ソフトウェアの科学
玉井哲雄

身の回りのソフトウェア

現代生活の身の回りにはソフトウェアが溢れています．誰でも持っている携帯電話．この中には膨大な量のソフトウェアが組み込まれています．部屋にあるクーラー，冷蔵庫，テレビ，CD プレイヤー，さらには炊飯器，洗濯機といった家電製品も，さまざまなソフトウェアで制御されています．これらの機器のソフトウェアは，人からの操作の指示を受けて対応し，また機器の状況を把握して画面や音声で知らせ，さらに常時機器を制御しているわけです．

自動車に乗ってでかけるとしましょう．そのクルマにはたとえば 30 個ぐらいのコンピュータ（マイクロプロセッサと呼ばれるコンピュータの中枢をなす半導体チップ）が載っています．それらをソフトウェアが動かしています．目に見えるカーナビのような機器だけではなく，エンジン制御，ブレーキ制御，電気系統の制御など，多くの部分にソフトウェアが使われているのです．

今度は電車に乗るとしましょう．駅の自動改札機は切符やカードを素早く読み取り，問題がなければ切符やカードを返して開閉器を開く，という仕組みを実現するために，機械的・光学的な技術の粋を尽くしています．しかし，データを読み取った後の処理はもちろん，機械光学的な制御のすべてにわたってコンピュータが用いられ，その中では複雑なソフトウェアが作動しているわけです．

このように活躍しているソフトウェアですが，物として目に見えないためにほとんどその存在は意識されません．新聞などでソフトウェアが話題になるのも，たとえば銀行システムがソフトウェアの不具合で停止したというような，マイナスのイメージを与える場合に限られているようです．

ところでソフトウェアという言葉は，現在ではさまざまな機器で再生される音楽や映像を指すのにも用いられることが普通になっていますが，この章で扱うソフトウェアは，コンピュータによってプログラムとして実行される「コンピュータ・ソフトウェア」を想定しています．とくにデジタル化された音楽や映像は，デジタル機器により処理されるデジタル情報という意味ではコンピュータ・ソフトウェアとよく似ていますが，機器の動作そのものを変化させるコンピュータ・ソフトウェアと異なり，処理の対象となるデータという意味であくまでも受身のものです．しかし，現代のコンピュータ・ソフトウェアの動作結果は音声や映像を伴った豊かな表現形態で表されますので，両者の違いはますます縮まってきていることも事実です．

ソフトウェアはどう作られるか

さて，ソフトウェアはどのように作られ，どのように動くのでしょうか．ソフトウェアという言葉は抽象的な語で，コンピュータ上で動作する具体的なソフトウェアはプログラムと呼ばれます．英語では software は抽象名詞で，softwares というような複数形はありません．しかしプログラムは具体的に数えることができ，programs という複数形が存在します．そのプログラムはプログラミング言語という人工的な言語で書かれる記述として製

作されます．代表的なプログラミング言語には，C, Lisp, Java, Cobol などがあります．言語という以上，文法があり意味があり，語彙があり言い回し(慣用的な表現)があります．一方で，プログラムはコンピュータによって解釈され，実行されるものですので，自然言語（日本語や英語）と比べてあいまいさを許さず，論理性が要求されます．

図 1 にごく簡単な Java プログラムの例を挙げます．

```
class ChoHan {
  public static void main(String args[]) {
    int n;
    n = Integer.parseInt(args[0]);
    if (n%2 == 0)
       System.out.println("チョー");
    else
       System.out.println("ハン");
  }
}
```

図 1 簡単な Java プログラムの例

このプログラムは外から整数を受け取って，それが偶数なら「チョー」，奇数なら「ハン」と表示するものです．このようにプログラムはアルファベットなどの文字で記述され，プログラムを作成する行為は，文章による表現行為に似たところがあります．たとえばプログラムの大きさは，通常，行数で測られますが，この測り方は作家が書く小説の量を 400 字詰めの原稿用紙で何枚と数えるのと似ています．現在の携帯電話に載っているソフトウェアの量は，この測り方で 250 万行ぐらいあるそうです．また，高級乗用車の場合，そこに搭載されているすべてのソフトウェアを合わせると，何と 750 万行ぐらいにもなるといいます．

