一般情報教育におけるプログラミング入門教育に関する研究

内田　奈津子

電気通信大学大学院情報理工学研究科

博士 (工学) の学位申請論文

2021 年 9 月

一般情報教育におけるプログラミング入門教育に関する研究

博士論文審査委員会

主査　中山 泰一　　教授
委員　岩崎 英哉　　教授
委員　小林 聡　　教授
委員　寺田 実　　准教授
委員　小花 貞夫　名誉教授

A Study on Introductory Programming in General Information Education

Natsuko Uchida

Abstract

This dissertation is a study of introductory programming within the context of general information education as programming education that everyone should learn in higher education.

Programming education in elementary schools began in Japan in April 2020. With the current revision to the Courses of Study, all middle school and high school pupils will be required to learn programming from 2022. However, there are no required IT courses in higher education, and not all college students have the opportunity to learn programming.

Looking ahead to the spread of programming education in elementary schools, universities will need to change how introductory programming is taught in their general information education courses. In addition, since it will be another seven to eight years before the first cohort to undergo programming education in elementary school enters tertiary education, universities now must face the urgent task of developing opportunities for all students to learn programming effectively.

Until now, programming education has focused on teaching individual learners how to master discrete sets of knowledge. Consequently, software development curriculums at elementary, secondary, and tertiary levels have all targeted individual learners. However, when one considers both the continuity of primary, secondary, and tertiary education, and the evolving connectivity between universities, society, and industry, the ability to learn basic programming and develop software in teams is increasingly required of higher education graduates. It is clear that the exigencies of today's information society require the adoption of a new educational methodology in which students can learn together in teams or groups.

To meet the needs of contemporary society, this dissertation proposes a team-based higher-education curriculum for introductory programming utilizing PBL (Project-based Learning) for software development projects. In addition, since beginner-level students usually find it challenging to develop full-fledged software in software development projects, this dissertation proposes teaching material that simulates developing software by integrating objects made up of picture parts.

The research presented in this dissertation demonstrates that students can gain a sound understanding of programming principles and software-development project methodology, and moreover, do so in a short timeframe such as a two-credit university course. Furthermore, this dissertation highlights the effectiveness of visualization-based pedagogy in Introductory Programming Education and PBL.

一般情報教育におけるプログラミング入門教育に関する研究

内田　奈津子

和文要旨

情報機器の発展とインターネットの急速な普及による情報革命により，これまで専門家を中心に，限られた人々のみで使われていた情報通信技術は，一般の人々の暮らしの中に浸透してきた．そして，情報通信技術が積極的に使われるようになった情報化社会においては，人々の生活をより良くするための様々な課題解決に情報通信技術が使われるようになった．しかし，私たちの身近な課題の解決には，IT のスペシャリスト集団のみで対応できるわけではなく，多様な人々が協力して関わって解決する必要がある．なぜなら，IT スペシャリストは，社会のことについては精通しているとは限らず，他方で，一般市民は，情報通信技術に精通しているとは限らないことから，両者が変わることが重要であるからである．そのため，これまで求められていた専門分野に特化した IT 人材の枠にとどまらず，誰もが基本的な IT スキルを身につけ，社会全体のスキルの底上げをはかる必要がある．

2013 年の世界最先端 IT 国家創造宣言により，小学校へのプログラミング教育の導入が決定し，2020 年 4 月より，小学校でのプログラミング教育が始まった．学習指導要領の改定により 2022 年度までに，中学校，高等学校においても，継続して全員がプログラミングを学ぶ機会が整う．しかし，高等教育においては，必須化された科目がなく，誰もがプログラミングを学ぶ環境にはなっていない．そのため，小学校におけるプログラミング教育が普及した先を見据え，大学一般情報教育におけるプログラミング教育のあり方も変化すべきである．また，現状，小学校におけるプログラミング教育を受けた生徒が大学に入学するまでには, まだ 7,8 年の時間がかかることから，高等教育においても，誰もが効果的にプログラミングを学ぶ機会を設けることは急務である．

高等教育においては，初等中等教育からの連続性と社会への接続に配慮し，プログラミングの基礎知識を学ぶだけでなく，学んだ知識を活用することを含めて学ぶ必要があると考える．これに対応するための 1 つの方法として，初等中等教育では取り上げられていない，チームでソフトウェア開発プロジェクトを実践する内容を含む教育を提案する．

従来のプログラミング入門では，ソフトウェア開発のことをカバーした例はなく，またソフトウェア開発に焦点をあてている科目は，プログラミング入門を済ませていることが前提で，1 科目の中に両方を盛り込むのは無理であるとあきらめ

られていた．

本論文では，高等教育において誰もが学ぶべき一般情報教育におけるプログラミング入門を取り上げ，1 科目でプログラミングの基礎知識を学ぶだけでなく，チームでソフトウェア開発プロジェクトを実践する内容を含む入門レベルのカリキュラムとして，PBL(Project-based Learning) によるカリキュラムを提案する．描画教材を用いることにより，プログラミングの原理の理解とソフトウェア開発プロジェクトの知識の獲得を，90 分 15 回 2 単位で実現できることを示し，このカリキュラムによって，プログラミングの概念の理解と社会で求められる資質・能力の育成が図れることを示す．さらに，カリキュラムで使用する描画教材とプロジェクトの効果について示す．

第 1 章では，情報化社会における時代の変化，教育の変化から，これからの高等教育におけるプログラミング教育の取り組み方について課題提起を行ない，本研究の目的について述べる．

第 2 章では，提案するカリキュラムを構築するために，高等教育におけるプログラミング教育とアクティブラーニング・PBL に関する関連研究・先行研究を取り上げる．

第 3 章では，プログラミング入門に PBL を加えて学ぶ方式を提案し，高等教育における一般情報教育のカリキュラムを設計する．PBL によるプロジェクトには，チームで取り組むソフトウェア開発プロジェクトを取り上げる．対象は，全学生 (情報系を専門としない学生が中心) であり，本格的なソフトウェアを開発することは困難であることから，1 つの絵をソフトウェアに模擬した描画教材を提案する．

第 4 章では，設計したカリキュラムを実装し，実践授業を行い， 次の評価と検証を行う．

1. 90 分 15 回 2 単位のカリキュラムの妥当性について，到達目標の観点から実装したカリキュラムの評価
2. 社会で求められる資質・能力の育成について，コンピテンシー評価の分析による検証

第 5 章では，第 4 章で行なった実践授業での提出物を分析し，プログラミングの概念の理解と活用について検証する．

第 6 章では，第 4 章および第 5 章の実践授業の結果から明らかになった問題の見直しを図り，カリキュラムと教材の修正を行う．修正に基づき実践授業を実施し，履修者の提出した課題から，プログラミングの概念の理解と活用について再度検証する．さらに，描画教材とプロジェクトによる効果を検証する．

第 7 章では，社会への接続の観点から追跡調査を行う．実践授業の履修者を対象に，履修後の活動への影響をアンケートにより調査する．

最後に，第 8 章で，本研究の成果をまとめ，初等中等教育における学習指導要領改定によるプログラミング教育が普及し定着した後の展望について述べる．

目 次

図 目 次

表 目 次

第1章

研究の背景と目的

1.1　はじめに

科学技術の進歩が急速に進み，AIや機械学習の向上により，これまでできなかったことができるようになるなど，その発展は収束することを知らず日々加速している．これまで，専門家が利用してきたコンピュータをはじめとする情報機器やインターネットは，一般家庭にも普及し，誰もが利用する身近なものとなった．そして，情報機器の普及は，人々の暮らしを豊かにし，生活をしていく上で不可欠なものとなった．本論文では，このような社会を，情報化社会と呼ぶ．

これからの情報化社会においては，より良い社会を構築していくために，それぞれの知識を集め，互いに協力して活動していくことが重要となる．特に，プログラミングの知識は，コンピュータがどのような仕組みで動いているのかを理解し，社会でどのように機能しているのかを理解し活用していく上で重要であると考える．そのため，情報関連の基礎知識を持つことは，これからの社会でよりよく生活していくために誰にとっても必須のものとなってきている．加えて，情報化社会においては，急速な変化に対応する力や社会課題解決に対応する力が求められており，このような社会に対応する力を育成するためにも，継続して教育していくことが重要であると考える．そのため，専門によらず誰もがわかりやすく学びやすい情報教育のための環境が必要である．

プログラミング教育は，世界各国で幼稚園から高等学校 (K-12) での取り組みに関する話題で盛り上がりを見せてきている [13][17][52][116]．わが国でも，2020 年度から小学校におけるプログラミング教育の必須化が始まり，さらに，就学前であってもプログラミングを学ばせるべきであるという考えもあり，未就学児を対象とした研究も行われている [32][131][132]．社会への接続を視野に入れると，高等教育においても，初等中等教育と接続し，連続して学べる環境が必要であると考える．

しかし，わが国では，高等教育においては，これまで，調査研究 [122] が行われ

たり，カリキュラムモデル [74] が示されたりしているが，プログラミングは，誰もが学ぶ必須化されたカリキュラムにはなっておらず，継続して学べる環境は整っていない．これからの社会での活動を考えると，高等教育においても，誰もが継続してプログラミングを学べる環境を整え，初等中等教育から途切れることなく社会へ接続できるような環境を作るべきだと考える．

現状，小学校におけるプログラミング教育を受けた生徒が大学に入学するまでには，まだ 7,8 年の時間がかかることから，高等教育においても，誰もが効果的にプログラミングを学ぶ機会を設けることは急務であると考える．また，小学校プログラミングが普及した先を見据え，大学一般教育におけるプログラミング教育のあり方も変化が必要であり，一過性の対応でなく，小学校におけるプログラミング教育を受けた児童が大学に入学した際にも有益なものであることが望まれる．

Soloway らは，コンピュータと人間との関わりを，1980 年以前は，技術の進歩発展による恩恵を一方的に享受していた技術中心主義の時代，その後，ユーザがその技術を使いこなすようになったことからユーザ中心主義の時代とし，1990 年代を学習者中心主義と区分している [29]．そして，学習者中心主義においては,「タスクの実行」をサポートするインターフェースを構築することに加えて,「タスクの実行中の学習」をサポートするインターフェースが必要だと述べている． 生徒は，やる気を起こさせる本物のタスクに従事し，タスク，アクティビティ，およびツールが「適切に足場を組んでいる」ときに最もよく学び，足場は通常，教師，メンター，親，および/またはカリキュラムによって提供されるが，インターフェースも学習者の足場にする必要があると述べている．

佐伯は，Soloway らの言葉を用い,「学習者中心社会においては，学習の概念自体も個々の学習者が習得するという発想から脱皮し，市民が社会実践活動の中で互いに学び合うという側面を重視した概念に変わる必要がある」[69] と述べている．

このように，技術革新とともにコンピュータと人間の関係は変化している中で，これからは，教育においても学習者が相互に学び合い関係していく教育へと変改していかなければならない．

著者は，25 年に渡り，文系女子大学で情報教育のサポートに従事する中で，大学での学びだけでなく，卒業後の社会での活動をよりよくするために，さまざまな情報教育を伴ったプロジェクトを行なってきた．専門の専攻でないから学ぶ必要がないということではなく，専門でないからこそ，これからの社会において必要となる基礎知識を身につける必要があると考え取り組んできた．また，情報化の加速に伴い，より早い段階からの教育環境の構築が重要であると考え，小学生を対象とした地域活動 [30][45] で，ロボット教材を用いたプログラミング教育の実践を行なってきた．この活動は，個々の学習が中心ではなく，グループで取り組むものであった．当時は，先行研究等から得た知識で取り組んだわけではなく，人と人との関係の希薄さから，子どもたちのコミュニケーションや友だちとの関わりを中心に考えて，自然とグループでの取り組みとなった．この経験から，著者は，グループでのプロジェクトによる教育に注目するようになった．

このような経緯から，本論文では，高等教育における誰もが学ぶべきプログラミング教育として，一般情報教育を取り上げ，プロジェクトによるプログラミング入門を提案する．高等教育においては，専門教育に限らず，一般教養科目があり，さまざななレベルでの教育が行われているが，本論文では，専門教育は除外し，一般教養に該当する情報教育を，一般情報教育とする．

これまでの知識伝達型の教育から，他者が伴って学ぶ教育への変化が求められる社会となっていること, そして著者の経験から，プログラミングの基礎知識を学ぶだけでなく，チームでソフトウェア開発プロジェクトを実践する内容を含む教育が必要であると考え，PBL(Project-based Learning) によるカリキュラムを提案する．本研究では，ソフトウェア開発プロジェクトを入門教育に取り入れ，1 枚の絵をソフトウェアと模擬して,「お絵かき」を「チームで解決する」という具体的な教材で実現させる．

以下では，本研究の背景について述べた後，1.6 節において，本研究の目的を整理して示す．

1.2 社会的な背景

情報機器の発展とインターネットの急速な普及による情報革命は，工場での大量生産に象徴されるようなモノに価値のある時代から，知識や情報に価値がおかれる社会へと変化させた．そして，これまでは限られた人々のみで使われていたコンピュータは，人々の暮らしの中にも入ってきた．そのため，情報通信分野の知識は，専門家だけが知っていれば良い知識から，基本的な原理原則は，万人に必要な知識へと広がってきた．

そして，この発展は，人々の生活をより良い方向に変化させるデジタルトランスフォーメーション (Digital Transformation，DX) の概念 [75] を生んだ．この概念により，これからの時代は，AI や IoT などを導入し，単なるもの作りにとどまらず，あらゆるものを繋げることによる新たな価値の創造へと進化している．このような情報通信技術の一般市民への広がりは，様々な分野に寄与しており，企業のコスト削減や業務の効率化，人口減少社会における労働力不足の解決と持続的成長のためなどがある．そのため，このような社会においては，“変革の担い手”が必要であり，前述したように，佐伯は,「学びとは，他者とともに知を分かち合い，学びの共同体づくりに参加していく学びであるべきであり，これからの情報教育は，情報化社会に「適応させる」教育ではなく,「変革の担い手」を作る教育であるべきである」[69] と述べている．

Society5.0[82] では，IoT（Internet of Things）で全ての人とモノがつながり，様々な知識や情報が共有され，今までにない新たな価値を生み出すことにより，課題や困難を克服するとしている．Socirty5.0 で実現する社会として以下が示された．

- IoT で全ての人とモノがつながり，様々な知識や情報が共有され，新たな価

値がうまれる社会

- 少子高齢化，地方の過疎化などの課題をイノベーションにより克服する社会
- AI により，多くの情報を分析するなどの面倒な作業から解放される社会
- ロボットや自動運転車などの支援により，人の可能性がひろがる社会

しかし，社会システムの構築には，IT の専門家集団のみでは対応できることが少なく，社会をよく知る IT の専門家でない集団とともに協力して進める必要がある．しかし，IT の専門家集団は，社会に十分精通しておらず，逆に，IT の専門家でない集団は，情報システムに精通していないことから，有効な議論をすることが難しく，結果，使えないシステムが作られてしまうこともある．そして，それが時には大きな問題を起こし，訴訟にまで発展することもある [33].

そのため，これまで求められていた専門分野に特化した IT 人材の枠にとどまらず，誰もが基本的な IT スキルを身につけることが重要である．社会全体のスキルの底上げをはかることにより，1 人 1 人が変革の担い手として活躍する社会になることを期待する．そして，それぞれの得意な分野を活かして関わることにより，よりよい社会となっていくであろう．

一方で，このような社会の進歩発展により，単純労働は，コンピュータに取って代わられ，人が従事する仕事は変化を余儀なくされ，雇用環境も大きく変化する [15] といわれる時代となった．雇用についての変化があるものの，IT 人材の需要は右肩上がりに伸びており，2019 年 3 月の経済産業省の調査 [61] によると，IT 人材の不足は，2030 年には，最大で約 79 万人に拡大する可能性があると試算されており，これは社会的課題の一つとなっている．

一般的に，IT 人材とは，システムエンジニアやプログラマー，IT コンサルタントなど，IT スキルと知識を必要とするシステム開発や IT 戦略支援などを行う人材のことを指す．しかし，このような人材育成に関わるわが国の情報専門学科の定員数は多くなく [117]，これら専門学科の卒業生だけで，これからの IT 人材の需要を満たすことは不可能であり，より広い範囲からの人材の育成が求められている．

より良い暮らしを実現するために，高度な IT 人材の育成の枠にとどまらず，社会全体のスキルの底上げを目指し，誰もが情報技術に関する基礎知識を学ぶ機会を設けるべきである．

60 歳を過ぎてからパソコンを始め，80 歳を過ぎてからシニア向けゲームアプリを開発した事例 [87] がある．この事例のように，簡単なものであれば，年齢に関係なく，プログラミングを少し勉強すれば，誰にでもゲームアプリを作れるような時代となった．学習者の意欲があれば，情報技術に関する基礎知識を学ぶ機会は年齢に関係なく，学ぶ方法も様々ある．

社会活動においては，プログラミングの知識を活かし，解く問題が与えられた課題解決の実践の場としてハッカソン [25] が開催されている．市民による課題解決「CivicTech」により，行政サービスの課題を解決していく事例 [51] もある．Code

for プロジェクト [11] は，世界各国 [12][66] で行われており，技術のレベルに依らず，多様な人材のチームによる社会課題解決に民間の力が使われている．

このように，すでにプログラミングの知識は，専門家だけに求められるものではなく，課題解決のために広く活用されるようになっており，個人の力だけでなく，チームとして取り組まれている．著者は，単にプログラミングを学ぶのではなく，社会の要請に応えられるような知識を持って学ぶことが重要であると考える．

情報技術の発展と市民への普及は，社会的な変化だけでなく，教育現場の教育方法についても変化をもたらせている．小林は,『変貌する高等教育』[67] で，大学で何を学ぶかについて述べ，大学という場を知を知識ではなく，知の行為という視点から考え直している．

著者は，単に知識を習得するのではなく，身につけた知識や技術を活用して課題を解決することを重点に置いた，教える教育から何をどのように学ぶのかを重視した教育に転換し，これからの社会で役立つような教育を行いたいと考えるようになった．

しかし，多くの入門教育では，知識伝達型のものが多い．特に，これまでのプログラミング教育では，初めに学ぶべき要素が多いこともあり，知識伝達型の講義と演習で構成されたものが多く，個々の能力をテストにより評価する点数や偏差値重視で行われてきてた．初めに学ぶべき要素が多いプログラミングの学びは，短い時間では自分の考えをプログラムにするという経験に結びつくことが困難であり，開発工程を経験的に学ぶことまで及ばない．

筆者が期待することは，誰もがプログラミングの概念を学び，入り口としての基礎を作ることであり，テストの点数で高得点を取ることではなく，学んだ知識を応用して活用する力をつけさせることである．スペシャリストを養成しても，使う側のスキルが足りないために開発途中で問題が発生することがある．これから，ますます発展するであろう情報化社会においては，裾野を広げた教育をすることにより使う側のスキルアップを図ることにも注力していくべきである．Code for プロジェクトでは，正に個々の得意不得意を活かし，市民の力により社会の課題解決に取り組んでいる．80 歳のプログラマーの事例では，少し学べば誰でも簡単なアプリケーションを作ることができる時代となったことを示している．

プログラミング入門での学びは，プログラム言語を学ぶことを目的にするのではなく，プログラミングの概念を学び,「深く考えること」,「表現すること」を経験することが重要であると考える．そして，これからの社会で活躍するための視点としてチームでの活動を意識し，協働作業を伴ったチームでのプロジェクトを取り入れることを提案したい．チームでのプロジェクトは，単なるものづくりではなく，プログラミングを活用する場面を想定し，ソフトウェア開発プロジェクトとして位置づけた PBL を提案する．

1.3　プログラミング教育の背景

2013 年，世界最先端 IT 国家創造宣言 [70] が閣議決定され，プログラミング教育の必要性が示され，小学校へのプログラミング教育の導入が決まった．また，日本学術会議において，2016 年に「情報学の参照基準」[89] が，2020 年に「情報教育課程の設計指針―初等教育から高等教育まで」[88] が取りまとめられ，公表された [91].

初等中等教育における情報教育は，1996 年に開催された臨時教育審議会で「情報活用能力」の育成が提言され，その後の調査を経て，情報教育の目的として，以下の 3 点が示された．

1. 情報活用の実践力
2. 情報の科学的な理解
3. 情報社会に参画する態度

初等教育においては，子どもたちの情報活用能力の育成を目的とし，特に，教科は設置せず，各教科指導での ICT 活用により，それらの科目の目標を達成する際に，効果的に情報機器を活用することとして進められてきた．

2016 年 4 月の第 26 回産業競争力会議 [73] において，IT や人工知能による「第 4 次産業革命」の推進に向け，初等中等教育段階でのプログラミング教育を必修化する方向性が示された．その後,「日本再興戦略 2016」[72] を経て，2016 年 6 月 2 日の閣議決定により，小学校におけるプログラミング教育 [123] の 2020 年度からの実施が決定された．

小学校におけるプログラミング教育では，育む資質・能力として，以下が示されている [108].

> 各教科等で育む資質・能力と同様に，資質・能力の「三つの柱」（「知識及び技能」,「思考力，判断力，表現力等」,「学びに向かう力，人間性等」）に沿って，次のように整理し，発達の段階に即して育成する．
>
> **【知識及び技能】** 身近な生活でコンピュータが活用されていることや，問題の解決には必要な手順があることに気づくこと．
>
> **【思考力，判断力，表現力等】** 発達段階に即して,「プログラミング的思考」を育成すること．
>
> **【学びに向かう力，人間性等】** 発達段階に即して，コンピュータの働きを，より良い人生や社会づくりに生かそうとする態度を涵養すること．

特定の科目を設定することなく，各科目においてプログラミング教育を実施する方針とし，算数・理科・総合などの各教科においてプログラミング的思考を育

んで行くこととなった [123]．子どもたち個々の知識の習得を目指したプログラミングの体験をすることにあり，この体験をトリガーとして，各教科でプログラミングの基本である順次処理，ループ，分岐を見いだすことにより，体験にとどまらず，能力の伸長と各教科の深い理解を期待する．

中学校の学習指導要領では，小学校段階で身に付けた基本的な操作に関する知識を深め，技能を高めたり，情報機器やソフトウェアの活用の幅を広げたりできるようにすることにより，情報活用能の実践力を身につけることが掲げられている [121]．そして，これに加えて，2002 年の学習指導要領の改訂 [120] により，既存の技術・家庭科にプログラミングが追加され,「プログラムによる計測・制御」の一単元として取り組むこととなった．

高等学校の学習指導要領は，1998 年（平成 10 年）7 月 29 日の教育課程審議会の答申を受け，1999 年に改訂された [112][113]．この改訂により，2003 年に，2 単位の必履修とする教科「情報」が，義務付けられるようになり,「情報 A」,「情報 B」,「情報 C」のどれか 1 科目以上を必須教科として設置した.「情報 A」では，実践力を,「情報 B」では，科学的理解を,「情報 C」では，参画する態度を重点に置く内容となっている．プログラミングの内容は，情報 B に含まれており，各科目で取り扱う内容は，以下に示す．

情報 A： 問題解決の工夫，情報伝達の工夫，情報の検索と収集，情報の発信と共有に適した情報の表し方，情報の収集・発信における問題点，コンピュータによる情報の統合，情報の統合適な処理，情報機器の発達とその仕組み，情報化の進展が生活に及ぼす影響，情報社会への参画と情報技術の活用

情報 B： 問題解決における手順とコンピュータの活用，コンピュータによる情報処理の特徴，コンピュータにおける情報の表し方，コンピュータにおける情報の処理，情報の表し方と処理手順の工夫の必要性，モデル化とシミュレーション，情報の蓄積・管理とデータベースの活用，情報通信と計測・制御の技術，情報技術における人間への配慮，情報技術の進展が社会に及ぼす影響

情報 C： 情報のディジタル化の仕組み，情報機器の種類と特性，情報機器を活用した表現方法，情報通信ネットワークの仕組み，情報通信の効率的な方法，コミュニケーションにおける情報通信ネットワークの活用，情報の公開・保護と個人の責任，情報通信ネットワークを活用した情報の収集・発信，社会で利用されている情報システム，情報化が社会に及ぼす影響

その後，すべての生徒に履修させる科目として,「社会と情報」及び「情報の科学」の 2 科目に再編され，生徒の多様な能力・適性，興味・関心等に応じてどちらか 1 科目を選択的に履修することとなった．プログラミングの内容は,「情報の科学」に含まれている．

2022 年から実施される高等学校情報科の新教育課程では，情報の科学的な理解に重点を置き「情報 I」(2 単位) を必履修科目とした上で，発展的内容として情報

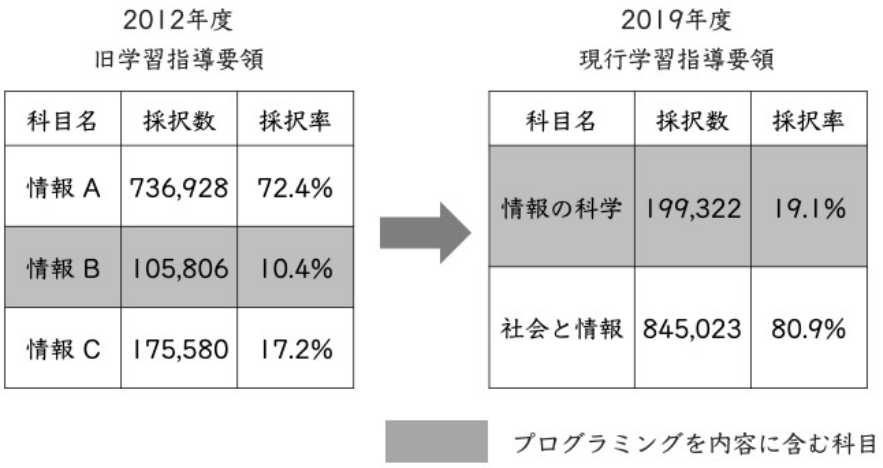


図 1.1: 高等学校における情報科目と教科書の採択率

システムなどを扱う「情報 II」(2 単位) を選択科目とすることになっている [133].「情報 I」の内容は,「情報社会の問題解決」,「コミュニケーションと情報デザイン」,「コンピュータとプログラミング」「情報通信ネットワークとデータの活用」の 4 項目から構成され，プログラミングは，全員が学ぶこととなる.

さらに，大学入学時点での情報の素養を問うために，2025 年の大学入学共通テストから,「情報」が新設されようとしている [84][90].

このように，情報の能力は，文系の専攻，理系の専攻に関わらず求められるようになり [16]，大学において学士力 [119] を身につけるための土台となるものとして必須のものとなってきた.

しかし，現在の状況を調べると，大学入学までにプログラミングを学ぶ機会がある学生は，教科書の採択率の数字から推測すると，2012 年度には約 10%[129], 2019 年度には約 20%[130] であり（図 1.1），少しは増えつつあるが，8 割以上の生徒は高等学校でプログラミングを学ばずに大学に進学していることになる.

高等教育においては，日本学術会議が，各分野での知識体系と各分野の学習を通じて獲得するべき能力を定めた参照基準を定めており，2016 年,「大学教育の分野別質保証のための教育課程編成上の参照基準情報学分野」[89] を策定した．この情報学の参照基準では，学部レベルで教えるべき情報学の知識体系を，文系・理系をまたいで 5 つの分類にまとめて定義した.

参照基準は，情報の専門課程を対象にしたものではあるが,「専門基礎教育および教養教育としての情報教育」の節が設けられており，大学教育に特化せず誰もが必要となる知識としての情報教育に関して，次のように書かれている.

情報学はメタサイエンスとして，すべての諸科学の基盤の一つと考えられる．また，情報学の中でも主として計算機科学に由来する，抽象化，モデル化，形式化，アルゴリズムの理解と設計，再帰的な思考，並列処理に関する理解，計算量の把握などを含む思考様式・スキル・技術は，「計算論的思考」（Computational Thinking）と呼ばれ，新たなジェネリックスキルとして，3 R（Reading，wRiting，aRithemtic ー日本語の読み書きそろばん）に加えるべき重要な能力であるとする考えが世界的に広まりつつある．

したがって，情報学は，情報学を専門に学ぶものに限らず，広く市民が持つべき教養の一部ともなっている．それゆえ，初等中等教育においては，教科によらず情報教育の機会が設ける方針がとられ，加えて中学校においては技術科の中で，高等学校では普通教科情報科の必履修科目の中でまとまった形で情報学に関して教育が行われているし，大学においては，「大学一般情報教育」という名の下で，教養教育（共通教育）の一分野として情報学が教えられている．

高等教育における情報教育は，情報学を学ぶことであるとすると，この参照基準の情報学の定義を借り，「情報によって世界に意味と秩序をもたらすとともに，社会的価値を創造することを目的とし，情報の生成，探索，表現，蓄積，管理，認識，分析，変換，伝達に関わる原理と技術を探求する学問」を学ぶことと言うことができるであろう．そして，このように，大学を超え，そして専門や一般教養をまたいだものは，情報学の性質であり，今後，学校教育にとどまらず，大学，そして社会の一員としての市民としても，プログラミングに関するスキルは，持つべき教養としてますます求められるものとなっていくであろう．

しかし，現状はまだまだ誰もが学ぶ環境とはなっていない．

2016 年 11 月から 12 月にかけて，情報処理学会は，文部科学省からの委託により，「超スマート社会における情報教育の在り方に関する調査研究」[122] の一環として，国内の全大学における情報学分野の教育に関する調査を行なった．

この調査結果 [76] によると，各大学が最も注力している領域は「一般情報教育」だが，「アルゴリズムとプログラミング」は，「コンピュータリテラシー」，「情報倫理とセキュリティ」など他の項目と比較して “教えていない ” 学部学科の率が高い結果が出ている．そして，実施が難しい原因として，時間確保の問題，教員の不足などが上げらている．

高等教育におけるプログラミング教育は，初等中等教育のような学習指導要領は存在せず，カリキュラム編成は各学校の判断に任されている点が問題である．現状，小学校におけるプログラミング教育を受けた生徒が大学に入学するまでには，まだ 5 年以上の時間を要することから，高等教育においても，誰もが効果的にプログラミングを学べる機会を設けることは急務である．さらに，小学校におけるプログラミングが普及した先を見据えた，高等教育におけるプログラミング教育のあり方も変化が必要である．

時間確保の難しさや教員の不足の課題に対応し，誰もがプログラミングを学ぶ環境を作るためにも，一般情報教育でのカリキュラムの見直しが有効である．さらに，社会への接続を考えると，チームで取り組むソフトウェア開発については，初等中等教育には含まれていないことから，これを補う形でカリキュラムを検討する必要がある．情報化社会においては，身近なところに様々なソフトウェアが存在し，専門家集団のみでは対応が難しく，より良いソフトの開発は困難である．様々な人々の知識や知恵を出し合うことで，より良いものを作ることができるであろう．そのため，専門家の育成に限らず，誰もが基本的な知識を持ち，専門家と協働できるような資質・能力を育成する教育を行う必要がある．

1.4　時代の変化と求められる資質・能力

旧来の「読み・書き・そろばん」が表すような個人の能力から，情報活用能力やコミュニケーション力などを最大限発揮でき，それらを結集したチーム力が求められるようになった．

時代の要請に伴って，求められる資質・能力がある．本節では，3 つの目標 (社会人基礎力・学士力・21 世紀型スキル) を示して議論する．

我が国では，経済産業省が,「職場や地域社会で多様な人々と仕事をしていくために必要な基礎的な力」として社会人基礎力 [63] を示した．社会人基礎力は，図 1.2 に示すように,「前に踏み出す力」,「考え抜く力」,「チームで働く力」の 3 つの能力と 12 の能力要素から構成された．

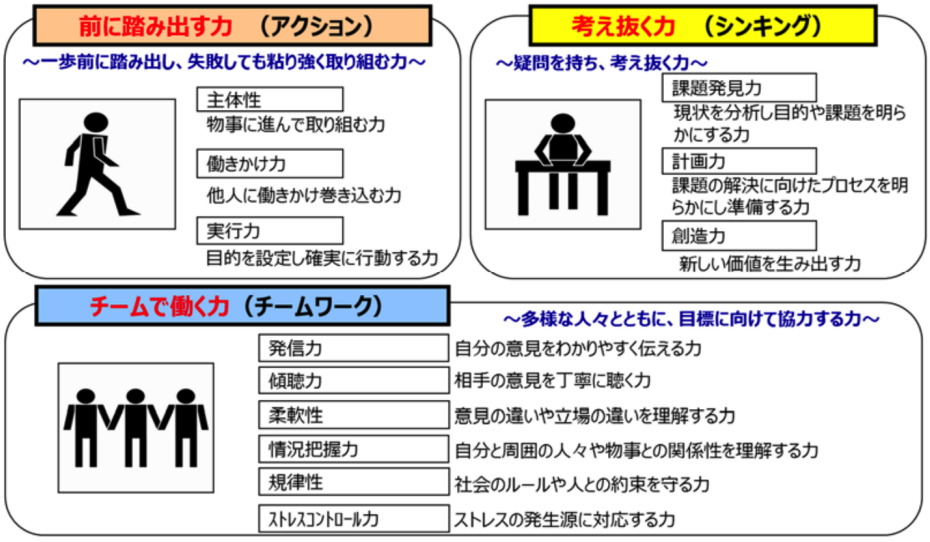


図 1.2: 社会人基礎力の 3 つの力

一方，文部科学省は，分野横断的にわが国の学士課程教育が共通して目指す「学習成果」に関する参考指針として，学士力 [110] を示した．

学士力は，教養を身に付けた市民として行動できる能力として,「知識・理解 (文化，社会，自然 等)」,「汎用的スキル (コミュニケーションスキル，数量的スキル，問題解決 等)」,「態度・志向性 (自己管理力，チームワーク，倫理観，社会的責任 等)」,「総合的な学習経験と創造的思考力」の 4 項目から構成された.

これは，高等教育を前提として考えられているが，汎用的スキルや態度・志向性は，高等教育によらず，21 世紀の社会に暮らす一般市民としてのスキルも含まれている．特に，自分の意見を相手に的確に伝え，立場や背景の異なるメンバーとともに行動する能力は，これまでの知識伝達型の教育スタイルから脱却し，アクティブラーニングによりチームで取り組むプロジェクトで育成することが望まれている [124].

このような資質・能力の指針は，日本に限らず国際的に広く知られているもののひとつに，ACT21S[4] によって提唱された 21 世紀型スキル [6] がある．

21 世紀型スキルは,「社会・経済の急激な変化に対応するには，従来の知識偏重型の教育では難しく，グローバル社会を生き抜くために必要な能力として，批判的思考力，問題解決能力，コミュニケーション力が問われる」という背景から，10 のスキルと 4 つの分類に整理してまとめられている．

1. 思考の方法（Ways of Thinking）
   (1) 創造力とイノベーション
   (2) 批判的思考，問題解決，意思決定
   (3) 学ぶことの学習，メタ認知（認知プロセスについての知識）
2. 働く方法（Ways of Working）
   (4) コミュニケーション
   (5) コラボレーション（チームワーク）
3. 仕事のツール（Tools for Working）
   (6) 情報リテラシー
   (7) ICT リテラシー
4. 世界の中で生きる方法（Skills for Living in the World）
   (8) 地域と国際社会での市民性
   (9) 人生とキャリア
   (10) 個人及び社会における責任（異文化の理解と異文化への適応力を含む）

現代の社会においては，教育も企業でのビジネスも情報通信技術の利用なしに円滑な遂行はできない．しかし，ICT のスキルは誰にでも不可欠なスキルであるが，単にスキルだけを持っているだけでは機能せず，他者とともに目的を達成したり，問題解決をしたりする際に有効に機能しなくてはならない．そのため，単なるスキルの育成を行うのではなく，活用する体験を持って身につけていくことが重要であると考える．

学士力・社会人基礎力・21 世紀型スキルの 3 つのどれにも，コミュニケーション力やチームワークが示され，アクティブラーニングや PBL による教育が求められるようになり，さまざまな分野で取り組みがなされている (2.3 節で取り上げる)．

しかし，著者の環境においては，アクティブラーニングや PBL の中で行われているコミュニケーションは，単なるおしゃべりであることも多く，プロジェクトに必要なコミュニケーションであるか，そうでないものなのか判断が難しいことがある．学習者もこれらを意識して取り組んだり，取り組んだ成果が目に見えて現れるような経験も乏しく，これらのスキルの効果的な習得の機会や教材を作ることも必要であると考える．

著者が提案する方法では，チームによるソフトウェア開発プロジェクトを行う中で，プログラミングの知識を学ぶだけでなく，それを利用して，自分で考え，チームで課題に取り組むことにより，目的を持ったコミュニケーションを図ることにつながり，議論するスキルを身につけ，真のコミュニケーションの力をつけることにつながると考える．さらに，チームで課題に取り組むことからチームワークも育成されることが期待できる．

さらに，情報通信技術の発展は非常に早く，その都度，必要に応じて新しいことを学んでいかなくてはならない．このような状況に対応するための，学習意欲や取り組みの姿勢なども重要な要素となる．

小林は，『変貌する高等教育』[67] で，知には 2 通りの様態（知識としての知 (knowing waht) と行為としての知 (knowing how)) があり，どちらが欠けても知の十全な運用はできないと述べており，著者は，情報技術における知は，知識とその活用によって機能する視点が重要であると考える．

プログラミングによるプロジェクト学習は，規模の大小によらず，ゴールを明確にすることができ，タスクオリエンティッドなコミュニケーション環境が構築しやすくなると期待できる．これにより，プログラムの知識のみならず，社会人基礎力，学士力，21 世紀型スキルなどに挙げられた資質・能力の育成につながると考える．

前述した，「情報教育課程の設計指針―初等教育から高等教育まで」[88] においても，学習内容について，情報学固有の知識・理解とジェネリックスキルを区別せず，近いものをグループ化して，11 カテゴリに整理している (図 1.3)．

本研究では，図 1.3 の A，C〜E，G，I〜H を中心に検討し，プログラミングの基礎知識の理解に加えて，「プログラムの知識を活用すること」を学ぶ視点で，プロジェクトによるプログラミング入門を提案する．これにより，プログラムの知識のみならず，社会人基礎力，学士力，21 世紀型スキルなどに挙げられた資質・能力の育成を図り，入門教育におけるプロジェクトの活用についての有効性を探る．

表１　情報教育における分野の分類（用語は参考資料５を参照）

領域	カテゴリとその記号	情報学固有の知識	ジェネリックスキル	専門的能力
情報とコンピュータの仕組み	A．情報およびコンピュータの原理	情報一般，機械情報，情報処理，人間社会，システム	論理，問題解決	倫理社会，システム
プログラミング	C．モデル化とシミュレーション・最適化	情報一般，機械情報，システム	創造性，論理，問題解決	情報処理，システム
	E．計算モデル的思考	情報一般，機械情報	創造性，論理，問題解決	情報処理，システム
	F．プログラムの活用と構築	機械情報，情報処理，システム	論理，問題解決	情報処理，システム
情報の整理や作成・データの扱い	B．情報の整理と創造	人間社会	創造性，論理，コミュ，主体性	
	D．データとその扱い	情報一般，機械情報，情報処理，人間社会	創造性，論理，問題解決	情報処理，システム
情報コミュニケーションや情報メディアの理解	G．コミュニケーションとメディアおよび協調作業	情報一般，機械情報，人間社会	創造性，問題解決，コミュ，チーム	倫理社会
情報社会における情報の倫理と活用	H．情報社会・メディアと倫理・法・制度	機械情報，人間社会，システム	論理，問題解決，コミュ，チーム	システム，倫理社会
（総合情報処理能力）	I．論理性と客観性	機械情報，人間社会，システム	論理，問題解決，コミュ，チーム	倫理社会
	J．システム的思考	人間社会，システム	問題解決，コミュ	システム
	K．問題解決		問題解決，チーム，主体性	システム

図 1.3: 情報教育における分野の分類

[88]P.5 より引用

1.5　アクティブラーニング・PBL

これまでの基礎教育では,「読み・書き・そろばん」が重視されてきた．この時代は，批判的に考えながら読むことや説得力のある自己表現ができるようになることは，それほど重視されていなかった．しかし，20 世紀の終わりには，技術革新，国際化および，情報化の波により求められる能力・資質が変化し，21 世紀型スキル [6] が求められるようになった．そして，21 世紀型スキルは，デジタル時代となる 21 世紀以降に必要とされるリテラシー的スキルとして定義された．

1.4 節で示したように，21 世紀型スキルは，社会の変化に対応できる能力として，従来の知識偏重型の教育に基づいたスキルでは対応が難しいとし，デジタル時代に必須の情報リテラシーや情報通信技術に関するリテラシーに加えて，批判的思考力，問題解決能力，コミュニケーション力などを取り上げている．

Center for Curriculum Redesign[9] は，21 世紀の教育のカリキュラムデザインのための概念として，図 1.4 を作成した．知識・スキル・人間性の 3 要素からなり,「何を知っているか」,「知っていることをどう活用するか」,「社会の中でどのように関わっていくか」，そして,「どのように反応させ適応するのか」を示した．著

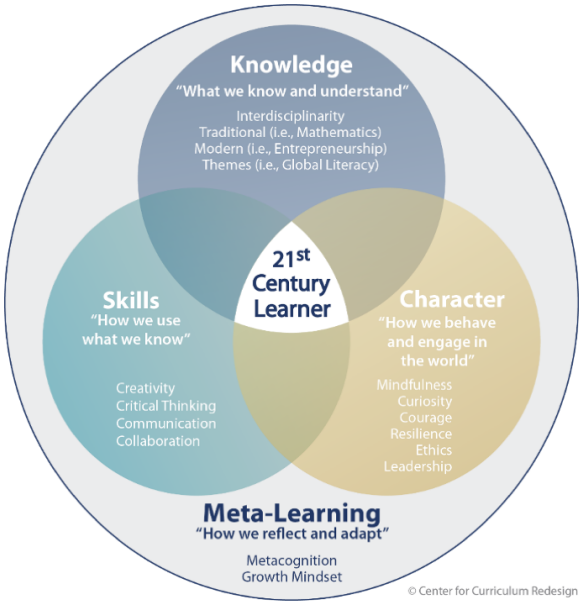


図 1.4: 21 世紀の教育

者の考えるカリキュラムでは，知識としてプログラミングの基礎を学び，コードを書きもの作りをする過程において，創造性，思考力，協働作業が伴いスキルの向上が期待でき，チームによるプロジェクトを加えることで人間性を育むことができる．

時代の経過とともに，覚えて暗記する教育から情報を発見し利用できることへと変化し，学習とは，受動的に知識を取り入れることではなく，題材に能動的に取り込み，それによって学習者の内部にある既存知識と新たに学ぶことがらの関連が多く作られることを通じて行なわれる，ということが今日の中心的な考え方になった [23][26]．

Buransford らは，学習者が自分の学習の過程を自分自身でコントロールする能動的学習は，次の点を重視すると述べている [23]．

- 学習者が理解の程度を自分自身で認識する
- 他者の意図を正しく理解しているかどうかを自分で確認する
- あることを主張するためにはどのような検証が必要であるかを認識する
- 自分自身で構築した理論を自分自身で検証したりできるようになる

Papert は，LOGO でコンピュータに絵を描かせることにより遊びの経験が知的好奇心へと繋がり学習効果が高まると主張している [26]．

能動的学習の考え方を取り入れた学習の進め方は，アクティブラーニング (active learning) と呼ばれるようになった．そして，「教師の説明を聞いて受け身で学ぶの

ではなく，各学習者が主体的に課題に取り組むことを通じてさまざまな事柄を学ぶ」ことが望まれるようになった．

文部科学省の高大接続システム改革会議では，従来型の (知識伝達型の) 学習方法の限界について言及し，これからの学校教育では，アクティブラーニングを中心に据えるべきであると提言している [111]．これを受け，文部科学省が取りまとめた 2017 年〜2018 年公示の小学校・中学校・高等学校学習指導要領でも，同様の考え方を重視している[1]．

2012 年 8 月，文部科学省は中央教育審議会において「新たな未来を築くための大学教育の質的転換に向けて–生涯学び続け，主体的に考える力を育成する大学へ–（答申）」を出し，アクティブラーニングへの転換の必要性について示した [118]．予測困難な時代において高等教育で培う「学士力」として,「成熟社会において求められる能力」を次のようにまとめた．

- 答えのない問題に解を見出していくための批判的，合理的な思考力等の認知的能力
- チームワークやリーダーシップを発揮して社会的責任を担う，倫理的，社会的能力
- 総合的かつ持続的な学修経験に基づく創造力と構想力
- 想定外の困難に際して的確な判断ができるための基盤となる教養，知識，経験

これらの能力の育成をどのように効果的に高等教育の中で育成していくかに関しては，難しい課題である．その 1 つの解として，本提案のチームで取り組むプロジェクトが考えられる．

アクティブラーニングの 1 つに，PBL と呼ばれるものがある．PBL には,「問題解決型 (Problem-based Learning)」と「プロジェクト学習型/課題解決型 (Project-based Learning)」の 2 つがある．この 2 つの PBL を厳密に分けずに用いるケースもあるが，本論文では，ソフトウェア開発プロジェクトを取り上げることから，PBL を後者のプロジェクト学習の意味で用いる．

溝上は，PBL について,『アクティブラーニングとしての PBL と探究的な学習』[103] で，次のように定義している．

> 問題解決型は，1960 年代の後半，カナダのマックマスター大学メディカルスクールで開発されたものだと考えられている．
> 問題解決学習とは，実世界で直面する問題やシナリオの解決を通して，基礎と実世界とをつなぐ知識の習得，問題解決に関する能力や態度等を身につける学習のことである．
> プロジェクト学習は，20 世紀初頭の，主として初等教育におけるキル

[1]学習指導要領では，これまでに多く使われてきて人により定義が異なる「アクティブラーニング」の用語を避け,「主体的で深い学び」という用語を用いている．

> パトリックの「プロジェクトメソッド (project method)」にルーツがあると説明されることが多い．
> プロジェクト学習とは，実世界に関する解決するべき複雑な問題や問い，仮説を，プロジェクトとして解決・検証していく学習のことである．学生の自己主導型の学習デザイン，教師のファシリテーションのもと，問題や問い，仮説などの立て方，問題解決に関する思考力や協働学習等の能力や態度を身につける．

PBL の取り組みの中には，単に商品を作って販売するようなものもあり，著者は，過程を重視せずに結果ありきでものを作って終わってしまうケースを見てきた．アクティブラーニング・PBL の実施には，様々な困難があり，失敗事例も報告 [115] されていることから，これらの知見をもとに，失敗を回避するよう留意したカリキュラムの検討ができるであろう．

著者は，前述した Buransford らの能動的学習の重視する点を，数値化したり可視化することができれば，より効果的な学習が可能になると考える．しかし，学習の状況を，学習者が逐次確認しながら学ぶことが望ましいが，学びを可視化することは難しい．著者は，その 1 つの解として，プログラミングは，効果的なツールであると考える．

プログラミングでは，コードを書く前に設計をする．その際，作りたいものを考え細分化して擬似コードに起こしたり，方眼紙に下絵を描いたりする．それをコードにし，実行するが，その際に，間違いがあればエラーが出て，正しければ実行結果が出力される．このプロセスにより，学習者が個々に確認しながら学習することができる．また，コードを書く行為は，社会に出れば 1 人でやることよりも，チームで関わり分担しながら 1 つのシステムを完成させることになる．この擬似的な環境を構築し，授業設計ができれば，チームでプログラミングを学ぶカリキュラムが構築できると考える．

著者は，こういう点で，プログラミングと PBL との親和性が高く，プログラミング入門科目であっても PBL を取り入れる意義があると考える．

さらに，著者が用いた描画教材によるお絵描きプロジェクトでは，プログラミングの実行結果が絵で示されることから可視化され，視覚的に分かりやすくなる．これにより，考えやプロセスが可視化でき，少ない教材であっても効果的な学習につながることを期待する．

1.6 本論文の目的

本論文の目的は，初等中等教育でのプログラミング教育が体系化されたことを受け，小学校プログラミングが普及した先を見据え，大学の一般教育におけるプログラミング教育のあり方も変化が必要であると考え，社会へ出る前に誰もが学ぶべき一般情報教育におけるプログラミング入門のカリキュラムを示すことであ

る．現在，大学に在学する学生は，学習の機会がないまま，社会に出るものもいる．そのため，初等中等教育からの連続性と社会への接続に配慮した学習環境の整備は，急務であると考える．

現状，高等教育におけるプログラミング教育は，各学校の裁量となっており，誰もが学ぶ必須の科目がないことから，問題提起として次の 2 点にまとめる．

- 小学校プログラミングが普及した先を見据え，大学一般教育におけるプログラミング教育のあり方も変化が必要ではないか？
- 現状，小学校におけるプログラミング教育を受けた児童が大学に入学するまでには, まだ 7,8 年の時間がかかることから，高等教育においても，誰もが効果的にプログラミングを学ぶ機会を設けることは急務ではないか？

これに答えるために，本研究では，誰もが学ぶ観点から一般情報教育でのプログラミング教育を取り上げ，対象は，情報系の専攻でない学生が学ぶためのカリキュラムを検討する．

これまでに実施された実態調査 [54] からは，多くの大学で既に実施されている一般情報教育に該当する時間は 2 単位であったことから，最小の時間として 2 単位で実施することが望ましいと考える．また，初等中等教育で行われている内容と同じ内容を繰り返すことも意義がない．

1.1 節で述べたように,「学習の概念自体も個々の学習者が習得するという発想から脱皮し，市民が社会実践活動の中で互いに学び合うという側面を重視した概念に変わる必要がある」[69] と言われており，1.5 節で述べたように，座学中心であったものからアクティブラーニングや PBL が求められていることから，プロジェクト型のカリキュラムを検討する必要があると考える．

そして，コミュニケーション力やチーム力が求められる時代となったが，初等中等教育では，チームによるソフトウェア開発プロジェクトに関する内容は含まれておらず，社会におけるプログラミングの活用を考えると，チームによるソフトウェア開発プロジェクトの内容を含んだ入門レベルのカリキュラムが有効ではないかと考える．

このような観点から，本論文では，2 単位で構成するカリキュラムを提案し，プログラミングの概念の理解とソフトウェア開発の概要の理解に加えて，チームでの活動から社会で求められる資質・能力の育成を図ることができるかどうかを明らかにする．

まず，基本となるプログラミングの概念の理解とジェネリックスキルの育成の観点から，リサーチクエスチョン 1 (RQ1) を以下のようにまとめる．

RQ1： PBL により，プログラミングの概念の理解と社会で求められる資質・能力の育成をはかることができるか？

著者は，前述した問題提起に対しての解決策として，プログラミングの基本知識を学ぶフェーズと，学んだ知識を活用するフェーズに分け，PBL を組み合わせ

る方法を提案する．これにより,「プログラミングの基礎知識」とチームによる「ソフトウェア開発プロジェクト」の両方を学ぶことができると考える．

しかし，初学者に，本格的なソフトウェアを開発させることは困難であること，教員の負担も少なく，誰にでも取り組みやすい入門レベルの教材である必要があることから，プログラミング入門で広く活用されている描画教材を用いることを考える．そして，これを応用し，プロジェクトでは，絵を描くことをソフトウェアに見立てることを提案する．

描画教材は，ソフトウェアを模擬するだけでなく，プログラミングの実行結果が可視化されことにより，考えやプロセスが可視化でき，効果的な学習につながるのではないかと考える．さらに，可視化により体験的に理解が進むことも期待できることから，リサーチクエスチョン 2(RQ2) を次のようにまとめる．

RQ2： PBL によるプログラミング教育は，学習の過程を可視化することができるのか？

これまで，プログラミング入門において描画教材を用いた先行研究はあるが，チームによるプロジェクトに関する研究はない．ここでいう “視覚化” とは，目に見えることだけでなく，体験により体感して理解を深めることも含める．

これまで，情報を専門としない入門レベルの教育に対して，プログラミングとチームでソフトウェア開発の両面を組み合わせて 2 単位で取り組まれた事例はないことから，次の仮説を立てる．

初等中等教育からの連続性と社会への接続に配慮し，高等教育における一般情報教育では，プログラミング入門とソフトウェア開発プロジェクトを組み合わせたカリキュラムでの実施が有効である．

本論文では，この仮説に基づき，カリキュラムを設計し，そのカリキュラムを実装した実践授業を行い，次のことを明らかにする．

1. 入門レベルの 2 単位で行うカリキュラムを示し，構築したカリキュラムを実装した実践授業を行い，次の項目を明らかにする．これにより，提案するカリキュラムの妥当性を検証する (RQ1).
   - プログラミングの概念の理解
   - ソフトウェア開発プロジェクトの理解
   - 社会が求めるスキル (ジェネリックスキル) の育成
2. 描画教材による学習者への効果について，次の項目を明らかにする (RQ2).
   - プログラミングの活用とソフトウェア開発プロジェクトによるプロジェクトへの理解と効果
   - 社会が求めるスキル (ジェネリックスキル) の育成への効果

1.7　まとめ・論文の構成

プログラミング教育は，初等教育での実施が始まり，義務教育における教育環境が整ってきた．しかし，高等教育での学習環境は必須化されていないことから，著者は，高等教育においても，継続してプログラミングを学べる機会を設けること，そして，社会への接続を意識したカリキュラムが必要であり，小学校プログラミングが普及した先を見据え，教育方法も変化が必要であると考え，高等教育におけるプログラミング入門教育をテーマに取り上げた．

本章では，問題提起として次を示した．

- 小学校プログラミングが普及した先を見据え，大学一般教育におけるプログラミング教育のあり方も変化が必要ではないか？
- 現状，小学校におけるプログラミング教育を受けた児童が大学に入学するまでには, まだ 7,8 年の時間がかかることから，高等教育においても，誰もが効果的にプログラミングを学ぶ機会を設けることは急務ではないか？
- プログラミングを学ぶことで獲得されるスキルは，どのように獲得されるか? 描画教材により，可視化できるか？

そして，RQ1,2 を次のように示した．

RQ1： PBL により，プログラミングの概念の理解と社会で求められる資質・能力の育成をはかることができるか？

RQ2： PBL によるプログラミング教育は，学習の過程を可視化することができるのか?

