報告
翻訳

学問としての計算機分野†,††

ピーター・デニング(主査), ダグラス・E・カマーデイビッド・グリース,
マイクル・C・マルダーアレン・タッカー,
A・ジョー・ターナーポール・R・ヤング著,
木　村　　泉†††訳

ACM 創立 42 年目を迎えたいま，なおも昔ながらの論争が続いている．計算機科学は科学なのだろうか．それともそれは工学なのだろうか．または単に，機械を発明し，供給するだけの技術なのか．学問分野としての知的内容は何か．その知的内容なるものは永続性のあるものか．または世代とともに消え失せるものなのか．計算機科学ないし工学の現行のカリキュラムは，分野を正しく反映しているか．理論科目と実験科目は，計算機関係のカリキュラムの中で，どのように統合したらよいのか．われわれのコアカリキュラムは，計算機分野における職業能力を養い得ているか，といったぐあいである．

われわれは自分たちの分野に，とかく数学と工学に基づく技術的分野，というイメージを投影したがる．たとえばわれわれはアルゴリズムを最も基本的な対象と見なし，プログラミングとハードウェアの設計を主要な活動と考える．「計算機科学イコールプログラミング」という見かたは，現行のたいていのカリキュラムにおいて，ことのほか強く見られるものである．入門コースはプログラミングを内容とし，技術が基本コースに，科学は選択科目に置かれる，というぐあいである．この見かたがカリキュラムの再編成を阻害し，ひいてはよい学生がもっとやりがいのあることのほうがいいといって，ほかへ行ってしまうもととなっている．この見かたは，計算機科学の実験的部分および理論的部分を，統合され互いに調和した二つの部分としてカリキュラムに取り入れようという，一貫性のある方針を否定するものである．

専門家ならだれでも知っているように，計算機科学はプログラミングにとどまらない．たとえば

ハードウェア設計
システムアーキテクチャ
オペレーティングシステムの階層設計
応用に合わせたデータベースの構造決定
モデルの検証

などはいずれもこの分野のれっきとした一部であるが，プログラミングではない．プログラミングの重視は，プログラミング言語こそ計算機科学の全分野に手をつけてゆくためのよい道具，という昔ながらの信念に由来するものであるが，この信念こそわれわれから，自分たちの分野についてその幅広さと豊かさを余すところなく表現するような言葉で語る，という能力を奪っているものである．

計算機科学は，その知的内容を新しい，説得力のある形で記述できる段階に達している．1984年7月，ユタ州スノーバードで，計算機科学ないし工学関係博士号授与学科の学科長たちによる討論の結果，そういう認識が得られた．これをはじめとして，あちこちでなされた同種の討論に基づいて，新しいアプローチを作り上げるためのタスクフォースがいくつか，ACM および IEEE Computer Society 内に作られた．1985年春には，

1. 計算機科学の中核に関するタスクフォース(本報告書作成母体)，任命者 ACM 会長 Adele Goldberg

† *Computing as a Discipline*, by Peter J. Denning (Chairman), Douglas E. Comer, David Gries, Michael C. Mulder, Allen Tucker, A. Joe Turner, and Paul R. Young, *Comm. ACM*, Vol. 32, No. 1 (Jan. 1989), pp. 9-23 より，許可を得て翻訳．

†† この記事は，Report of the ACM Task Force on the Core of Computer Science の縮約版である．原報告書は下記に前払いで注文すれば入手できる．
ACM Order Department
P.O. Box 64145
Baltimore, MD 21264
品番は「order #201880」と指定されたい．価格は ACM 会員に対しては7ドル，非会員に対しては12ドル．

††† 東京工業大学理学部
Izumi KIMURA (Tokyo Institute of Technology, Department of Information Science).

および ACM 教育評議会議長 Robert Aiken, IEEE Computer Sciety 後援，および

2. 計算機実験実習に関するタスクフォース，IEEE Computer Society 内に結成，ACM 後援

が結成された．

この報告書に示す成果は，本タスクフォースの元来の任務を超えた意義をもっていると信じたい．われわれは，計算機科学の対象の中核部分が何であるかを明らかにすることによって，両学会におけるカリキュラムおよび模範教科課程の開発プロセスを円滑化しようと志している．またこの報告書は，①将来，職業としての計算機科学ないし工学について議論するための基盤を提供し，②高校における計算機分野の教育に改善に向けての刺激を与え，③計算機分野の重要性が社会により広く受け入れられるもととなるものと考えられる．

われわれの目的は，計算機分野というものについての新しい考えかたを作り出すことにあった．このわれわれの学問分野の本性に対する探求が広く巻き起こることを望んで，われわれは処方よりは枠組みを，手順書よりはガイドラインを狙った．読者におかれては，この枠組みを採用し，個々の状況に合わせて適用するよう願いたい．

計算機科学に関する新しい知的枠組みと，われわれのカリキュラムへの新しい基盤を示すことは，われわれのよろこびとするところである．

本タスクフォースの任務

われわれは，大まかにいえば次の三つの任務を与えられた．すなわち，

1. 計算機科学とは何であるかを，基本的な問いと主要な成果に重点を置いて述べること．ただしこの分野はつねに変化しつつあり，何を述べたとしても成長の一段階を切り出した「スナップ写真」にすぎないことを銘記すること．

2. 既存の科学的水準に合致し，計算機分野における職業能力 (Competence) の開発を重視し，かつ理論，実験，および設計を円満に統合しているような，計算機科学の教育パラダイムを提案すること．

3. そのカリキュラムモデルと分野の記述に基づいて，計算機科学入門科目系列* の詳細例を示すこと．

*【訳注】科目の早い時期に扱った事項があとで扱う事項において予備知識となるように設計された，複数学期にまたがって開講されるひとまとまりの入門科目を指して，系列と呼んでいる．くわしくは後出の「入門科目系列」の節を参照されたい．

まずわれわれは，計算機科学だけでなく計算機工学を含めることにした．というのは，中核部分ではそれらの間に本質的な違いは認められなかったからである．その間の違いは，むしろ中核部分を実体化してゆくやりかたにみられる．計算機科学は解析と抽象化に重点を置く．計算機工学では抽象化と設計に重点がある．以下「計算機分野」とは，計算機科学ないし工学全般を包含していうものとする．

二つの重要な論点が，本タスクフォースの任務から除外されている．具体的にいうと，

a．ここで示すカリキュラム案は，入門科目系列のみを示しており，コアカリキュラム全体の設計というより重要で大きな問題に触れていない．実際，ここに示す入門科目はコアカリキュラムの残部を設計しなおさないかぎり無意味であるような性格のものである．

b．ここで示す入門科目案は，入門段階の学生をこの分野の全容へと，厳格かつ触発的な形で導き入れるためのアプローチの一例として作られたものである．すなわちそれは，ここで示す計算機分野の定義が現実性をもつことを示すための，一つの「存在証明」にすぎない．この枠組みを個々の事情に応じて適用し，固有の入門科目群を開発することは，各学科にゆだねたい．

計算機分野のパラダイム

われわれが仕事に立ち向かってゆくやりかたを見ると，三つの主要なパラダイム，ないし文化的スタイルが見出される．それらはわれわれの，計算機分野とは何であるか記述するという試みにおいて，一つの手がかりを与える．そのパラダイムの第1は，理論 (theory) である．その源泉は数学にあり，次の4段階を経て，首尾一貫し，筋のとおった理論へと導くものである．

（1） 研究の対象を特徴づける（定義）．

（2） 対象間に存在すると思われる関係について仮説を立てる（定理）．

（3） それらの関係が確かに存在するかどうか確かめる（証明）．

（4） 結果を解釈する．

これが，（誤りや整合性の欠如が発見されたような場合には）繰り返される．

第2のパラダイムは抽象化 (abstraction) である．その源泉は実験科学の方法に見出され，次の4段階を経て，現象の解明に導くものである．

（1） 仮説を形成する．

（2） モデルを構成し，予測を立てる．
（3） 実験を設計し，データを集める．
（4） 結果を解析する．

これが，（モデルの予測が実験結果と合わないような場合には）繰り返される．モデリングとか実験とか呼んだほうが適切である可能性もあるが，ここでは計算機分野ではふつうそういわれているという理由によって，「抽象化」と呼んでおくことにする．

第3のパラダイムである設計（design）は，工学に源泉があり，次の4段階を経て与えられた問題を解くシステムないし装置の構築に導くものである．

（1） 要求を述べる．
（2） 仕様を述べる．
（3） システムを設計，製作する．
（4） システムをテストする．

これが（テストの結果，システムが要求を十分に満たしていないことが知れたような場合には）繰り返される．

理論は数理科学の基盤である．応用数学者はみな，これなくしては科学の進歩はありえないと信じている．抽象化（モデリング）は自然科学の基盤である．科学者はみな，科学の進歩は主として仮説を立て，モデル化のプロセスを組織的に遂行し，それらを検証，評価することによって起こると信じている．同様に設計は，工学の基盤である．技術者はみな，進歩の源は問題を設定し，設計のプロセスを組織的に遂行することによって，それを解くシステムを作り上げることにあると信じている．数学と科学と工学のいずれに優位性を認めるかをめぐってなされる多くの論争は，その根本において，これら三つのプロセス（理論，抽象化，設計）のいずれを最も基本的と見なすかに起因している．

だがよく考えてみると，計算機分野ではそれら三つのプロセスは深くからみ合っており，どれが一番基本的かを論ずることは合理的ではない．理論は，抽象化と設計の各段階にあらわれる．モデリングは，理論と設計の各段階にあらわれる．設計もまた，理論と抽象化の各段階にあらわれる．

また一方，そのような不可分性にもかかわらず，これら三つのパラダイムは，職業能力の異なる領域を示しているという意味では確かに異なっている．理論は対象間の関係を記述し，証明するという能力に関係している．抽象化はそれらの関係を利用して，実世界と比較可能な予測をする，という能力に関係している．設計はそれらの関係の特定の実現形を作り出し，それを有用な動作をさせるために利用する，という能力に関係している．応用数学者，計算科学者，および設計技術者の技能は，一般には交換可能でない．

また計算機分野では，情報変換プロセスに係わる人々を支援するための，計算機による道具を研究する，ということが行われる．たとえば設計の場面では，複雑な VLSI 設計・シミュレーション用システムを用いて，効率がよくて正しく動作するような半導体回路を設計するとか，プログラミング環境によってソフトウェアの設計作業を効率化するとかいうことが行われている．モデリングについていえば，スーパーコンピュータで数学的モデルを評価することによって実世界に関する予測を立てるとか，科学的実験によって得られた知見をネットワークを通じて頒布する，とかいった話がある．また理論についていえば，計算機に定理の証明を手伝わせたり，仕様の一貫性を検証したり，反例をチェックしたり，テストケースを確かめたり，ということが行われている．

