DIGITAL SERIES

未来へつなぐ
デジタルシリーズ

コンパイラ

佐渡一広
寺島美昭
水野忠則　著

24

共立出版

DIGITAL SERIES

未来へつなぐ
デジタルシリーズ

コンパイラ

佐渡一広
寺島美昭
水野忠則　著

24

共立出版

Connection to the Future with Digital Series

未来へつなぐ デジタルシリーズ

未来へつなぐ デジタルシリーズ　刊行にあたって

デジタルという響きも，皆さんの生活の中で当たり前のように使われる世の中となりました．20 世紀後半からの科学・技術の進歩は，急速に進んでおりまだまだ収束を迎えることなく，日々加速しています．そのようなこれからの 21 世紀の科学・技術は，ますます少子高齢化へ向かう社会の変化と地球環境の変化にどう向き合うかが問われています．このような新世紀をより良く生きるためには，20 世紀までの読み書き（国語），そろばん（算数）に加えて「デジタル」（情報）に関する基礎と教養が本質的に大切となります．さらには，いかにして人と自然が「共生」するかにむけた，新しい科学・技術のパラダイムを創生することも重要な鍵の 1 つとなることでしょう．そのために，これからますますデジタル化していく社会を支える未来の人材である若い読者に向けて，その基本となるデジタル社会に関連する新たな教科書の創設を目指して本シリーズを企画しました．

本シリーズでは，デジタル社会において必要となるテーマが幅広く用意されています．読者はこのシリーズを通して，現代における科学・技術・社会の構造が見えてくるでしょう．また，実際に講義を担当している複数の大学教員による豊富な経験と深い討論に基づいた，いわば“みんなの知恵”を随所に散りばめた「日本一の教科書」の創生を目指しています．読者はそうした深い洞察と経験が盛り込まれたこの「新しい教科書」を読み進めるうちに，自然とこれから社会で自分が何をすればよいのかが身に付くことでしょう．さらに，そういった現場を熟知している複数の大学教員の知識と経験に触れることで，読者の皆さんの視野が広がり，応用への高い展開力もきっと身に付くことでしょう．

本シリーズを教員の皆さまが，高専，学部や大学院の講義を行う際に活用して頂くことを期待し，祈念しております．また読者諸賢が，本シリーズの想いや得られた知識を後輩へとつなぎ，元気な日本へ向けそれを自らの課題に活かして頂ければ，関係者一同にとって望外の喜びです．最後に，本シリーズ刊行にあたっては，編集委員・編集協力委員，監修者の想いや様々な注文に応えてくださり，素晴らしい原稿を短期間にまとめていただいた執筆者の皆さま方に，この場をお借りし篤くお礼を申し上げます．また，本シリーズの出版に際しては，遅筆な著者を励まし辛抱強く支援していただいた共立出版のご協力に深く感謝いたします．

「未来を共に創っていきましょう．」

編集委員会
白鳥則郎
水野忠則
高橋　修
岡田謙一

はじめに

コンピュータを使えるようにするためには，まず自分の意思をコンピュータに知らせ，コンピュータで目的とする実行をさせることである．では，自分の意思をどうやってコンピュータに知らせるかが問題となる．コンピュータ黎明期においては機械語でプログラムを作成し，続いてアセンブラ言語でプログラムを作るようになった．しかしながら，人間が通常使う自然言語になるべく近いかたちでプログラムを作れたらということになり，高水準言語 COBOL や FORTRAN が生まれ，高水準言語を機械語に翻訳するコンパイラが開発された．その後，コンピュータ言語は PL/I，PASCAL，Ada などと発展し，それに伴い，コンパイラが各種開発されてきた．

このように，歴史的に各種のコンピュータ言語が開発されてきたが，オペレーティングシステム開発用に開発された C 言語が，簡単ながら，きめ細かい記述が可能なこともあり，一般的になってきた．また，Java がケータイ，スマホの開発に適したこともあり，Java もスマホプログラミングを中心に利用されてきている．本書では，主として C 言語をコンパイラの対象として説明し，Java についてはコンパイラの構成の観点から説明することとする．

本書は，15 週講義の教科書として使用することを想定しており，中間試験と最終のまとめの 2 週分を加味し，13 章構成となっている．各章のはじめにはその章のポイントやキーワードを示し，各章の内容を確認できるようにした．また，各章の終わりには演習問題をつけており，読者の理解度を確認できるようにしている．