これらを検証するために，

初等中等教育からの連続性と社会への接続に配慮し，高等教育における一般情報教育では，プログラミング入門とソフトウェア開発プロジェクトを組み合わせたカリキュラムでの実施が有効である．

という仮説をたてた．

この後の章の第 2 章では，提案するカリキュラムを構築するために，高等教育におけるプログラミング教育とアクティブラーニング・PBL に関する関連研究・先行研究を取り上げる．

第 3 章では，プログラミング入門に PBL を加えて学ぶ方式を提案し，高等教育における一般情報教育のカリキュラムを設計する．PBL によるプロジェクトには，チームで取り組むソフトウェア開発プロジェクトを取り上げる．対象は，全学生 (情報系を専門としない学生が中心) であり，本格的なソフトウェアを開発することは困難であることから，1 つの絵をソフトウェアに模擬した描画教材を提案する．

第 4 章では，設計したカリキュラムを実装し，実践授業を行い， 次の評価と検証を行う．

1. 90 分 15 回 2 単位のカリキュラムの妥当性について，到達目標の観点から実装したカリキュラムの評価
2. 社会で求められる資質・能力の育成について，コンピテンシー評価の分析による検証

第 5 章では，第 4 章で行なった実践授業での提出物を分析し，プログラミングの概念の理解と活用について検証する．

第 6 章では，第 4 章および第 5 章の実践授業の結果から明らかになった問題の見直しを図り，カリキュラムと教材の修正を行う．修正に基づき実践授業を実施し，履修者の提出した課題から，プログラミングの概念の理解と活用について再度検証する．さらに，描画教材とプロジェクトによる効果を検証する．

第 7 章では，社会への接続の観点から追跡調査を行う．実践授業の履修者を対象に，履修後の活動への影響をアンケートにより調査する．

最後に，第 8 章で，本研究の成果をまとめ，初等中等教育における学習指導要領改定によるプログラミング教育が普及し定着した後の展望について述べる．

第2章

関連研究・先行研究

2.1 はじめに

急速に進む情報技術の発展は，学校教育にも影響を与え，学校での学び方や社会が求める能力・資質などに，さまざまな変化をもたらしてきた．本章では，本論文の目的である，社会へ出る前に誰もが学ぶべき一般情報教育におけるプログラミング入門のカリキュラムを提案する準備として，対象とする高等教育における関連する研究を整理して議論する．

1.3 節で，プログラミング教育についての経緯・動向について取り上げた．本章では，これに基づき，本研究の中心となる高等教育を取り上げ，一般情報教育とプログラミング教育についてカリキュラムモデルと実践事例を取り上げる．

1.4 節で取り上げた求められる資質・能力の育成は，1.5 節で取り上げたアクティブラーニング・PBL と関連することから，これらの項目をまとめて 2.3 節で取り上げる．さらに，ソフトウェア開発プロジェクトを取り入れる提案に対し，専門でない専攻における事例がないことから，専門教育としてのソフトウェア開発における研究を取り上げて検討する．

以下の節では，次の項目について取り上げて，議論する．

- 高等教育におけるプログラミング教育
  - 一般情報教育とプログラミング教育
  - プログラミング入門教育の実践事例
- アクティブラーニング・PBL
  - アクティブラーニング・PBL によるカリキュラムの考え方
  - 高等教育における PBL の実践事例
  - ソフトウェア開発における PBL
  - プログラミング入門教育と PBL

2.2　高等教育におけるプログラミング教育

2.2.1　一般情報教育とプログラミング教育

カリキュラムモデル

我が国の高等教育における情報教育は，古くは 1970 年代から情報処理教育として始まった．技術の進展と普及に伴う利用の拡大は，インターネットの到来とともに速さを増し，国立大学での全学必修化の実施を経て，私立大学を含めた多くの大学に浸透してきた．

大学教育においては，専門教育に限らず，誰もが学ぶ一般教養科目があり，さまざなレベルでの教育が行われている．一般教養に該当する情報教育に関連する科目として，一般情報教育がある．

大岩は，1991 年に，一般情報教育に関しての総論をまとめた [49]．その中で，一般情報教育の教育目標に関して，次のように述べている．

1. 「知識」と「情報」を資産とする情報化社会において，情報の価値を知るとともに，これを資産として使いこなし生きるための対応力を習得させる．
2. 情報機器に慣れ親しむ機会を与え，情報システムに対するアレルギーがないようにする．
3. 情報に関する基本的な概念 (情報処理の動作原理とその可能性，限界) を身につけさせる．

そして，一般教育の目指すものは，専門教育が学生の能力を十分に発揮できるようにする基礎を与えるものでなければならないとし，次の 2 つの視点を挙げた．

- 有用な情報技術であっても適用分野に固有のものは，一般情報教育の中核としては取り上げない方が良い
- 計算科学は，専門外の人には理解されにくいが，思考の枠組みを与えるものであり，その基礎的な部分は，一般情報教育の中核となり得る

大岩は，この考えに基づいた一般情報教育の内容を「計算機リテラシ教育」と「システム構築能力の育成」に分け，以下の具体的内容を示した．

A. 計算機リテラシ教育

- キーボード教育
  キーボードが原因で計算機アレルギーを起こすことから，数時間の訓練が必要であり，情報教育を成功させる第 1 歩として行うべき
- ワープロと文書作成，電子郵便
  知的生産における計算機の利用や KJ 法のような発想法を合

わせて教育することが良い
ワープロ，ネットワーク，電子郵便や電子掲示板さえ整えば，
情報化社会が実感できる

- 表計算，データベース

B. システム構築教育

- 問題のモデル化と情報の表現
- アルゴリズムの作成と評価
- 大規模系の構築技法

情報機器を単に道具として使えるようになっただけでは，一般教育としての意義が薄れるとし，以下を挙げている．

- 進歩の早い情報機器の場合，単に使い方を教えただけでば，すぐに時代遅れになってしまうため，道具の使用を通じて，能動的な思考訓練を行うと同時に，新しい道具が出現した時に，それを自習できる能力を身につけること
- 使い方を覚えるだけでなく，どうしてそのような使い方をするかという理由や意味まで理解できるようになること

大岩の論文から，すでに 30 年ほどが経過しており，情報通信機器の急速な発展は，身近なところまで入り込み，誰でも自由に触れるようになってきた．いつでもどこでもネットワークにつながり，授業での教育環境も変化している．プログラムは，書いてすぐに実行できる時代となり，大学の特別な設備でなくても対応できる．これでまでできなかったことができる環境が整ったからこそ，教育内容もそれに伴い変化する必要があると考える．

現在では，計算機リテラシ教育に掲げられた内容は，中等教育における学習指導要領の中で実施されるようになり，システム構築教育に掲げられたモデル化やアルゴリズムについても，高等学校の学習指導要領に含まれている．しかし，著者の環境においては，大学入学までにこれらの知識を十分に身につけられている学生は少数であり，社会に出る前に学習する機会の拡大が望まれる．特に，高等学校までの学習指導要領に含まれていない，システム構築教育については，興味深い．

大岩は，システム構築教育について，これを単なるプログラミング教育であると捉えるべきでなく，抽象化の概念が知的生産一般に有効な概念であると述べている．対象とする問題自体に対して何が本質的で重要なのかを考察し，重要度の低いことは切り捨てる抽象化を行わなければ良いプログラムは書けないと述べ，このような抽象化を正面から取り上げたプログラミングの教科書は見当たらないと述べている．

また，大岩は，従来の大学教育において，問題とそれに対する考察は教育されてきたが，情報教育の場合，考察が不十分であるとプログラムの作成に時間がか

かり，作成されたプログラムも満足のいくものにならないことから，プログラムは，考察結果が具体的なものとして得られる点で教育効果が大きいと述べた．そして，計算機に何かある程度複雑な仕事をさせるには，その内容を計算機が理解できるような形式で命令していかなければならないが，このような命令を下すことは，自分のしたい仕事がよく理解できている社会人の場合には可能である．しかし，受け身の勉強しかしたことのない大学生に対しては，教育が必要となり，仕事を能動的に行う体験を持たせなければならないと述べている．

そのため，大岩は，このような教育には，個々のソフトウェアの使い方を教えるよりも，基礎プログラミング教育を行う方が適切であると述べ，アルゴリズムに関すること，学生が作成したプログラムが，作成者以外にも読めるように作らせなければならないと述べている．そして，これを行う方法の一つとして，チームでの共同作業によりプログラムを作らせることを挙げ，数年間にわたってある規模のシステムを学生に構築させ，先輩学生の成果を土台として，システムを発展させていくことができれば，さらに効果的であると述べている．

著者は，本論文で検討するプログラミング教育のカリキュラムでは，身につけた知識を実社会で活用することを重視している．そのため，著者は，大岩のチームでの共同作業によるシステム構築の趣旨には，大いに賛成である．しかし，システム構築は，教える教員の能力や機器などの教育環境の準備が難しく，誰にでも対応できるとは思えない．また，毎年入れ替わる学生が，継続的にシステムを構築していくような仕組みを整えることは，誰にでも容易にできることではなく，誰もが学ぶ環境として構築するには，困難が伴うと考える．抽象化の概念を学ぶ要素を含め，誰にでも取り組みやすいカリキュラムの構築をする必要があると考える．

その際，誰もが取り組めるためであっても，簡略化しすぎて手抜きになってはならない．「計算機に命令を下すことは，自分のしたい仕事がよく理解できている社会人の場合には可能であるが，受け身の勉強しかしたことのない大学生に対しては，仕事を能動的に行う体験を持たせなければならない」という部分に注意し，検討する必要があると考える．

少ない教材であっても，十分学べるような教材を検討する必要があるが，著者は，ソフトウェア開発の工程に注目し，描画教材によりソフトウェア開発プロジェクトを教材化できると考えた．プログラミング入門に PBL を組み込んだ方法を提案し，大岩が述べた抽象化と十分な考察を参考にカリキュラムを構築を検討できると考える．

河村は，2013 年度から 2015 年度の 3 年間で採択された科研費（課題番号:25350210（研究代表者 河村一樹））「大学における一般情報教育モデルの構築に関する研究」の成果報告として，『これからの大学の情報教育』[57] をまとめた．

2013 年施行の学習指導要領により，初等中等教育における情報関連の取り扱いが改訂されたことから，大学の一般情報教育も見直しを図る時期にきているとし，全国規模で大学の一般情報教育の実態調査を行い，その分析をもとに，2016 年度

以降における一般情報教育に関するモデル (General Education Model，以降 GEM と略す) を構築した．

この調査では，河村は，GEBOK[74] を構成する知識群をもとにアンケート項目を編成した．GEBOK は，2001 年，2002 年の文部科学省委託研究により情報処理学会が，全国規模での大学などにおける一般情報処理教育の実態調査 [77] を実施した結果に基づき策定された．GEBOK は，大学での教養教育としての情報教育に関するカリキュラム標準の基礎となる一般情報教育の知識体系である．

河村は,「GEM の構築」にあたり，GEBOK の策定後 10 年余りの経験から，GEBOK が提唱した知識体系を公開するだけでは，全国の高等教育機関への一般情報教育の推進は困難と考え，高等教育における「入口」の基準と「出口」の基準を明確にた上で，一般情報教育における「内容」に加え「教授法」「教材」「評価法」も検討対象として，最適なモデルを提案することを計画した．図 2.1 に，河村が構築した，大学における一般情報教育のモデルを示す．

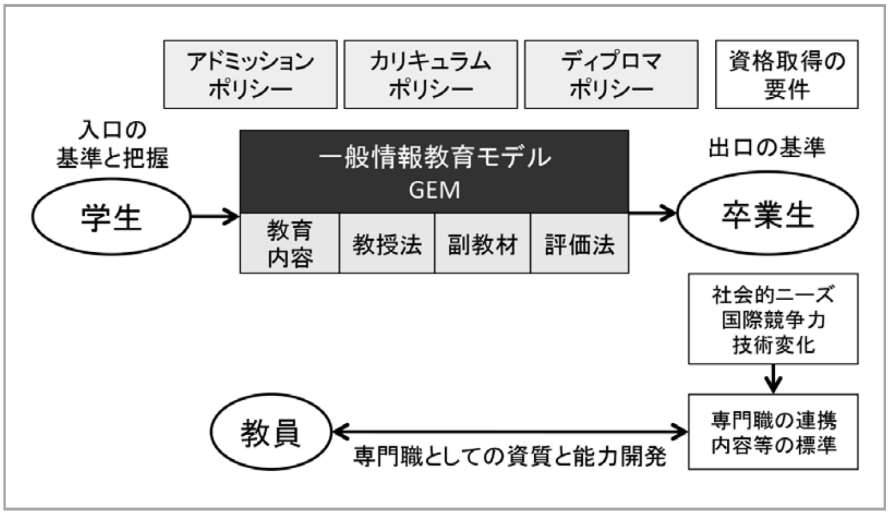


図 2.1: 大学における一般情報教育のモデル

河村は，教育の情報化を踏まえた教授法や，ルーブリックの導入の提案をした．そして，大学におけるプログラミング教育について，これまでの経緯と現状を踏まえ，今後のあり方を展望した．情報概念を意識させるためには，まず，計算機リテラシ教育を行う必要があり，それには，ワープロなどの情報機器を使用する経験を通じて養うことができる．しかし，単なる使用経験だけでは情報機器の可能性やその限界を知ることができず，将来，情報化社会のリーダーとなる大学生に対しては，システム構築教育までを行うことにより，情報機器の使用の有無のような基本的問題に対する判断能力を養う必要があることを述べている．

大岩は，大学の一般教育と実社会での活用を重視し，河村は，初等中等教育での情報教育の流れを汲み，入口の把握と入口の基準を加え，初等中等教育での情報教育の意義と課題から，大学への接続を検討している．そして，どちらも同様

に，システム構築による教育について触れている．

しかし，著者の環境においては，情報関連の専門家が少ないことから，このような理解には発展せず，操作中心の教育が要望されており，システム構築による教育に至るまでには，ハードルが高い．スマートフォンの普及に伴い，学生はデジタルネイティブ世代と言われ，機器の使用には慣れてきている．また，初等中等教育における情報教育も普及し，情報機器の活用は，一から教える必要もなくなってきている．しかし，高等学校における教科「情報」の導入は，大学入学時点でのスキルアップが期待されたが，著者の環境においては，事前の習熟度にばらつきがあり，いまだに初年次教育には苦労している．大学入学以前の学習経験の違いにより学生は多様であり，このような学生をうまくまとめて授業を行うための工夫をする必要がある．

加えて，著者の現場の体験からは，不足する知識は，学んだ知識を活用する部分であると感じている．本研究で提案する方法であれば，ワープロなどの活用については，授業を進める中で習得できると考える．現在行われている授業の中に押し込む形で，これからの大学における情報教育を考えるべきである．

実態調査

これまでに，文部科学省および学会等により，一般情報教育の実態調査が行われてきた．情報処理学会では，2001 年，2002 年に文部科学省委託研究を実施し，全国規模での大学などにおける一般情報処理教育の実態調査を実施した [77]．これにより，以下の 9 項目が明らかになった．

1. 開講している科目は 2 単位
2. 操作演習を主としていると思われる科目名が多い
3. 全体的には必修よりも選択での開講の方がやや多い
4. 教えている項目（頻度の多い順）はソフトウェアの操作＞ 文書作成＞ ネットワークアプリケーション＞ オペレーティングシステム＞ ハードウェアの操作＞ 日本語入力＞ キーボード＞ Web ブラウザ＞ マウス＞ ワードプロセッサというようになり「操作」に関するものが上位を占める
5. 52.7% の科目で「ワードプロセッサ」を教えている内容にあげている
6. 学生 10 に対して教員 1（TA を含む）が平均的な構成員の比率である
7. 一般情報処理にかかわる教員数は情報系以外の分野が圧倒的に多い
8. 一般情報処理教育（授業）の責任を負っている組織の特定は困難
9. 一般情報処理教育を支える環境はある程度までは整備されている

2008 年には，中央教育審議会答申「学士課程教育の構築に向けて」[119] において，汎用的技能のひとつとして「情報リテラシー」が含まれ，情報教育の推進がますます強まっていった．

河村らは，科学研究費（課題番号：25350210（研究代表者 河村一樹））「大学に

おける一般情報教育モデルの構築に関する研究」の成果報告『これからの大学の情報教育』[57] において，中核的科目群（全学部必修，各 2 単位）として「情報とコンピューティング」「情報と社会」を，補完的科目群（学部毎に選択，各 1-2 単位）として「プログラミング基礎」,「情報システム基礎」,「システム作成の基礎」,「情報倫理」,「コンピュータリテラシー」を，それぞれ取り上げた一般情報処理教育カリキュラムを策定した．そして，その当時，多くの大学で実施されている半期 2 単位の一般情報処理教育を，通年（半期 4 単位でも可）に増やすことを提言した．

情報化の急速な進歩発展に伴い，情報教育の重要度が増し，授業時間も増やすことは理想であるが，他の科目との兼ね合いもあり，どの大学も時間数を増やすことは，容易なことではない．また，誰もが学びやすくするためには，時間はなるべく短い方が好ましいと考える．そのため，著者は，2 単位であっても，展開できる授業を検討したい．

2016 年 3 月には，日本学術会議において「大学教育の分野別質保証のための教育課程編成上の参照基準 情報学分野」（情報学分野の参照基準）が策定された．ここでは，”エ：情報を扱う人間社会に関する理解 ”という項目が付け加えられた．情報学分野の参照基準は，専門教育をターゲットとしているが，情報教育一般との関係についても記載があり，文系・理系によらない基準が示された．

その後，2016 年 11 月から 12 月にかけて，文部科学省の「超スマート社会における情報教育の在り方に関する調査研究」[122] の一環として，情報処理学会は委託研究として国内の全大学 (約 750 大学) における情報学分野の教育に関する調査を実施した [54][55][76]．多様な大学における情報教育の分類を 4 種類に分け (図 2.2)，プログラム構成，教育内容と教育レベル，プログラム履修者，担当教員・補助者，教育環境などのデータを収集した．

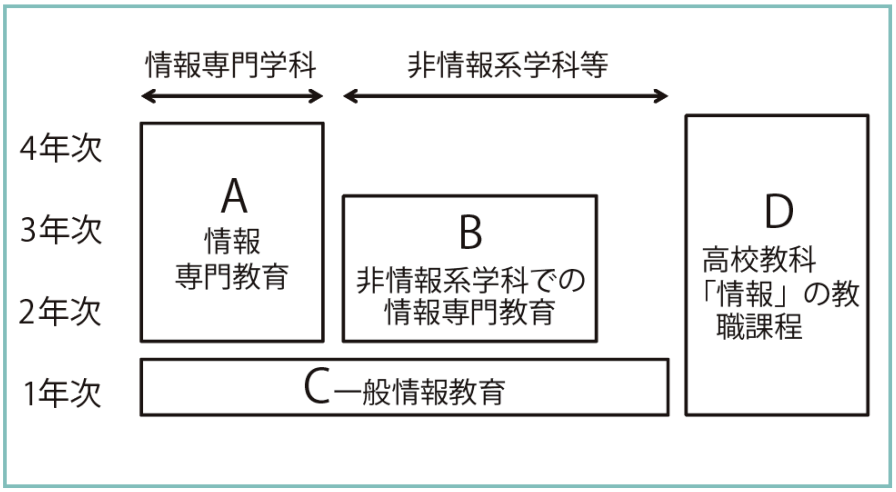


図 2.2: 大学における情報教育の分類

教育内容と教育レベルの調査では，情報学の参照基準および J07-GEBOK に基

づき，21 領域・90 項目の調査項目を定義し，知識到達度と技能到達度を 0〜5 の 6 段階で定義したレベルを用いた [54].

この調査によると,「アルゴリズムとプログラミング」は,「コンピュータリテラシー」，「情報倫理とセキュリティ」など他の項目と比較して「教えていない」学部学科の率が高い [76].

前述の河村らの調査から 15 年が経過しているが，高等教育における情報教育には，プログラミングに関する内容は浸透してきておらず，今後の学習の機会の拡大が望まれる [76]．小学校でのプログラミング教育が始まり，高等学校までにプログラミングを学ぶ環境は整ってきた．しかし，社会に一番近い高等教育においては，全員が学ぶ環境はいまだに整っていない．これまで行われてきた調査結果を振り返ると，使える時間は，2 単位であると考える．一般情報処理にかかわる教員数は情報系以外の分野が圧倒的に多いことから，授業の準備の軽減や誰にでも教えやすい教材の開発も重要な項目であると考える．教えやすい教材であり，少ない内容であっても，誰もが理解を深められる教材開発を心がける必要があると考える．

文系学部における情報教育

ここでは，文系大学における情報教育を取り上げる．

原田は，文科系大学・学部における情報教育での目的と問題について述べ，一般情報教育の目標として次の 3 つをあげた [92].

目標 1 リテラシ教育としての情報教育

目標 2 教養としての「情報」教育

目標 3 考える訓練，知的創造の場としての情報教育

著者は，原田の掲げた目標 3 に注目した．著者の所属する大学も文科系大学であるが，学生のアンケートからも考える機会が少ないとの回答を得ている．また，自己表現をする機会も少ない．

原田は,「ごく簡単なものであっても，自分の思う通りにコンピュータを制御するというプログラミングは，考えることが楽しいものだということを伝える力を持つ．また,「論理的に考える」ことが，決まりきったことを小難しくいうのではなく，自分の意図を正確に伝えるために重要なのだ，ということに気付く契機ともなる.」と述べている．

この点は，著者の考えと同様であり,「考えることの訓練」の場としてのカリキュラム作成に意義を感じている．そして，ごく簡単なものであってもという点に注目したい．文系大学に入学してくる学生は，理系科目で躓いている者や，コンピュータ苦手意識を持つ者も多く，苦手意識が先行し，本質を学ぶ以前の先入観から躓いてしまっていることを見てきた．そのため，先入観を持たせずに，楽しい，できた，わかったというようなポジティブな感覚を持たせるところからスタートすることが重要であると，これまでの経験から考える．

2.2.2　プログラミング入門教育の実践事例

高等教育におけるプログラミング教育は，情報系の専攻では，当然のことながら多くの教育実践が行われている．しかし，前述の調査結果にも現れていたように，文系の大学においては，いまだに実施の体制が整わない．著者の在籍する大学においても同様である．

河村らは,「大学における一般情報教育モデルの構築に関する研究」の成果報告『これからの大学の情報教育』[57] において，補完的科目群（学部毎に選択，各 1-2 単位）に,「プログラミング基礎」,「情報システム基礎」,「システム作成の基礎」を取り上げ，一般情報処理教育カリキュラムを提案しているが，多くの学校に浸透していない．

プログラミング言語を用いたプログラミング教育を行っている事例として，河村は,「手順的な自動処理」を体験する場としてのプログラミング教育という視点から，JavaScript を用いた教育の留意点や事例を提示している [56].

河村は，これまでに，情報専攻向けのプログラミング教育については，コンピュータサイエンス領域を基盤にして確立されたカリキュラムがある．しかし，非情報専攻向けのプログラミング教育については，必要性の有無から始まり，紆余曲折していて，確固たる指針が出されているとは言い難い状況であることから，一般情報教育におけるプログラミング教育の重要性について，以下の留意点や教育事例を交えて検討した．

- 情報に関する教育は，学校における ICT 環境が基盤となり，プログラミングだけでなく，アプリケーションソフトウェアや Web ソフトウェアの利用と多様化が進んできたこと
- プログラミング教育の変容により，実用言語によるプログラミングの技を習得することが目的ではなく，問題解決能力や論理的思考力の育成を目指すといった視点も組み込まれるようになったこと

さらに，河村は，一般情報教育におけるプログラミング教育の方針と教育目標は,「手順的な自動処理」の体験から得られる学習効果のあり方について検証することし，学習効果として以下を挙げた，

- プログラミングを通してアルゴリズムをデザインするための論理的な思考力が育まれること
- 解決すべき問題をアルゴリズムに置き換え定式化することによって抽象化能力が身につけられること
- プログラムを実行することによって自分で作り上げたアルゴリズムの良し悪しが判断できること

- プログラミングを実体験することによってコンピュータサイエンスの基礎的な概念（頻出概念）を直感的に把握できること

そして，プログラミング言語の選択のために，(1) 手続き型言語 (2) オブジェクト指向言語 (3) スクリプト言語について整理し，スクリプト言語に注目した．教育用言語として，JavaScript を選択し，その特徴について，以下の項目を挙げている．

- OS に依存しない
- 特別な開発環境を必要としない
  テキストエディタと Web ブラウザのみで OK
- スクリプトの動作確認が簡単
  Web ブラウザにスクリプトのデバッグ機能も標準で搭載されているため，簡単に構文エラーなどを検出できる．
- 学習者が興味を持つ題材を提供
  一連の教育内容に加えて，Web ページの見栄えをよくする動的な表現もできるため，学生の興味関心を引く

実施する上での留意点として，半角と全角の相違，大文字と小文字の区別，コメントの扱いを含むコードの記述などを挙げた.「手順的な自動処理」を体験できることの意味に基づき，アルゴリズム指向か実用指向かについて次のようにまとめた．

- 「伝統的」な構文 (if 文，if else 文，for 文，while 文，do while 文，関数と再帰) が用意されていることから，手続き的なアルゴリズムを記述することに適している
- 「伝統的」なアルゴリズム (整列，検索，文字列処理，簡単な数値計算など) を学習させるプログラミング教育ができる
- Web ページに動的な表現を組み込むことができる
- 見栄えの良い Web ページを作成するためのプログラミング教育ができる
- 学習者にとってもその実用性がよく理解できることから，学習の動機付けが高まる

JavaScript 特有のプログラミングテクニックに集中することは，言語の学習になってしまうが，教育目標達成のために，アルゴリズム指向を中心に，一部実用指向を取り入れるのが妥当であると述べている．

上記に基づき河村は，通年 1 コマ，4 単位の 1 年次の必須科目を取り上げ，このうちの後期 15 回を JavaScript を用いたプログラミング教育に当てた授業を実施例

として示した．8 回までに構文を学び，9〜12 回で配列，探索，整列などのアルゴリズムを学習し，13・14 回で実用的なアルゴリズムを学習する．最終回の 15 回では，教育目標の実現を目指し，アルゴリズムを記述したプログラムが，コンピュータの中でどのように動作しているのかという仕組みを明らかにするために，プログラム内蔵や逐次制御について説明する．

河村は，JavaScript 特有の Web ページの動的な表現を取り込もうとすると，アルゴリズムよりも言語の方に学習の比重が移ることになるため，どの程度まで扱うべきか思案中であると述べている．

著者の研究に対し，アルゴリズム指向か？実用指向か？の比重は，カリキュラムを構築する上で非常に参考となった．「手順的な自動処理」を体験する場としてのプログラミング教育という視点は，著者がカリキュラムを構築する上でも同様であり，プログラミング言語の選択の観点や授業を実施する上での細かい留意点は，非常に参考となった．しかし，4 単位の後半の 15 回の内容は，著者が置かれている学校環境下では難しく，決められた時間内での習得が難しいと感じた．誰もが学ぶことを考えると，前半の 2 単位の知識がなくても，半期 15 回で実現できるカリキュラムが望ましいと考える．

岡本らは，[48] で，プログラミングの学習の問題点として，サンプルプログラムを使って，プログラムの命令とその動作から各処理概念を学んでいく経験学習の方略が用いられるが，それを暗記するだけで，概念あるいは機能の理解といった段階にまで到達していない学習者が散見されることに注目した．このような事例に対し，「現在使用されているカリキュラムや教材が，経験学習の構造的特性に合致していない」ことが理解を妨げている要因の一つでなはないかと考えた．これを解決するために，プログラムと動作の関係を視覚的に「顕在化」することに配慮したカリキュラムと教材を開発した．

著者も，当初は，小学校での実践の経験から視覚化教材の有用性は認識していた．しかし，著者の経験は，ロボットキットの活用であったことから，準備や片付けに多くの時間を要することや，プログラミング以外のロボットの組み立てに関して，ハードウェアやロボットの機構の知識も合わせて習得する必要があることから，教材が複雑化してしまった．この経験から，ロボットキットの活用は，短時間での実施には向いておらず，導入は難しいと判断した．

一方で，視覚化教材としては，描画を用いる事例が多くあり，その中でも，初学者向けプログラミング学習環境である PEN を用いた先行研究 [86][127] がある．

PEN は，日本語ベースのプログラミング言語を用い，構文エラーの発生を低減させるための入力支援機能，プログラムの実行の様子を把握しやすくする機能を備えている．そのため，PEN は，初学者が短時間でプログラミングの学習をすることを容易とし，高等学校や大学等で広く活用されている．

吉田は，共通教養科目「情報処理」の授業でのプログラミングの実践授業を，情報教育へのプログラミングの導入事例として紹介している [127]．京都ノートルダム女子大学では，UNIX 系 OS をサーバーおよびクライアントとしたコンピュータ

センターが1991年から発足している．発足当初より，BASICでプログラミングを学ぶ授業や，UNIXワークステーションを用いてLaTeXによる文書作成をする授業など，リテラシー教育以外にも積極的に実施されてきた．1998年度から数年は，JavaScriptを利用したインタラクティブなWebページを作成する実習を行なった経験もある．その後，C言語やRubyを用い，半期90分15コマのプログラミング教育を行ってきている．

吉田は，限られた時間数の授業において，複雑な記述の意味を教える時間は，少しでも節約したいこと,「コンピュータによる自動的な処理」に接した体験をさせたいという目的に適したプログラム環境であることから，PENを導入した．特に，PENには，エラーが出た際や，繰り返し処理の状況を詳しく把握したい場合に,「1行実行」の機能があり，どの行で止まってしまったのかを追跡することができるため，初心者のデバッグの助けになることや各学生の実行結果がログファイルとして残せる機能があることを利点としてあげている．

吉田は，授業実践の考察として，2005年度から3年間のデータに基づき，次の評価をしている．

- PENを使うことにより，4コマあれば，プログラミングの導入は可能である．
- 5回目の授業で，繰り返し処理のプログラムを学習したが，半分近くの学生が繰り返し処理のプログラムを書けなかった．
- 7回目の授業で，画像描画の課題に取り組んだところ，提出された32作品のうち，17作品に繰り返し処理が使われていた．しかし，その作品は，サンプル画像のプログラムを真似たものが多かった．
- 繰り返し処理の理解や運用能力が身についたかどうかに注目すると，8コマを費やして学んだ学生と比較し，試験時間を含み4コマしか学んでない学生の理解度が明らかに低いことが，試験結果から得られている．

吉田の知見からは，限られた時間数の授業におけるプログラミングの授業の実施の観点は，著者の授業設計に参考となった．環境は異なるが，プログラミングの基本概念については，4〜8回の授業で対応できそうであると感じた．プログラミングを学びそれを身につけるためには，ある程度の学習時間が必要であることがわかる．この知見は，著者のこれまでの経験とも合致するものがあり，基本概念を単に学ぶだけでなく，それを活用する時間を設け，各自が自由に運用できるような授業を検討する必要がある．

吉田は，最後に，次のようにまとめている．

> どのような職業についても，情報システムを発注する側になることや，運用に携わることは，十分に考えられる．そのような場合に，プログラミングを経験しておくことで，スムーズに開発者側と打ち合わせを

したり，コンピュータのトラブルの原因を予測したりできるようになっているのなら，この体験は十分に意味があるものだと考える．

著者も，この意見には賛成であるが，この数回のプログラミングの体験を通じただけでは，このような対応ができるようになるかは疑問が残る．特に，開発者側とのスムーズな打ち合わせについては，個々のプログラミングの知識だけでは対応ができるようになるとは考えにくい．これに対処するためには，著者は，入門レベルから，プログラミングの概念だけを学ぶのではなく，チームによるソフトウェア開発プロジェクトを取り入れることで，他者が伴って活用するということを体験することができるようになり，さらに意味のあるカリキュラムが構築できるのではないかと考える．

次に，西田らは，情報科学を専門としない学部の情報リテラシ科目 6 クラスにおける 90 分× 4 回のプログラミング演習でのコースウェアが学習に与える影響について，2 つのコースウェアを用いて検証した [86]．

2 つのコースウェアとは，次のようなものである．

- 従来型：キーボードからの入力に対して，計算結果をコンソールに文字列として出力する形式の演習を繰り返す
- 図形描画型：描画用のウインドウに図形を描画する形式の演習を繰り返す

授業の進め方は，例題を開設し，その後その例題に関連した 1 つもしくは 2 つの練習問題に取り組ませることを繰り返した．また，複数の担当教員による授業であるため，各クラスの進め方が統一できるよう「授業時間記録シート」を用いた．

この授業の進め方は，著者の授業でも例題を解説し，関連した練習問題に取り組ませる点は，著者の考えと同様である．ただし，著者の授業では，練習問題は複数用意し，学習者のスキルに合わせて取り組む問題が選択できるようにした点に相違がある．各クラスの進め方の統一の工夫は，今回の著者の場合には必要ないが，今後の参考となる．

西田らは，4 年間にわたり，図形描画を主としたコースウェアと，従来型のコースウェアとで学生のモチベーション，理解度，成績に差があるかどうを調査した．繰り返し処理の内容を学習する際，従来型の授業では，その内容が難しいと評価されているが，図形型授業では，同じ内容であっても難しいと評価されず，つまずきやすい学習内容でも理解度や楽しさをほとんど下げることなく学習できていた．そして，図形描画を用いた課題は，学習者のモチベーションや理解度を上げ，これを主としたコースウェアは，初学者に対する短期間のプログラミングに有用であると考えられると述べている．

図形描画を用いた課題の活用は，著者のカリキュラムの構成に有益な知見であり，短期間のプログラミングの学習に有用であることは非常に参考となる．

河村と吉田は，コンピュータによる自動処理の経験を目的として実践提案を行なった．どちらも，学習後の実用性について触れているが，実用に対応した授業時

間数が少なく，チームで取り組むようなカリキュラムにはなっていない．河村は，一部実用指向を取り入れることが妥当であると述べ，プログラミングの実用性については触れているが，その内容は，15 回のうちの 2 回であり，また，深掘りするとアルゴリズムよりも言語の方に学習の比重が移る課題を挙げるにとどまっている．

いずれの研究も，プログラミングの知識を個人で学ぶような形式の授業であり，最終的には，履修者に対して試験で評価をしている．

著者は，試験での評価を否定するわけではないが，卒業後を考慮した実用性を考えると，吉田が述べているように，社会に出たあとのソフトウェア開発に必要な事柄 (ソフトウェアを開発する立場のみならず，依頼者の立場にあっても) を学ぶという視点を強めたカリキュラムの検討は，十分価値があると考える．そのためには，個人の知識としての学びも必要であるが，これからの情報教育としてのプログラミング教育は，入門レベルであっても PBL の要素を取り込んだ，チームでの学習の機会が含まれることが望ましいと考える．

2.3 アクティブラーニング・PBL

2.3.1 アクティブラーニング・PBL によるカリキュラムの考え方

21 世紀型スキルが求めらるような時代となり，アクティブラーニングが求められるようになってきた [105].

1.5 節で述べたように，アクティブラーニングは,「教師の説明を聞いて受け身で学ぶのではなく，各学習者が主体的に課題に取り組むことを通じてさまざまな事柄を学ぶ」ことである．

アクティブラーニングは，2010 年 8 月の中央教育審議会の答申 [105] で初めて取り上げられた．高等教育から始まった改革により，現在では，小学校から大学まで全ての学校でアクティブラーニングが求められるようになった．この答申では,「教員と学生とが意思疎通を図りつつ，一緒になって切磋琢磨し，相互に刺激を与えながら知的に成長する場を創り，学生が主体的に問題を発見し解を見出していく能動的学習 (アクティブ・ラーニング)」と定義した．

アクティブラーニング型の授業には，PBL，反転学習，ジグソー学習などさまざまな手法があるが，本論文では，ソフトウェア開発プロジェクトを取り上げ，プロジェクトベースでの授業展開を検討するため，PBL を取り上げる．そして，”P” は，様々な意味として扱われるが，本研究では，プロジェクトを取り上げる．

PBL(Problem/Project-Based Learning，課題・問題解決型学習) は，医学や工学分野で始まり他の分野にも広がってきた．今では，さまざまな分野の教育に PBL は取り入れられており，目的は，コンピテンシー育成，問題解決力の育成，地域振興など様々である．

中山は,「アクティブ・ラーニング（能動的学修）」(原文まま) の推進に効果的な

教育方法の 1 つとして，PBL 教育を紹介している [85]．大学教育における PBL 教育導入の主な目的は，現実の問題解決のイメージや実践力（能動的学修力）を培うことであり，PBL 教育を行うことそれ自体ではなく，導入の際には，導入数よりも学生の学びに関心を置き，個々の授業だけでなく教育プログラム全体を見据えた内容・導入先科目の検討を行うことが重要であると述べている．中山は，論文の中で，PBL 教育の基礎要件として，三重大学高等教育創造開発センターが策定した，6 つの PBL 教育の基礎要件 [101] を示した．

6 つの PBL 教育の基礎要件

1. 学生は自己学習と少人数のグループ学習を行う
2. 問題との出会い，解決すべき課題の発見，学習による知識の獲得，討論を通じた思考の深化，問題解決という学習過程を経た学習を行う
3. 事例シナリオなどを通じて，現実的，具体的で身近に感じられる問題を取り上げる
4. 学習は，学生による自己決定的で能動的な学習により進行する
5. 教員はファシリテータ（学習支援者）の役割を果たす
6. 学生による自己省察を促し，能動的な学習の過程と結果を把握する評価方法を使用する

中山は，この資料に基づき，自己学習とグループ学習の機会を多く設けることが要件とされ（要件 1），このことにより，学習方略について，経験を重ねることによる試行錯誤的な獲得や，模倣・観察などによる仲間からの獲得（ピア・ラーニング）が期待できること，学生による自己省察を促すこと（要件 6）は，学習過程を自らコントロールしようとする意識や，実際にコントロールしていくための能力・スキルの育成につながると考えられると述べている．

続いて，中山は，PBL 教育の導入を成功させるためのポイントとして，以下の 3 点にまとめている．

1. 問題や課題を扱うという点だけでなく，解決までの道のりを予測し整備しながら授業設計を行うこと
2. 学生のレディネス（準備状態）を把握すること
3. 授業（例えば半期 15 回）全体，全ての学習内容を無理に PBL 形式で扱おうとは考えないこと

著者は，TA のおけない演習を伴う授業に有効であると考える．著者の実施するプログラミング入門科目も，TA の配置が困難であるためペアプログラミングやチームによる実習を行うことで，TA サポートの補完を考えている．事例シナリオは，ソフトウェア開発の工程に基づき進めるプロジェクトとしてシナリオ化することにり，PBL による効果を期待したい．ソフトウェア開発では，プロジェクトマネージャーが重要な役割を果たすが，ファシリテータは，教材や教員がこれを

兼ねることにより，著者の提案する授業では，うまく進めることができるのではないかと考える．

池本らは，PBL 型教育での学生のコンピテンシー育成について述べている [38]．３年生後期までに修得してきたメディア情報の知識や技術を活用したプロジェクト演習における，PBL 型教育の有用性と課題をまとめている．この演習を通じて学生は，課題を発見し，その課題を解決する方法を考え，解決策を遂行する計画を練り，チームで役割を決め，柔軟に主体的に実行する経験をした．その過程で，学生は，プロジェクトをまとめて説明し，他の学生の批評に答え，クライアントに説明して採用してもらう交渉を行うことや，実際に遂行する上での様々な障害に対する対応，全学レベルではなく，コース内で複数プロジェクトが同時に進行することから焚き付けられる競争心，成果品の検証と納品など，様々なストレスに耐え，これらを仲間とともに克服する経験をした．

池本らは，経済産業省が 2006 年から提唱している社会人基礎力 [64]，

1. 前に踏み出す力（主体的に実行する）
2. 考え抜く力（課題発見と計画力）
3. チームで働く力（柔軟にストレスコントロールする）

という３つの要素を示し，これらに加えて実社会における個人の能力を表す概念である「コンピテンシー」（高い業績を示す者に共通してみられる行動特性）を，[83] で定義されている「ある役割において優秀な成果を発揮する行動特性」を用い，プロジェクトを成功に導く上では，コンピテンシーが重視されると述べ，この実践を通じ，コンピテンシー修得のプログラムを実施したことになると述べている．

池本らの研究では，これまでに蓄積してきた知識や技術を活用している点は，著者のプログラミングの新しい知識を蓄積しながら進める点とは異なる．プロジェクトを行うことにより，それに付随したコンピテンシーの習得が期待できることは，著者も同様に期待している．

本研究で提案するカリキュラムでは，プログラミングを学ぶだけでなく，チームでプロジェクトを進めることから，他者が関わることによる項目に期待が持てる．

2.3.2 高等教育における PBL の実践事例

学習とは，受動的に知識を取り入れることではなく，題材に能動的に取り込み，それによって学習者の内部にある既存知識と新たに学ぶことがらの関連が多く作られることを通じて行なわれる [23][26]．

文部科学省の高大接続システム改革会議では，従来型の (知識伝達型の) 学習方法の限界について言及し，これからの学校教育では，アクティブラーニングを中心に据えるべきであると提言している [111]．

アクティブラーニングの一つの方法として PBL がある．近年では，医学や工学

に限らず多くの分野でPBLによる教育が行われ，目的もコンピテンシー育成，問題解決力の育成，地域振興など様々である [85][38][46].

PBLでは，様々なことを主体的に学ぶことにより，コンピテンシーの育成や問題解決力の育成に効果的であり，本提案でも，同様に期待する項目である.

しかし，総合的なアウトプットを求める性質のPBLは，基礎知識を身につけた後の教育課程終盤での実施が学生にとって受講しやすいと述べられており [64][102]，はこだて未来大学の事例 [104] では，開学当初から3年生の必修科目として位置付けられている.

一方で，内田は，PBL実施後の学習に対するモチベーションの向上や就職活動に必要なグループワーク等のスキルの獲得を狙い，教育課程中盤となる2年生後期のカリキュラムに取り入れた [46].

基礎知識を身につけることにより，それを活用するという考えのもと，著者は，入門レベルにおけるカリキュラムの提案を行なっている．内田の知見からは，教育課程終盤によらず，教育課程の前半でもPBLでの授業実践において効果が得られている点は，著者が考える入門レベルの授業においても，PBLの導入に期待が持てる.

2.3.3 ソフトウェア開発教育におけるPBL

ソフトウェア開発の教育は，もともと課題としてソフトウェアを作るという明確な“問題”がそこに存在していることと，ソフトウェアの規模によってはグループ開発が必須となることから，PBLの枠組みに馴染みやすい．このため，大学の情報専門学科では，PBLという用語が一般的になる以前から，プロジェクト型でソフトウェア開発の実習授業が行なわれてきた．さらに，近年では，PBLの特質を意識したソフトウェア開発教育を行なう実践が現れている [41][68][97][98].

以下では，これらの実践を紹介し，本研究との関連を検討する.

駒谷は，[68] で，日本経済団体連合会の提言 [40] のデータに基づき，情報系専門教育の内容と企業が専門教育に求める内容が，前者が理論・研究指向，後者が実践・プロジェクト指向という形で乖離していることを示した (図2.3).

その上で，PBLは，実践力・主体性・問題解決力を高め，産業界の求める人材を育成することに効果的であるとし，前述の乖離を是正する教育手法として期待されていると述べている．さらに駒谷は，リアルな顧客の課題解決を目的とする，情報システム開発カリキュラムを提案している．このカリキュラムは，システム開発経験をもつ常勤教員が担当する点が特徴的であるが，教員の確保が難しいという課題がある．そのため，一般的な大学教員が，それほど負担とならない準備時間で取り組める枠組みがあれば，広く教育が行われると考える.

松澤らは，[97] で，人間と調和する情報システムを構築できる創造的な技術者の人材育成を行う環境として，産学が共同して行う「コラボレイティヴ・マネジメント型情報教育」を提案している．ここでは，学生のみでプロジェクトをマネジ

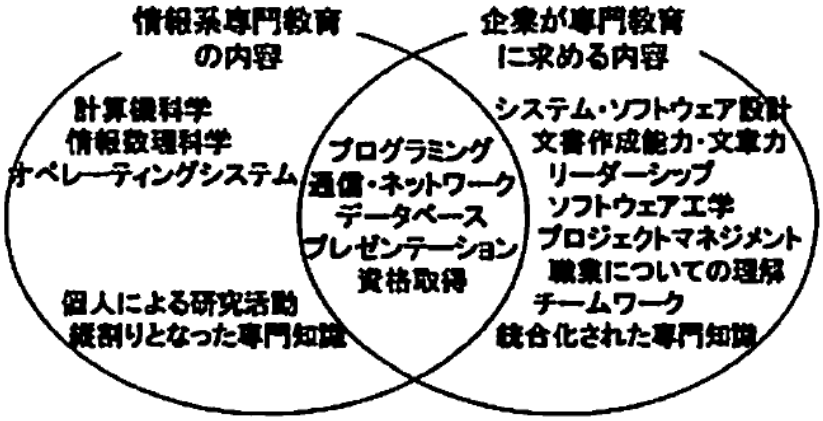


図 2.3: 専門教育の内容と企業が求める内容 [68] p. 132

メントすることは難しいと指摘し，産業界の企業人が，プロジェクトマネジメントマネジャーとなり，一緒に学ぶ環境を構築した．これにより，実践的に学生と企業人の双方を効果的に教育する．さらに，この環境において，顧客満足度の評価までが問われるプロジェクトを反復して行い，開発プロセスの質を上げながら螺旋状に学習して行く学習システムの有効性を述べている．一方で，プロジェクトマネージャーの能力によって，プロジェクトの成果・学習効果に差が出ることを問題としてあげている．大学で広く実施可能なカリキュラムとするためには，プロジェクトマネージャや担当教員の技量にあまり多くを依存しないような枠組みが望ましいと考える．

「成長分野を支える人材育成の拠点の形成 (enPiT)」[65] とは，4 つの分野 (クラウドコンピューティング・セキュリティ・組込みシステム・ビジネスアプリケーション) において，知識に加え，課題を発見して解決する実践力をつけることを目的に，人材育成を図る文部科学省が推進するプロジェクトである [42][47][126]．enPiT では，複数大学が連携してプロジェクトを実施することから，全員が一堂に介しての実施が難しい．そのため，分散 PBL を採用している．効果測定も分野や大学間で違いがあり，大学教員による評価基準の統一が困難であることから，(1) 実践力の自己評価，(2) 行動特性の客観評価，(3) 社会での評価を組み合わせている (図 2.4)[125]．このうち (2) には，標準化されたテストである PROG コンピテンシー・テスト [36][58] を採用している．山本は，[125] で，2016 年度の調査結果として，アンケートによる自己・他者の評価に，標準化されたテストの客観評価を加えることで，実践力の成長を確認できたと述べている．

評価基準の統一は，複数の組織横断では問題になるが，本研究での評価には，単一の組織での取り組みを対象としていることから，統一基準の必要はない．ただし，評価基準に関しては，本研究でも活用する必要があることから，(1) の実践力

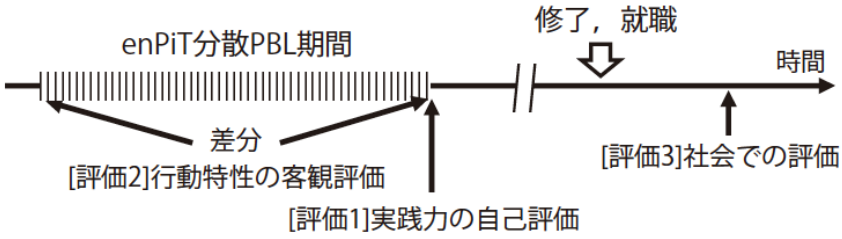


図 2.4: enPiT で行う 3 種類の評価 [125]p. 1125

の自己評価に関しては，PROG を参考にこれに変わるものを検討する必要がある．独立行政法人情報処理推進機構から出されている『コンピテンシー評価項目表 (参照モデル)』[80] は，広く活用実績があることから，利用できると考える．

2.3.4 PBL とプログラミング入門

プログラミング入門における PBL の事例として，Nuutila らが行った，医学部で広く使用されている 7 ステップメソッド (表 2.1) に着目したものがある [24]．

7 ステップメソッドでは，学生は 7〜10 人のグループに分けられ，週に 1 回，3 時間の PBL セッションを行う．その際，前の事例があれば，その事例のクロージングセッションから始め，その後，新たな案件に取り組む．

ステップ 1 と 2 は，事例の検討と問題の特定で，最初の数回は，学生がその事例のタイトルを見つけるのに苦労することがある．

ステップ 3 のブレーンストーミングでは，学生のこれまでの知識や経験と事例とを結びつけることを目標とする．

ステップ 4 は，説明モデルのスケッチである．ホワイトボードの上で付箋に書いたメモを再グループ化する．説明モデルの内容を提供することが重要であり，それによって現在の理解にどのような弱点やギャップがあるかが明らかになるはずであると述べられている．

ステップ 5 は，学習目標の設定である．自主学習のための資料（本，論文，ウェブサイトなど）をグループ内で確認する．

ステップ 6 は，自主学習期間であり，このステップの中で最も重要な期間である．学生が十分な勉強をしなければ，PBL は効果を発揮しない．最初から自学自習の重要性を強調し，受講生の士気を高く保つ必要がある．

ステップ 7 は，学習した教材についてのディスカッションである．ディスカッションの質は，自己学習の量に直結する．複数の視点が存在する場合，議論の質は大幅に向上し，議論のモチベーションが高まり，より徹底した議論ができるようになる．ここでのディスカッションにより，ソース資料の誤りや弱点が明らかになり，修正されやすくなるので，メソッド全体がより強固なものになる．

表 2.1: PBL の 7 ステップメソッドの内容

Opening session – half an hour, in the group
Step 1: Examination of the case. The group gets familiar with the case material. Step 2: Identification of the problem. An initial title for the case is specified. Step 3: Brainstorming. The students present their associations and ideas about the problem to find out what is already known and how does the case relate to the previous knowledge. The ideas are said aloud and written on self-stick notes, which are organized on a white board. Step 4: Sketching of an explanatory model. An initial version of the explanation for the problem is constructed and most important concepts and their relations are identified. Step 5: Establishing the learning goals. Those parts of the explanatory model that are mysterious, fuzzy, or simply unknown are identified and the central ones are chosen as learning goals for the group.
Study period -- one week, each student working independently
Step 6: Independent studying. Each student independently studies to accomplish *all* learning goals. This phase includes information gathering and usually a substantial amount of reading (e.g., 50--150 pages).
Closing session -- one to two hours, in the group
Step 7: Discussion about learned material. Equipped with the newly acquired knowledge, the group reconvenes to discuss the case. The discussion includes *explanation* of central concepts and mechanisms, *analysis* of the material, and *evaluation* of its validity and importance.

[24] P.127 より引用

Nuutila らは，様々な種類の PBL の事例と他の学習の組み合わせをテストし，プログラミング入門教育における PBL のカリキュラムを実装する最良の組み合わせを検討した．そして，情報系コースの初年時の学生を対象とした Java によるオブジェクト指向プログラミングを学ぶクラスでアプリケーション開発を課題とした実践を行なった．

Nuutila らは，PBL セッションは，抽象概念を学ぶモチベーションを作り出す効果的な方法であり，問題解決とプログラミングの設計を学習する上で有益であると述べている．一方で，プログラミングスキルを習得するためには演習が必要であり，反復練習によりスキルを身につけるような分野では，PBL は必ずしも有効でないとしている．そのため，反復練習によりコーディングスキルを身につけさ

せるには，プログラミング演習を追加する必要があると述べている．

研究の成果として，PBL セッションは，従来型授業と比較してドロップアウト率の大幅な低下 (従来のプログラミングコースの 45%に対し 17%) と，プログラミングスキル獲得に加えて，グループワーク・共同作業・独立した学習・知識の外部化に関するジェネリックスキル獲得に有益であったと述べている．

著者は，PBL とプログラミング入門について，非情報系の学生を対象にした実践事例は探すことができなかった．しかし，Nuutila らの先行研究は，抽象概念を学ぶモチベーションを作り出す効果的な方法であること，問題解決とプログラミングの設計を学習する上で有益であるという結果から，うまく設計すれば，著者の提案する入門教育におけるカリキュラムへの反映も期待できる．

Nuutila らが行った 7 ステップメソッドは，我が国における標準の 90 分 15 回で構成される授業の枠に当てはめることは困難であるが，この考え方を応用し，Step1〜6 を各グループで取り組み，Setp7 を教室全体でのディスカッションと捉えることにより，本提案の PBL のカリキュラム構成の参考となる．あわせて，履修後の活動へつなげる項目として，ジェネリックスキルの向上も期待するところである．

多くのプログラミング入門では，一人で取り組む授業が多いが，プログラミングを学ぶ際に期待することは，学習後の活用であり，それは，ソフトウェアの開発などにつながる．プロジェクトとして設計し，ステップ 7 の学習した教材についてのディスカッションをうまく組み込むことができれば，効果的な授業設計が可能になると期待できる．ソフトウェア開発の工程に合わせて，仕様検討やレビューを組み込むカリキュラムが作成できれば，学んだ知識を活用し，ディスカッションの時間を効果的に作ることができるであろう．これにより，実用的な学びの場を作ることができると著者は考える．

2.4 本章のまとめ

本章では，本論文で取り上げた高等教育における一般情報教育におけるプログラミング教育と PBL を取り上げ，関連研究・先行研究を示した．

2.2 節では，高等教育におけるプログラミング教育について取り上げた．これまで，高等教育におけるプログラミング教育は，主に専門分野での科目であった．しかし，世の中の急速な変化とともに情報教育は，基礎教養に欠かせない教育となり，プログラミング教育は，一般情報教育の一部となってきている．

大岩は，一般情報教育の教育目標の一つとして,「「知識」と「情報」を資産とする情報化社会において，情報の価値を知るとともに，これを資産として使いこなし生きるための対応力を習得させる．[49]」と述べている．

これは，1991 年に示されたものであるが，著者の所属する環境では，残念ながら今もまだこの目標に至れておらず，専攻や学校の環境などによる格差を感じる．情報機器が身近なものとなり，誰もが容易にパソコンやスマートフォンなどの情

報通信機器を持てる時代となり，プログラミング教育が義務教育となった今，知識と情報を活用する教育を広く浸透させるべき時がきたと考える．そのためには，誰もが教えやすく学びやすいカリキュラムや教材が必要である．

しかし，2.2.1 節で示したように，高等教育における全国的な調査結果からは，「アルゴリズムとプログラミング」は,「コンピュータリテラシー」,「情報倫理とセキュリティ」など他の項目と比較して「教えていない」学校が多くなっており，高等教育においては，誰もが学ぶ環境は整っていない．さらに，教える教員については，一般情報処理にかかわる教員数は情報系以外の分野が圧倒的に多く担当している．このような状況から，誰にでも教えやすく学びやすい教材が求められるであろう．そして，教員・学生の時間的な負担も少なく，社会の要請に応える形で，学ぶ機会を提供するべきである．

カリキュラムについては，30 年以上前から，体系化されたカリキュラムモデルが示されてはいるが，著者の所属する学校を含め，浸透していない学校もある．さらに，2 単位を拡大して 4 単位のモデルが示され，プログラミングは，その中の一部での実施となっている．ただし，全国調査の結果からは，どこの大学も時間の確保は困難であることがわかった．そのため，最小の単位時間の 2 単位であれば，時間の確保ができ実現できそうではないかと考える．

2.2.2 節では，実践事例を取り上げて示した．図形を用いた描画教材は，これまでにも多くの学校で活用されており，視覚的に分かりやすく効果があることがわかっている．

プログラミング入門の事例では，15 回を使って行っているものもあるが，15 回の授業のうちの 3〜5 回をそれに当てているものが多くあった．著者の経験からも，基本的な事項は，３〜5 回で実施できていた．しかし，プログラミングの知識を学ぶことに閉じていて，学んだ知識を活用するような内容にはなっていない．これからますます発展するであろう情報化社会で生きていくためにも，新しい知識の学び方や活用の仕方を体験的に学び，次のステップの学習に繋げられるようなカリキュラムを構築すべきである．

しかし，チームで取り組むプロジェクトを含むものはなく，これからは，チームで取り組むプロジェクトを積極的に導入していくことが有益であると考える．

新たな未来を築くための大学教育の質的転換が求められるようになり，アクティブラーニングや PBL への転換が求められるようになった．2.3 節では，これらについて取り上げた．

PBL の活用は，さまざまな分野で取り入れられるようになってはきているが，プログラミングに関するものについては，主に専門教育での活用が多く，専門外でのプログラミング入門における PBL の事例は見つけることができなかった．

PBL においては，これまで学んだ知識や技術を活用して行われる．そのため，入門レベルの科目では，知識の獲得のために学ぶことに重点が置かれ，プロジェクトにまで至ることが難しいことが要因のひとつにある．しかし，著者は，学んだ知識や技術のまとまりを小さくし，獲得した知識をもとにプロジェクトを構成す

ることができれば，入門レベルのカリキュラムであってもPBLを導入できると考える．

2.3.3節で取り上げた，駒谷が示した図2.3は，ソフトウェア開発人材育成に対するものであったが，これからの情報化社会における市民のスキルを考えると，情報システムとの関わりが身近になり，ある程度の概要と基礎知識を誰もが身につけておく必要があると考え，カリキュラムの設計に有益な資料となった．

著者が考えるこれからのプログラミング教育では，専門家と非専門家が交わってそれぞれの知を分かち合い共同して活動する社会に貢献できる教育であるべきであると考える．そのための方法として，プログラミング入門にPBLを組み合わせることは新たな試みであり，90分15回2単位で展開するカリキュラムは，実践事例はなく，上手く設計して実現できれば，学習者にとって有効に働くと考える．

第3章

プロジェクトによるプログラミング入門カリキュラムの構築

3.1 はじめに

情報技術 (IT) は，今ではあらゆる領域で活用され，人々の暮らしの中にまで入り込み，無くてはならない存在となった．IT の利活用により豊かな暮らしを実現するために，プログラミングの知識は，専門に限らず，誰もが社会で求められるようになった．

高等教育においては，プログラミング入門の内容を含む必須カリキュラムの実施には至っていないことを受け，著者は，高等教育における学習の機会を補うだけでなく，情報化社会における教養として全ての人に求められる内容を包括したプログラミング入門のカリキュラムが必要であると考える．

プログラミングの学習を通じて，最低限学習するべきであると期待される内容は，言語を学びコードが書けるようになることではなく，プログラミングの概念を学び，プログラムがどのように作られ，どのように動いているかの原理を理解することである．また，プログラムをどのように活用するかといった点において，ソフトウェア開発プロジェクトに関する知識も含めて学ぶ必要がある．

なぜなら，コンピュータに依存する現代社会における活動を考慮すると，専門家のみならず誰もがソフトウェア開発の基礎的知識を持ち，専門家と一緒にチームを組み，どのようなソフトウェアを作るかを適切に判断したり，スケジュールに合わせて開発を進めソフトウェアを完成させたりする能力が求められているからである．

そのため，高度な IT 人材の育成のみならず，誰もが基礎知識を持ち活動できるよう，人材育成の底上げを図る必要があると考える．

英国内閣府の Potter は，次のように述べている [35].

