

第1章

プログラミングとは何か

著：柴田文彦

1-1

いろいろな角度から見る プログラミング

著：柴田文彦

講座を始めるにあたって、まずはプログラミングという行為そのものについて考えてみましょう。「プログラム」という言葉から、その仕組みに近付いてみます。また、一般的なプログラミングの手順や準備すべきことについても確認します。



1-1-1 「プログラム」のいろいろ

日本語でもよく耳にする「プログラム」

プログラミングとは、「プログラムする」という行為を表す言葉です。プログラミングという言葉は、ほとんどの場合コンピューター上のプログラムに対して使われますが、日本語でも一般的に「プログラム」という言葉を耳にすることはあるでしょう。

たとえば、「運動会のプログラム」、「コンサートのプログラム」、「テレビ番組のプログラム」、あるいは「ダイエットのプログラム」といった使われ方もあります。

こうした用法には、

- ・ 何らかの作業、行動の手順を示すものであること
- ・ 順に並んだ項目の名前、内容を示すものであること
- ・ 時間の経過と項目を関連付けるものであること
- ・ 始め、途中、終わりなどの条件を示すものであること

などの、ほぼ共通する機能があることに気付くでしょう。

こうした機能を並べてみると、コンピューターのプログラムも、一般のプログラムとほぼ同類のものと考えられるように思えます。

コンピュータならではの「プログラム」とは

とはいえ、一般のプログラムをそのままコンピューターに置き換えただけでは、コンピューターを使う価値がありません。それは、一般のプログラムを「実行」するのが、ほとんどの場合人間であるのに対し、コンピューター上のプログラムを実行するのは、当然ながらコンピューターだからです。

コンピューターは、人間とは桁違いの処理能力を持っています。それは単に同じことを速く実行するという意味の能力ではありません。コンピューターの動作をプログ

ラムすることによって、コンピューターの持つさまざまな能力を引き出すことができます。プログラミングの目的は、まさにそこにあると考えることもできます。

コンピューター上で実行するプログラムは、一般のプログラムに加えて、以下ののような機能を持つことが期待されます。

- ① 動的に動作、出力が決まる
- ② 条件判断や、それによる分岐がある
- ③ 繰り返し処理ができる

これらは、必ずしも独立した能力ではありませんし、他にもいろいろと考えられるでしょう。①は、プログラムを実行するたびに、前回とは異なる結果が得られるということです。それは気まぐれという意味ではなく、状況に応じた処理が実行されるという意味です。②は、①を実現するための仕組みですが、コンピューターのプログラムは、常に周囲から与えられる条件を考慮して、プログラムのどこをどのように実行するのかを自ら判断しています。③は、もう少し原始的な話になりますが、コンピューターのプログラムは、同じ(ような)動作を、文句も言わずに延々と繰り返すという特徴も持っています。これも、コンピューターの持つ高い処理能力の重要な要素の1つです。

こうしたコンピューターのプログラムの特徴については、第5章で実例を通して学びます。今の時点で、ピンと来るものがなくても、心配はいりません。



1-1-2 コンピュータープログラムの基本的な動作

コンピューターの簡単なしくみ

それでは、コンピューターがどのようにしてプログラムを実行するのかを見ていきましょう。実際のコンピューターや、これからプログラミングしていくことになるAndroidスマートフォンの中身の構成、仕組みについては第2章で詳しく学びます。ここでは、そのしくみを極限まで簡略化して考えます。それは、コンピューターを「CPU」と「メモリー」の2つの要素だけで考えるというものです。

CPUは、「Central Processing Unit(中央処理装置)」の略ですが、最近ではあまり聞かれなくなりつつある言葉かもしれません。最近では「プロセッサー」と呼んだり、場合によっては「コア」と呼ぶこともあるでしょう。いずれも指しているものはほぼ同じで、コンピューターの頭脳とも呼ばれる部分です。つまり、計算を実際に行ったり、データを処理する部分です。

