プログラミングを通してコンピュータを操る

下島　真

大学でプログラミングの講義をしていると、プログラミングそのものではなく、プログラミング「言語」の講義となってしまうことがあります。もちろん、基本的な文法を知らなくてはプログラムを書けるようにはなりませんし、他人の書いたプログラムを理解することもできません。文法を覚えることは必要ですので、ある意味仕方のないことでしょう。といって、文法だけ知っていればプログラムを書けるようになるかと言われると、そうでもないのは、そういった講義の期末試験の出来を見れば一目瞭然です。文法を一通り習っても、それを具体的にどう組み合わせればよいのかよく分からないという人がかなりの数いるようなのです。

プログラミングの教科書は単なる文法書になっているものが多く、なかなかプログラミングとは何かということに触れているものがありません。実際、どの「プログラミング言語」入門書も細かな文法の違いを除けば似たり寄ったりの内容です。プログラミングは実装言語に因らないことが多いですし、結局のところ、コンパイルしてしまえばどれも似たような機械語になるわけですから、これは当然のことかもしれません。「プログラミング言語なんて基本はどれも同じで、どれかひとつを使えるようになればあとは何とかなるものだ」というのも確かに一理あるでしょう。では、どうすれば使えるプログラムを書けるようになるのでしょうか。

そもそも、プログラムを書くということはどういうことでしょうか。プログラミングの習得を英語などの外国語と比較することがあります。プログラミングもコンピュータと話をするために使用する言語だと考えれば似てなくもありません。文法だけ知っていても一向に話せるようにならないところなんか良く似たところでしょう。しかし、外国語を使うのはあくまでも皆さんと同じ人間であり――まぁ、あれはとても同じ人間とは思えないっていう人も中にはいますが――外国に行けば、文法など全然理解していなくてもそのうち何とか意思疎通ができるようになります。実際、皆さんも日本語はいつの間にか話せるようになっていたハズです。誰でも言葉を習得する能力はもって生まれているのです。プログラミングの場合も、ある程度量をこなして身に付けていくという側面はあります。しかし、コンピュータが相手の場合、何をどうしたいのか正確に記述する必要があるのです。所詮、コンピュータは人間の創った単なる機械です。言われたことは極めて高速に、間違いなく処理することが出来ます。途中で疲れたり、怠けたりすることもありません。しかし、コンピュータが一緒になって考えたり、気を利かせてこちらの意図を汲み取ってくれたりすることは絶対

にありません。いちいち事細かに指示を与える必要があります。この指示を与える唯一の手段が「プログラム」なのです。ですから、プログラムを書くにはコンピュータにさせたいことを細かく書き下せなくてはなりませんし、いろいろな場合を想定してあらかじめ対処方法を検討しておかなくてはなりません。「かくかくしかじかで、あとは…、まぁ適当にやっておいてくれたまえ」では済まないのです。

しかし、処理の手順を事細かに書くだけでは、なかなか良いプログラムが書けるようにはなりません。プログラミングの力をつけるには、自分でプログラムを作り、人のプログラムを読み、感触をつかむことが大切です。あまり細かなことに気を取られないで、ある程度まとまった規模のプログラムを見て何をしているのか抽象的なレベルで考えられるようになることが必要になってきます。物事を理解するには、具体から抽象へという思考活動からすべて始まるのです。抽象的なレベルにもいろいろありますが、ひとつのポイントは「モジュール」という概念ではないでしょうか。Ｃ言語ではファイル単位のインターフェイスに相当します。`static`と云うキーワードが使えるようになれば大分わかってきたと言えます。抽象的に考えられるようになると、自然にプログラムをいくつかの小さなカタマリに分割できるようになるでしょう。読むことを意識するようになれば読みやすい書き方に注意するようになるでしょうし、変数や関数の名前の付け方にも気を使うようになります。良い名前がすぐに思いつかないようなら構成をもう一度考え直した方が良いのかもしれません。

次に重要になってくるのが、プログラムの実行環境を熟知することでしょうか。WindowsでもUnixでも、あるいはMacでも、標準ライブラリとして様々な機能がAPIの形で提供されています。どのようなものがあって、どのように使うのか理解しておくのは良いプログラムを効率よく書く上で大事な作業です。標準ライブラリと同様のものを作るほど無駄な努力はありません。自分で変な関数を作り上げたりする前に、マニュアル等を調べて何か使えるものが用意されていないか考える癖を付けておきましょう。優秀なプログラムがインターネット上でたくさん公開されています。参考にしてノウハウを身に付けていきましょう。場合によっては作りたいプログラムそのものが既に存在していることだってあるのです。

プログラミングの中でも、ソケットを使ったネットワークプログラミングは、あまり初心者向きとはいえないかもしれません。構造体がたくさん出てきますし、オペレーティングシステムや通信プロトコルなどの理解が不可欠だからです。しかし、ネットワーク通信自体は以前よりずっと身近になってきています。いまや、インターネット利用がパソコンを使用する理由のほとんどを占めているのではないでしょうか。プログラムをまったく作ったことがなくても、自宅にADSLなどを引い

てLAN設定をしたことのある人は多いでしょう。ゲームにしても、インターネットを介して他のプレーヤーと対戦できるものが人気あるようです。具体的に何がどうなって通信ができているのか理解できなくても、抽象的なレベルでは既に十分な経験を積んでいると言えるのです。手紙とか電話にたとえて話をすれば、プログラミング上の諸手続きもそれほど特殊ではないのです。

ですから、ネットワークプログラミングは、取っ掛かりをつかんでしまえば理解するのがそれほど難しいというわけではありません。データ構造も接続方法も、いくつかパターンはありますが、基本的には同じような繰り返しですから、慣れるまではとりあえずブラックボックスのままで構わないのです。「セットアップ関数」にまとめておけばそれで十分です。あまり硬く考えずに、ネットワーク上で動く簡単なゲームのプログラミングでも楽しみながら、ゆっくりとネットワークプロトコルの基本やプログラミングのテクニックを習得することができるでしょう。まず、完成したプログラムを動かしてみて、面白いと感じることが大事です。それから中身を理解していけば良いのです。

ゲームのもうひとつの良い点は、中身が比較的単純であることです。もちろん、格好よく仕上げるためにはグラフィクスに凝ったりアルゴリズムを工夫したりすることが必要になってくるでしょう。しかし、単に動かすだけなら、ルールを極力単純化しても十分に楽しむことができます。何か物足りないと感じる様になればしめたものです。余裕が出てきたところで少しずつプログラムに手を加えていくことができます。ルールを少し複雑にして変化を加えても良いし、見栄えを改善するのも良いでしょう。自分からこう変えたいと積極的に思うことが第一です。違いが目に見えてわかるというのは結構大事なことですね。プログラミングの動機付けには最適ではないでしょうか。

最近は、CPUなどのハードウェアもプログラムを書いて設計します。ハードウェア記述言語という少し特殊な言語を使いますが、ソフトウェアには変わりありません。ソフトウェアによって中身の変えられる素子（例えばCPLDやFPGA）もたくさん発売されています。マルチコアCPUだって作れるのです。お金もかからないし特別な設備だって必要ありません。FPGAの中に独自のCPUをデザインしてネットワークで結ぶなんて最新の研究にも劣らないことが一人でできるようになってきたのです。

これからはコンピュータが様々な場所で使われる時代です。インターネットもますます利用されていくでしょう。さあ、皆さんも「コンピュータに振り回される人間」から「コンピュータを自由自在に操る人間」になろうではありませんか。