コンピュータサイエンスの大衆化 ―教科「情報」に CS の「知恵」を―

久野　靖
電気通信大学

教科「情報」と情報入試のインパクト

高等学校の教科「情報」は，2003 年に選択必履修の形で始まった．筆者をはじめ情報教育に関心を持つ研究者は，これで「すべての高校生が情報を学ぶ」のだから，世の中は大きく変わる（つまり情報，ひいてはコンピュータのことをみんなが知っている世の中になる）と期待した．

しかし，そうはならなかった．大学入試にほとんど関係せず，単位数も少ないことから，「情報」は日陰の存在で在り続けた．教える教員も非常勤や「情報」の免許を持たない教員への臨時免許等の付与が多かった．そしてコンピュータサイエンス（CS）について言えば，選択される情報科目のうち，プログラミングをはじめ CS に関係の深い内容を含む科目の履修人数は 10 ～ 20% であり，その選択は生徒ではなく学校によってなされた．

ただ，CS 以外の内容で，目に見える変化もあった．それは「著作権」で，文部科学省がその教育に力を入れ，どの情報科目選択でも教えられるようになっており，また専門性が高くない先生であっても扱いやすかったためか，高校卒業生はだれでも著作権とその遵守について知っていて，その点では社会人より優っている，という状況にある☆1．

そうこうするうちに，2022 年から情報科は「情報 I」が（選択なしに）必履修となり，すべての生徒がプログラミングを学ぶこととなった．そして文部科学省の指導もあって，情報の免許を持つ教員が情報科を教えるという「当たり前の」ことが，ようやくほぼ実現しつつある．

これに合わせて，2025 年大学入試から「情報」が大学入学共通テストに加わり（科目は「情報 I」），国立大学受験生は必ずこれを受験することとなった．2025 年の「情報 I」の受験人数は 30 万人超で，共通テスト受験者の 60% 強であった．

これらにより，CS に関係の深い内容についても，著作権と同様，「高校卒業生は誰でも知っている」になるのでは…… このように「必履修」と「情報入試」の情報科に与えるインパクトは大きく，これによって 2003 年当時の「期待」を再度押入から取り出している，というのが筆者の現況である．しかし虫食いやカビは大丈夫だろうか…….

「情報 I」とコンピュータサイエンス

コンピュータサイエンス（CS）とは，コンピュータの仕組みやデータ処理，アルゴリズム，プログラミングなど，情報技術に関連する幅広い分野を研究する学問を指し，情報科学や情報工学とも呼ばれる．より広く，情報の原理や情報社会（いずれも文系）等まで含めた場合は情報学[1]と呼ばれるが，本稿では理系の CS の範囲を考える☆2．

「情報 I」の内容範囲と CS とのかかわりを図 -1 に示した．こうして見ると，およそ半分の内容が CS に関係しているといえる．「情報デザインと効果的なコミュニケーション」については，情報デザイン，特にユーザインタフェースが CS の内容となる．「モデル化，シミュレーションとモデル評価改善」については，シミュレーションの部分，「データの蓄積管理と

☆1　情報科の始まりから 20 年が経過した今日では，「若手は知っているが，シニアはあまり知らない」状況になっている．

☆2　今日では CS に「社会的視点と情報倫理」など文系的なものも含めるようになっている．

表現，データの収集整理分析」についてはデータベースやデータの表現の部分が CS の内容となる．

コンピュータサイエンスで学ぶこと

ここで「期待」の中身を見直してみる．「みんなが情報のことを知っている」というのは，たとえばプログラミングであれば，誰もがコードを書いたことがあり，相当数がそれを得意とする，つまり必要ならコードを書いて問題を解決できる，ということになるだろうか．

これは，我々コンピュータ屋（大学等で CS を学んだ者）には「当たり前」だったが，それが「我が国の全国民」になるとすれば（時間はかかるにせよ），画期的な変化だと言ってよい．まさにコンピュータサイエンスの大衆化である……，と喜んでばかりはいられない．