このような大きなプログラムは，もちろん一人で書くわけではありません．開発チームを作って組織的に製作します．小説と違うところは，書かれたものがコンピュータ上で実行されて，携帯電話でメールの送信や受信を可能にしたり，自動車のエンジンを制御したりというように，直接，実世界に働きかける点です．したがって，言語による記述を行うものでありながら，工業製品の生産と同じように「エンジニアリング」によって作られるのです．

ソフトウェアを構成するもの

ソフトウェアを構成する要素を，具体的に見ることにしましょう．ソフトウェアには次の 3 つの重要な側面があります．

- 情報の表現
- 情報の操作
- プログラムの部品と全体構造

情報の表現

ソフトウェアの役割は，広い意味での問題解決ということができます．たとえば，デジタルカメラのソフトウェアでシャッターボタンが押されたときに作動する部分は，レンズから知覚された外界の光学的データをもとに，適切な焦点距離を計算し，絞りとシャッター速度を定め，それに基づいてレンズと絞りとシャッターを制御する，という問題を瞬時に解決しているわけです．

そこで，問題をいかに表現するかが，ソフトウェア開発の重要な鍵になります．この表現のレベルは１つではありません．まず，ソフトウェアを設計する段階では，問題の表現は日本語や英語の普通の文章と図を用いてなされることが普通です．それでは表現としてあいまい性を含むという問題があるので，設計記述用の特別な人工言語（仕様記述言語などと呼ばれます）を用いることもあります．とにかく，問題表現の良し悪しによって，その解決の難易度も変わってきます．よい表現は，簡潔でエレガントな解法を導くものです．

情報の表現方法は，実はより広い範囲にわたる問題です．たとえばウェブのホームページをどのようにデザインしたらよいか，そのページがたとえばチケット予約用だった場合，利用者に選択させるメニューをどう表示したらよいか，というのも表現の問題です．

あるいは写真や絵をデジタル符号で表すのには，多くの形式が存在しますが，これも情報の表現法の問題に他なりません(図２参照)．

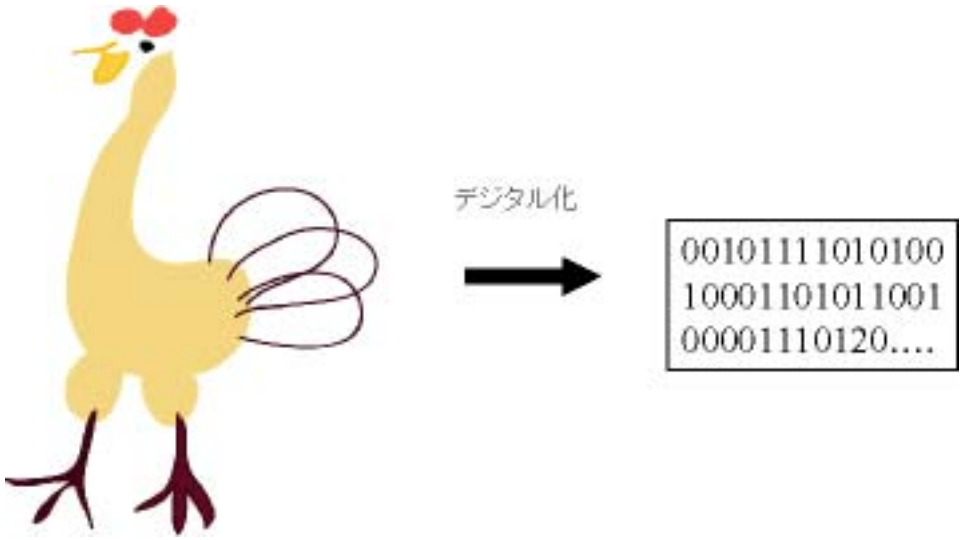


図２　画像のデジタル化

そもそもコンピュータで文字や数値を処理するのに，それらを２進，つまり０か１の２つの値をとる符号(コード)で表します．これがデジタルな情報処理のもっとも根本にある情報表現法です．たとえばアルファベットや数字の１文字を表すには，通常０か１という値をとる１ビットと呼ばれる単位を８個並べた符号を用います．この８ビット分の情報単位を１バイトと呼びます．日本語のかな漢字１文字を表すには１バイトでは足りず，通常２バイトが使われます．