計算機分野は，応用数学，科学，および工学の中心的プロセスが形作る交差点に立っているのである．それら三つのプロセスは，計算機分野においてどれも同程度の，基本的な重要性をもち，計算機分野はそれら相互作用から形成される独特の配合物となっている．そしてそれらを結びつける力は，①情報交換機としての実験と設計，②上述のプロセスの各段階に対する計算機支援，および ③効率への共通の興味に存する．

プログラミングの役割

計算機分野では，プログラミング以外にも多くの活動がなされている．たとえばハードウェア設計とか，システムアーキテクチャとか，オペレーティングシステムの構成とか，データベース応用システムの設計とか，モデルの検証とかがそれである．したがって，「計算機科学イコールプログラミング」という考えは，誤解を招くものである．では計算機分野におけるプログラミングの役割は何か．またそのカリキュラムにおける役割はどうか．

明らかにプログラミングは，この分野の標準的技能の一部であり，計算機分野を主要専攻分野とする卒業生はプログラミング技能を身につけている必要がある．だが，だからといって，カリキュラムはプログラミングが基本だとか，入門科目はプログラミング科目でなければならない，ということにはならない．

どんな分野でも，その分野の固有領域に立ち入るにはある言語が使われるものであり，計算機分野の固有領域のほとんどがプログラミング記法に基づいている，ということも明らかである．プログラミング言語は，計算機分野固有の領域に立ち入るための道具として有用である．それゆえ，コアカリキュラムで身につけることが求められる能力のうちにはプログラミングが含まれているべきであり，プログラミング言語は計算機分野の固有領域に立ち入るための有用な道具という立場から扱われるべきであると，われわれは考える．

計算機分野とは何か

本報告書における計算機分野の記述は，次の4項目からなる．

（1） 要件，
（2） 手短な定義，
（3） 副領域への分割，
（4） 副領域の詳細説明．

本文の説明は，同じことを4回だんだん詳細に立ち入ってゆきながら述べる，という形をとる．

ここで述べるのは，変化しつつあるダイナミックな分野の「スナップ写真」にすぎない．われわれとしては，これを「生きた定義」であって，今後ときどき分野の成熟と変化を反映して改訂されるべきものである，と考えたい．改訂は，副領域の内部において最も頻繁に起こり，ときには副領域のリストも改訂され，手短な定義の部分は稀にしか改訂されないものと予想される．

要 件

定義の立てかたはいろいろあり得る．ここではそれが次の4項目を満たすことを必要条件とする．

1. 分野外の人々にとって，理解可能なこと．
2. 分野内の人々にとって，共通の広場となること．
3. 具体的ではっきりしていること．
4. 分野の数学的，理論的，および工学的源泉を明らかにしていること．
5. 分野の各領域について，基本的な問いと重要な成果を述べること．

計算機分野の定義を書き出すに当たって，われわれは従来出された種々の定義を参照し，われわれの定義が上記の必要条件を満たすためには，それは詳細さに関し，複数のレベルをもっている必要があるとの結論に達した．以下，他の定義について簡単に述べる．

1967年 Newell, Perlis, および Simon[6] は，計算機科学とは計算機，およびそれをめぐる諸現象を研究することであり，この定義に対して広くなされている批判は，他の科学に対してもそれが科学ではないことを証明するのに用い得るものだ，と主張した．彼らの雄弁さにもかかわらず，あまりにも多くの人々がこの定義を循環定義であって，他の分野の人々から見て軽佻浮薄である，と見なした．しかしこの定義は出発点としてはよいものといえる．というのは，ここでのちに述べる定義は，計算機をめぐる主要な現象を数え上げたもの，と見なし得るからである．

この Simon らのアイディアの，もう少しだけ手の込んだ変形が，最近 Computing Sciences Accreditation Board (CSAB, 計算機科学科認定委員会) によって採用された．それは「計算機科学とは，計算機，および計算に係わる知識の総体である．それは理論分野，実験分野，および設計分野をもち，(1)計算装置，プログラム，およびシステムを理解するための理論，(2)概念の開発および試験のための実験，(3)実際的実現のための設計方法論，アルゴリズムおよび工具，(4)これらの実現が要求に合っているか否かを検証するための解析の方法の，以上4項目を含む．」と述べている．

第3の定義は，「計算機科学とは知識表現，およびその実現の学である．」というものである．この定義の難点は，抽象化の度合いが高すぎること，および「知識表現」という言葉の意味づけに同意する人がほとんどいそうもない，ということである．同じ欠点をもつ類似の定義として，「計算機科学とは抽象化と複雑さの克服に関する学である．」というものがある．この定義は物理学にも数学にも哲学にも，そのまま当てはまってしまうであろう．

ここではあと一つ，Abelson および Sussman の次の言葉を掲げておく．『計算機革命はわれわれがものを考えるやりかたと，われわれが考えたことを表現するしかたにおける革命である．この変化の本質は，手続き的認識論とでもいうべきもの，すなわち古典的な数学の主題がそうであったようなより宣言的な見かたと対照的な，命令的な視点からの知識構造の研究の出現にある．数学は「何であるか」という観念を正確に取り扱うための枠組みである．これに対して計算は，「どのようにして」という観念を正確に扱うための枠組みを与える．』[1]

手短な定義

計算機分野とは，情報を記述し変換するアルゴリズム的プロセスに関する系統的な学であって，それらのプロセスの理論，解析，設計，効率，実現，および応用を含む．計算機分野全体の根底にある基礎的問いは「何が（効率よく）自動化できるか」である．

副領域への分割

われわれは分野を副領域にわけるという問題に，かなりの時間をかけて取り組んだ．はじめわれわれは，たとえばモデル対実現とか，アルゴリズム対機械とかいうような少数の副領域のほうが好ましいと考えた．しかしながらわれわれが考えた候補は，いずれもあまりにも抽象的で，部分間の境界が明確でなく，たいていの人にしっくりしているとは思ってもらえそうのないものであった．

次にわれわれは，この分野の基本が，各副領域においてその目標を達成するために用いられている三つの基本的プロセス，すなわち理論，抽象化，および設計に含まれている，ということに気づいた．とすれば，この分野の副領域，およびそれらと上記三基本プロセスとの関係を記述すれば有用であると考えられる．分野の一部が一つの副領域として成立するためには，次の4つの要件を満たすことが必要である．

（1） 主題の背後にある統一性，

（2） 相当量の理論的成分，

（3） 顕著な抽象化，

（4） 設計および実現における重要な問題点．

その上われわれは，各副領域は固有の文献を集積している一つの研究コミュニティー，ないしはいくつかの関係し合った研究コミュニティーの集まりと対応しているべきだと考えた．

理論という中には，その副領域に内在する数学を開発するというプロセスを含む．それらのプロセスは，他の分野の理論からの援助を受ける．たとえば，アルゴリズムとデータ構造という副領域は計算量理論を含み，グラフ理論からの援助を受ける．抽象化は，可能な実現をモデル化するという内容をもつ．それらのモデルでは，中心的な特徴を保持しながら，細部を省くということが行われる．それらは解析の対象として役立ち，モデル化されたシステムの挙動を計算によって予測する手段を提供する．設計は，問題を定式化し，問題の記述を設計仕様に変換し，設計案を作り出しては検討し，信頼性が高く，保守が可能で，文書化された，テストずみの，しかも経費上の条件を満足する設計が得られるまでこれを繰り返す，というプロセスを取り扱う．

われわれは計算機分野を覆う次の9つの副領域を取り出した．

1. アルゴリズムとデータ構造
2. プログラミング言語
3. アーキテクチャ
4. 数値的および記号的計算
5. オペレーティングシステム
6. ソフトウェア方法論およびソフトウェア工学
7. データベースシステムおよび情報検索システム
8. 人工知能およびロボティックス
9. 人間コンピュータ間のコミュニケーション

副領域の詳細

われわれは，各副領域の内容を提示するには図-1に示すような9行3列のマトリックスを考えるとよい，という結論に達した．図において各行は副領域に対応し，三つの列はそれぞれ理論，抽象化，および設計に対応している．

このマトリックスの個々の矩形には，その副領域に固有の説明を書き入れる．これらの説明においては，問題点と主要な成果を記述する．

	理論	抽象化	設計
1. アルゴリズムとデータ構造			
2. プログラミング言語			
3. アーキテクチャ			
4. 数値的および記号的計算			
5. オペレーティングシステム			
6. ソフトウェア方法論とソフトウェア工学			
7. データベースと情報検索			
8. 人工知能とロボティックス			
9. 人とコンピュータのコミュニケーション			

図-1 計算機分野の定義マトリックス

科学的文献群を保有する研究集団のうちには，ここには副領域として示してないものもある．それはそれらの研究集団が，計算機分野全体にまたがる基本的問題を扱っていることによる．たとえば並列性はすべての副領域において，並列アルゴリズムとか，並列言語とか，並列アーキテクチャとかいうような形で，しかも理論，抽象化，および設計のすべてにまたがって顔を出している．同様のことは保安性，信頼性，および性能評価についてもいえる．

計算機科学者はマトリックスの左二つの列に，計算機工学者は右二つの列におもな関心を向ける傾向がある．ここで規定する意味での計算機分野についての詳細な説明は付録に示す．

カリキュラムのモデル

計算機分野における職業能力

教育の目的は，その分野における職業能力を開発することにある．職業能力，すなわち効果的な行動の能力とは，個々人の能力をその分野の標準的な水準に照らして評価したものである．評価の基準は，その分野の過去の歴史によって定まる．職業能力を身につけるための教育のプロセスは，次の５段階からなる．(1)その領域を学びたいという動機をもつこと．(2)その領域で何ができるかを知ること．(3)その領域固有のことがらに触れること．(4)それらのことがらの理論による裏づけを身につけること．(5)それらのことがらを実際に演習すること[4]．

この考えかたは，カリキュラム設計上興味深い意味をもつ．そこから導き出される第１の疑問は，「計算機分野を主要専攻分野とする学生は，そのいかなる領域において職業能力をもつべきか？」というものである．職業能力という中には，大きく二つの領域が含まれる．

1. **分野固有の思考**—その分野において，新しいことがらを発明し，そのことがらが他の人々にも使えるようにするための新しい行動様式と道具を作り出すこと．

2. **道具の利用**—その分野の道具を，他の領域において有効な行動をするのに役立てること．

われわれは，①計算機分野を主要専攻分野とする学生のためのカリキュラムにおいては，分野固有の思考を身につけることがおもな目標であるとともに，②彼らが他の分野の人々と協力して，それらの分野における有効な行動様式の設計を助けることができる程度に，道具に精通することが必要条件である，と考えている．