本書の構成は以下の通りである．

まず，第 1 章でコンピュータアーキテクチャの簡単な紹介から，プログラムがどのように動作するかを述べる．続いて第 2 章では，コンパイラ設計に伴う準備として，コンピュータ言語の紹介から，コンパイラを開発するためにアルゴリズムを紹介し，言語が動作する仕組みを概説する．アルゴリズムが実際にコンパイラ上でどのように使用されるかについては，第 3 章以降で述べる．

第 3 章では，コンパイラの最初の処理である字句解析を紹介し，文字の並びからトークンへの並びへの解析について述べる．第 4 章では，プログラミング言語の持つべき文法について，その規定方法を述べ，コンパイラ処理で必要な構文木を中心に述べる．

第 5 章から第 9 章で，構文解析について述べる．まず，第 5 章では，トークンの列から，文法の一種である生成規則を逆に適用して，開始記号にさかのぼる道筋を求める下向構文解析につ

いて述べる．第 6 章では構文解析処理系作成ツールである yacc を用いて，構文解析処理の例を示す．第 7 章では，文法から終端記号の列を求める上向構文解析の概要と，SLR および LR と呼ばれる構文解析の概要を述べる．第 8 章では，yacc の仕組みを LALR 構文解析で述べる．第 9 章では，構文木に対して意味づけを加え，名前管理の方法を述べる．

第 10 章から第 13 章で，コンパイラから機械語が生成されるために，コンピュータとの関わりについて述べる．まず，第 10 章では機械語を生成するコード生成について説明する．そのために，仮定するコンピュータの機械語の命令を紹介し，コード生成，レジスタ割り当てを中心に述べる．第 11 章では，関数と手続きに関する処理方法を，関数呼び出し，引数，スコープなどについて述べる．第 12 章では，最適化について述べる．生成するコードは，実行速度などに多大に影響するため，機械独立レベル，機械依存レベル，機械語命令レベルで最適化が可能となる．第 13 章では，プログラムの実行方法として，インタプリタ型や仮想計算機型を紹介し，コンピュータ，オペレーティングシステムとの関わりと実現方式を述べる．

本書をまとめるにあたって大変なご協力をいただきました，未来へつなぐデジタルシリーズの編集委員長の白鳥則郎先生，編集委員の高橋修先生，岡田謙一先生，および編集協力委員の片岡信弘先生，松平和也先生，宗森純先生，村山優子先生，山田圀裕先生，吉田幸二先生，ならびに共立出版編集制作部の島田誠氏，他の方々に深くお礼を申し上げます．

2014 年 2 月

執筆者　佐渡　一広
寺島　美昭
水野　忠則

目　次

第 1 章
プログラムが動作する仕組み

□ 学習のポイント

近年，ICT (Information Communication Technology) は，商取引，エンターテイメントなどのビジネスだけでなく，我々の日常を支える生活基盤として社会に浸透している．ここでプログラミング言語は ICT の進展を支える重要な技術であり，複雑なソフトウェアを大量に開発するために不可欠である．従来はプログラマのような専門知識を持つ技術者がソフトウェアを開発していたが，近年は一般の人も自身のホームページを公開するためにソフトウェアを駆使するなど，さまざまな目的，方法での開発が登場している．このような，さまざまな目的や知識を持つ開発者による多様なソフトウェア開発では，それぞれの用途別に設計されたプログラミング言語を用いた開発が必要であり，これを可能としているのがコンパイラである．

コンパイラの役割はプログラミング言語を用いて開発するソフトウェア，つまり人が記述したテキストをコンピュータが解釈，実行できる機械語に変換することである．単にソフトウェアを解釈するだけでなく，コンピュータ上で最適に動作する効率のよい機械語を生成しなければならない．

本章ではコンパイラの目的を理解するために，コンピュータ上でソフトウェアが動作する基本的な仕組みと，プログラミング言語として定義された文法，さらに，この文法に従って開発するソフトウェアの目的を説明する．

- コンピュータを構成する基本要素である，CPU（中央処理装置），メモリ（記憶装置）がソフトウェアを動作させる仕組みを理解する．
- コンピュータを利用するために行うプログラム開発の役割，およびコンピュータ上でソフトウェアが動作する基本的なモデルを理解する．
- 近年のコンピュータ性能向上の状況を理解する．