情報の操作

表現された情報は，プログラムによって操作されることにより，初めて意味をもちます．操作のもっとも単純なものは，ある場所にある情報を別の場所に移すことでしょう．情報が移動する距離はさまざまです．1 つのコンピュータのメモリ上のある場所にあるデータを，同じメモリ上の別の場所に移すのは，もっとも近距離の移動になるでしょう．ハードディスクにある情報を主メモリ上に移すのも，それに次ぐ近さです．「メールを送る」とか「ウェブページをダウンロードする」という操作も情報の移動ですが，これはネットワークを経由し，地球の反対側とも行き来が可能な長い距離の移動です．なお，情報の移動のうち，もとの情報はなくならずにそのまま残る場合は，複写（コピー）と呼ばれます．

移動では，情報の中身は変わりません．むしろ長い距離を移動しても，雑音や妨害で中身が変わることがないように保証することが重要です．一方，情報の変換とか計算と呼ばれる操作は，情報の中身を変える，あるいは既存の情報から新しい情報を作り出すものです．

コンピュータとは，日本語にすれば「計算機」です．そこで，コンピュータは足し算や掛け算のような計算をする機械，というイメージをもつかもしれません．しかし，コンピュータで行われる処理は，たとえば「指定されたキーワードを含むウェブページを検索し，それを重要なものの順に並べて表示する」とか「人の話す言葉を文字による文章に変換する」とか「部屋の温度を一定に保つように空調機の運転を調節する」といった，単なる四則演算とは異なる操作の方がむしろ中心です．これらの機能もすべて，デジタル化された情報表現を一定の規則に従って変換させることによって実現されており，広い意味で「計算」と呼ばれます．

このようなコンピュータの動作は，人間の脳における情報の処理によく似ています．しかし，コンピュータの処理がもっとも基礎のレベルでデジタルな信号処理として実現されているのに対し，人間の脳内のもっとも基礎的なレベルの処理は，脳神経細胞の間の電気の流れや化学変化というアナログ現象に基づいているという違いがあります．もちろん，人間も一定の規則を活用して足し算や掛け算を行うことができるという意味では，デジタルな情報処理をする能力を備えていますが，それは数の概念，数字の表記，10 進法というシステムとその上での計算ルール，に基づく抽象化されたレベルの処理になっています．

ウェブページの検索や話し言葉の文章化のような広い意味の計算を，具体的な計算手順として厳密に記述したものを，アルゴリズムといいます．したがって，アルゴリズムはソフトウェアの重要な構成概念になりますが，これについては後で改めて説明します．

プログラムの部品と全体構造

携帯電話のソフトウェアの大きさは 250 万行ほどであるといいました（しかしその規模はさらにどんどん大きくなっています）．パソコンでもっともよく使われている Windows というオペレーティングシステムももちろんソフトウェアですが，その大きさは数千万行です(正確な量は公開されていません)．このような巨大なソフトウェアをどう作るのでしょうか．

他の巨大で複雑な人工物を考えてみましょう．たとえば航空機，スペースシャトル，医療用磁気共鳴断層撮影装置，あるいは原子力発電プラント．これらを開発するには，まず全体の構造をシステムとして設計し，それをサブシステムに分け，さらにそれをサブサブシ

ステムに分解するという大から小への構造分割を行います．一方で，部品として供給されるモーターやセンサーや制御装置を組み上げて，小さなものから大きな構造を作っていきます．部品としてのモーターも，さらに分解してみればコイルや磁心や電極というより小さな部品で作られているはずです．

ソフトウェアも同様に，全体の構造の設計とその段階的な分割，そして部品からの合成による組み立てという両方向のプロセスで設計開発されます．全体の構造は建築学から言葉を借りてアーキテクチャと呼ぶことが多く，部品もまた英語をそのまま用いてコンポーネントと呼ぶことが一般化しています．コンポーネントをなるべく標準化して数多く用意し，必要に応じてそれらを組み合わせてシステムを構築するという方向に向けて，さまざまな努力が続けられています．