職業能力とは何か，という問いを追及してゆくと，計算機分野の現存のコアカリキュラムがいくつかの領域において不適切である，ということがわかってくる．たとえば計算機分野の歴史的背景の話に重点が置かれていない例が多い．そしてその結果，多くの卒業生たちは，計算機分野の歴史に対して無知であり，ひいては分野の過去の誤りを繰り返すことを避けがたい立場に置かれている．計算機分野の卒業生の中には，事務データ処理方面で働く者がたくさんいるが，その計算機分野のたいていのカリキュラムは，その方面の職業能力を育てようとしておらず，その職業能力は計算機関係学科で育てるべきか，それともビジネス関係学科で育てるべきかという論争が，古くから跡を断たない．分野固有の思考はしっかりした数学的基礎の上に立つべきものであるが，理論は多くの計算機分野のカリキュラムにおいて，不可分の一部となっていない．計算機分野で標準的に行われていることがらのうちには，実験を計画し実施すること，チームによるプロジェクトに貢献すること，および他の専門分野と交流してそれらの分野における計算機の有効利用を助けることが含まれているが，たいていのカリキュラムは実験室における訓練，チームプロジェクト，ないし他分野との交流を無視している．

計算機分野のカリキュラムによっていかなる結果が得られるべきかについては，まだ十分な検討がなされていないので，われわれはあえてここで詳細な分析を示そうとはしないことにする．われわれはこの問題が，計算機分野の新しいコアカリキュラムの設計において最初に考慮される問題の一つとなるべきであると，強く提言したい．

生涯学習

計算機分野のカリキュラムは，卒業生が卒業後の職業生活全般にわたって学習を続けてゆくことの重要性を十分認識するように設計されるべきである．とかく学科目は，すべての学習に内在する探求のプロセスに焦点を当てることをせず，講義形式によって「解答」を提示する，という方式で設計されていることが多い．われわれは後継の委員会において，探求のプロセスを内包する教育方式，計算機分野の文献利用に対するオリテンテーション，および生涯にわたる学習プロセスへの決意の育成について考慮されるよう提言する．

入門科目系列

このカリキュラムモデルでは，分野の内容の教示および学習に先立って，分野への動機づけと，そこで何かできるか知ることとが必要である．入門科目系列は，まさにこのことを目的としている．計算機分野の主要な各領域（それが，学生たちが職業能力を身につけることを必要とする場であるが）を学生たちに，十分な深さと厳密さをもって提示し，彼らにそれらの領域がもつ力と，それらにおける職業能力を身につけることによって得られる利益をわからせる必要がある．このカリキュラムのそれよりあとの部分は，それらの領域を組織的に探求し，新しい概念とことがらを開示し，かつそれらについて学生たちに演習をさせるように，注意深く設計されなければならない．

そこでわれわれは入門科目を，通常の講義，およびこれと密接に連係した毎週の実験から構成するよう提案する．講義では基礎を強調し，実験では技法とノーハウを強調する．

この，講義は永続性のある原理と概念を強調し，実験科目は変化しつつある内容と現行の技法にかかわる技能を強調するというのは，物理的科学および工学における伝統的モデルである．たとえば講義ではアルゴリズムの設計および解析，ネットワークプロトコルの機能階層への分割などを論ずる．対応する実験科目では，講義で解析されたアルゴリズムを実現するプログラムを書いてその性能を測定したり，ネットワークインタフェースを実現，試験してそのパケット転送スループットを測定したりする，というぐあいである．

本文の案では，すでに計算機科学の第1科目において，プログラミング，アルゴリズム，およびデータ構造に対する入門にとどまらず，他のすべての副領域からの題材をも取り扱う．数学をはじめとする基礎理論は，講義の適当な場所に組み入れることになろう．

入門科目には，全計算機分野にわたる厳密で意欲を掻き立てるような概説を含めることとしたい．たとえば物理学において，ファインマン物理学講義がしているようなことを，ここでは入門科目の基本方針として頭に置いている．

単にこの提案に従って入門科目の教科内容を設計しなおし，学部レベルのカリキュラム全体については従前のままとする，というのは深刻な誤りである，ということを強調しておきたい．この点物理学科における過去の経験は，計算機関係学科に対する多くの教訓を提供している．

科目履修条件

われわれは計算機分野専攻学生が，なんらかの言語によるプログラミングについて若干の学習歴をもち，またワープロ，表計算，およびデータベースのような，計算機に基づく工具に対する若干の経験をもっているものと仮定する．高校および家庭における計算機の普及を考慮すれば，大学入学者の大部分がこのような背景をもっていると仮定し，それをもたない学生には「補習」科目を提供するという方針でゆけるのではないか，と一応思われよう．しかしながらわれわれの知ったところでは，高校におけるプログラミング学習が適切になされているかどうかには多くの議論の余地があり，適切な準備がなされていることはむしろ稀であると考えられる証拠がある．そこでわれわれは，計算機関係学科において，本文の入門部分の前，またはこれと同時に履修することを条件とする，プログラミングおよび計算機用工具類についての予備的入門科目を開設するよう提案する．また併せてわれわれは，その際飛び級に関する規則を用意し，高校において適切な準備をととのえてきた学生については，この予備的入門科目を省くことができるようにするよう提案する．

数学関係の前提科目および同時履修科目に関する公式的な規定を述べることはより困難であって，大学固有の事情に依存する．しかしながら計算機関係の多くの学科認定委員会は，離散数学，微積分学，および確率統計を含むかなりの数学関係科目を要件としてあげている．またこれらの要件を超過するカリキュラムをもった大学も見られる．ここに示す計算機関係入門カリキュラムでは，9つの副領域のおのおのにおいて，いかなる数学が関係をもっているかを，ある程度くわしく書き出した．また可能な場合には，各教授単位においてどのような数学的予備知識が必要であるかを明示した．これをもとに個々の学科において，本文の科目の各モジュールとその学科の数学関係科目履修規定を適宜同期させることが可能であると信ずる．場合によっては，計算機分野の特定の話題において必要とされる数学的背景を，必要になったところで説明する，というやりかたも考えられる．一般にわれわれは，学生たちが計算機分野の学習を進めてゆく過程において，関連する数学の応用になるべく早い時期に触れるよう推奨する．

モジュール構成

この入門科目系列は，計算機分野の背後に横たわる一貫性を明るみに出し，話題から話題へと教育的に見て自然な流れをもつものでなければならない．したがってこの科目を9つの副領域に対応する9つの節を並べたものとして構成するのは不適切である．それでは結構なものの単なる寄せ集めという印象を与え，また節間の移行に困難が感じられることになろう．話題の，上に記した条件を満たす配列方法の一つとしては，次のようなものが考えられる．

アルゴリズム概念の基礎
計算機構成（「フォン・ノイマン」）
数学的プログラミング
データ構造とデータ抽象
計算可能性の限界
オペレーティングシステムと保安性
分散処理とネットワーク
人工知能のモデル
ファイルとデータベースシステム
並列計算
ヒューマンインタフェース

われわれはこれらの話題を 11 のモジュールに分割した．各モジュールは，各側面ないし話題の表面的な概説に陥ることなしに，主題を代表する，学び甲斐のある題材を提示するように考えられている．各モジュールは，必要に応じて定義マトリックス（図-1）の複数の箱から題材を得ている．その結果多くのモジュールは，必ずしも定義マトリックスの各行と1対1の対応をもっていない．たとえばわれわれの科目例では，最初のモジュールは「アルゴリズムの基礎概念」と名づけられている．それは定式化と理論の役割，プログラミングの方法，プログラミングの概念，効率，およびいくつかの特定のアルゴリズムについて述べており，定義マトリックスの第1，2，4，6行から情報を得ている．ただしここでは逐次的アルゴリズムのみを扱う．後出のモジュール「分散処理とネットワーク」および「並列計算」では，第1モジュールの内容を拡充し，定義マトリックスの第3，5行から新しい題材を取り入れている．

一般的方針として各モジュールでは，理論の必須部分および抽象化の大部分を，講義の中で取り扱うこととしている．一般に理論は，必要が生じるまでは導入しない．各モジュールは，実験セッションと密接に関連づけられている．実験課題をどのようなものにするかは，各モジュールの説明の中で規定しておいた．本文の入門部分は1年半3学期構成となっており，各学期は 42 回の講義と 35 回の定時実験セッションを含む．入門科目系列の詳細はここでは割愛し，縮約前の報告書に含めてある．

この科目案は，分野の定義をカリキュラムの入門部分に写像するやりかたの一つの例にすぎない，ということを再度強調しておきたい．これ以外のやりかたを，いくつかの大学の既存の入門カリキュラムの中に見ることができる．

実験科目

ここまでで述べてきたカリキュラムは，原理を技法と分離し，しかもその間の関連性を保つようなものであった．またわれわれは講義では原理を扱い，実験科目では技法を扱い，しかも両者の間に密接な関連を保つよう提案した．

実験科目には次の三つの目的がある．

1. 実験科目は，講義で示された原理が実際のソフトウェアおよびハードウェアの設計，実現，ならびに試験にいかに適用されるかを示すべきである．それはまた学生たちに，抽象的な概念を理解する助けとなる具体的な経験を与えるべきである．実験は，計算機技術の実際に対する学生たちの直感を鋭くし，正しくかつ効率的な計算機プログラムないしシステムを作り上げるに要する知的努力の重要性を示すために，必要不可欠である．

2. 実験科目は，計算機分野における良質のノーハウに到達するための，プロセスを強調すべきである．それはまた，プログラムよりはむしろプログラミングを，実験の技法を，ハードウェア能力への理解を，ソフトウェア工具の正しい利用を，解説文書の正しい利用を，そして実験およびプロジェクトの適切な文書化を強調すべきである．副システム上で行われる実験の構成，制御，監視を助けるため，ホスト計算機上に多くのソフトウェア工具を用意することが必要となろう．実験科目では，それらの工具の適切な利用法を教えるべきである．

3. 実験科目では，実験方法を教える必要がある．ここで実験方法という中には，実験計画法，ソフトウェアモニタおよびハードウェアモニタ，結果の統計的解析，ならびに知見の適切な提示方法の利用を含む．学生は，注意深い実験と気楽な観察の間の区別を学ぶ必要がある．

これらの目標を達成するためには，実験科目を注意深く計画し，指導することが必要である．学生は実験科目について，指定された時間に出席すべきである．週の時間数は名目3時間とする．実験はあらかじめ計画されていることを要し，学生には各実験の目的および方法を記した文書を配布することが必要である．その記述の深度は，学生の過去の実験経験に即応しているべきである．初期には細部の記述がより多く必要である．実験はインストラクタの指導のもとに行われることが必要である．インストラクタは全学生が正しい方法に従うように取り計う．

入門科目に付随する実験科目は，より綿密な監督を要し，また十分よく計画された活動でなければならない．したがってこれらの実験科目では，高学年の同種の科目と比べて，学生一人当たりのスタッフが，より多数必要である．

実験科目の課題は，科目の講義部分で説明される題材と密接な関係をもっている必要がある．一般に実験課題は，ハードウェアとソフトウェアの組み合わせを取り扱うものとなろう．課題の一部には，ソフトウェア開発プロセスを助ける技法および道具を主題とするものが含まれよう．また他の一部は，既存のソフトウェアの解析，測定，および既知のアルゴリズムとの比較を主題とするものとなろう．さらに別の一部は，講義で学んだ原理に基づくソフトウェア開発を主題とするものとなろう．