> 技術そのものの導入が重要なのではなく，公共サービスにおける職員の業務がどのようなものなのか，ユーザーがそれをどう使っているのか，そこに発生している課題は何なのかを知ることが重要です．そこにどうやって技術を使い改善していくかを見極める力が必要なスキルだと考えています．

Potter は，技術者の面から求められるスキルについて述べているが，著者は，発注者や利用者の立場にあっても，技術者と一緒に仕事をしていく上で，情報技術の基礎的知識は必要であると考える．情報化社会においては，多くの課題解決には技術だけで解決できる問題は少ない．そのため，誰もが情報技術に関する基礎的な知識を持つことにより，様々な立場の人が関わり技術者と交わり円滑なコミュニケーションを図ることができるようになることが期待でき，より良い社会を築くことができると考える．

社会に出る以前に，ソフトウェア開発プロジェクトに関する知識を含めたカリキュラムで学ぶことにより，チームで協力して何かを作る経験ができ，プログラミングの基礎知識に加え，「問題認識から課題解決までを主体的に学ぶ」ことの理解が進むと考える．

Weinberg も，『プログラミングの心理学』[34] で，“社会活動としてのプログラミング”について，プログラマは孤立して働いたりしないことやチームでの効率について述べている．

しかし，既存の多くのプログラミング入門の授業は，プログラミングの技能を学ぶことを目的としていて，それを使ってチームで何かをするという内容にはなっていない．情報系の専攻であれば，入門科目の後，情報系の実践科目等様々な科目を通じて，ソフトウェア開発的な考え方を学び実践する機会が提供されるであろう．しかし，それ以外の専攻，とくに文系の専攻では，プログラミングの内容を含む科目はあっても，ソフトウェア開発の体験を含みチームでプロジェクトを行うような学習の機会は無いままに終わってしまう．

そのため，専攻によらず誰もがプログラミングの概念を学び，その原理を理解することに加え，“プログラムを活用するためのソフトウェア開発プロジェクトを理解することを含む”という要求を満たすことはできていない．初等中等教育におけるプログラミング教育が体系化されたが，チームでソフトウェア開発を伴って学ぶ内容は学習指導要領にも含まれていないないことから，高等教育においては，このような学習機会を設けて取り組む意義があると考える．

そこで，第 1 章では，リサーチクエスチョンを 2 つ示し，仮説を以下のように示した．

初等中等教育からの連続性と社会への接続に配慮し，高等教育における一般情報教育では，プログラミング入門とソフトウェア開発プロジェクトを組み合わせたカリキュラムでの実施が有効である．

本章では，この仮説を検証するための準備として，プログラミング入門に PBL を組み合わせた 1 科目のみで構成する，誰もが学ぶべき入門レベルのカリキュラムを設計する．

以下では，カリキュラムの設計方針を示し，カリキュラムを構築する．

3.2　カリキュラムの提案

2003 年に高等学校に情報科が設置され，2020 年には，小学校でのプログラミング教育が始まった．学習指導要領の改訂により，小学校から高等学校まで体系的に積み上げる内容となっている．一連の初等中等教育の改革を受け，高等教育におけるカリキュラムを検討する必要があり，高等教育においては，その先の社会での活動を意識したカリキュラムを検討する必要がある．

高等教育において，プログラミングの学習を通じて期待されることは，就職後の活動を考えると，プログラムを書くことだけでなく，プロジェクトへの関わりが大事な要素であるというのが，著者の考えである．なぜなら，ソフトウェア開発においては，どのようなシステムを作るかを適切に判断したり，スケジュールに合わせて開発を進めシステムを完成させたり，役割や立場を判断して行動することも重要であると考えるからである．

しかし，既存の多くのプログラミング入門の授業は，プログラミングの技能を学ぶことを目的としていて，それを使って何かをするという経験を得るまでに至らない．それは，プログラミングの知識を身につけるということ自体が多くの労力と時間を必要とする学習内容であるためである．情報系の専攻であれば，プログラミングに関する入門科目の後，情報系の実践科目等さまざまな科目を通じて，ソフトウェア開発的な考え方を学び実践する機会が提供されると期待できる．しかし，それ以外の専攻，とくに文系の専攻では，時間の制約もあり，プログラミングを題材とした充分な科目が用意されておらず，上述の重要な点を学ぶ機会は無いままに終わってしまう．

初等中等教育においても，学習指導要領の改定により，プログラミング教育の実施が進められているが，チームによるソフトウェア開発プロジェクトに関する内容は含まれていない．

高等教育におけるプログラミング入門科目では，これらの問題を克服し，「プログラミングを初めて学ぶ」ことと「ソフトウェア開発プロジェクトにおいて重要なことを実践を通じて学ぶ」ことの両方を達成するため，両者を組み合わせた新たな授業カリキュラムが必要だと考え，これらを両立させるカリキュラムを提案する．

3.3　カリキュラムの設計方針

提案するカリキュラムは，大学における一般教養に相当するものであり，多様な学生を対象とすることになる．高等学校までの学習経験により，特定の教科の向き不向き，できるできないなど，個々のスキルや知識は様々であり，それらの多様な学生がそれぞれに満足できるような配慮が必要である．得意・不得意に関わらず誰もが学ぶためには，多くの時間を割くようなカリキュラムでは浸透しない．そのため，誰もが学び易くなるような教材や時間配分を考える必要がある．

また，前述した調査結果 [122] の教える教員の不足や専門性の課題に応えるためには，誰にでも教え易い教材やカリキュラムへの配慮もすべきである．

複数科目で構成されるようなカリキュラムは，実現性が薄いと考え，1 科目のみで実現できるように設計することが望ましいと考える．そのため，高等教育における最小単位である，半期 90 分 15 回 2 単位に収めるべきである．

そして,「情報化社会における教養として，すべての人に求められる内容を包括した高等教育におけるプログラミング入門のカリキュラムを構築すること」である．社会に出てから役に立つような形で「プログラミングを初めて学ぶ」環境を作ることを念頭におく．

対象とする学生は，プログラミングを初めて学ぶ学生である．しかし，初等中等教育局でのプログラミング教育が普及した先を見据え，その後も継続して意義のあるカリキュラムとなるよう留意するべきである．従来のプログラミング入門のカリキュラムでは，プログラミングの技能を学ぶことを目的としており，様々な立場の人が関わり，協力・分担してより良いものを築きあげていくという経験をするようにはなっていないことが多い．そのため，従来行われてきたような技能を学ぶプログラミング入門の内容に加えて，初等中等教育では取り上げれていないチームで取り組む「ソフトウェア開発プロジェクト」の要素をカリキュラムに加える．

「ソフトウェア開発プロジェクト」を含める意義は,「ソフトウェア開発プロジェクト」の基礎知識を持つことにより次のことを期待するからである．

- 技術者と一緒に仕事をする上で，コミュニケーションが上手く取れる
- 社会活動において，チームで協力して取り組む必要があり，チーム力の育成ができる
- 「問題認識から課題解決までを主体的に学ぶ」という経験の場を作ることができる

プログラミング入門としての基礎的な内容をおさえながら，入門レベルであっても，ソフトウェア開発プロジェクトの知識を含めたカリキュラム設計をすることにより，これからの社会で役に立つような知識・技能の習得の入り口となることを期待する．

そして，高等教育のカリキュラムとして，学習成果に関する指針として 4 分類からなる「学士力」[119] の向上が図れるよう配慮することも忘れてはならない項目である．特に，コミュニケーションスキルやチームワークは，個々での学びからの習得は難しく，チームで取り組むことにより育成することができる．情報化社会においては，利用者の立場にあっても技術者と一緒に協力関係を持って取り組めるような知識・技能が求められていることから，知識を学ぶだけでなく，それを活用する視点を持ち，技術者の立場や，利用者の立場なども考慮して学ぶ環境をつくることを考える．

3.4　シラバスに掲げる目的・到達目標

カリキュラムの設計方針に従い，カリキュラムを構築する際に，カリキュラムの実施に向けて，シラバスに掲げる目的と到達目標を検討する．

本研究で提案するカリキュラムの目的は，社会に出てから役に立つような形で「プログラミングを初めて学ぶ」ことである．これを実現するために,「ソフトウェア開発プロジェクトにおいて重要なことを実践を通じて学ぶ」ことをカリキュラムに加える提案をすることから，シラバスに掲げる目的を，次のように定める．

> 目的：
> 現代は，情報社会である．身の回りには，情報技術によって実現されたサービスが氾濫し，個人的利用からビジネス等での活用まで不可欠であり，情報技術を用いないサービスを見つけることは困難である．特に，これらを構成するプログラミングの基礎知識は，誰もが必要とされる知識のひとつである．
>
> 「プログラミング」の知識は,「単に言語の文法を学ぶ」「例題をそのまま打ち込んで動かす」だけでは明らかに，社会に出て役立つという要件は満たせない．プログラミングを伴う活動には,「ソフトウェア開発プロジェクト」がある．これからの社会においては，ソフトウェア開発に直接関わる関わらないに関係なく，誰もがある程度の基礎的な知識を持っていることが期待される．そのため，単にプログラミングを学ぶのではなく，プロジェクトもあわせて学ぶことが重要である．
>
> 本授業では，初めてプログラミングを学ぶ人を対象に,「社会に出てから役に立つような形でプログラミングを学ぶ」ことを目的とする．

到達目標は，プログラミング入門に加えてソフトウェア開発プロジェクトの要素を加えることから,「ソフトウェア開発プロジェクト」に係わる能力について考える．

表 3.1: 産業界 (IT 企業) と教育界が重視する教育項目の比較

順位	項目	企業 (%)	教育 (%)
1	コミュニケーション能力	61.2	59.4
2	問題解決能力	42.6	29.3
3	文章力・文書作成能力	30.7	20.1
4	チームワーク・協調性	25.5	24.7
5	プログラミング技術	22.2	19.2
6	要求分析	12.1	3.3
7	プロジェクトマネジメント	11.7	8.8

しかし，情報を専門としない専攻での先行事例がないことから，IT 人材白書 2013 年版 [81] の産業界 (IT 企業) が教育機関に重視して欲しい教育項目と教育機関が重視している教育項目を比較したデータを参考に検討する．このデータは，企業 564 社，教育界 239 機関からの回答 (選択肢は最大 3 つまで選択可能) によるものである．

まず，表 3.1 に，産業界が重視する項目の順位を基準に，上位項目を著者がまとめたものを示す．1〜4 位は，汎用的能力であり，産業界も教育機関も同様に上位に挙げている．一方で，産業界では 6，7 位にある「要求分析」,「プロジェクトマネジメント」は，ソフトウェア開発に限らず仕事をする上で重要なキーワードとなるが，教育機関では重視する割合が低い．これは，駒谷が指摘した産学の意識の乖離 [68] に対応している．本提案においては，学習後の活動にも有効に機能するようにカリキュラムを設計する必要があることから，産業界で求められている項目に注目する．

また,「要求分析」については，著者が社会活動をする中で，企業の方から専攻に関係なく大学時代に学生に学ばせて欲しい項目であると要望を受けた．情報通信技術が発展し，情報通信機器がより身近な生活の中に入ってくるにつれ，社会システムの開発は，技術者だけで対応できなくなり，様々な人がチームを組み仕事をするようになってきている．そのため，専攻に関係なく新卒人材の採用がされている．この要望は，多様な人材が企業で求められていることがひとつの要因としてある．

本提案のカリキュラムでは，入門レベルであること，および卒業後の社会へ接続することを意識して到達目標を設定する．また，従来の情報リテラシー科目で取り上げられてきた IT スキルに関する要素を加えて，到達目標には，以下の 5 項目を設定する．

到達目標：

(1) プログラミング

- プログラムを記述・実行するための手順を理解し，プログラ

ムを実行することができる．

- プログラム中の誤りを発見し，修正することができる．
- 制御構造・抽象化について，説明することができる．
- 自ら考えたことをプログラムとして記述し，表現することができる．

(2) プロジェクトマネジメント

- チームで決めた計画 (スケジュール) に沿って自分の作業を行うことができる．
- 自分の役割を理解し，チームの仲間と協力して取り組むことができる．
- 問題解決とそのための思考過程を体験し，問題認識から課題解決までを主体的に取り組むことができる．

(3) 要求分析

- 目的にかなった開発のために「何が重要か」「どのような機能を実装するか」を理解し，創意・工夫することができる．
- 相手の要求を隅々まで聞き出すこと，顧客にもチーム内にも正確に伝えること (コミュニケーション力，記述力) について理解し，適用することができる．

(4) チームメンバーの多様性

- チームメンバーの多様性を理解し，個々の特性に合わせた役割分担を行うことができる．
- 個性やそれぞれが持つ知識が異なる集団でプロジェクトを進める際に，個人を理解し認め，尊重し，各自の得意不得意を理解してプロジェクトを行うことができる．

(5) IT スキル

- プログラムの記述・実行に必要なツールを操作することができる．
- 課題やレポート作成，発表に必要なツールを適切に活用することができる．
- メールや LMS などを用い，チームのメンバーとコミュニケーションすることができる．

到達目標 (1) は，プログラミングの入門レベルであること，向き不向きによらず，誰もが学ぶ点を配慮し,「自分で思ったようにコードを書く」力を養うことを基準とした項目を設定する．

到達目標 (2)〜(4) の 3 項目は，チームによるプロジェクトの要素と他者との関係性を考えた内容とする．社会におけるプログラミングの活用を意識させた体験から，コミュニケーション，問題解決，文章・文書作成，チームワーク・協調性などのジェネリックスキルの向上も含めて，学習効果を期待するところである．

到達目標 (5) は，15 回の授業を通じてコードの編集やデバックだけでなく，課題やレポートの作成，グループワークのための情報共有でメールやその他のツールを使う場面がたくさんある．操作に陥りがちな情報教育を脱し，実践の中で様々なシステムやツールを活用することによりスキルアップをはかることを期待して設定する．

3.5 カリキュラムの構築

3.5.1 全体の概要：プログラミング入門とプロジェクト

設計方針で示したように,「プログラミングを初めて学ぶ」ことと,「ソフトウェア開発プロジェクトにおいて重要なことを実践を通じて学ぶ」ことの両方を達成するために，両者を組み合わせた構成を考える．

初学者の学生たちに，何も知らない状態からいきなりチームでプログラムを作らせることは難しい[1]．

先行研究では，プログラミング入門は 4 コマあれば実施できていたこと [100]，著者のこれまでの経験 [45]，および，[24] で示されている反復練習の必要性から，本カリキュラムでは，授業構成の前半 (以下，プログラミングパートと記す) を，個々にプログラミングの知識を学び・習得する時間として 6 回を設定する．後半のプロジェクトにつながるプログラミングの基礎知識を精選した上で，実習中心で個々に取り組ませる．

後半 (以下，プロジェクトパートと記す) は，グループワークでソフトウェア開発プロジェクトを実施する構成を考え [31][43]，ソフトウェア開発の工程を参考にして 9 回を設定する．

授業の構成とその流れを，図 3.1 に示す．

この構成をすることにより，プログラミングパートで充分に身につけることができなかったプログラミングの基礎知識は，後半のプロジェクトパートで，繰り返し取り組むことができ，知識の定着につながると考えた．プロジェクトを通じて，チームで助け合いながら体験的に身につけることを期待する．

[1] プログラミングの素養を持つ学生が一定の割合でいたり，多数の TA がついたりなど，ファシリテータ役を勤められるような人材がいる環境があればこの限りではないが，そのような恵まれた状況はあまりない．

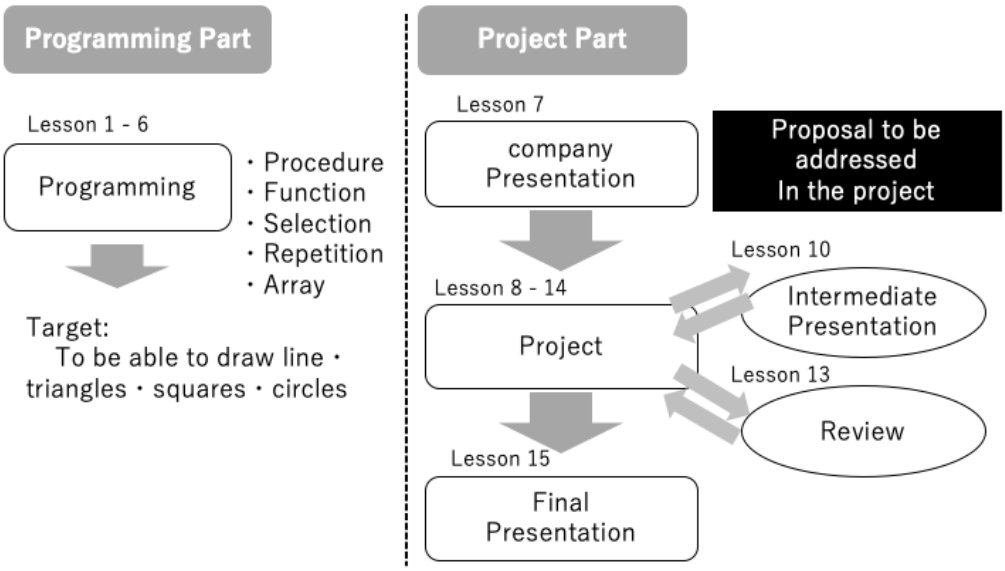


図 3.1: 授業の構成

3.5.2　カリキュラム構築における工夫

本提案では，90 分 15 回の内容を構築するだけでなく，テキスト，演習課題，ワークシート，授業時のアンケート，発表の場の設定など，授業の進め方も含めて検討する．

多様な履修者を想定し，プログラミングパートでは，個々の能力に応じた課題に取り組めるよう授業時の課題・宿題の取り入れ方を工夫する．プロジェクトパートでは，グループワークでの進捗管理，役割の明確化のためのワークシートを用いる (3.5.6 節で，詳しく述べる．)．各回，授業時にアンケートを行うことで，理解度の把握を行うだけでなく，学習者への気づきを与えるきっかけとなるよう工夫する．

最終課題には，リアルなものづくりを取り入れる．これにより，チームでのプロジェクトのゴールが明確化できるだけでなく，チームで活動することにより，他者の理解やチーム力の育成につながると考える．

さらに，授業を通じてオンラインシステムを活用することにより，作業の記録と共有，IT スキルの共有を期待する．これに加えて，2 回目の授業後と 15 回目の授業後に，教育の成果や影響を分析するために，コンピテンシー評価のチェック (3.6 節で，詳しく述べる．) を行う．

表 3.2 に，課題を含めた 15 回の構成を示す．授業の内容は，内容欄に記述し，履修者が取り組む課題について，課題欄に示した．各パートの内容の詳細は，以降の節で説明する．

表 3.2: 15 回のカリキュラム内容

回数	項目	内容	課題
1 回目	プログラミングの導入	プログラムとは何であるか理解し，簡単なプログラムを動かせるようになる.	課題 A/課題 B
2 回目	分岐と繰り返し	基本的な制御構造を理解し，これらを使ったプログラムが書けるようになる.	課題 A/課題 B, コンピテンシーチェック
3 回目	制御構造と配列	制御構造の組み合わせについてとデータ構造と配列を理解する.	課題 A/課題 B
4 回目	手続き･関数と抽象化	手続き (メソッド) による抽象化を理解する.	課題 A/課題 B
5 回目	2次元配列と画像	2 次元配列・レコード型と画像の表現を理解し，様々な形を描画する.	課題 A/課題 B
6 回目	プログラミングまとめ	自分が生成したいと思う画像のために何が必要かを考え，画像の内容に合わせてプログラム構造を実現する.	中間課題
7 回目	グループ作成･課題とゴール設定	プロジェクトの課題説明とグループ作りおよび役割分担を決定する.	ワークシート
8 回目	アイデア出し･ブレスト･仕様決め	ソフトウェア開発について説明し，デザイン決定に向けてのアイデアを話し合う.	ワークシート
9 回目	デザイン・設計・分担・制作	作品のデザイン決定と各パーツの分担を決める.	ワークシート
10 回目	試作･中間ドキュメント作成	パーツごとの試作および中間発表に向けてのドキュメントを作成する.	ワークシート
11 回目	中間発表･仕様見直し	中間発表のコメントを受け，仕様の見直しを行う.	ワークシート
12 回目	制作･単体テスト	単体テスト，統合テスト，コードレビューなどの概念について理解する．仕様の見直しを受けての修正と単体テストを行う	ワークシート
13 回目	コードレビュー	コードレビューについて理解する．試作したパーツについてのレビューを行う.	ワークシート
14 回目	制作･統合テスト	各自で作成したパーツを統合し，作品を完成させる.	ワークシート
15 回目	最終成果物 (まとめ)	グループの成果について発表し，完成した作品の評価を行う.	最終レポート, コンピテンシーチェック

3.5.3　教材：描画教材

これまで，描画教材は，入門レベルの教育に多く利用されてきた．本提案においても描画教材を活用する．

描画教材には，事前に準備した (1) 各種図形 (丸，三角，四角，楕円) を描くメソッドと (2) 生成した絵をファイルに出力するメソッド (図 5.1) を用いる．

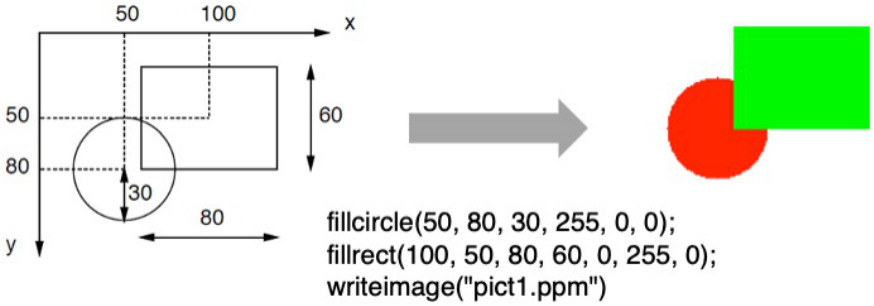


図 3.2: 図形描画の例

例えば，円を描くメソッドでは，中心座標，半径の大きさ，色の RGB 値 (0〜255) を引数として指定することで円を描くことができる．

入門レベルのため，複雑な構造は作れないことから，必要な引数の部分を考えさせることにより，データ構造と抽象化を理解させることができると考えた．初回は，教材として与え，メソッドはブラックボックスとして活用するが，回を重ねるごとにその仕組みを紐解いて教えることにより，メソッド作成の理解が深まると考える．

描画教材は，絵のパーツを組み合わせることにより，描きたい絵のデザインと設計の関係を視覚的に捉えることができる．これにより，描きたい絵をさまざまな図形で分割して構成することにより抽象化し，さらに，いくつかのパーツに分けて考えることができる．そして，その方法は，１通りではなくさまざまな考え方ができる．これにより，多面的・多角的に考える力の育成を期待する．

ソフトウェア開発では，1 人で対応できるような規模の開発はなく，複数の人が分担して関わり合って完成する．１つの絵をグループで分担して完成させる工程をソフトウェア開発の工程と見立てることにより，ソフトウェア開発の専門知識がなくても，擬似化した環境を作ることができ，後半のグループワークへの応用にも効果的に働くと考える．

3.5.4　プログラミング入門の進め方

本授業は，初めてプログラミングを学ぶ学生が対象であり，後半のプロジェクトパートまでに「自分で思ったようにコードを書く」力を養う必要がある．プロ

グラミングを学ぶ際の指針として，以下にあげる久野のまとめた指針 [60] を参考にする．

久野の指針：

- プログラミング言語の文法や機能を逐一，順番に説明していくのではなく，初回から数行のプログラムを書く
- 例題を選んで丁寧に説明し，試験をするような暗記中心の授業構成にはしない
- 例題をその通りに打ち込んで実行させることに注力するのではなく，自分のレベルにあった問題で取り組ませる

コードを書く際には，実行時のエラー処理をはじめ，初学者が一人で学習する際には困難を伴うが，それを軽減させるために，ペアプログラミング [20][21][94][95] が有効であると考える．特に，TA 不在の授業運営では，演習時の対応を教員一人で賄わなければならず，その対応で授業時間の浪費が発生しかねない．ペアで取り組むことにより，学生同士での解決が期待できるだけでなく，1 人→ 2 人→ 4 人へとプロジェクトパートを意識し，緩やかに牽引できると考える．

各回は,「説明→演習→時間内の課題 (課題 A) →次回までの課題 (課題 B)」で構成する．課題は，前述した久野の指針を参考に，非常に平易なもの (例題を少し手直しすれば提出できる) から高度なものまで用意し，学生は自分のレベルに応じて選んで解答する．これにより，レベルに応じた学びを得ることができ,「例題をコピーして動かす」のではない，自分なりのコードを書くことが可能となることを期待する．最低限，自分で考えコードを書き動かすことを念頭に置き，プログラムを記述・実行するための手段を理解し，プログラムを実行することができることを目標とし，プロジェクトパートにつなげることを考える．

各課題には，3〜4 問で構成されたアンケート (授業や課題の感想，今後の展望など) をつけ，理解の進捗や授業の感想などを求める．これにより，授業ないでは拾いきれない学生の進捗を把握し，理解が芳しくないところについては，翌週の授業でフォローすることができると考える．

また，プログラミングパートで十分に理解できなかった内容は，プロジェクトパートで繰り返し学習することで，学び直しができ，知識の定着が促進させることを期待する．プログラミングパートの 6 回の内容を 5.1 に示す．

1 回目は，初めてプログラミングに触れることを想定して，プログラムとは何であるかを理解し，簡単なプログラムを動かせるようになることを目指す．2 回目以降，プログラミングの基本となる制御構造について取り上げ，後半のプロジェクトにつながるプログラミングの基礎知識を学ぶ．

表 3.3: プログラミングパートのカリキュラム内容

回数	項目	内容	課題
1 回目	プログラミングの導入	プログラムとは何であるか理解し，簡単なプログラムを動かせるようになる.	課題 A/課題 B
2 回目	分岐と繰り返し	基本的な制御構造を理解し，これらを使ったプログラムが書けるようになる.	課題 A/課題 B, コンピテンシーチェック
3 回目	制御構造と配列	制御構造の組み合わせについてとデータ構造と配列を理解する.	課題 A/課題 B
4 回目	手続き･関数と抽象化	手続き (メソッド) による抽象化を理解する.	課題 A/課題 B
5 回目	2次元配列と画像	2 次元配列･レコード型と画像の表現を理解し，様々な形を描画する.	課題 A/課題 B
6 回目	プログラミングまとめ	自分が生成したいと思う画像のために何が必要かを考え，画像の内容に合わせてプログラム構造を実現する.	中間課題

3.5.5 プロジェクトの進め方

プロジェクトパートでの目的は，グループワークにより，実践的にプログラミングの基礎知識を定着させるとともに，ソフトウェア開発プロジェクトを理解し，チームのメンバーと共に，プロジェクトの運用能力を身につけることである．プログラミングパートで用いた描画教材を継続して利用し，ソフトウェア開発プロジェクトを基としたPBL を取り入れる.

3〜4 名で 1 チームを構成し，本格的なソフトウェアの開発は難しいことから，1 つの絵をソフトウェアと見立てて，チームで 1 つの絵を描くプロジェクトに取り組ませる.

1 つの絵を各パーツに分け，メンバーでそれぞれ役割を分担をしてパーツを作成し，それを統合して 1 つの絵に完成させる．図 C.2 に，チームで描く絵の分担の例を示す.

1 回のプロジェクトで全てを学ぶことは難しいが，プロジェクトマネジメントを意識し，期限内にプロジェクトを完了できるよう経験的に学ぶ環境を構築し，ソフトウェア開発プロジェクトの工程に倣って，表 3.2 に示した 9 回で構成する．これにより，前半で身につけたプログラミングの知識の定着と活用を目指す.

本授業設計では，全員が同じゴールに向かってグループワークを進めることから，チーム内のみならず，チーム間の教室全体で，同じ視点でディスカッションできるメリットがあると考える.

3 回の発表の機会を設けることにより，チーム内の情報共有のみならず，教室内での情報共有を図ることができ，ソフトウェア開発の工程の理解だけでなくモチベーションの維持につながると考える．“グループワーク→発表”のサイクルを 3 回 (7 → 11，12 → 13，14 → 15 回) 繰り返すことで，各チームでは，“制作 (創造) →共

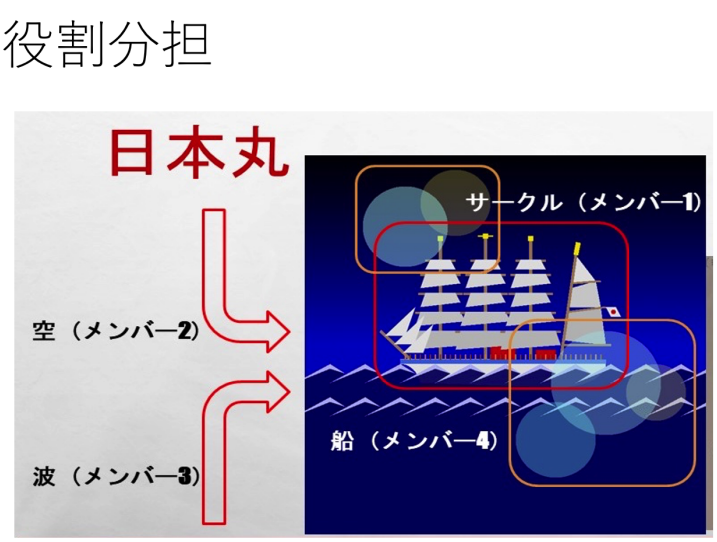


図 3.3: チームで描く 1 つの絵の例

有→振り返り”のスパイラルが生まれ，教室内では，“共有→共感→競争”という流れが生まれることを期待する (図 3.4).

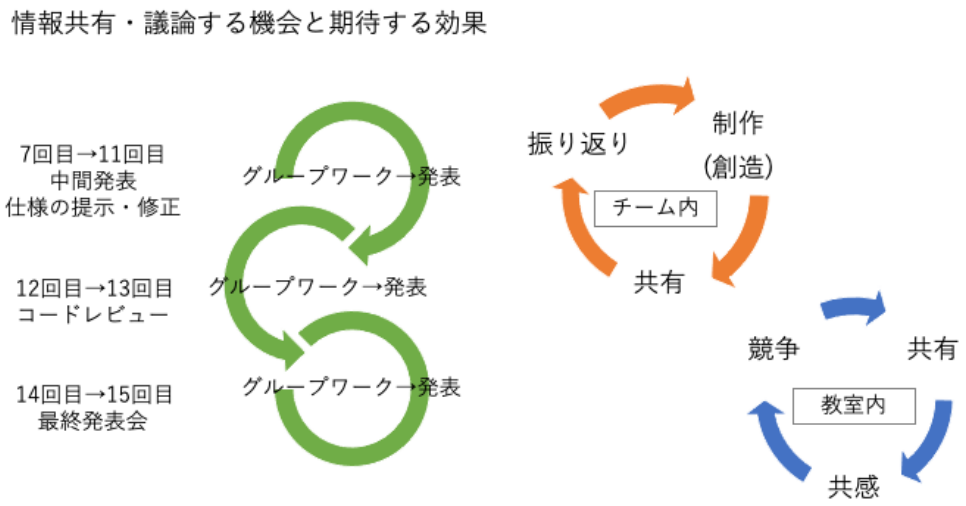


図 3.4: グループワークと発表のサイクルと期待する効果

また，多くの授業では，与えられた問題を解き，出来た・出来ないで終わってしまうことが多い．数学を例に取ると，対象とする層は，これまで与えられた問題を解くことだけにとどまり，できなかった問題を再考したり，別解を考えたりする経験が乏しい．しかし，社会では，どこが問題であるかを考えたり，不具合を見つけて修正したりすることが重要な要素となる．“考えをまとめる→試作する→再考する→作品を完成させる”というプロセスの中で，ソフトウェア開発の工程

について理解を深め，前半で不十分であったプログラミングのスキルを補うことができると考える．さらに，図形描画を用いることにより，設計，コードの実行時のエラー，出力結果が視覚的にわかりやすくなると考え，このプロセスにより思考力や課題解決力の育成につながると考える．

3.5.6 PBL の欠点の克服とワークシート

プロジェクトを行う際には，さまざまな問題がある．限られた制約のある時間の中で，進めていくためには，この問題を克服し，うまく進められるような工夫が求められる．

著者の経験や，先行研究からの課題からの PBL の欠点 [41][68][97] を以下にまとめる．

1. 学生も教員も多くの時間を要する
2. ファシリテータのレベルに応じてアウトプットの質に差が生じる
3. 分担による学生の活動量に差が生じる
4. メンバーの能力やスキル差によりアウトプットの質に差が生じる

これらの欠点を克服し，カリキュラムを実現するための案を以下に示す．

欠点 1 については，絵を生成する教材を用いた課題のみで完結するようにカリキュラムを設計するという本論文の提案により，学生・教員に要求する時間を自ずと制限することとなり，解決策となると考える．著者がこれまで取り組んできたプロジェクトにおいても，制限を設けず自由に取り組ませたプロジェクトでは，膨大な時間がかかっていた．しかし，ある程度教材を制限し，課題設定することにより，時間的な問題は解決できた経験から，教材によりコントロールできると考える．基本となる骨格を残し，教材を精選することにより，解決を図ることができると考える．

欠点 2 については，多様なプロジェクトを扱うのでなく，画像を生成するという枠組みに限定することで，教材や学習内容を予め準備することによりチーム毎の進捗に大きな差が生じないと考える．また，ワークシートを活用することにより均一なファシリテートを期待する．さらに，教員がファシリテータの役割をすることにより，ファシリテータのスキルへの依存を相対的に軽減することができると考える．

欠点 3 および 4 については，チーム内での分担の明確化とチーム内・チーム間での情報共有が有効であると考える．著者は，これまでに，単にグループで取り組むだけで，役割が明確化されていないグループワークを見聞きしてきた．このような状況を回避するために，本提案では，最初に話合いで担当を決めた後，どの部分が誰の担当か，それがどれだけ進捗しているのかを誰の目にも明らかなように常に維持するため，ワークシートの活用 (役割分担の明記) と発表の場の活用 (11・13 回目) が有効であると考える．これにより，自分の分担について可視化し，言い

逃れできなくするだけでなく，遅れていればチーム全体の問題としてメンバーが手助けすることで，チーム内での学びが起こり，スキルの劣ったメンバーが学んで追い付くことを促すことにつながると期待する．

プロジェクトパートで用いるワークシートは，グループ用と個人用の 2 種類のワークシートを用意する．チーム用のワークシートには，各回の活動の記録をまとめ，チーム内で進捗状況を共有する．これにより，プロジェクトの進捗の明確化だけでなく，チーム内での役割の明確化を測ることができ，各自の行動に責任を持たせることができると考える．個人用のワークシートには，授業のまとめと翌週の個人の課題・チームの課題，授業の感想を毎回共通で書く．また，学生の活動状況とプロジェクトの理解を確認するため，到達目標に基づき該当する項目についての質問も加える．これにより，単に物を作り上げるのではなく，プロジェクトの進め方への理解が深まったり，気づきが与えられたりする効果があると考える．

各回の異なる確認項目の内容については，表 3.4 に示す．各項目の () 内の数字は，到達目標の番号と同様である．(5) では，特に情報共有についての内容を取り上げる．

3.6　コンピテンシー評価の設計

教育の成果や影響を分析する上で広く用いられている評価指標として，コンピテンシー評価がある．

RQ1 に掲げた「PBL により，プログラミングの概念の理解と社会で求められる資質・能力の育成をはかることができるか？」の後半部分,「社会で求められる資質・能力の育成をはかることができるか？」について調べるために，コンピテンシー評価を活用することを考える．

独立行政法人情報処理推進機構は，大学の出口と企業の入口における人材のマッチングを円滑するため大学と企業のシームレスな評価モデルとして活用することを目的とし,『コンピテンシー評価項目表 (参照モデル)』[80] を提供している．大学全体，学部全体などで実際に運用されている先行的な実施例や，[80] の WG メンバーの企業各社における実例・意見をもとに，6 分類 17 項目が設定された．また，これらの項目は，一般的なコンピテンシー項目として挙がる項目である．加えて，経済産業省から提唱された概念である『社会人基礎力』[62] で定義されている「3 つの能力 12 の要素」も参考にされている．

著者は，カリキュラムの目的や到達目標に，コミュニケーション，問題解決，文章・文書作成，チームワーク・協調性などの項目を設定した．この背景から参照モデルの策定の経緯に賛同し，コンピテンシー評価には，参照モデルを活用することを採用する．カリキュラムに該当しない，2 分類 4 項目 (自己実現力，多様性の理解) を除外し，独自の 4 項目を追加して 12 項目を決める．以下で，詳しく述べる．

まず，基本項目として,『コンピテンシー評価項目表 (参照モデル)』[80] から，本提案で活用する 8 項目を表 3.5 に示す．

参照モデルの表の「目標の設定/到達レベルの確認」列を基準として，傾聴力・読解力，学習力・応用力・知識想像力，および，役割認識（チームワーク）・主体性・協働は，項目をまとめて扱う．

回答内容は，レベル 1(基本行動) からレベル 3(卓越行動) の 3 段階での自己評価とする．表 3.5 では，3 段階のレベルについては省略する (文献 [80] 参照)．

次に，カリキュラムの到達目標に合わせ，独自に追加する 4 項目 (計画力・共有データ・要求仕様の利用・メンバーの多様 (多様性の活用)) を表 3.6 に示す．基本項目と同様，回答内容は，3 段階のレベルに分ける．

レベル 1 では，個人の行動に関わることを挙げる．レベル 2 では，チームで相談する行動を挙げる．レベル 3 では，改善提案を示せるかどうかを挙げる．

プロジェクトマネジメントに関連する項目としては，計画力とメンバーの多様性を項目に取り上げる．このカリキュラムは入門レベルであるため，資格取得が目的ではないことから，基本的な概念を学び，プロジェクトを実践することでプロジェクトマネジメントの概念を体験的に学び，今後の活動に活かすきっかけとなることを目的として設定する．

3.7 本章のまとめ

本章では，高等教育におけるプログラミング入門の内容を含む必須カリキュラムが整っていないことを受け，社会に出る前に誰もが学ぶ高等教育におけるカリキュラムの必要性から，仮説に基づきカリキュラムの設計について検討した．

本論文で設定した仮説は，以下の通りである．

初等中等教育からの連続性と社会への接続に配慮し，高等教育における一般情報教育では，プログラミング入門とソフトウェア開発プロジェクトを組み合わせたカリキュラムでの実施が有効である．

プログラミングの学習を通じて，最低限学習するべきであると期待される内容は，言語を学びコードが書けるようになることではなく，プログラミングの概念を学び，プログラムがどのように作られ，どのように動いているかの原理を理解することである．また，高等学校までの学習指導要領には含まれていない，ソフトウェア開発プロジェクトに関する知識も含めて学ぶ必要があると考えた．

そこで，プログラミング入門の内容に加えてソフトウェア開発プロジェクトを加えて学ぶカリキュラムを提案した．ソフトウェア開発プロジェクトを含めることにより，プログラミングの知識を学ぶことに加えて，プログラムをどのように活用するかといった点を含めて学ぶことができ，これからの社会での活動に有効

に働くことを期待した．

カリキュラムの設計方針は,「情報化社会における教養として，すべての人に求められる内容を包括した高等教育におけるプログラミング入門のカリキュラムを構築すること」である．社会に出てから役に立つような形で「プログラミングを初めて学ぶ」環境を作ることを示した．

教員の負担の軽減や学習者の履修のしやすさを考慮し，高等教育における最小限の時間 2 単位で実施することを条件としてカリキュラムを設計した．

入門レベルにおけるプロジェクトでは，本格的なソフトウェア開発プロジェクトは困難であることから，描画教材を用い，1 つの絵のをソフトウェアに見立て模擬化し，チームで 1 つの絵を完成させる方法により，プロジェクトを実現することを考えた．前半 6 回をプログラミング入門とし，後半の 9 回でその知識を活用したソフトウェア開発プロジェクトとして構成した．

さらに，授業の 15 回の内容だけでなく，ワークシート，アンケートおよび，コンピテンシー評価項目を含めて検討し，シラバスに掲げる目的および，到達目標を設定した．

以降の章では，本章で設計したカリキュラムを実装する．

表 3.4: 各回の確認項目

回数	項目	確認内容
7	(2)	プロジェクトの運営をどのようにしていくかという予定
	(4)	チームでそれぞれの個性を活かし役割分担するのにどういう風にすればいいと思うかの目論見
	(5)	運営においてコンピュータはどのように活用していきたいか
8	(3)	仕様策定に当たって「発注者の要求」はどのように活かされましたか
9	(2)	やることが沢山あるうちで重要度をどのように考え作業を進めましたか
	(4)	実際に役割を割り振るところで予想外のことはありましたか
	(5)	設計の内容をどのように記録し管理することにしましたか
10	(2)	ここまでのプロジェクトの運営の流れはどうだったか
	(3)	開発予定のものが「発注者の要求」と合致することをどのように確認していますか
11	(4)	メンバーのこれまでの分担状況についてよかったこと，悪かったことは何ですか
	(5)	管理してきた情報をどのようにうまく修正しましたか
12	(2)	プロジェクトはどのようにうまく進められましたか
13	(4)	コードに不具合が見つかるとして，それは個人が「悪い」のでなく誰でも間違いはおかすという姿勢を保てましたか．やったことが「平等でない」 としてもそれは織り込みずみであるという姿勢を保てましたか
14	(2)	プロジェクトをどのように進められたと認識していますか
	(3)	要求にかなった成果物になったと考えますか
	(4)	役割分担はうまく行きましたか
	(5)	情報活用はうまく行きましたか

項目：(2)PM (3) 要求分析 (4) メンバーの多様性 (5)IT スキル

表 3.5: コンピテンシー評価の基本項目

・コミュニケーション力	
1. 傾聴力 読解力	相手の意見を丁寧に聴き，正しく理解し，受け止める 記述された内容を正しく理解する
2. 記述力	正しい文章で他人が理解できるように記述する
3. 議論力	議論の目標を設定し，それに合わせて効果的な議論を進める
・問題発見・解決力	
4. 課題発見力	現状と目標（あるべき姿）を把握し，その間にあるギャップの中から，解決すべき課題を見つけ出す．現状の問題点自体が明確でない場合も，顕在化している問題や現場観察から，真の課題（テーマ）を見出す.
5. 課題分析力	課題の因果関係を理解し，真の原因を見出し，その本質を整理する
6. 解決策立案力	問題点の原因に対し，それを解決する方策を複数立案し，その中で解決につながるものを選択し，アクションプランを導く（解決のプロセス全体を論理的に思考し実践する）
・知識獲得力	
7. 学習力 応用力 知識創造力	専門知識のみならず人文社会に関するものも含めて，幅広い分野で知識やノウハウを習得する 入手した知識やノウハウを関連付けて活用する 学んだ知識や自身で経験したこと・気づきなど全体の関係性を整理し，単なる知識の断片にとどめないで，新たな知恵を生み出すための知識化の作法を習得する
・組織的行動能力	
8. 役割認識 （チームワーク） 主体性 協働	チーム，組織の目標を達成するために個人の役割を理解し，当事者意識を持ってチームとして行動する 物事に対して自分の意志・判断で考え，自分から進んで行動する 共通の目標を達成するためにお互いの考えを尊重し，信頼関係を築くような行動をとる

表 3.6: コンピテンシー評価の追加項目

9. **計画力**：目標達成のために，計画的に作業を進める
1. チームで決めた計画に沿って自分の作業を進められる
2. チームで決めた計画に不備を発見して皆と相談できる
3. チームで決めた計画をさらに改良する提案が出せる
10. **共有データ**： 効果的にデータを作成し，共有する
1. チームで決めた共有場所のデータを参照して作業できる
2. チームで決めた箇所に共有データを追加して他人に使ってもらえる
3. チームで情報を共有する場所の新しい使い方の提案が出せる
11. **要求仕様の利用**：要求仕様に沿って作業を行う
1. チームで決めた仕様に合わせて自分の作業を行える
2. 自分の担当箇所がチームで決めた仕様と合わない時に相談できる
3. チームで決めた仕様が発注者の意図と合わないことを指摘できる
12. **メンバーの多様性** (多様性の活用)：個々の性格や特性を理解し，それを活かしたチーム作業を行う
1. チームで決めた自分担当の作業をこなすことができる
2. チームで決めた自分担当と他者担当の曖昧な部分を相談して調整できる
3. チームで担当を決める際にチームに適した分担方法を提案できる

第4章

カリキュラムの評価と社会が求める資質・能力の検証

4.1 はじめに

本研究の目的は，小学校プログラミングが普及した先を見据え，大学の一般教育におけるプログラミング教育のあり方も変化が必要であると考え，社会へ出る前に誰もが学ぶべき一般情報教育におけるプログラミング入門のカリキュラムを示すことである．対象は，大学の一般教養レベルであり，情報系の専攻ではない層を中心に考える．

これを検証するため，仮説として，以下を設定した．

初等中等教育からの連続性と社会への接続に配慮し，高等教育における一般情報教育では，プログラミング入門とソフトウェア開発プロジェクトを組み合わせたカリキュラムでの実施が有効である．

第 3 章で，高等教育におけるプログラミング入門科目に，ソフトウェア開発プロジェクトを組み込むことを検討し，高等教育の最小限の時間 2 単位で実施すること，多種多様な学習者が学ぶことを前提とし，前半 6 回をプログラミング入門とし，後半の 9 回でその知識を活用したソフトウェア開発プロジェクトとして構成したカリキュラムを設計した．

1.6 節では，以下のリサーチクエスチョンを提示した．

RQ1： PBL により，プログラミングの概念の理解と社会で求められる資質・能力の育成をはかることができるか？

RQ2： PBL によるプログラミング教育は，学習の過程を可視化することができるのか?

本章では，第3章で設計したカリキュラムを実装し，実践授業を行う．そして，構築したカリキュラムの妥当性とRQ1の社会で求められる資質・能力の育成について，次の観点から検証する．

- コンピテンシー評価の分析
- 到達目標からの評価

なお，RQ1のプログラミングの概念の理解に関しての検証は，第5章で詳しく述べる．

4.2 実践授業の環境

著者は，2017年度後期(2017年9月〜2018年1月)に，フェリス女学院大学国際交流学部国際交流学科専門科目「情報発信と世界」で提案するカリキュラムを実装した実践授業を行った．授業は，週1回90分で，15週で実施した．実践授業の環境について，表3に示す．

この科目は，1年生から4年生までが履修可能な選択科目である．他学部・他学科開放科目でもあるため，他学部の学生の履修もあった．履修登録は26名で，うち23名が最後まで履修した．途中で中断した3名のうちの2名は，4年生であり，卒業論文の提出時期(12月2週目)と重なり，両立ができなかったことが理由であった．もう1名は1年生で，授業開始時から欠席が多かった．この3名は，プロジェクトパートが始まる以前に履修を中断したため，チーム分けには含めなかった．必須の情報系科目がないため，学生の知識レベルは様々であり，パソコンが苦手であることを理由に履修した学生も多数いた．履修状況を，表4.1に示す．

表 4.1: 履修状況

学年	国際交流学部	その他の学部学科	合計
1	9	5(1)	14(1)
2	1	2	3
3	4	2	6
4	2(1)	1(1)	3(2)
合計	16(1)	10(2)	26(3)

かっこ内は履修中断人数

TAなどのサポート要員はなく，教員が1名で対応した．

教室の環境は，Windowsで，プログラミング言語RubyとImageMagick(画像表示ソフト)をインストールしただけの簡易的な設定で，学生の所有するパソコンでも対応できるよう，特別な設定は行わなかった．エディタには，メモ帳を使用し

た．宿題など教室外での学習のため，Mac ユーザには，エディタとして Atom を勧めた．

授業のテキストは，LMS(Moodle) を通じて配布し，課題等の提出もこのシステムを利用した．

コンピテンシー評価は，履修登録の都合から，授業の 2 回目と 15 回目に行なった．

4.3 実践授業

4.3.1 プログラミングパート

プログラミングパートは，6 回で実践した．授業の前半で，その時間の項目の説明 (前週の課題 B の解説を含む) をし，残りの時間で演習を行なった．5 回目以降で実施した図形の生成に関しては，図形ライブラリとしてあらかじめ教員が用意したメソッドを教材として配布した．

使用するプログラミング言語は，以下の理由から，構文がシンプルで初学者にも比較的学習が容易なプログラミング言語 Ruby を採用した．

- 社会で一般的に使われている環境でコードを書く体験をさせたいこと
- 記述量が少なく構文がシンプルであること
- 他のプログラミング言語に応用しやすいこと

サポートスタッフとしての TA の確保ができないことから，授業の設計段階ではペアプログラミングを検討したが，結果としては，単にペアを組んで取り組ませることとなった．なぜなら，1 人 1 台のパソコンがあること，課題のコードの行数がそれほど多くないことからである．コードを書き実行すること，授業内の課題 A と宿題の課題 B は，学生ごと個別に行ない，コードを考えることやエラーの処理は，ペアで取り組ませた．

これまでに，ペアで相談しながら学習する経験が乏しかったとのことで，最初の数回は，うまくいかない様子もあった．しかし，最終的には，ペアでのワークによる考える行動が，相互学習の効果として現れた．授業後のフィードバックには,「分からない時にペアの子に相談できるのは先生に頼りっぱなしでなく，自分たちで考えながらできるので良かったです」や,「ペアに教えてあげたり，エラーが出たときに何故だろうと考えたりする上で，解決策，間違った箇所を見つけられたので自分にも力がついた」などの回答があった．

各回の授業は，カリキュラムの設計に基づき，表 3.2 のスケジュール通りに進めた．

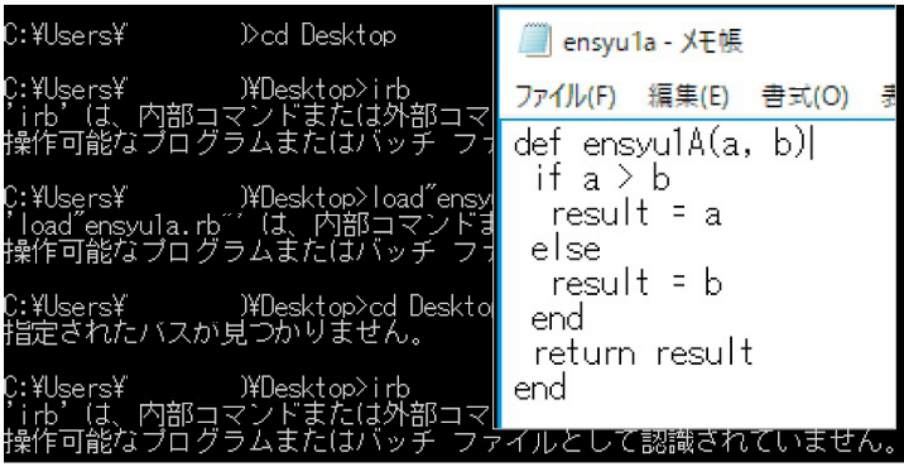
1 回目は，履修登録期間であることから，履修者が確定してないため，授業のガイダンスを含め，コードを書き，実行する手順を体験をした．例題には，三角形の面積を求めるプログラムを取り上げた．

2〜3 回目には，制御構造を取り上げた．2 回目の宿題の課題 B では，考え方はできているが，細かい文法でエラーとなるケースがあった．また，学生が所有する実行環境がないパソコンで課題に取り組むケースや，キーボードの入力が遅いことからくる遅れなど，プログラミングではない部分で躓いているケースなどもあった．提出された課題の一部を図 4.1 に示す．

```
def fact_(n)
        for i in n*(n-1) ..2
 i=1
end
```

```
irb(main):002:0> fact_ 4
=> 24
irb(main):003:0>
```

例 1：階乗の計算をするプログラム



例 2：環境がないためエラーとなる

図 4.1: 課題の回答例

図 4.1 の例 1 では，繰り返しを用いて，階乗の計算のプログラムを書いているが，繰り返しの条件の記述の仕方に戸惑っていることがわかる．例 2 では，プログラムは正しいが，環境がないことを理解しておらず，コマンドをむやみに打ち込んでいる様子がわかる．

4 回目までは，数学的な演習問題を中心として行なった．5 回目以降は，描画教材を用いた．最初は，円を描くメソッドのみを用い，6 回目に，線，三角，四角，楕円を描くメソッドを追加した．

5 回目のフィードバックからは,「いつもよりもわかりやすくて，パズル感覚でできた」,「久しぶりに自分自身が満足いく結果を出せた」,「自分の打ち出したプログラムが実際に画像になるのがすごく面白かった」などの回答があった．この回答から，対象とした層では，数学的な課題よりも明らかに視覚的に理解しやすい描画教材の方が有効であることが伺えた．

6 回目の学生のフィードバックからは,「絵が描けた時は，感動した．色々な絵を描いてみたい」,「最初は理解できなかったけど，実際に絵がかけたら，なんとなく

理解できました」,「だんだん今までやってきたことが形になっていっているのがわかって面白くなってきた」などのポジティブな回答が得られた．絵を描くことにより理解が進んでいるが，一方で，提出されたコードは，図形を単にプロットしたものであった．

他の学生と比較して出席率や課題の提出率が悪い学生からは，基礎知識の理解や考える行動につながっておらず,「まだ，言われたことを入力することしかできない」,「 難しくてついていけなくて焦ります」などの回答があった．これらの回答から，反復練習の重要性がわかる．

プログラミングパートでは，4 回目の抽象化の単元で理解されていない部分があった．また，5，6 回目では，図形の生成の仕組みは，理解されたが，その中で，制御構造や抽象化を使うところまでは及ばなかった．この点は，後半のプロジェクトパートで繰り返し活用しながら，補う授業設計により，後半のパートにつなげた．

取り組んだ課題については，5.3.1 節で，詳しく述べる．

4.3.2 プロジェクトパート

プロジェクトパートでは，1 チーム 4 名を基本として，6 チームを構成した．チーム分けは，プログラミングパートの各学生の提出物や授業時の様子などから見とった理解度と中間アンケート (5 回目の授業後) の個々の回答に基づき，担当教員である著者が行った (プログラムが得意なものが含まれることと，協力関係が築けることを考慮した).