というのは，我々が CS で学んでいることは決して「プログラムが書ける」「データベースにデータが保管できる」「ネットワークが使える」こと**だけ**ではないからである．

我々は「〜できる」に加えて，さまざまな「知っておくべきこと（知恵）」を身に付けさせられ，その結果その道の専門家になっている．それ自体はこれからも続くだろう（図 -2 左）．

一方，「大衆化」によって「〜できる」を学んだ多数の人は，専門家までは必要としない範囲の CS のあれこれを担うようになると期待される．しかしその人たちは，「知っておくべきこと」を学ぶ機会はない，というのが現状である（図 -2 右）．

（1）情報社会の問題解決
　情報と情報技術を活用した問題発見・解決
　情報と法・制度，セキュリティ，責任とモラル
　情報と情報技術の活用，望ましい情報社会構築
（2）コミュニケーションと情報デザイン
　メディアとコミュニケーション手段の特徴
　情報デザインが人や社会に果たす役割
　情報デザインと効果的なコミュニケーション☆
（3）コンピュータとプログラミング
　ハードの仕組み・特徴，情報表現と計算の限界★
　アルゴリズムとプログラミングによる活用★
　モデル化，シミュレーションとモデル評価改善☆
（4）情報通信ネットワークとデータの活用
　ネットの仕組み，プロトコル，セキュリティ★
　データ管理，ネットと情報システムのサービス★
　データの蓄積管理と表現，データの収集整理分析☆

★—CS が中心となる，☆—CS と一部関係

図 -1 「情報 I」の内容と CS

そうでなく，「〜できる」とともに「CS の知っておくべきこと」を，大衆化に応じたレベルで身につけてもらう必要はないだろうか．つまり「情報科で全員が学ぶ内容」[☆3] に，そのような内容を加えるような活動をするべきではないだろうか．それがあるべき「コンピュータサイエンスの大衆化（Popularization of Computer Sciense, PCS）」であろう．これが現在筆者が思うことである[☆4]．

プログラミング：CS で学ぶこと

いちばん分かりやすいのは「プログラミング」なので，これから取り上げよう．プログラミングは「書ける」ようになるまで大変なので，とかく「〜できる」方向だけに注力されがちである．しかしすこし考えてみれば，我々は「書ける」に加えて次のことも知っておくことが大切だと分かっている．

- プログラムは「何をするかが明確で」「それを正しく行う」ように作るべきだ．きちんと考えずにプログラムを作るとこれらが毀損しやすいが，速く動き便利に見えても，正しくないプログラムは無価値（むしろ有害）である．
- プログラムは速さや小ささよりまず「分かりやすく」書くべきだ．そのほうが間違えにくく（間違ったコードは速くても無価値），直しやすく（大半のコードは間違いや改修のため直される），誰にも読みやすい（コードは他人が直すことも多いし，1 週間後の自分は他人[☆5]）．

☆3 現時点では必履修科目「情報 I」2 単位がこれに相当するが，将来的には時間数や科目が強化されてほしいと思う．

☆4 本会情報入試委員会での議論では，CS を学ぶとは「作る人」を目指すことであるが，それは行きすぎで，「よく分かっている，使う人」を目指すのでいいのではないかという意見もあった．