アルゴリズム

ソフトウェアの機能を実現する核となるアルゴリズムという語については，すでに紹介しました．アルゴリズムをもう少し明確に定義すると「問題解決のための操作手順で有限回で終了するもの」となります．有限回で終了するという条件がないと，いつまで経っても結果が出ないという手順も許すことになり役に立ちません．銀行システムのようにノンストップで動き続けているソフトウェアはありますが，それも有限回で結果の出るアルゴリズムを繰り返し実行しているわけで，客を無限に待たせているわけではありません．

具体的なアルゴリズムの例を挙げてみましょう．

(1) 11 で割り切れるかどうかの判定

与えられた整数が 11 で割り切れるかどうかの判定をするアルゴリズムです．たとえば 135795 という数が与えられたとすると，このアルゴリズムは以下のように働きます．

S1: 与えられた数の 10 進表示の各桁の値を左から右に順に見て，交互に引き算と足し算を繰り返す．すなわち

1-3+5-7+9-5

を計算する．

S2: 結果が 0 なら割り切れると判定する．結果の絶対値が 1 から 10 までの数なら，割り切れないと判定する．今の場合は結果が 0 なので，割り切れることになる．

S3: 結果の絶対値が 11 より大きい場合は，その絶対値に対して S1 からの操作を繰り返す．たとえば，初めに与えられた数が 91919 だったとすると，S1 の結果は 25 となる．したがって S2 の段階では判定ができず S3 に進んで，25 に対して再び S1 を適用すると -3 となる．その絶対値は 0 でなく 10 以下だから 91919 は 11 では割り切れない．もし，最初の値が 91916 なら，割り切れることも分る．

(2) 最大公約数

2 つの整数の最大公約数をもとめるのに，ユークリッドの互除法と呼ばれる古くから知られた方法があります．ユークリッドの時代にはアルゴリズムという言葉は使われていませんでしたが，それに相当する概念は明確にありました．

2 つの数 x と y の最大公約数を求めるとします．

S1: x を y で割った余りを z とする．
S2: z が 0 なら y が求める最大公約数である．そうでなければ y を新しい x，z を新しい y として S1 に戻る．

このアルゴリズムで最大公約数が正しく計算できることは，それほど自明なことではありません．いろいろな x と y の組み合わせに対して，このアルゴリズムがどのように動くか，試してみてください．とくに与えられた y の値が x より大きい場合でも正しく動くことを確かめてください．

(3) 2 進探索

一定の順序で並べられたデータ（たとえば英単語のデータならアルファベット順，数値データなら小さいものから大きいものへの順に並べられたもの）から，特定のデータを探し出す，という問題を考えます．たとえば五十音順に並んだ名簿からある人の名前を探し出して，その住所や電話番号を知る，というような使い方が考えられます．

並んだ順に探していけばよさそうですが，たとえば 100 万件のデータがある場合，最悪 100 万回調べなければなりません．平均しても 50 万回調べることになります．しかし，このアルゴリズムを使うと，最悪でも 20 回調べればすみます．

データの並びを s とし，探したいデータを x とします．

S1: s のちょうど真ん中にあるデータを y とする．すなわち s の長さを n とすると，n が奇数の場合は(n+1)/2 番目，n が偶数の場合は n/2 番目を取って y とする．
S2: x と y を比べた結果の 3 つの場合分けに従い，次のように実行する．
　S2-1: x=y の場合．求める x が見つかった．
　S2-2: x<y の場合．x は s の左半分にあることが分るので，s の左半分（y より手前の部分）を改めて s として，S1 に戻る．
　S2-3: x>y の場合．x は s の右半分にあることが分るので，s の右半分（y より先の部分）を改めて s として S1 に戻る．

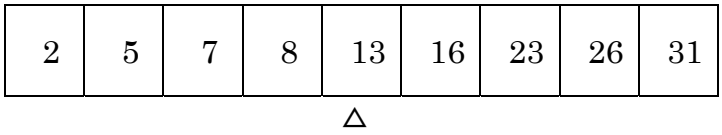
具体例をあげましょう．図 3 は一番上に示す整数の並びの中から，データ 16 を探す様子を示しています．

ソフトウェアの科学と技術

これまでの説明で，ソフトウェアにも固有の「科学」があるということの一端がうかがえたでしょうか．ここでは述べることができませんでしたが，ソフトウェアの科学には，計算の可能性，計算の複雑性，形式言語，抽象データ型，関数型プログラミング，論理型プログラミング，オブジェクト指向プログラミングなどのさまざまな理論が含まれます．一方で，すでに見てきたようにソフトウェアは実社会で大きな役割をになうものとなっており，社会に役立つソフトウェアを開発し発展させていくには，科学だけでなくそれをベースとした技術がきわめて重要です．