中間アンケートの項目には，プログラミングの理解，グループワークに関連すること，その他のスキルに関することの 3 項目を挙げ，4 択式 (理解している・していない，好き・嫌い) の自己評価とした．中間アンケートの内容を，表 4.2 に示す.

アンケートに基づき割り振った，各チームの特徴を表 4.3 に示す.

プロジェクトでは，リアルな課題を設定することにより，学生のモチベーションを高め，社会における知識の活用イメージを持たせることができると考えた．取り組み課題を設定するにあたり，印刷機を扱う機械メーカーに協力を依頼し，完成した作品を実際に印刷して製品化する環境を作った．

また，他者を意識させるために，ターゲットを決め実際に販売することを目標としたネイルシールの作成を課題として取り上げ，ネイルシールとその PR のための gif アニメーションを作成する課題を設定した．

前半で学習したことを活用して，1 人で 1 作品を作成するのではなく，デザインを複数のパーツに分け，チーム内で分担し，ソフトウェア開発の工程に従って 1 つの作品とすることを条件とした．これにより，各パーツを抽象化してメソッドを作り統合作業を行うことや制御構造を活用することを期待した．制御構造は，アニメーションを作る工程でも活用されることを期待した．

ネイルシールは，枠内に制作した図案が収まるよう，ハガキサイズ大のフォー

表 4.2: 中間アンケートの内容

項目	質問内容
プログラミングの理解に関すること	メソッドについて理解していますか？ 分岐・ループについて理解していますか？ アルゴリズムについて エディタ・コマンドプロンプトについて コマンドプロンプトと Ruby の実行環境について
グループワークに関すること	グループワークは好きですか？ 人と話をすることは好きですか？ スケジュール管理をすることは好きですか？ 資料を整理したり，話し合いの記録をとったりすることは好きですか？
その他の項目	プログラムを作ることは好きですか？ アイデアを考えることは好きですか？ レポートを書くこと，文章を書くことは好きですか？ パソコンを使うことは好きですか？

マット (図 4.2 参照) を示した．アニメーションのサイズは，特に指定はせず各チームで自由とし，10 コマ程度で構成するよう指示した．

初回 (7 回目) の授業には，協力企業の方に参加いただき，課題の発表を行った．これにより，課題がより現実的なものとなり学生たちのモチベーションがさらに高まった．

しかし，プロジェクトパートの最初の 2 回は，コードの検討ではなく，図柄のデザインに集中してしまう傾向が見られた．その結果，ワークシートの記録や中間発表では，全てのチームで図 4.3 に示すようなイラストによる設計図となった．

プロジェクトの進め方については，表 3.4 の 10 回目の (2) のワークシートの記

表 4.3: 各チームの構成と特徴

GID	構成
A	留学生 3 名＋ 1 名の日本人学生
B	国際交流学部 2 名 (学年混在) ＋音楽学部 2 名
C	1 年生のみ
D	1 年生 3 名＋ 3 年生 1 名
E	1 年生 1 名＋ 3 年生 3 名
F	3 年生 3 名 (学部混在)

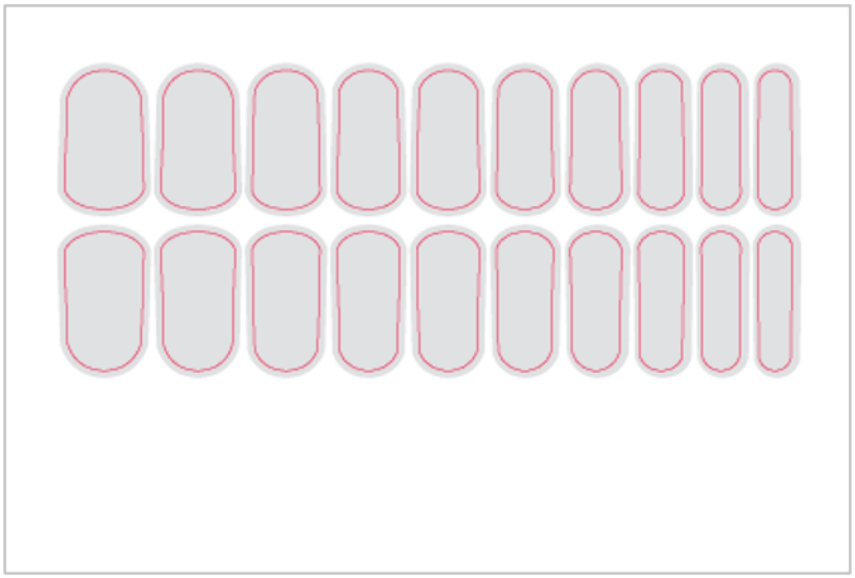


図 4.2: ネイルシールのフォーマット

録に，以下のような記述があり，各チーム様々な工夫を凝らし，ゴールに向かってうまく進める工夫をしていたことがわかる．

- デザインが難しそうなものを 2 人 1 組で分担して協力する
- 時間がかかる作業を先に考えてやる
- 得意な分野で分担して進める
- 一人でできること，チームで集まらないとできないことに分ける

途中，天候による授業時間の短縮による予期せぬ事故が発生し，後半の授業を予定通り進めることができなかった．特に，13 回目のコードレビューが実施できなかったことは，コードの質を高めることに大きな影響があった．また，成果物の印刷のための提出期限がタイトであったことから，学生は，絵を生成することに集中してしまった．

最終回の 15 回目は，学内の通常教室ではなく協力企業のスペースに場を移して発表会を行なった．

グループワークを通じ，チーム内の役割分担やコミュニケーションに注力した．役割分担は，グループワーク 3 回目 (9 回目) のワークシートで，チーム内の分担を記録し，役割の明確化をはかった．

最終発表会では，分担についてスライドを用い，各チームが発表した．学生の活動の様子を示すため，そのスライドの一部を著者が修正し，個人名を消去したものを図 4.4 に示す．上段は，D チーム，下段は，F チームが作成したものである．

D チームは，デザインごとに分担をし，ペアで作業をしていた．F チームは，チームメンバーが 3 名であったことから，パーツごとに役割分担をし，アニメーション

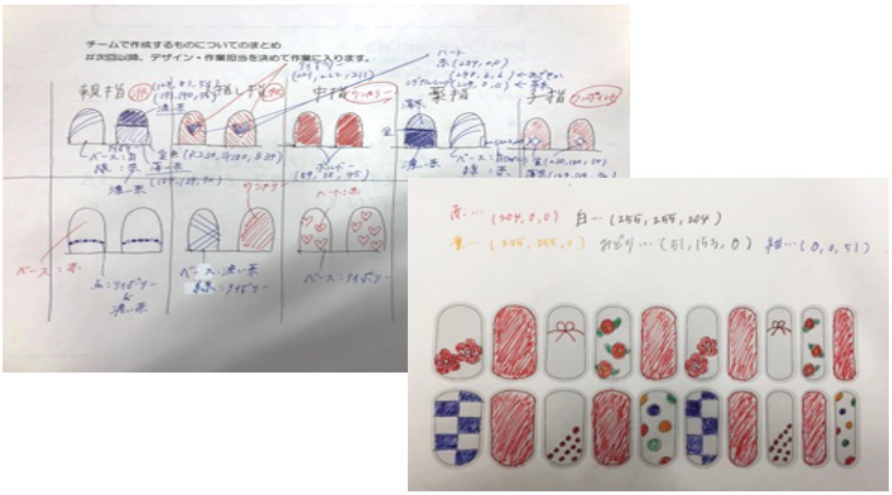


図 4.3: ネイルシールの設計図

と発表のパワーポイントの作成で分担を分け，個々の特性に応じて，たくさんある仕事を効率を考えて分担していたことがわかる．

役割分担については，12 回目の個人用のワークシートで次の 2 つの質問をし，作業状況について記録した．ただし，計画通りに授業の進行ができなかったことから，前掲の表 3.4 の質問項目とは，異なっている．

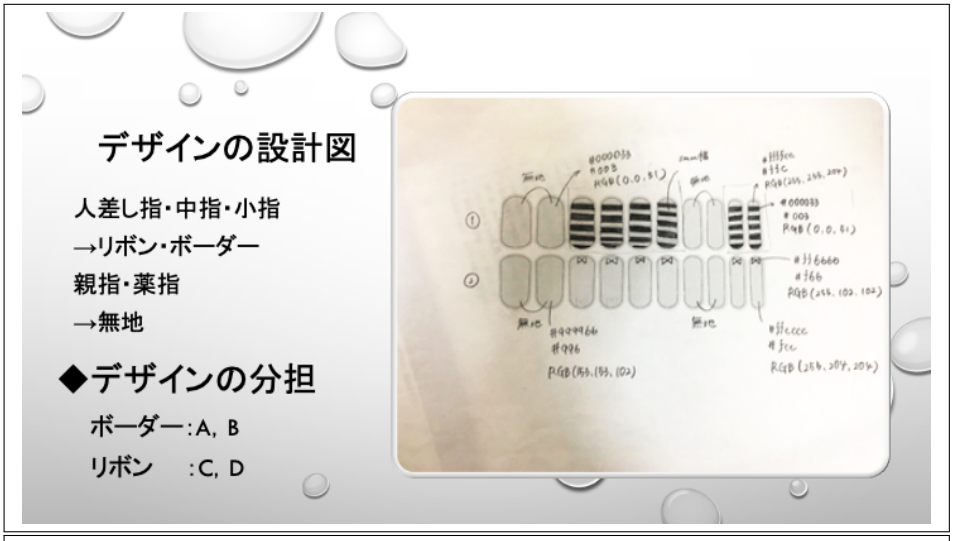
- チームで分担して作業を進めていますが，これまでの問題点を教えてください．
- チームで作業していく上で，個々の能力の違いや特性を活かしてチーム作りをしていますか？特定の人に負担がかかったりしていませんか？

6 チーム中 5 チームに属していた学生が,「特に大きな問題はない」と回答した．また,「得意分野でそれぞれ役割分担をしており，特に問題はなく，むしろ順調である」と回答した学生は，6 チーム中 3 チームに属していた．(到達目標 (4) に該当)．

当初の計画通りにはいかない部分があったが，全てのチームが自ら考えたデザインでネイルシールとアニメーションの作品を完成させ，期限通り成果物を提出することができた (到達目標 (1)(2) に該当)．

図 4.5 に，作成したネイルシールの作品の一部 (上段は E チーム，下段は F チーム) を示す．

図 4.6 に，gif アニメーションの作品の一部 (C チームの最終発表の発表資料より) を示す．



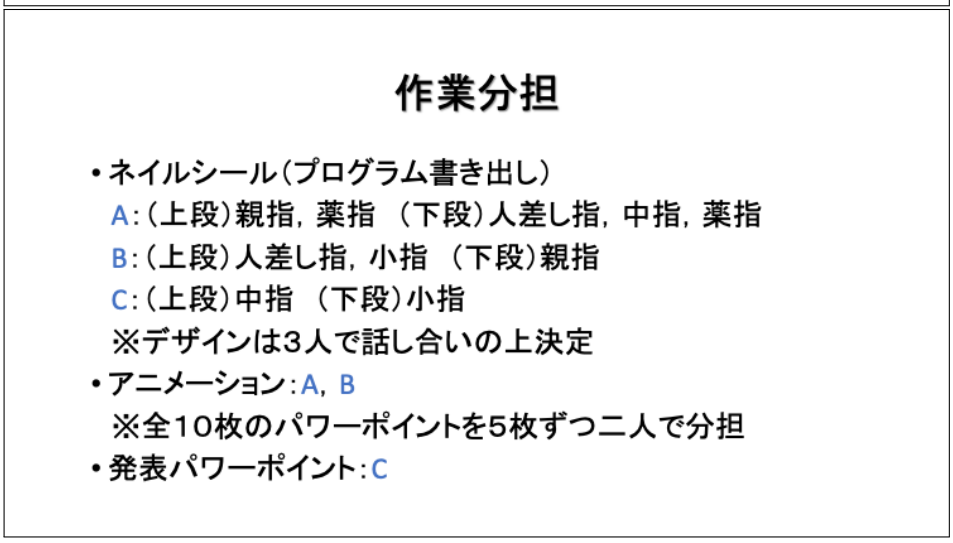


図 4.4: チーム内での作業分担の事例

4.4　カリキュラムの評価・考察

4.4.1　PBL の欠点の克服

文系大学においては，座学中心の授業が多いため，実践的な学びの場が限られてしまう．プログラミング入門科目に，成果物の作成を伴う実践的なチームによるソフトウェア開発プロジェクトを組み合わせることは，挑戦的な試みであった．

PBL の欠点の克服について，考察する．

- 欠点 1「学生も教員も多くの時間を要する」について，教員は，通常の授業と同程度の時間を本授業に費しており，また学生の授業時のワークシートの記録でも「過大な時間が掛かった」のような記述は見られなかったことから，欠点 1 は，克服できていたと考える．教員の対応については，15 通のメール

図 4.5: 作成したネイルシールの一部

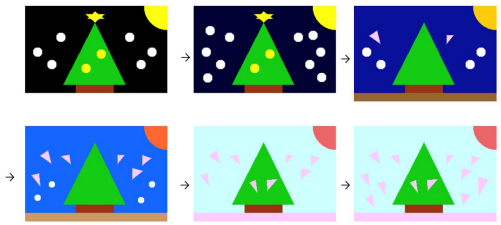


図 4.6: 作成した gif アニメーションの一部

のやり取りが記録として残っている．この程度の量であれば，特に大きな負担にはならない．

- 欠点 2 については，カリキュラム設計において，教材を制限し，教員がファシリテーターとなり，改善を図ることを考えた．各回の授業の初めに説明を加え，終わりにはワークシートへの記録をし，プロジェクト管理をすることにより，各チームをある程度ファシリテートできていた．数値化して示せるデータはないが，各チーム，出来上がったネイルシールの出来に達成感を感じていることから，欠点にはなっていなかったと推察する．

- 最終レポートでは，計画，役割の認識に関する記述が多く，適切な計画や役割認識ができていたことを示唆し，4.3.2 の図 4.4 からも，欠点 3「活動量の差が生じる」を回避できていた可能性が伺える．最終レポートの記述では，「役割分担できた」と 78.95%の学生が書いており，一方,「分担が偏っていた」のような記述は見られなかった．

- 欠点 4「メンバーの能力やスキル差によりアウトプットの質に差」については，最終レポートでは,「メンバーと助け合って (84.21%)」や,「得意な分野を活かして (42.11%)」などの記述があり，メンバーの能力の差にを助け合いや得意とする分野を活かすことにより，チーム内でのバランスをとっている回答があった．このことから，欠点 4 を克服できている可能性がある．

以上のことから，PBL の欠点は，本実践においてはカリキュラムの設計により欠点を克服し，問題となっていなかったと解釈できる．この結果から，欠点を考

慮して PBL を導入できたと考える．さらに，欠点 3，4 の克服は，到達目標 (2)(4) に関係しており，到達目標 (2)(4) の到達を示唆するものであると考える．

4.4.2　到達目標からの評価・考察

到達目標に掲げた項目から，本提案のカリキュラムの妥当性について考察する．

プログラミングの基礎知識については，演習問題では理解されていたと思われる分岐や繰り返しは，プロジェクトパートでの実践においては，絵を描くことに注力してしまい運用するところまでには至れなかった．

知識の習得と活用は，数学の学習経験を例にとると，例題を学び，演習問題をこなして終わってしまい，学習した知識を応用する経験にまで至らないことがある．本実践を行なった環境では，このような経験が乏しい学生が対象であり，プログラミングでも同様に応用することが難しく，前半で学んだ知識を応用して後半のプロジェクトにつなげることができなかったと推察する．しかし，最終発表では，繰り返しや分岐を活用する部分ついて，どの部分で活用するべきであったかを理解しており，プログラムの改善提案をするなど，今後の課題として発表するチームもあった．このことから，全く理解できていなかったわけではなく，到達目標 (1) の「制御構造について，説明することができる」が，達成できていると考えることができる．課題として取り上げた，ネイルシールのデザインは，指の数だけデザインを考えなければならず，自分たちが作りたいものへの具体的なイメージが強く，設計をして考える行為に至らなかったと推察する．制御構造・抽象化を活用するまでは至らなかったが，最終課題のネイルシールやアニメーションが出来上がっていることは到達目標 (1) の「制御構造について，説明することができる」以外の 3 項目に該当し，これらが達成できたと言える．

以上のことから，結果としては，到達目標 (1) は，概ね達成されたと考えることができる．

到達目標 (2) については，各チームがそれぞれ相談して成果物を完成できたことは,「問題認識から課題解決までを主体的に学ぶ」を実現したことに相当する．また，各チームが期限内に成果物を完成させたことは,「一定期間内に目標を達成」を実現したことに相当する．このことからも，到達目標 (2) が達成できたと言える．

次のレポートの回答からは，授業を通じて考える行為 (到達目標 (2)) に該当し，プロジェクトを通じて多面的・多角的に考える経験をしていた事がわかる．

- プログラミングの授業を受講したことで，他のチームの作品を見て「これはこういう風にプログラミング作成している」,「なぜ，これはこういう線が出せるのだろう」と考えられるようになった．
- 物事を進める時に，ある一定の視点で考えるのではなく，視点を変えてみることで，今後何かをする時に広い視野で物事を考えられ自分たちができることも増えるのではないかと思いました．

Ritchhart らは，思考の可視化は，理解を深めることも助けることに加え，思考と学習の関係を示すことである [27] と述べており，図形の生成を用いたプログラミングの学習は，可視化により学習者の思考と理解を明確化できたことから，学習者のモチベーショを向上させ，満足いく経験になったと推察できる．

小林は，大学という場の知を知識ではなく，知の行為という視点から考え直し，大学で何を学ぶかについて述べている [67]．

学生のレポートの記述にも複数，プログラムを書くことを通じて問題に遭遇し，それを解決するためのプロセスから学んだとあり，本提案のカリキュラムから得た学びは，まさに，小林のいう知の行為であり，大学で何を学ぶかと言う点において有効であったと考える．

課題に取り入れたネイルシールの制作は，本実践授業で対象とした女子大においては，学生のモチベーションも向上させられ，授業後の満足度を高められるエンゲージメント [109] を実現できたと考える．しかし，男女混在の大学においては，学生の興味関心も異なると考えられるため，課題の内容に関しては学習者の集団を意識して検討する必要があるであろう．

到達目標 (3) については，ソフトウェア開発では，他者がともなって行われることから，プロジェクトの課題でターゲットを設定した．授業内で要求仕様についての説明を加えながら，他者についての意識をさせるよう，配慮をした．プロジェクトパートにおける中間発表のワークシートでは，発表資料の整わなかったチームからは,「口頭発表だけでよくわからなかった」や「視覚的な発表はわかりやすかった」などの記録があり，他者へ正確に伝えることについては理解が進んだ様子が残っている．

コンピテンシー評価の分析からは，統計的な有意差は出ていることから，概ね到達できたと考える．しかし，他の項目と比較して，扱いが難しかった．

到達目標 (4) については，4.3.2 で述べたように，6 チーム中 5 チームの学生について該当する記録を残しており，PBL の欠点 3，4 でも述べたように，メンバーの得意な分野を活かして役割分担をしプロジェクトを進められていたことからも，この目標は達成できたと言える．

到達目標 (5) については，15 回の授業を通じ，タッチタイピングができるようになったり，課題提出や発表資料の作成などを通じて，様々なツールが使いこなせるようになったり，チームでの情報共有のためのデータ共有やメールおよび学修支援システム (LMS) の日々の利用などから，達成できたと判断した．

上述のことから，各チームでそれぞれ相談して役割分担をし，成果物を構築できたこと，期限内に成果物を完成させたことを総合すると，到達目標の (1)〜(5) の全ての項目において，概ね到達できたと考える．

次に，客観的な評価を行うために，最終レポートを到達目標に照らし合わせて，KH Coder[93] により分析した．分析したデータは，1 段落 1 名分とし，教員への謝辞などの不要な部分を削除した．KH coder で抽出された語から該当する語を到達目標の 5 項目に割り当て，コーディングルールを作成した．

各学生が到達目標の何項目に該当するレポートを書いていたかについて，分析した結果を図 4.7 に示す．

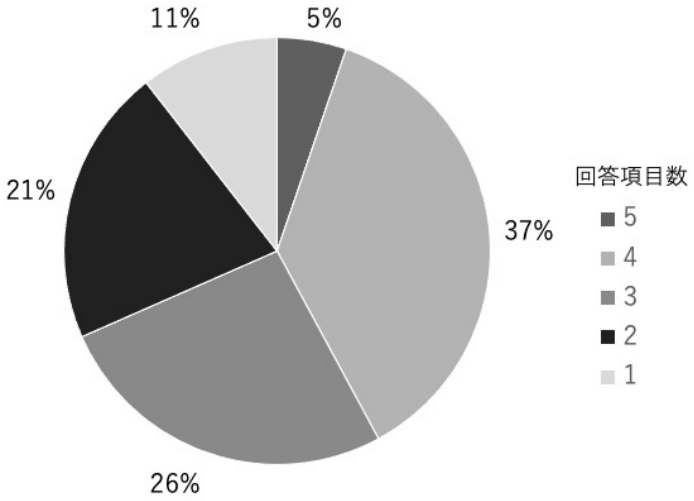


図 4.7: KH Coder による分析 1：到達目標の該当項目数

次に，到達目標の各項目についての回答された人数について分析した結果を，図 4.8 に示す．

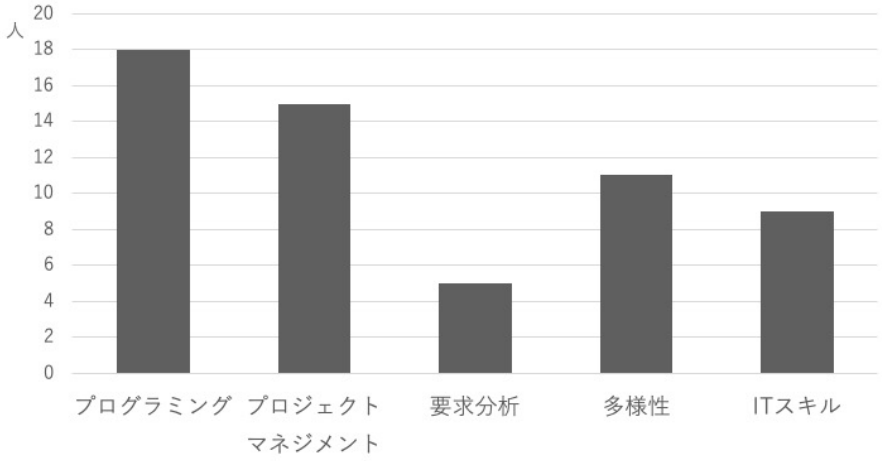


図 4.8: KH Coder による分析 2：到達目標の各項目の回答数

図 4.7 からは，到達目標の項目を 3 個以上含む学生は，68%いたことがわかる．また，図 4.8 からは，要求分析に対しての記述は少ないが，その他の 4 項目については，半分以上の学生が回答で触れていたことがわかる．

ここでの評価は，あくまでも学生の自由記述レポートに基づいており，網羅的な評価ではないが，学生は実践授業を通じて印象に強く残った特筆すべき点があればそれを最終レポートとして書いたと想定するのであれば，図 4.7，図 4.8 より，到達目標 (1)〜(5) は概ね到達できたと解釈できる．

4.5　コンピテンシー評価の分析

PBL による取り組みで期待される効果として，コンピテンシーの育成がある．本実践授業では，授業の 2 回目と最後の回に LMS(Moodle) を通じて，コンピテンシー評価を行った．その結果を，表 4.4 に示す．

各質問項目に対する回答は，レベル 1(基本行動) からレベル 3(卓越行動) の 3 段階の自己評価による選択であり，これを数字の 1〜3 に置き換え計算した．対象は，履修者 23 名のうち 2 回とも回答した 16 名である．

12 項目のうち半数の 6 項目 (表 4.4 の*，**) で，t 検定により有意水準 5%で有意差ありと判定された．また，4 項目 (表 4.4 の $\star$) で有意傾向あり ($0.05 \leq p < 0.10$) と判定された．特に，解決策立案力，メンバーの多様性 (多様性の活用) の 2 項目では，t 検定により有意水準 1%で有意差ありと判定された．

解決策立案力に対しては，本実践授業で対象とした学生は，単に図形をプロットするだけのプログラムでも，出力される絵には微妙な色の違いも含めてこだわりがあり，この色やデザインの出力結果の修正やプログラム実行時のエラーの対応などの作業を通じての効果であると推察する．

本研究のみで追加した到達目標に関連する 4 項目については，要求仕様とメンバーの多様性の 2 項目において統計的に有意判定された．

最終レポートの回答 (有効回答：19 名) では，著者が意味を理解し，該当するものをカウントしたところ，16 名 (84.2%) が，役割認識 (チームワーク)・主体性・協働について,「一人一人やるべき事をしっかりと考え行動」,「滞ったプログラミングの分は得意な人がサポートするなどして補った」と回答した．この項目については，統計的な有意判定はされなかったが，統計的に有意傾向ありと判定できる範囲であり，最終レポートの回答と総合して解釈すると役割認識 (チームワーク)・主体性・協働についても有効であったと推察できる．

上述のことから，本提案のカリキュラムは，15 回という限られた時間の中でも，コンピテンシーの育成に有効に働いたと推測できる．加えて，この結果は，カリキュラムの妥当性を示唆するものにもなり得ると考える．

4.6　本章のまとめ

本章では，仮説

初等中等教育からの連続性と社会への接続に配慮し，高等教育における一般情報教育では，プログラミング入門とソフトウェア開発プロジェクトを組み合わせたカリキュラムでの実施が有効である．

に基づき設計したカリキュラムを実装し，4.4 節でカリキュラムの有効性を示すための評価を行なった．

表 4.4: コンピテンシー評価

項目	1 回目平均	2 回目平均	t 値	p-value
	2-1 回目平均		95%信頼区間	
傾聴力・	1.412	2.000	2.582	0.020 *
読解力	0.588		0.105 ～ 1.071	
記述力	1.471	1.706	1.167	0.260
	0.235		-0.192 ～ 0.663	
議論力	1.235	1.588	1.852	0.083 ⋆
	0.353		-0.0511 ～ 0.757	
課題発見力	1.412	1.824	1.951	0.069 ⋆
	0.412		-0.036 ～ 0.859	
課題分析力	1.176	1.882	2.634	0.018 *
	0.706		0.138 ～ 1.274	
解決策	1.118	1.941	4.667	0.0003 **
立案力	0.824		0.138 ～ 1.274	
学習力・応用力	1.235	1.471	2.219	0.041 *
・知識創造力	0.235		0.010 ～ 0.460	
役割認識・	1.353	1.765	1.595	0.130
主体性・協働	0.412		-0.135 ～ 0.959	
計画力	1.471	1.941	2.057	0.056 ⋆
	0.471		-0.0144 ～ 0.956	
共有データ	1.529	2.000	2.057	0.056 ⋆
	0.471		-0.014 ～ 0.956	
要求仕様	1.412	1.882	2.219	0.041 *
の利用	0.471		0.021 ～ 0.920	
メンバーの	1.294	1.941	4.400	0.0004 **
多様性	0.647		0.335 ～ 0.959	

* ：$p < 0.05$，** ： $p < 0.01$，⋆ : $0.05 \leq p < 0.10$

$n = 16$

さらに，リサーチクエスチョンで示した，

RQ1： PBL により，プログラミングの概念の理解と社会で求められる資質・能力の育成をはかることができるか？

の後半部分，社会で求められる資質・能力の育成がはかられるか？について，4.5 節でコンピテンシー評価の分析を行い，評価した．

高等教育におけるプログラミング入門科目に，ソフトウェア開発プロジェクトを組み込むことを検討し，高等教育の最小限の時間 2 単位で実施すること，多種多様な学習者が学ぶことを前提とし，前半 6 回をプログラミング入門とし，後半の 9 回でその知識を活用したソフトウェア開発プロジェクトとして構成したカリキュラムを実装した．26 名の履修者に対して，実際に授業を行なった．設計に基づき，入門レベルにおけるプロジェクトではソフトウェア開発プロジェクトは困難であることから，1 つの絵のをソフトウェアに見立て模擬化し，チームで 1 つの絵を完成させる方法により，プロジェクトを実現すること実装した．

PBL の欠点の克服については，4.4.1 節で，授業の進め方の工夫および履修者のレポートや授業時のフィードバックの実践記録から，4 つの欠点について克服できていることを確認した．

15 回の授業でプログラミングとソフトウェア開発プロジェクトのすべての知識を身につけたと言い切ることは難しいが，全チームが，最終課題の作品 (ネイルシールのデザインとアニメーション) を提出することができた．

4.5 節で取り上げたコンピテンシー評価の分析結果からは，設定した 12 項目のうち半数の 6 項目で，t 検定により有意水準 5%で有意差ありと判定された．また，4 項目で優位傾向ありと判定された．優位傾向が判定されなかった項目についても，履修者のレポートなどから効果があったと読み取れる部分があり，社会で求められる資質・能力の育成に効果があったと解釈できた．

このコンピテンシー評価の分析結果は，RQ1 の後半部分が検証できたことを示唆している．

到達目標 (1)〜(5) を概ね達成していると解釈できたことから，90 分 15 回の中で，プログラミングを学び，チームでプロジェクトに取り組む入門科目として成立するカリキュラムが構築できたと解釈する．また，PBL による欠点を克服していることから，この方式であれば，短い時間で有効な PBL によるカリキュラムが構築できたと考える．

第5章

プログラミングの概念の理解と活用についての検証

5.1 はじめに

本研究では，高等教育における最小限の時間2単位で「プログラミング入門」とチームによる「ソフトウェア開発プロジェクト」を両立させるカリキュラムを検討している．

これまで，高等教育におけるプログラミング入門は，15回のうちの4〜6回で行われている事例 [100][127] があるが，多くは，基礎知識を学び，テストで評価するようなものであった．しかし，社会での活用を考えると，テストで出来た・出来なかったを判断するだけでは十分ではないと考える．そのため，学んだ知識を自分のスキルとして定着させ，運用できる状態にすることが望ましいと考える．しかし，著者の所属する大学においては．多くの学生は，十分手を動かし，学んだ知識を活用する経験がないまま社会に出ていく．プログラミングの学習で，十分な演習時間を設け，それを活用したプロジェクトを行えるような環境を整えようとした場合，多くの時間が必要となり，容易に教育環境を設けることは困難となる．

このような考えから，本研究では，2単位でプログラミング入門とソフトウェア開発プロジェクトを組み合わせたカリキュラムを検討してきた．描画教材を用い，チームによるプロジェクトを取り入れることで，学んだ知識を活用する場を設け，プログラミングの知識の定着と運用体験を目指した．

本章では，本研究で提案するカリキュラムのプログラミング入門部分の進め方と教材につい取り上げる．そして，実践授業から得られた提出物のプログラムのコードを分析して，教材と学習者の理解度から評価し，リサーチクエスチョンで掲げた，

RQ1： PBLにより，プログラミングの概念の理解と社会で求められる資質・能力の育成がはかられるか？

のプログラミングの概念の理解について検証する．

5.2　プログラミングパートのカリキュラムと教材

3.5.4 節では，プログラミングパートの 6 回の構成と進め方についての方針を示した．本節では，6 回の教材に踏み込んで議論する．

本提案では，後半のプロジェクトに繋げる必要があり，それまでに,「自分で思ったようにプログラムを書く」力を養うことを目的とする．

後半のプロジェクトでは，1 枚の絵をソフトウェアに見立ててプロジェクトを行うわけであるが，必要な最小限の知識として，制御構造と抽象化を取り上げた．2.2.1 節で述べたように，大岩は，抽象化の概念が知的生産一般に有効な概念であり，対象とする問題自体に対して何が本質的で重要なのかを考察し，重要度の低いことは切り捨てる抽象化を行わなければ良いプログラムは書けないと述べ，このような抽象化を正面から取り上げたプログラミングの教科書は見当たらない [49] と述べている．

著者は，絵のパーツをモデル化し，それを組み合わせることによりオブジェクトとしてのまとまりや絵の構造から，ソフトウェアを設計する仕組みの理解につながることを期待した．

入門レベルの教材は，これまでに様々な教材があることから，新たに作ることはせず，既に利用されて実績のあるものから応用することを考えた．本研究では，大学の共通教育の教材として既に利用されている，電気通信大学共通教育部情報部会が発行する「基礎プログラミングおよび演習 2017」[79] の#1〜#5 を使用した．文系の学生には困難であると思われる，微分積分などの高度な数学の内容で構成された例題・演習問題は削除し，ボリュームを落とすなど問題を吟味して修正を加えた．付録 A に，使用した演習問題のリストを示す．

プログラミングパートの前半は，定義がはっきりしている，数学モデルを用いた．描画教材は，実行結果から生成されたファイルに変換作業が必要となり，作業工程が煩雑になる．パソコンに不慣れな履修者を想定して，ある程度プログラムの実行手順に慣れてから，描画教材を導入する方が，プログラミングに集中して取り組めると考え，5 回目に導入することとした．そして，図形生成のメソッドでは，配列の知識が必要となることから，配列に関する説明を加えて，6 回を構成している (表 5.1 参照)．

1 回目は，初めてプログラミングに触れることを想定して，例題に倣ってプログラムとは何であるかを理解し，簡単なプログラムを動かせるようになることを目指す．三角形の面積を求める例題を示し，コードの書き方，実行の仕方，データの入出力について学ぶ．

2･3 回目は，プログラミングの基本となる制御構造について学習する．まず，絶対値の例題により，枝分かれを学ぶ．次に，平方根の計算の例題により，while 文

表 5.1: プログラミングパートのカリキュラム内容

回数	項目	内容	課題
1 回目	プログラミングの導入	プログラムとは何であるか理解し，簡単なプログラムを動かせるようになる．	課題 A/課題 B
2 回目	分岐と繰り返し	基本的な制御構造を理解し，これらを使ったプログラムが書けるようになる．	課題 A/課題 B，コンピテンシーチェック
3 回目	制御構造と配列	制御構造の組み合わせについてとデータ構造と配列を理解する．	課題 A/課題 B
4 回目	手続き･関数と抽象化	手続き (メソッド) による抽象化を理解する．	課題 A/課題 B
5 回目	2次元配列と画像	2 次元配列・レコード型と画像の表現を理解し，様々な形を描画する．	課題 A/課題 B
6 回目	プログラミングまとめ	自分が生成したいと思う画像のために何が必要かを考え，画像の内容に合わせてプログラム構造を実現する．	中間課題

を学ぶ．続いて，係数ループを学び，for 文を学ぶ．

3 回目には，制御構造の組み合わせについて，fizzbuzz 問題[1]を用いて学習する．4 回目は，手続き・関数と抽象化を取り上げ，例題には，逆ポーランド記法 (RPN, Reverse Polish Notation) 電卓を用いる．この回までは，例題・演習問題は，数学モデルを用いる．

定義がはっきりしていて，入出力の関係から，学習しやすい演習課題であると考えた．ただし，対象となる学生は，理系の学生でないことから，ある程度のハードルは設けるが，高等学校 1 年生程度以下の内容で対応できるよう考えた．

最初から，複数の作業が伴うことを避け，プログラムを書くことと，動かすことに慣れてきた 5 回目に，描画教材を導入する．

描画教材については，事前に準備した (1) 各種図形 (丸，三角，四角，楕円) を描くメソッドと (2) 生成した絵をファイルに出力するメソッドを活用させる (図 5.1)．

例えば，円を描くメソッドでは，中心座標，半径，色の RGB 値 (0〜255) を引数として指定することで円を描く．入門レベルのため，複雑な構造は作れないことから，必要な引数の部分を考えさせることにより，データ構造と抽象化を理解させることができると考えた．構造が単純なことから，後半のプロジェクトにおいても，設計を考えやすくできると考えた．

[1]海外で古くからある言葉遊びに fizzbuzz というのがある．輪になって「1, 2, . . .」と順に数を唱え，数が 3 の倍数なら「fizz」，5 の倍数なら「buzz」，3 と 5 の公倍数なら「fizzbuzz」と (数の代わりに) 言わなければならず，間違えたりつっかえたりしたら負けで輪から抜けるというもの．

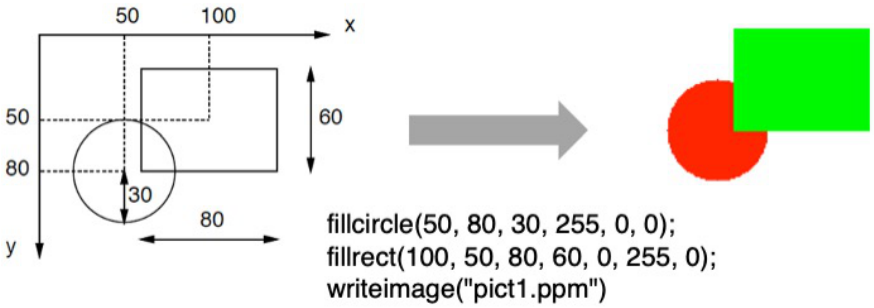


図 5.1: 図形描画の例

5.3 プログラミング課題の分析

5.3.1 プログラミングパートの課題

本節では，4.2 節で述べた，2017 年度に実施した実践授業の中から，プログラミングパートの 2 回目から 5 回目の提出物について取り上げる．

履修登録のルールにより，1 回目の授業は履修者が確定しておらず，履修人数が流動的であったこと，提出用の学修支援システム (LMS) の準備ができなかったことから，この期間を選択した．

課題 A は，授業時に取り組んだもの (締め切りは，授業の 2 日後) であり，課題 B は，宿題 (締め切りは，翌週の授業日の昼休み中) である．各課題は，複数の演習問題で構成されており，学生は，その中から指定されたものや各自で選択したものに取り組む．

ここでは，有効なデータとして取得できた履修者 18 名を対象とする．提出物のコードの行数とファイル数 (回答した演習問題の数) についてまとめた結果を表 5.2 に示す．

ID は個々の学生に番号を割り当てたものである．GID は，プロジェクトパートのチーム名を表す．課題名のカッコ内の数字は，演習問題の数を表す．表中の数字は，「(コードの総行数) / (ファイル数 (回答した演習問題の数))」を表す． () 内の数字にばらつきがあるのは，問題中の小問 (1-(a) のようなもの) を含めてカウントしたためである．また，課題 B には，課題 A で対象とした問題も含んでおり，重複してカウントしている．

提出されたファイル数が 2 つ以上のものは，複数の演習問題を提出している (したがって意欲の高い) 学生であることを表す．また，演習問題の数よりも大きな数字は，複数の考え方で回答していることを表す．

行数はすべてのファイルの合計であることから，ファイル数のわりに行数が多い場合は，より複雑な (行数の多い) 演習問題にチャレンジしたことを表している．

各課題は，コードと実行結果・コードに対する考察・アンケートから構成するように指示している．“0” は，課題は提出されたが，コード・コードに対する考察が

含まれていなかったこと (アンケートのみ) を示す．“-”は，実行結果のみで，コードが含まれていないため，行数がカウントできなかったものを表す．

表 5.2: 提出されたプログラムのコードの行数とファイル数

ID	GID	2A(3)	2B(10)	3A(3)	3B(17)	4A(3)	4B(9)	5A(7)	5B(17)	合計
1	A	8/1	21/2	17/1	17/2					63/6
2	A	14/2	15/3	33/3	24/3	6/1	25/1	108/4	0	225/17
3	B		15/2							15/2
4	B	8/1		68/4						76/5
5	C		17/2	7/1	11/2	16/1	0	17/1	17/1	85/8
6	C							0		0
7	C	8/1	17/2	18/2	18/2	13/1	0	29/2	13/1	116/11
8	C			8/1				-		8/1
9	D	8/1	31/3	16/2	17/2	6/1		17/1	0	95/10
10	D	7/1	25/2	9/1	17/2	26/2	0	17/1	0	101/9
11	D	8/1		11/1	10/1	7/1	22/1	17/1	24/1	99/7
12	D	8/1		13/1						21/2
13	E	7/1	42/6	13/1	61/4			26/1	54/2	203/15
14	E	7/1	20/2	13/1	17/1			16/1		73/6
15	F	8/1	24/2	29/3		16/1		-		77/7
16	F			-				0		0
17	F	105/15	21/4	27/3	16/3	34/5	4/1	51/2	82/3	350/36
18	-		15/2	8/1						23/3
提出人数		12	12	16	10	8	6	13	7	-

次に，各課題で取り組んだ演習問題を表 5.3 に示す．演習問題の番号は，付録 A のものと同様である．

課題は，完璧に出来上がらなくても，わからなかったことやできなかったことを説明したものでも可としたが，表 5.2 の結果からは，提出状況は良くなかったことがわかる．

この原因は，学生が過去の経験から「完璧に出来上がったものしか提出してはならない」と考えていたことと，大学における宿題文化のないことの 2 点が考えられる．また，学修支援システム (LMS) の使い方に慣れていないことから，提出ができないものもいた．

留学生は，日本語能力の問題から読み書きに時間がかかり，時間通りに提出することが困難であったと，後日，該当の学生との会話から事情を得た．

表 5.2 からは，3 回目に提出のピークがあり，その後，課題が難しくなり提出状況が悪くなった様子が読み取れる．5 回目は，絵を描く課題であり，再び学生の興味が現れるが，宿題までには手が及ばなかったものと推察される．

半分以上提出している学生は 11 名しかおらず，全回提出できた学生は 4 名のみであった．

表 5.3: 提出された課題の演習問題

課題番号	演習問題の番号 (人数)
2A	1a(12), 1b(2),1c(1)
2B	1a(1),1b(6),1c(4),2(6), 3a(4),3b(3),3c(2)
3A	1a(11),1b(9),1c(5)
3B	1a(1),1b(3),2(4),3(5),6a(3),6d(1),7(1)
4A	1a(6),2a(1),2b(2),2c(1)
4B	1a(1),2a(2),2b(1),2c(1)
5A	2a(6),2b(2),2c(1),2d(1),3a(2)
5B	2a(4),3a(1),3b(3),3c(1),3d(1)

課題 A については，授業時間内に実施し，まとめる時間を作って提出させていることから，課題 B と比較し実行可能なプログラムと実行結果が提出されていた，しかし，課題 B については，宿題であり，課題 A と比較すると提出率が低く,「わからない」「エラーがでてしまった」などの回答もあり，提出された物すべてが正解というものではなかった．ただし，実行時にエラーが出ているケースでも，考え方はできており，細かい文法のミスによるものが目立った．

表 5.3 からは，A 課題は，授業中に教員の説明に続いて課題に取り組んでいることから，取り組んだ課題に偏りが見える．一方，B 課題は，宿題であり，各自が自分にあった課題を選択して取り組んでいることから，選択した課題も様々であったことがわかる．これは，3.5.4 節での設計方針で述べたように,「学生は自分のレベルに応じて選んで解答する」に該当し，この方針が機能していたと推察する．

制御構造に関する課題は，2A〜3B が該当する．この提出状況は，2B(52.17%) と 3A(69.57%) となっており，10 回の中で提出率は一番高かった．

課題 2B の中で行なったアンケートでは，提出者のうちの 63.6%の学生が分岐について理解し，45.5%の学生が繰り返しについて理解したと回答していた．この回答からは，繰り返しよりも分岐の方が容易であったのではないかと推察する．

提出されたコードやアンケートの記述からは，2A では，テキストの模範回答に倣い，入力して実行するケースが多くあったことが読み取れた．2B からは，自分で考え，エラーを修正する行動が見られるようになった．一方で，分岐や繰り返しの考え方はできているが，end を入れる場所の理解が不十分なケースもあった．

抽象化については，4A，4B が該当するが，課題のアンケートからは,「RPN 電卓の仕組みについてよく理解出来たが，コードを書くところが難しかった」との回答があった．この回答からは，機能ごとに抽象化して，構造をつくることへの理解が難しいと推測する．また，提出できていない学生は，数学が苦手なことから，授業時の学生の様子から，モチベーションが保てず，プログラミングの知識としての抽象化の理解にまで至らなかったと考える．演習問題としては，加減乗

除の計算をする比較的易しいものであったが，本実践授業で対象とした学生が興味関心を持てる内容ではなく，抽象化を学習する目的ではない部分でマイナス要素があった．

数学的な課題が要因となり，提出率の低下や理解不足につながったと推察する．

5A からは，描画教材となり，前述した通り，わかりやすく理解が進んだことにより，提出率が上がっている．また，授業時のフィードバックには,「図を用いた課題はわかりやすく楽しかった」などの回答が多数あった．しかし，Ruby のプログラムの実行環境と画像ファイルの変換については，同じコマンドプロンプトを使って行なわせたことから，この相違について，理解が難しい学生がいた．補足資料を提示して，説明を行なったが，手順が増えることで理解が追いつかない学生がいた．5A，5B が提出できていない学生は，ファイルの変換作業のところで躓いているものが目立った．

この点は，プログラミングの問題だけでなく，コンピュータの仕組みの理解や操作について，大学入学以前の経験が乏しいことが要因として考えられる．

授業時の見取りからは，全く理解できずに取り組むことができない学生はおらず，出席率も高かったが，課題の提出率には反映されなかった．

提出された課題の質については，入門レベルということもあり，課題 3B までは数行程度のものであり，難しすぎる設定にはなっていなかったと考える．

以上の結果から，例題や数学的な教材については，見直す必要があることがわかった．描画教材の活用は，プログラミングの基礎知識を学んだ上での導入を考え，授業を設計した．提出状況や学生の回答から効果があるが，4 回目までの教材との接続を含めて，導入のタイミングを再考する必要があることがわかった．

5.3.2 プロジェクトパートの課題

プロジェクトパートでは，チームでネイルシールとアニメーションの 2 つの課題に取り組んだ．本節では，このうちネイルシールのプログラムについて取り上げる．

チーム毎のファイル数とコードの行数などを調べた結果を，表 5.4 に示す．チームの構成は，表 4.3 と同様である．

表 5.4: 最終課題におけるネイルシールのプログラムの状況

GID	A	B	C	D	E	F
ファイル数	19	4	1	1	1	3
合計行数	2396	515	279	94	102	258
制御構造	0	0	0	0	0	0
作品：ネイルシール	○	○	○	○	○	○
作品：アニメーション	○	○	○	○	○	○

プログラミングパートでは，プログラムに自信がなく課題の提出ができなかった学生も，プロジェクトパートの終わりにはチーム全体で少なくとも 100 行程度のプログラムが作れるようになり，作品も完成している．特に，A，B グループは，前半の課題の提出状況と最終課題の状況を比較すると，作成したプログラムの数や行数に大きな成長が見られる．

提出するファイルは，各自が作ったパーツをチームで 1 つのファイルに統合して提出されること，を想定していた．しかし，次の理由から統合の作業ができず，チームごとに提出されたファイルの数に大きな差が生じたと推察する．

- 授業時間の不足 (初回授業の進行遅れ，天候による授業時間の短縮) による作業時間の不足
- 印刷の都合上の課題の提出期限の時間の問題
- 複雑なデザインにより統合が困難 (メソッド化ができてない)

著者の授業時の見取りでは，チーム内での分担に多少の負担の差はあったものの，4.3.2 節で述べたように適切に分担され，それぞれが手を動かしており，また他のメンバーのコードについて意見を互いに述べたり，助け合いをしたりするなどのことも多くの学生ができていた．

最終レポートでは，23 人中 19 人の有効回答があり，そのうち，17 名 (89.47%) が学習した範囲でプログラミングについて理解したという内容の記述をしており，また，8 名 (42.11%) が自分たちで考えた絵をプログラムで実現したことについて述べている．ただし，プログラムの水準については，図形を順にプロットしていくコードであるために行数が多くなっており，制御構造や手続きを組み合わせたプログラムで 100 行のものが作れているわけではない．

この原因としては，前述の図 4.3 でも示したように，作成する絵のデザインにこだわり，図形をどのように組み合わせたら，そのデザインに近づけるのか，より具体的な絵を考えることに注力しまったことが考えられる．予期せぬ事情により授業時間も不足したこともあり，再考することもできず，総合するところまで至れなかったことが挙げられる．

この点に関しては，制御構造・抽象化を教えるだけでなく，設計するということに重みをつけ説明を加えることが必要であったと考える．また，再考するためのレビューの時間や統合の作業も十分な時間がとれなかったことから，改めてレビューや統合作業についても検証する必要がある．

15 回目の発表会では，制御構造を使う部分についての見当はついていたが，実際に使うことができなかったという報告も複数のチームからあった．気がついてはいたものの，運用レベルにまで到達させることができなかった点は，繰り返し練習する時間の不足もあったが，やはり，4 回目の授業の構成に課題があり，十分な理解につながらなかったと考える．

プログラミングパートで理解が不十分であった項目については，プロジェクトパートで補うことを考えたが，一つ一つの項目は理解されても，それをどのように運用したら良いのかを考えて実践に移すことは難しく，優しい方向に流れてしまうことが，実践に現れていたと推察する．この点に関して，カリキュラムを見直すとともに，新しい知識を学び，それを練習し，応用するということを，プロジェクトが始まる以前に練習させる必要性を感じた．

自分たちで考えた絵のデザインに対してコードを書き，表現でき，作品が完成していることから，概ね前章で示した，到達目標 (1) は達成できたと解釈はできるが，活用や応用という点に関しては，改めて検討する必要が見えた．

5.4 本章のまとめ

本章では，

RQ1： PBL により，プログラミングの概念の理解と社会で求められる資質・能力の育成をはかることができるか？

のプログラミングの概念の理解について，明確にするために，2017 年度に行なった実践授業の結果より，教材と実践授業における提出物から議論した．

5.3 節では，5.3.1 節で，授業の前半のプログラミングパートの提出物を分析し，5.3.2 節で，後半のプロジェクトパートの提出物を分析した．

5.3.1 節の分析結果からは，1〜4 回目の例題・演習問題に数学モデルを用いたことが，数学に苦手意識を持つ学生にとっては本質でない部分で足枷となってしまっていたことが明らかになった．特に，4 回目の抽象化については，その仕組みや機能が十分理解されないままプロジェクトを進めることになったしまった．プロジェクトパートで繰り返し学ぶことで理解が不十分な部分については，補足し，定着を図ることを考えていたが，天候による授業の短縮などの影響もあり，計画時の目論見通りには遂行することができなかった．

5.3.2 節の分析結果からは，各チームとも課題の作品は完成してることがわかったが，前半で学んだ制御構造はどのチームも活用できていないことがわかった．15 回目の発表会では，活用できなかったが活用する場所について理解されていたという報告もあったが，運用するところまでには至れなかった．前半のプログラミングパートでの理解不足と後半の授業時間の不足に加えて，容易にできる方向に流れてしまう履修者の特性が現れた結果になったと推察する．

しかし，どのチームも自分たちで考えた課題に対して役割分担をし，コードを書き，表現でき，作品が完成していることから，3 章で掲げた到達目標 (1) は達成できた．

提出された課題からは，教材に用いた例題・演習問題の内容や繰り返し取り組む時間の配分などに関する課題が明らかになり，授業前半のプログラミングパー

トの教材にとりあげた例題と図形描画の例題を導入するタイミングを修正する必要があることが明らかとなった．

初学者を対象としたプログラミング入門においては,「絵を描く」という題材の分かりやすさは，効果的であることから，後半のプロジェクトに関連しない要素は取り除き，関連する内容で教材を構成する方向に修正することが望ましいと考える．

以上このとから，RQ1 のプログラミングの概念の理解については，到達目標の観点からは概ね達成できていることが示された．問題点として明確になった部分については，第 6 章で修正して，再度検証する．

第6章

プログラミング入門と描画教材の効果の検証

6.1 はじめに

第3章では，プログラミング入門教育の新たな提案として，専攻によらず誰もがプログラミングの概念を学び，その原理を理解することに加え，“プログラムを活用するためのソフトウェア開発プロジェクトを理解することを含む”という要求を満たすために，PBLを組み合わせた授業カリキュラムを提案した．そして，第4章で，提案に基づき構築した半期90分15回2単位のカリキュラムを実装し，実践授業を行い，カリキュラムの妥当性について検証した．

描画教材の特性を活かし，前半のプログラミングパートでは，プログラミングの概念を学ぶ過程で，図形の生成ができるようになることを目標に取り組み，後半のプロジェクトパートでは，前半で学んだ知識を深め，活用し，1つの絵をソフトウェアと見立てて，チームで1つの絵を描く課題に取り組み，ネイルシールのデザインとアニメーションを制作した．

この実践を通じて，社会に求められる資質・能力の育成をはかるため，コンピテンシー評価の分析を行った．また，第5章で，プログラミングの知識について，提出されたプログラムを分析した．カリキュラムの設計では，授業の前半部分で理解が乏しい部分については，後半のプロジェクトにおいて補うことを考えたが，設計通りには機能しなかった．そのため，プログラミングの知識の獲得は概ね達成したが，知識の活用までは至らなかった．この要因は，カリキュラム前半の進め方と演習課題にあることが明確となった．

本章では，前章までに明確となった課題の改善を検討し，プログラミング入門とソフトウェア開発プロジェクトを組み合わせたカリキュラムを再構築し，リサーチクエスチョンの検証を行う．

本論文では，リサーチクエスチョンとして，以下の2点を掲げた．

RQ1： PBLにより，プログラミングの概念の理解と社会で求められる資質・能

力の育成をはかることができるか？

RQ2： PBL によるプログラミング教育は，学習の過程を可視化することができるのか?

