序文

C++プログラミングへの新しいアプローチ

この本を手にするみなさんは、なるべく早く、C++で役に立つプログラムを書きたいと考えていることと思います。ですから、この本はC++のもっとも役に立つ部分から解説を始めます。このような方法は、ちょっと聞くだけでは何の変哲もないかもしれませんが、実は過激なものです。これは、C++が基礎としているCから説明するのではない、ということです。この本では、最初から高レベルのデータ構造を扱い、その構造の基礎は後になってから説明するスタイルを取ります。この方法により、みなさんは、定石的なC++プログラムをすぐ書くことができるようになるのです。

私たちのアプローチは別の意味でも普通ではありません。この本では言語やライブラリの解説より、問題解決に集中します。もちろん、言語やライブラリの説明もしますが、それはプログラムを書くのに必要となってからです。言語やライブラリの仕組みを説明するために、プログラムの例を出すのではないのです。

この本が説明するのは、そのような仕組みではなく、C++プログラムの書き方なのです。そのため、C++について多少の知識がある人や、より自然で効果的なプログラムをC++で書きたい人たちに、特に有益だと考えます。初心者はC++の仕組みだけ学ばされて、この言語を使ってどのように日々の問題を解決するかは教えられていないことが、あまりにも多いものです。

私たちのアプローチは有効です。—プログラミングの初心者にも経験者にも

私たちは、Stanford大学で、毎年夏にC++に関する1週間の集中講座を持っていました。この講座では、もともと伝統的な方法を使っていました。つまり、学生はすでにCを知っていると仮定して、彼らにクラスというものの定義の仕方からはじめ、少しずつシステマティックに言語の残りの部分を説明していったのです。2日ほどが過ぎ、彼らが役に立つプログラムがようやく書けるようになるまで、混乱し不満を持つのがわかりました。もちろん、ポイントをつかんでからは、すばやい進歩を見せてくれましたが。

C++コンパイラが新しくなって、当時できたばかりの新しい標準ライブラリを十分使えるようになると、私たちは講座の進め方を全面改訂しました。新しいやり方では、はじめからライブラリを使い、役に立つプログラムを書くことに集中し、詳細な説明をするのは学生がそれを役立てられるほど、知識が増えてからにしたのです。

結果はドラマティックでした。前のやり方では、説明するまでにほぼ1週間かかったようなプログラムを、講座の初日が終わると、もう書けるようになったのです。さらによいことに、学生の不満はなくなっていました。

抽象

このようなアプローチが可能になったのは、C++言語自身と、私たちの理解が成熟してきたからです。成熟のおかげで、これまでC++やそのプログラマの中心的関心事だった低レベルの問題を無視できるようになったのです。

詳細を無視できるということは、テクノロジーの成熟に特徴的です。たとえば、初期の自動車は、あまりによく壊れたので、ドライバーはみなちょっとしたメカニックエンジニアでなければならなかったのです。何かトラブルが起こったときにどうすればいいのかを考えずに自動車で出かけるのは、おろかなことでした。しかし、現在のドライバーには、自動車を運転するだけなら、何も細かい工学的知識は必要はありません。もちろん、別の理由で工学的詳細を知りたいと考える人もいますが、それはまったく別の話です。

ここで言う抽象とは「選択的な無視（selective ignorance）」のことです。それは、今現在重要なことに集中し、他のことは無視するということです。これは現代的プログラミングではもっとも重要な考え方だと考えます。プログラムをうまく書くコツは、問題のどの部分が重要で、どの部分は無視していよいかを知ることです。すべてのプログラミング言語は、有用な抽象化をサポートするツール（道具、仕組み）を持っていて、有能なプログラマはその使い方を知っています。

抽象という考え方はとても重要なので、この本全体でその話をしていきたいと思います。ただし、実際にはいろいろな形態があるので、それらを毎回「抽象」とは呼びません。C++における抽象とは、関数、データ構造、クラス、継承などの形態をしています。これらがみな「抽象」なのです。この本では抽象を紹介するだけでなく、いたるところで使って見せるのです。