実験科目の課題は，平均的学生が作業を割り当て時間内に完了できる，という意味で完結しているべきである．実験課題は，学生がものごとを自分で発見し，学習することをうながす必要がある．学生には，実験，観察，およびデータを記録するための適切なノートを保持するよう要求すべきである．また学生たちには，彼らのソフトウェアを保守し，以後の実験プロジェクトで使用可能なライブラリを作り上げるように要求すべきである．

学生たちには講義，実験の双方において，監督下にある実験室以外の場所で計算機を使用することを必要とする宿題が課せられるものと考える．いい換えれば，組織的な実験科目は通常のプログラミング方式および記述方式の宿題を補うものであって，それに代わるものではない．

プログラム開発を主題とする課題のかなりの部分において，課題はインストラクタから与えられた既存のプログラムを変更することを内容としているべきである．こうすることにより，学生はよく書けたプログラムを読むことを強制されることになり，またソフトウェアの統合についての経験を得ることができ，より大きくて満足感を与えることの多いプログラムを作ることが可能になる．

計算機技術は絶えず変化しつつある．したがって実験室に設けるハードウェアシステム，ソフトウェアシステム，各種機器，および工具類の詳細な仕様を定めることは困難である．実験室における機器の選択および人員配置は，次の原則に従っているべきである．

1. 実験室には最新のシステムおよび言語を備えることが必要である．プログラミング言語は，計算機分野に対する学生のものの見かたに，大きな影響を与える．実験室には，学生によい習慣をつけるようなシステムを配置すべきである．コア科目で時代遅れのシステム（ハードウェア，ソフトウェアの両面とも）を使わないようにすることは，特に重要である．

2. ハードウェアおよびソフトウェアは完全に保守されていることが必要である．誤動作をともなう機器は学生に欲求不満を起こさせ，学習に対してマイナスになる．実験室のハードウェアおよびソフトウェアを保守するため，適正なスタッフを配置する必要がある．事情は，他の学問分野における実験室と同じである．

3. 機能が省かれていないことが重要である．（こういう中には，共同利用システムにおいて，応答時間が適正であることを含む．）学生を言語，またはシステムの小さい部分集合に接触させることは，初体験の段階では有用であることもあるが，制限は学習の進行につれて取り除いてゆく必要がある．

4. よいプログラミングの道具が必要である．コンパイラには多くの注意が払われているが，それ以外のプログラミングの道具も同程度に頻繁に利用されるものである．たとえば UNIX システムの場合，学生たちに emacs のようなエディタを使用させ，シェル，grep, awk, および make のような工具の使いかたを学ばせる必要がある．その種の工具が実験室で使えるようにするに十分なだけの，記憶容量および処理速度が用意されていることが必要である．

5. ハードウェアおよび測定器の利用に関して，適正な用意が必要である．課題によっては，ハードウェアユニットを相互接続し，信号の測定を行い，データ経路を監視するなどのことが必要になる可能性がある．雑部品，コネクタ，ネーブル，測定機器，および

テスト機器が利用できなければならない.
IEEE Computer Society の Task Force on Goal Oriented Laboratory Development では，この主題が深く検討された．彼らの報告書には，カリキュラムの各レベルにおいて必要とされる資源（すなわち人員および機器）に関する討論が含まれている．

学科認定

本文の検討は，例示の科目が Computing Sciences Accreditation Board (CSAB) の現行のガイドラインと整合することを意図してなされた．その内容とCSAB のガイドラインの詳細な対応づけは，この委員会の職務範囲を超える．

むすび

この報告書は，学問としての計算機分野がいかなる内容をもつかを，その基本的な概念，原理，およびこの分野に固有のことがらを強調するような形で提示することによって，この分野に関する新しい思考を呼び起こそう，という狙いをもって書かれた．また本報告書は，他の学問分野で用いられている教育的モデル，すなわち有用なことがらの存在を例示したあとで実習によって職業能力を開発する，というやりかたに従うコアカリキュラム改革の一方向を示唆した．さらにその方向を例示するものとして，概念と原理を講義に，技法をそれと密接に関係づけられた実験科目に置いた，一つの厳密な入門科目を示した．

この入門科目系列は，各学科に従来あった入門科目系列を単にこれと入れ替えれば足りる，というようなものではない．新しい入門部分が全体の中で一貫性を保った一部となるように，カリキュラム全体を設計しなおす必要がある．それゆえわれわれは ACM において，コアカリキュラム再設計の残部を受け持つ後継の委員会が結成されるように提言したい．

新しいカリキュラムモデルがこの分野の一部となる前には，多くの実際問題を解決する必要がある．たとえば，

1. 各大学においては，カリキュラムを新しい概念構成に基づいて再設計する必要があろう．
2. 本報告書で提案した枠組みに基づく教科書，教材は，現在のところ存在していない．
3. たいていの学科が，本文で示唆した教育活動に適合する実験室，計算機設備，および教材を持ち合わせていない．
4. 教育助手および教授がこの新しい見かたを知らない．
5. 高校において，計算機分野におけるよい準備教育がなされている例は稀である．

われわれは，本文の提言の多くが実施に移す上で困難をともない，かつ多量の作業を要するものであることを認識している．われわれはここでなされた提言によって達成される計算機分野の教育改善が，その努力に値するものであると確信しており，その達成に向けて読者がわれわれと手を結ばれるよう希望するものである．

付 録
学問としての計算機分野
—その定義—

計算機科学ないし工学は，情報を記述し変換するアルゴリズム的プロセス（その理論，解析，設計，効率，実現，および応用を含む）の，系統的な研究を内容としている．計算機分野全体の背後に横たわる基本的な問いは，（効率よく）自動化できるものとはどんなものか[2),3)]というものである．この分野は1940年代初頭に，アルゴリズム理論，数理論理，およびプログラム内蔵方式計算機の発明の3者が結びついて生まれたものである．

計算機分野は，数学および工学に深く根を張っている．数学は解析を，工学は設計を，そこにもたらした．計算機分野はそれ自身の理論と実験方法と工学をもっている．この点計算機分野は，たいていの物理的科学において，科学とその発見を応用する工学が（たとえば化学と化学工学，というように）別物とされているのと対照を示している．この分野において科学と工学が不可分なのは，この分野の科学的および工学的パラダイムが，根本的な相互関係をもっていることによる．

数千年間にわたって計算は，数学の一つの主要な課題であり続けてきた．物理現象のモデルから方程式が導き出され，それを解くことによって現象の予測がなされる，という例が多かった．たとえば軌道計算，天気予報，流体運動などにその例がある．またそれらの方程式を解くために，多くの一般的方法が考案された．たとえば連立1次方程式，微分方程式，および積分方程式を解くためのアルゴリズムはその例である．
一方ほとんど同じ時期に，機械系の設計のための計算手法が，工学における一つの主要な課題となった．その例としては，静止物体内の応力を計算したり，運動

物体の運動量を計算したり，われわれの感覚器官で直接観測するには大きすぎ，または小さすぎる距離を測ったりするアルゴリズムがあげられる．

工学と数学の長期にわたる相互作用の結果として生まれたものの一つに，計算のための機械装置がある．測量家や航海家が使う器具類は，1000年もの歴史をもっている．パスカルおよびライプニッツは，1600年代のなかばに算術用計算器を作った．1830年代にはバベッシが，機械的に，かつ誤りなく対数，三角関数などのような算術的関数を計算する「解析エンジン」なるものを考え出した．彼のエンジンはついに完成しなかったが，後世に向けてインスピレーションのもととなった．1920年代にはブッシュが，一般の連立微分方程式を解くための電子的アナログ計算機を作った．同じころ，加減乗除および平方根の計算ができる電子機械的計算機械も出現した．電子的フリップフロップは，これらの機械から，それらと同様の，ただし可動部分を含まない電子的な機械への，自然な足掛かりとなった．

さて論理学は，数学の一部であって，推論の正しさに関する判定条件と，論証の形式的原理を主題としている．ユークリッドの時代以来それは，厳密な数学的および科学的議論の道具として使われてきた．19世紀には，既知の推論システムに見られる不完全性が存在しないような，完全な論理体系への探求がはじまった．そのような完全な体系があれば，与えられた主張が真であるか偽であるかは，機械によって決定できることになる．しかるに1931年ゲーデルは，彼の「不完全性定理」においてそのようなシステムは存在しないということを示した．また1930年代の終わり近くチューリングは，他のいかなる計算機械の一歩一歩の動きでも模倣することができる万能計算機，という考えを出した．彼の結論は，ゲーデルのそれと同様であった．すなわち，はっきり定義できる問題のうちに，機械的手順では解けないものがあるというのであった．論理学は，自動的計算の限界に対する深い洞察において重要であるのみならず，記号の列（たとえば数として表現された）はデータとしてもプログラムとしても解釈できる，という洞察においても重要である．

この洞察が，プログラム内蔵方式の計算機と単なる計算器を区別する中心的アイディアとなった．すなわち前者においては，アルゴリズムの各手順が機械内の表現形式で符号化され，記憶装置に格納され，あとで処理装置によって解読，実行されるのである．機械内の符号は，高水準の記号的形式，すなわちプログラミング言語から機械的に導出できる．

このように，計算機分野という学問を生み出したのは，計算という古代以来の糸と，電子工学および情報の電子的表示という現代の糸の複雑なからみ合いであった．

われわれは計算機分野に，次の9つの副領域を見出した．

1. アルゴリズムとデータ構造
2. プログラミング言語
3. アーキテクチャ
4. 数値的および記号的計算
5. オペレーティングシステム
6. ソフトウェア方法論およびソフトウェア工学
7. データベースシステムおよび情報検索システム
8. 人工知能およびロボティックス
9. 人間コンピュータ間のコミュニケーション

これらはいずれも背後に主題の統一性を背負い，相当量の理論的成分と，顕著な抽象化と，設計および実現に関する重要な問題点を含んでいる．理論は各副領域に内在する数学の開発を取り扱い，グラフ理論，組み合わせ理論，形式言語理論のような他の理論からの支援を受ける．抽象化（ないしモデリング）は，可能な実現をモデル化するという内容をもつ．それらのモデルでは，中心的な特徴を保持しながら細部を省く，ということが行われ，システムの将来の挙動を予測する手段を提供する．設計は問題を特定し，問題の要件と仕様を導き出し，プロトタイプによるテストを繰り返し，システムを作り上げる．設計という中には実験的方法が含まれる．計算機分野では，それはプログラムおよびシステムを計測し，仮説を検証し，抽象化を実現形に拡張するためにプロトタイピングを行うというような，いくつかのスタイルをもっている．

ソフトウェア方法論は本質的には設計に関係するものであるが，それはまた理論および抽象化にもかなりの成分をもっている．われわれがそれを一つの副領域と認めたのは，そのためである．一方並列計算および分散計算は，すべての副領域とそのすべての要素（理論，抽象化，および設計）にまたがる問題点である．それゆえわれわれは，それらを副領域とも，また副領域の要素とも考えなかった．

