●免責

本書に記載された内容は，情報の提供だけを目的としています。したがって，本書を用いた運用は，必ずお客様自身の責任と判断によって行ってください。これらの情報の運用の結果について，技術評論社および著者はいかなる責任も負いません。

本書記載の情報は，2012年1月現在のものを掲載していますので，ご利用時には，変更されている場合もあります。

以上の注意事項をご承諾いただいた上で，本書をご利用願います。これらの注意事項をお読みいただかずに，お問い合わせいただいても，技術評論社および著者は対処しかねます。あらかじめ，ご承知おきください。

●商標，登録商標について

・本書に登場する製品名などは，一般に各社の登録商標または商標です。なお，本文中に™，®などのマークは省略しているものもあります。

はじめに

本書はコンパイラの入門として企画された本です。従来，コンパイラとインタプリタは対比できるものとされ，それを解説した書籍もコンパイラ用とインタプリタ用とにそれぞれある状況でした。しかしながら最近ではコンパイラとインタプリタの境目がますますあいまいになり，そもそも両者は対比するものではないことが身のまわりのプログラミング言語を見てもわかるようになってきました。

そのようなわけで，本書はコンパイラの入門というよりも，言語処理系の入門ととらえるほうが適切かもしれません。言語処理系とは，プログラミング言語を実行するためのソフトウェアを指す言葉で，コンパイラとインタプリタの両方を含みます。本書の大半の章ではインタプリタの説明をしているように見えるかと思いますが，実際にはそれらの章はコンパイラとインタプリタの共通部部分を説明しています。それぞれの章が何を説明しているか，詳しくは1章で述べます。

言語処理系のプログラムを書くにあたって本書はJava言語を用います。言語処理系を作るといえば，CあるいはC++言語を使うべき，という向きもあるでしょうから，本書は実用性を無視していると思うかもしれません。また従来は一般的であったyaccのような構文解析器生成系も本書では用いません。

本書では，比較的新しいやり方を積極的に取り上げて言語処理系の作り方を解説するように努めています。CあるいはC++言語とyaccを使って言語処理系を作るやり方は遅くとも1980年代には完成されたやり方です。その後，プログラミング言語は進歩して高度になりましたし，プログラムの実行速度も非常に速くなりました。そろそろ従来とは異なったやり方を説明する入門書にも実践的な価値がでてきたと思います。

ソフトウェアの世界はあいからわず日進月歩ですが，大勢に使われるようになったソフトウェアは思わぬ長生きをするものです。長生きの間に技術は進歩しますから，その進歩にもある程度ついていけるよう，ソフトウェアは最初は少し新しすぎるくらいの技術で設計開発するのがよいと感じています。

著者はずいぶん昔，クロック100MHz程度，メモリ数百MBのワークステーション向けに言語処理系をC++言語で開発したことがあります。幸い，そのソフトウェアは10年以上あちこちで使われ続けたのですが，最後のころにメーリングリストにあるメールが届きました。それは確か若いドイツ人らしき人からだったのですが，曰く，このソフトウェアの設計を見てみたが，いろいろ問題がある。ちゃんとテンプレートを使ってライブラリを活用して書くべし，開発者はデザイン・パターンを勉強したほうがよい，抽象構文木はXMLで云々，という調子でした。

いろいろ書いてあったのですが，要はメモリをけちったり，速度を稼ぐことに何を汲々としているのだ，プログラムの見通しが悪すぎる，という指摘でした。そのメールが書かれた時の一般的なハードウェア，ソフトウェア技術を考えると妥当な指摘なのですが，作られたのは何せ10年前です。その当時の非力なハードウェアとソフトウェア向けに，その指摘に従って作っていたら，非実用的な言語処理系，という烙印をおされていたかもしれません（ちなみに指摘した当人は指摘だけしてコードは1行も書いてくれませんでした）。

しかしながら，この一件から学べることは，ソフトウェアは意外に長生きすることがあるので，その開発時点でベストな設計であっても，すぐに時代遅れの悪い設計になりがちだ，ということです。そうすると，ソフトウェアは少し新しすぎるくらいの技術で開発する，という戦略も一理あるように思えます。もっとも他人に非難されても気にしない，ソフトウェアは長生きさせないで，積極的に古いものを捨てて常に新しいものを作り続ける，という戦略もありえます。どれを取るかは人それぞれでしょう。

本書を読みながら，そんなことにも思いをはせていただければ幸いです。読者の皆様が本書の中から何かひとつでも得るものがあれば，著者の望外の喜びです。

2012年　新春
千葉滋

序文

本書はコンパイラの入門書ですが，コンパイラだけに限らず，プログラム言語の各種の機能とそれを実現する方法の基本的な考え方を教えてくれる本です。ただし，従来の多くのコンパイラ入門書と大分違った新しい内容の本です。従来の本は正規表現とオートマトン，LL文法やLR文法とその構文解析の方法，などの基礎知識を天下り的に解説し，小さなC風の言語のコンパイラを作ってみせる，という形のものが多いのですが，本書ではそれらについては基本的な考え方だけを丁寧に説明するにとどめ，とりあえず利用できるライブラリを利用して字句解析や構文解析を実装してしまっています。

そのように，字句解析や構文解析などのコンパイラの前半部分の解説は短くして，そのかわり後半部分に重点を置いて，従来の本にはあまり書かれていなかったいろいろな言語機能とその実現の仕方を，簡単な機能から始めて高度な機能まで，順を追って少しずつ追加修正する方法で解説しています。それらの機能としては，変数宣言の無い簡単な式だけの言語からはじめて，関数とクロージャ，配列，オブジェクト指向のクラス，型推論，インタプリタからコンパイラへなどがあります。

実装はJava言語を使って行われているのですが，このように追加修正をしていくと，通常はそれまでに作ったプログラムを書き直していくことになりますが，それはあまり好ましくありません。本書では著者が開発したGluonJという言語処理ソフトを使うことによって，以前のプログラムには手を付けずに追加修正する部分だけを別のプログラムとして書いています。ですから，いくつかの追加機能の組み合わせを簡単に変更することも出来ます。これはプログラムの大変好ましい開発法です。

そのようにすることによって，本書でとりあげている多彩な機能に対して実際に動くそれぞれ独立したプログラムのすべてが比較的短いプログラムとして完全な形で本書に収録されているのです。そのためにGluonJが使われているのですが，それをより生かすためにはもとのプログラムを拡張しやすいようなすっきりした形にしておくことが有効です。そのようなプログラミングのセンスも本書から学ぶことが出来ると思います。

本書では，基礎知識を天下り的に与えるのではなく，なぜそうするのかという基本的な概念を分かりやすく説明しています。また，クラスと関数は一見まったく異なる概念ですが，実はクラスは関数の延長線上の概念で説明できるといったことも説明されています。さらに随所に学生と教師との会話を挿入することで，疑問や反論を述べたり，関連する話題を提供したりして，読者にいろいろ考えさせるように書かれています。

本書は入門書として優れた本ですが，「今風の言語を今風の作り方」で作ってみせている本であり，コンパイラについてある程度知識のある人にも考え直させるところが多い本です。

中田育男