以降の節では，カリキュラムの改善の方針と再設計について述べ，改善後のカリキュラムを実装した実践授業について，RQ1 のプログラミングの概念の理解についての点から評価して，補強する．さらに，2 回の実践授業の結果をまとめ，RQ2 の答えとして，PBL を組み合わせたことによる効果を検証する．

カリキュラムの修正により，授業の 1 回目から絵を描くことを積極的に使うこと，課題にものづくりの要素を入れ，テキストの課題を解くだけではない，実践的な学びによる効果を目指す．

本研究で提案するカリキュラムの目的は，高度なプログラムが書けるようなプログラマーを養成することではなく，プログラムの概念を学び，自らの考えを形にするプロセスを学ぶことで，今後の活動に活かせるような知識・能力を育成することである．履修者全体がどうであったかを統計的に示すのではなく，多様な履修者がどのように学んでいるか履修者の課題・レポートなどの記述データから検証する．

6.2　カリキュラムの再考

第 3 章では，プログラミング入門に PBL を組み合わせたカリキュラムを検討し，90 分 15 回 2 単位の授業として実装した．このカリキュラムは，概ね成功したが，第 4 章で示したように，ループや枝分かれの制御構造および抽象化の理解と活用に関して課題が明らかとなった．

描画教材を利用することにおいては，先行研究 [127][100] と同様の問題が発生し，最終課題では，学んだ知識 (ループや分岐) を活用せずに単に図形をプロットするのみのプログラムが提出された．

この原因は，授業の後半の悪天候による授業時間の短縮など，時間不足による影響もあったが，1 番の要因は，第 4 章で明らかになったように，授業前半のプログラミングパートの教材にとりあげた例題と図形描画の例題を導入したタイミングであった．特に，1〜4 回目は，数学モデルの例題であり，数学に苦手意識を持つ学生にとっては本質でない部分で足枷となってしまっていた．

実践授業を行なった環境は，文系女子大学であり，学生たちの多くは，数学への苦手意識が高く，問題の理解に加えて，興味を持って取り組むまでのモチベーションを維持することが困難であった．そのため，前述したように，プログラミングの学習とは別の要素が躓きの原因となっていた．また，学習した知識を応用するという行為に慣れていないことから，簡単にできる方法に流れやすく，後半のプロジェクトパートで絵を描くことへの繋がりも良くなかったと考える．

表 6.1: プログラムパートの再構成

回数	項目	内容・修正点
1 回目	プログラミングの導入	プログラムとは何であるかを理解し，簡単な絵を描く
2 回目	分岐と繰り返し	絵を描くことを通じて制御構造を理解し，そのプログラムを書く
3 回目	制御構造と配列	配列について理解し，それを用いたプログラムを書く
4 回目	手続き・関数と抽象化	手続き (メソッド) の理解と抽象化について理解し，各自の考えた絵を手続きを用いて書く
5 回目	2 次元配列と画像	配列について理解を深め，2 次元配列と動画の仕組みについて理解する
6 回目	プログラミングまとめ	5 回の学習内容を理解し，プロジェクトにつなげる

プログラムを考えることとコードを書き実行するまでの作業工程を覚えることの両方がそれぞれ難しく，これらを同時に進行させることについては，習得するまでに時間がかかることから，反復練習する時間を十分に取る必要がある．当初は，コンピュータの知識が乏しい履修者もいることから，プログラムの実行過程に慣れたところで，描画教材を導入することを考えた．しかし，実行過程が煩雑であっても，視覚的に理解しやすい教材で最初から学ぶほうが，困難が少ないという履修者の反応もあったことから，描画教材を導入するタイミングを再検討することに至った．

授業時のフィードバックの回答から，該当する部分を示す．

> プログラミングの細かい内容よりも絵などから入っていったほうが やはりわかりやすいのではないかと思います．文章を先に学んでも活用の仕方が難しいです．

上述したことをふまえ，著者は，1 回目から描画教材を導入することし，全体の構成は変えず，例題や課題の内容を絵を描くことに置き換えて進めるよう修正した．これにより，学習者の数学的な演習課題による苦手意識を軽減し，図形生成のプロセスは繰り返し練習できるようになり，問題を解消できると考えた．表 6.1 に，再構成した内容を示す．

1〜6 回の項目は，変更せず，1〜4 回の例題や演習問題の内容の改善を図った．教材は，当初の教材と同様に，事前に用事した丸や四角などの図形を描くメソッドを使いる．初回は，プログラミングの仕組みとエディタやプログラムの実行環境などについて触れる．1 回目の最初の例題は，1 回目の試行と同様，三角形の面積の例題を用い，プログラムの実行手順を理解したところで，描画教材を導入し

表 6.2: 試行の環境

対象学部	国際交流学部 (他学部・他学科開放科目)
対象学年	1 - 4
期間	1 セメスター (15 回)
コマ数と単位	週 1 コマ (90 分)，2 単位
前提知識	特に必要なし

た．2 回目は，初回の復習をしながら，分岐と繰り返しの制御構造を学ぶ．3 回目は，複数の制御構造の組み合わせと配列の仕組みを学び，4 回目で，初回から使ってきたメソッドの機能の説明を加えて，各自で簡単な絵のメソッドを考える．5 回目は 2 次元配列と動画の仕組みについて理解を深め，その後の活動への接続とする．テキストに用いた演習問題を付録 B に示す．

プログラミングパートのまとめとして，中間課題には，単なるお絵描きの課題とはせず，ものづくりの要素を含んだ内容の課題を取り入れることを追加した．コードを書く目的をはっきりさせるとともに，設計する点を重視させつつ，課題にメソッドやループ・分岐を使うよう促す．

最終課題についても同様の条件をつることにより，個人からチームへの取り組みと発展させることができ，個人とチームでの活動の相違について考えることができるのではないかと考えた．

この一連のカリキュラムの修正により，ループや分岐の制御構造の活用の課題解決に繋げることができ，中間課題におけるもの作りでの成功体験や失敗体験は，後半のプロジェクトパートへのモチベーションにもつながると期待した．

カリキュラムの修正に伴い変更した，各回の授業後のアンケートの項目を付録の表 D.1 に示す．

6.3 教材修正後の実践授業

6.3.1 実践環境

著者は，2018 年度後期，2017 年度と同様に，フェリス女学院大学国際交流学部国際交流学科専門科目「情報発信と世界」で実践授業を行なった．実践の環境に関しての状況を，表 6.2 に示す．この科目は，1 年生から 4 年生までが履修可能な選択科目である．他学部・他学科開放科目でもあるため，他学部学生の履修もあった．

授業は，週 1 回 90 分で 15 週 (2 単位) で実施した．教員は 1 名で，TA などのサポート要員はなし．履修登録は 19 名で，うち 17 名が最後まで履修した．履修状況を表 6.3 に示す．

プロジェクトパートでは，4〜5 名で 1 チームを構成した．グループ分けは，プ

表 6.3: 履修者の学年・学部の内訳

学年	国際交流学部	その他の学部学科	合計
1	7	3(1)	10(1)
2	1	0	1
3	5	3(1)	8(1)
4	0	0	0
合計	13	6(2)	19(2)

かっこ内は履修中断人数

表 6.4: チーム構成の概要

チーム	人数	構成の概要
A	4	他学部からの履修者で，1，3 年生各 2 名
B	5	国際交流学部 1〜3 年生混在
C	4	国際交流学部 1，3 年生各 2 名
D	4	国際交流学部 1 年生 3 名，3 年生 1 名

ログラミングパートの各学生の提出物や授業時の様子から見とった理解度と中間アンケート (表 4.2) の回答に基づき，著者が行った．その際，プログラムが得意と回答した学生が含まれることと，協力関係が築けることを考慮した．中間アンケートの項目は，プログラミングの理解，グループワークに関連すること，その他のスキルに関することの 3 項目について 4 択式 (理解している・していない，好き・嫌い) の自己評価とした．

アンケートに基づき作成したチームの構成について，表 6.4 に示す．

6.3.2 実践授業

再考した教材を用い，当初の授業設計と同様に，授業の 1 回〜6 回をプログラミングパートとし，7 回目以降をプロジェクトパートとして，授業を実施した．カリキュラムの各週の実施内容を，表 6.5 に示す．

1 回目の実践と同様，プログラミング言語には Ruby を採用した．教室の環境は，Windows で，Ruby と ImageMagick(画像表示ソフト) をインストールした．エディタは，メモ帳を変更して Atom とした．変更の理由は，個人の所有するパソコンがの OS が混在していることから，授業と宿題をする環境の統一のためである．

2017 年度の最初の実践の中間課題では，プログラミングの知識のまとめの課題として単に絵を描く課題としたが，本実践では，その本質は変更せず，完成した絵を印刷して形作ることを取り入れ，トイレットペーパーの外装のデザインとした．図 6.1 に，提出された課題の作品を示す．

プロジェクトの課題は，ネイルシールのデザインでは，学生の興味を引きモチ

表 6.5: 15 回の授業内容

回数	内容
1 回目	プログラミングの導入
2 回目	分岐と繰り返し
3 回目	制御構造と配列
4 回目	手続き・関数と抽象化
5 回目	2 次元配列と画像
6 回目	プログラミングのまとめ
7 回目	グループ作成・課題とゴール設定
8 回目	アイデア出し・ブレスト・仕様決め
9 回目	デザイン・設計・分担・制作
10 回目	試作・中間ドキュメント作成
11 回目	中間発表・仕様見直し
12 回目	制作・単体テスト
13 回目	コードレビュー
14 回目	制作・統合テスト
15 回目	最終成果発表会 (まとめ)

ベーションを保つためには良い教材であったが，指ごとにデザインすることが課題の目的とのバランスが悪く，大きな負担となったことから，1 チームで 1 つの絵を描くことのポリシーを保ち，A4 サイズのクリアファイルのデザインと，それを宣伝するための gif アニメーションの作成とした．実践の記録を，付録 E に付す．

6.4 成績評価の観点とルーブリック

6.4.1 ルーブリックの作成

カリキュラムの修正後，実践授業を評価するために，ルーブリックを検討し，作成した．

前述したカリキュラムの設計の方針や到達目標に基づき，本授業での目的であるプログラミングの観点を厚くし，到達目標の 5 項目を包含した評価項目を考える．

ルーブリックの作成は，独立行政法人情報処理推進機構 IT 人材育成本部イノベーション人材センターが作成した『実践的講座構築ガイド～産学連携教育の自律的展開を進めるために～第 3 部評価基準編』[80] の「コンピテンシー評価項目」および Association of American Colleges & Universities が発行する『Obtain ALL 16 VALUE Rubrics』[7] を参考にした．

著者が作成したルーブリックを，表 6.6 に示す．

本カリキュラムは，入門科目と位置付けていることから，チェック項目は細かく

表 6.6: 成績評価のためのルーブリック

ID	項目	レベル：1	レベル：2	レベル：3
#1	プログラミングの仕組み理解	プログラムの一部を見て文の構造や変数の違いを見分けることができる.	与えられたプログラムを見て，入出力の関係を理解し動作を追跡することができる.	与えられたプログラムを見て，書かれたプログラムの動作を説明できる.
#2	プログラム作成	お手本通りにプログラムを書き実行することができる.	お手本を真似て，自分なりのプログラムを書き実行することができる.	自分で考えた動作をプログラムすることができる.
#3	アルゴリズム	アルゴリズムの概念を理解することができる.	アルゴリズムが与えられれば，それを実現するプログラムを書くことができる.	問題が与えられた時，その問題を解くようなアルゴリズムを考えてプログラムを書くことができる.
#4	計画	決められた計画に沿って自分の作業を進められる.	決められた計画に不備を発見して皆と相談できる.	決められた計画をさらに改良する提案が出せる.
#5	役割の認識	決められた自分担当の作業をこなすことができる.	決められた自分の担当と他者の担当の曖昧な部分を相談して調整できる.	担当を決める際にチームに適した分担方法を提案できる.
#6	チームビルディング	個人の役割を理解して主体的に行動することができる (個人的行動)	個人の役割を全うするだけでなく，お互いの考えを尊重し，信頼関係を築いて行動できる．(サブリーダー)	チームの目指す方向 (ビジョン) を共有し，目標を達成するために率先して行動することができる. (チームリーダー)
#7	要求分析	仕様に合わせて自分の作業を行うことができる.	自分の担当箇所が仕様と合わない時に相談できる.	仕様が発注者の意図と合わないことを指摘できる.
#8	情報共有	共有場所のデータを参照して作業できる.	共有データを追加して他人に使ってもらうことができる.	情報を共有する場所の新しい使い方の提案をすることができる.
#9	問題解決	問題解決のため，一つのアプローチを考えて対応することができる.	問題解決のため，受け入れ難いアプローチについて考え，それを却下することができる.	いくつかの選択肢の中から選択して，問題解決のために論理的で一貫した計画を立てることができる.

図 6.1: 中間課題の作品

複数のレベルに分けず，3.6 節のコンピテンシー評価に倣って，3 段階（基本行動～卓越行動）とした (表中の 1～3).

表中の項目の#1～#3 は，プログラミングに関する内容である．ここでは，プログラミングの仕組みが理解できているかどうか，コードを真似して書けるか自分の力で書けるか，そして，アルゴリズムの概念を理解し，自分自身でそれを導くことができるかどうかを問う項目で構成した．

項目#4～#9 は，プロジェクトマネージメント，要求分析，情報共有の観点から構成した．

#4 では，プロジェクトを進める際に重要な計画について，決められた計画を遂行できるか，それを発展させて不備の指摘や改良提案ができるかを項目として取り上げた．

#5 では，チーム内での各自の役割の認識についての理解を問う項目とし，自分の行う作業分担の認識について意識を持たせることが目的である．#4 と#5 は，意味が似ているように取れる部分もあるが，計画は，日程的なものを示し，役割は，各自の作業の内容を意味する．

#6 では，個人の役割をチームとしてまとめる際に重要な項目を取り上げた．

#7 の要求分析は，ものづくりの際に，独りよがりにならないよう，受注者・発注者について意識することを目的とした．

#8 の情報共有では，到達目標 (5) に挙げた IT スキルの観点から，チームで作業する際のデータ共有にフォーカスして項目を作成した．

#9 の問題解決については，社会で求められる資質・能力にもある項目でもあり，プロジェクトによるプログラミング入門を通じて多様な考えとアプローチを体験的に学ぶことにより，解が 1 つでない社会への接続を意識することを目的とした．

なお，このルーブリックは，本研究のために実践授業後に作成したことから，今回の実践においては，学生には公開していない．今後の実践授業においては，履

修者に公開して活用する予定である．

6.4.2　ルーブリックに基づく評価

提案するカリキュラムの実践授業を評価するために，授業の最終回に実施したレポート (#3〜#9 で使用) と提出されたプログラム (#1,#2 で使用) を用いて評価した．結果を，表 6.7 に示す．ルーブリックの表は，履修した学生には事前に参照させていない．また，ルーブリックに示した 9 項目に関して回答を求めた訳ではないことから，学生の回答結果には偏りがある．しかし，学生が授業を通じて一番印象に残っていることを率直に書いていると考え，学生が 15 回を通じてどのようなことが印象に残っているか，どのようなことを学んだかを判断する資料になると考える．データは，履修者 17 名中 16 名の回答を用いた．

プログラミングに関しては，自分で考えたイメージをプログラムで表現でき，課題の条件である，制御構造や抽象化を活用して作品が完成していることから，#1,#2 は，全員が達成できていると判断した．役割の認識やチームビルディングの部分に回答が集中しており，作品を作る課程で PBL を組み合わせた効果があったと推察する．要求分析，問題解決については，最終レポートの記述からはルーブリックに該当する項目が書かれていないものもあり，読み取ることができなかった．#3〜#9 の項目は，回答数の多い少ないはあったが，誰も回答されてない項目はなかった．

この結果から，提案のカリキュラムを通じて学生は，少なくとも 1 つは学んだ項目があると考えることができ，それぞれの学生が自分に合った場所で貢献し，自分に合った学習成果を持ち帰ることができたと解釈できる．

6.5　プログラミングの概念の理解と活用についての検証

6.5.1　プログラミングの課題の分析方法

制御構造は，プログラミングの学習をする際に，最も基本的な事項である．第 5 章で示した 2017 年度の実践授業では，制御構造と抽象化の活用は，プログラミングパートでは概ね理解され，テキストの課題では，実行できたもの見受けられたが，その後のプロジェクトパートでの活用はできなかった．

この結果に基づき，6.2 節で教材の見直しを行った．視覚的に分かりやすい画像生成のコンテンツは同様のものを使うが，プログラミングパートの構成を初回から描画中心とした授業カリキュラムへと再構築し，実践授業を行ない検証する．

以下の節では，このカリキュラム修正後の実践授業において，プログラミングパートのまとめとして，個人で行った中間課題とチームで行った最終課題を取り上げ，第 4 章と同様に，提出されたプログラムからメソッド，分岐・ループの理解

表 6.7: ルーブリック評価における結果

ID	項目	レベル	回答数 (人)
#1	プログラミングの仕組み理解	1	0
		2	0
		3	16
#2	プログラム作成	1	0
		2	0
		3	16
#3	アルゴリズム	1	2
		2	3
		3	3
#4	計画	1	1
		2	4
		3	1
#5	役割の認識	1	10
		2	4
		3	0
#6	チームビルディング	1	5
		2	3
		3	3
#7	要求分析	1	3
		2	0
		3	0
#8	情報共有	1	2
		2	3
		3	0
#9	問題解決	1	3
		2	0
		3	1

表 6.8: 提出された中間課題のプログラムについて

ID(GID)	メソッド	ループ	メイン	ループ
1(A)	2	0	1	1
2(C)	0	0	2	0
3(B)	3	0	1	1
4(D)	3	2	3	0
5(C)	0	0	3	0
6(B)	2	1	1	0
7(A)	4	0	4	0
8(A)	3	0	1	0
9 (C)	0	0	1	0
10(C)	0	0	5	0
11(B)	0	0	1	0
12(B)	0	0	1	1
13(D)	2	0	1	1
14(D)	6	4(if:4)	1	1
15(D)	1	0	1	1

について分析する．

6.5.2 中間課題の分析

本節では，プログラミングパートのまとめの課題として，6 回目終了後に課した中間課題を取り上げる．中間課題は，前半で学んだ繰り返しや分岐および手続き（丸や四角を逐一プロットするのではなく，まとまった図形はメソッドにする）を使い，絵のデザインをし，トイレットペーパーの外装をデザインすることとした．絵のデザインは，縦 11cm 横 32cm に収まるサイズとし，1 人 1 作品を作る．

履修者 17 名中 15 名の提出があった．この結果について，表 6.8 にまとめる．メソッドについては，15 名中 9 名の学生が実践できていた．ループを用いたプログラムは，8 名の学生であった．絵のパーツのメソッド内で活用しているものが 3 名，メインプログラムで活用しているものが 6 名いた．分岐を用いたプログラムは，1 名のみであった．制御構造もメソッドも活用できていない学生は，5 名いた．提出されたコードを詳しくみると，この時点では，メソッドの作り方が不十分なものもあったが，1 回目の実践授業と比較して，メソッド作成やループの活用ができていることがこのデータから読み取れる．この結果から，プログラミングパートの進め方の修正が効果的に働いたと解釈できる．

表 6.9: ループの出現回数

GID	メソッド	ループ	メイン	ループ
A	10	6	3	0
B	9	3	1	0
C	4	4	1	12
D	5	3	4	1

6.5.3 プロジェクトパートの課題の分析

後半のプロジェクトパートでは，チームごとに，クリアファイルとアニメーションの 2 つの課題に取り組んだ．ここでは，その課題のうち，クリアファイルのデザインのプログラムについて調べる．絵のパーツのメソッド内で用いられたループと，実行するメインプログラムでのループの出現回数をカウントした結果を表 6.9 に示す．分岐を使ったものがなかったため，表にはループのデータのみを用いた．

中間課題で，メソッドもループも活用できていなかった学生が 5 名いた．この 5 名のうち 4 名は，C チームのメンバーに偏っていたが，表 6.9 から，最終課題では，これらの活用ができていることが読み取れる．

最終レポートの回答には，

> 「自分がわかることを教えあったり，わからないことを教えてもらったりととてもいい関係が築けました.」

という回答が複数あった．中間課題では，うまく活用できなかった学生も，この回答にあるよう，チーム内で教え合うことにより理解を深めることができ，全員が制御構造とメソッドを使うことができるようになったと推察できる．

次の回答からも，メソッドや制御構造に対しての理解に関して，読み取ることができる．

> 「ループ・メソッドのプログラムを覚えてからは，それらを上手く利用して出来るだけシンプルなプログラムを作れるように心がけました.」
> 「15 回を通して手続きやループの仕方が分かり，作る中でどこをどうすれば，より自分のイメージに近づけるのか考えてプログラミングを作成出来るようになり，1 回目の授業と比べてテキストを応用して作れる様になった自分に驚きました.」
> 「同じ絵が書けるプログラムでも，一つにまとめたり，引数を使ったり，少しでも短く，分かりやすくする工夫を学んだ.」

これらの回答をまとめると，以下のことが言える．

- 15 回の授業を通じて繰り返し学習する効果
- テキストをそのまま真似るのではなく，考える態度

- 多角的・多面的なもの見方

中間課題では，制御構造やメソッドがうまく活用できなかった学生も，これらの回答にあるよう，繰り返し学んだり，チーム内で教え合うことにより理解を深めることができたと推察できる．リアルなものづくりの体験を通じて，自分の考えを理想の形にするための考えを形にするプロセスは，1 回の講義と演習では身につけることが難しいものも，視覚化と反復練習により理解が進み，スキルの定着につながる効果を与えたと考える．

6.5.4 描画教材による抽象化とパラメータの理解への効果

プログラムを作成する際の重要な項目として，抽象化とパラメータがある．実践授業を通じて，対象とした学生の特徴として，具体的に絵を描く傾向があり，抽象化したメソッド作成ができない学生が目についた [59]．まとまりとして考えたり，後からの活用を想定して再利用を考えるなどの行為に結び付かず，簡単にできることに流れてしまう傾向が見られた．

中間課題では，6.5.2 節で述べたように，抽象化することがうまくできない学生が 6 名 (40%) いた．提出されたコードを評価したところ，図 6.2 に示すように，ハートを描く際のメソッド化の事例が 2 件あった．

Student A:

```
def heart(x4, y4, x0, y0, x1, y1, x2, y2, x3, y3, r1, g1, b1, x, y, r,
r2, g2, b2, r3, g3, b3)
    fillrect(x4, y4, x3, y3, r3, g3, b3)
    filltriangle(x0, y0, x1, y1, x2, y2, r1, g1, b1)
    fillcircle(x, y, r, r2, g2, b2)
    fillcircle(x, y, r, r2, g2, b2)
  end
```

Student B:

```
def heart(x, y, u, r, g, b)
  fillcircle(x, y, u, r, g, b)
  fillcircle(x+18, y, u, r, g, b)
  filltriangle(x-9, y+5, x+10, y+20, x+27, y+5, r, g, b)
end
```

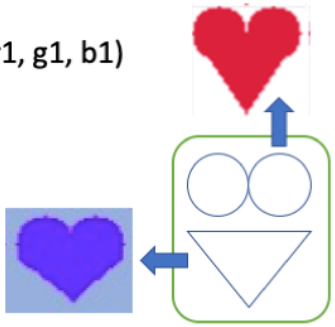


図 6.2: ハート形の描画方法の例

2 人の学生は，ともに 2 つの円と 1 つの三角形を使用し，ハートを描いた．しかし，学生 A は，メソッドと引数について明確に理解していなかったことから，全ての引数を列挙していた．そのため，取り扱う引数の数が多くなり，途中で混乱

が発生した．このプログラムは完成せず，正しく動いていない．学生 B は，円と三角形の関係をうまく抽象化することにより，少数のパラメーターで構成されるメソッドを作成した．

プログラミング初学者は，絵を描く課題では，学生 A のように，設計をせずに必要なパーツを単にプロットしてしまうだけで，データ構造を検討しない傾向がある．著者は，クラス内でこの例を共有して，クラス全体で考える時間を設けた．テキストの例題に取り組む際に，方眼紙を用い，設計することをも説明を加えているが，授業内およびテキストの演習問題だけでは全員が理解を深めることができない．また，授業後に課題を与えてはいるが，定着するまでには至っておらず，中間課題においても，理解が十分でないことがわかる．

具体的な事例を示すことで，学生は，抽象化とパラメータについて理解を深めることができ，また，チームで取り組むことにより，全員の理解が進んでいることを，6.5.3 節で示した．

このように，教室内で，共通の問題を共有して考えることは，学生の興味やモチベーションを高めることにつながり学習が促進され，テキストと演習課題のみでは困難であった知識の定着に繋がったと考える．

以上のことからまとめると，前半の教材を再考し，修正したことにより，2017 年度の実践授業で課題となった点は改良でき，学んだ知識を活用することができた．しかし，一方で，制御構造の活用について詳しく調べると，分岐の機能は使われていないことがわかった．

この点については，前半のテキストの演習課題などの内容や分岐を使う絵が思い浮かばない，という絵を描く課題そのものが要因として明らかになっている [128]．この点については，今後，テキストや課題の設定の見直しを行い，改善策を検討したいところである．

6.6 描画教材と PBL による効果の検証

6.6.1 検証の方法

本節では，学生たちがどのようにプロジェクトに関わり，どのようなことを理解したかについて，検証する．

プロジェクトを行う目的として，著者は，以下を掲げてきた．

チームで協力して取り組むことで，1 人ではできないプログラム作成に取り組むこと (相互学習) ができ，この経験を通じて，社会で求められているジェネリックスキルを身につける．

情報教育とジェネリックスキルの関係については，[89][88][91] に示されている．しかし，学習者がそのスキルをどのように滋養し，どのように定着したかを判断することは難しい．その 1 つの方法として，PBL が有効であると考え，ソフトウェア開発プロジェクトにチームで取り組むプロジェクトパートを構成した．

2017年度の実践授業において，プロジェクトパートのカリキュラム設計には，大きな課題はなかった．しかし，前半のプログラミングパートの進め方の課題や実践時の天候の事情などによる障害があり，計画通りに進めることができなかったことから，プロジェクトパートの有意性を明確に示すことができなかった．2回目の実践授業では，15回の授業が設計通りに進めることができたことから，提案のカリキュラムと描画教材とで，履修者がどのように学んだのかについて，授業のフィードバックおよび最終レポートの提出物からの記述を用いて調べる．

6.6.2 コードレビューと他者からの学び

後半のプロジェクトパートにおける効果として，相互の学びがある．

カリキュラムの設計においても，これを意識し，3回の発表の場を設けた．特に，実践授業において13回目で行ったコードレビューに解のない問題へ取り組む際の思考に関する点で有効な事例があった．

プログラミング教育やものづくりによるPBLの授業はさまざま行われているが，授業時間が限られているため，十分な時間を取ることは難しい．そのため，一度アウトプットしたものを再考したり，改良したりすることは容易ではない．しかし，中間発表やプロジェクトパートのコードレビューなどでその時間を取り入れることができた．

ここで行ったコードレビューは，各自が作ったパーツを持ち寄り，出来上がっていない部分は，相談して解決すること，出来上がっている部分は，プログラムの改善の余地があるかどうかを検討することを目的とした．実社会の現場で行われている方法とは異なるが，授業内では，コードレビューについての説明は行ったうえで，このような方法を取った．

図6.3に示すようなビルを描く際に，この学生は，当初，三角形のみを用いていた．なぜ三角形のみで分割したのか，他の方法はなかったのかを問い，クラス全体で考え，再考する時間を設けた．

この学生は，三角形のメソッドのみを利用することにより，構造がシンプルで簡単だと考えたようである．しかし，コードレビューの結果，三角形と長方形を使うことで，プログラムの行数が減ることに気がついた．1つのメソッドを用い，長いプログラムを書くことが良いのか？2つのメソッドでプログラムの行数を短くするのが良いのか？どちらもメリット・デメリットを含んでおり，正解はない．この事例は，非常に単純な例の1つであるが，描画教材では単純な図形を組み合わせることで，視覚的に考えることが容易になり，様々な構成の方法を検討できるメリットにつながった．コードレビューの時間を設けたことにより，再考する時間を確保ができ，より深いグループでのディスカッションにつながった．

学生のレポートからは，以下の回答が得られた．まず，このプログラムを担当した学生の回答である．

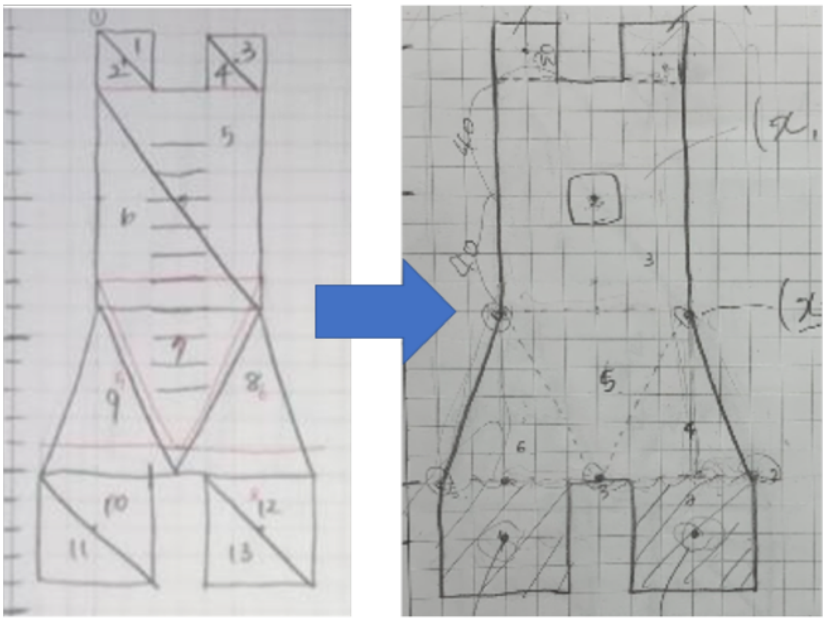


図 6.3: コードレビューによる再考

> 「私はランドマークを担当しました．初めにランドマークをどのように分割するかを考えた時に私は全てを三角形で分割しましたがそれはかなり無駄な工程が多すぎるということを知り，コードを作る上で，絵の形をいろいろな角度から見ることでいろいろな分割の仕方があるということを先生に教わりました．1 番無駄なく絵を書くことがミスにも繋がらなくなるのかなと思います.」

また，他の学生からは，次の回答が得られた．

> 「できるだけコードを少なく描くということを心がけて書いた（つもり）ものを，まだ省略の余地があるということを知り驚いた.（後で見返すと，わざわざ複雑なコードで書いていた.）コードを書く際，様々な方向から見ることが重要なのだと気づかされ，その後，どうすれば少ないコードで書けるのかを意識し始め，おそらく，以前よりも簡略することができるようになったと思う.」

当事者の学生からもそうでない学生からも，回答が得られ，学生自身が関わる具体的な事例を提示することにより，相互学習へと発展し，クラス内での理解が得られたと推察できる．構造が簡単ないくつかの図形を組み合わせて絵を描く行為は，それぞれの考えを視覚化することが容易であり，教室内での共有や再考についても扱いやすく，限られた時間であっても学生の理解に有効に働いていることがわかった．

その他にも，最終レポートからは，次のような記述があった．

> メンバーの仕事をこなしていく態度であったり姿勢をみていくうちに自分でも要領がわかってきた，できるようになったことが沢山あった，ことも事実でした．教科書に書いてあることを自分で読み解いてみることはもちろんですが，自分でメンバーの作品を分析して学んでやり方がわかったなんてことがあったのは，恥ずかしながら自分の中では今までにない体験だったように思います．自分で検索して学んだり教科書から学び取ることは非常に重要なんだということはわかりました．

文系大学では，講義が中心で個人の学びに閉じていることが多く，先生の話を聞くだけや，教科書を読み解くような知識移転型が多い．自分で考えて対応する場面が少なく，他者から学ぶようなことも少ない．

変化の激しい情報化社会においては，1つのスキルを身につければそれで終わりではなく，変化する社会に応じて新たな知識を吸収したり，スキルを高めたりしていく必要がある．そのためには，教科書に書いてあることを読み解いて学ぶだけでなく，様々な学び方が必要である．教科書だけでなく，自ら検索して学ぶ姿勢や，メンバーとの関わりの中からの学びはこれからの社会での活動にプラスに働くであろう．

チームで働く力は，学士力・社会人基礎力でも求められる力として示されている．「情報教育課程の設計指針」においても，対応するジェネリックスキルとして示されている．本実践授業においては，新たな知識としてのプログラミングを学ぶことにより，適度な困難を伴いながらチームでのプロジェクトの実践を通じ，他者からの影響をうけ，チームで働く力が育成されていると推察できる．

6.6.3　コードを綺麗に書くこと

コードをきれいに書くこと，読みやすく書くことについて取り上げる．個々に課題や宿題に取り組む際，特に，入門レベルであれば，記述するコードも数行のみの簡単なものであることから，コードの書き方については，あまり議論にはならない．しかし，チームでの開発を伴う際には，コードを綺麗に書いたり，コメントを入れたりすることは重要な要素の一つとなる．

これに関して，次のような回答があった．

> 「最終課題までプログラミングをしてきて，重要だと感じたのは，なるべく分かりやすくて綺麗なプログラミングを作成する事です．シンプルにまとまっていた方がエラーの出る確率も低いですし，万が一エラーが出ても，どこがおかしいのかを見定めやすくなります．また，後半のグループワークのように，他の人にも自分のプログラムを見てもらう必要がある時など，シンプルにまとまっていたり，コメントを残

してあるプログラムの方が，第三者が見てすぐに作業に取り掛かりやすいと思いました.」

そして，これと同様の回答は，上記以外に 4 件あった.

Kernighan の『プログラミング書法』[19] の最初の章に,「わかりやすく書こう」とある．プログラミングパートの演習問題では，個々に取り組むものであり，数行のプログラムで対処できてしまうことから，コメントの利用は見られなかった．初学者が，初めからこの意識を持ってコードを書くことは，授業時に解説しても，なかなか実践に結びつかない．しかし，この学生の回答からもわかるよう，チームでプロジェクトに取り組んだことで，第三者に対して配慮が必要なことから，コードをわかりやすく書くことへの配慮やコメントの活用についての理解が進んだと推測できる.

Boswell らの『リーダブルコード』[18] でも，コードは理解しやすくなければならず，また他の人が短時間で理解できるように書かなければいけないと言われている．プログラミング言語を学ぶ際，さまざまなテキストで，可読性や保守性について書かれているが，テキストで学んだだけではなかなか定着しない.

本カリキュラムでは，入門レベルを対象としていることから，高度な内容を扱っているわけではないが，実際にコードを書き，ソフトウェア開発の要素を取り入れたことにより，リアルな体験と知識の定着が図られたと考える.

6.6.4 設計図を書くこと

本論文では，ソフトウェア開発プロジェクトの要素を加えることにより，ものづくりの体験を含めたカリキュラムを提案した．プロジェクトをを行う際には，ゴールを定め，どのようなものを作るのかの計画を立てる必要があり，その中でも設計図の作成は重要である.

設計図の重要性，グループで取り組む際の配慮として，以下の回答があった.

「方眼紙に書くこと，考えをまとめて整理すること，何のためのプログラムなのかを#と説明文を入れるなどして，自分とグループのメンバーに分かりやすくすることの重要性に気付いた．図を書き，説明を書くことで，自分が行き詰まった時に協力を求めやすく，チーム内での情報共有もスムーズになった．同じ絵が書けるプログラムでも，一つにまとめたり，引数を使ったり，少しでも短く，分かりやすくする工夫を学んだ.」

この回答から，本実践を通じてプログラムを書くことだけでなく，設計図や文章でまとめることの重要性についても理解されていることがわかる.

大岩は，プログラミング教育における設計の重要性について述べ，日本語プログラミングを推奨している [50]．著者の提案による方法であれば，入門レベルの教

育であっても，汎用言語を用い，設計の要素を組み込んだプログラミング入門科目として構成できたと裏付けられる．

6.6.5　考える力・問題解決力

著者は，プログラミングを学ぶことにより，考えることやそれを活用することを目的とした．また，プログラミング言語の習得ではなく，基本的な概念の習得を目指した．以下の回答では，プログラミングの基礎概念を習得し，言語が違ってもこの授業で得た知識を今後の活動に応用して対応することの意義が理解されていることが読み取れる．

> 「授業の初めの方は，インターンの時と同様に，エラーを出す度に，上手く出来なかった事に対してショックを受けて，やはり向いていないのかもしれないなと何度も思いました．しかし今回は，1day インターンと比較すると明らかにプログラムに向き合う事の出来る時間が多い為，次第にエラーが出たとしても，何回でもやり直し，何が原因でこのプログラムは正常に動かないのか考えるようになりました．(中略) 何が原因で動かないのか考えるという習慣が付き，操作に少しずつ慣れてからは，エラーが出ても自分で少しは対処できるようになったと思います．何か上手くいかない事に直面した時に，ただ落胆するのではなく，原因を考えられるようにするのはプログラミングだけではなく，例えば就活や，これから先の人生においても大事な事だと思います．(中略) もし Python をもう一度触ったとしたら，私はまたエラーをたくさん出してしまうのだろうなと思います．しかし，この授業で言語は違えど，プログラミングの基礎知識，エラーのパターン，エラーが起きた時どのような部分に着目するとよいかなど，学んだ知識を応用すれば，あの時よりは，意味が分かって動かせるのかなと感じています．そう考えるとプログラムについて学んだのはもちろんですが，問題が起きた時の解決力も少しではありますが身に付いたのではないだろうかと思いました.」

この学生は，ワンデイインターンシップでのプログラミング経験がある．その際，うまくできなかったことから授業を履修した．繰り返し学習する時間の必要性も大事な要素である．エラー処理を繰り返す中で，行動に変化がうまれていることが読み取れる．そして，この行為により自から原因を考え問題解決に向かう姿勢の理解がされていることがわかる．

問題に対し原因を考える行動は，この回答の他にも 6 名が同様の回答している．対象とした学生たちは，仕組みを理解し，原因を考えるような学びの場や経験が

これまでにほとんどない．エラー処理については，初学者の躓きの原因の 1 つになることから，これを避けるような事例 [22] もある．しかし，本実践では，エラーの対応をする経験を積み重ねることにより，問題が起きた時の解決力の理解につながっていることがわかる．

能動的学習 [23] では,「学習者が自分の学習の過程を自分自身でコントロールする能動的学習は，学習者が理解の程度を自分自身で認識したり，他者の意図を正しく理解しているかどうかを自分で確認したり，あることを主張するためにはどのような検証が必要であるかを認識したり，自分自身で構築した理論を自分自身で検証したりできるようになることを重視する.」とある．

描画教材によるプログラミングの学びは，学習者の考えや理解の程度を出力結果から可視化しやすく，チームでものづくりに取り組むことにより，他者の意図を正しく理解しているかどうかを確認することも容易であり，主張や理論を検証することにもつなげることができたと考える．

入門レベルのカリキュラムであることから教材にはたくさんの要素を盛り込むことはせず，比較的平易な内容であったが，PBL での取り組みとすることにより，描画教材の活用とカリキュラムの設計が機能したと考える．

6.6.6 意思表示することの重要性

さらに，以下に示すように，わからないことを明らかにしチームで共有し助け合うことについての回答が目立った．

> 「これはプログラミングの授業だが，学んだものはプログラムの書き方やコンピューターのことよりもむしろ，できることもできないこともはっきりと言うことの大切さや今の自分がいかに狭い世界しか知らないかというような自分のスキルアップや人生を充実させるために必要な心の持ち方についてのことが大きかった.」
>
> 「とにかく時間が無制限にあるわけではないので，どうしても解決までに時間がかかりそうな問題は，メンバーや先生に，「分からない」と助けを求めることも大事だと思いました．効率よく進めるため，でもありますが，実際に自分がどのような状態で，どれくらいの進行状況なのか，を示すことが非常にチームとして重要だったんだなと思いました.」
>
> 「チームでのプロジェクトを通して，自分が分からない事を説明できる力を少し身につけられたのではないかと思います．普段は分からないと感じても自分で何とかしようと思いがちですが，チームでやる事，チームで完成を目指すものであれば，自分が分からない・出来ない事をきちんとメンバーに分かってもらう，そして，その上でどのように

> 対策を取ればよいのか話し合う事がチームの為にもなるという事を学び，上手く出来ない事を認め，アドバイスを頂く事が出来たと思います.」
> 「分からないことや，出来ないことがあった時に自分でテキストを見返したり，何度も繰り返してみたり，限界まで頑張ることも大切だが，「できない」や「分からない」などと意思表示をすることの重要さを学ぶことができました.」

社会においては，正解は１つだけではなく，また，全員が同じ回答を求められることは少ない．それよりもむしろ，わからないことやできないことを説明して助けを求め，相互にコミュニケーションをはかり問題解決に取り組むことの方が重要である．

プログラムは，実行することにより，結果の良し悪しがその場で判定できる．特に，本研究で用いた描画教材は，自分たちの考えた絵が結果として出力されることから，設計から結果の出力までが，視覚的にわかりやすい．さらに，どの図形を組み合わせて絵を構成するのか？という設計や，プログラムの実行時のエラーの出力など，結果だけでなく，その過程においても可視化でき，チームでの活動に有効であったと推察する．

プロジェクトでは，ゴールが明確であることから，進捗を管理する上でも有効なコミュニケーションになっていたことも読み取れる．

また，次のような失敗を共有することによる学びについての回答もあった．

> 「「失敗作の共有が大切」というのは新しい気づきで，心に留めておこうと思いました.」

これは，中間発表やコードレビュー，最後のまとめの発表会を行ったことによる効果が大きいと考える．失敗を共有し，チームや教室全体で原因を考えることにより，問題解決力の育成につながる．著者の働いている環境では，単にグループで取り組み，まとめたものを発表するであったり，商品ができたであったりと，結果重視でそれを発表し評価することが多い．そのため，どのように失敗して，困っているのかや改善したのかを共有して考える場がほとんどなく，このような経験をこれまでにできずにいた学生がたくさんいるということがわかった．

個別学習においても進め方を工夫することにより取り組むこともできると考えるが，チームで１つのものづくりに取り組むことは，より効果的であり，さらには，描画教材とプロジェクトを取り入れたカリキュラムの効果であったと考える．

6.7　2回の実践授業を通じてのまとめ

6.7.1　プログラミングの概念の理解と活用

本研究では，高等教育における誰もが学ぶべき科目として，プログラミング入門にチームによるソフトウェア開発プロジェクトの要素を含めたPBLによるカリキュラムを構築した．高等教育における最小の時間として，90分15回で実施する構成をとった．前半の6回でプログラミングの基本を学生それぞれが個別に学び，後半の9回では4〜5名を1チームとし，プロジェクトを行った．本授業では，視覚的にわかりやすい描画教材を用い，プログラミング言語Rubyを用いた．

本提案の授業の目的は，これからの社会で誰もが必要とされるであろうプログラミングの基本的概念を学び，その知識をどのように活用するかを体験的に学ぶことである．授業の位置付けは，プログラミングの入門科目であり，そのスペシャリストを養成することが目的ではない．

RQ1として,「PBLにより，プログラミングの概念の理解と社会で求められる資質・能力の育成をはかることができるか？」という問いを立てた．

そして，この問いに答えるために，2回の実践授業を行った．

プログラミングの概念の理解と活用については，1回目の実践授業では，提出されたプログラミングパートの演習問題の状況から，ある程度理解されたように見えたが，活用するところまでは至っていないことがわかった．その要因は，テキストの構成と例題や演習問題の内容にあることがわかった．

そこで，要因として明確になった点を修正し，2回目の実践授業を行った．プログラミングパートの課題だけでは，十分理解されておらず，活用できていない学生がいたが，チームによるプロジェクトを行うことにより，全員がプログラミングの概念の理解をし，活用するところまでできたことが確認できた．

Nuutilaらの先行研究[24]でも言われていたように，プログラミングスキルを習得するためには演習が必要であり反復練習が必要であることから，プログラミングパートで十分習得できなかった内容については，プロジェクトパートで実践的に身につける本提案の構成に効果があったと考える．

また，個別に取り組むのではなく，ペアやチームでの取り組みによるPBLを行うことにより，中山のいうピア・ラーニング[85]を取り入れることにより，機能したと考える．

そして，PBLによるリアルなものづくりの体験により，試行錯誤を繰り返し考える行為と完成した際の達成感により，知識の定着がなされたと考える．

6.7.2　PBLによる効果

本研究では，専門の専攻ではない学生の入門レベルのカリキュラムに，ソフトウェア開発プロジェクトによるPBLを導入した．入門レベルで，プログラミング入

門とソフトウェア開発プロジェクトを同時に行うようなカリキュラムはこれまで存在しないことから比較するものがないが，学生の回答に次のようなものがあった．

> 他の PBL 系科目は，文系大学ということもあって，上手く行かないという状態にぶち当たることがそれほどない．しかしプログラムは，正しく書かれないと思ったものとは違うものが必ず返されてしまう．この授業のテキストの小問やプロジェクトではそうしたエラーなどが何度もあり，その度に見直し・修正を行った．どんな物事でも一回で完遂しようとせず，工程を細かく分けた上で見直し・修正を繰り返し，完成に近づけるべきだと，授業から学ぶことができた．

近年，様々な PBL が行われているが，この学生は，必ず成功するようなゴールが設定されている商品開発を経験していた．本実践授業においては，プログラムを書くことを通じて問題に遭遇し，それを解決するための思考や，プロジェクト実践から，一度作って終わりでなく，見直しや修正の重要性を認識している．

社会における活動では，簡単に解ける問題はなく，試作や修正を繰り返して製品化するプロセスが重要であるが，15 回の授業で実施する PBL では，試行錯誤をする時間を取ることが難しく，決められた狭い道を進むだけとなってしまうことが多い．しかし，プログラミングは，コードを実行する課程でエラー処理をしたり，実行結果を精査して見直しや修正のサイクルを経験することが容易である．簡単な描画教材であったが，実行結果は視覚的に分かりやすく，その分，学習効果が高かったと考える．

次に，プロジェクトの運用に関することについて，学生のレポートの回答からは，チームワーク，役割分担，スケジュール，マネージメント，多様性，情報共有，ツールの使い方など様々な気づきの学びがあったことが読み取れる．チームワーク，役割分担，スケジュール，マネージメントに関する点について，以下に示す．

役割分担に関しては，次の回答があった．

> 今回の授業で特によかった点として，いかに効率よく，時間の無駄が無いように，役割分担をしてプロジェクトに取り組めたかというところだったと思う.（中略）みんなが同じ仕事量で得意な分野を生かして最後までチームワークを発揮できたと思う．

チームで取り組む際には，できる人だけが手を動かし，やらない人が出てきてしまうことがよくある．学校教育の中では，全員が一定のスキルの習得を目指すが，本実践においては，ある程度チームの裁量に任せた．このチームは，仕事量は同じになるような配慮をし，各自の得意とする分野を活かして取り組んだ様子がわかる．効率よく，効果的に仕事をする上では，できない人にできないことを求めるのではなく，このような得意なところで役割分担で仕事をする理解も求められる．役割分担とチームビルディング，メンバーの多様性の理解が認識された回答であると評価する．

> 私たちのチームは皆がプログラミングする公平な分担をした．プログラミング作業が滞ったが発表が得意というチームメイトは急遽発表者の役回りに立つなどその時の場合によって物事に対応した．滞ったプログラミングの分は得意な人がサポートするなどして補った．

この回答からは，公平な役割分担とスケジュール管理の観点から，作業を見直しチームを築いていったことが推察できる．

チーム力は，前述したように，学士力にも社会人基礎力にも共通して求められている力である．チームでよい成果を出すために，自分の意見を相手に的確に伝えるとともに，立場や背景の異なるメンバーを尊重して集団の一員として行動する能力が求められている．また，文部科学省の知識基盤社会が求める人材像の資料 [107] には,「産業構造の変化も急速に進んでいる現代においては，多種多様な個々人が力を最大限発揮でき，それらが結集されるチーム力が必要とされている.」とある．

この 2 件の回答は，知識基盤社会が求める人材像の要請と合致している．15 回の授業で完全に習得できたとは言えないが，体験的に学び，このように回答されている点では，今後の気づきになっていると推測し社会活動に活かされることを期待する．

チームで活動する中で，チームビルディングだけでなく，チームのリーダーとしての要件も求められる．ジョセフは,『ROBOT-PROOF』[5] で，最近の大学卒業生に最も要求されるスキルとして「リーダーシップ」を挙げている．回答者の 80%以上がリーダーシップを応募者の履歴書で確認すると答え,「チームで働く力」(79%) がそれに次いだ．これは実世界での他者との共同によってに身につく社会的スキルである」と述べている．

本実践授業を通じて得られたリーダーシップに関する回答として，以下があった．

> この授業を通してグループで一つのプロジェクトに取り組むということがとても好きだと感じました．リーダーという立場にいても，下から支える立場にいても，どちらにも適応できる能力を身につけたいと思ったし，今回の経験で気づいたことや細かな反省点を活かしてまた別の場でもグループで一つのプロジェクトを成功させる経験をしたいと思いました．

文章の記述からは，細いことが読み取れないが，チームでの活動を通して，リーダーシップやチームで働く力に関して，気づきを持たせ，次のステップに繋げることができたと考える．

また，次のような回答もあった．

> どうしてもチームで作業するときにやるんだったら自分たちができる最高を目指したい，あきらめたくないという気持ちが突出してしまい，チーム員と足並みをそろえてすすめられませんでした．もっと周りをみて行動することを覚えたいです．

この学生は，リーダーをしていた．自分の思いが強く，1人で暴走してしまい，チームのメンバーがついて来れなくなってしまった部分があった．気づきがなければチームとしての課題は出来上がらず，リーダー1人の作品になってしまった可能性がある．しかし，チームが崩壊する前にこのことに気づいたことから，チームを立て直し，チームのメンバーのスキルの違いを認識して作業分担を見直すことにより，チームを立て直すことができ，プロジェクトの課題は計画通り出来上がった．

高度なものを作ることも大事ではあるが，本実践授業でのものづくりを通じ，学生たちは，メンバーの多様性を理解し，チームで取り組むプロジェクトの進め方を理解したと考える．

社会で求められる資質・能力が求められるが，これらをどのように学習し，身についたかをはかることは，難しい．しかし，本提案のカリキュラムと描画教材により，より具体的に理解が進むであろう．そして，プログラミングは，チームによるプロジェクトを組み合わせることにより，これらの資質・能力を育成する教材として有益であると考える．

6.7.3 視覚的教材の活用と思考

プログラミング入門では，視覚的に分かりやすいとされる描画教材が用いられている．日本では，日本語ベースのプログラミング言語を用いたPENが多く使われている[86][127]が，本実践では，汎用プログラミング言語を用い，描画教材として丸・三角・四角などから構成される図形生成のメソッドを用いた．

1回目の実践のレポートの学生の回答には，次のような回答があった．

> プログラミングの細かい内容よりも絵などから入っていったほうがやはりわかりやすいのではないかと思います．文章を先に学んでも活用の仕方が難しいです．

1回目の実践授業では，数学的な例題を用いて導入をお行い，プログラムの実行手順に慣れてから描画教材を取り入れたが，この回答に現れているように，文系の学生にとっては，数学的な例題よりも描画教材で可視化して学ぶことで，理解が進んだことがわかる．

別の学生は，次のように回答している．

> 数学的要素が強かったと思う．学校の数学は嫌いではなかったが，応用的なモノが全くできなかった為，本質を理解していなかったという事がよくわかった．自分の考えたプログラムが動かない時，何故動かないのか，どこのどの数字をどう変化させたらいいのかなど考えるのは大変だったが，それなりに楽しかった．上手く動いた時はとても快感だったし，少しずつ理解していってるのが自分でもわかったのでより楽しかった．

この回答からも，数学的要素はハードルがあることがわかる．しかし，この学生は，これまでなぜ，応用的なものができなかったのか，この授業を通じて解決できたと言っており，思考するプロセスでどこで躓いているのかを実行時のエラーや実行結果で視覚的に確認することができ，可視化による効果が理解につながったと推察する．