□ キーワード

コンピュータ，CPU，記憶装置，プログラム，機械語，チューリングマシン，バグ

1.1 コンピュータの仕組み

図 1.1 はコンピュータを構成する要素を示している．基本的な構成要素は，頭脳となる CPU

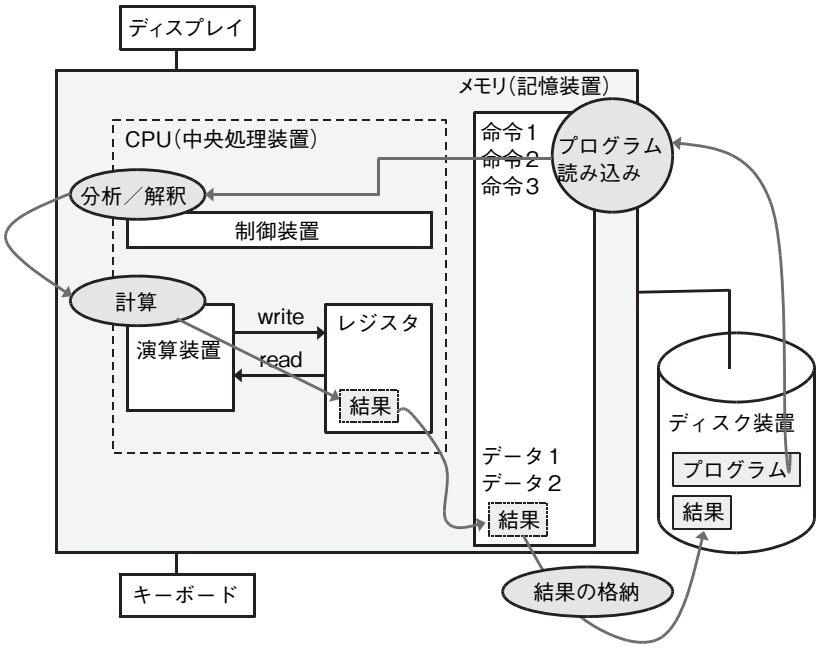

図 **1.1** プログラムが動作する仕組み

（中央処理装置）と情報を記憶するメモリ（記憶装置）である．CPU はメモリに記録されているプログラムを制御装置に読み出して分析，解釈し，演算装置が，その意味を理解しながら実行する．当然ながら CPU は人が指示する人間の言葉を直接的に理解できないため，プログラムとして記述された指示書を読みながら実行することになる．この指示書を記述する文法が（プログラミング）言語である．コンピュータが理解できる言語は，CPU とメモリが連携して行う計算処理の手順を定義できなければならない．ここで注意すべき点は，現在のコンピュータは定義されていないことや矛盾する手順を人間のように自動的に発見して補正する能力も知識もないことである．つまり正しい手順であろうが問題がある手順であろうが，CPU とメモリが行う有限個の手続きが動作可能である限り，高速に繰り返すだけの単純な機械である．

CPU と記憶装置によるコンピュータの仕組みを，図 1.2 を用いて，もう少し詳しく見てみる．CPU は内部にも，計算に用いる数値を格納する，レジスタと呼ばれる一種の記憶装置を保持している．計算を行う演算装置は，このレジスタに記録されている数値を読み取り，また一時的な計算結果を書き出しながら計算を実行していく．レジスタは演算装置とのやりとりを高速に行えるが，利用できる数は限定的である．この構成から CPU の計算速度が向上する，つまりコンピュータ性能が高くなるためには，まず演算装置が高速に計算できること，加えてレジスタへの読み込み，書き込みが高速にできること，さらに一度にレジスタに読み込める情報量を大きくすることが必要になる．

計算手順と計算に用いる情報を定義しているプログラムは，実行する前にはハードディスク

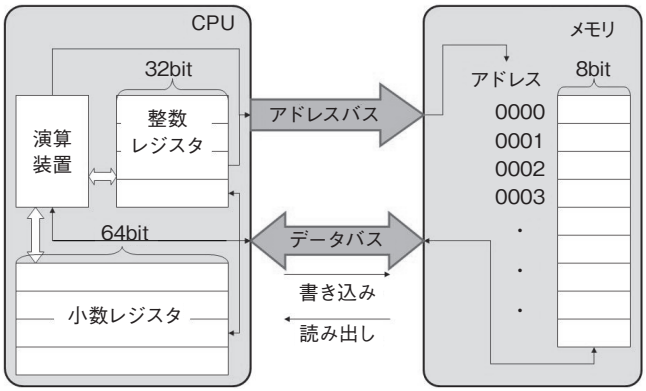

図 **1.2** コンピュータの詳細な動作

などに格納されている．利用者がプログラムの起動をコンピュータに指示すると，プログラムがメモリに記録（ロード）される．そして，CPU は記憶装置から，順次，計算単位を一時的に内部記憶装置に読み込みながら，このデータに対して計算を加えて，結果を記憶装置に登録する手順を繰り返す．