抽象がうまくデザインされ、適当なものが選ばれたなら、それらがどう動作するのかについて詳細に理解していなくても、使うことはできると思います。自動車を運転するのに自動車エンジニアになる必要はないのです。同様に、C++を使う前にそのすべてを理解する必要もないのです。

カバーする範囲

C++プログラミングをまじめに身につけたいなら、この本に書いてあることのすべてを理解しなければなりません。しかし、この本が、必要なことのすべてでもありません。

この少々矛盾したような言い方は、実はおかしくありません。このサイズの本で、みなさんが将来必要になるC++に関するすべての知識をカバーすることはできません。プログラマにより、またアプリケーションにより、必要なものが異なるからです。そう言うわけで、Stroustrupの「The C++ Programming Language(Addison-Wesley, 2000)」[*1]のように、C++のすべてを書いた本には、おそらくみなさん一人ひとりには必要のないことがたくさん書かれているはずです。ある人には有用な知識でも、他の人には不要だからです。

一方、一般的にとても重要で、それを知らなければよいものができないという知識があります。私たちはその部分に集中するのです。この本にあることだけを使うと、有用なプログラムをいろいろと書くことができます。実際、この本を出版前にチェックしてくれた人の中には、しっかりした市販のシステムをC++で書いているトッ

[*1] 訳注：翻訳は、『プログラミング言語 C++ 第3版』（アスキー）。

プレベルのプログラマがいますが、彼は「仕事で使っている仕組みの本質的部分のすべてが書かれている」とコメントしてくれました。

これらの仕組みを使って、みなさんは本当のC++プログラムが書けるようになります。それはCスタイルや他の言語のスタイルのC++プログラムではありません。また、この本の内容をマスターしたなら、さらに必要な知識も、その知識はどうすれば得られるかもわかってくるはずです。アマチュア天文家はよく「3インチ望遠鏡を作ってから6インチのものを作る方が、最初から6インチ望遠鏡を作るよりやさしい」と言います。

標準のC++だけを扱い、特定のベンダが提供する拡張の話はしません。このアプローチのおかげで、この本で書き方を説明するプログラムはどのような環境でも実行できるものになるはずです。しかし、ウインドウシステムで実行できるプログラムは扱わないということにもなります。そのようなプログラムは特定の環境、またしばしばそのベンダに結びついているからです。特定の環境で動くプログラムを書きたいなら、そのやり方を別のところで学ばなければなりません。しかし、この本を手放すことはありません！ この本のアプローチは普遍的で、ここで学ぶことは、将来どのような環境でプログラムを作成する場合にも、有効なのです。何にしても、お好みのGUIアプリケーションの本を読んでください。ただ、その前にこの本を読んでみてください、ということなのです。

CやC++の経験があるプログラマ諸氏へ

新しいプログラミング言語を学ぶとき、その言語のプログラムをすでに知っている言語のスタイルで書いてしまうことがよくあります。この本では、C++の標準ライブラリにあるハイレベルな抽象を最初から使うことで、そのようなことを避けるようにしたいと思います。もしみなさんがCやC++の経験者なら、この本のアプローチには良いニュースと悪いニュースがあるのです。（実はどちらも同じニュースの側面なのですが。）

そのニュースとは、この本にあるC++を理解するには、これまでの知識が驚くほどほとんど役に立たないということです。最初は予想以上にいろいろ学ばなければならないでしょう（これが悪いニュースです）。しかし、予想以上に速くいろいろなことが身についていくと思います（これが良いニュースです）。特に、すでにC++を知っている場合、最初にCを学び、Cの基礎の上にC++プログラミングを書いているかもしれません。それが悪いということではありませんが、しかし私たちのアプローチは全然違うのです。たぶん、C++の今まで知らなかった側面を見ることになるでしょう。

もちろん、文法上の細かい点には馴染みがあるかもしれません。しかし、所詮それは「細かいこと」なのです。これから扱う重要概念の説明の仕方は、おそらく、これまでみなさんが見てきたものとはまったく異なるものになるでしょう。たとえば、ポインタや配列は第10章ではじめて説明します。また、昔からよく使われてきた`printf`や`malloc`はまったく説明しません。一方で、標準ライブラリの`string`クラスを第1章から使います。私たちが「新しいアプローチです」と言うのが本気であることがおわかり頂けるかと思います。

