□ プログラミング言語

アルゴリズムをコード化する手段であるプログラミング言語とその仕組み

私たちがパソコン、携帯、音楽プレーヤなど、コンピュータを内蔵した機器を使っているとき、その動作はすべて、コンピュータ内部で動いている**プログラム**が作り出しています。これらの機器では、すぐ使えるように、あらかじめプログラムが記憶させてあります。たとえば、パソコンではOSやブラウザなどのプログラムが最初から入っています。機器によっては新たなプログラムを入れることで機能を追加できます（図1）。

また、ネットワーク経由で、検索サービスやネットショップなどを利用するときも、サーバの上でこれらの機能を提供するプログラムが動いていて、それを使っています。これらはすべて、プログラムの働きなのです。

なぜプログラミング言語が必要か？

ソフトウェアの技術が発達してきて、人間と会話できるような人工知能（AI）のソフトも作られているのに、なぜプログラミングが必要なのでしょうか。面倒なプログラミングなどしなくても、コンピュータに「あれをやってくれ」と命令するだけで済みそうに思えます。

それは確かに、一面では正しいです。簡単な「そのときだけの」仕事であれば、人間が「メールを○○さんに出したい」などと話しかければ、すぐにソフトが動いてくれます。しかし、それができるのは「人間がその場で命令して」、「正しいかどうかを確認するから」なのです。

実はコンピュータがいちばん役に立つのは、そういう場面ではなく、人間の介在なしに動くときです。たとえば大量の計算をするときには、人間がいちいち指示していたら人間の速度でしか計算できません。「このようなデータをこうする、そして次は…」といった手順としてプログラムを与えておくことで、コンピュータに目にもとまらない速さで大量に計算してもらえるのです。

また、人間が寝ていたり出かけたりしているときでも、ずっと動き続けて、何か必要なことがあれば対応する、というのもプログラムの得意とするところです。

これらを実現するにはやはり、プログラミング言語を使って「このような動作」というのを指示していくことが今のところ最善なのです。

プログラミング言語は、目的や用途に応じて、さまざまなものがあります。表1に、代表的なプログラミング言語を挙げました。

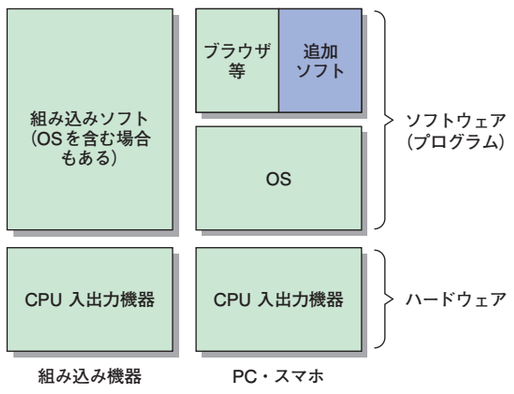

図1　機器の動作はプログラムが作り出している

表1　代表的なプログラミング言語

名称	主な用途
Python	初心者でもプログラムが見やすく配置されるように、インデント（字下げ）で構造を表す方式の言語。豊富なライブラリがあり、各種のデータ処理に広く使われるようになった
Ruby	日本発の言語であり、プログラムを簡潔で柔軟に書ける。Ruby on Railsというフレームワーク（ライブラリの一種）は多くのWebサイトの構築に使われている
JavaScript	Webページの中でHTMLと組み合わせて使われるように作られた言語。Webアプリケーションの柔軟な操作や表示はJavaScriptのプログラムを組み込むことで実現されている
Java	Webサーバ上のシステムや、Androidアプリなどの作成に使われる。大規模なシステム開発に適している
C C++	高速であり、パソコンやサーバ上の大規模なプログラム、高性能を要求されるプログラムに多く使われる
Swift	iOSアプリの作成などに使われる
VBA	Microsoft Officeのアプリケーションの機能を拡張するための言語

言語処理系

プログラミング言語で記述した**ソースコード**は、そのままではコンピュータのCPUが実行できません。CPUが実行できる命令（**機械語**）に直す必要があります。この作業を行って、ソースコードを実際にコンピュータ上で動かすためのプログラムが**言語処理系**です。これには大きく分けて次の2種類があります（図2）。

