解説

# 教養としてのコンピュータ・サイエンス教育
## ——東京工業大学での試み

**渡辺　治**
東京工業大学 情報理工学研究科 数理・計算科学専攻

### コンピュータ・サイエンスを全学教育に

東京工業大学では，1998 年度より，全学の学生に対する情報の基礎科目として，「コンピュータリテラシ」と「コンピュータサイエンス入門」を開講している．共に学部 1 年次の教養科目の 1 つに位置づけられている．

前者は，コンピュータ・システムの基本的な使い方（Web，メール，文書作成等）と情報倫理に関する教育を行う科目である．一方，後者は，コンピュータ・サイエンス（CS）を基礎科学の 1 つ[4]として教える科目である．

「コンピュータリテラシ」のような授業は多くの大学で行われている．けれども，利用技術や倫理だけでなく，情報学的な考え方や研究手法を伝える授業も必要である．CS の基本の教育を通して，情報学的な見方を教えるのが「コンピュータサイエンス入門」の大きな目標である．

このような目標を持った授業の試みは，いくつかの大学で始まってはいるが[1]，その数はまだ多くないだろう．本稿では，東工大での「コンピュータサイエンス入門」の実施事例を紹介する．科目設立の経緯や実施状況も含めた全般的な紹介[2]や拙著[5]などとともに，今後，「コンピュータリテラシ」の次に位置する科目を計画されている大学の参考になれば幸いである．

なお，「コンピュータサイエンス入門」も，まだまだ体系化された授業ではなく，担当各教員の試行錯誤にゆだねられている点が多い．本稿の事例も筆者の授業経験の紹介が中心になることをご了承いただきたい．

#### CS は古い？

「いまさら CS ？」と思われる方も少なくないでしょう．確かに情報に関連する分野は，もはや CS という枠組みをはるかに超えて広がっています．しかしながら，「教養科目」としては，これらの広がりを考えつつも，その核となる考え方を教えるのが妥当だと思います．その核が CS だと思います．力学，電磁気学が今でも物理学の基礎であるように．

### 「コンピュータサイエンス入門」の背景など

東工大の状況など，科目「コンピュータサイエンス入門」の背景を簡単に紹介しておく．

東工大では，学部 1 年生は類と呼ばれる 7 つの学科群に分けられており，カリキュラムも類ごとに組まれている．ちなみに，1 類が理学部，7 類が生命理工学部，その他が工学部である．

東工大では伝統的に教養課程を設けない教育を行ってきたが，一般教養科目に相当する科目群は存在する．自然科学系では理工系基礎科目と呼ばれている科目群で，「コンピュータサイエンス入門」も，この理工系基礎科目に分類される．これらはすべて必修ではなく，いくつかを選ぶかたちの選択必修である．

学部学生に対する情報科目は，上記の 2 つのほかに，2 年次以降，各学科で行われるコンピュータスキル教育がある．プログラミング教育などコンピュータスキルに関する教育は，各学科の学問分野に応じた教育の方が適切である，という考え方もあり，情報基礎科目を検討する段階で，2 年次以降，学科別に行われることになったのである．

以上の状況のもと，「コンピュータサイエンス入門」では，コンピュータの使い方やプログラミングを教えるのではなく，コンピュータ・システムとその周辺技術を産み出してきた CS 的な物の見方— CS の心—を伝える科目として設計されたのである．

学生の履修状況についても簡単に述べておく．選択必修である「コンピュータサイエンス入門」だが，類によって，その扱いはかなり異なる．2 年次学科進学時にその成績を使用する類もあれば，履修申告数の上限があり，間接的に履修が制限されてしまっている類（例：生命理工学部）もある．

全体的には，2004 年度の実績（概数）で，1,150 名の 1 年生のうち 720 名が履修申告しており（うち，単位取得は 540 名），全学で 11 クラスで授業を行っている．各クラスには，大学院生のティーチングアシスタントが 2 名ついている．

## 何を教えるのか?―キャッチフレーズ

ひとくちに「CS の心」を伝えようといっても，その定義はなかなか難しい．また，人によって異なるだろう．もちろん，これは他の分野でも同様で，「物理の心とは何か？」という問いに，真正面から答えるのは難しいし，答えも一通りではないだろう．というより，それは物理を学んでみて次第に分かってくることで，そのような定義から入ること自体，不自然だ，といわれるかもしれない．