☆5 コードは書いたときは当然分かっているが，1 週間もすると細かいところを忘れてきてしまうことを指して「1 週間後の自分は他人」という言い方をする．

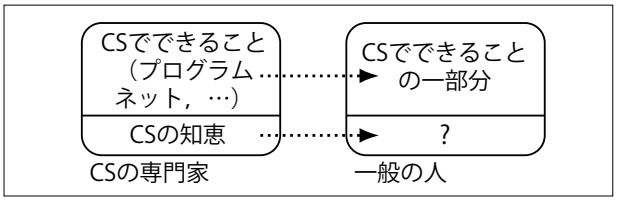

図 -2 コンピュータサイエンスの大衆化

- プログラムはきれいに「構造化」して（関数に分けたりオブジェクトを構成したりして）書くべきだ．そのほうが 1 つずつの単位が小さくて分かりやすいし，単位の外側からは内部の詳細を気にせず「抽象化」して扱える．
- プログラムには適切にコメントをつけ，どの関数は何をするか，どのデータ構造はどう使われるかなどが分かるようにする（コメントで不足ならドキュメントを書く）．そうしなければ，そのコードは苦労して読み解かなければ手がいれられず，使い続けられない．
- プログラムは適切にテストされるべきだ．コードはバグを含むのが普通であり，テストなしには信用できない．テストは保管され，コードを直したときは再実行されることで，信用を維持しなければならない．

CS の教えはまだまだたくさんあるが，大衆化のためということなら，この 5 つくらいが妥当そうである．そして，これらを知らないか無視していると必ず痛い目に遭うことも，我々はよく知っている．

一方で，「自力で」これらをクリアする人は多くなさそうである．ということで，これらは「全員が学ぶ内容」に入れてほしい．

ソフトウェア工学：CS で学ぶこと

前章では「プログラムを作る」と言ってきたが，自分以外の人が使うものを，かつ／または複数人数で開発する，つまり「ソフトウェア開発」の場合は，「作れる」に加えて，またさらにその方面の CS の教えを知っておくことが大切である．

- ソフトウェアはやみくもに作りはじめるのでなく，まず要件・要求をまとめ（＝利害関係者やその求めるものをはっきりさせ，どのように問題を解決するか決める），要求仕様を定める（どのようなソフトで，何と何が機能として含まれるか決める）．これらが明確でないと，誰も望まないものができてしまう恐れがある．
- ソフトウェア開発でコードを書く場合の生産性は単に「プログラムを書く」場合よりずっと小さくなる．これは文書化やテストに手間が取られることもあるが，複数人で 1 つのものを作ることの難しさによることが大きい☆6．
- ソフトウェア開発では開発プロセス（開発を進める枠組み）を決め，それに基づき進捗を管理する．分かりやすいのはウォーターフォール型☆7 だが，後になって仕様を変更するというトラブルが付きものである．これに対し，反復型のプロセス☆8 は仕様変更の問題は少ないが，運用に難しさがある．

ソフトウェア工学に関する CS の教えは，大衆化により，自分たちが開発者となるため必要になる面もあるが，開発者と協力してソフトを作る（ユーザサイド）場合にも大切である．

データベース：CS で学ぶこと

データベースは情報システムの中核であり，重要なものだということはよく知られている．しかし，現在の「情報 I」の教科書中では，データベース自体に割かれるページ数は多くなく，関係データモデルと DBMS の主要な機能について一通り述べられている程度であることが普通である．

一方，我々はデータベースについて，「データベースにデータが格納でき，問い合わせできる」ことに加え，次のことを知っているべきだと考える．

- データベースは「データ独立☆9」を実現すべきであり，これによって複数のプログラムが共通の

☆6 このため，早く開発しようと人を増やすと，逆にスケジュールは遅れること（ブルックスの法則）が知られている．
☆7 要求仕様・設計・制作・テストのように段階を経て開発を進め，後戻りを許さないプロセス．
☆8 何回も同じ段階を（らせん状に）繰り返しながら開発を進めてゆくもので，アジャイル型と呼ばれるものが代表的．
☆9 データとプログラムが互いに独立で，性能や機能のためにデータの形を変更してもプログラムの修正が不要なこと．たとえば，あるテーブルに新しい機能のためフィールドを追加しても，その機能に関係しないプログラムは修正不要なこと．

データにアクセスできる☆10.