以下の各節は，各副領域の各要素（理論，抽象化，および設計）を詳細に示すものである．理論と抽象化および抽象化と設計の間の境界線は，不可避的にぼや

けたものとなっている．事項によっては，それをどちらに置くかは個人的趣味の問題であるようなものもある．

われわれの意図は，計算機分野のおもな特徴を示すことによって，道案内をつとめることであって，詳細な地図を示すことではない．計算機分野へのこの道案内は，科目またはカリキュラムの案でないことを銘記する必要がある．それはカリキュラムを設計する上での枠組みであるにすぎない．さらにこの道案内は，絶えず変化しつつある有機体の「スナップ写真」だ，ということも銘記の必要がある．それは定期的に再評価と改訂を受ける必要がある．

1. アルゴリズムとデータ構造

この領域は，与えられた問題群とその効率のよい解法を取り扱うものである．基本的な問いには次のようなものが含まれる．①与えられた問題群に対して，それを解く最良のアルゴリズムは何か．②それらはどれだけの記憶容量および時間を必要とするか．③記憶容量と所要時間の間のトレードオフはどうか．④データを参照するための最善の方法は何か．⑤最良のアルゴリズムにおける最悪の場合はどのようなものか．⑥アルゴリズムの平均的挙動はどうか．⑦アルゴリズムはどの程度の一般性をもつか（どの程度広い問題群が，同様の方法で取り扱えるか）．

1.1 理 論

アルゴリズムとデータ構造の領域における理論の，おもな要素は次のとおりである．

1. 計算可能性の理論．それは計算機にできること，できないことの定義を与える．

2. 計算量理論（計算の複雑さの理論）．それは計算可能関数の時間および空間所要量を測る方法を明らかにし，問題の大きさとその問題を解くアルゴリズムの（最高および最低の）性能との関係を示し，与えられた問題のあらゆる可能な解がそれを下回ることのできないような下限が，いかなるものであるかを証明するための方法を与える．

3. アルゴリズムおよびアルゴリズム群の時間計算量および空間計算量．

4. 扱いにくさのレベル．たとえば多項式時間で決定論的に解ける問題の類（P問題），多項式時間で非決定論的に解ける問題の類（NP 問題），および並列計算機によって効率よく解ける問題の類（NC 問題）．

5. 並列計算，各種の下界，アルゴリズムに課せられるデータフロー的条件の，計算機の情報転送経路への写像．

6. 確率アルゴリズム，すなわち必ず結果が得られることが保証されている決定的アルゴリズムと比べてずっと効率よく，十分高い確率で正しい答えを与えるアルゴリズム．モンテカルロ法．

7. 暗号．

8. 支援領域としてのグラフ理論，再帰的関数，再帰的関係，組み合わせ理論，解析学，数学的帰納法，命題論理および時制論理，意味論，確率，および統計．

1.2 抽 象 化

アルゴリズムとデータ構造の領域における抽象化の，おもな要素は次のとおりである．

1. 重要な問題群における効率のよい，ないし最適なアルゴリズム，およびその最高性能，最低性能，平均性能の解析．

2. 種々の問題群における，制御構造およびデータ構造が所要時間および所要記憶容量に及ぼす効果の分類．

3. 重要な技法群．たとえば分割統治方式，グリーディーアルゴリズム，動的プログラミング，有限状態機械のインタプリタ，スタックマシーンのインタプリタなど．

4. 並列および分散アルゴリズム．問題を，別々のプロセッサで実行可能なタスクに分割するための方法．

1.3 設 計

アルゴリズムとデータ構造の領域における設計と実験の，おもな要素は次のとおりである．

1. 探索，整列，乱数の生成，およびテキスト上のパターン照合のような重要な問題群における，アルゴリズムの選択，実現，および試験．

2. 多くの問題群にまたがって適用可能な一般的方法（ハッシュ法，グラフ，木など）の実現と試験．

3. ネットワークのプロトコル，分散データの更新，セマフォ，デッドロックの検出，および同期方式のような，分散アルゴリズムの実現と試験．

4. ゴミ集め，バディーシステム，リスト，表，およびページングのような，記憶管理方式の実現と試験．

5. 組み合わせ的な問題に関する発見的アルゴリズムの，広範な実験的試験．

6. 確実な認証と機密裡の通信を可能にする，暗号プロトコル．

2. プログラミング言語

この領域は，アルゴリズムを実行する仮想機械のための記法，アルゴリズムおよびデータのための記法，および高水準言語から機械語への効率のよい翻訳方法を取り扱うものである．基本的な問いには次のようなものが含まれる．①言語によって規定される仮想機械の構成（データ型，基本操作，制御構造，新しい型，および基本操作の導入方法など）としては，どのようなものがあり得るか，②それらの抽象概念は計算機上ではどのように実現されるか，③計算機がする仕事を有効に，かつ効率よく指定するのに利用できる記法（構文）は，どのようなものであるか．

2.1 理　論

プログラミング言語の領域における理論の，おもな要素は次のとおりである．

1. 形式言語とオートマトン．構文解析および計算機言語の翻訳に関する理論を含む．

2. チューリング機械（手続き的言語の基礎としての），ポストシステム（文字列処理言語の基礎としての），λ記法（関数型言語の基礎としての）．

3. 形式的意味論，すなわち①計算機の数学的モデルおよびモデルの相互関係，②言語の構文，および③その実現を定義するための方法．主要な方法としては，表示的意味論，代数的意味論，操作的意味論，および公理的意味論がある．

4. 支援領域としての述語論理，時制論理，現代代数学，および数学的帰納法．

2.2 抽 象 化

プログラミング言語の領域における抽象化の，おもな要素は次のとおりである．

1. 言語の，構文的および動的意味に基づく分類．たとえば静的型づけ言語，動的型づけ言語，関数型言語，手続き型言語，オブジェクト指向言語，論理型言語，仕様言語，メッセージ交換言語，およびデータフロー言語．

2. 言語の，狙いとする応用領域に基づく分類．たとえば事務データ処理言語，シミュレーション言語，リスト処理言語，およびグラフィクス言語．

3. プログラム構造に関する主要な構文的および意味的モデルの分類．たとえば手続きの階層，関数の合成，抽象データ型，および相互に通信し合う並列プロセス．

4. 各種の言語の抽象的実現モデル．

5. 構文解析，コンパイラ，インタプリタ，およびコード生成の方法．

6. パーザ，スキャナ，コンパイラの構成部品，およびコンパイラのための自動生成手法．

2.3 設　計

プログラミング言語の領域における設計および実験の，おもな要素は次のとおりである．

1. 特定の抽象機械（意味論）と構文を取り集めて，ひとまとまりの実現可能な形にまとめた個々の言語．たとえば手続き的言語 (COBOL, FORTRAN, ALGOL, Pascal, Ada, C)，関数型言語 (LISP)，データフロー言語 (SISAL, VAL)，オブジェクト指向言語 (Smalltalk, CLU)，論理型言語 (Prolog)，文字列処理言語 (SNOBOL)，および並列処理言語 (CSP, Occam, Concurrent Pascal, Modula 2)．

2. 各種言語のための個別の実現方法．たとえば実行時モデル，静的および動的実行の方法，型検査，記憶装置およびレジスタの割りつけ，コンパイラ，クロスコンパイラおよびインタプリタ，プログラムの中から並列性を見つけ出すためのシステム．

3. プログラミング環境．

4. パーザおよびスキャナのジェネレータ（たとえば YACC, LEX），コンパイラ・ジェネレータ．

5. 構文的・意味的な誤り検査，輪郭（プロファイル）作成，虫取り，およびトレースのためのプログラム．

6. プログラミング言語に関する手法の，文書処理領域への応用（たとえば表，グラフ，化学式，表計算用の計算式などの生成，入出力，およびデータ処理）．その他の領域（統計処理など）への応用．

3. アーキテクチャ

この領域は，ハードウェア（およびそれに付随するソフトウェア）を，効率がよくて信頼性の高いシステムとして組織化するための方法を取り扱う．基本的な問いには次のようなものが含まれる．①計算機の内部のプロセッサ，記憶装置，および通信機構を実現するためのよい方法とは，どのようなものか．②大規模な計算システムを設計，制御し，それが誤りや誤動作があっても意図どおり働くことを説得力をもって実証するにはどうしたらよいか．③どのようなアーキテクチャによれば，多数の処理機構が一つの計算の中で同時に動作するような，効率のよい仕掛けを作ることができるか．④効率を測定するにはどのようにしたらよいか．

3.1 理 論

アーキテクチャの領域における理論の，おもな要素は次のとおりである．

1. ブール代数．
2. スイッチング理論．
3. 符号化理論．
4. 有限状態機械の理論．
5. 支援領域としての統計，確率，待ち行列，信頼性理論，離散数学，整数論，および各種の数体系における算術．

3.2 抽 象 化

アーキテクチャの領域における抽象化の，おもな要素は次のとおりである．

1. 回路の，その機能を挙動に結びつける，有限状態機械モデルおよびブール代数的モデル．
2. システムを基本素子に基づいて合成するための，その他の一般的方法．
3. 有限体上で算術関数を計算するための，回路のモデルおよび有限状態機械．
4. データ転送路およびデータ転送制御機構のモデル．
5. モデルおよび作業内容に応じて，命令体系を最適化すること．
6. ハードウェアの信頼性に関する諸問題．すなわち冗長性，誤り検出，回復，および試験．
7. VLSI 装置の設計における，面積，演算時間，および組織性に関するトレードオフ．
8. 個々の計算モデル（逐次計算，データフロー，リスト処理，配列処理，ベクトル処理，メッセージ交換など）に適する計算機構成．
9. 設計の諸段階（構成段階，プログラム段階，命令段階，レジスタ段階，ゲート段階）の区別．

3.3 設 計

アーキテクチャの領域における設計および実験の，おもな要素は次のとおりである．

1. 高速計算のためのハードウェアユニット．たとえば算術演算ユニット，キャッシュなど．
2. いわゆるフォン・ノイマン計算機（単一の命令系列をもつプログラム内蔵計算機）．その RISC 方式および CISC 方式による実現．
3. 情報の蓄積，記録，および誤りの検出・訂正のための効率のよい方法．
4. 誤りに対応するための個別的方法．すなわち回復，診断，再配置，およびバックアップの手順．
5. VLSI 設計のための CAD システムおよび論理シミュレータ．配置および故障診断のための生産用プログラム．シリコンコンパイラ．
6. 個々の計算モデルによる計算機（たとえばデータフロー，木，LISP，ハイパーキューブ，ベクトル，およびマルチプロセッサ）の実現方法．
7. スーパーコンピュータ，たとえば Cray や Cyber．