けれども，授業を設計する上では，その目標を設定する必要がある．特に CS のような歴史の浅い学問分野では，その分野を簡潔に表すキャッチフレーズが重要だ．現在の筆者の授業では


というスローガンを掲げている．このような世界観を「CS の心」としたのである．この意味を理解し，こういう見方もあるのだ，と知ってもらうのを目標としたのである．

もちろん，ここでいう「計算」とは数値の計算だけではない．コンピュータで行うことすべてを「計算」と総称している．言い換えれば「すべてはコンピュータ上で実現できる」ということである．また，非常に曖昧なもの，計算不可能なものなど，実際にはコンピュータ上で実現できないものもあるので，本当の意味での「すべて」ではない(そのことを知ってもらうことも重要である)．

「コンピュータに載せる」というのは，我々のジャーゴンかもしれない．そのジャーゴンの意味を学んでもらおう，というのを目標としたのである．情報処理学会からも最近，情報教育の目標として「手順的な自動処理の理解」が提案されたが[3]，それに近い目標だと解釈している．

### キャッチフレーズの意義

学問分野をひとことで表したり，教育目標を 1 つの標語にするのは大変難しいことです．標語など必要ない，あるいは，誤解を招くことも多いので危ない，という意見もあるとは思います．しかし，多くの人々（学生や他分野の教員）へ呼びかけるには，非常に強力な道具となります．その意味で，学会が「手順的な自動処理」を掲げたのは重要な意味を持っていると思います．

## 何を教えるのか?―目指す学習項目

「コンピュータに載せる」の意味を体験し，「すべては計算である」という考え方を理解するためには，具体的にどんなことを学んだらよいのだろうか？　実際の講義例を紹介する前に理想的な学習項目を簡単に整理しておく．

以下では，次のような 3 つの大きな学習項目にまとめ，その各々で理想的にはどのような内容を教えるのがよいかを簡単に述べる．

### 学習項目 1. 計算の基本要素を知る

「計算」を構成する基本要素を理解することである．たとえば，物理や化学では，物質を構成する最小単位である原子や分子について学ぶ．それと同様に，「計算」を理解するためには，その基本元素を知り，すべての計算は，それを元に構成可能なことを理解するのは重要だろう．

具体的には，データがすべて 0 と 1 のビット列から構成されていることの理解．そして，計算の処理は，非常に単純な演算や制御，そして単純な繰り返しで構成可能なことの理解などが目標である．その他，関連する学習項目については課題例のところで述べる．

### 学習項目 2.「計算」を組み立ててみる

目標の「計算」を行うには，上記の基本要素により「計算」を組み立てる（プログラミングする）必要がある．その際に，誤り（バグ）が混入すること，計算手順（アルゴリズム）が重要であること，の 2 点を体験を通して知って欲しい．

バグの混入を回避し，正確なプログラムを簡便に作るために，そして，より良いアルゴリズムの開発のために，コンピュータ・サイエンスで行われてきたさまざまな研究の一端を紹介したい．

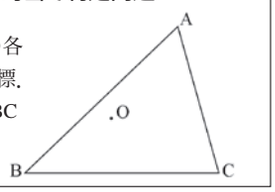
### 学習項目 3.「計算化する」手法を見る

コンピュータに載せるために一番難しいのは，載せたいものを計算の問題にするところである．これを計算化すると呼ぶ．いくつかの計算化の手法を見て，「ああ，こうやれば計算の問題になるんだ！」ということを実感してもらいたい．

ひとくちに「計算問題の形にする」といってもいろいろなレベルがある．最も典型的なのは仕様作成だろう．つまり，やりたいこと，コンピュータに載せたいことを明確にする作業である．最初は曖昧な目標を，厳密な文章，

**三角形的当て判定問題**

**入力**：△ABCの各頂点と点Oの座標．
**質問**：Oは△ABCの内部の点か？

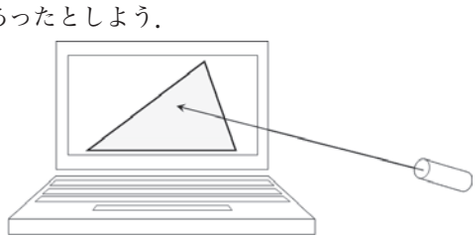



**解説**：いろいろな解法があるが，たとえば三角形の3辺を時計回りに囲むベクトル $\vec{AC}$, $\vec{CB}$, $\vec{BA}$ に対し，点Oがつねに左に位置することを，ベクトルの外積計算を用いて判定すればよい．

**図-1　三角形の的当て判定問題**

あるいは記号や式を用いて明確にしていく過程だ．できれば，こういった過程を体験させたい．

一方，たとえ問題が明確であっても，慣れない人には「どうやって？」と戸惑うものがある．たとえば，画面に表示した三角形の的に電子銃の光線を当てるテレビゲームがあったとしよう．