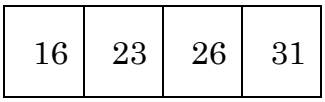
ソフトウェア技術の発達はこの 50 年ほどのことであり，建築土木技術，機械技術，電気電子技術などと比べれば，その歴史は浅いと言わねばなりません．しかし，この間にコンピュータの能力は飛躍的な進歩を遂げ，またインターネットの発達によりコンピュータに通信技術が結びついて，社会のあらゆるところで情報技術が活用されるようになりました．

16 を探すとする.



△

真ん中(13)と比べる．16 の方が大きいから右の区間にあると分る.



△

真ん中(23)と比べる．16 の方が小さいから左の区間にあると分る.

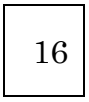


図 3　2 進探索の進み方

それを動かすには信頼性の高いソフトウェアを効率よく生産することが不可欠であり，そのためにソフトウェアの科学と技術の進展がますます求められています.

ハッカー文化

ソフトウェアは開発チームにより組織的に作られると言いましたが，個人であるいは 2-3 人の少人数で作られるソフトウェアもあります．それも家庭内で趣味的に作られるようなものばかりではなく，世界で広く使われ大きな影響力を与えているものの中に，個人や少人数で開発されたものがあるのです．そしてそのようなソフトウェアを創った人たちを，ハッカーと呼ぶことがあります.

ハッカーという言葉を辞書で引くと,「巨大システムに不法に侵入して内部を破壊する人」という意味のほかに,「コンピュータに精通し，熱中している人」「創造的なプログラムの作成に喜びを見出す人」という意味が挙げてあります(岩波「広辞苑」)．実際，ハッカーという言葉が 1950 年代の終わりぐらいに米国マサチューセッツ工科大学（MIT）で創られた頃から 20 年以上の間は，もっぱら後者の意味で使われていました．その意味でのハッカーの特徴は，若いということです．多くの天才的プログラマと呼ばれた人々は，10 代から 20 代で優れたプログラムを書いています．インターネットが爆発的に普及したきっかけは，WWW(ワールド・ワイド・ウェブ)というインターネット上の文書をリンクで結合して公開するソフトウェアが作りましたが，これを開発したのはティム バーナーズ・リーというイギリス人です．バーナーズ・リーは開発当時の 1989 年，CERN というジュネーブ近郊にある有名な素粒子研究所で働いていました．そのときの年齢は 34 歳で，典型的なハッカーというタイプではないかもしれませんが，一人で作ったソフトウェアがインターネットの世界を大きく変えたという意味では，やはりハッカーと呼ぶのにふさわしい

でしょう.

さらに WWW の普及を大きく後押ししたのは，WWW のページを簡単な操作で見やすく表示する閲覧ソフトウェアのモザイクで，これを開発し 1992 年に公開したのは米国の若い大学生マーク・アンドリーセンです．彼は当時イリノイ大学に在学中で 21 歳でした．このモザイクが現在広く使われているインターネット・エクスプローラーやネットスケープなどの WWW 閲覧ソフトウェアの元になっています.

ナップスターというファイル交換ソフトウェアは, 米国のレコード業界から著作権侵害で訴えられたこともあって物議をかもしましたが，そのアイディアと技術は斬新で，現在では合法的に音楽などのデジタル作品を頒布する手段として使われています．これを作ったのもショーン ファニングという米国の青年で，開発した 1999 年当時 18 歳でした．これらはハッカーの作ったソフトウェアが世界的に大きく受け入れられた代表的な例です.

日本のハッカーが作ったソフトウェアが世界で使われている例には，たとえば「まつもとゆきひろ」によるルビーというオブジェクト指向プログラミング言語などがありますが，その数はそれほど多くありません．これを読んだ高校生の皆さんの中から，センスがよくしかも世界の多くの人に使われるようなソフトウェアを作るような「ハッカー」が数多く生れ, 日本のソフトウェア技術を世界に向けて発信するようになることを強く期待します.