4. 数値的および記号的計算

この領域は，システムの数学的モデルから生じた方程式を効率よく，かつ正確に解くための一般的方法を取り扱うものである．基本的な問いには次のようなものが含まれる．①連続的または無限のプロセスを，有限の，離散的なプロセスで正確に近似するにはどうしたらよいか．②それらの近似から生じる誤差には，どのように対処したらよいか．③与えられた方程式群を，指定された精度で解くにはどれだけの時間がかかるか．④積分，微分および最小項への分割のような，方程式の記号処理のための方法はどうか．⑤以上の問いに対する答えを，効率がよく，信頼性が高い，高品質数学ソフトウェアパッケージとしてまとめるにはどうしたらよいか．

4.1 理 論

数値的および記号的計算の領域における理論の，おもな要素は次のとおりである．

1. 整数論．
2. 線形代数．
3. 数値解析．
4. 非線形力学．
5. 支援領域としての微積分，実解析，複素解析，および代数．

4.2 抽 象 化

数値的および記号的計算の領域における抽象化の，おもな要素は次のとおりである．

1. 物理的問題の，連続的（およびときには離散的）な数学におけるモデルとしての定式化．
2. 連続的な問題の離散的な近似．この文脈におけるうしろ向き誤差解析，誤差の伝播，および線形・非線形方程式の解の安定性．特殊な問題についての特殊な方法，たとえば高速フーリエ変換およびポアッソン解法．
3. 規則的なメッシュと境界値によって規定できる，広範な問題領域にわたる有限要素モデル．付随する操

り返し的解法および収束理論（具体的には，直接法，間接法，多重格子法，収束の速度）．並列的解法．数値的積分における刻みの自動調節．

4. 記号的積分および微分．

4.3 設　計

数値的および記号的計算の領域における設計および実験の，おもな要素は次のとおりである．

1. CHEM や WEB のような，高水準問題定式化システム．

2. 線形代数，常微分方程式，統計，非線形方程式，および最適化のための，個々のライブラリおよびパッケージ，たとえば LINPACK, EISPACK, ELLPACK.

3. 有限要素法のアルゴリズムを特定のアーキテクチャに当てはめる方法．たとえばハイパーキューブにおける多重格子法．

4. たとえば微分，積分，および式の因数分解のような，強力で自明でない処理をする能力をもった，MACSYMA や REDUCE のような記号処理システム．

5. オペレーティングシステム

この領域は，多様な資源をプログラムの実行のために，効率よく配分することを可能にする制御機構を取り扱うものである．基本的な問いには次のようなものが含まれる．①計算機システムの動作の各レベルにおいて，目に見える対象物および許される操作はどのようなものであるか．②各種の資源（あるレベルにおいて目に見える対象物）に対し，その効果的な利用を可能にするための最小限の基本操作群はどのようなものであるか．③利用者が資源の抽象的表現を，ハードウェアの物理的詳細に立ち入らずに取り扱うことができるようにするためには，インタフェースはどのように構成すべきか．④ジョブのスケジュリング，記憶管理，ソフトウェア資源へのアクセス，並行タスク間の通信，信頼性，および保安性のための，効果的な制御戦略はどのようなものであるか．⑤システムの機能を少数の構成規則を繰り返し適用することによって拡張できるようにするための基本的原理はどんなものか．⑥分散的計算を組織して，多数の自律的な通信ネットワークによって相互接続された計算機が計算に，ネットワークのプロトコルやホストの設置場所やバンド幅や資源の命名方法がほとんど目に見えないような形で参加できるようにするための方法はどんなものか．

5.1 理　論

オペレーティングシステムの領域における理論の，おもな要素は次のとおりである．

1. 並列性の理論．具体的には同期，決定性，およびデッドロック．

2. スケジュリングの理論，特にプロセッサのスケジュリング．

3. プログラムの挙動および記憶管理の理論．最適記憶割り当て手法を含む．

4. 性能のモデル化と解析．

5. 支援領域としての詰め合わせ理論(bin packing)，確率，待ち行列，待ち行列ネットワーク，通信・情報理論，時制論理，および暗号理論．

5.2 抽 象 化

オペレーティングシステムの領域における抽象化の，おもな要素は次のとおりである．

1. 利用者が，物理的な詳細に気を遣うことなく，理想化された資源を操作することができるようにするための抽象化の原理（たとえばプロセッサでなくプロセスを，主記憶と2次記憶からなる階層構造でなく仮想記憶を，ディスクでなくファイルを扱えるようにする）．

2. 利用者インタフェースで知覚される対象物と内部の計算構造の対応づけ．

3. 主要な部分問題（たとえばプロセス管理，記憶管理，ジョブのスケジュリング，2次記憶の管理，性能解析）に関するモデル．

4. 分散計算に関するモデル．たとえばクライアント・サーバモデル，共同動作する逐次プロセス，遠隔手続き呼び出し．

5. 計算の保安性に関するモデル．たとえばアクセス制御，認証，コミュニケーション．

6. ネットワーク技術．階層プロトコル，名前づけ，遠隔資源の利用，ヘルプ・サービス，トークン交換や共有バスのような局地ネットワーク用プロトコル．

5.3 設　計

オペレーティングシステムの領域における設計および実験の，おもな要素は次のとおりである．

1. タイムシェアリングシステム，自動記憶割り当てシステム，多層スケジュラ，記憶管理システム，階層型ファイルシステムなどのような，商用のシステムの構成要素としての役を果たしているシステム部品のプロトタイプ．

2. UNIX, Multics, Mach, VMS, および MS-DOS のようなオペレーティングシステムの構成技法.

3. ユーティリティ（たとえばエディタ，文書フォーマッタ，コンパイラ，リンカ，およびデバイスドライバのような）のライブラリを構成するための技法.

4. ファイルおよびファイルシステム.

5. 現実のシステムの性能を評価するための，待ち行列ネットワークのモデリングおよびシミュレーションのためのパッケージ.

6. イーサーネット，FDDI，トークンリングネット，SNA，および DECNET のような，ネットワークアーキテクチャ.

7. 米国防省プロトコル・スイートで実現されているプロトコル技法 (TCP/IP)，仮想回路プロトコル，インターネット，実時間コンファレンス，および X. 25.

6. ソフトウェア方法論およびソフトウェア工学

この領域は，与えられた仕様を満足し，安全性，保安性，信頼性，および信用性を備えた，プログラムおよび大規模ソフトウェアシステムの設計を取り扱うものである．基本的な問いには次のようなものが含まれる．①プログラムおよびプログラミングシステムの開発の背後に横たわる原理は何か．②プログラムまたはシステムが仕様を満足することを証明するにはどうしたらよいか．③重要な場合を洩れなく尽くしており，かつ安全性に関する解析を行うことが可能であるような仕様を作り出すにはどうしたらよいか．④ソフトウェアシステムは世代の変遷に応じてどのように進化するか．⑤ソフトウェアを，理解と変更が容易であるように設計するにはどのようにしたらよいか.

6.1 理 論

ソフトウェア方法論およびソフトウェア工具の領域における理論の，おもな要素は次のとおりである.

1. プログラムの検証と証明.

2. 時制論理.

3. 信頼性理論.

4. 支援分野としての述語論理，公理的意味論，および認知心理学.

6.2 抽 象 化

ソフトウェア方法論およびソフトウェア工具の領域における抽象化の，おもな要素は次のとおりである.

1. 述語変換器，プログラミング算術，抽象データ型，およびフロイド・ホア型の公理的記法などのような定式化技法.

2. 段階的詳細化，モジュラ設計，プログラムモジュール，分割コンパイル，情報隠蔽，データフロー，および抽象化の階層などのような方法論.

3. テキストエディタ，構文向きエディタ，画面エディタなどのような，プログラム開発の自動化技法.

4. フォールトトレラント性，保安性，信頼性，回復，複数バージョンをもつプログラミング，多重の冗長性，チェックポイントなどのような，計算の信用性を増すための方法論.

5. ソフトウェア工具およびプログラミング環境.

6. プログラムおよびシステムの計測と評価.

7. 問題領域を，ソフトウェアシステムを通じて特定の計算機アーキテクチャに適合させること.

8. ソフトウェアプロジェクトのライフサイクル・モデル.

6.3 設 計

ソフトウェア方法論およびソフトウェア工具の領域における設計および実験の，おもな要素は次のとおりである.

1. 仕様言語（たとえば PSL 2, IMA JO)，コンフィギュレーション管理システム（たとえば Ada APSE における)，版管理システム（たとえば RCS, SCCS).

2. 構文向きエディタ，行エディタ，画面エディタ，およびワードプロセッサシステム.

3. 個々の，ソフトウェア開発において喧伝され，実用化されている方法論．たとえば HDM, Dijkstra によるもの，Jackson によるもの，Mills によるもの，Yourdon によるもの.

4. テストのための手順と技法（たとえばウオークスルー，ハンド・シミュレーション，モジュール間のインタフェース検査，テストデータをもとにプログラムの経路を数え上げること，事象の追跡など)，品質保証，およびプロジェクト管理.

5. プログラムの開発と虫取り，輪郭（プロファイル）作成，文書整形，およびデータベース操作のためのソフトウェア工具.

6. 高保安性システム（たとえば米国防省の）における，判定基準のレベルおよび検証手続きの定式化.

7. 利用者インタフェースの設計.

8. 信頼性と，フォールトトレラント性と，信用性をもつ非常に大規模なシステムの設計法.

7. データベースシステムおよび情報検索システム

この領域は，保存され共有される非常に大きなデータ群を，効率よく検索・更新できるように構成する，という問題を取り扱うものである．基本的な問いのうちには次のようなものが含まれる．①データ要素とそれらの間の関係をあらわすには，どのようなモデル概念を用いるべきか．②書き込み，検索，照合，および取り出しのような基本的演算を組み合わせて，効率のよいトランザクションを作り上げるにはどうしたらよいか．③これらのトランザクションと利用者の間の情報交換が，有効になされるようにするためにはどうすればよいか．④高水準の問い合わせを効率のよいプログラムに翻訳するにはどうすればよいか．⑤どのような計算機アーキテクチャによれば，問い合わせと更新を効率よく行うことができるか．⑥データを権限のないアクセス，公開，または破壊から保護するにはどうしたらよいか．⑦大きなデータベースを，同時になされる更新によって一貫性が損なわれることから保護するにはどうしたらよいか．⑧データが多数の計算機に分散配置されているとき，保護と効率を両立させるにはどうしたらよいか．⑨テキスト情報を効率よく検索できるように索引づけし，分類するにはどうしたらよいか．

7.1 理　論

データベースおよび情報検索システムの領域における理論の，おもな要素は次のとおりである．

1. 関係代数と関係算術．
2. 依存性の理論．
3. 並行性の理論．特に直列化可能のトランザクション，デッドロック，および多重化記録の更新における同期問題．
4. 確率的推論．
5. 整列および探索．
6. 性能の解析．
7. 支援理論としての暗号理論．