- データベースは「データの一貫性☆11」が重要であり，一番避けるべきことは，データの損失や破損である．このためにさまざまな制約や不便さを甘受している．

これらのうち1番目については，今日では情報システムはDBMS☆12にデータを保管するのが当り前であり，システムの各所からその必要なデータにアクセスするように自然に作られるので，何かを変えるという必要性は大きくない．

一方2番目は，データ操作の記述が正しく，かつDBMSが普通に動いていれば達成されているはずだが，その原理や裏方仕事も含めて必要なことを知っていて分担しなければ，競合や事故によりデータの破損などの憂き目に遭いかねない．

「情報I」の教科書で，並行制御・障害対策・安全対策について説明しているものも複数あるが，並行制御についてはさらに次のことが必須である．

- 1カ所で複数のデータ操作を行う場合は，必ずトランザクションとして実行し☆13，原子性を持たせロールバックに備える．

これは，データベースのデータ操作を記述する者は必ず守るべきである☆14．

また障害対策について次のことが必要である．

- データベースのデータは管理者がスケジュールに従ってバックアップし，障害で必要な際はここからデータが復元される必要がある．
- 故障や電源喪失などの際にはDBMSが働いて最後の整合性のある状態が復元されるが，それがどの時点でそこから後の操作は必要なら再実行，などの判断は管理者の仕事になる．

これらは，データベースが使われるシステムを運用する者が必ず理解し，実行すべきである．

☆10 データ独立でないと，機能を増やすごとに既存のプログラムを修正するのが繁雑となり，新機能は共通データでなく別の新規データを使う方法で作りがちになる．

☆11 データが意図されたとおりに記録され取り出せること．

☆12 データベース管理システム（Data Base Management System）．データベースを実現するソフトウェア．

☆13 たとえばSQLなら複数の操作を`BEGIN`と`COMMIT`で囲む．

☆14 これを行わなくても負荷が軽いときは何事もなく動くが，負荷が大きくなると知らないうちにデータが破損する結果を招きがちである．

ネットワーク：CSで学ぶこと

ネットワーク（インターネット）は，我々の日常に深く根づいており，その使い方は言われるまでもなく分かっているという人が多いと思われる．また，現在の「情報I」でもネットワークの仕組みや構成要素，プロトコル，セキュリティについて扱うこととなっていて，ここまでに挙げたほかの分野のように大きく足りない部分はなさそうである．

それでも，「繋がらない」トラブルは，ネットワークに関する総合的な知識が必要で，簡単ではない．ただ，「繋がらない」害は目に見えるものであり，大きな危険にはつながりにくい．

どちらかというと，ネットワークが「使える」ことに加え，セキュリティの脅威に対する考え方を中心に，次のCSの教えを知っておきたい．

- ネットワークではEnd-to-End（E2E）の原則☆15が多くの場面で適用され，エラー処理やセキュリティ☆16など，これでしか解決できない問題も多い．一方で性能などE2Eだけでは対応しにくい問題もある．
- セキュリティを保った通信には暗号技術が使われるが，その際に「秘匿すべき情報を公開しない」「時とともに暗号技術の安全性は脅かされるので，時代に合ったものに乗り換えていく」などを原理とともに理解し，正しく使うことが不可欠である．

1番目については，セキュリティに関して，E2Eでない暗号化は危険ということである．

たとえば無線LANで安全な暗号をと言われるが，それはその無線LANで完結する通信（無線ルータの設定や無線接続プリンタへの印刷など）にのみ有用で，

☆15 信頼性やセキュリティなどアプリケーション固有のニーズに従う機能は，ネットワークではなく通信し合う終端ノードの機能として実現すること．

☆16 たとえば通信線ごとに暗号化する方法だと，通信線上のデータは安全でも，中継点で一度復号し次の通信線のため再び暗号化するので，中継点に対する攻撃には無力である．一方，通信の始点で暗号化し，終点まで復号しなければ（E2E），途中のどこを攻撃されても解読の心配はない．

より遠くへの通信は TLS（https）や ssh などの E2E 暗号化通信を使うべきである．

2 番目については，暗号技術の原理は現在の「情報 I」の教科書でも扱われているが，それが自分の行動とどう関係するのかという点では心許ない．たとえば公開鍵と秘密鍵の対をファイルに生成した後，公開鍵は必要に応じてコピーするが，秘密鍵は持ち出さず，他人に見られないよう保護しなければならない．また，その際に選ぶ暗号の適切な方式や鍵の長さは，計算機の速度向上とともに変わっていくことに注意を払わなければならない．

