問題があった。それは，デフォルトがシフトJISコードになっているため，「文字/改行コード指定保存(K)…」で「UTF-8」に変更しなければならないことである。学生の中には，このことを忘れてシフトJISのままプログラミングしてしまい文字化けを起こすというものであった。

VS Codeは，テキストエディタというよりも統合開発環境といえる。デフォルトの文字コードはUTF-8を採択しており，補完候補をあげてくれるインテリセンスやHTMLタグなどを少ない記述で補完入力できるEmmetといった機能が提供されている[21]。これらによって，TeraPadよりも効率よく，プログラミングとデバッグができるようになった。

3. 自作プログラミングから模写プログラミングへ

2で述べたように，科目「プログラミング基礎」では，2013年度から2022年度まで，自作プログラミングによる実習を進めてきた。自作プログラミングとは，プログラムの仕様を実習課題として与え，学生はそれを見てプログラミングとデバッグを行い，出来上がったプログラムをMoodleにアップロードするというものである。

一方，自作プログラミングに対して，模写プログラミングがある。ここでは，模写プログラミングに関する研究や2023年度に行う実証実験について述べる。

3.1 模写プログラミングとは

模写プログラミングは，もともと写経プログラミングという名称で呼ばれてきた。写経プログラミングとは，完成したプログラムのソースコードをそのまま写すことである。これによって，タイピングすることで記憶に定着できること，タイピングミスで生じる構文エラーをチェックし直すことで学ぶことができること，ソースコードに応じた実行結果を確認できること，プログラムが動いたことで達成感を味わうことができるなどの効果が期待できる。

写経プログラミングに関する研究として，喜多達のものがあげられる[22]。ここでは，反転授業に写経プログラミングを取り込むという試みについて述べている。反転授業とは，従来型授業における「講義（授業中）→復習（授業外）」を「予習（授業外）」→「演習/実習（授業中）」に反転させるというものである[23]。この「予習（授業外）」のところに写経プログラミングを取り入れるとともに，「演習/実習（授業中）」では教員の指導力を生かして課題の改変などを行うことにより学習内容の理解と定着をはかるというものである。さらには，写経型学習における初学者のつまずきに着目し，その学習方略についても報告している[24]。また，中田は，写経プログラミングにおけるタイピング速度と言語の理解度の関係について論じた[25]。

岡本は，写経プログラミングを学術レベルで取り上げる際に，さまざまな問題（写す過程の扱い，比較する学習法，実践上の比較など）に突き当たるとしている。写経プログラミングの達成感については，写すだけで満足してしまい何も学んでいないことを言及している[26]。

3.2　模写プログラミング導入の経緯

模写プログラミングを導入しようとしたきっかけは，オンライン授業から対面授業に戻ってからの個別指導で顕著化してきた問題である。

オンライン授業のときは，学生は自宅等からZoomのブレイクアウトルームに入室した。入室すると，他の学生と直接接触することや会話することはできない。このため，学生は実習課題を自分一人でプログラミングしてデバッグすることになる。一方，対面授業では，実習室に学生が集まりお互いの顔を見て話しをすることができるだけでなく，授業中には他の学生と教え合うことも容認していた。そして，上述したように，対面授業においてもZoomを用いており，ブレイクアウトルームにおいて教員と学生1対1の個別指導を行っていた。

その個別指導において，進捗の遅れている数名の学生に限って質問なしと回答することが明らかになってきた。当初は何も問題がないから質問もないのかと思ったが，そうではなくて，どうやってよいかわからないので，友人のプログラムをただコピペしているだけなので質問ができないとのことであった。これより，2013年度のコピペ問題が再度発覚したことになる。このときは，個別指導に切り替えることで回避したが，さらに別案を講じる必要になった。

そもそもコピペをさせないためには，最初からプログラムのソースコードをコピーすることを前提にすればよいことになる。これは，まさしく模写プログラミングとなることから，実習課題を自作プログラミングではなく模写プログラミングで行うことを考えついた。ただし，岡本の指摘にある何も学んでいないことにならないような策を講じる必要がある。