電子銃とその光線を受けるハードは用意されていたとして，光線が当たった座標から，それが与えられた三角形の枠内に入っているかを判定したい．人間ならば見れば明らかなことだ．では，これはどのように計算化するのだろうか？

この場合，問題の仕様ははっきりしている．**図-1** のように，点Oが，あらかじめ決められている三角形ABC内に入るか否かを判定する問題である．慣れている人には，この判定の計算化は明らかである．また，ちょっと説明されれば「なるほど」と理解もできるだろう．しかし，多くの人にとって，このような判定を通常の計算に結びつけるのは容易ではない．

このような簡単な計算化の実例はかなりある．大層なことでなく，我々が「当然！」と思っていることの中にも分野外の学生諸君にとっては新しい発見となるものが少なくない．それらを教えることも重要だと思う．

## 「コンピュータサイエンス入門」の授業例

「コンピュータサイエンス入門」では，講義にコンピュータを利用した演習を織り交ぜた授業形態をとっている．けれども，演習の取り入れ方や講義との関係なども，現在のところ各教員の裁量に任されている．私がこれまでに行ってきた授業では，以下に示した昨年の例のように，演習を主体とした3〜4個の課題を中心に授業を構成している．

**2005年度後期の授業例**

課題1：極詳細レシピの作成（演習3週）
課題2：究極の単純言語（演習4週，講義1週）
課題3：暗号文の解読（演習4週，講義1週）
まとめ：講義1週

（対象：生命理工学部1年（全150名中40名が選択）
形態：週1回90分の講義（または演習）×14週）

演習主体というのは，課題に対する演習で体験させ，その体験を補足しつつ，講義の中でCSでの主要な概念や結果を紹介する，というかたちである．各課題は，先に述べた3つの学習項目の多くの要素をカバーするように考えているが，まだまだ，毎年，試行錯誤を繰り返している．

## 課題の紹介：昨年度の授業から

上記に述べた昨年度授業の課題を紹介する．なお，各課題に付けられている：-）等の記号は，昨年度のアンケートでの学生の好感度を表している．

**課題1．極詳細レシピの作成：-(**

**概要**：イカ墨のパエジャ（Arroz Negro）のレシピの作成．単なるレシピではなく，料理初心者にも分かるように極々詳細に，しかも，分かりやすさや料理の効率の良さなども工夫したレシピを目指す．

資料として通常の料理の本にある程度のレシピを配布する．それに対し，まず，初心者(自分たち)に分かりにくい点を列挙し，それを調べさせ，詳細な情報のファイルを皆で作成する．それに基づき各自がレシピを作成する，という方法でレシピを作成させる．以上を演習3回で行う．

**評価**：昨年度試みに導入した課題だったが学生には不評

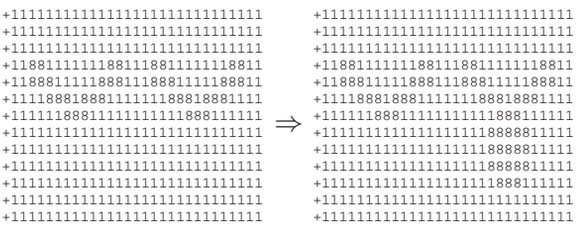
**実習の宿題はご法度**

生命理工学部の学生を担当した最初の年に，コンピュータを使った作業を宿題にしたために履修者が激減しました．専科の学生と違い，「要点は教えるから，後は時間のあるときに自分で演習室に来てやっておくように」は通じません．作業は，ほぼ演習の時間内に終えなければならないのです．これは，かなり厳しい制約です．未消化を避けるためには，あまり多くの課題をできないのが現状です．

```
$a1 = 5;   $a2 = 8;
$a0 = $a1;
while($a2 != 0) {
   $a0 = $a0 + 1;
   $a2 = $a2 - 1;
}
print $a0, "\n";
```