M. リチャートらも,『子どもの思考が見える 21 のルーチン：アクティブな学びを作る』[27] で,「思考の可視化は，重要な評価の場面を作り，また子どもの理解を深めることも助けるのである．思考を可視化することには，他にも目的がある．それは，思考と学習の関係を示すことであり，それによって考えを理解するということは，どういうことか，考えるということは，どういうことか，学習とはどういうことかについてのモデルが伝わる.」と述べており，描画教材を用いたプログラミングの学習は，実行過程の可視化により学習者の思考と理解を明確化できたと考える．さらに，理解が明確になることで，学習者のモチベーショを向上させ，満足のいく経験になったと推察する．

6.7.4　社会活動としてのプログラミング

プログラミングを学ぶ過程で，視覚化と思考の組み合わせにより “考える” という行為に効果的に働いたと推察できる次の回答があった．

> 「「コードを打つ際,「何を表して欲しいのか」「自分は何をかいているのか」を考えることが重要であるということを学んだ.」

Chad は,『情熱プログラマー』[14] で,「コーディングばもう武器にならない」と述べている．技術的スキルは決して高くないデータベース管理チームの事例を挙げ，彼らは，技術的なスキルだけでなくビジネスについてアナリストたちよりも詳しく理解していたため，社内で絶大な価値を認められていたと書かれている．Chad の事例を広く解釈すれば，コーディングよりもそれをどこでどのように使うのか，利用者の視点にたち，その内容を深く理解し対応する必要があることがわかる．今後，ソフトウェア開発の方法は，時代とともに変化し，ノーコード開発 [3][1][10] によるコードを書かない開発が勢いを増している．

コーディングを単に学ぶだけでなく，プログラムの概念を学び，ソフトウェア開発の工程を少しでも体験し，開発の仕組みの理解を深めることにより，これからの社会活動における活躍の場が広がると考える．

また，Weinberg は,『プログラミングの心理学』[34] で，社会活動としてのプログラミングについて次のように書いている．

> プログラマは，孤立して働いたりはしないのがふつうである．
>
> (中略)

> ここではプログラマの集団を三種類にわけてみる．すなわち，グループ，チーム，プロジェクトの三種類である．
>
> (中略)
>
> 一方，プログラミングチーム (Programming team) とは，一つのプログラムを共同で開発しようとしているプログラマ集団である．(邦訳本 P.92)
>
> プログラマの集団をさらに四半世紀にわたって行ってきた観察の結果として，私は「チーム」に関する右の定義を変更したい．いまならば，こういいたい.「プログラミングチームとは，共同して働くことによってよりよい製品を作り出そうとしているプログラマたちの集団のことである.」
>
> すなわち，チームをチームであらしめている特性は，メンバーたちが一人では作り上げることのできない (または，効率よく作り上げることができない) 製品を共同して作り上げる，そのあり方にある．

Weinberg は，20 年の時代の変化を振り返り，単に共同で開発している集団から一人では作り上げることができないものをそれぞれの知識や能力により共同して作り上げるチームについて修正している．

このように，社会変化に伴い，教育する内容も変化していく必要がある．専門でない誰もが学ぶ入門レベルの教育では，限られた知識，限られた時間の中で取り組まなければならず，本格的なソフトウェア開発には取り組むことができない．ソフトウェア開発では，細分化されたパーツを個別に作り，それを組み合わせることによって大規模なソフトウェアが出来上がっていくことから，著者は，一つの絵をソフトウェアに見立て，それぞれのオブジェクトをパーツに分け，統合して一つの絵を完成させることで，擬似的なソフトウェア開発プロジェクトを模倣できると考えた．これにより，絵の構造を考え，役割分担をし，チームのメンバーが与えられた場所で寄与する行動ができるようになる．

これまでの多くの教育は，知識蓄積で行われるものが多く，また，大学においても特に文系の学部においては同様であった．

小林は,『変貌する高等教育』[67] で，大学で何を学ぶかについて述べている．大学という場を知を知識ではなく，知の行為という視点から考え直し，次のものを必要としていると述べた．

- 一領域の専門知識にすぐれたエキスパートの知性ではなく，理科系文科系を貫いて，異質な領域のあいだにまたがりながら，行為の場をみずから創造していくような総合的な，しかし具体的な現場の知性
- 「知っていること」において行動するのではなく，むしろ「知らないこと」において行動することができるような知性

- さまざまな異なった能力をもった人々とともに共同で問題の創造的な解決にあたることのできるような開かれたコミュニケーション能力を備えた「知の行為者」

また,「行為する知」の教育理念とし，次の 4 つを挙げている．

1. 現場：行為にとって最も重要なファクター
   いったい誰に向かって，何のために，どのような行為を行うのか，を明白に意識化すること，また意識化させることが何よりも大事
2. 言語：客観的な記述に重点を置いた言語
3. 他者：自分とは異なるものとしての存在
   その他者に対する関係のあり方に，その行為のモラルがある
4. 歓び：表現，コミュニケーション，創造，発見などがもたらす無償のもの
   知や学問の中核にあるものであって，みずからが能動的に行為することによってしか獲得しえなく，教育はこの歓びを伝えなくてはならない

著者は，プログラミングは，自ら考え能動的な学びの場を作りやすく，プロジェクトは，小林のいう現場を構築しやすいと考える．そして，ソフトウェア開発プロジェクトは，他者との関わりから，多様な仲間との活動の場を構築できる．前述したように，これからのソフトウェア開発は，コードを書かずに行われるような開発方法も出現していることから，難しい文法や技法を学び，高度なソフトウェアなどを作成する必要はなく，むしろ，シンプルな教材で基礎的概念をしっかり学ぶことが重要であると考える．

これにより，開発者の視点に立ち学習することで，学んだ知識は，この後の社会生活において，専門家と共同して働く場で有効に働くと考える．また，この知識は，ユーザの立場にあっても，これまでブラックボックスで利用していたものに対し，仕組みの理解につながり有効に機能することを期待する．

6.7.5 他者を伴って学ぶ環境

本提案のカリキュラム設計においては，履修後の活動に効果的に働くように，社会への接続を考慮し，個々の資質・能力の向上だけでなく，チームとして取り組む中でのジェネリックスキルの向上にも配慮した設計を行なった．

以下では，他者が伴うことにより，影響のあった事例を取り上げて議論する．

情報を明確に伝えること

チームで取り組む際，情報を正しく伝え，共有することは重要な要素の一つである．学視力・社会人基礎力にもコミュニケーションの項目がある．しかし，具体的にどう学べば良いのか？こういった要素を直接学べる講義は少ない．次の回

答からは，情報を明確に伝えることの理解がわかる．

> 方眼紙に書くこと，考えをまとめて整理すること，何のためのプログラムなのかを#と説明文を入れるなどして，自分とグループのメンバーに分かりやすくすることの重要性に気付いた．図を書き，説明を書くことで，自分が行き詰まった時に協力を求めやすく，チーム内での情報共有もスムーズになった．同じ絵が書けるプログラムでも，1つにまとめたり，引数を使ったり，少しでも短く，分かりやすくする工夫を学んだ．

本提案のカリキュラムであれば，コードを書き，プロジェクトを進めていく工程で絵を描き，プログラムを描くひとつひとつの作業の中で，視覚的に分かりやすく，体験伴って，誰にでも分かりやすい形で認識することができ，深い理解につながったと考える．チームで取り組むことにより，自分と他者の関係も明確になり，より実践的な経験につながったと考える．

失敗からの学び

15回目の最終発表会で，たくさんの失敗が共有された．特徴的な回答を以下に示す．

> 「失敗作の共有が大切」というのは新しい気づきで，心に留めておこうと思いました.」

これまで，学生たちは，失敗は表に出さず，成功体験中心で学んできたことがわかった．次の回答にも同様のことが表れている．

> 「これはプログラミングの授業だが，学んだものはプログラムの書き方やコンピューターのことよりもむしろ，できることもできないこともはっきりと言うことの大切さや今の自分がいかに狭い世界しか知らないかというような自分のスキルアップや人生を充実させるために必要な心の持ち方についてのことが大きかった.」

佐伯の「そもそも『学習』の概念自体も,『教え込み』の結果として個々の学習者が習得するという発想から脱皮し，市民が社会的な実践活動の中で互いに学び合うという側面を重視した概念に変わる必要がある」[69] という指摘に代表されるように，情報化社会においては，従来の学習観の転換が必要である．佐伯は，つづけて「さまざまな知的資源を活用し，具体的な実践活動の中で，他者と協同的に実現することを通して，一人ひとりのアイデンティティを発揮し，それを他者と分かち合ってゆくということに，人間の知の営みの本質がある」[69] と述べ．このような学習観を背景にした授業観の変化について，以下の3点を指摘している．

1. 教師は知識の伝達者ではなくなり，子どもたち一人ひとりがみずからの学びの筋道を見出し，学習活動の実践に参加していくことの橋渡しの役目を担うようになること
2. 「 教材」は,「教えるべきことのパッケージ」ではなく，画一的に全員が同じ知を共有することを想定したものではなく，一人ひとりが自分らしい「参加」を深めていくきっかけを提供するものとなること
3. 学習は，常に他者と交流し,「教室」や「学校」を越えた，実社会の実際の文化にふれ，そこでの文化的な価値を味わい，共感しあい，なんらかの実践活動に参加していく活動によって行われるものであること

実践授業の履修者は，どちらかというと教え込みの教育を受けてきた学生が多く，本実践授業におけるプロジェクトを通じて，自らが作成したい絵を考え，設計してプログラムに実現していく過程を通じて，暗記ではなく考えて行動することを体験した．そして，互いに学び合うことにより，最初は，作品の結果を期待して活動していたが，最終的には，結果よりもその過程が重要であるという気づきに発展している．

Carl Rogers は，著書『Freedom to Learn』[28] で,「最も意味ある学びとは学ぶ過程そのものにある．すなわち新たな経験にいつも心を開いていること，それを自分自身の内に実体化すること．学びとは変化へのプロセスのことである.」と述べており，以下の回答からは，学び方の変化が読み取れる．

> 目的やビジョンを持って取り組む重要性に気づかされました．今後は行うことをゴールとして学習や制作に取り組むのではなく， そのプロセスを通して目的を達成することを意識して取り組みたいです.

プログラミング入門においては，エラーによるつまづきを避けたがる傾向があるが，本実践を振り返ると，学生の回答からは，エラー処理を体験する過程とプロジェクトの進め方により，効果が出ていた．

相互理解と多様性

到達目標 (4) にチームメンバーの多様性を取り上げた．以下の 3 人のの回答からは，それぞれの違いを認め合い，相互に理解しあってプロジェクトを進めてきた様子が伺え，多様性について理解するきっかけになったと推察できる．

> 人と協力して何かをすることは，意見の違いや人の性格の違い，コミュニケーションのレベルなどいくつか壁があり大変なことだと思いますが，困難も仲間と共に乗り越えて，成功したときの達成感を共に味わえる人がいることはとても素晴らしいことだと思います.

> グループワークはみんなでワイワイできる点は楽しいのですが，正直苦手で，できるできない，やるやらない問題のいざこざが本当に面倒で揉めるくらいだったら一人でやってしまいたいと思ってしまったこともありました．しかし，そういう経験をしたからこそ，そういう問題が起こった時に，寛容な精神を持って，解決策を導く大切さを学んだり，こういうグループワークで，個人個人の良さやできることを見抜いたり，発見して，うまく分担することの難しさをリーダーという役目を通して，実感しました．

> わたしはプログラム経験が以前あるものの，3 人グループの中で一番時間がかかるという欠点があったため，プログラムが得意な他の 2 人が中心となってプログラミングの方は中心的に進めてくれた．だからといって，わたしはやるべきことが減ったわけではなく，その他の発表時のパワポの作成や，書記，そして，スケジュールの管理を任せてくれたので，みんなが同じ仕事量で得意な分野を生かして最後までチームワークを発揮できたと思う．

相互理解や多様性については，社会が求める資質・能力のひとつである．これらを，学ぶためのカリキュラムを個別に準備するのは難しいが，本カリキュラムを通じて学ぶことができたと考える．

また，佐伯が,「さまざまな知的資源を活用し，具体的な実践活動の中で，他者と協同的に実現することを通して，一人ひとりのアイデンティティを発揮し，それを他者と分かち合ってゆくということに，人間の知の営みの本質がある」[69] と述べているように，この授業でも，学生個々の持つ知的資源を活用し，他者との共同作業を通じてプロジェクトが完了できたことは，今後の知の営みとなると期待する．

> 目的やビジョンを持って取り組む重要性に気づかされました．今後は行うことをゴールとして学習や制作に取り組むのではなく， そのプロセスを通して目的を達成することを意識して取り組みたいです．

この回答からは，学び方の変化が読み取れる．カール･ロジャースは，著書『Freedom to Learn』[28] で,「最も意味ある学びとは学ぶ過程そのものにある．すなわち新たな経験にいつも心を開いていること，それを自分自身の内に実体化すること．学びとは変化へのプロセスのことである.」と述べており，この学生も，15 回の授業を通じて結果だけが重要ではないことを理解し，意味のある学びができたと推察できる．

6.7.6　発表の場の設定と学び方の変化

教室に集まって授業をする際，教室内でのインターラクティブな学びの環境は，教育効果を高める上で有益である．本カリキュラムにおいては，チームでの考えが纏まるタイミングで発表の場を設定した．実践授業では，3 回の発表の場を設けた．

単にチームで取り組むだけでなく，人前で発表することにより，たくさんの影響があった．特徴的なレポートの回答を以下に示す．

> 「実際に発表をしてみると，自分達の計画の曖昧さに気づかされた．他の班の発表から学ぶものも色々とあったので，それを活かしていきたい.」
> 「様々なチームの中間制作過程を知ることができ，自分のチームに取り込めることがたくさんあると思いました.」
> 「コードレビューでみんなで情報共有すると他の人のコードを見て知りえることも多かったです.」

などの回答から，他者からの学びや影響があり，モチベーションにつながっていることがわかる．

前述した小林が言う，知の行為の教育理念の “現場：行為にとって最も重要なファクター”にもあるよう，経過報告にとどまらず，計画の見直しやスキル向上につながる場の設定は，重要な要素であり，効果があったと考える．

小林は,「おそらくわれわれは，いま，そして今後，これまでとはまったく異なったタイプの知性を必要としている．(中略) 個人であれ，集団であれ，自然と備わっている欲望，自己愛，帰属意識といった「我」の「Interest」から離れることができること，その自己からの離脱において自己を表現し，他者とともに行為すること，それこそ大学が教えるべき最も初歩的にして，究極の行為論である.」[67] と述べており，上述した学生の回答から，本提案のカリキュラムは，いま，大学が教えるべき内容を備えているといえるであろう．

次の回答からも，自己表現や学びに対する変化がうかがえる．

> 「何回考えても上手くいかなかったり，プログラムの仕組みがわからなかったりした時，初めは自分以外の人に分からないと言うことが出来ず，一人で悩むことが多かった．授業を通して，分からないことは恥ずかしいことではなく，どこが分からないのか，何が原因かを考えることが重要だということを学んだ.」

さらに，次のような回答もあった．

> 「教科書に書いてあることを自分で読み解いてみることはもちろんですが，自分でメンバーの作品を分析して学んでやり方がわかったなんてことがあったのは，恥ずかしながら自分の中では今までにない体験

だったように思います．自分で検索して学んだり教科書から学び取ることは非常に重要なんだということはわかりました.」

知識の詰め込みによる教育で育ってきたことが読み取れるが，本授業を通じて，新たな気づきがあったことは幸いである．描画教材を用い，絵を描くことで，他のチームの作品がどのように作られているのか？作品を分析して学ぶことにより理解が深まっている．ほかにも同様の回答があった．描画教材は，結果が可視化できることから，誰にでもわかりやすく，プロジェクトとの親和性も高かった．

初等中等教育における学習指導要領が,「思考力」「判断力」「表現力」の育成を提唱しているが，高等教育においてもそれを引き継ぎ，継続して能力の向上していく必要があると考える．情報化社会やグローバル化の進展により，変化の激しい社会においては，状況に応じて学んだ知識を活用する能力が求められる．Ambrose らは,『How Learning Works: Seven Research-Based Principles for Smart Teaching』[2] で，丸を知識の塊 (ノード) とした知識の構造を模式化している．図 6.4 の A は，

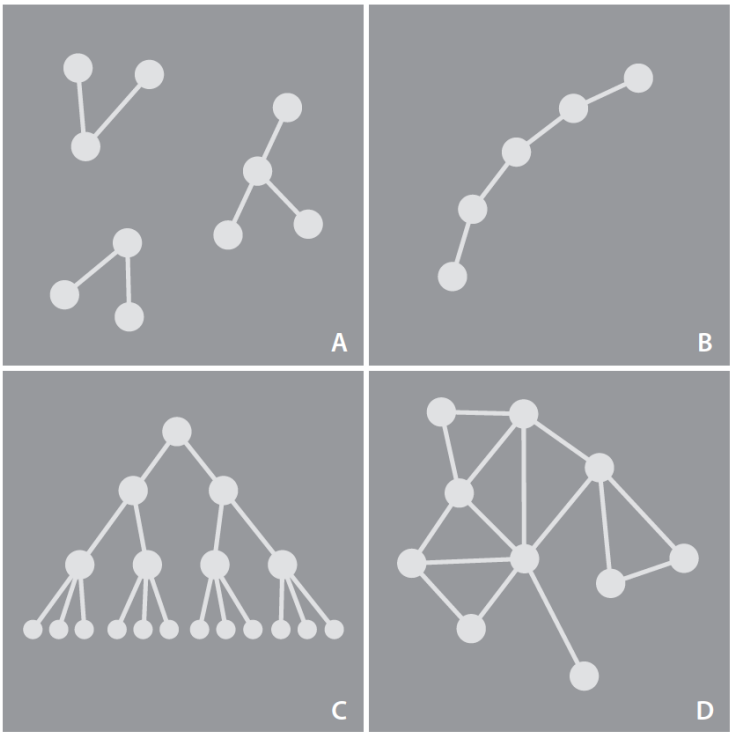


図 6.4: 知識構造の例

知識のつながりが弱く，学習により最終的には図 6.4 の D のような構造を目指したい．3.5.4 で前述した久野の指針にあるよくない学び方は，図 A に属するのではないかと考える．プログラミング入門では，それぞれの機能を個別に学習するケースが見られるが，図 D を目指すのであれば，文脈のあるカリキュラムを構築することが有効であると考える．プログラミングとチームによるソフトウェア開発プロジェクトを合わせて学ぶ本提案は，ここまでに示した学生からの回答からもわ

かるように，プログラミングの基礎知識を身につけ，社会で求められる資質能力を育成する上で有効に機能したと考えられる．

6.8　本章のまとめ

本研究では，大学から社会への接続に必要な知識に焦点を当て，制限された環境下でプログラミング教育を有意義に導入する方法を検討した．

本提案のカリキュラムにおける目的は，学生がプログラミングを通じて自分の考えを容易に確認できるようにすることであり，コンピューターに意図した処理を実行させるために必要な論理的思考スキルを育成することであった．加えて，プロジェクトを通じて，1 人ではできないことを実現することにより，相互学習や，社会で求められるジェネリックスキルの育成を図ることでもあった．

本研究では，リサーチクエスチョンとして，次の 2 点を掲げていた．

RQ1： PBL により，プログラミングの概念の理解と社会で求められる資質・能力の育成をはかることができるか？

RQ2： PBL によるプログラミング教育は，学習の過程を可視化することができるのか?

本章では，第 3 章で提案したプログラミング入門のカリキュラムを実装し，実践授業を行った結果から得られた問題点を修正した．そして，再度，実践授業を行い，RQ1 の前半部分「プログラミングの概念の理解」について，提出された課題のプログラムを検証し，RQ1 の解を補強した．さらに，RQ2 について検証をするために，実践授業での事例と履修者の提出物から検証した．

カリキュラムの改修では，第 5 章で明らかになった問題点として，前半のプログラミングパートを取り上げた．テキストを再考し，描画教材を初回から用いることにより，学習者の苦手な要素を取り除き，また，繰り返し練習できるよう，絵を描くことに特化した．

さらに，中間課題を，与えられた課題を単に解き，レポートにまとめることで終わらせず，ものづくりの要素を加え，よりリアルな実践課題に修正した．出力された絵を単に印刷しただけであったが，効果があったと推察される．

第 5 章で明確にした問題点の制御構造と抽象化について，前半のプログラミングパートのまとめとして課した中間課題では，全員が活用することはできなかったが，後半のプロジェクトパートでは，チームによる活動の中で繰り返し学び，教え合うことにより，理解が深まり活用できている結果が得られた．

これにより，本章では，第 5 章で明らかとなったプログラミングの基礎知識の理解と活用に対する課題については，前半で学んだ知識を活用し，さらに，未習熟であった部分も繰り返し活用することにより，定着させることができていることが示せ，RQ1 のプログラミングの概念の理解についての解を補強できた．加えて，

この結果から，プロジェクトを行うことにより，学んだ知識を活用することができていることを示すことができた．

RQ2 の検証では，6.6 節で描画教材とプロジェクトの効果を評価した．

後半のプロジェクトパートでは，チームによる取り組みだけでなく，中間発表やコードレビューを通じで，クラス内での情報共有や検討を行なったことにより，チーム内での相互学習にとどまらず，チーム間での共感や競争が生まれ，モチベーションを高める効果につながった．これは，描画教材を用い，視覚的に分かりやすい教材を用いたことによる効果であったと考える．

各チームで考えた絵を，チーム内で分担して作成したパーツを組み合わせることにより完成させる工程は，思考と結果をプログラムの実行により可視化でき，プログラムを書き，実行し，考えた通りの結果が得られたかどうを体験することによって効果が高まったと考える．

描画教材を用い，ソフトウェア開発プロジェクトを加えたプログラミング入門は，他にもさまざまな効果があり，履修者のレポートの記述回答からは，プログラミングの基礎知識を身につけただけでなく，チームによるプロジェクトを行うことにより，以下の項目が効果として示された．

- 考える力・問題解決力
- 他者からの学び・他者への配慮
- 役割分担・プロジェクト
- 失敗の共有
- 意思表示することの重要性
- スケジュール管理
- 多角的・多面的な視点や考え方

チームによるプロジェクトに取り組むことにより，描画教材と他者との関係からの効果があった．RQ2 の解としては，描画教材は，思考・表現に対してだけでなく，他者と関係を持ちプロジェクトを遂行する際の情報を共有する部分において大きな効果があった．

半期 90 分 15 回 2 単位の入門科目であるため，情報系の専攻で取り扱う内容・レベルとは異なるが，簡素化した描画教材を用いることで理解しやすくなり，15 回という限られた時間の中で，初学者であっても入門レベルとしての基礎知識を固めることができたと推測できる．

以上より，カリキュラムの妥当性の検証を行い，2 つのリサーチクエスチョンを明らかにしたことから，本研究の目的は達成できたと考える．

「プログラミングを初めて学ぶ」ことと「ソフトウェア開発プロジェクトにおいて重要なことを実践を通じて学ぶ」という，本研究におけるカリキュラムの提

案は，入門科目として大学における標準的な 2 単位の授業として構築し，実装することができ，概ね成功であったと考える．

第7章

追跡調査

7.1　はじめに

本論文では，社会への接続を考えた高等教育における一般情報教育のプログラミング入門を検討してきた．前章までに，カリキュラムの提案をし，設計方針に基づきカリキュラムを構築し，実装し，実践授業を行なった．

実践授業の履修者のその後の活動において，本提案の授業で得た知識がどのように有益に働いているかを調べるために，アンケート調査を行なう．

本章では，調査の目的，方法，その結果について，まとめる．

7.2　調査の目的・方法

調査の目的は，本提案のカリキュラムが履修後の活動において，どのように有益に働いているのかを調べるためである．提案のカリキュラムを実装した受領を履修した学生に対して，その後の活動において，どのような効果が出ているのかについて，記述式のオンラインアンケートで実施する．

質問項目は，次の項目を設定する．既に卒業している者に対しては，2つの項目を追加する．

- 就職を考える際や就職活動をする際に，本授業がどのように影響したか？
- 履修後の活動や仕事でどのように役立っているか？(到達目標を示し，その観点から)
- 授業の際に，もっとやっておけばよかったことについて
- 授業の履修の有無により，他の社員と違いがあるか？(既卒生のみ)
- 現在の仕事で情報システム（レジ，業務システムなど）に関すること (利用者，開発者どちらの立場でも) があればその内容と課題について (既卒生のみ)

対象は，2017 年度～2019 年度に当該の授業を履修した学生である．

7.3 調査結果

2020 年 12 月 17 日～2021 年 1 月 30 日の期間で，オンラインでの記述式アンケートを行った．対象は，これまでに実施した 3 回の授業の履修者 (51 名) が対象である．しかし，すでに卒業している者もあり，連絡先がわからないものもあった．最終的に，アンケートを依頼できたのは，30 名であった．

この 30 名のうち 12 名回からの回答があった．答者の内訳は，在学生 8 名 (就職活動中 4 名，SE 内定 2 名，その他の職に内定 2 名)，既卒生 4 名 (全て IT 関連企業ではない) である．個人情報に触れるため，既卒生の現在の職業は伏せる．

SE に内定している 2 名は，授業を受けたことが就職を決める際に直結したと回答した．現在就職活動中の学生も，大学入学時は視野に入れていなかった IT 企業や SE を視野に入れ，授業での経験が，将来を考えるきっかけになったと回答した．

業種は様々であるが,「グループで協力し，1 つの目的に向かってそれぞれの作業を進めることを意識出来るようなり，選考の際のグループワークの場面で，授業を通して学んだ，一方的な意見を押し付けるのではなく，他人の意見を聞き入れ，互いの意見を尊重して最終の結論に導くことが実践できた」「授業ではグループで 1 つの課題を役割分担して行い，タイムマネジメント，進捗状況なども共有してきた．その経験が就職活動面接時におけるグループディスカッションに役立った」など，全員が就職活動の選考面接において，プロジェクトパートでのグループワークの経験が役に立ったと回答した．

これらの回答は，3. 3 節で示した，到達目標 (2)(3)(4) の達成に該当する．

著者は，対象とした学生のうちの数名から，様々な授業でグループワークを取り入れているが，何週にもわたり同じメンバーで活動することがなかったと聞いている．対象とする学生の環境では，多くのグループワークは，単に数名が集まってのワークであり，分担が決まりゴールに向かって何かに取り組むようなグループワークではない．そのため，多くのグループワークは，仲良しグループでの活動の域を抜けることがなかったと聴者は，履修生から聞いた．

しかし，本授業実践では，プログラミングを伴うことにより，論理的に考えプログラムを書くことと役割を分担することを通しての学びにより大きな効果があったと推察できる回答があった．

> 一番役に立っていることは，課題発見から分析，解決までの課題解決能力です．実際にエラーから，何が違うのか，どこを直したら動くのかの試行錯誤を経験することで，課題解決型のインターンでも焦らず学んだことを活かすことが出来ました．

実践授業では，学年も専攻も関係なく，初めて授業で顔を合わせた仲間と 1 つのものを作り上げていった．チームのメンバーで，何を作るか？対象は誰か？単

に描きたい絵のデザインをだけを考えるのではなく，対象やテーマを決めて取り組んだ．プログラミングでは，適切に設計しなければ期待通りの絵が出力されず，コードを適切に書かなければエラーが生じる．このプロセスを授業の中で繰り返し体験することにより，その後の活動に役立っていると考える．

また，既卒生からは，プログラマーやスペシャリストにならずとも，職種にかかわらずシステムを伴う業務があり，授業での学びが効果となっているとする回答があった．

> 「学習経験からトラブル時の初期行動が取れるようになった」
> 「学生時代に情報技術系のことに慣れ親しんでいる者は業務も効率的に進められている」
> 「情報共有システムを使い，仕事効率を大いに高めていくことの重要さを学び，実践につなげていくことができた」

この回答は，既卒生 4 名全員からの回答であり，どんな職種に就こうとも，職場にはさまざまなシステムがあり，それらに関わって仕事をしていかなければならない．授業を履修したことにより，履修前はコンピュータが苦手であった学生も，自分のできる範囲で対応を試みたり，業務の効率化に努めたりする行動について，授業での学習の効果が出ていると回答されており，授業での学びの効果が出ていることがわかる．

これから SE として就職する学生は，授業後の活動においてどのように役立っているかという問いに対し，「プログラミング自体に関心を持つようになりパソコンを使って何かを作り出すことに興味を持った」，「ニュースなどでも IT に関わる内容に関心を持つようになった」と回答した．

アンケート後に，この 2 名に対して，履修後のプログラミングの学習の継続や入社までの事前研修でプログラミングを学んでいるかについて聞いた．しかし，履修後，継続してプログラミングの学習をすることはなく，就職先の事前研修もまだ始まっていないとのことで，コードを書くことに対してどのように影響しているかについては調査できなかった．

今回の調査では，アンケート期間が短かったこともあり，プログラムを書いたり，ソフトウェア開発に直接関わったりするような内容のコメントは，収集することができなかった．しかし，職種は異なるが要求分析の観点で，仕事をする際に役に立っているという既卒生からの回答もあった．以下にそれを示す．

> 「お客様の要望を聞き出すことに加え，チーム間で共有することで更なるサービスをすることが可能なのではないかと考えコミュニケーションをとることに繋げられている.」

要求分析については，実践授業では大きな効果は見えなかったが，実際の現場に立ち，相手が伴うことで学んだ経験を活かすことにつながったと推察する．ま

た，この回答には，情報の共有についても触れられている．情報の共有については，次のような回答もあった．

> 「自身の進捗状況を他のメンバーと共有することで，何が出来ていて，出来ていないか把握でき次に何をすればいいのか考えることに繋げられている.」

情報を共有し，プロジェクトの進捗管理ができていることが読み取れる．

コードを書くことに対しては，授業後，環境がなくなってしまい覚えていないと回答しているものが多数いたが，考え方や開発工程については，実践に活きている回答があった．

> 「家電からアプリまで，様々な電子機器の制御がどのようにプログラムされているか想像出来るようになり，非常に興味をもつようになった.」
> 「プログラムを完成させるプロセスが，ものづくりや開発のプロセスに繋がっていて社会に出て何かを目標に取り組む際にもとても役に立っています.」
> 「システムのトラブルにみまわれ，必要書類を印刷できなかった際に，すぐに担当者に聞いて問題を対処するのではなく，授業などで学んだ経験から，瞬時に今何をどうすればいいのか，必要最低限の初期対応を自分自身でこなすことができている.」

また，授業を履修したことにより，初動対応を試みれるようになったが，最終的には自己解決できず，情報システムに関する知識をもっと学ぶ必要性を感じているという回答もあった．

以上から，チームで働くこと，IT を活用する場面において，授業で学んだ知識が有効に働いていることがわかり，社会で役に立つような教育カリキュラムが構築できたと裏付けることができる．

7.4　本章のまとめ

プログラミング入門の授業が，履修後の活動にどのように影響し，効果があったかを調べるために，記述式のアンケート調査を行なった．

履修後，プログラムを書く機会は無くなってしまっているが，役割分担，プロジェクト (業務) の進め方，お客様の要求を聞き出して対応することなど，様々な活動に効果があったことがわかった．

プログラムを書く環境は無くなってしまっているものの，興味関心を持ち視野が広げられていることがわかった．卒業した学生の働く環境では，システムを使

わない環境がないことから，この授業での経験が活きているが，不足する知識もあり，カリキュラムの内容や実施時間の検討の必要性もある．

しかし，最小時間の 2 単位での実践であったが，履修後の活動にもプロジェクトの進め方，役割分担，要求分析など，少数の回答からの評価ではあるが，様々な効果が出ていることから，本提案のカリキュラムは履修後の活動においても有効であることが示唆できたと考える．

アンケート項目の回答ではないが，学生時代だけでなく，卒業後も活かせる学びが出来たのはこの授業だと，感想に書いてきた学生が複数いた．チームによるプロジェクトの効果が高く，今後，こういった授業の拡大の必要性を感じた．

アンケート調査については，期間が短く，オンラインによる記述式の調査であったこと，回答数が少ないことから，今後，範囲と内容を広げて調査する必要がある．

第8章

本論文の総括

8.1　本論文のまとめ

本論文では，高等教育において誰もが学ぶべき一般情報教育におけるプログラミング教育について取り上げた．

情報化社会においては，急速に変化する社会に対応する力や，情報通信技術により社会課題を解決する力を養い，専門的な立場になくてもある程度対応でき，専門家とともに問題解決に努められるような基礎知識を養う必要がある．

そのひとつにプログラミングがあり，今では，プログラミング教育は義務教育化されて，誰もが学ぶ環境が整った．しかし，現在，大学に通う学生はプログラミングを学ぶ機会が得られずに社会に出るものも多くいる．情報化社会においては，情報通信技術の基礎知識なくして，よりよい暮らしは成り立たない．そのため，これからの社会での活動をより良くしていくために，社会への接続に対して教育する最後の橋渡しの場所である高等教育においても，初等中等教育から継続してプログラミングを学ぶ環境が必要であり，誰もが等しく学ぶ環境の構築が急務である．

本論文では，高等教育において誰もが学ぶべき一般情報教育におけるプログラミング入門を取り上げ，1 科目でプログラミングの基礎知識を学ぶだけでなく，チームでソフトウェア開発プロジェクトを実践する内容を含む入門レベルのカリキュラムとして，PBL(Project-based Learning) によるカリキュラムを提案した．描画教材を用いることにより，プログラミングの原理の理解とソフトウェア開発プロジェクトの知識の獲得を，90 分 15 回 2 単位で実現できることを示した．

リサーチクエスチョンには，次の 2 点を示した．

RQ1： PBL により，プログラミングの概念の理解と社会で求められる資質・能力の育成をはかることができるか？

RQ2： PBL によるプログラミング教育は，学習の過程を可視化することができるのか?

初等中等教育では，チームによるソフトウェア開発プロジェクトを含む教育項目がないこと，高等教育におけるプログラミング教育の実態は厳しく，時間の制約や教員の確保の問題などもあることから，初等中等教育からの連続性と，誰もが学ぶ環境をつくるための仮説として，次を掲げた．

初等中等教育からの連続性と社会への接続に配慮し，高等教育における一般情報教育では，プログラミング入門とソフトウェア開発プロジェクトを組み合わせたカリキュラムでの実施が有効である．

従来のプログラミング入門では，ソフトウェア開発のことをカバーした例はなく，またソフトウェア開発に焦点をあてている科目は，プログラミング入門を済ませていることが前提で，1科目の中に両方を盛り込むのは無理であるとあきらめられていた．しかし，情報通信技術の発展とともに教室環境だけでなく，授業時間外でのコンピュータ利用も柔軟にできるようになってきていることから，上手く教材を作成することにより，導入レベルのカリキュラムが作成できると考えた．

この仮説に基づきカリキュラムを設計・構築し，これを実装して実践授業を行なった．この結果，RQ1については，提出されたプログラミングの課題を分析し，プログラミングの概念の理解と学んだ知識の活用について明確化した．また，授業内で実施した2回のコンピテンシー評価の分析結果から，社会で求められる資質・能力の育成が図られたことを示した．RQ2については，実践授業を通じて得られた提出物や行動から効果を検証し，描画教材による効果を検証した．

第1章では，本研究の背景について述べ，本研究の目的を示した．情報化社会においては，これまで専門家のものであった情報通信技術が一般市民に拡大し，それに伴い，教育も変化している．小学校でのプログラミング教育が始まり，これが普及した先を見据え，高等教育の一般情報教育におけるプログラミング教育のあり方も変化が必要である．

プログラミング教育を通じて期待されることは，プログラミング言語を学ぶのではなく，これからの社会での活動において効果的に働くよう，学んだ知識を活用したり，応用したりする力をつけることである．また，専門家と連携して相互に補完関係を築けるような力も重要である．そのため，入門レベルであっても，プログラミングの経験や知識がこれからの活動において効果的に働くように学ぶべきであるという考えのもと，誰もが学ぶべき科目として，プログラミング入門にPBLを組み合わせたカリキュラムを提案した．特に，初等中等教育においては，チームでソフトウェア開発を行うようなカリキュラムがないことから，プロジェクトには，ソフトウェア開発を取り上げ，プログラミング入門とソフトウェア開発を合わせて学ぶカリキュラムを考え，研究の目的を示した．

第2章では，高等教育におけるプログラミング教育とアクティブラーニングについての先行研究・関連研究を取り上げた．高等教育においては，プログラミン

グは，一般情報教育として位置づけられており 15 回のうちの 3〜5 回程度で取り上げられている事例が多いことから，カリキュラム設計の時間配分の参考となった．プログラミング言語は，JavaScript を用いた事例もあったが，日本語で学ぶことを推奨している事例も多くあった．教材としては，視覚的にわかりやす描画教材を使った事例を参照した．

近年，アクティブラーニングや PBL が注目されているが，PBL における欠点をまとめ，この欠点を回避したカリキュラムの設計に努めた．プログラミングを伴う PBL では，専門分野におけるソフトウェア開発プロジェクトによるものが多く，専門でない専攻の初学者を対象としたプログラミング入門における PBL の事例を見つけることはできなかった．

第 3 章では，誰もが学ぶべき一般情報教育におけるプログラミング教育の新たなカリキュラムと位置づけ，設計方針について示し，提案するカリキュラムの目標・到達目標を設定し，カリキュラムを構築した．

現在，高等教育で行われている情報教育は各学校の判断に任されており，内容・質ともにバラバラであることや，新たなカリキュラムの設定も，時間や人材の都合からプログラミング教育の実施は難しい．そのため，誰もが限られた時間で容易に取り組めるカリキュラムの検討が望まれる．これまでの多くのプログラミング入門教育では，コンピュータがプログラムによってどのように動作するかを学ぶような技術的なことに重点が置かれ，学んだ知識をどのようにこれからの活動に活かすかということまでを学ぶ機会がなかった．

小学校におけるプログラミング教育が始まり，高等学校までの一連のカリキュラムも整ってきたが，社会に一番近い高等教育においては，情報を専門としない専攻では必須で学ぶ環境がない．特に，現在，在学中の大学生は，義務教育の範囲でもプログラミングを学んでいないものが多く，社会に出るまでに少なくとも 1 度は学習の機会を持たせたいが，全員が学ぶことは難しい．そのため，誰もが取り組みやすく，教えやすい教材として最小限の時間で実施できるカリキュラムを検討した．

プログラミングの学習を通じて期待されることは，就職後の活動を考えると，プログラムを書くことが目的ではなく，プロジェクトへの関わりが大事な要素である．情報技術は専門的に学ぶ人だけの学問分野ではなく，誰もがある程度の基礎知識を身につける必要がある．そして，知識を身につけるだけでなく，それを活用する力や急速な変化に対応する力を養うことができるようなカリキュラムを設計することが，これからの社会に求められていると考えた．

第 4 章では，構築したカリキュラムを実装した実践授業を行い，カリキュラムの妥当性を示した．PBL には，ソフトウェア開発プロジェクトの概念を用いた．初学者が，初めから高度なソフトウェアを作ることはできないことから，初学者でも容易に学べる教材として描画教材を用いた．描画教材は，出力結果が絵として出されることから，視覚的に判断でき，理解されやすい．1 枚の絵をソフトウェアと見立て，絵を構成するパーツをオブジェクトとして捉え，それらを組み合わせ

ることで教材化がはかれた．これにより，プログラミング入門と PBL を組み合わせたカリキュラムは概ね成功した．到達目標の観点から，90 分 15 回 2 単位のカリキュラムの妥当性を示した．そして，RQ1 の社会で求められる資質・能力の育成がはかれるかの観点について，てコンピテンシー評価の分析により検証した．

第 5 章では，第 4 章で行った実践授業より，RQ1 のプログラミングの概念の理解について，プログラミングの課題の理解度から分析した．プログラミングパートでは，一般教養レベルの教材として既に活用されているテキストを用い，本提案のレベルに合わせて修正したものを利用した．プログラミングの実行手順を理解することに時間を要すると考え，描画教材の導入は，ある程度プログラミングに慣れてきたと思われる 5 回目に設定してカリキュラムを設計し，それ以前は，定義が明確な数学による例題・演習問題で構成した．制御構造および抽象化は，前半の課題では概ね理解されていたと思われるが，後半のプロジェクトパートにおいては，活用することできなかった．

この原因は，実践授業を対象とした層では，プログラミングの本質でない部分が足枷となり，理解が進んでいなかったことが明らかになった．入門レベルとして設定した到達目標には到達できていたが，知識の活用という点においては，教材の見直しが必要であることがわかった．

第 6 章では，実践授業で見つかった問題点の修正を行った．そして，修正したカリキュラムを実装し，再度実践授業を行ない，履修者が，実践授業を通じてどのように学んだのかを明らかにした．第 4，5 章で見つかった問題として，プログラミングパートの例題と演習問題の数学的な要素と描画教材を導入するタイミングがあった．これに関しては，プロジェクトに必要な内容に特化した教材とする方針を示し，授業の初回から描画教材を使う方針に変更した．授業の 15 回の進め方は変更せず，1〜4 回目のテキストの例題・演習問題のみの修正を行った．

その結果，課題であった制御構造と抽象化の理解と活用ができた．

PBL を組み合わせた事により，プログラミングの基礎知識の習得に加えて，グループワークにより以下の効果が得られた．

- 考える力・問題解決力
- 他者からの学び・他者への配慮
- 役割分担・プロジェクト
- 失敗の共有
- 意思表示することの重要性
- スケジュール管理
- 多角的・多面的な視点や考え方

カリキュラムの実装で活用した教材は，図形の描画を用いた非常にシンプルな描画教材であった．しかし，構造がシンプルでかつ視覚的にわかりやすいことから，思い描いた絵をプログラミングにより実行する過程で，思考のプロセスが可視化できる効果があった．そして，プログラミングの概念を学ぶだけでなく，学んだ知識を活用したチームによるプロジェクトに取り組むことにより，論理的な議論ができたり，設計・試作・再考・統合の工程から，どこが問題か，エラー処理だけでなく，作品の質も含めてチーム内での議論により，深い学びへとつながったと考える．結果として，プログラミングの知識だけでなく，プロジェクトを通じて学生個々に得られたものは異なるが，ジェネリックスキルの育成に繋がった．

文部科学省は，初等中等教育において「主体的・対話的で深い学び」の視点に立った授業改善を求めているが，高等教育においても繰り返し経験する場はこの視点が必要であり，本提案の方法であれば，視覚的にも理解しやすく，社会が求める能力や資質，学び方など様々なことに深い学びと理解があったと推察する．さらに，失敗からの学びや，他者からの影響から，学び方の変化についても効果が見られた．

第 7 章では，履修後の効果について調べるために，履修者への追跡調査を行なった．履修後，プログラムを書く行為は，なくなってしまっているが，情報通信機器への興味は高まっていることはわかった．また，視野が広がったことにより，情報系の企業への関心が広がり，SE として就職するものもいた．プログラムで絵を出力するというシンプルな教材であったが，チームでの活動は，役割分担や多様性の理解などに繋がっていることがわかった．また，実践授業の際には要求分析については大きな効果が得られていなかったが，卒業した学生からは，実際に現場に立ち，相手が伴うことで学んだ経験が実践につながっていることがわかった．

以上のことをまとめると，90 分 15 回の限られた時間でのカリキュラムであったが，描画教材を用いることで，プログラミング入門とチームによるソフトウェア開発プロジェクトを伴うカリキュラムが構築でき，プログラミングの基礎知識の習得とプロジェクトによりジェネリックスキルの育成ができ，社会での活動に活かされていることから，本提案のカリキュラムは有効であると結論づけることができる，

8.2　研究の限界と今後の課題

プログラミング教育が義務教育となり，初等中等教育からの連続性を社会への接続に配慮した，高等教育におけるプログラミング教育について検討した．高等教育における誰もが学ぶべき科目として，一般情報教育を取り上げた．

高等教育では，必須化されたカリキュラムがなく，教える教員の専門も様々であることから，誰もが教えやすく学びやすいカリキュラムが必要であると考え，高等教育における最小限の時間 90 分 15 回で実施できる方法を提案した．従来のプ

ログラミング入門教育の内容に，初等中等教育で取り上げられていない，チームによるソフトウェア開発プロジェクトの内容を加えた．

このような方法は，これまで初学者を対象として，2 単位のなかで実施するカリキュラムは存在しなかった．本提案が最良の提案であるか，他にも取り組み方はあるかもしれないことを断っておく．

カリキュラムの妥当性を示すために，実践授業を行ったが，対象は，文系女子大であり，履修対象者が女性のみと偏りがあった．加えて，少人数教育を中心とた選択科目であったことから，1 クラスの履修者も 20 名前後と少なく，誰もが等しく学ぶ環境ではなかった．

今後は，男女が混合したクラスにおける実践や必須科目として位置付ける工夫などを行い，対象人数を増やして検証をする必要がある．初回の実践授業では，課題に女性が好むネイルシールのデザインを取り上げたことにより，モチベーションが保てた．その後，課題をネイルシールのデザインからクリアファイルやステッカーのデザインに変更した．この変更は，モチベーションを保つことに対して大きな相違は発生しなかった．しかし，対象とする層により教材とする課題のテーマも検討する必要があると考える．今後，機会があれば，教材や課題のテーマについても対象とする層の相違に関しての効果を含めて検討する必要があると考える．

社会につながるかきりキュラムであったかどうか，履修後の追跡調査を行なったが，短期間の限られた対象の調査に留まった．長期間に渡った追跡調査に関する研究実績が少ないことから，さらに対象の範囲や調査内容を広げて実施すること，および中長期的なスパンでの調査をする意義があると考える．

高等教育における必須化された誰もが学ぶべきプログラミング入門教育の必要性から，2 単位 1 科目で行う提案をした．しかし，この構成が最良であるかどうかは正解はなく，他にもっと良い方法があるのかもしれない．しかし，現状では，専門でない専攻の学生を対象とした 2 単位でプログラミングとソフトウェア開発プロジェクトの両方をこなしている実践はないことから，本研究を出発点として，さらに良い方法を探究していくべきである．今後，小学校から継続してプログラミングを学んだ学生が入学してくることから，対象とする学生のバックグラウンドが変化してくる．そのため，この変化に合わせた対応も検討していく必要がある．

本提案は，出発点であり，変化の激しい情報化社会においては，時代の変化に合わせて変わっていく必要もあることから，最小限の基本的なことが抑えられていれば，後は，必要な時に必要な内容を学ぶことができると考える．入口で躓いてしまい，嫌いを作るのではなく，入口のハードルはなるべく低くし，できたやもっとやってみたいという気持ちを持たせることで，少しずつハードルを上げ，より良い暮らしを築いていくための知識やツールになることを願う．

8.3　今後の展望：高等学校「情報I」の実施を踏まえて

高等学校においては，学習指導要領の改訂により，平成21年度改定の高等学校学習指導要領の「社会と情報」および「情報の科学」の２科目からの選択必修を改め，2022年度からは，共通の「情報I」が必履修となる．

「情報I」の内容は，以下の４項目から構成され，プログラミングが必須となった．

1. 情報社会の問題解決
2. コミュニケーションと情報デザイン
3. コンピュータとプログラミング
4. 情報通信ネットワークとデータの活用

「情報I」では，問題の発見と課題解決に向け，さまざまな事象を情報とその結びつきとして捉え，情報技術を適切かつ効果的に活用する力を育むことが期待されており,「情報デザイン」・「プログラミング」・「データの活用」がポイントとなっている．教育目標は，次のように設定されている．

> 情報に関する科学的な見方・考え方を働かせ，情報技術を活用して問題の発見・解決を行う学習活動を通して，問題の発見・解決に向けて情報と情報技術を適切かつ効果的に活用し，情報社会に主体的に参画するための資質・能力を次のとおり育成することを目指す．
>
> **(1)** 効果的なコミュニケーションの実現，コンピュータやデータの活用について理解を深め技能を習得するとともに，情報社会と人との関わりについて理解を深めるようにする．
>
> **(2)** 様々な事象を情報とその結び付きとして捉え，問題の発見・解決に向けて情報と情報技術を適切かつ効果的に活用する力を養う．
>
> **(3)** 情報と情報技術を適切に活用するとともに，情報社会に主体的に参画する態度を養う．

そして，コンピュータとプログラミングの実践の考え方については，次のように記述されている．

> コンピュータで情報が処理される仕組みに着目し，プログラミングやシミュレーションによって問題を発見・解決する活動を通して，次の事項を身に付けることができるよう指導する．
>
> ア　次のような知識及び技能を身に付けること．
>
> （ア） コンピュータや外部装置の仕組みや特徴，コンピュータでの情報の内部表現と計算に関する限界について理解すること．

（イ） アルゴリズムを表現する手段，プログラミングによってコンピュータや情報通信ネットワークを活用する方法について理解し技能を身に付けること．

（ウ） 社会や自然などにおける事象をモデル化する方法，シミュレーションを通してモデルを評価し改善する方法について理解すること．

イ 次のような思考力，判断力，表現力等を身に付けること．

（ア） コンピュータで扱われる情報の特徴とコンピュータの能力との関係について考察すること．

（イ） 目的に応じたアルゴリズムを考え適切な方法で表現し，プログラミングによりコンピュータや情報通信ネットワークを活用するとともに，その過程を評価し改善すること．

（ウ） 目的に応じたモデル化やシミュレーションを適切に行うとともに，その結果を踏まえて問題の適切な解決方法を考えること．

これらは，初等中等教育からの連続性を踏まえ，さまざまなプログラミング言語を用いた実習が提案されている．2021 年 8 月時点で閲覧できた教科書見本から，各教科書で使われているプログラミング言語をまとめたものを，表 8.1 に示す．

なお，この表に加えて一部，ドリトルや micro:bit が使用されている教科書もある．どの教科書も共有して，プログラミングの基礎知識となる制御構造 (順次・選択・反復) は取り扱っている．しかし，データの入出力や配列や関数を扱っていない教科書もあり，探索や並べ替えのアルゴリズムを扱っているものはさらに少数となる [39].