- **コンパイラ**：ソースコードを機械語に変換して実行可能にする
- **インタプリタ**：ソースコードを解釈し、記述されている動作を直接実現する

コンパイラを使うと、一度変換する手間がかかりますが、その後はCPUが命令を直接実行するので実行が高速です。インタプリタは変換作業が不要で、プログラム上の間違い情報などもていねいに示してくれますが、プログラムの実行はコンパイラを使ったときよりも遅くなります。

実際のCPUではなく、仮想マシン（VM、Virtual Machine）の命令（仮想マシン用の機械語）に変換する方式もあります。VMは仮想マシン用の機械語を読み込み、その命令を実行しますが、実行しながら部分的にCPU用の機械語に変換するなどの工夫を凝らして高速実行できるように工夫されています。Javaがこの方式を採用しています。

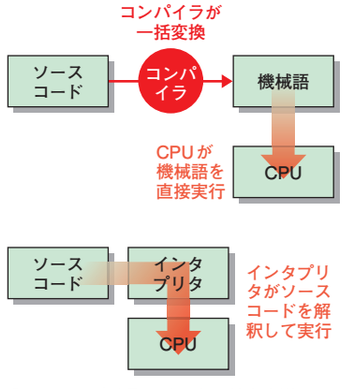

図2　コンパイラとインタプリタの違い

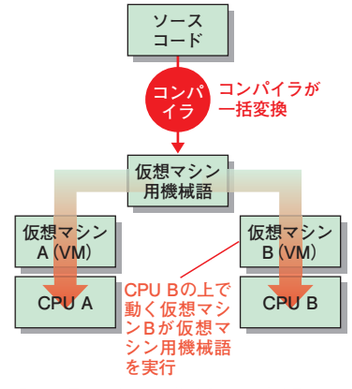

図3　仮想マシン方式でのコンパイラ

オブジェクト指向

オブジェクト指向とは、ソフトウェアの構造や機能を「もの（オブジェクト）」とそのはたらきを中心に設計するという考え方です。一度作成した部品をオブジェクトとして利用できるようにすることで、さまざまな機能を組み合わせたプログラムが比較的容易に組み立てられることから、今日のプログラミング言語の多くはオブジェクト指向の考え方を取り入れています。

たとえば、GUI（グラフィカルユーザーインタフェース）を持つプログラムは、ボタンや入力欄などの部品オブジェクトを生成し、これらに動作を記述することで作成できます。

たとえば、図4で画面に多くの図形が見えているものは、それぞれがオブジェクトで、色や形や位置を設定して「表示」することで画面に現れています。また、ボタンやスライダーなどのGUI部品もオブジェクトで、押したり動かしたりする機能を内蔵しています。そして、押したり動かしたりしたときに起こることは「アダプタ」という（画面上には見えない）オブジェクトを介して、他のオブジェクトに伝えられています。

GUIのような見えるものだけでなく、内部で動作している機能も、機能ごとにオブジェクトとして分けて開発できます。そのようにすると、その機能をそっくり他のプログラムに流用することもできますし、開発するときにその機能の部分だけを分けて開発できるので開発がしやすくなります。

このように、オブジェクト指向は高度なソフトウェアを作る上で必要不可欠なものとなっているのです。

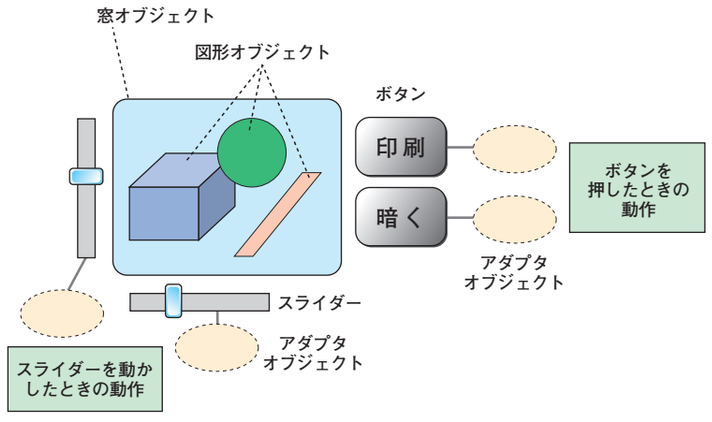

図4　さまざまなオブジェクトから成るプログラム