以上の操作により，利用者は最終的に期待する計算結果を得ることができる．利用者がコンピュータにプログラムの実行開始を伝えたり，コンピュータが利用者に結果を伝えたりするのは，ディスプレイやキーボードなどの入出力装置の役目である．

1.2 プログラム開発の目的

手順を定義しているプログラムの役割を，図 1.3 を用いて確認する．利用者はコンピュータに実行させたいこと，つまり何かしらの「課題」を持っている．現在のコンピュータには，人が

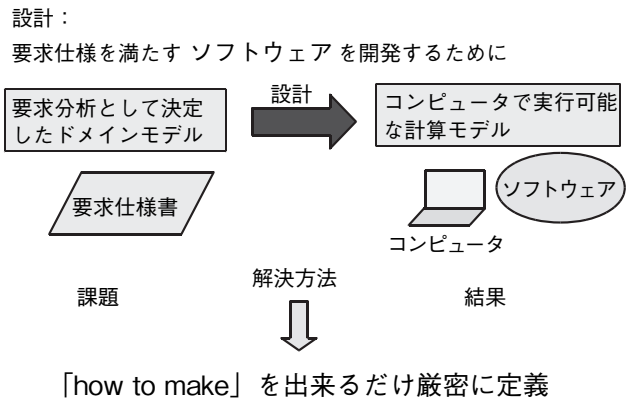

図 **1.3** プログラムの役割

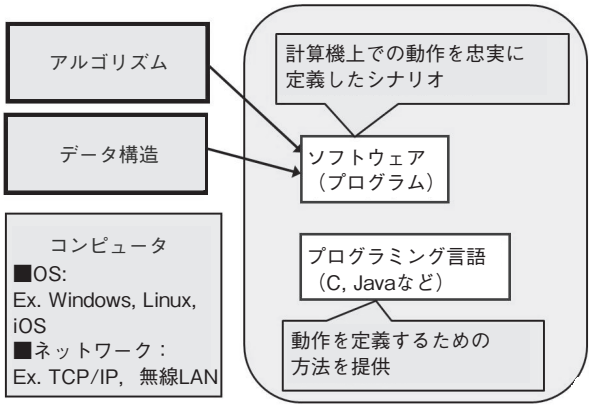

図 **1.4** プログラムとして動作するアルゴリズム

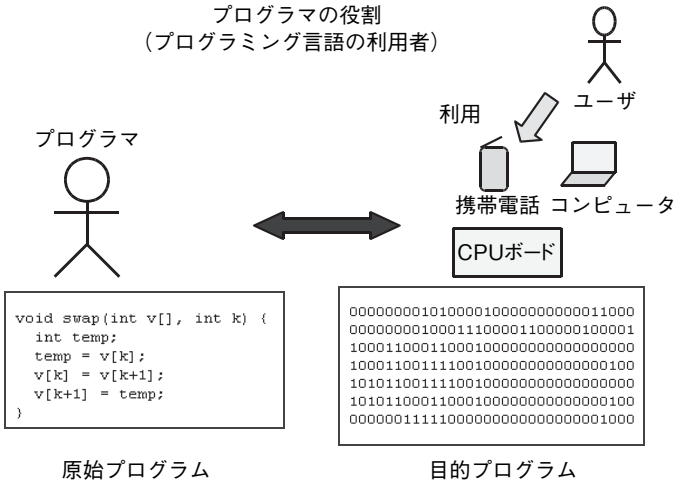

図 **1.5** プログラムと機械語

示す「課題」を見て，自身で調査，議論，検討を行い「解決方法」を決定する能力はない．人が少しコンピュータに歩み寄り，解決手順として何をして，その次に何をしてなどを定義して教えなければならない．つまり「how to make」は人が考えてコンピュータに与える必要がある．