+1, −1 と単純な繰り返しのみから構成されるプログラムの例．2 つの変数に与えられた数の和を計算する．

**図 -2　言語 QL によるプログラムの例**



64 桁の自然数を用いたアニメーション．アルゴリズム的には，ステップごとに画面に出す数字を調整しているだけの単純なものだが，涙がポロッと落ちる雰囲気がよい．

**図 -3　言語 QL によるアニメーションの例**

だった．アルゴリズムという概念の導入として手順を正確に書く経験を持たせること，詳細な情報を分かりやすく，手際よくまとめる工夫を考えさせること，などをねらったのだが，時間数が少なかったために内容が未消化だった．また，学生にとっては，授業の趣旨はある程度分かったものの，新しく学ぶことも少なく，面白味に欠けた課題だったようである．

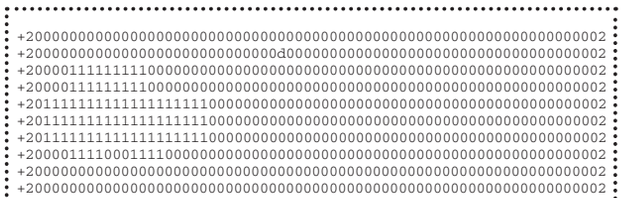
### 課題 2．究極の単純言語 QL ：-）

**概要：**計算要素として，自然数上の +1, −1，単純な比較，そして単純な繰り返しのみ，という究極に単純なプログラミング言語を用いて簡単なプログラムを作成する．

具体的には，言語 Perl の，ごくごく限られた機能だけを用いた極単純言語を QL と称し（「きゅーる」と読む），それを用いて**図 -2** のように，通常の四則演算（含：余りの計算）のプログラムを作成させる．

さらに，Perl では 100 桁の数でも扱えることを利用し，**図 -3** のようなアニメーション・プログラムを作成させる（この場合には，実行効率の面から四則演算は使ってもよいことにした）．以上を演習 4 回で行う．

講義（1 回）では，演習での経験をふまえ，単純な演算と制御の組合せだけでも計算が可能であること，さまざまなデータを数字（さらには 0 と 1 の二進列）で表せること，などを説明する．それが汎用コンピュータを可能にしたことも述べる．また，実際のプログラミングを行うための高級言語の話もし，それを実現するためのインタプリタやコンパイラについても解説する．さらには，計



車が左から右に動きます．見たときは感激しました！四則演算だけで実現しています．

**図 -4　単純だけれども工夫は可能！**

算可能性の話まで触れる場合もある．

また，関数や手続きを適切に定義し，分かりやすいプログラムを組み立てる必要性についても述べる．

**評価：**計算の原理を教えることが当初の目的だったが，それだけではないことが，やってみて分かった．

非常に単純な言語でアニメーションの作成をさせたことで，画像の処理も原理的には計算である，という点を実感してもらうことができたようだ．データの基本要素も含め，計算の基本要素全般を知ることのできる課題だと思う．

学生にも好評で，自分の発想で物を作る（たとえば，**図 -4**）という点に，おもしろさを感じた学生も多かった．

また，言語が単純なため，単純な処理でも結構複雑なプログラムになってしまう．そのため，注意してプログラミングしなければならない，という経験を得た者も多かったようである．

### 課題 3．暗号文解読に挑戦 ：-）

**概要：**シーザー暗号（アルファベットをある個数シフトして得られる暗号）の解読プログラムの作成．まずは，暗号文を作成するプログラム，および復号プログラムを Perl（前述の QL + 配列機能）を用いて作成する．

その上で，たとえば

```
dp uvri nrkjfe, kyzj zj r hlzkv zekvivjkzex
gifscvd, slk, fw tflijv,...（以下略）
```

のような暗号文を解読する（または，その手助けをする）プログラムを作成する．具体的には，文字の出現頻度を求め，英語ならば e が，日本語ならば a または i などが多く出現することを利用して，シフト量を推定するプログラムを作るのである．以上を演習 4 回で行う．

講義（1 回）では，公開鍵暗号の例や最新の情報セキュリティ技術などを紹介する．

**評価：**文字もコンピュータ内ではコード化（数字化）されていることを学ばせるための課題として考えた．一方，学生にとっては「頻度を求める方法」が新たな発見だったようである．

各文字に対する出現頻度を配列 hindo[26] にと

っておき，文字 c の出現ごとに hindo[code(c)-code('a')] を増加させる，という常識的な方法だが，多くの学生にとっては新鮮な方法だったようだ．こんなところにも「計算化」に対するギャップがあったのに気づかされた．