計算論的思考との違い

多くの人に CS のエッセンスを学んでもらう，というと「計算論的思考（Computational Thinking, CT）」[2] が思い浮かぶ．CT は「コンピュータサイエンティストのように考える」とも言われ，多くの人が手順的・プログラミング的思考と，さらには抽象化，仮想化，メタ認知など，より広い CS の思考法を身に付け，より良く問題解決でき，CS に親しむことをめざす考え方である（図 -3 左）．それはもちろん，素晴らしいことだと思う．

一方，本稿で取り上げているのは，プログラミング，データベース，ネットワークなどが広く世の中に行き渡りつつある中で，これらをキャッチアップしようとする人たちのために何をすべきか，である．つまり，そのような人たちが，これらを長く扱ってきた CS の中にある「教え」を知らないがために，先人の落ちた「落とし穴」に落ちそうになるのを回避させなければ，と考えるわけである．

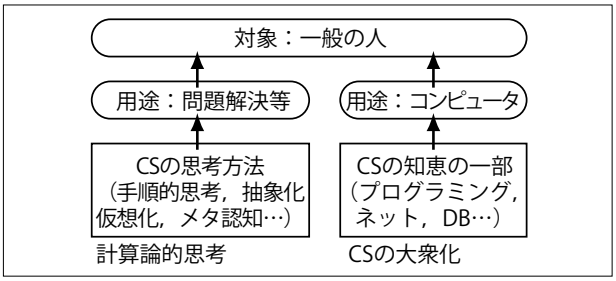

図 -3　計算論的指向と CS の大衆化の違い

そのための運動を「コンピュータサイエンスの大衆化」と名付けているわけだが，それは直接コンピュータを扱うことが前提になるので（図 -3 右），その点が計算論的指向とは別のものと言える．

コンピュータサイエンスの大衆化の今後

それでは「コンピュータサイエンスの大衆化」運動はこれから何をすべきだろう．まず，ここで挙げたものが本当にふさわしいか，時間をかけて学んでもらう価値があるか，精選していく必要がある．また，ほかに追加するものの検討もしたい，

そして，これらをどのように学んでもらうかも，考えていく必要がある．というのは，我々は CS を膨大な時間を掛けて学ぶ中でこれらを身に付けてきた．しかし，初中等教育の中で「情報」にかけられる時間は限られており，一方で CS の教えはそのエッセンスを述べただけで身に付くようなものではないと考えられるからである．

課題は山積みだが，まずはこのような課題があることを多くの人に知ってもらうこと，そして現在改訂を進めている[3]「情報教育課程の設計指針」[4] をはじめ，カリキュラム検討のもととなる文書に組み込むことを目指して活動していきたい．

参考文献

1) 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 (2016), https://www.scj.go.jp/ja/info/kohyo/pdf/kohyo-23-h160323-2.pdf
2) Wing, J. M：Computational Thinking, CACM, Vol.49, No.3, pp.33-35 (2006), https://www.cs.cmu.edu/~15110-s13/Wing06-ct.pdf, 中島秀之 訳：Computational Thinking 計算論的思考，情報処理，Vol.56, No.6, pp.584-587 (2015), https://ipsj.ixsq.nii.ac.jp/records/141823
3) 久野　靖：「情報教育課程の設計指針」の改訂について，高校教科「情報」シンポジウム 2024 秋論文集，pp.19-26 (2024).
4) 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, pp.e61-e68 (2021).

(2025 年 1 月 27 日受付)

久野　靖（正会員） skuno@acm.org

1984 年東京工業大学理工学研究科情報科学専攻単位取得退学．同大学助手，筑波大学講師，助教授，教授，電気通信大学情報理工学研究科教授を経て，同大学特命教授．筑波大学名誉教授．理学博士．プログラミング言語，プログラミング教育，情報教育に関心を持つ．