7.2 抽 象 化

データベースおよび情報検索システムの領域における抽象化の，おもな要素は次のとおりである．

1. データおよびデータ間の関係の論理的構造を表現するためのモデル．関係モデルと実体関係モデルの両方を含む．
2. ファイルの，高速検索に適する諸構成．たとえば索引つきファイル，木構造ファイル，反転ファイル，および連想記憶．
3. 更新（多重化記録の同時更新を含む）に際して，データベースの一貫性を保証するための諸方法．
4. 権限のない公表や変更を防止し，統計的推論を最小化するための諸方法．
5. 種類の異なるデータベース（たとえばハイパーテキスト，テキスト，空間図形，図，イメージ，規則群など）にまたがった問い合わせを行う方法．情報検索システムについても同様．
6. ハイパーテキストのような，テキスト情報を複数のレベルに含むこと，およびビデオ情報やグラフィクス情報や音声情報を含むことを可能にする文書モデル．
7. 人間的要因およびインタフェースに関する諸問題．

7.3 設　計

データベースおよび情報検索システムの領域における設計の，おもな要素は次のとおりである．

1. データベースの関係型，階層型，ネットワーク型，および分散型の実現を可能にする設計技法．
2. INGRES, System R, dBase III, および DB-2 のような，データベースシステムの設計技法．
3. LEXIS, Osiris, および Medline のような，情報検索システムの設計技法．
4. NLS, NoteCards, Intermedia, および Xanadu のような，ハイパーテキスト・システム．
5. 大規模データベースを磁気ディスク記憶装置に割り当てるための技法．
6. 読み取りに限定された大規模なデータベースを，光学的記憶媒体（たとえば CD/ROM や WORM）に割り当てるための技法．

8. 人工知能およびロボティックス

この領域は，動物および人間の（知的な）行動のモデル化を取り扱うものである．基本的な問いには次のものが含まれる．①行動の基本的モデルは何か．またそのシミュレーションをする計算機はどのように設計すればよいか．②知能は，規則の評価や，推論や，演繹や，パターン処理によって，どの程度記述できるか．③これらの方法で行動をシミュレートする計算機の，究極的性能はどのようか．④知覚データはどのようにして，似たパターンが似たコードをもつようにコード化されるのか．⑤運動コードは，どのようにし

て知覚コードと関連づけられるか．⑥学習システムの基本構造はどうなっているか．またそれらのシステムは，どのようにして外界に関する知識を表現しているか．

8.1 理 論

人工知能およびロボティックスの領域における理論の，おもな要素は次のとおりである．

1. 論理．たとえば単調論理，非単調論理，およびファジー論理．
2. 概念の依存関係．
3. 認知．
4. 自然言語理解のための構文的および意味的モデル．
5. ロボットの運動のための運動学および動力学．ロボットが使用する世界モデル．
6. 支援分野としての構造力学，グラフ理論，形式文法，言語学，哲学，および心理学．

8.2 抽 象 化

人工知能およびロボティックスの領域における抽象化の，おもな要素は次のとおりである．

1. 知識の表現（たとえばルール，フレーム，論理）およびその処理方式（たとえば演繹，推論）．
2. 自然言語理解および自然言語表現（音素の表現を含む）のための各種のモデル．機械翻訳．
3. 音声の認識および合成．テキストから音声への翻訳．
4. 推理および学習のモデル．たとえば不確実性，非単調論理，ベーズ推論，信念．
5. 発見的探索手法，分岐限定法，制御探索法．
6. 生体システムを模倣する計算機アーキテクチャ．たとえばニューロン回路網，コネクショニズム，疎な分散記憶．
7. 人の記憶のモデル，自律的学習，およびその他のロボットシステムの構成要素．

8.3 設 計

人工知能およびロボティックスの領域における設計および実験の，おもな要素は次のとおりである．

1. 論理プログラミング，定理の自動証明，およびルールの評価のためのソフトウェアシステムを設計するための技法．
2. 狭い適用領域に特化したエキスパートシステム（たとえば Mycin, Xcon），および新しい適用領域に合わせてプログラミング可能なエキスパートシェル．
3. 論理プログラミングのための実際的システム（たとえば Prolog）．
4. 自然言語理解システム（たとえば Margie, SHRDLU, および意味の選択方法）．
5. ニューロン回路網と疎な分散記憶の実現．
6. チェッカー，チェスなどのような，戦略を含むゲームをするプログラム．
7. 実際に稼働している音声合成装置，音声認識装置．
8. 実際に稼働しているロボット（固定型および自走型）．

9. 人間コンピュータ間のコミュニケーション

この領域は，各種の人間のそれに類似した知覚器官および運動器官を介してなされる，人間とコンピュータの間の情報交換と，人間側の概念構成を反映した情報構造を取り扱うものである．基本的な問いとしては次のようなものが含まれる．①対象物を表現し，またそれを視覚化する図形を生成するための，効率のよい方法はどのようなものか．②入力を受け入れ，出力を表示するための効率のよい方法はどんなものか．③誤った認識が生じ，そのために人間側で誤りが起こる，という危険を減らすにはどうしたらよいか．④データ群の中に蓄えられた情報を通じて物理的現象を理解するために，グラフィクスおよびその他の道具はどのように役立つか．

9.1 理 論

人間コンピュータ間のコミュニケーションの領域における理論の，おもな要素は次のとおりである．

1. 2次元以上の幾何学．解析幾何学，射影幾何学，アファイン幾何学，および計算幾何学を含む．
2. 色彩理論．
3. 認知心理学．
4. 支援領域としてのフーリエ解析，線形代数，グラフ理論，オートマトン理論，物理学，および解析学．

9.2 抽 象 化

人間コンピュータ間のコミュニケーションの領域における抽象化の，おもな要素は次のとおりである．

1. 図形を表示するためのアルゴリズム．平滑化，陰影づけ，隠れ線，光線追跡，隠れ面，透明な面，影，照明，境界線，色彩分布，スプライン曲線による表示，レンダリング，テクスチュア，アンチエイリアシング，コヒーレンス，フラクタル，アニメーション，および対象物の階層による図形の表現に関する諸

方法を含む.

2. CAD のためのモデル.

3. 物理的対象の計算機による表現.

4. イメージ処理と画像強調手法.

5. 人間計算機間のコミュニケーション. 対話のモードに関する心理学的研究を通じて人の側の誤りを減らし, 人の生産性を高めることを含む.

9.3 設 計

人間コンピュータ間のコミュニケーションの領域における設計および実験の, おもな要素は次のとおりである.

1. 各種のグラフィクス装置 (ベクトル型ディスプレイ, ラスタ型ディスプレイ, および一連のハードコピー装置を含む) のためのグラフィクス・アルゴリズムの実現.

2. 実験的なグラフィクス・アルゴリズムを, 新規に導入されつつあるモデルや現象に向けて設計, 実現すること.

3. ディスプレイ上のカラーグラフィクスを適正に使用すること. ディスプレイおよびハードコピー装置における, 色彩の正確な再現.

4. グラフィクスの規格 (たとえば GKS, PHIGS, VDI), グラフィクス言語 (たとえば PostScript), および特定分野専用のグラフィクス・パッケージ (たとえば化学のための MOGLI).

5. 各種の利用者インタフェース技術の実現. ビットマップ・ディスプレイにおける直接操作, および文字デバイスにおけるスクリーン技法を含む.

6. 異なるシステム, 機種の間の情報交換のための, 各種の標準的ファイル交換フォーマットの実現.

7. 実際に稼働している CAD システム.

8. 実際に稼働している画像強調システム (たとえば JPL における, 宇宙空間のプローブから受信される画像のための).

謝辞 多くの人々が, この報告書の原案に対して, 快く書面で意見を示された. それらの意見全部を詳細にわたってここに取り入れることは不可能であったが, 著者らはこの報告書を改訂するに当たって, そのすべてを考慮した. 意見を送付されたことに対し, 下記の人々に感謝する.

Alan Biermann, Frank Boesch, Richard Botting, Albert Briggs, Jr., Judy Brown, Rick Carlson, Thomas Cheatham, Neal Coulter, Steve Cunningham, Verlynda Dobbs, Caroline Eastman, Richard Epstein, Frank Friedman, C. W. Gear, Robert Glass, Nico Habermann, Judy Hankins, Charles Kelemen, Ken Kennedy, Elliot Koffman, Barry Kurtz, Doris Lidtke, Michael Loui, Paul Luker, Suzan Merritt, John Motil, J. Paul Myers, Bob Noonan, Alan Perlis, Jesse Poore, Terrence Pratt, Jean Rogers, Jean Sammet, Mary Shaw, J. W. Smith, Dennis Smolarski, Ed Upchurch, Garret White, Gio Wiederhold, Stuart Zweben

参 考 文 献

1) Abelson, H. and Sussman, G.: *Structure and Interpretation of Computer Programs*. MIT Press, Cambridge, Mass. (1985).

2) Arden, B, ed.: *See What Can Be Automated*? Report of the NSF Computer Science and Engineering Research Study (COSERS), MIT Press, Cambridge, Mass. (1980).

3) Denning, P.: What is computer science? *Am. Sci.* 73 (Jan.-Feb. 1985), 16-19.

4) Flores, F. and Graves, M.: Education. (working paper available from Logonet, Inc., 2200 Powell Street, 11th Floor, Emeryville, Calif. 94608.).

5) Newell, A., Perlis, A. and Simon, H.: What is computer science? *Sci.* 157 (1967), 1373-1374. (reprinted in *Abacus* 4, 4 (Summer 1987), 32.).