そこで，一つ目の策としてソースコードの説明追加，二つ目の策として確認テストの実施を試みることにした。

(1) ソースコードの説明

実習課題の提出に関して，今まではソースコードと実行結果だけのアップロードとしていたが，ソースコードの該当箇所に説明文を，コメント（1行の場合は//，複数行の場合は/*…*/）として新たに追加することにした。説明文については，

〇変数（宣言：var）について
- ・何に使うのか
- ・初期値を与えた理由

〇構文（代入文：=，演算式：+-*/%，選択文：if文 if else文 else if文 switch文，繰返し文：for文 while文 do while文，関数：function文 呼び出し文）について
- ・各文の文法
- ・各文の書き方に関する規則
- ・各文の働き
- ・各文を実行するとどうなるか

〇ロジック（論理的な手順）について
- ・代入文において，値を代入した理由
- ・選択文について，プログラムの仕様のどこに反映されるのか
- ・なぜ，ここに選択文を入れているのか

・選択文はどういう意味をもっているのか
・繰返しによって何をしようとしているのか
・繰返しによって，何が変化するのか
・各ライブラリ関数（eval，Math.floor，…）はどういう働きをするのか
・オリジナル関数（自分で宣言する関数）について，どういうまとまった処理を行うのか
・オリジナル関数における引数と戻り値はどういう値になるのか

○アルゴリズム（プログラムの全体的な処理手順）について
・各整列アルゴリズムによって，どういう手順で配列の中身を並び替えているのか
・各探索アルゴリズムによって，どういう手順で配列の中身を探索しているのか
・再帰アルゴリズムによって，どのような処理が繰り返されていのか

をできるだけ詳しく記述することとした。

(2) 確認テスト

模写プログラミングの学習効果を測るためには，自作プログラミングとの比較が必要になる。これについては，クラス全体の進捗状況だけでなく，新たに確認テストを実施することにした。

プログラミング教育における理解度テストとしては，いくつかの報告がある。田中達が取り上げた小テストでは，不完全なプログラムを修正して正しい結果を導き出す問題，実行結果を見せてプログラミングさせる問題，あるプログラムを実行した結果を表示する問題などを取り上げて

表1　確認テストの概要

問題番号	問題数	素点	Moodleの問題種別	概要
問題1	6問	各1点	○/×問題	JavaScriptの基本的な書き方（フォーマット，コメント，テキストエディタ）に関する正誤問題
問題2	6問	各1点	○/×問題	変数の宣言（命名規約，データ型，初期値）に関する正誤問題
問題3	3問	各2点	作文問題	JavaScriptのソースコードにあるエラー（スペルミス，ダブルクォーテーションの扱い，ロジックミス）の検出とその理由を記述する問題
問題4	1問	5点	作文問題	document.write文の構文（文字列・変数の表示，改行）に関する問題
問題5	1問	5点	作文問題	あるJavaScriptのプログラム（値の入れ替え）を実行した結果の表示に関する問題
問題6	1問	5点	作文問題	prompt文の構文（ガイド文，省略時の初期値）に関する問題
問題7	1問	5点	作文問題	i=i+1;についての説明と別表記（代入演算子，インクリメント）に関する問題
問題8	1問	5点	作文問題	あるJavaScriptのプログラムで実行時エラー（計算の右式に値を設定していない変数）の検出とその理由を記述する問題
問題9	1問	10点	作文問題	あるJavaScriptのプログラムで実行時エラー（eval関数の使用）の検出とその理由を記述する問題
問題10	1問	10点	作文問題	if文のネストを用いて値の比較をするプログラムの作成問題
問題11	1問	10点	作文問題	問題10をswitch文を用いて書き換える問題
問題12	1問	12点	作文問題	配列を用いた昇順ソートをするプログラムの作成問題
問題13	1問	15点	作文問題	オリジナル関数（引数・戻り値あり）を用いたアルファベットの大文字小文字を判定するプログラムの作成問題