ソフトウェア開発のさまざまな定義や手法を提供するソフトウェア工学では，この課題を「要求仕様」と呼んでいる．プログラムという指示書に記述されているのは，要求仕様を解決するためにコンピュータが実行する手続きである．1 つ以上の規則を有限回適用して問題を解く手続き，つまりアルゴリズムが記述される（図 1.4）．

このアルゴリズムは，当然ながらコンピュータ上で動作可能な手続き，つまりコンピュータ上で実行可能な計算モデルとして表現される必要がある．しかし，人が考案したアルゴリズムを直接コンピュータが読み上げて実行できるわけではない．コンピュータが実行できるのは CPU

が解釈できる機械語である．それならば，アルゴリズムを機械語へ翻訳する作業を行わなければならない（図 1.5）．機械語に定義されていることは，すでに説明したように CPU が実行する手順を解釈し，記憶装置に記録されたデータを操作する手順である．

ここではコンピュータがプログラムに指示された手順を実行する代表的な数学的モデルを説明する．代表的なモデルはチューリングマシン (Turing Machine) である．チューリングマシンは，1936 年にイギリスの数学者 Alan Mathison Turing (1912–1954) が発表した論文「計算可能数，ならびにそのヒルベルトの決定問題への応用」にて発表された理論である．チューリングマシンとは，テープとヘッドから構成されている機械である（図 1.6）．ヘッドは，テープの単位ごとにデータの書き込みと読み出しを行う．

図 **1.6** チューリングマシン (Turing Machine)

次の問題を解いてみよう．

【問題】ある数値 (423) に，1 を加える 「計算」

1. まず最下位の数字を知るため，ヘッドはテープに書かれた 3 の上にあってこれを読み取り，テープに 4 を書き込む．
2. たとえば最下位の数字が 9 の場合は 0 を書き込み，次にヘッドは左に移動してとなりの数字も読み取る必要がある．この場合は移動した先の数字は 2 なので，これを 3 にすることになる．
3. 移動した先の数字が 9 ならさらに左… を見る．数字がすべて 9 の場合は，全部 0 に書き換えて，さらに左の空白を 1 に，たとえば 999 であるならば 1000 にする．

コンピュータは，このような単純な手順を限りなく繰り返すことにより，あらゆる計算を行う．この単純なチューリングマシンの考え方により，コンピュータが動作する基本的な理論が説明できる．

1.3 コンピュータの発展と問題点

チューリングマシンによりコンピュータ上の計算が説明できるのであれば，CPU の処理速度が速ければ速いだけ，コンピュータ性能が高いことを意味する．この性能が我々の生活や社会に与える影響が大きいことは言うまでもない．最近の地球レベルの自然現象のシミュレーションにもスーパーコンピュータと呼ばれる超高性能コンピュータが用いられている．このコンピュー

タの性能が高いからこそ，我々は今日や明日の気象を予測するだけでなく，さらに何千万年，何億年先の地球環境の姿を予測する複雑かつ膨大な計算も可能なのである．もっと身近なところでは，次々に新たなサービスが展開されるインターネットを支えているのも絶え間ない CPU 能力の向上である．通信機器の速度向上，情報検索エンジンの高速化など，さまざまな要因が快適なネットワーク時代を演出している．CPU 速度はトランジスタの集積度に依存しているが，この集積度は 1970 年代以降，24 ヶ月で倍増している（図 1.7）．この傾向は「ムーアの法則」と呼ばれ，CPU 速度向上の目安として活用されてきた．