## その他の課題例

その他の課題例として，私が電気系学科群での授業で用いた課題や他の教員の使っている課題などから，いくつか紹介する．

### アルゴリズムの効率解析 :-}

同じ計算(目標)であっても，アルゴリズムの差によって簡単にできたり，とても時間がかかったりすることを経験させるための課題．具体的には，速度の異なるアルゴリズム 2 種に対し，その各々の計算量をアルゴリズムから導出し，アルゴリズムの実際の実行結果と照らし合わせてみる，などの実験を行う．

アルゴリズムの対象となる計算問題としては，最大公約数の計算，整列化，文字列照合，少し変わったところでは石選び問題(**図 -5**)，などがよく用いられている．

最大公約数の計算は，単純なものとユークリッドの互助法に基づくものとでは計算時間に指数関数的な差があるが，問題に面白味を感じてもらえない場合がある．一方，整列化や文字列照合は問題としては自然だが，アルゴリズムの良し悪しの差が驚くほどではない．石選び問題は，多少人工的で，しかも指数関数時間アルゴリズムしか知られていないが，$O(2^n)$ のアルゴリズムと $O(2^{n/2})$ のアルゴリズムの差($n = 40$ のとき，前者は 1 時間，後者は数秒)は，印象的である(また，教訓的でもある)．

### 計算精度の解析 :-}

有限精度の小数を用いて計算した場合の誤差の問題は，将来，数値計算を用いる可能性の高い学科の学生にとっては重要な話題である．誤差が生じること，アルゴリズムによっては誤差が大きく異なることなどを，実験で確認する．多少高度だが，アルゴリズムの誤差解析を行う場合もある．

私には，この種の課題の経験はないのだが，担当された教員方の話では，課題の選定や教え方によって好不評が大きく異なるようである．補数や浮動小数点での表し方だけの講義は受けが悪い．誤差解析のような高度な話でも，誤差発生の仕組みが「わかった！」というところまで到達するように講義をまとめられるし，受講者の多くがそう思えれば成功である．

**石選び問題**

**入力：** $n$ 個の石の重さ $s_1,...,s_n$ と目標の重さ $W$．

**質問：** 合計がちょうど $W$ となるような石の選び方を求めよ．

**解説：** ナップザック問題の特殊形で，一般には「部分和問題」と呼ばれている．公開鍵暗号の 1 つであるナップザック暗号の元問題である．

**図 -5　石選び問題**

### 再帰によるプログラミング :-)

パズルなどを例にとり再帰によるプログラミング，再帰的な見方を教える．プログラミングを経験した学生たちでも，再帰でプログラミングする経験はほとんどない．そのような学生にとって，再帰的な見方は新鮮な発見である．これも私には講義経験がないのだが，題材ではドラゴンカーブが好評のようだ．

### 情報セキュリティ技術に関する講義 :-)

演習課題ではないが，情報セキュリティ技術の原理や公開鍵暗号の仕組みなどを講義の中で教えている．原理としては，NP 問題の計算の難しさが多くの情報セキュリティ技術の基礎になっていることなどを説明する．公開鍵暗号の仕組みの説明には，RSA 暗号よりも，前述の石選び問題を用いたナップザック暗号の方が受けがよい．途中の数論的な話を飛ばしても「わかった！」という気になってもらえるからだろう．

## 授業におけるその他の要素

### 期末試験

1 年次の基礎科目として期末試験を行う．これは他の基礎科目と歩調を合わせるためもあるし，「基礎科目ならば期末試験ができて当然」という学内の(といっても一部の教員の)意見に配慮したためでもある．筆者の場合，最後のまとめの講義で，講義のときに説明した重要な項目等を復習し，それを元に文章題を 3 問程度出している(ノート，参考書等持込み可)．

### 技術教育の側面

学内からは「コンピュータの利用技術に関する教育もして欲しい」という要望も強い．その要望も考慮しているのだ，という姿勢は学内的に重要である．

実際，技術教育も当然行われている．利用技術の教育

は，前期の「コンピュータリテラシ」で行っているわけだが，夏休みを経て，操作だけでなく基本的な概念も完全に忘れている(!?)学生が少なくない．そうした学生諸君も，課題の演習を通して「実際の仕事で使う」経験を積むことで，ファイルをフォルダに整理したり，エディタでデータや文章を作ることが身についていくようである．一度忘れて，再び覚えるのもよいのかもしれない．