いる[27]。勝間田達は，プログラミングの理解度を把握するための指針として，入出力（変数），演算（式），順次・分岐・反復構造，配列，二重ループ，基本アルゴリズム（ソート，探索）といった項目を取り上げ，それぞれの問題をMoodleの小テスト機能を用いて出題している[28]。

これらを参考にした上で，Moodleにおいて独自の確認テストを[小テスト]として作成することにした。具体的には，**表1**のような内容とした。

3.3　実証実験について

自作プログラミングと模写プログラミングの学習効果の比較をするための実証実験を，2023年度に試みることにした。科目「プログラミング基礎」は，春学期と秋学期の両方で開講しているので，これを利用することにした。なお，履修者は事前登録による抽選となり，実習室の収容人数の関係から最大30人までとした。

（1）春学期

授業準備として，各種の設定（Zoomのミーティングルーム，Moodleの個人別フォーラム，Moodleの確認テスト）を行った。Moodleの教材コンテンツ・ヒントと実行結果・実習課題・事前/中間/事後アンケートについては，前年度のものをバックアップ/リストアした。履修者は27人となり，授業形態は自作プログラミングによる実習とした。

学生は，実習室に来て，Zoomのミーティングルームに入室してから割り当てられたブレイクアウトルームに入る。個別指導になるまで，Moodleにある教材を自学自習しながら，実習課題のプログラミングとデバッグに取り組む。個別指導では，わからないことを教員に質問しアドバイスを得て実習を進めることになる。そして，授業の最終回（2023年7月6日）に，確認テストを実施した。テストはMoodleの[小テスト]で行い，60分間で遮断するように設定した（**図6**）。

図６　確認テストの設定

図 7　実習課題毎に，png ファイルを追加

(2) 秋学期

秋学期は模写プログラミングによる授業を行う予定である。そのために，教材コンテンツの一部変更を行った。プログラムを模写するために，64 問すべての実習課題のソースコードを png ファイルとして新たに作りこみ，Moodle にアップロードした（図 7）。

png ファイルにすることによって，学生はソースコードをそのままコピペすることができないので，キー入力することで模写をすることになる。それとともに，3.2 の（1）で取り上げたように，ソースコードに説明文をコメントとして入力することで，学生は JavaScript の構文やプログラムのロジックあるいはアルゴリズムについて理解を深めることができるはずである。そして，授業の最終回には，春学期と同じ確認テストを実施する予定である。

お わ り に

以上，科目「プログラミング基礎」におけるプログラミング教育の変遷と自作プログラミングから模写プログラミングへ移行する経緯について述べてきた。実証実験では，プログラミングスキルを効率よく習得するためには，自作プログラミングの方がよいのか，あるいは，模写プログラミングがいいのか，比較することを目指している。比較にあたっては，クラス全体の進捗状況

だけでなく，確認テストによる出来具合を考慮したい。秋学期からは，春学期と同じ形（Zoomによる個別指導中心）での授業を行うが，模写プログラミングによる実習に切り替える予定である。

今後の課題としては，秋学期終了後に，Moodleの各種学習データ（進捗データ，小テストの結果，アンケートデータ）を取得して，学習効果を比較することを予定している。

注

1）ただし，2019年4月15日にサービス終了.
2）Moodleのモジュール名には，鍵括弧[ ]をつけている.

参考文献

[1] Allen B.Tucker, Bruce H.Barnes, Robert M.Aiken, Keith Barker, Kim B.Bruce, J.Thomas Cain, Susan E.Conry, Gerald L.Engel, Richard G.Epstein, Doris K.Lidtke, Michael C.Mulder, Jean B.Rogers, Eugene H.Spafford, A.Joe Turner : Computing Curricula 1991 – Report of the ACM/IEEE-CS Joint Curriculum Task Force, acm PRESS, 1991

[2] 河村一樹：情報ビジネス学科におけるプログラミング教育，東京国際大学論叢商学部編第84号，pp. 23–39，2011年.

[3] 情報処理教育委員会J07プロジェクト連絡委員会：情報専門学科におけるカリキュラム標準J07（中間報告），情報処理学会，2007年.