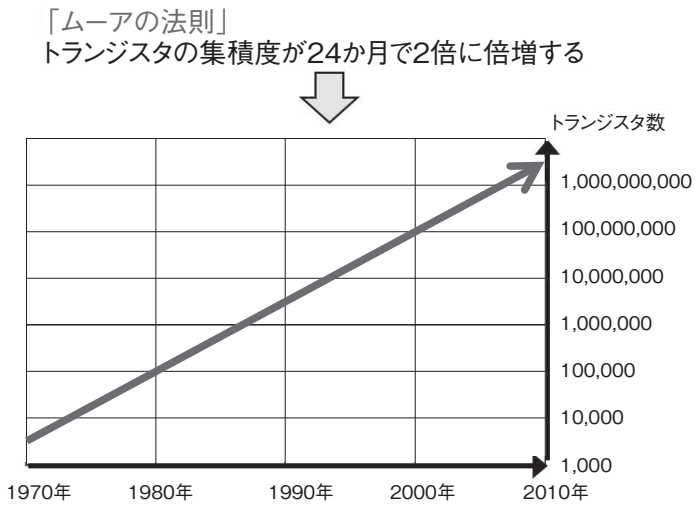

図 1.7 ムーアの法則

コラム バグの話

プログラムのエラーをバグと呼ぶ．世界最初のバグは 1947 年 9 月 9 日，米海軍と協力してハーバード大学が開発したコンピュータ「Mark II Aiken Relay Calculator」で発生したと言われている．この様子を記録したノートには，「First actual case of bug being found」と記録されている．この頃のコンピュータにはリレーが用いられていたため，この接点にバグ，つまり Bug（虫）が入り込んだため電気的に接触不良となり，コンピュータが誤動作を起こした，つまり利用者が意図しないプログラムの動作が発生したのである．

演習問題

設問 1 コンピュータを構成する基本機能において，高い性能を実現するためには何が必要か説明せよ．

設問 2 プログラムの役割を説明せよ．

設問 3 プログラムがコンピュータ上で動作できる仕組みを説明せよ．

設問 4 チューリングマシンにおいて，“189”に“2”を加える場合の動作を説明せよ．

参考文献

[1] 小迫秀夫 編,『コンピュータ概論』，共立出版 (1990)
[2] 飯島淳一,『情報システムの基礎』，日科技連 (1999)
[3] 矢沢久雄 著／日経ソフトウエア 監修,『プログラムはなぜ動くのか』，日経 BP 社 (2007)
[4] 白鳥則郎 監修,『コンピュータ概論（未来へつなぐデジタルシリーズ 17)』, 共立出版 (2013)

第2章
言語が動作する仕組み

□ 学習のポイント

コンピュータ上でソフトウェアが動作する方法は，プログラミング言語により異なる．たとえば C/C++ では機械語が動作する基本的な仕組みが用いられるが，Java の仮想マシンのように，特定のソフトウェアが特定のコンピュータに依存せず動作するために工夫されたものもある．本章では基本的な言語設計について述べ，プログラミング言語による動作の違いを説明する．

また，コンパイラを理解するための準備として，プログラミング言語を解析して機械語を生成するために必要となる基本的なアルゴリズムを紹介する．コンパイラによる言語解析を説明するために，オートマトンなどのモデルが用いられるが，これらのモデルに基づいて解析した結果を格納，また必要に応じて検索するために，コンパイラはさまざまなアルゴリズムを駆使する．ここではアルゴリズムとは何かについて述べ，最も基本的なアルゴリズムであるデータ構造と検索を説明する．

- 自然言語と人工言語の比較から，プログラミング言語の位置づけを理解する．
- 開発者による理解性の点から，プログラミング言語，アセンブリ言語，機械語の違いを理解する．
- 代表的な C++と Java プログラムがコンパイルされて，コンピュータ上で動作する仕組みの違いを理解する．
- アルゴリズムとは何か，また最も基本的なアルゴリズムであるデータ構造と検索を理解する．

□ キーワード

自然言語，人工言語，高級言語，アセンブリ言語，アルゴリズム，データ構造，検索

2.1 自然言語と人工言語

「言語」とは人が何かを伝えるための手段であり，一種の記号体系である．人と人とは，日常の生活の中で当たり前に言語を用いて会話することにより，お互いの意志疎通を図っている（図 2.1）．音声の記号列，文字のような記号列，あるいは身振り手振りに加え，手旗信号のような方法も，何かを伝える手段という意味では言語といえるかもしれない．現在の人間はさまざまな言語を駆使して，情報化社会と呼ばれる，伝えたい情報が溢れている世界を生きている．近年の研究ではイルカなどの動物も，いくつかの鳴き声を使い分けて意思疎通を図っているら