もう少し高度な技術としては，グラフ表示ソフト(例：gnuplot）の使い方がある．アルゴリズムの性能解析などの課題の演習で登場する．この種のソフトの使い方として重要なのは，実験結果に合う関数を求める際の考え方だろう．結果にうまく合わせようと，数十個のパラメータを使った高次式をデータにフィットさせようする学生が少なくない．適度な自由度の式でフィットさせることの重要性，そのためのこつなどを，アルゴリズム解析などの演習で教えることができるのである．

## 広めよう，基礎学問としての情報学教育

世の中では，コンピュータを，処理を高速で行う便利な道具としか見ていない．メールやWebなど，情報通信を含めた技術も，道具の使い方の延長としか捉えられていない．この狭い見方の転換が必要であることを，我々は啓蒙していかなければならないと思う．その手始めが新たな全学情報教育である．この種の教育を多くの大学に広めることが重要だろう．

東工大での経験を踏まえ，教養としての情報学教育を充実させるために鍵となる点を述べたい．

### 「何を教えるべきか」には信念を持って

全学教育をすると，「うちの学科の学生には，こういう内容を教えて欲しい」という要望が必ず出てくる．もちろん，これらの要望を考えつつ教材を設計するのは重要だが，科目の目標については確固たる信念を持った方がよい．

実際，1年生では，まだ各専門分野の知識も低い．生命理工学部の学生のために，たんぱく質のフォールディングを題材にした演習を行ったときも，学生の興味や理解がそこまで進んでいないことを実感させられた．

もちろん，間違った信念では話にならない．この点，大学の教養としての情報学教育では何を教えるべきか，という点を学会などを中心に皆で議論した方がよいと思う．

### おもしろい授業をする

課題や講義の題材選びでは，良い意味で，学生が喜び，おもしろい，と感じる授業を目標とするのが重要だと思う．良い意味で学生に受ける授業は学内でも支持や評価が得られるからである．

では，おもしろく感じてもらえる講義の秘訣は何だろう？　まず，第1に，自分で何かを創造させることである．プログラミングは器用不器用に関係なく，誰もが創造のおもしろさを感じることのできる非常に良い題材である．

第2に，何かを「あぁ，わかった」と感じさせることである．公開鍵暗号の仕組みのようにかなり高度な議論も，適度に誘導し，適度に端折ることで，多くの学生に，分かることの喜びを感じてもらえる．積み上げが必要な数学や物理などよりも，分かることのおもしろさを伝えやすい分野であると思う．

### 結局，プログラミング？

ではプログラミングの授業をすればよいのだろうか？それも1つのやり方かもしれない．しかし，プログラミングの習得を目標にしてしまうと，それにかなりの時間をとられ，情報学的観点を伝えるまでにいかなくなる恐れがある．

我々は，プログラミングをあえて目標とせず，CSの基礎概念を教えることで，情報学的観点を直に伝えようと試みたのである．もちろん，このアプローチ以外にも，いろいろな方法があると思う．今後，全学情報教育について多くの議論が出てくることを期待したい．

**謝辞**　東工大で「コンピュータサイエンス入門」を発足できたのは，多くの方々のご努力のおかげです．関係者の方々，特に佐々政孝教授，米崎直樹教授に敬意を表します．また，本稿執筆にあたり，授業内容についていろいろと議論させていただきました担当教員の方々に感謝いたします．

**参考文献**

1) 川合　慧：一般学生のための新しい情報教育―その理念と実践，東京大学教養学部情報・図形科学部会公開シンポジウム，(Jan. 2006).
2) 佐々政孝：東京工業大学の全学情報科目，情報理工学のすすめ，数理工学社，pp.39-56 (2005).
3) 日本の情報教育・情報処理教育に関する提言2005，情報処理学会情報処理教育委員会 (Oct. 2005).
http://www.ipsj.or.jp/12kyoiku/proposal-20051029.html
4) Wing, J. M. : Computational Thinking, Comm. ACM, Vol.49, No.3, pp.33-35 (2006).
5) 渡辺　治：教養としてのコンピュータ・サイエンス，サイエンス社 (2001).

(平成18年9月8日受付)

**渡辺　治**（正会員）
watanabe@is.titech.ac.jp

1980年東京工業大学情報科学科卒業，現在，東京工業大学情報理工学研究科数理・計算科学専攻教授．1987年工学博士．計算の理論，特にアルゴリズムの設計と解析，計算の複雑さの理論の研究に従事．電子情報通信学会，EATCS，LA各会員．