このように，各出版社で取り上げているプログラミング言語も内容も様々であり，学校がどの教科書を選択するのかによって，学習する内容も様々である．そのため，大学での授業においては，特に，情報を専門としない専攻の学生に対しては，これらのギャップを埋めることから始める必要がある．

高等学校教科「情報」が導入され，2006 年度以降,「情報」を履修した生徒が大学に入学してくるようになった．しかし，高等学校における履修状況は学校によって様々であり，15 年以上が経過した現在も，大学入学時点の学生のスキルはさまざまであり，初年時の教育に苦労することがある．

この状況の要因の 1 つには，高等学校の情報は，大学入学試験の科目として取り上げられなかったことが挙げられる．高等学校の学習指導要領の改訂を受け，2025 年からの大学入学共通テストには教科「情報」が取り入れらることが決まった [78]. これにより，今後は，学習指導要領の教科の目標に掲げられた内容については，基本事項をある程度身につけた学生が大学に入学してくることが期待できる．

本研究で提案したカリキュラムは，プログラミングを初めて学ぶ学生を対象として設定したが，高等学校で使用されるプログラミング言語や学習内容のばらつ

表 8.1: 高等学校「情報 I」の教科書に採用されたプログラミング言語

発行者	記号・番号	書名	Python	Java Script	VBA	Scratch
東京書籍	情 I 701	新編情報 I	○			○
東京書籍	情 I 702	情報 I Step Forward!	○	○		
実教出版	情 I 703	高校情報 I Python	○			
実教出版	情 I 704	高校情報 I JavaScript		○		
実教出版	情 I 705	最新情報 I			○	
実教出版	情 I 706	図説情報 I				○
開隆堂	情 I 707	実践 情報 I			○	
数研出版	情 I 708	高 等 学 校 情報 I	○	○	○	
数研出版	情 I 709	情報 I Next	○	○	○	
日本文教出版	情 I 710	情報 I	○	○		
日本文教出版	情 I 711	情 報 I 図解と実習—図解編				○
日本文教出版	情 I 712	情 報 I 図解と実習—実習編				○
第一出版	情 I 713	高 等 学 校 情報 I			○	

きを考慮すると，しばらくはそのギャップを埋めることが必要となると考える．そのため，本提案で対象とする層については，習熟度の程度も考慮し，あと数年は本提案で進めていくことができるであろう．

しかし，小学校から継続して学ぶ環境で育ってきた層に対しては，プログラミングの基礎知識については，継続的に学習され，ある程度の知識を持ち合わせた上で入学してくることが期待される．特に，著者が提案したカリキュラムの前半部分で取り扱ったプログラミングの基礎知識である，制御構造については，初等中等教育において繰り返し学んでくるであろうことから，高等教育におけるプログラミング入門では，簡略化できると考える．そして，余裕のできた部分に対しては，内容の拡大や課題の高度化など，学生の習熟度を見据えながら時代の流れに沿った改良をする必要があると考えている．

高等学校においては，目標に問題の発見・解決が掲げられ，方法として情報デザイン，プログラミングそしてこれに加えてデータの活用がツールとして機能し

ており，コンピュータに関わる知識に加えて，コミュニケーション能力の育成に関する学びも重視されている．しかし，学習指導要領では，実習時間を設けるよう記載されているが，演習時間については，特に決まった時間数が設定されていない．この点は，学校によってのばらつきが学生の理解度や知識の定着具合に差が生じると予測される．著者は，十分な演習時間をとり，体験を繰り返しながら知識の定着を図ることが重要であると考え，プロジェクトによる取り組みを重視している．

本研究では，社会への接続を考えたプログラミングの活用を重視し，チームによるソフトウェア開発プロジェクトを取り入れることにより，問題解決をはかり，チーム力，コミュニケーション能力の育成をはかることを考えた．プログラミングの基礎知識に関して，大学入学までに学ぶ機会が得られ，ある程度の知識が担保されれば，チームによるプロジェクトの部分の内容を厚くし，様々な形態のプロジェクトを取り入れることも可能になるかもしれない．

さらに，逆の提案として，高等学校「情報I」で扱われることになっている「情報デザイン」へ，本提案のカリキュラムを用いるような考えも期待できるであろう．

「情報デザイン」は，平成30年高等学校学習指導要領解説（情報編）第1部第2章第1節2(2)[114]で，次のように示めされている．

> 「情報I」で扱う情報デザインは，効果的なコミュニケーションや問題解決のために，情報を整理したり，目的や意図を持った情報を受け手に対して分かりやすく伝達したり，操作性を高めたりするためのデザインの基礎知識や表現方法及びその技術のこと

プロジェクトパートのチームで絵をデザインして作品をつくる工程において，意図した絵をコードを書き実現する中でコミュニケーション，情報の明確な伝達．情報の整理などを含む活動ができる．

来年度以降，各学校の教科書の採択状況や大学に入学してくる生徒の状況を調査しながら，カリキュラムの冗長な部分の軽減と不足部分の補充の検討を行い，より実践的で社会への接続に配慮したカリキュラムの改良を考えたい．

8.4 結言

急速な情報通信技術の進展により，知識や情報に価値が置かれるような社会へと変革してきた．これにより，情報の収集・伝達・処理を中心とした情報革命と情報技術による情報化社会において，これまで求められていたIT人材の枠を超え，誰もがある程度の情報リテラシや情報技術の基礎知識を持つことが求められる社会になってきた．

わが国では，2020年4月より，初等教育におけるプログラミング教育が始まり，初等中等教育でのプログラミング教育を含む情報教育の体系が出来上がった．そして，プログラミング教育は，義務教育として位置付けられるまでになった．し

かし，高等教育におけるプログラミングを含む教育は，現状では，必須化されたものはなく，各大学の裁量に任されている状況である．社会への接続を考えると，初等中等教育での環境は整ったものの，大学での学習の機会がなく，初等中等教育で学んだ知識をさらに継続して学ぶ機会がないままに社会へ出ることになる．

我が国における大学は，文部科学省『学校基本調査』(令和元年)[106] によると，学部学生数の 61.4%が文系を占めており，義務教育におけるプログラミング教育が浸透するまでのしばらくの期間，これらの学生の多くはプログラミングを学ばないままに社会に出て行くこととなる．これからの情報化社会においては，誰もが基本的なプログラミングの素養を求められる社会 [71] となっており，より良い社会を築いていく上でも，プログラミングを学ぶ適正な環境の整備が望まれる．そして，この整備は，初等中等教育においてプログラミング教育を受けた生徒が，大学に入学してからも継続して進められるべきである．

本論文で提案したプログラミング入門と PBL を組み合わせた方式では，初等中等教育の内容に含まれていない，チームによるソフトウェア開発プロジェクトの内容を含んでいる．そのため，プログラミングの学習の経験の有無に関係なく，有効に働くはずである．そして，チームによるプロジェクトは，多様な学生が持ち合わせた個々の知識を組み合わせて取り組むことにより，有意義な体験型の学びとなるであろう．

本提案の方法は，非常にシンプルな描画教材で構成されており，教員・学生共に負担が少なく，高等教育における標準的な 90 分 15 回の枠に収まるカリキュラムとして実施可能である．

プログラミング教育に関する研究は，これまでにも多くの研究が存在するが，社会との接続を考え，人材の裾野を広げることも目的とした，本提案のような取り組みはこれまでになく，本研究が起点になると考える．

利用者の視点に立つと，私たちの暮らしの中には，既に情報通信技術は欠かすことができなくなっており，よりよい暮らしをしていく上では，それらをうまく使いこなすことが求められる．そのため，情報通信技術の基礎知識はだれもが必要である．また，現在の社会においては，専門家のみで行えるソフトウェア開発は少なく，様々な知識をもった各分野の人材が集まってコミュニケーションを図りながら進めていくことが重要であり，情報通信技術に関する基礎知識は，専門によらず誰もが持ち合わせることが求められている．このような集団での取り組みでは，情報通信技術の他に，論理的な会話によるコミュニケーションや情報を明確に伝えたり記述したりする力などのジェネリックスキルも重要である．

テストのための学問ではなく，誰もが楽しく取り組めて，よりよく生きていくための知恵となるような資質・能力を育む教育カリキュラムを今後も検討していきたい．

謝辞

本研究を進めるにあたり，多くの方々にご指導とご協力を賜りました．みなさんのお力添えがなければ，この論文は完成できませんでした．お世話になった全ての方々に心より感謝申し上げます．

初めに，本論文を審査していただきました論文審査委員の本学大学院情報理工学研究科情報・ネットワーク工学専攻の中山泰一教授，岩崎英哉教授，小林聡教授，寺田実准教授，本学理事の小花貞夫名誉教授に深く感謝いたします．

中山泰一教授には，私の博士後期課程在学中の最後の 1 年でしたが，主任指導教員として，たくさんの助言をいただきました．おかげで，本論文をまとめることができました．深く感謝いたします．久野靖教授には，私の博士後期課程進学への扉を開いていただきました．そして，指導教授として，研究への助言だけでなく教材の提供および論文指導，そして，ドイツでの国際会議への参加など，公私を含め，たくさんのご指導をいただきました．深く感謝いたします．

津田塾大学の小川貴英名誉教授には，論文指導だけでなく，博士課程在籍期間の全ての期に渡り，たくさんの助言をいただきました．ここに深く感謝いたします．

本学大学院情報理工学研究科情報・通信工学専攻の角田博保元准教授・客員研究員，大学院情報理工学研究科情報・ネットワーク工学専攻の赤池英夫助教，本学共通教育部赤澤紀子特任准教授にも，本研究に対して様々なアドバイスをいただきました．深く感謝いたします．

Silliman University の Dr. Chuchi Montenegro には，本研究の導入部分での実験に協力いただきました．また，国際会議での発表や研究への助言をいただきました．公私ともに相談に乗っていただき，応援のメッセージはいつも励みとなりました．深く感謝いたします．

Silliman University の Dr. Dave Marchial，Ostfalia University の Dr. Markus Launer には，本研究に対してご理解いただき，様々な国際会議での発表に招待いただきました．そして，常に応援のメッセージを送ってくださいました．深く感謝いたします．

フェリス女学院大学の Benjamin Middleton 教授には，論文執筆の際にたくさんの助言をいただき，公私ともに支えていただきました．ここに感謝いたします．

授業実践の場を提供いただいた，協力企業の金子哲也氏，木村英貴氏，河野厚志氏にも，深く感謝します．

フェリス女学院大学の「情報発信と世界」の履修者の皆さんに，感謝いたしま

す．みなさんの真摯な学びは，私の研究の励みでもありました．また，論文執筆や国際会議での発表を一緒にしてくれたみなさんにも，感謝いたします．

久野研究室，中山研究室のみなさん，特に，渡辺勇士博士、渡邉景子准教授、中鉢直宏講師には，研究だけでなく，さまざまなことにアドバイスをいただき，励まされ，支えられました．深く感謝します．

フェリス女学院大学の福永保代名誉教授，東京通信大学の鈴木範子講師には，日々，励まされ，支えられました．深く感謝します．

大学院への進学を応援し，常に励ましてくれた両親に，深く感謝します．

そして，どんなに忙しい時にも平常時と同じく接してくれた家族に，感謝します．

参考文献

[1] amazon Honeycode, https://www.honeycode.aws/, 2021 年 5 月 20 日閲覧.

[2] Susan A. Ambrose, Michael W. Bridges, Michele DiPietro, Marsha C. Lovett, Marie K. Norman : How Learning Works: Seven Research-Based Principles for Smart Teaching, Jossey-Bass(2010).

[3] AppSheet, https://www.appsheet.com/, 2021 年 5 月 20 日閲覧.

[4] Assessment & Teaching of 21st-century Skills(ACT21S), http://www.atc21s.org/, 2019 年 2 月 20 日閲覧.

[5] Joseph E. Aoun, 『ROBOT-PROOF: Higher Education in the Age of Aryificial Intelligence』, The MIT Press,(2017). (邦訳) 杉森公一, 西山 宣昭, 中野 正俊, 河内 真美, 井上 咲希, 『ROBOT-PROOF:AI 時代の大学教育』, 森北出版 (2020)

[6] Assessment & Teaching of 21th Century Skills, http://www.atc21s.org/, 2019 年 2 月 20 日閲覧.

[7] Association of American Colleges & Universities, VALUE Rubrics, https://www.aacu.org/value-rubrics, 2019 年 2 月 20 日閲覧.

[8] BREHMER, Berndt: The dynamic OODA loop: Amalgamating Boyd ' s OODA loop and the cybernetic approach to command and control, Proceedings of the 10th international command and control research technology symposium, pp. 365-368.(2005)

[9] Center for Curriculum Redesign : What should students learn for the 21st century?, https://curriculumredesign.org/,2019 年 2 月 20 日閲覧.

[10] Claris, https://www.claris.com/ja/filemaker/, 2021 年 5 月 20 日閲覧.

[11] Code for All Members, https://codeforall.org/members, 2021 年 5 月 20 日閲覧.

[12] Code for America, https://www.codeforamerica.org/,2019 年 2 月 20 日閲覧.

[13] Code.org, https://code.org/,2019 年 2 月 20 日閲覧.

[14] Chad Fowler: The Passionate Programmer: Creating a Remarkable Career in Software Development, Pragmatic Bookshelf(2009). (邦訳) でびあんぐる：情熱プログラマー ソフトウェア開発者の幸せな生き方, オーム社 (2010).

[15] Carl Benedikt Frey, Michael Osborne : The Future of Employment, Oxford Martin Programme on Technology and Employment(2013).

[16] Hagiya, M.: Defining Informatics across Bun-kei and Ri-kei，Journal of Information Processing，Vol. 23，No. 4 pp. 525-530(2015).

[17] Hour of Code, https://hourofcode.com/jp,2019 年 2 月 20 日閲覧.

[18] Boswell, D.,Foucher, T.: The Art of Readable Code, O'Reilly Media, Inc.(2012). (邦訳) 角征典：リーダブルコード，オライリー・ジャパン (2012).

[19] Bruna W. Kernighan, P. J. Plauger : The Elements of Programming Style, Bell Telephone Laboratories, Incorporated(1978). (邦訳) 木村 泉：プログラミング書法, 共立出版株式会社 (1982).

[20] McDowell, C., Werner, L., Bullock, H. and Fernald J.: The effects of pair-programming on performance in an introductory programming course, Proc. The 33rd SIGCSE technical symposium on Computer science education(SIGCSE '02), Association for Computing Machinery, pp. 38-42(2002).

[21] McDowell, C., Hanks, B. and Werner, L.: Experimenting with Pair Programming in the Classroom, Proc. The 8th annual conference on Innovation and technology in computer science education, Association for Computing Machinery, pp. 60-64(2003).

[22] Mironova Olga, Amitan Irina, Vilipõld Jüri, Saar Merike : Active Learning Methods in Programming for Non-IT Students, International Association for Development of the Information Society, International Conference on e-Learning, Madeira, Portugal(2016).

[23] National Research Council and Division of Behavioral and Social Sciences and Education：How People Learn, National Academy Press, Wasington D.C.(1999). 森, 秋田監訳，授業を変える, 北大路書房 (2002).

[24] Nuutila, E., Törmä, S. and Malmi, L.: PBL and Computer Programming — The Seven Steps Method with Adaptations, Computer Science Education, vol. 15, no. 2, pp. 123-142(2005).

[25] Open government hackathons matter, Mark Headd, govfresh, August 24, 2011

[26] Papert, S.: Mindstorms: Children，Computers，and Powerful Ideas，Basic Books, Inc., NewYork(1980). 奥村貴世子 (訳)：マインドストームー子供，コンピューター，そして強力なアイデア (新装版)，未来社 (1982)．

[27] Ritchhart,R., Church, M. and Morrison, K.: Making Thinking Visible: How to Promote Engagement, Understanding, and Independence for All Learners, Jossey-Bass(2011). 黒上晴夫，小島亜華里 (訳)：子どもの思考が見える 21 のルーチン：アクティブな学びを作る，北大路書房 (2015)．

[28] Carl R. Rogers, H. Jerome Freiberg : 『Freedom to Learn』, Prentice Hall(1994).

[29] Elliot Soloway, Amanda Pryor : The Next Generation in Human-Computer Interaction, Vol. 39, No. 4 COMMUNICATIONS OF THE ACM(1996).

[30] Natsuko Uchida, Chuchi Montenegro: Robotics Learning in Japan: Experiences and Lessons for Establishing Outcomes-based STEM Learning, Philippine eLearning Society Online Journal, Volume 1, Issue 1, (ISSN 2094-781x)(2019).

[31] Uchida, N. and Kuno, Y.: Programming Education in a Women University in Japan: A Case Study of Performance-Based Learning in Liberal Arts, Proc. International Congress on eLearning (ICE) 2019, Philippine eLearning Society (PeLS), pp. 175-189 (2019).

[32] Watanabe, T., Nakayama, Y., Harada, Y., & Kuno, Y.: Analyzing Viscuit Programs Crafted by Kindergarten Children. In Proceedings of the 2020 ACM Conference on International Computing Education Research, pp.238-247(2020).

[33] IT メディア：私が決めた要件通りにシステムを作ってもらいましたが、使えないので訴えます，https://www.atmarkit.co.jp/ait/articles/2105/31/news009.html，2021 年 6 月 2 日．

[34] Gerald M. Weinberg: The Psychology of Computer Programming: Silver Anniversary Edition Annual, Subsequent Edition, Dorset House(1998). 木村泉，久野靖，角田博保，白浜律雄 (訳)：プログラミングの心理学—または，ハイテクノロジーの人間学 25 周年記念版, 毎日コミュニケーションズ (2005)．

[35] Potter, J. : 英国政府内閣府業務自動化への取り組み—RPA 導入を政府横断で推進する Centre of Excellence—，行政&情報システム，2019 年 8 月号，一般社団法人 行政情報システム研究所 (2019).

[36] PROG 白書プロジェクト (監修　河合塾, リアセック) : PROG 白書 2015, 学事出版 (2015).

[37] 荒木恵, 松澤芳昭, 杉浦学, 大岩元 : プログラミング授業の導入としての「お絵かきプログラム開発演習」， 日本教育工学会研究報告集 言語力を育む授業づくり，pp.111-117(2008).

[38] 池本有里，鈴木直美，近藤明子，山本耕司:学生のコンピテンシー育成を目指す PBL 型教育プログラムの実施と考察，四国大学紀要，no. 42，pp. 1-11(2014).

[39] 井出広康：情報 I の教科書におけるプログラミング分野の比較と分析，第 14 回全国大会講演資料集，日本情報科教育学会 (2021).

[40] 一般社団法人日本経済団体連合会　:　産学官連携による高度な情報通信人材の育成強化に向けて，　2005 年 6 月 21 日, http://www.keidanren.or.jp/japanese/policy/2005/039/，　2018 年 5 月 10 日閲覧.

[41] 井上明，金田重郎：実システム開発を通じた社会連携型 PBL の提案と評価，情報処理学会論文誌，vol. 49，no. 2，pp. 930–943(2008).

[42] 井上克郎, ぺた語義:enPiTってなんですか?, 情報処理, vol. 58, no. 12,pp. 1123–1123(2017).

[43] 内田奈津子 : ぺた語義 : プログラミング入門をプロジェクトでやってみた，情報処理学会学会誌「情報処理」，Vol. 59，No. 3，pp. 268-271(2018).

[44] 内田奈津子，伊地知咲希，永井絵梨奈，堀池悠那：Society　5.0　with　Ruby　—社会で役立つ人材育成の鍵—，Ruby　World　Confernce　2020，https://2020.rubyworld-conf.org/ja/program/session-4/(参照 2020-12-18).

[45] 内田 奈津子，柏 舜: 初等教育・特別活動 (クラブ活動) における実践事例 : 2011 年度からの実践を振り返り今後の活動を考える, 情報処理学会 第 62 回プログラミング・シンポジウム, 2021.

[46] 内田康之 : 問題解決力を身につける創造教育の実践，工学教育，vol. 63，no. 2，pp. 2_40–2_46(2015).

[47] 遠藤慶一, 黒田久泰, 小林真也 : アプリケーション開発を題材とした PBL 型教育によるコンピテンシー育成の効果, 情報処理学会第 77 回全国大会講演論文集, vol. 2015, no. 1, pp. 589–590(2015).

[48] 岡本雅子, 村上正行, 吉川直人, 喜多一 :「視覚的顕在化」に着目したプログラミング学習教材の開発と評価, 日本教育工学会論文誌, 37(1), pp.35-45, 2013.

[49] 大岩元：一般情報教育, 情報処理, Vol. 32, No. 11, PP. 1184-1188(1991).

[50] 大岩元：プログラミング教育の社会的意義—設計から始めるプログラミング教育—，情報教育シンポジウム 2016 論文集，pp.122-128(2016)．

[51] 大阪市：ICT を活用した新たな市民活動「CivicTech（シビックテック）」の取組, https://www.city.osaka.lg.jp/shimin/page/0000329043.html，(参照 2020-09-28).

[52] 太田 剛, 森本 容介, 加藤 浩 : 諸外国のプログラミング教育を含む情報教育カリキュラムに関する調査, 日本教育工学会論文誌, 40 巻, 3 号, p. 197-208(2016).

[53] 小川南，平ひかり，内田奈津子：プログラミング入門をプロジェクトでやってみた ー Ruby で取り組むプログラミング実践ー，Ruby World Conference 2018，https://2018.rubyworld-conf.org/, (参照 2020-09-28).

[54] 掛下 哲郎, 高橋 尚子 : 国内 750 大学の調査から見えてきた情報学教育の現状 : 国内 750 大学の調査から見えてきた情報学教育の現状 —（1）調査の全貌編—, 情報処理,vol. 58,no. 5pp. 420-425，情報処理学会 (2017).

[55] 掛下 哲郎 : 国内 750 大学の調査から見えてきた情報学教育の現状—(2) 情報専門教育編—, 情報処理，vol. 58, no. 6, pp. 520-525, 情報処理学会 (2017).

[56] 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，情報処理学会研究報告 コンピュータと教育 (CE)，16(2011-CE-108)，pp. 1-8(2011)．

[57] 河村一樹ほか：これからの大学の情報教育, 日経 BP マーケティング (2016).

[58] 河合塾グループ 教育研究開発活動 PROG サイト, https://www.kawaijuku.jp/jp/research/prog/, 2018 年 9 月 1 日閲覧.

[59] 葛原志保，米屋美雪，伊地知 咲希，大坂 祥子，内田 奈津子：プログラミング初学者にとって抽象化はなぜ難しいのか—お絵かきプログラミングによる学習を分析する—，研究報告コンピュータと教育（CE）2021-CE-159(2021).

[60] 久野靖 : プログラミング入門をどうするか : 1．プログラミング教育／学習の理念・特質・目標，情報処理学会学会誌「情報処理」，Vol, 57，No. 4，pp. 340 - 343(2016)．

[61] 経済産業省, 平成 30 年度我が国におけるデータ駆動型社会に係る基盤整備 (IT 人材等育成支援のための調査分析事業)—IT 人材需給に関する調査—調査

報告書, https://www.meti.go.jp/policy/it_policy/jinzai/houkokusyo.pdf，2017 年 9 月 1 日閲覧.

[62] 経済産業省:社会人基礎力 (オンライン)，https://www.meti.go.jp/policy/kisoryoku/index.html(参照 2021-01-19).

[63] 経済産業省，社会人基礎力, https://www.meti.go.jp/policy/kisoryoku/index.html(参照 2021-01-19).

[64] 経済産業省：社会人基礎力に関する研究会・中間とりまとめ (2006).

[65] 高度 IT 人材を育成する産学協働の実践教育ネットワーク,http://www.enpit.jp/, 2018 年 4 月 1 日閲覧.

[66] コード・フォー・ジャパン, https://www.code4japan.org/(参照 2021-01-19).

[67] 小林康夫：岩波講座現代の教育第 10 巻 変貌する大学教育 大学教育での意味を再考する—大学で何を学ぶか—，pp. 315-330，岩波書店，東京 (1998).

[68] 駒谷昇一：PBL は教育にどのようなインパクトがあるのか，情報教育シンポジウム 2009 論文集，2009，no. 6，pp. 131–138(2009).

[69] 佐伯胖,「高度情報化と教育の課題」『岩波講座現代の教育第 8 巻情報とメディア』, 岩波書店 (1998).

[70] 首相官邸：世界最先端 IT 国家創造宣言 (2013 年 6 月 14 日)(オンライン), http://www.kantei.go.jp/jp/singi/it2/kettei/pdf/20130614/siryou5.pdf(参照 2020-08-23).

[71] 首相官邸:未来投資会議　林文部科学大臣提出資料 (2018 年 5 月 17 日) (オンライン), http://www.kantei.go.jp/jp/singi/keizaisaisei/miraitoshikaigi/dai16/siryou6.pdf(参照 2020-08-23).

[72] 首相官邸ウェブサイト,「日本再興戦略 2016— 第 4 次産業革命に向けて —」(平成 28 年 6 月 2 日閣議決定)，https://www.kantei.go.jp/jp/singi/keizaisaisei/pdf/zentaihombun_160602.pdf(参照 2020-08-23).

[73] 首相官邸ウェブサイト,「第 4 次産業革命に向けた人材育成総合イニシアチブ」(産業競争力会議 第 2 回 資料 2 文部科学大臣提出資料) 2016.4.19. https://www.kantei.go.jp/jp/singi/keizaisaisei/skkkaigi/dai26/siryou2.pdf(参照 2020-08-23).

[74] 情報処理学会：カリキュラム標準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(参　照 2020-08-23)．

[75] 総務省，平成 30 年版　情報通信白書 | 人口減少によって生じる課題，https://www.soumu.go.jp/johotsusintokei/whitepaper/h30.html(参照　2020-08-23)．

[76] 高橋尚子：国内 750 大学の調査から見えてきた情報学教育の現状: 国内 750 大学の調査から見えてきた情報学教育の現状 (3) 一般情報教育編，情報処理学会学会誌「情報処理」，vol. 58，no. 6，pp. 526-530(2017)．

[77] 大学等における 一般情報処理教育の在り方に関する調査研究委員会 ：大学等における 一般情報処理教育の在り方に関する調査研究平成 13 年度報告書（文部科学省委嘱調査研究），情報処理学会 (2002).

[78] 大学入試センター：平成 30　年告示高等学校学習指導要領に対応した令和 7 年度大学入学共通テストからの出題教科・科目について，https://www.dnc.ac.jp/kyotsu/shiken_jouhou/r7ikou.html（ 参 照　2021-08-25）．

[79] 電気通信大学共通教育部情報部会：基礎プログラミングおよび演習 2017(オンライン)，https://joho.g-edu.uec.ac.jp/joho/fp2017/fp2017-180203.pdf(参照 2021-01-19).

[80] 独立行政法人情報処理推進機構 IT 人材育成本部イノベーション人材センター：実践的講座構築ガイド 産学連携教育の自律的展開を進めるために 第 3 部評価基準編，pp. 3-14，独立行政法人情報処理推進機構 (2013)．

[81] 独立行政法人情報処理推進機構 IT 人材育成本部 編：IT 人材白書 2013，p. 150，独立行政法人情報処理推進機構 (2013)．

[82] 内閣府，Society 5.0，https://www8.cao.go.jp/cstp/society5_0/(参照 2020-08-23)．

[83] 永井隆雄，コンピテンシーの正しい理解と使い方，http://www.itmedia.co.jp/im/articles/0312/06/news001.html(参照 2020-08-23)．

[84] 中山泰一：大学入学共通テストへの情報の出題について，ニューサポート高校「情報」，No. 18 (2021 予定)．

[85] 中山留美子：アクティブ・ラーナーを育てる能動的学修の推進における PBL 教育の意義と導入の工夫，21 世紀教育フォーラム，vol. 8，pp. 13–21，弘前大学 21 世紀教育センター (2013)．

[86] 西田知博，原田章，中西通雄，松浦敏雄：プログラミング入門教育における図形描画先行型のコースウェアが学習に与える影響，情報処理学会論文誌教育とコンピュータ (TCE)，vol. 3，no. 1，pp. 26-35(2017)．

[87] 日経 NETWORK ： 80 歳からのプログラミング『 「若者に勝てるゲームが欲しい」 ，82 歳アプリ開発者が世界に注目される理由』, https://xtech.nikkei.com/it/atcl/column/17/110800498/110900001/(参照 2020-08-23)．

[88] 日本学術会議 : 情報教育課程の設計指針—初等教育から高等教育まで (オンライン)，http://www.scj.go.jp/ja/info/kohyo/kohyo-24-h200925-abstract.html(参照 2021-01-23).

[89] 日本学術会議 : 大学教育の分野別質保証のための教育課程編成上の参照基準情報学分野 (オンライン)，http://www.scj.go.jp/ja/info/kohyo/pdf/kohyo-23-h160323-2.pdf(参照 2020-08-23).

[90] 萩谷昌己 : 未来投資会議における大学入学共通テストに情報の試験を入れる方針に賛同する提言について—大学情報教育体系化の必要性—，情報処理，Vol. 59，No. 9，pp.778-781 (2018)．

[91] 萩谷昌己 :「情報教育課程の設計指針」解説, 情報処理, Vol.62, No.4 (2021).

[92] 原田悦子 : 文科系大学・学部における情報教育〜その目的と問題〜, 情報処理, vol. 41. no. 3, 情報処理学会 (2000).

[93] 樋口耕一 ：社会調査のための計量テキスト分析 —内容分析の継承と発展を目指して— 第 2 版，ナカニシヤ出版 (2020).

[94] 平井佑樹, 井上智雄：ペアプログラミング学習における状態の推定—つまずきの解決の成功と失敗に見られる会話の違い, 情報処理学会論文誌, Vol. 53, No.1, pp. 72-80(2012).

[95] 松浦佐江子, 相場亮 : グループワークによるソフトウェア工学教育の試み, 情報処理学会研究報告コンピュータと教育 (CE), 13(2002-CE-068), pp. 1-8(2003).

[96] 松尾尚：事例研究：石垣市と産業能率大学による地域振興を目的とする PBL の運営, 経営情報学会　全国研究発表大会要旨集, pp. 127–130(2017).

[97] 松澤芳昭，大岩元：産学協同の Project-based Learning によるソフトウェア技術者教育の試みと成果，情報処理学会論文誌, vol. 48，no. 8，pp. 2767–2780(2007)．

[98] 松澤芳昭, 杉浦学, 大岩元 : 産学協同の PBL における顧客と開発者の協創環境の構築と人材育成効果, 情報処理学会論文誌, vol. 49, no. 2, pp. 944–957(2008).

[99] 松澤芳昭，酒井三四郎：ビジュアル型言語とテキスト記述型言語の併用によるプログラミング入門教育の試みと成果，情報処理学会研究報告 コンピュータと教育 (CE)，2013-CE-119，pp. 1-11(2013)．

[100] 松本このみ，吉田智子：文系学部における PEN を用いたプログラミング授業の実践例—繰り返し処理の理解を助ける教材の提案—，情報処理学会研究報告 コンピュータと教育 (CE)，2011-CE-109，pp. 1-7(2011)．

[101] 三重大学高等教育創造開発センター，三重大学版 PBL 実践マニュアルー事例シナリオを用いた PBL の実践ー, (2007).

[102] 三重大学高等教育創造開発センター : Problem-based Learning 実践の方法論報告書 (国際シンポジウム・ワークショップ)，(2006)．

[103] 溝上慎一，成田秀夫：アクティブラーニングとしての PBL と探究的な学習, 東信堂 (2016).

[104] 美馬のゆり (著, 編集), 冨永敦子 (著), 田柳恵美子 (著) : 未来を創る「プロジェクト学習」のデザイン，近代科学社 (2018).

[105] 文部科学省, 新たな未来を築くための大学教育の質的転換に向けて～生涯学び続け，主体的に考える力を育成する大学へ～（答申）, https://www.mext.go.jp/b_menu/shingi/chukyo/chukyo0/toushin/1325047.htm

[106] 文部科学省 : 学校基本調査 (オンライン), https://www.mext.go.jp/b_menu/toukei/chousa01/kihon/1267995.htm(参照 2020-08-28).

[107] 文部科学省, 基本計画特別委員会（第 4 期科学技術基本計画）（第 4 回）配付資料 資料 3- 2 知識基盤社会を牽引する人材の育成と活躍の促進に向けて（案） 第 1 章 知識基盤社会が求める人材像, https://www.mext.go.jp/b_menu/shingi/gijyutu/gijyutu13/siryo/attach/1285416.htm

[108] 文部科学省：教育課程企画特別部会 参考資料 2 小学校段階におけるプログラミング教育の在り方について（議論の取りまとめ）

[109] 文部科学省 : 教育課程企画特別部会論点整理 補足資料 (5)(オンライン), http://www.mext.go.jp/component/b_menu/shingi/toushin/__icsFiles/afieldfile/2015/09/24/1361110_2_5.pdf(参照 2018-03-05).

[110] 文部科学省，教育振興基本計画：第1期計画について（対象期間：平成20年度～平成24年度），https://www.mext.go.jp/a_menu/keikaku/detail/1335036.htm

[111] 文部科学省：高大接続システム改革会議:高大接続システム改革会議「最終報告」(オンライン)，http://www.mext.go.jp/component/b_menu/shingi/toushin/__icsFiles/afieldfile/2016/06/02/1369232_01_2.pdf(参照 2018-03-08).

[112] 文部科学省，高等学校学習指導要領（平成11年3月告示，14年5月，15年4月，15年12月一部改正），https://www.mext.go.jp/a_menu/shotou/cs/1320221.htm

[113] 文部科学省，高等学校学習指導要領　第2章　普通教育に関する各教科　第10節　情報, https://www.mext.go.jp/a_menu/shotou/cs/1320338.htm

[114] 文部科学省：高等学校学習指導要領（平成 30 年告示）解説 (情報編), https://www.mext.go.jp/content/1407073_11_1_2.pdf(参照 2020-12-10).

[115] 文部科学省, 「産業界ニーズに対応した教育改善・充実体制整備事業」中部圏の地域・産業界との連携を通した教育改革力の強化平成 26 年度 東海A(教育力) チーム成果物『アクティブラーニング失敗事例ハンドブック』, https://www.nucba.ac.jp/archives/151/201507/ALshippaiJireiHandBook.pdf(参照 2017-07-05).

[116] 文部科学省，情報教育指導力向上支援事業（諸外国におけるプログラミング教育に関する調査研究），https://www.mext.go.jp/a_menu/shotou/zyouhou/detail/1408119.htm(参照 2017-07-05).

[117] 文部科学省, 「大学における工学系教育の在り方について（中間まとめ）」について, https://www.mext.go.jp/b_menu/shingi/chousa/koutou/081/gaiyou/1387267.htm(参照 2017-07-05).

[118] 文部科学省, 中央教育審議会「新たな未来を築くための大学教育の質的転換に向けて— 生涯学び続け, 主体的に考える力を育成する大学へ —（答申）」, http://www.mext.go.jp/b_menu/shingi/chukyo/chukyo0/toushin/1325047.htm(参照 2017-07-05).

[119] 文部科学省：中央教育審議会答申「学士課程教育の構築に向けて」(オンライン)，https://www.mext.go.jp/b_menu/shingi/chukyo/chukyo0/toushin/1217067.htm(参照 2020-08-23).

[120] 文部科学省，中学校学習指導要領（平成10年12月告示，15年12月一部改正），https://www.mext.go.jp/a_menu/shotou/cs/1320101.htm(参照 2017-07-05).

[121] 文部科学省，【総則編】中学校学習指導要領（平成 29 年告示) 解説，https://www.mext.go.jp/component/a_menu/education/micro_detail/__icsFiles/afieldfile/2019/03/1
照 2021-09-10).

[122] 文部科学省：「超スマート社会における情報教育の在り方に関する調査研究」報告書 (オンライン)，http://www.mext.go.jp/a_menu/koutou/itaku/1386892.htm(参照 2017-07-05).

[123] 文部科学省，プログラミング教育, https://www.mext.go.jp/a_menu/shotou/zyouhou/detail/1375607.htm(参照 2017-07-05).

[124] 文部科学省，我が国の高等教育の将来像（答申）,https://www.mext.go.jp/b_menu/shingi/chukyo/chukyo0/toushin/05013101.htm(参照 2017-07-05).

[125] 山本雅基：ぺた語義:実践的情報教育の教育効果をはかる, 情報処理,vol. 58, no. 12, pp. 1124–1127(2017).

[126] 山本雅基, 小林隆志, 宮地充子, 奥野拓, 粂野文洋, 櫻井浩子, 海上 智昭, 春名修介, 井上 克郎：enPiT における教育効果測定の実践と評価, コンピュータソフトウェア, vol. 32, no. 1,pp. 213–219(2015).

[127] 吉田智子 : 文系学部の情報教育へのプログラミングの導入—PEN を用いた実践例—，情報処理学会研究報告 コンピュータと教育 (CE)，64(2008-CE-095)，pp. 71-78(2008)．

[128] 米屋美雪，内田 奈津子：分岐を学ぶために効果的な教材を考える—お絵かきプログラミングによる入門科目を振り返って—，研究報告コンピュータと教育（CE）2021-CE-159(2021).

[129] 渡辺敦司：情報 A が再び現象に　2012 年度高校教科書採択状況—文科省まとめ (下)，内外教育，2011 年 12 月 13 日号，pp.8-15 (2011)．

[130] 渡辺敦司：英語 III，高学年周期でも冊数減 19 年度高校教科書採択状況—文科省まとめ（下），内外教育，2019 年 2 月 22 日号，pp.12-19 (2019)．

[131] 渡辺勇士, 中山佑梨子, 原田康徳, 久野靖：幼稚園児のビスケットプログラムにおける動きの方向の理解についての分析, 情報処理学会論文誌教育とコンピュータ, Vol.6, No.1, pp.28-39(2020).

[132] 渡辺勇士, 中山佑梨子, 原田康徳, 久野靖：幼稚園児のビスケットプログラムにおける繰り返し続けるプログラムの理解の分析, 情報処理学会論文誌教育とコンピュータ，Vol.7, No.1, pp.38-49(2021).

[133] 和田勉：小中高等学校の新学習指導要領とそれを取り巻く情報教育の状況, 情報処理，Vol. 59，No. 8，pp.742-746 (2018).

付録

A　プログラミングパートの演習問題 (2017 年度)

1〜5 回のプログラミングパートで用いた演習問題について，電気通信大学「基礎プログラミングおよび演習 2017」[79] から利用した演習問題の対応と，追加した問題を示す.

第 1 回目

演習 1〜3：P.7 演習 1〜3

第 2 回目

演習 1 P.21：演習 1
演習 2：独自問題

> 演習 1 の問題のプログラムを打ち込んで動かせ．動いたら「誤差を指定する版」でも途中経過と回数が表示させるように直して動かしてみよ.

演習 3：P.27 演習 4
演習 4〜6：P.28 演習 5〜7

第 3 回目

演習 1〜 4 ：P.38 演習 1〜 4
演習 5：P.40 演習 5
演習 6〜7：P.41 演習 6〜7

第 4 回目

演習 1〜2：P.53 演習 1〜2

第 5 回目

演習 1：P.63 演習 1
演習 2：P.65 演習 2
演習 3〜4：P.67 演習 3 〜4
演習 5：P.68 演習 5

B　プログラミングパートの演習問題 (2018 年度)

第 1 回目

例題：三角形の面積

```
def triarea(w, h)
  s = (w * h) / 2.0
  return s
end
```

演習 1 例題の三角形の面積計算メソッドをそのまま打ち込み、irb で実行させてみよ。数字でないものを与えたりするとどうなるかも試せ。

演習 2 三角形の面積計算で、割る数の指定を「2.0」でなくただの「2」にした場合に何か違いがあるか試せ。

演習 3 次のような計算をするメソッドを作って動かせ。[1]

a. 2 つの実数を与え、その和を返す (ついでに、差、商、積も)。何か気づいたことがあれば述べよ。

b. 「%」という演算子は剰余 (reminder) を求める演算である。上と同様に剰余もやってみよ。何か気づいたことがあれば述べよ。

c. 円錐の底面の半径と高さを与え、体積を返す。[2]

d. 実数 x を与え、x の平方根を出力する。さまざまな値について計算し、小数点以下何桁くらい表示されるか (計算の精度がどれくらいあるか) 検討せよ。[3]

e. その他、自分が面白いと思う計算を行うメソッドを作って動かせ。

[1] 1 つのファイルにメソッド定義 (`def ... end`) はいくつ入れても構わないので、ファイルが長くなりすぎない範囲でまとめて入れておいた方が扱いやすいと思います。

[2] 円錐の体積は「底面積×高さ× $\frac{1}{3}$」。底面積は「πr^2」。

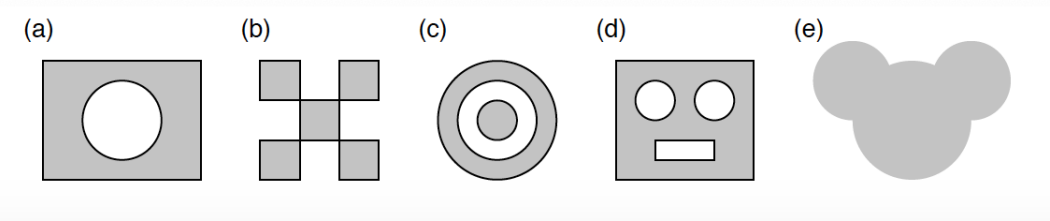
[3] x の平方根 (square root) は `Math.sqrt(`x`)` で計算できます。または x の 0.5 乗 (`x ** 0.5`) としても計算できます。

例題：図形生成のプログラム

```
$img = Array.new(200) do Array.new(300) do [255,255,255] end end
def pset(x, y, r, g, b)
  if 0 <= x && x < 300 && 0 <= y && y < 200
    $img[y][x][0] = r; $img[y][x][1] = g; $img[y][x][2] = b
  end
end
def writeimage(name)
  open(name, "wb") do |f|
    f.puts("P6"); f.puts("300 200"); f.puts("255")
    $img.each do|a| a.each do |p| f.write(p.pack("ccc")) end end
  end
end
def fillrect(x0, y0, w, h, r, g, b)
  (y0-h/2).to_i.step(y0+h/2) do |y|
    (x0-w/2).to_i.step(x0+w/2) do |x| pset(x, y, r, g, b) end
  end
end
def fillcircle(x0, y0, rad, r, g, b)
  (y0-rad).to_i.step(y0+rad) do |y|
    (x0-rad).to_i.step(x0+rad) do |x|
      if (x-x0)**2 + (y-y0)**2 <= rad**2 then pset(x, y, r, g, b) end
    end
  end
end

def mypict1
  fillcircle(50, 80, 30, 255, 0, 0);
  fillrect(100, 50, 80, 60, 0, 255, 0);
  writeimage("pict1.ppm")
end
```

演習 4 例題プログラムの mypict1 をそのまま動かし、生成される画像を確認しなさい (画像の表示方法は環境により異るが、「display pict1.ppm」「gimp pict1.ppm」などでできる環境が多いはず)。確認できたら、色や図形の位置、大きさを変更して変化したことを確認しなさい。それができたら、以下のような図形を描くプログラムを作成してみなさい (色は自由に決めてよいです)。



演習 5 長方形と円を組み合わせた自分の好きな絵を考え、それを描くプログラムを作ってみなさい。

本日の課題 1A

「演習 3」「演習 4」で動かしたプログラム (どれか 1 つでよい) を含むレポートを提出しなさい。プログラムと、簡単な説明が含まれること。

次回までの課題 1B

「演習 4」(の小課題)、「演習 5] から 2 つ選択してプログラムを作り (ただし 1A で提出したものは除外、以後も同様)、レポートを提出しなさい。プログラムと、課題に対する報告・考察 (やってみた結果・そこから分かったことの記述) が含まれること。

第 2 回目

演習 1 絶対値計算プログラムの好きなバージョンを打ち込んで動かせ。動いたら、枝分かれを用いて、次の動作をする Ruby プログラムを作成せよ。

a. 2 つの異なる実数 a、b を受け取り、より大きいほうを返す。

b. 3 つの異なる実数 a、b、c を受け取り、最大のものを返す。(やる気があったら 4 つでやってみてもよいでしょう。)

c. 実数を 1 つ受け取り、それが正なら「`'positive'`」、負なら「`'negative'`」、零なら「`'zero'`」という文字列を返す。

演習 2 「ブレーキのかかるボール」の例題をそのまま動かしなさい。動いたら、次のように直してみなさい。

a. 例題ではボールが水平に動いていたが、斜めに動くようにしてみなさい。ヒント: y も dy を用いて同じようにする。

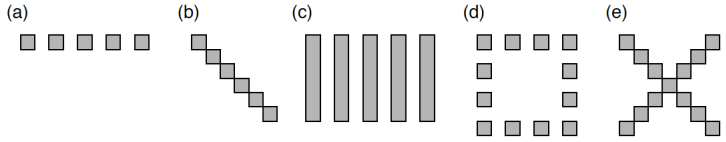
b. 水平方向にはブレーキがかかるが、縦方向には一定の速さで動き続けるようにしてみなさい。ヒント: dy は一定のまま変化させない。

c. 水平方向には一定の速さで動き続けるが、縦方向には徐々に速くなるようにして「落ちる」感じにしてみなさい。ヒント: dx は一定で、dy は小さい値からはじめて 1 より大きい値を掛けて大きくしていき、一定値 (たとえば 80) より小さい間だけ繰り返す。

d. a〜c と似ているが、画像の下端に来たら跳ね返るようにする。跳ね返った後どうするかは自分で好きに決めてよい。

e. その他自分がやってみたいと思う好きな「ボールの軌跡」を描く。

演習 3 上の例題のプログラムを動かせ。動いたら、図のようなものを作ってみよ。色や繰返し回数などは好きにしてよい。



演習 4 はね返りの例題をそのまま動かして確認しなさい。円の間隔や色などを変更してみるとよい。そのあと、次のことやりなさい。

a. 間隔が一定でなく徐々に長く/短かくなるようにしてみなさい。

b. 1 回だけ色を変えるかわりに「1 つおき」や「2 つおき」に変えるようにしてみなさい。ヒント: 「2 で割った余り」「3 で割った余り」が特定値になるかどうかで判定する。

c. X 方向だけでなく Y 方向も跳ね返るようにしてみなさい。

d. 上と同様だが、画像の上/左端でも跳ね返るようにしてみなさい。ヒント: 「dx が正の時は画像の幅を越えそうなら反転し、dx が負のときは画像の左から出そうなら反転する」などとする。

e. ボールが 1 つでなく 2 つになるようにしてみなさい。ヒント: x, dx, y, dy の 4 つの変数が 1 つのボールを表しているので、もう 1 組変数を用意する (名前は違える必要)

f. その他好きな跳ね返りの絵を作成してみなさい。

本日の課題 2A

「演習 1」「演習 2」で動かしたプログラム (どれか 1 つでよい) を含むレポートを提出しなさい。プログラムと、簡単な説明が含まれること。アンケートの回答もおこなうこと。

Q1. プログラムを打ち込んで動かすのに慣れましたか?

Q2. 自分にとって次の「難しいポイント」は何だと思いますか?

Q3. 本日の全体的な感想と今後の要望をお書きください。

次回までの課題 2B

「演習 2〜演習 4」の (小) 課題から選択して 2 つ以上プログラムを作り、レポートを提出しなさい。プログラムと、課題に対する報告・考察 (やってみた結果・そこから分かったことの記述) が含まれること。アンケートの回答もおこなうこと。

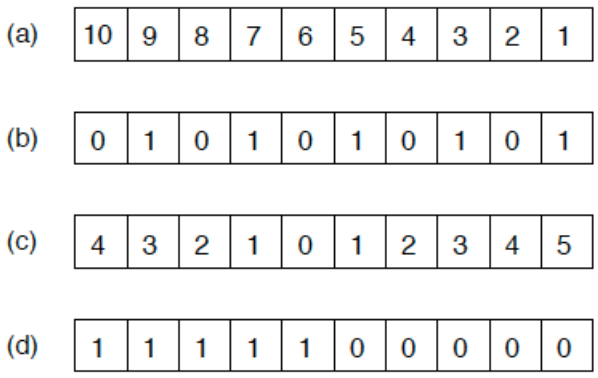
Q1. 枝分かれや繰り返しの動き方が納得できましたか?

Q2. 枝分かれと繰り返しのどっちが難しいですか? それはなぜ?

Q3. 課題に対する感想と今後の要望をお書きください。

第 3 回目

演習 0 ブロックを指定する形で 10 要素の配列を生成し、初期値を (a)〜(d) のようにしなさい。



演習 1 上記の配列合計プログラムの好きな方をそのまま打ち込んで動かせ。動いたらこれを参考に下記のような Ruby プログラムを作れ。[4]

a. 数の配列を受け取り、その平均値を返す。

b. 数の配列を受け取り、その最大値を返す。

[4]「返す」の場合は上の例と同様に `return` を使い、「出力する」の場合は `puts` を使って画面に直接 (その場で) 出力させてください。`return` は使った瞬間にそのメソッド呼び出しは終ってしまうので、複数回 `return` を使うことはできません。

c. 数の配列を受け取り、最大値が何番目かを返す。なお先頭を 0 番目とし、最大値が複数あればその最初の番号が答えであるとする。

d. 数の配列を受け取り、最大値が何番目かを出力する。なお先頭を 0 番目とし、最大値が複数あればそれらをすべて出力する。

e. 数の配列を受け取り、その平均より小さい要素を出力する (例: 1、4、5、11 → 1、4、5)。

f. 数の配列を受け取り、その内容を「小さい順」に並べて出力する (例: 4, 11, 5, 1 → 1, 4, 5, 11)。

例題：複数のボール

前回の演習の中に「ボールを 2 つに増やしてみなさい」というのがありました。しかし、ボールの数だけ変数 x、y、dx、dy を増やすのは大変ですね。でも、配列を使えば簡単です。x、y、dx、dy をすべて配列にし、その i 番目は「i 番目のボールの X 座標」等にすればよいのですから。とりあえず、3 つのボールでやってみましょう。

```
def threeballs
  x = [100, 130, 160]; y = [20, 20, 20]
  dx = [3, 8, -2]; dy = [3, 6, 7];
  50.times do
    3.times do |i|
      fillcircle(x[i], y[i], 10, 255, 0, 0);
      if dx[i] > 0 && x[i] + dx[i] > 300 then dx[i] = -dx[i] end
      if dx[i] < 0 && x[i] + dx[i] < 0 then dx[i] = -dx[i] end
      if dy[i] > 0 && y[i] + dy[i] > 200 then dy[i] = -dy[i] end
      if dy[i] < 0 && y[i] + dy[i] < 0 then dy[i] = -dy[i] end
      x[i] += dx[i]; y[i] += dy[i]
    end
  end
  writeimage("pict3.ppm");
end
```

演習 2 「複数のボール」の例題をそのまま動かして確認しなさい。ボールの動き方や色をいろいろ変えて試してみること。納得したら、次のことをおこなってみなさい。

a. 例題ではすべてのボールが同じ色だったが、ボールごとに色を違うものにしてみなさい。(ヒント: 色の RGB 値も配列で保持する必要がありますね。)

b. 例題ではボールの数が固定でしたが、一定時間たつと 1 つのボールが分裂するようにして、個数が増えていくようにしてみなさい。(ヒント: 配列の末尾に push を使って新しいボールのデータを追加する。繰り返しの数も増やす必要があるので注意。なお、増えたボールの速さや色は元のボールと変えないと意味がないので、乱数を使うとよい。乱数は「rand(N)」で 0 以上 N 未満の整数がランダムに返されるという機能を使うとよい。)

c. 絵のプログラムに配列を使った好きな工夫を施してみなさい。

本日の課題 3A

「演習 1」「演習 2」で動かしたプログラム (どれか 1 つでよい) を含むレポートを提出しなさい。プログラムと、簡単な説明が含まれること。

次回までの課題 3B

「演習 1」「演習 2」の (小) 課題から選択して 2 つ以上プログラムを作り、レポートを提出しなさい。配列の内容を含むことを強く勧めます。プログラムと、課題に対する報告・考察 (やってみた結果・そこから分かったことの記述) が含まれること。

第 4 回目

例題：最大公約数

```
def gcd(x, y)
  while x != y do
    if x > y then x = x - y else y = y - x end
  end
  return x
end
```

例題：約分

```
def yakubun(a, b)
  c = gcd(a, b)
  return [a / c, b / c]
end
```

演習 1 gcd と yakubun を打ち込み、動かして動作を確認しなさい。それができたら、次のことをやってみなさい。

a. 2つの正の整数 x, y が互いに素 (coprime) であるとは、両者の最大公約数が1であることと同じである。2数 x, y を受け取り、互いに素なら `true`、そうでなければ `false` を返すメソッド `coprime` を作成しなさい。

b. 4数 a, b, c, d を受け取り、分数の和 $\frac{a}{b}+\frac{c}{d}$ を既約分数 (を表す2要素の配列) として返すメソッド `tashizan` を作りなさい。

c. 配列の要素の和を計算するメソッドを下請けにもちいて、配列の平均を返すメソッド `heikin` を作成しなさい。(ヒント: x が整数のとき、実数に変換するには x`.to_f` を使います。)

d. 次のメソッド `swap` は配列 `a` の `i` 番目と `j` 番目の値を交換する。

```
def swap(a, i, j)
  x = a[i]; a[i] = a[j]; a[j] = x
end
```

これを下請けに利用して、配列 a の長さを n として、「0番とと n-1 番」「1番と n-2 番」…「$\frac{n}{2}$ 番と $n-\frac{n}{2}-1$ 番」を順次交換することで、配列の中身を前後逆転するメソッド `hanten` を作りなさい。

e. 次のメソッド `arrayminno` は配列 `a` の `i` 番目から `j` 番目までの範囲で最小の値が何番目かを返す。

```
def arrayminno(a, i, j)
  p = i; min = a[i]
  i.step(j) do |k|
    if a[k] < min then p = k; min = a[k] end
  end
  return p
end
```

これと上記の `swap` を下請けにして、配列 a の長さを n として、「0 番目と 0〜n-1 番の範囲の最小要素を交換」「1番目と 1〜n-1 番の範囲の最小要素を交換」…「n-1 番目と n-1〜n-1 番の範囲の最小要素を交換」することで、配列の中身を小さい順に並べるメソッド `naraberu` を作りなさい。

例題：旗を描く

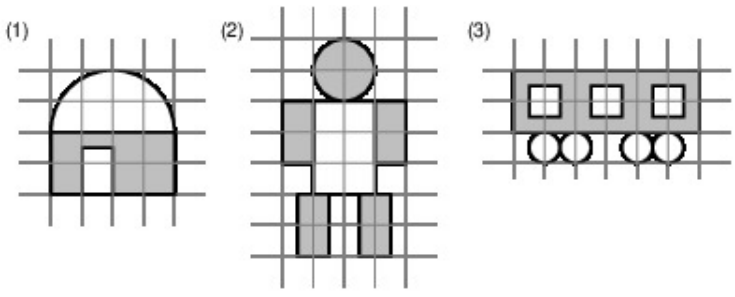
```
def flag(x, y, r, r1, g1, b1, r2, g2, b2)
  fillrect(x, y, r*6, r*4, r1, g1, b1)
  fillcircle(x, y, r, r2, g2, b2)
end
def mypicture4
```

図 B.1: 旗が 3 つある絵

```
    flag(100, 120, 20, 150, 150, 150, 255, 0, 0)
    flag(180, 80, 15, 200, 250, 40, 0, 150, 100)
    flag(220, 110, 10, 20, 40, 80, 250, 200, 200)
    writeimage('pic4.ppm')
end
```

演習 2 旗が 3 つの例題をまずそのまま動かしてみなさい。旗の位置、大きさ、色は適宜変更してみること。納得したら、次のことをしてみなさい。

a. 下の設計図のような部品を描くメソッドを作り、それが複数回含むような絵を生成してみなさい。絵の適当な箇所の座標 (x,y) とあみ目の大きさ u を指定し、色は 2 色の RGB 値 r1,b1,g1 と r2,b2,g2 を指定するようにしなさい。



b. 好きな国の「旗」を描くメソッドを作って絵を生成してみなさい。とりあえず円と長方形で済むものに限る必要があります。

c. 複数の部品が入った好きな絵を設計し作ってみなさい。

本日の課題 4A

「演習 1」または「演習 2」で動かしたプログラム (どれか 1 つでよい) を含むレポートを提出しなさい。プログラムと、簡単な説明が含まれること。

次回までの課題 4B

「演習 2」から選択して 2 つ以上プログラムを作り、レポートを提出しなさい。「演習 2c」は必ず含まれること。プログラムと、課題に対する報告・考察 (やってみた結果・そこから分かったことの記述) が含まれること。

第 5 回目

例題： 2 次元配列

```
irb> b = Array.new(5) do |i| Array.new(5) do |j| i+j end end
=> [[0,1,2,3,4], [1,2,3,4,5], [2,3,4,5,6], [3,4,5,6,7], [4,5,6,7,8]]
irb> pp b
[[0, 1, 2, 3, 4],
 [1, 2, 3, 4, 5],
 [2, 3, 4, 5, 6],
 [3, 4, 5, 6, 7],
 [4, 5, 6, 7, 8]]
=> [[0,1,2,3,4], [1,2,3,4,5], [2,3,4,5,6], [3,4,5,6,7], [4,5,6,7,8]]
```

演習 0 上の例を実際に実行してみよ。納得したら、図のような 5 × 5 の整数配列を作ってみよ。

(a)

1	2	3	4	5
2	3	4	5	6
3	4	5	6	7
4	5	6	7	8
5	6	7	8	9

(b)

1	2	3	4	5
2	4	6	8	10
3	6	9	12	15
4	8	12	16	20
5	10	15	20	25

(c)

0	1	0	1	0
1	0	1	0	1
0	1	0	1	0
1	0	1	0	1
0	1	0	1	0

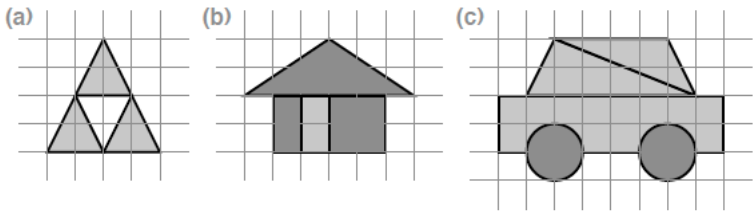
(d)

1	0	0	0	0
1	1	0	0	0
1	1	1	0	0
1	1	1	1	0
1	1	1	1	1

(e)

9	8	7	6	5
8	7	6	5	4
7	6	5	4	3
6	5	4	3	2
5	4	3	2	1

演習 1 図に示すような、三角形を含んだ図形を描くメソッドを作成しなさい。



演習 2 例題をまずそのまま動かしなさい。動いたら、次のことをやってみなさい。

a. 円の色や動き方を変更してみる。円の色が徐々に変化するとなおよい。

b. 円ではなく別の図形を動かしてみる。図形の形や色が徐々に変化するとなおよい。

c. これまでに作った「絵の手続き」を使って、家や自動車などのものが動くようにしてみる。色が徐々に変化するとなおよい。

演習 3 これまでの例題に含まれていた絵の手続きや自作した絵の手続き (これから新たに作ってもよい) を用いて、簡単なストーリー性のある動画 (例: 太陽が沈むと家のあかりが灯るなど) を作ってみなさい。

本日の課題 5A

「演習 1」または「演習 2」で動かしたプログラム (どれか 1 つでよい) を含むレポートを提出しなさい。プログラムと、簡単な説明が含まれること。

次回までの課題 5B

「演習 3」を題材としてプログラムを作り、レポートを提出しなさい。プログラムと、課題に対する報告・考察 (やってみた結果・そこから分かったことの記述) が含まれること。

C　2019 年度の事例：役割分担とプログラム

プログラミングパートの演習問題を見直し，1 回目から図形生成の教材を用いる方法に変更した．プロジェクトパートの課題は，パーツの細かいネイルシールのデザインから，変更を行った．2019 年度は，プロジェクトパートの課題を 4 分割したステッカーへの変更したところ，学生が一人 1 メソッドの作成を行い，1 つの絵に統合するという目的に叶ったプロジェクトができるようになった．図 C.2 に，2019 年度の分担とメソッド化と統合の事例を示す．

このチームは，4 種類の図案を作成した．図 C.2 は，その 1 つである．サークル，空，海，船をそれぞれ分担し，分担に対応したパーツのコードを書き，最後に 4 つのパーツを統合している．改善に至ったポイントは，プログラミングパートの演習問題の修正とプロジェクトパートの課題の設定に依るところであると考える

表 D.1: プログラミングパートの各回のアンケート

1A	Q1. プログラムに初めて取り組んでの感想． 第 2 外国語と比べてどう? Q2. Ruby 言語のプログラムを打ち込んで実行してみて, どのような感想を持ちましたか?
1B	Q1. プログラムを作るという課題はどれくらい大変でしたか? Q2. 実際に複数のプログラムを打ち込んで動かしてみて，想像したのと違うことはありましたか?
2A	Q1. プログラムを打ち込んで動かすのに慣れましたか? Q2. 自分にとって次の「難しいポイント」は何だと思いますか?
2B	Q1. 枝分かれや繰り返しの動き方が納得できましたか? Q2. 枝分かれと繰り返しのどっちが難しいですか? それはなぜ?
3A	Q1. 制御構造の組み合わせができるようになりましたか. Q2. 配列について学びましたが, 使えそうですか.
3B	Q1. 配列が使いこなせるようになりましたか. Q2. 配列を使うと「複数のもの」を統一的に扱えることに納得しましたか.
4A	Q1. 手続きの働き・役割について納得しましたか. Q2. 絵を描く手続きについてはどうですか.
4B	Q1. 手続きが使いこなせるようになりましたか. Q2. 複雑な絵を作り出すにはどうするのがコツだと思いますか.
5A	Q1. 手続きを使った絵が作れるようになりましたか. Q2. 動画の原理を理解しましたか.
5B	Q1. 動画を構想して作ることができるようになりましたか. Q2. プログラミングのコツや難しいところは何だと思いますか.