[4] 情報処理教育委員会J07プロジェクト連絡委員会：情報専門学科におけるカリキュラム標準J07，情報処理学会，2009年.

[5] 情報処理学会カリキュラム標準一般情報処理教育（GE）
https://www.ipsj.or.jp/annai/committee/education/j07/ed_j17-GE.html（※2023年8月現在）

[6] 情報処理学会一般情報教育委員会編：一般情報教育，オーム社，2020年.

[7] 森畑明昌：東京大学におけるプログラミング教育，情報処理 Vol. 57 No. 4，pp.362–365，2016年.

[8] 多田知正，丸田寛之：プログラミング教育における反復学習を取り入れた授業方式，京都教育大学紀要，No. 116，pp. 123–134，2010年.

[9] 河村一樹：一般情報教育におけるプログラミング教育のあり方について，情報処理学会コンピュータと教育研究報告，2011-CE-108（16），2011年.

[10] 河村一樹：JavaScriptによる情報教育入門，大学教育出版，2011年.

[11] 河村一樹：講義レスによるプログラミング実習教育の試み，情報処理学会コンピュータと教育研究報告，2015-CE-128（21），2015年.

[12] サイボウズ社：サイボウズLive
https://cybozu.co.jp/products/old-products/cybozulive/（※2023年8月現在）

[13] 河村一樹：グループウェアを用いたプログラミング実習教育，情報処理学会コンピュータと教育研究報告，2016-CE-134（26），2016年.

[14] 河村一樹：開講コマの違いによる学習進捗の相違について――自学自習ベースのプログラミング教育の場合――，情報処理学会コンピュータと教育研究報告，2017-CE-139（8），2017年.

[15] 河村一樹：Moodleを用いた自学自習ベースのプログラミング教育，東京国際大学論叢 人間科学・複合領域研究，第6号，2021年.

[16] 河村一樹：Moodleを用いたプログラミング教育の事例，TIU学内報FD Newsletter“SEED”，2018年.

[17] 河村一樹；Moodleを用いたプログラミング教育における評価法――「正確性」に「迅速性」を加味した結果――，e-Learning教育研究，第14巻，pp. 34–42，2020年.

[18] 福村裕史・河村一樹・後藤顕一編：すぐにできる！双方向オンライン授業【試験・評価編】インターネットを活用した学習評価，化学同人，2020年.

[19] 河村一樹：対面授業からオンライン授業へ：プログラミング教育事例，ソフトウェア技術者協会新春教育フォーラム2022，2020年.

[20] 河村一樹：科目「プログラミング基礎」における対面授業とオンライン授業の比較，情報処理学会教育とコンピュータ，Vol. 8，No. 1，pp. 100–107，2022年.
[21] SKILLHUB：VS Codeの使い方入門！～基本からWeb制作まで
https://skillhub.jp/blogs/235#skillhub-chapter-5（※2023年8月現在）
[22] 喜多　一，岡本雅子：写経型プログラミング学習と反転授業，第60回システム制御情報学会研究発表講演論文集2016，2016年.
[23] 河村一樹，今井康博：大学における反転授業，大学教育出版，2017年.
[24] 岡本雅子，喜多　一：プログラミングの「写経型学習」における初学者のつまずきの類型化とその考察，滋賀大学教育学部附属教育実践総合センター紀要，第22巻，pp. 49–53，2014年.
[25] 中田豊久：写経プログラミングの学習効果に関する考察，第27回人工知能学会全国大会論文集，2013年.
[26] 岡本雅子：ぺた語義 Vol. 77 写経プログラミングをめぐる終わりそうもない論争，情報処理，Vol. 59，No. 1，p. 81，2018年.
[27] 田中善雄，三宅芳雄：プログラミング教育における小テストの実践報告，情報処理学会情報教育シンポジウム，2008年.
[28] 勝間田仁，加藤利康，中村一博：プログラミング教育必修化世代を対象としたプログラミング理解度判定テストの構築，情報処理学会第82回全国大会論文集，2020年.