この本の構成

この本は大きく2つの部分に分かれていると考えてください。前半は第7章までで、標準ライブラリの提供する抽象を使うプログラムを考えます。後半は第8章からで、自分で抽象を定義する方法を説明していきます。

ライブラリを最初に紹介するのは普通ではありませんが、私たちはこれが正しいアプローチだと思っています。C++のなかで特に難しい部分は、ライブラリ制作者のためにあります。ライブラリのユーザが、言語のこの部分を理解する必要はまったくないのです。この本の後半部にいくまでこのような話を省略すると、伝統的な方法に従うよりずっとはやく有用なプログラムが書けるようになるのです。

ライブラリの使い方を理解すれば、ライブラリの基盤である低レベルの仕組みを学ぶことができます。そして、それらを使って独自のライブラリを書くこともできるわけです。それだけではなく、「有用なライブラリとはどのようなものか」あるいは「新しいライブラリを書くべきでないのはいつか」までが、わかるようになっているのです。

この本は、他の C++の本に比べて小さいものですが、重要なことはどれも最低 2 回、特にポイントとなるものは数多く、繰り返し説明しています。その結果、この本には、参照個所が多々あります。たとえば、§ 39.4.3 とあれば第 39.4.3 節という意味です。(実際そんな節があればの話ですが。) 新しいものを説明するときには、見つけやすくし、重要ポイントとして目立たせるために、**太字**で書くことにします。

また各章の終わり (ただし最終章は除く) には、詳細という節を付けました。これらの節の役割は 2 つあります。まず 1 つは、その章で紹介したものを覚えやすくするためです。そして、いずれ必要になると思われる追加・関連事項をまとめてもいるのです。これらの節は最初に読むときは、ざっと目を通すくらいでよいかもしれません。それから必要になったときに読み返してください。

2 つの付録では、言語とライブラリについて、プログラムを自分で書くときにおそらく役に立つほどのレベルで、まとめと説明をしました。

この本の中身を大部分取り出す

プログラミングに関する本には、たいてい例題プログラムがあります。それはこの本でも同じです。これらのプログラムがどのように動作するかを理解するのに、実際にコンピュータ上で走らせてみることは、とても重要です。しかし、そのようなコンピュータの種類は多く、常に新しいものが出てきています。これは、コンピュータに関して何を言っても、いつかそれらの言葉は不正確になり得るということです。もし、C++プログラムのコンパイルと実行方法を知らないなら、`http://www.acceleratedcpp.com` を訪れ、私たちのコメントを読んでください。C++プログラムを実行する仕組みについて、情報とアドバイスを折に触れ更新していく予定です。また、このサイトには、いくつかの例題プログラムをコンパイルしたものや、みなさんに役立ちそうな情報も置いておきます。

謝辞

もし、その人からの助けがなければ、この本を完成できなかった方々がいます。まず彼らに感謝したいと思います。出版前にこの本を読んでチェックしてくださったおかげで、体裁も整ったと思います。それは、Robert Berger、Dag Brück、Adam Buchsbaum、Stephen Clamage、John Kalb、Jeffrey Oldham、David Slayton、Bjarne Stroustrup、Albert Tenbusch、Bruce Tetelman、Clovis Tondo です。また、Addison-Wesley のたくさんの人がこの本の出版に携わっています。特に私たちが知っているのは、Tyrrell Albaugh、Bunny Ames、Mike Hendrickson、Deborah Lafferty、Cathy Ohala、Simone Payment でした。Alexander Tsiris には、§ 13.2.2 で、ギリシャ語の語源を調べてもらいました。高レベルのプログラムから解説をはじめるというアイデアは、何年もかけて成長したものです。これは、私たちの講座に参加してくれた数百人の学生や講演を聞きに来てくれた数千人の聴衆から刺激を受けてのことです。

Andrew Koenig Gillette, New Jersey
Barbara E. Moo June 2000