訳者コメント

木村 泉 (東京工業大学理学部)

大学における計算機科学のカリキュラムに関して ACM は, 1966 年にはいわゆるカリキュラム 68[1]を, 1978 年にはカリキュラム 78[2] を制定した. 前者は計算機科学 (Computer Science) というものが, 学問としてひとかどの内容をもっていることを (各大学の学長, 理事といった) 有力者たちに納得してもらうためのマニフェスト, という性格をもっていた. また後者は, その後の 10 年間に各地で多数設立されるに至った計算機科学科(ないし工学科)のカリキュラムの中か

ら平均値を見出す試み，という性格のものであった．これらのカリキュラム案のアイディアは，我が国におけるいわゆる情報関係専門学科のカリキュラムにも，かなり広く取り入れられている．

これに対し，ここに訳出した1989年1月出版の報告書は，カリキュラム 88 といったものを作ろうという動きの中で生まれてきたものであって，いまや一応常識化した計算機科・工学科のカリキュラムをもう一度洗いなおし，その内容を縦横十文字に並べなおして一つの筋を通そうとしたもの，として位置づけることができる．カリキュラム 68，78 と違って，これはカリキュラムに関する具体的な提案を示したものではなく，むしろ今後そういう具体的なカリキュラムを開発してゆくためになされた，一つの基礎工事に当たっている．

この報告書を ACM の許可を得てここに訳出することにしたのは，それが上記のような意味で豊かな示唆と説得力に富む文書だからである．特に我が国の大学における情報関係専門学科の将来を考えてゆく上で，議論の出発点としての価値がはなはだ大きいのではないかと思われる．ただしこの報告書を見てゆく上で，ぜひとも読者にご留意いただきたい点が2，3あるので記す．

その第1は，これは（上にも記したように）具体的なカリキュラムの提案ではないということである．たとえば付録「学問としての計算機分野」は，ちらと見るとカリキュラム案かと思い違いかねない顔をしているが，それは間違いである．基礎工事のコンクリートをでき上がった建築物と見違えて，そこに住み込むようなことはなさらないようにお願いする．

第2に，この報告書の記述の一部は報告書が書かれた時点で「撮影」された「スナップ写真」だ，ということである．この報告書は9行3列のマトリックスを導入して，計算機分野（報告書において，計算機科学と計算機工学を包括する意味で使われている言葉）をさいの目に切って見せている．マトリックスの三つの列は，理論，抽象化，設計となっている．また各行は，アルゴリズムとデータ構造にはじまり，人間コンピュータ間のコミュニケーションに終わる9つの「副領域」に対応している．報告書はこの 9×3 のマトリックスの各項目について詳細説明を与えることによって，計算機分野全体を定義しようとしている．ただし報告書にも明記されているように，この定義は時代とともに改訂されることが予定されている．三つの列からなる構造はかなり末長く生き延びるものと期待されるが，9つの副領域にはときには差し替えの必要が生じることも予想され，マトリックスの各項目の内容（報告書の付録に示されている）の改訂はある程度頻繁に必要となるものと考えられる．

このことに対応して付録の内容は，（恐らくは意図的に）やや粗削りなものとなっている．訳者は文部省の依託に基づいて情報処理学会内に設けられている「大学等における情報処理教育検討委員会」において，この報告書の勉強会に参加したが，その際報告書の詳細部分については「なんでここにこんなローカルな話が書いてあるんだ」とか「そんな話題はもう古いのでは」とかいうようなコメントが聞かれた．ただしそれらのコメントはおおむね特定の「副領域」の専門家から出され，面白いことに他のメンバーは「うん，でも全体的なイメージはこれで十分よくわかるじゃないか」ととりなすことが多かった．いずれにせよ読者としては，この報告書の「スナップ写真」的部分を金科玉条と思い誤ることのないように，ぜひ注意されたい．

第3に，この報告書には「入門科目系列」なるものが示されている．これは上記の 9×3 のマトリックスから，おもなところを拾ってうまく並べ，入門科目の中でひととおり教えてしまおうというものである．大変面白いアイディアで，大いに検討に値するが，報告書にもくどいほど断わってあるように，計算機分野のカリキュラムの全体を覆っているものではない．カリキュラム 68，78 における「コア」の部分をすら覆ってはいない．むしろ1年半3学期間にわたり，おおむね週3時間の講義と週3時間の実験を通じて，計算機分野の全体像がほぼわかり，各副領域の専門家と話ができるようになることを狙いとしている，と考えられる．副領域ごとのくわしい話は，この「入門科目系列」を前提としてなされる第4学期以降の科目の中で扱うことになる．またこの「入門科目系列」は一般的なプログラミング入門は含んでいない．それはこれに先立つ一般科目の中でやってある，という建て前である．さらに（計算機分野で深く必要とされる数学は，入門科目系列およびそれ以降の専門科目の中で扱うこともあるが），何ほどかの数学的基礎は他のしかるべき数学関係科目を通じて与えられるものと仮定している．この入門科目系列だけやればコンピュータのことは卒業などと思わないでいただきたい．もっともたとえば物理学や電子工学や建築学のような，他の分野を主専門分野 (major) とし，計算機科・工学を副専門分

野 (minor) とする学生にとっては，この入門科目系列までで計算機分野に関する，ほどよい知識，訓練が得られよう．

そして第4にだめ押しとして，これは計算機科学ないし工学に関する話だ，ということをいっておいたほうがいいかもしれない．計算機科学ないし工学は，たとえば米国では，物理学や電子工学と対等の，まとまった学問として意識されている．我が国では，そのあたりの理解がまだ必ずしも世間に行き届いていない．そのため，なぜコンピュータには半導体を使うのに，情報工学科では量子力学を必修にしないのか，などという議論がひょっくり飛び出したりする．それは，物理分野では計算機を道具として使うのだから，物理学科でもオペレーティングシステムやデータベースやソフトウェア工学のこまかい話を必修にすべきだ，という議論と同程度にバランスを欠いている．一人の学生が学部4年間でカバーできる領域の広がりには，おのずと限度がある．この報告書で示されているのは，「フルコース」として学生がほどよく満腹する程度の分量と学んで悔いのない知的内容をもち，かつ将来の成長に向けての基盤となり得るような，一つのまとまった学問体系である．軽々しくあれがない，これがないとあげつらう前に，そのことを考えてみる必要がある．

さてこの報告書は，(冒頭の注にも記してあるように) 縮約版であり，別に ACM から詳細報告書が発売されている．詳細報告書はここに訳出した内容において，現在の付録を付録Ⅰとし，別に付録Ⅱとして「入門科目系列」の詳細内容を追加したものに当たっている．その付録Ⅱを見ると，入門科目系列の具体的意図が一層よくわかり大変興味深いが，翻訳の許可が得られなかった．実際，付録Ⅱは付録Ⅰよりさらに一層「スナップ写真」的であり，粗削りな部分を多く残しているので，許可が得られなかった理由は，十分よく理解できる．ご興味のあるかたは直接原典に当たってみていただきたい．

ただこの付録Ⅱは，実験科目の取り扱いについて，特にわれわれ日本人にとって学ぶべき点が多いので，付録Ⅱのその側面について，ここで少しだけ論評しておくことにしたい．

とかくわれわれ日本人は，講義が中心で実験は補助，と考える傾向がある．まず講義でありがたい真理を承り，次に実験でその真理を謹んで確認するとともに，技能と精神を養う，という発想が抜きがたく普及している．だが，これは本報告書でも論じられているところであるが，計算機分野はそういう一般原理主導型の理解に適合しない性格をもっている．むしろちょっと手を動かして何かやってみて，その経験に基づいて考え，その繰り返しによって原理を身につける，というのが自然である．付録Ⅱの科目設計においては，そのことがしっかりと考慮に入れられている．

いま試みに，付録Ⅱの各段階における実験課題を拾い出してみると，次のようになる．

C1 1. アルゴリズムの基本概念 (6週)

a．サンプルプログラムを編集し，実行し，計測し，その輪郭 (プロファイル) を作成する．

b．与えられたプログラムを変更して，機能のちょっと違うプログラムにする．

c．仕様を与えられてプログラムを作る．

d．複数のアルゴリズムを比較し，与えられた条件下ではどれがよいか調べる．

e．再帰的プログラムと反復型プログラムの相互変換．

C1 2. 計算機の構成 (5週)

a．アセンブラ語による簡単なプログラムを作る．

b．コンパイラからの，アセンブラ形式の出力を調べ，翻訳の様子を見る．

c．算術代入文のハンドコンパイル．隠れた型変換を見つける．

d．コンパイラ語で書かれたプログラムをアセンブラ語で書きなおし，性能および開発の手間を比較する．

C1 3. 数学的プログラミング (3週)

a．連立一次方程式を解くプログラムを作り，たちの悪いデータでいじめる．

b．同じ問題をソフトウェアパッケージでやってみる．

c．数学的困難問題をふつうの計算機と並列計算機 (のモデル) で解こうとしてみる．

C2 4. データ構造とデータ抽象 (5週)

a．実現ずみの抽象データ型を利用して何かを作る．

b．抽象データ型における基本演算の，ある実現が正しいことを証明する．

c．抽象データ型の二つの実現のよしあしを比較する．

d．まったく新規の抽象データ型を定義し，実現し，評価する．

C2 5. 計算可能性の限界（3週）

a．シミュレータを使っていろいろなチューリング機械を作る．

b．計算量に関する理論的予測を実際のプログラムの性能測定結果と突き合わせる．

c．困難問題を設計し，実現し，評価する．

C2 6. オペレーティングシステムと保安性（3週）

a．協同動作する複数のプロセスを使って何か簡単な仕事をやってみる．性能を逐次的解法と比較する．

b．階層型ファイルシステムにおいて，ファイルを見つけて移動し，改名するコマンドスクリプトを作る．

c．適当な保安上の問題を解決する簡単なコマンドスクリプトを作る．それをほかの学生たちに破らせてみる．

C2 7. 分散処理とネットワーク（3週）

a．複数の計算機のファイルにまたがる宝探しをする．

b．データベースシステムにおいて，問い合わせの最適化とかアクセスの並列化とかいったことが効率にどう影響するか見る．

C3 8. 人工知能のモデル（5週）

a．記号処理のための簡単なLISPプログラムを開発する．

b．知識表現と質問応答のための，ルール形式による簡単なプログラムを作り，動かしてみる．

c．既存のもっと複雑なシステム（MYCINなど）をいじって，複雑さがどういう問題をもたらすか見る．

C3 9. ファイルとデータベースシステム（3週）

a．企業の様子を記述した簡単な関係データベースを作る．

b．教師が与えたデータベースについて，問い合せ言語を使って必要な情報を取り出す．

c．生成，更新，および検索のコマンドをもつハッシュ方式またはインデックス方式のファイルを設計する．

C3 10. 並列計算（3週）

a．哲学者の食事問題の解がどう動くかを，さまざまな事象系列について追いかける．

b．コネクションマシーンなどのような機械のシミュレータを使ってみる．

C3 11. ヒューマンインタフェース（3週）

a．ファンクションキーと適当なソフトウェア工具を使って，簡単なウィンドウ形式のインタフェースを作る．

b．上記においてファンクションキーをマウスに変え，優劣を評価する．

c．常識的でないヒューマンインタフェース（音声など）を，より常識的なメディア（キーボードなど）と比較する．

ここでたとえば「**C3 11**」などとあるのは，この科目の第3学期目において実施される第11モジュール，という意味である．

この課題群においてことのほか印象的なのは，課題がどれも「ひねった」ものではなく，むしろ「当たり前」のものばかりであって，講義の教えが完全にわかったかどうかテストし，至らない点があれば徹底的に叩きなおすという（日本的な）スタイルでなく，むしろ講義と両々あいまって原理の理解へと導くという，直感養成により適したスタイルを取っていることである．大いに学ぶべき点と思う．

訳者コメント参考文献

1) Curriculum Committee on Computer Science: Curriculum '68, Recommendations for Academic Programs in Computer Science, *Comm. ACM*, Vol. 11, No. 3, pp. 151-197 (1968).

2) Austing, R., Barnes, B., Bonnette, D., Englel, G. and Stokes, G. (eds.): Curriculum '78, Recommendations of the Undergraduate Program in Computer Science, A Report of the ACM Curriculum Committee on Computer Sceince, *Comm. ACM*, Vol. 22, No. 3, pp. 147-166 (1979).

(平成2年5月8日受付)