役割分担

図 C.2: 2019 年の作品と分担の事例

D　カリキュラム修正後のプログラミングパートのアンケート

E　2018 年度の授業実践の記録

E.1　プログラミングパートにおける各週の記録とまとめ

1 回目の授業実践を通じて，前半 6 回でプログラミングの知識を学ぶことに対しての方針は，特段の問題はなかった．しかし，プログラミングの基本となる制御構造の活用に課題が見つかった．授業時の課題では理解したように見えた制御構造であるが，続く後半のプロジェクトパートでは活用ができなかった．先行研究 [24][60] でも述べられていたが，プログラミングのスキルを身につけるためには，反復練習が重要であることから，この課題を改善するためにゴールを意識し，初回授業から，絵を描くメソッドを用いたカリキュラムに変更した．

以下では，修正したカリキュラムにおけるプログラミングパートの実践について週ごとにまとめる．

表 E.2 に，プログラミングパートのスケジュールを示す．

プログラミングパートのテキストは，週ごとに作成し，その他の資料やフィードバック・課題とともに，授業支援システム (moodle) を活用した．その一部を図 E.3 に示す．

表 E.2: プログラミングパートの授業内容

回数	内容
1 回目	プログラミングの導入
2 回目	分岐と繰り返し
3 回目	制御構造と配列
4 回目	手続き・関数と抽象化
5 回目	2 次元配列と画像
6 回目	プログラミングのまとめ

図 E.3: 授業支援システムの一部

1 週目 (プログラミングの導入)

履修者が確定していないため，授業の概要の説明から始め，授業の進め方と教材の説明を行い，テキストに基づき，以下をおこなった．

- 履修のためのガイダンス
- エディタや Ruby の実行環境についての説明
- Ruby のプログラムの基本であるメソッドの定義とパラメタの説明
- 例題：三角形の面積を求めるプログラム

エディタや Ruby の実行環境についての操作説明など，使うツールの準備と Ruby のプログラムの基本であるメソッドの定義とパラメタについて説明した．擬似コードを用い，三角形の面積を求めるプログラムについて説明をした後，テキストに示したプログラムを打ち込み，プログラムを書くことと実行することを実習した．

テキストに加えて操作手順の説明資料を準備し，初回 (2017 年度) の実践で不足した部分を補った．例題のプログラムは，テキストで詳細に示しているが，実行時のエラーの対応ができない学生が複数いたことから，教員 1 人での対応は苦労した．履修登録期間中のため，履修の可否が決まらない学生が多数含まれていたため，ペアを作ることはしなかった．ただし，教室がグループ机であるため，隣同士での協力を指示した．

画像を生成するプログラムについては，時間の都合で，概要だけ説明して翌週に持ち越すこととなった．時間を要した原因は，次の 2 点が考えられる．

- タイピングが遅く，プログラムの入力に時間がかかること
- パソコンに不慣れであるため，コマンドプロンプトの起動，ファイルの取り扱い (保存先，文字コードの設定，拡張子など）に時間がかかること

また，学生のフィードバックには，次のような回答もあった．

> - irb を最初に打ちこんだ時，エラーとなったため 2 回目を打とうとしましたが，何故かローマ字入力が出来ませんでした．

パソコンに不慣れであり，対応ができなかったと考える．さすがに，パソコンの電源の入れ方がわからない学生はいないが，いずれもプログラミングの本質でないコンピュータの取り扱いのところで躓いている学生がいることが読み取れる．

しかし，初めてのプログラミングであったが，楽しかったや達成感を感じたと回答したものが 17 件あった．特にペアで取り組むような指示は出していないが，教室がグループ机であることから，周りの友達と相談しながら進めるように指示をしたことで，効果的に機能したと推察できる以下の回答もあった．

> - 周りの人とも協力して頑張りたい
> - 友人と一緒に考えた出来たので面白かった

2 週目 (逐次処理と描画)

- 1 週目の復習
- 画像を生成するプログラムについての説明
- 例題・課題の取り組み

プログラムの書き方，実行の仕方を確認し，1 週目の復習を行ない，画像を生成する課題に取り組んだ．

丸と四角を描くメソッドと実行された絵のデータをファイルに出力するメソッドなどの教材は，事前に準備し，授業支援システムで配布した．

プログラムの全体構造を説明した後,「中心 (50, 80) に半径 30 の真っ赤な円，その上に中心 (100,50)，幅 80，高さ 60 の緑の長方形を描く」課題 (図 E.4) に取り組んだ．

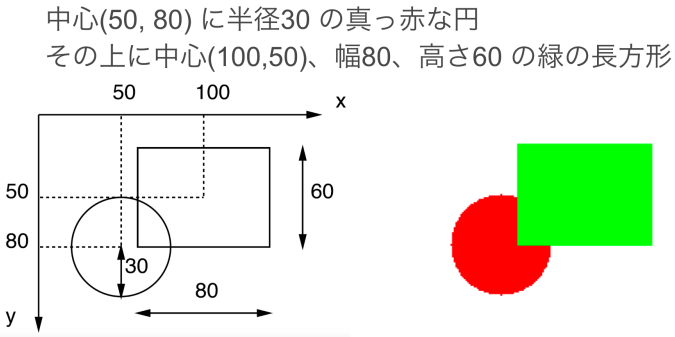


図 E.4: 画像を生成するプログラムの課題

この際，プログラムを打ち込み，プログラムをロードし，実行させるプロセスと出力された画像ファイル (ppm) を gif ファイルに変換するプロセスがあるが，後半のプロセスで，つまづく学生が目についた．その要因は，Ruby の実行環境とファイルの変換を行う環境は，コマンドプロンプト上で作業するが，その相違に関しての理解が，十分にできていなかったためである．

以下の学生のフィードバックからも，その様子がわかる．

- 行程が多くて何が，どこが間違っているのかを見つけることが，難しかったです．
- 自分が今何をするためにどのプロセスを行っているかを理解しながらプログラムを実行することが難しかったです．

学生によっては，スピードが早く，もう少し説明をして欲しいと回答するものがいたが，別の学生は，十分理解しておりスピードアップを望むものもいた．資料を配布し丁寧に説明しているが，学生毎に理解度が異なるため，歩調を合わせることは難しい課題である．

3 週目 (制御構造・分岐と繰り返し)

- 2 週目の復習
- 基本的な制御構造 (分岐・反復)

制御構造について説明し，2017 年度の実践と同様，絶対値の問題で枝分かれを学んだ．その後，円を描く仕組みを使い，繰り返しの課題に取り組んだ．

最初にブレーキのかかるボールの例，次に係数ループの仕組みとして，ボールを 10 個並べる例を提示した．

練習として，各自で四角をかく課題 (図 E.5) に取り組んだ．

演習 3 上の例題のプログラムを動かせ。動いたら、図のようなものを作ってみよ。色や繰返し回数などは好きにしてよい。

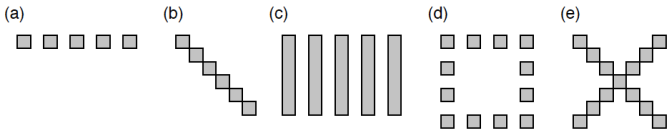


図 E.5: 授業時に取り組んだ課題

制御構造の仕組みの理解はできたが，数学的要素に対して理解が難しく，特に係数ループのカウンタの仕組みの部分でのつまづきがあった．例えば,「カウンタ変数 i を 0 から始めて 1 つずつ増やしながら n-1 まで繰り返していく」ということを考える際につまづきが多い．

多少，困難が伴った部分もあったが，授業後のフィードバックでは，提出のあった 18 名中 14 名が慣れてきたと回答していた．反復練習の重要性が理解できた学生や，自分の考えをプログラミングで表現できている学生がいた．初めはエラーの対応で苦労したが，理解して自分でエラー対応できるようになりたいと前向きな回答もあった．

4 週目 (制御構造と配列)

- 3 週目の復習
- 配列
- 課題の解説

新しい項目として配列について説明し，例題に取り組んだ．

例題で扱った内容は，配列 a の数値の合計を求める問題である．その後，最大値・最小値や平均値を求める課題に取り組んだ．やはり，算数・数学的な課題となると学生の理解度が低くなる．

授業時のフィードバックには，次のようなものがあった．

- 今回の授業は私にとって，理解するのが非常に難しかったです．算数を忘れていることを授業でわかりました．
- 私は数学，数学の解き方とかを考えるのが苦手なので，この分野は結構苦労しています．とても難しいなと思っています．
- a = Array.new(10) |i| の i とかの数字の求め方が難しかったです．

苦手意識を持つ学生にとっては，数字や数式を見た瞬間，アレルギー反応を起こすことから，理解を妨げないような配慮が必要である．一層の数学的内容の取り扱い方に工夫が必要であると感じた．

5 週目 (手続きによる抽象化)

- 配列の復習
- 手続きと構造化グラフィクス

前週の復習と配列の課題についての説明をし，新しい項目として，手続きと構造化グラフィクスについて説明した．手続きによる抽象化として，日本の国旗を描く例題に取り組んだ．絵を描く課題になると，学生からのフィードバックがポジティブになり，提出のあった 13 名中 12 名から「納得した」「理解できた」と回答があった．

以下の学生の回答からも，絵を描くことにより理解が進んでいることが読み取れる．

- 絵を描く手続きについては，完璧にこなせるようになりました．
- 絵を描き始めるとしっくりきます．
- やっぱり，数の配列とかより絵を描く手続きのほうが私は理解しやすいです．
- 今回は絵を描くプログラム作りだったのでいつもよりスムーズにできていたかなと思います．

前週と比較して，絵を描く行為は，数学的な課題よりも視覚的に理解しやすく学生の理解も深まっており，2017 年度の課題となっていた，抽象化の部分について，理解が深まり，解決できている手応えがあった．

6 週目 (描画ライブラリの内容と動画)

- 5 週目の復習
- 描画ライブラリ

- 動画生成の仕組み

5 週目の復習に加え，さまざまな図形 (三角，楕円など) を描くメソッドを加え，自分で考えた絵を描くプログラムを書くことと，動画の生成について取り組んだ．

授業後のフィードバックに回答のあった 13 名中 13 名が，手続きを使った絵が作れるようになったと回答した．また，動画の生成の仕組みも 10 名が理解したと回答した．提出された課題からも，理解されていることがわかった．

- 方眼紙を使って考えるのは大切だと改めて思いました．
- コンピューターは決まったことを繰り返すのが得意だからそれを利用すれば案外簡単なんだなと分かった．

設計図の重要性についての回答もあった．また，コンピュータの特性とプログラムを書くことに対しての理解が進んだと思われる回答もあった．

中間課題

前半のプログラミングパートのまとめとして，中間課題を課した．授業開始後，ある企業から協力をいただけることとなり，急遽，絵を描く課題を印刷してトイレットペーパーの外装をデザインすることとした．その理由は，家庭や大学にあるプリンタでは印刷できない，薄い用紙への印刷をさせてもらえること，また合わせて会社訪問をさせてもらえることからである．単に，課題に取り組むだけでなく，作品として仕上げること，企業訪問をして先端技術に触れることができることで，学生のモチベーションが高まった．

課題には，以下の条件を課した．

- 繰り返しや分岐を使うこと
- 絵のパーツごとに手続き (丸や四角を逐一プロットするのではなく，まとまった図形はメソッドにする) を用いること

図 E.6 に，授業支援システムに掲載した課題の内容を示す．

授業では，縦 200 ピクセル×横 300 ピクセルというサイズを使用したが，この大きさは適宜変更して指定のサイズ (縦 11cm ×横 32cm) に収めるよう指示をした．文字を入れたい学生もいたことから，多少の自由度を持たせ，最終的なデータは，学生が使い慣れているパワーポイントでまとめることとした．完成した作品は，図 6.1 を参照のこと．

提出されたプログラムについては，6.5.1 参照のこと．

トイレットペーパーの外装の課題

前半のまとめの課題です。

トイレットペーパーの外装をデザインしてください。

ルールは、前半で学んだ

・繰り返しや分岐

・手続き（丸や四角を逐一プロットするのではなく、まとまった図形はメソッドにする）

を使うこと。

授業では、縦200×横300というサイズを使っていましたが、この大きさは適宜変更してデザインしてください。

プログラムから出力された図をPowerPointにはり、データを完成させます。サイズは、A3用紙の真ん中に縦11cm×横32cmの枠に収まるように貼り付けます。このサイズの枠内に収まれば、複数の図を貼り付けても構いません。また、文字なども適宜追加しても結構です。

提出するものは、

1. デザインの下書き

2. プログラムと出力されたデータ

3. トイレットペーパーの外装をデザインしたPPT

締め切りは、11月26日（月）20時までとします。

図 E.6: 中間課題の内容

E.2　プロジェクトパートにおける各週の記録とまとめ

プロジェクトパートでは，1 チーム 4〜5 名で構成されたチームで，クリアファイルのデザインとそれを PR するアニメーションの作成をプロジェクトとして行った．

1 つの絵を 1 人で完成させるのではなく，チームのメンバーで分担して作成することをルールとし，前半で習得した，枝分かれ，繰り返し，手続きを活用することを指示した．各週の内容について，以下で詳しく述べる．

7 週目 (グループ作成・課題とゴール設定)

- グループを作る
- 役割分担をする
- チーム内での課題とゴール設定をする

企業の方を招き，制作依頼を受けるという形で，課題テーマを提示した．「横浜」「バラ」「ズーラシア」の 3 つのキーワードから複数のキーワード (2 つ以上) を選択し，ターゲットを決め，クリアファイルのデザインをすることと，それを PR するための gif アニメーションを作成することを課題とした．

特に，次の 3 点について示し，チームでの活動をワークシートを用いて記録をとった．

- チームの目的を共有する
  チームとして達成すべき目的を共有し，そのために何をしなければいけないかを認識することで，チームとしての役割の存在を認識する必要性
- メンバーに役割を意識させる
  それぞれに役割を割り振ることで，自分がチームのためにやるべきことを認識
- メンバーに役割（仕事）を任せること
  自分がチームのためにやるべきことについて一任することで，責任感をもって役割を果たす

授業後のフィードバックでは，次のような回答があった．

- 目的やビジョンを持って取り組む重要性に気づかされました．今後は行うことをゴールとして学習や制作に取り組むのではなく，そのプロセスを通して目的を達成することを意識して取り組みたいです．
- 今までは課題を提出することを目標としてしまったので，これからはさらに一工夫してオリジナルの絵が描けるようにしたいと思いました．

これらの回答からは，中間課題で作品を作る経験があったこと，企業の方のプレゼンテーションの影響により，結果よりもプロセス重視するという姿勢への変化があったのではないかと考える．

8 週目 (アイデア出し・ブレスト・仕様決め)

- 要望・要求・要件・仕様の違いについて
- システム開発の進め方
- 今後の流れ (プロジェクトの進め方とスケジュール)
- 前週の振り返り (チーム名，役割分担)
- アイデアを話し合い，作成するものについてまとめる
- 次回に向けての課題の確認をする
- 情報共有の方法を決める

授業の初めの 20 分ほどで，前週の振り返りと本時の説明を行い，残りをチームでの作業として進めた．グループワークでは，ワークシートを用い取り組んだ．図 E.7 に，チーム用のワークシート Step0〜2 を示す．

授業後のフィードバックでは，プロジェクトを進める上で，チームとしてうまくやっていくための工夫について聞いた．代表的なものを，以下に挙げる．

- GoogleDrive, LINE などを使って情報共有をする
- それぞれが担当の業務をしっかりこなして，さらに担当だけでなく，チームで意見を出し合ったり，時間を見て協力し合う
- 画像での共有 (自分が思っている，考えているものを正確に伝えるため)
- 活動内容を LINE のノートなど，わかりやすい形でまとめ，進み具合ややってほしい内容の見える化を図りたいです．

多くの学生は，情報共有について書いていた．次いで，自分の担当を全うすることがあった．少数であるが，考えていることを正確に伝えるために，画像での共有や，わかりやすくまとめること，見える化することなどの回答があった．

単に情報を共有するだけでなく，明確に伝える意識が現れていることは，プログラムを書く上で重要な要素であり，前半で学んだことが定着していると推察できる．

9 週目 (デザイン・設計・分担・制作)

- デザインの決定と設計
- 作業分担の決定
- 試作

試作から完成への手順をどうするかについて，より具体的に進めるよう指示をし，グループワークを行なった．

学生のフィードバックには,「自分が作るものが決まりまった」「なるべく想像している完成形に近づけたい」というものが複数あり，ゴールが決まり，意識が高まっていることが確認できた．

その他の特徴的な回答を，以下に示す．

- デザインをもっと広げるかっていう話が出たが，もうある程度いいものが決まっていたので，デザインは今のものを各自進めつつ変えたい部分があればその都度話し合うという話がでたので．最重要だったのは，やはり役割分担を決めることでした．それが決まらないと何も進められないから．

チーム名：

グループワーク２

STEP0　先週の振り返り

チーム名、各自の分担：理由をかならずかくこと

#グループワーク①に追記

依頼は何？

STEP1 アイデアを話し合う

グループで話し合ったことを記録しましょう!!

アイデアがまとまったら、設計図にまとめていきます。

テーマ：

対象：

STEP2

情報共有の仕方について、グループで決めたルールをまとめましょう。

チームで作成するものについてのまとめ

#次回以降、デザイン・作業担当を決めて作業に入ります。

図 E.7: Step0～2 のワークシート

- 人数が増え，やることや細かい調整などを綿密な連絡が必要であると感じましたが，実際の仕事の様なチームでの取り組みを経験できると思うと楽しみです．
- 最初のうちはみんなの理解度やアイデア力，発言力にばらつきがあったのでごたついたが，デザインの案を考えみんなに話を振りいろいろな役割のきっかけを与えて得意分野をなんとなくとらえて分担することができた．
- ３人でやっていく前提で考えなくてはいけなくなり，話し合いに多くの時間を割きました．画像作成は個々で行うことになりますので，意見を共有しあうことで間違いがないようにしたいと思いました．

正解がない課題であるため，さまざまな思いがあるとは思うが，ゴールを変えず，役割分担をしっかり行い，完成を目指す意識が見られた．ゴールがぶれないことは，プロジェクトの基本であり，経験から学ぶ部分が大きい．そのため，こういう回答が出てきたことは，この授業の効果であると考える．

また，チームの個々のメンバーの多様性やチーム内での共通理解を理解し，チーム力が増してきている様子がうかがえる．また，リーダと連絡が取れなくなったチームがあり，作業の方向変換を真剣に検討した様子がわかった．

複数で取り組むプロジェクトでは，多様な人々との関わりや，予想外の事故なども起こりうることから，こういった経験も良い学びになってくると考える．

10 週目 (試作・中間ドキュメント作成)

- 中間発表用ドキュメント作成

この回は，教室に集合して実施せずにチームごとの作業とした．誰が見ても同じように図の生成ができるよう，ドキュメントを書く事を目標とした．

各チームから提出されたドキュメントについてまとめたものを，表 E.4 に示す．

表 E.3: 中間ドキュメントの内容

項目	A	B	C	D
テーマの設定	○	○	○	○
役割分担	○	○	○	○
デザイン	○	○	一部	一部
試作	×	○	一部	×
設計図	×	イラスト的	イラスト的	△
その他		一部，統合テスト		

テーマの設定や役割分担は，どのチームもできているが，絵のアイデアはできているが，設計図ができていなかったり，イラストを描き，試作をしているとこがあったりとさまざまである．

以下に，特徴的なものを取り上げる．

下絵はあるが，座標の指示がないために，どうやってプログラムを書くのかわからないものの例を，図 E.8 に示す．

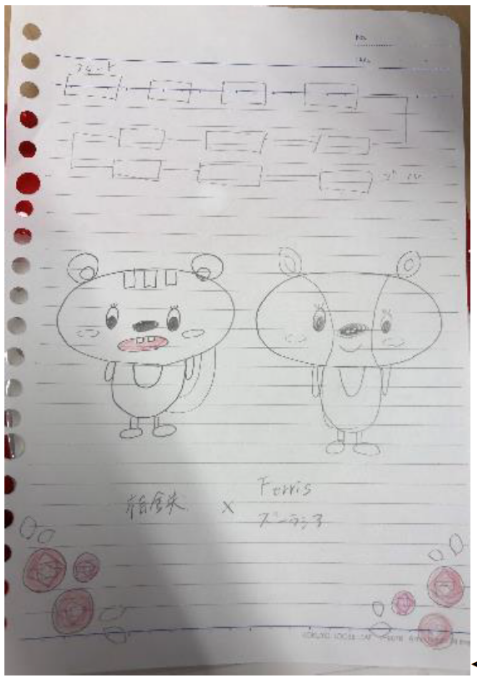


図 E.8: 中間ドキュメントの事例 1

下絵と座標の指示があるが，メソッド化できるかどうか，この時点では不明な書き方をしているものの例を，(図 E.9) に示す．

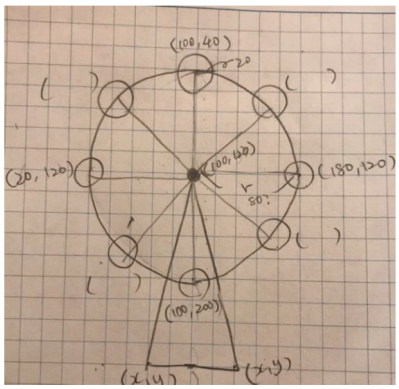
全体授業を行わなかったため，授業のフィードバックは実施しなかった．

11 週目 (中間発表・仕様見直し)

- 中間発表
- 仕様見直し

前週の中間ドキュメント作成の準備を活用して，発表を行なった．中間発表会用のコメントシートを用意し，各チームの良かった点と改良点や自分たちのチームに取り入れたい点，全体の感想をそれぞれ書き，発表会後に全員で共有した．

⑷横浜観覧車
　次にみなとみらいの有名な遊園地にある観覧車のデザインです。円と線、三角形を用いて観覧車を表す絵をプログラムしました。また、観覧車の彩りを表現するために、ゴンドラ部分は鮮やかな色彩になるよう設定させます。
以下が設計図と、実際にプログラムした画像です。



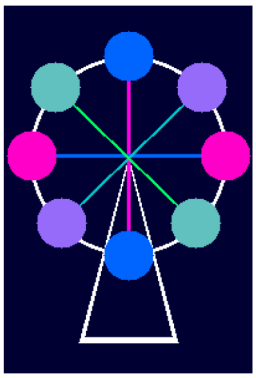


図 E.9: 中間ドキュメントの事例 2

表 E.4 に，改良点や自分たちのチームに取り入れたい点を，表 E.5 に，全体の感想を著者がまとめたものを示す．

教室内では，活発に質疑応答ができなかったが，コメントシートには，それぞれの意見がや感想が書き込まれたいた．班ごとの進捗に差があるものの，指摘しているポイントに関しては，学生個々に大きな差はなかった．

表 E.4 からは，自分たちのチームと他のチームを比較して回答されているものが多かった．良い点として上がっていた内容は，次の 3 点である．

- デザインの観点
- 役割分担
- スケジュール

説明だけで終わってしまい，視覚的に見せるものがなかった箇所については,「下書きがあるとわかりやすかった.」「下書きが細かくないため，全く同じものが作れない.」など，それぞれに気がつき，設計の重要性や情報を明確に伝える重要性などの理解が深まったと推察される．

表 E.5 の個々の感想では，作業の進め方や情報共有の重要性が多く回答されていた．下書きの設計図とプログラムの関係についても，うまく表現できていないが，理解していると思われる回答があり，この後の進め方に期待ができる．

授業後のフィードバックには，次のような回答があった．

- 実際に発表をしてみると，自分達の計画の曖昧さに気づかされた．他の班の発表から学ぶものも色々とあったので，それを活かしていきたい．

表 E.4: 中間発表会のコメントシートの回答

チーム名	良かった点	改善点や取り入れたい点
A	・アニメーションにも取り掛かっている計画性 ・役割分担がいいと思った. ・私のチームは，バラの花を描くのが難しいと思っていたのですが，花びらだと描きやすいと思った. ・ひとりひとりが自分の分担に対して責任を持って取り組んでいるところ.	・下書きが細かくないため，全く同じものが作れない
B	・過程やターゲットがしっかりしていたと思いました. ・デザインがすでにだいぶ出来上がっている仕事の早さ役割が明確になっている	・自分たちのグループにも言えることですが，下書きがあるとわかりやすかった. ・下書きが細かくないため，全く同じものが作れない. ・メインをはっきり決めた方が良い.
C	・バラの作り方なども自分のチームと違っていて面白いと思いました. ・オリジナルキャラクターを作っているのがすごいと思った.	・作るパーツが多く，難しそう. ・下書きが細かくないため，全く同じものが作れない ・今後の計画たてが難しそうだった. ・個々の単体としての下書きは見れましたが，その絵がどのように並ぶのかわからなかった.
D	・販売の方法等，事細かに決めていること，デザインのこだわり ・今後の予定を立てている.	・複雑なプログラムが多く，難易度が高いと思った. ・下書きが細かくないため，全く同じものが作れない ・役割分担ができているかがあいまい (?) ・色などの共有も必要だと思う.

表 E.5: 中間発表会のコメントシートに書かれた感想

・コンセプトとずれずに今後も作っていけたらいいです．
・もう少し中間発表のために時間を割いてチームで協力しながら役割をなるべく均等に分けられたらよかった．
・班のメンバーが誰も予定が合わず，集まって進めることが出来ませんでしたが，その分，Googledrive やラインで状況を共有しながら進めることが出来た．
・他の班がアニメーションまで行っていないのに驚いた．私たちのグループは，アニメーションとファイルを完全に分割していたので，進みが早いように見えるけど，まだより良くすることができると思うので，試行錯誤していきたい．
・どの班もそれぞれいろんな視点で考えていて面白かった．
・ここで描いた絵が合わさった時に，色のバランスや形が見えて調整をどうしたらいいかなど，改善点がわかりました．下書きしたものと，プログラミングで写し出したものとの再現度がどれだけ高いかが大事なのだなと思いました．
・PowerPoinr をやっつけ仕事のように作成したことを非常に後悔した．
・PowerPoint にまとめられていない班とそうでない班で，伝わり方に大きく差があったので，次の発表時に備えようと思いました．また，プログラムの書き方や誰が見てもかけるように準備してけたら，様々な FeedBack が得られたかと思うので，次回頑張ります．
・パワーポイントの準備を怠るなど，準備不足が多かったので改善すべきだと思いました．準備不足の原因としては，グループ間の連絡不足だと思うので，連絡を取っていきたいと思いました．
・プログラムの問題としては，配置やどのように作りたいのか，視覚的に情報共有した方がわかりやすいと思いました．

- 様々なチームの中間制作過程を知ることができ，自分のチームに取り込めることがたくさんあると思いました．

この回答からもわかるよう，チームを超えて相互に影響を与えることがあり，チーム内では気が付かない点にも目が向いていることがわかる．

全体的に，デザインが決まり動き始めたばかりなこともあり，直接プログラミングに関わる回答は少なかったが，プロジェクトを通じて総合的な学びがうかがえる．

12 週目 (制作・単体テスト)

- 中間課題および中間発表の振り返り
- 仕様見直し
- 単体テスト

中間課題と前週の中間発表から，コードを書く上で考えてもらいたいこと 3 点 (図の抽象化と引数, ループ) を学生の提出物を用い，補足説明を加えた．その後，チームごとに，中間発表でのコメントシートを振り返り，仕様の見直しと単体テストを行う時間とした．

学生のフィードバックには，

- 中間課題のように 1 人で熱中することも楽しくやりがいがあったが，グループで行うとまた違う面白さや難しさがあった．メンバーが困っている際に私が改善点を指摘することで解決に繋がることがあったので，今まで学習したり身につけたことが活きて嬉しかった．
- 誰が見ても分かる設計図がないので，本来作った後に書くものではないのは分かっているがしっかりしたものを作っておこうと思う．
- 教えながらだとなかなか進まないことがわかりました．事前にそれぞれの課題を確認しておくべきでした．
- 自分の作ったプログラムを他の方が作ったプログラムと重ねてみて，少しずつですが形になっているのが実感できました．

これまで学習してきた知識の活用に関する回答があった．また，1 人でやること，グループで取り組むことの面白さについて回答されている点は，本研究で目指す部分である．

前週の中間発表から，設計図の重要性にようやく気がつくケースも見られ，順序が多少異なっても，残りの授業に活かされ定着することで，本授業の目標が達成されると考える..

グループでの取り組みが，完成に近づき少しずつ形が見えてくる体験は，他の授業では得られ難く，とくに，実践で対象となっている学生は，他の授業でこのような経験をする機会が乏しいことから，新鮮であり，モチベーションの向上に繋がっていると観察できた．

13 週目 (コードレビュー)

- コードレビュー

Google スライドの共有を使ってコードレビューを行った．各自，分担している部分を一つ取り上げ，下書きの絵，プログラム，出力された絵をまずスライドに貼り，チーム内でのレビューを行った (図 E.10 参照)．その後，各チーム 1〜2 名の代表者を決め，クラス全体でのレビューを行なった．本来，コードレビューは，完成したものに対して行うことを説明した上で，本授業では，完成していなくても良いこととし，進捗の共有や困っていることの相談の場とし，クラス全体で課題解決をする場とした．

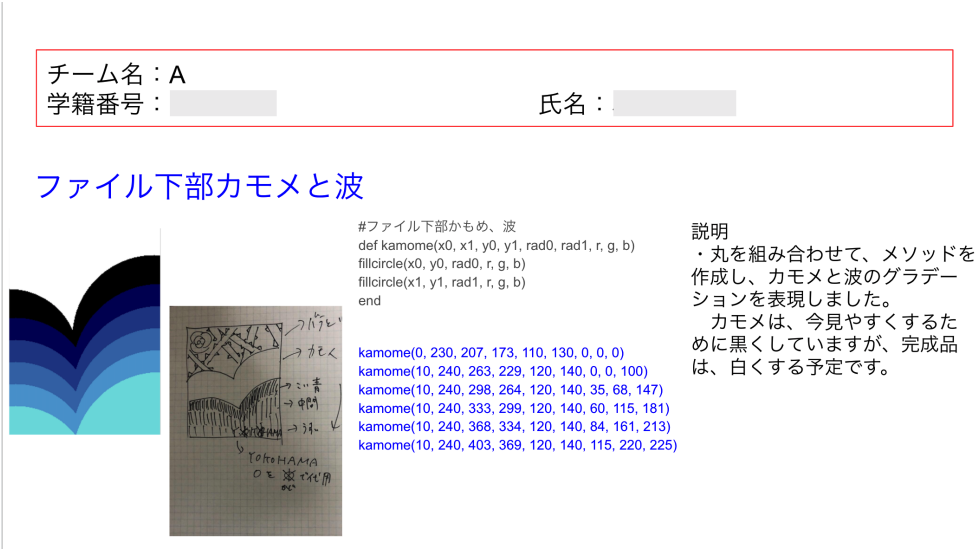


図 E.10: コードレビューの例

授業後のフィードバックからは，次のような回答があった．

- 自分が作成したコードの問題点について，対処法をグループ全体で話し合うことができました．グループメンバーが過去に作成した類似プログラムを参考にしながら取り組めました．
- 他の人の意見なども聞く事で，自分ではどうしてもうまくいかない部分について少し解決の糸口を見出すことが出来たので，第三者から頂いた意見をもとに，プログラムの改善をしていきたいと思いました．

- コードレビューでみんなで情報共有すると他の人のコードを見て知りえることも多かったです.
- 作成した絵などもみることができ，完成にむけて意識があがります.

これまでは，チームのメンバーとの相談であったが，全体で共有したり相談したりすることにより，解決につながったとの回答が複数あった．チーム内の状況は，常に確認できるが，他のチームの状況も共有できたことで競争心が芽生えモチベーションの向上となっている様子も，授業中の様子からうかがえた.

また，次の回答では，リーダシップや，チームビルディング，チームワークについての認識が現れている.

- 本日の授業では，こうしようと素早く決断する人がグループワークには必ず一人は必要だと実感しました.
- 今回のコードレビューでとことん話し合いそれぞれの進捗状況がはっきり分かってよかった．ようやくチームという感じになったと思う.
- 個人の中で起きていた問題をチームで共有することでいくつかが解決し，チームプレイをしているという実感が湧いた.

以下のような回答もあった.

- この授業全体を通して言えるのは手伝ってくれる人がいる事はありがたいという事と，積極的に手伝って欲しいと伝えることが大切だと思います.

著者は，毎回の授業で，わからないことは悪いことではないので，どこがわからないのかをはっきり伝えるように指導してきた．その効果だと思うが，一人ではなできないことを協力して行うことの重要性が理解されている点は，本授業が有意に機能している結果だと考える.

14 週目 (制作・統合テスト)

- 統合テスト
- 最終成果発表会準備

教室で行う最後の授業で，個別に試作してきた各パーツのプログラムの統合作業を行い，完成させることを目標とした.

授業後のフィードバックからは，前週のレビューの効果が現れている回答があった.

- 新な課題，改善策など，先週よりもより良いものになっていっていることが感じられてうれしい．

多くのものづくりの授業では，時間の都合もあり，一度完成したらそこで終わってしまうケースが多々ある．しかし，本取り組みでは，発表の時間や再考する時間を設け，チーム内や教室内で検討する時間を設けた．これにより，試行錯誤を繰り返し，教えあうことによりトラブルを回避したり作品の質を向上させている様子があった．

また，以下の回答からは，スケジュール通り進められている様子がうかがえる回答や，作品を完成させるための調整や完成度を上げるための工夫が見られる．さらに，仕事の段取りや仕事の分担を，個々の特性に合わせて調整する姿勢も見られる．期限に間に合わせるための意識を持って取り組む姿勢や，チームのメンバーと協力して完成させる態度などは，これからの社会で求められている資質・能力の向上につながるであろう．

- 本日の授業ではそれぞれの担当パーツを一つの画像としてまとめることができました．
- 遅れながら進み，最後に詰めて作業をして進め，本来目指していた目標に何とかたどりつきました．最後にしてやっとやりたいことができた授業でした．
- それぞれの完成してる絵を一つずつ統合していって，微調整していきました．クオリティをあげるため，変更すると起こるエラーで，どうしてもわからないものについては，諦めて元のままのデザインにしました．(現状を考えて，時間もあまりないため．)
- 互いに意見を出し合って改良した．よりよくするためのアイデアだけでなく期限に間に合わせるために妥協する部分も考えた．
- 全員の課題を出して，今日の授業でやるべきポイントを確認してから，取り組んだので，効率的にタスクを分担しながら進められたと思います．
- 皆で協力して行ったため，早く仕事が終わりました．また提出までの間の担当がわかってきました．
- できることをできる人で分担し効率よく進めていけました．プログラムの統合とパワポの制作を並行してやっていきました．

15 週目 (最終成果発表会 (まとめ))

場所を教室から協力企業のショールームに移動して，作品完成と発表会を行なった．1 チーム 15 分 (発表 10 分・質疑応答 5 分) で，次の項目について発表するよう事前に伝えた．

- テーマとコンセプト・対象
- プロジェクトをどのようにすすめて来たか
- デザイン・設計図，作業分担等を含めて
- 完成した作品とアニメーション
- プログラムについて
- 各自分担したところの説明，プログラミングを勉強してのまとめ
- 授業を通じて学んだこと（成功・失敗両面から，チームでまとめたものと個人的なもの両方あると良いです.）

中間発表の際と同様に，評価シートを用い，各チームのよかった点・改良点についてコメントし，終了後に共有した.

各チームの発表内容を，表 E.6 に, 完成した作品を図 E.11 に示す.

図 E.11: 最終成果物のクリアファイルのデザイン

チームで話し合い，ターゲットやデザインを決め，プロジェクトを進めたことについて発表した．クリアファイルは，発表会以前に企業へデータを渡し，完成した作品は，当日の発表の後で手渡した.

各チームが，授業を通じて学んだことについて，当日の発表資料および，発表から著者がまとめたものを表 E.7 に示す.

各チーム，それぞれ多くの学びがあったと思われるが，特に印象的であったものをまとめて話をしていた.

どのチームも，一人ではできないことが，チームで協力することにより達成できたことが理解されたことがわかる．習熟が浅いものは，前半の振り返りをし定着を図り，また，新しいことは自ら調べて対応する力がついた姿勢も現れていることがわかる.

表 E.6: 最終成果発表会のチーム毎の発表内容

チーム名	発表内容
A チーム	タイトル：薔薇とカモメと横浜と．Ruby... デザイン決定後，ファイル班とアニメーション班に分かれて作業． ファイル班：4 つの絵を作成，アニメーション班：2 つのアニメーションを作成 デザイン案から試作，完成まで，段階を追って説明．プログラムをまとめて 1 回で出力できるように繰り返しのプログラムを変更し，その結果として，最終的に，1 つのメソッドにできた．
B チーム	タイトル：『魅力あふれる夜のみなとみらい』 5 名で 7 つのオブジェクトを作り，1 つの絵に仕上げた．アニメーションは，完成しなかった． 抽象化する際に，様々な考え方があることをコードレビューで学び，試作から完成で改良を行った．
C チーム	タイトル：相鉄線認知度ＵＰ作戦〜相鉄で子供たちを集めよう！脱少子高齢化〜 ファイルは，4 つのオブジェクトで構成．アニメーションは，ストーリーがある． コードを統合した際に発生した失敗談を共有
D チーム	タイトル：ズーラシア × 濱うさぎ × 写真 途中で，リーダーとの連絡が取れなくなり，最後は 3 名で完成させる ファイルは，5 つのオブジェクトを作成．アニメーションは，完成せず． 作成した人以外にも理解しやすいよう，コードにコメント

表 E.7: 最終成果発表会のチーム毎の発表内容：授業を通じて学んだこと

チーム名	授業を通じて学んだこと
A チーム	・学んだことを振り返ることと，わからないこと，新しいことを調べて対応する力 ・はじめは，わからないことがあった時に，相談するタイミングがわからなかったが，自己表現できるようになり，コミュニケーション力が向上した ・1 人で悩むのではなく，グループで相談して取り組んだことは支えになり，1 人では成し遂げられないこともグループでできた． ・チームで集まってやらなくてはならないことは，授業時間内で十分話し合うよう心がけて取り組み，チーム力で納得がいく作品を完成させた．技術だけでなく，プロジェクトを進める知識を学んだ．
B チーム	・授業内での話し合いにおいてはスムーズに作業できたが，離れている時にうまく作業ができなかったためもっと情報共有すればよかった． ・絵を描くのにも何通りものやりかたがあること ・個人でできなかったことが，グループワークで上手く対応することができた
C チーム	役割分担の難しさ ・計画を立て，計画通りに行うことの難しさ ・情報共有の難しさ ・もらった資料・以前行った作業を振り返ることの大事さ
D チーム	計画とチーム内での協力 ・進行状況の共有と連絡 ・分担と作業量の差

良かった点

各チームの発表に対してコメントシートの良かった点についての回答から，特徴的なコメントを著者がまとめたものを表 E.8 に示す．

改良点・改善点

コメントシートから，各チームの発表に対して改良点・改善点について，特徴的なコメントを著者がまとめたものを表 E.9 に示す．

自己評価

コメントシートから，発表後の自己評価に対して，特徴的なコメントを著者がまとめたものを表 E.10 に示す．A チームは，最初のデザインが明確でなかったために，進めていく過程で修正を加え，最終的には良いものに出来上がった．分担が明確であったことが，効果的に働いたと考えられる．今回は，初めてのプログラミングということだったので，ゴール設定をしてプロジェクトに臨むカリキュラムを準備したが，進めていく中で，近年言われている OODA(Observe（観察），Orient（状況判断，方向づけ），Decide（意思決定），Act（行動）) ループ [8] が体験できたと思われる．

B チームは，アニメーションの完成ができず，悔しい気持ちが残ったが，他のチームの発表を聞き，自分たちが足りなかったことやできていなかったことがより明確になったと推察できる．プロジェクトを通じて，できたことできなかったことが明確になったことで，プログラミングの知識だけでなく，社会に出てから必要なスキルも学べたと考える．

C チームは，チーム内のスキルの差と意識の差が，他のチームと比較し，大きかった．リーダーのやりたい気持ちと他のメンバーの気持ちのズレからコミュニケーションがうまくいかなことがあった．最終的には，技術的にも素晴らしい作品が出来上がったが，技術面の前に，チーム力を高めるべきだったという回答が得られたことは，この授業の醍醐味であったと考える．コードを書きものを作る経験を通じで，一人でできること，一人ではできないことを認識できたと推察できる．

D チームは，プロジェクトの後半で，リーダーとの連絡が取れなくなり，作業分担から再調整をするなど苦労したが，クリアファイルまで完成できたことはよかった．途中の状況は問題があったが，それにより，残ったメンバーでの協力と分担は良くできており，プログラムにコメントを入れ，チーム内での共有に配慮した点にもそのことが現れていた．表 E.9 の各チームの改良点・改善点のコメントにもあるように一人抜けても，人が入れ替わってもプロジェクトが進められることの重要性を，チームのメンバーだけでなく，他のチームにとっても認識され

表 E.8: 各チームの良かった点

A チーム
・問題点を把握して，解決策をチームで考えているところが良かった． ・繰り返しなどをうまく使って表現しているのが驚きだった． ・授業中は，情報共有の場という認識が大切 ・ファイルの画像とアニメーションの動画を変えている点． ・バラのコードを見て，r,g,b の値を 255〜0 のようにしていて，ひとつひとつ r,g,b を打ち込むのではなく，こういったやり方にすることで，プログラミングをやっているという感じがして真似したいなと思いました．
B チーム
・最初の設計図から改良したもの，最終デザインがわかるようになっていて，どの部分を直してきたのかがはっきりしていて良かった． ・できないところを共有しあって，最後まで試行錯誤をでたきた点． ・授業内で言われたことをきちんと活用していて良かった． ・できること，得意な点をできる人が対応していくという形を自然にできていた． ・効率を考えたりとプログラムにもやりやすくする工夫がされていてとても良かった．
C チーム
・アニメーションがすごすぎる!! ・テーマの対象がはっきりしていることや，分担がはっきりしているのがいい点だと思いました． ・アニメーションがストーリーになっていてびっくりした． ・失敗からその失敗の原因も分析されていたり，それを発表に組み込んでいた点． ・プログラムができない人は，パワポをするという切り替え
D チーム
・リーダーが抜けても，諦めずに進めていたところ． ・いかにして，分担をし，連絡を取り合うのか，締め切りから逆算することの大切さを学びました． ・表・裏を作成しようという心意気がすごいと思いました． ・思いとおりに行かなかったところも多かったと思いますが，その中で，できること，できないことを分別して，ファイルデザインまで行っていたのがとても良かったと思いました． ・ファイルだけで完結させるのではなく，使う人が写真を入れることでその人だけのファイルになるというアイデアが面白いと思った． ・バラはとても精密であり，一番良くできていました．

表 E.9: 各チームの改良点・改善点

A チーム
・コミュニケーションの点 ・情報共有の点において，改善点を早い段階で見つけられていたら，伝達ミスや構想の違いでロスタイムが生まれずに進められたかと思います. ・波の色のグラデーションとカモメの色の関係
B チーム
・スケジュール管理については，もう少し考えてやった方が，アニメーションの完成につながったのではないかと思います. 私たちが制作した以外の場所を動かすという案をいただき，工夫がもっともっと考えられたと感じた. ・ランドマークタワーの下の部分が，他のものに隠れてしまい，上の部分しか写っていなかったので，その部分だけ作れば良かったのではないかと思う.
C チーム
・特定の 1 人の負担がとても大きいように感じられた. ・作業者同士での情報共有が足りなく，時間が奪われてしまった点. ・作るものがお互い勘違いしていた点.
D チーム
・先生がずっとおっしゃっていた，一人抜けても人が変わっても大丈夫なように，プログラムの共有，方法や書き方の工夫が必要であったと思いました.

表 E.10: 自己評価

A チーム
・最初に考えたものよりも良い案が浮かんだ時，ためらわずにポンポン変えていったことが，結果的に想像もしなかったような良いものができた． ・他のチームに比べて分担はできていた．
B チーム
・諦めることも大事で，この次に何ができるかを考えることが大切だと気付きました． ・失敗点とその原因をもっと分析し，発表に組み込めることができたら，失敗が失敗で終わらずに自己分析が次に活かせたなと感じます． ・授業を通して学んだことで，理由がわかり，確かに私自身もただ自分で悩み，解決するのではなく，班に聞くのが大切だと再確認できた． ・情報共有の難しさ，大切さを学びました． ・「これは 2 日でできるだろう」と思っていたのですが，大間違いでした．しっかり余裕を持って調整するべきでした．
C チーム
・できないことをやってみる，どうしてできないのかを考える力がついたと思いました． ・授業を通して，必要な考え方，社会に出た時に困らない物事の考え方を学べました． ・技術面の前に，チーム力を高めるべきでした．
D チーム
・もう少し余裕を持って取り組んでいたらより良いものができた．

たと考える．

ものづくりのプロジェクトでは，作品を完成させることも大事であるが，それ以上に完成に至るまでのプロセスが重要であり，この授業を通じて，それぞれに重要なことが認識できたと考える．

以下では，授業後のフィードバックから，特徴的な回答を示す．

発表会に臨む態度

- 中間報告をはじめとして，チームメンバーの作業段階を定期的に把握していたため，これらの内容を活かしたものを作成しました．内容としては，それぞれのパーツの初期のデザインから，どのような改善点を経て最終的なデザインになったかというプロセスが明確なものになるようこだわりました．
- 動きをなめらかに見せる工夫のことや自分の失敗から伝える難しさを学んだエピソード，プロジェクトの期間だけでなく授業全体を通して成長できたことなどについてしっかり伝えることが出来て良かった．

これらの回答からは，プロジェクトの工程を振り返り，チームでどのような活動をしてきたのか，どのように考えて行動したかをよく振り返り，正しく伝える行為につながっていると感じる．チームワークを通じで，伝えることの難しさや重要さを体験できたことで，自信に繋がった回答であると考えられる．

また，作品についての感想を述べるのではなく，チームでの取り組みのプロセスや，プロジェクトだけではなく授業全体を通じて触れられている点は，プロジェクトによるプログラミング入門での効果であると考える．

プログラミングの知識の習得

プログラムの知識の習得の際に必要と言われる反復練習の成果について，

- ここまで早く手続きが作れる様になったのは，授業の成果だと感動しました．
- 2月1日の発表までのことを振り返ると，最後の一週間でプログラムの理解が深まり，力がついたと思います．プログラムの基礎を学んだ前半はできなかった動画や，手続きもやるしかない！という気持ちで，わからなくてももらった資料を見て，考えてやればできることがわかりました．もらった資料をしかっり読むこと，自ら調べてやってみることが大切だということを学びました．

多くのプログラミングの授業では，例題を演習し，課題問題に取り組んで終わってしまうことが多い．また，例題を少し手直しし，暗記してテストに臨むような学び方も多い．さらに，実践を行なった学校では，1回の授業で完結する内容が10数回行われるような授業が多い．

しかし，本取り組みでは，初期に学んだことを後半で活用するような仕組みとしており，初期に学んだ知識が曖昧であったり定着していなければ，うまく活用することに至れない．

これらの回答からは，繰り返し学んだことによる効果が表現されていると推察できる．また，複数人のプログラムを一つにまとめることで，問題が発生し，振り返って学ぶことにより，足りないスキルが補われたと考える．

チームワーク・役割分担の重要性

- 四人の内，一人でも抜けてしまったらチームが回らなくなるという認識を共有できませんでした．
- 私たちのチームはハプニングがあって，当初予定していたものとは違うものになってしまいました．でも，それをカバーしようと残りのみんなで最後の数日ものすごい時間をかけてやってなんとか完成できてよかったと思います．与えられた役割をきちんとこなす，全うすることがいかに大事なのか実感しました．
- 発表会直前になってから三人で作業を進めることが出来たが，一人に遠慮することなくもっと自分の意見を言い，出来ていないところ，曖昧なところを早く指摘するべきだった．
- 発表の準備の途中からチームで仕事が出来る人が減ってしまい，私は書記の職を担っていたため，リーダーの仕事の代わりとまではいかないけど，なるべくチームの人達に早く連絡を回したり，提案したり，提案があった決定しなくてはいけない事を決めたり，といった事を意識して行動しました．

これらは，主に，リーダーが途中から来なくなってしまたチームのメンバーからの回答である．デザインが決まり，試作を始めて完成に向けて改良したり統合したりという重要な時期に，リーダが欠席するようになり，最初は連絡が取れていたものの，連絡が取れなくなった．チームのメンバーは，連絡が取れていることではじめは安心していたが，その連絡が途絶え，残ったメンバーで完成させることとなった．そのため，特に，チームワークと役割分担，そしてコミュニケーションについて回答されたと推察できる．しかし，突然のトラブルも柔軟に対応し，努力した様子が読み取れる．

他者からの学びと失敗の共有

- 他のグループの発表を聞いて，どのグループも失敗を繰り返してきましたが，それをどのようにそれを改善してきたかということを発表しており，私たちのグループもああしていればよかったと思えるような改善策も見つかったりと，そういった発表でしか見つけることのできないことが多くありました．終わった今だからこその発見もかなりありました．

失敗の共有をすること，そしてそれをどのように改善したかという点を共有することは，どの授業でも経験できることではない．プログラミングの良い点は，やりたいこととゴールを明確に示すことができ，その途中の過程も実行時の動作で確認できることである．どのチームにも同様の経験があり，自分たちの経験と重ねて考えることで，共感を得て効果的に働いたことがわかる．この回答に限らず，コードを書き，絵を描くこと，プロジェクトを進める上でのプロセスの両方で，失敗や成功があったと思われるが，授業を通じてこの両方の学びが体験でき，共有と今日考えらていることは，プロジェクトによるプログラミングの学びの効果であると考える．

ICT リテラシーに関すること

著者は，近年の学生の動向を見ていてパワーポイントなどのツールは，ほとんどの学生が触れる程度には利用したことがあると想定していた．しかし，次のような回答があり，驚いた．

- パワーポイントはこの授業（中間発表時）で初めて使用し，どのようにすれば見やすくなるか，注目を集めることができるか等考えたが，中間発表を含め，納得できるものを作成することができなかった．

大学入学までの多様な環境においては，こういうオフィスツールを使ったことのない学生も少なからずいることがわかった．本授業では，レポートにまとめたり発表する機会を用意しており，これらを通じてオフィスツールなども活用する機会が作れ，ツール活用のスキルアップにもつながったと考える．しかし，納得できるものを作成することができなかったとの回答から，著者がもう少し注意深くフォローするべきであったかもしれない．

関連論文の印刷公表の方法および時期

(1) 全著者名：内田奈津子，久野靖，中山泰一
論文題目：PBL によるプログラミング入門科目の提案：一般情報教育における入門カリキュラムの構築
情報処理学会論文誌，第 62 巻第 7 号，pp.1393-1414，情報処理学会.
(第 3，4，5，7 章の内容に関連)

(2) 全著者名：Natsuko Uchida, Yasushi Kuno
論文題目：Programming Education in a Women University in Japan: A Case Study of Performance-based Learning in Liberal Arts
2019 年 9 月, International Congress on eLearning (ICE) 2019 Conference Proceedings, pp.175-189, Philippine eLearning Society.
(第 6 章の内容に関連)

著者略歴

内田　奈津子（うちだ　なつこ）

山梨県甲府市に生まれる

津田塾大学学芸学部数学科 卒業
山梨大学大学院工学研究科博士前期課程
電子情報工学専攻 修了

フェリス女学院大学
教育・研究用情報システムセンター助手を経て

2003年4月～　フェリス女学院大学
情報センター講師 現在に至る

2016年10月　電気通信大学大学院情報理工学研究科博士後期課程
情報・ネットワーク工学専攻 入学
2021年 9月　電気通信大学大学院情報理工学研究科博士後期課程
情報・ネットワーク工学専攻 修了