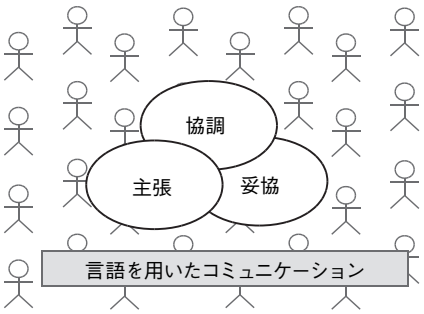

図 2.1 言語を用いたコミュニケーション

しいことが報告されている．この例を引くまでもなく，鳥の求愛ダンスや野生動物の威嚇なども，目的を持って何かを他に伝えるための言語と言える．

本書が取り上げるのは，コンピュータが他のコンピュータや人，その他のデバイスに対して，目的を持った実行手順を伝えるために，人が設計，開発したプログラミング言語である．この実行手順とは，すでに説明したように目的を実現するアルゴリズムである．我々が日常的に用いている日本語や英語は，もちろん人が歴史の中で開発，発展させてきた言語ではあるが，誰か特定の人が設計したものではなく，長い年月をかけて情報伝達の必要に迫られて作り上げてきたものである．このような背景から，人が用いている言語は，以下の 2 種類に分類できる．

- 自然言語 (Natural Language)
  自然発生的にできた言語．日本語，英語，フランス語などが該当する．
- 人工言語 (Artificial Language)
  ある情報を伝える目的のために人が作り出した言語．モールス信号やプログラミング言語などが該当する．

以下の英語を例に，言語の構成要素を見てみよう．

You are a blog writer.
Your editor has asked you to propose the good advice.

この英語は 2 つの「文」という要素から構成されており，それぞれが「単語」により構成されている．言い換えると，この英語は「文」の有限個の集合であり，さらに「文」は「単語」の有限個の集合である．単語は，Alphabet（アルファベット）や Vocabulary（語い）と呼ばれる「A, B, C, ...」などの言語の最小構成要素を，有限個並べた記号の集合である．これで記号列として言語を定義できるが，さらにアルファベットの並び，単語の並び，文の並びにより，伝えたい情報を表現する「意味」が定義されている．人工言語であるプログラミング言語は自然言語と異なり，コンピュータが解釈して実行できるように人が設計した言語であるため，その

解釈は厳密的に行われなければならず，あいまいな表現は許されない．たとえば

a=b+c*d

を計算する場合，一般的には

ケース 1: a=(b+(c*d))

と解釈される．しかし，

ケース 2: a=((b+c)*d)

と解釈する場合もあり得る．これらの違いは人のように一般常識を持たないコンピュータには，あらかじめ明確に定義しておかなければ実行できない．ケース 2 と定義して実行する言語の設計も可能である．自然言語は人によるあいまいな解釈でも知識や経験，常識が補って意図することを正しく実行できるが，コンピュータには通常，あらかじめ定義されている一通りの意味でしか解釈できない．このように人工言語に属するプログラミング言語は，自然言語と比較して目的が限定的であり，かつ表現力に乏しい言語といえる．

2.2 プログラミング言語と機械語

コンピュータが実行する言語は，一般にプログラミング言語と呼ばれるが，ここで少し言葉を整理しておこう．プログラミング言語は文法定義を意味し，この文法に従ってアルゴリズムが記述された文章をプログラムと呼ぶ．表 2.1 に示すように，プログラミング言語には，FORTRAN, ALGOL, BASIC, Pascal, C/C++, Smalltalk, Java, Lisp, Prolog など，さまざまな種類がある．これらは，多くの人が共通に利用する一般的な例であるが，このほかにも特定の開発プロジェクトの中で，試験の手順（シナリオ）を記述するために設計された試験シナリオ記述言語，自然現象の科学的な計算手順をコンピュータに入力するために設計された計算記述言語など，利用の目的が特殊であり，利用する人が限られる独特なものも多数ある．これらはプログラミング言語としての表現の特徴から，手続き型言語，関数型言語，論理型言語，オブジェクト指向型言語などに分類されており，目的に応じてアルゴリズムを記述しやすい方法が工夫されている．

では，プログラミング言語が，コンピュータの仕組みのうえでどのように動作するか考えてみよう．プログラミング言語は，人が目的に応じてアルゴリズムを記述しやすいように開発された人工言語であるため，逆にコンピュータ，つまり CPU は当然ながら直接的には理解できない（図 2.2）．CPU が理解できるのは，機械語で表現されるコードである．このため，プログラムを機械語に置き換えて解釈する仕組みが必要となる．

この変換を行うツールとして，コンパイラ，インタプリタ，クロスコンパイラなどの処理形態がある．これらは入力となるプログラムに記述されている，コンピュータに動作させたい手