メモリーについては、日常的にも使われる言葉なので、なんとなく機能は分かっているでしょう。メモリーはコンピューターの中にあってデータを蓄える機能を持った部分です。メモリーには色々な種類がありますが、ここではコンピューターの中で、CPUのすぐそばにあって、CPUが実行するプログラムを記録したり、CPUが処理するデータを蓄えたりするものを考えます。他にもSDカードなどのかたちでデータを移動

したり、長期保存したりするためにも使われますが、そちらではありません。また、最近ではパソコンでも、ハードディスクの代わりにフラッシュメモリー（SSD）を装備しているものがありますが、そのメモリーともちょっと違います。

コンピューターのプログラムは、このCPUとメモリーの間でやり取りされながら動いているのです。それについて言えば、他に介在する要素はありません。

CPUとメモリーのやりとり

ここで、メモリーとCPUの間のプログラムのやりとりを、少し詳しく見てみましょう。まず前提条件として、メモリーには、すでにプログラムと、処理の対象となるデータが読み込まれているものとします。

ちなみに、これはプログラムがコンピューターに「インストール」されている状態を指すものではありません。プログラムのインストールとは、外部のメディアやネットワークから、プログラムとデータがハードディスクやフラッシュメモリーに転送されている状態を指します。ここでは、インストールされたプログラムが起動された状態。もう少し正確に言えば、実行するために「ロード」された状態です。

コンピューターの動作を単純化して、プログラムが実行される手順を考えると、おおよそ次の5段階になるでしょう。

- ① プログラムの読み込み
- ② プログラムの解釈
- ③ データの読み込み
- ④ 演算の実行
- ⑤ 結果の書き込み

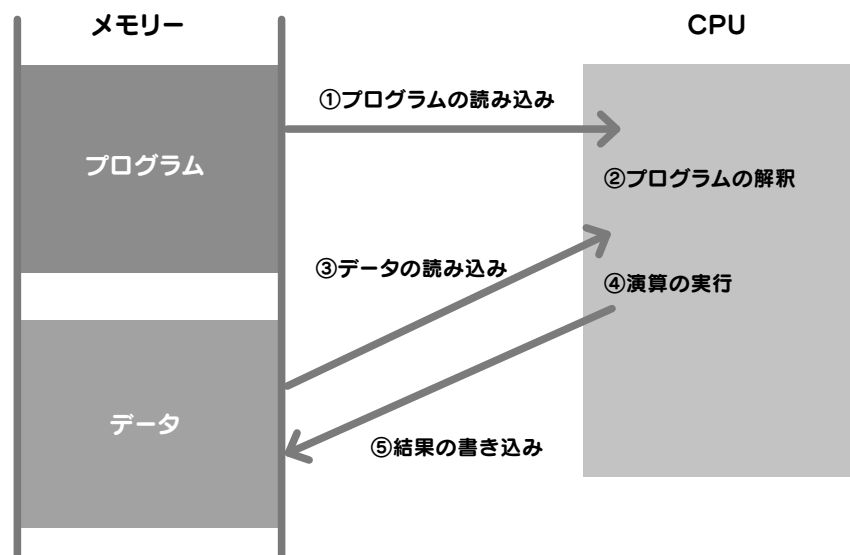


図1:「コンピューターの動作を極限まで単純化して考えると、メモリーとCPUのこうしたやりとりに集約できる

まず、CPUは①で、1回に実行する分のプログラムをメモリーからCPUの内部に読み込みます。ここで言うプログラムは、CPUが直接実行可能な「機械語」であって、普通の人間が見ても理解するのが難しい微細な動作を定義しているものです。

読み込まれたプログラムがどういう意味を持った命令なのか、②でCPUが解釈します。それだけで実行できる命令もありますが、通常はその命令が示す処理の対象となるデータを読み込む必要があります。そこで、③でCPUはその処理に必要なだけのデータを、再びメモリーから読み込みます。

これで命令とデータが揃ったので、④でCPUは「演算」を実行します。演算というのは、CPUが何らかの計算などを実行することを指す言葉です。一般的な数値の計算よりも広い処理を指しています。CPUが演算を実行した結果は、命令によっても異なりますが、多くの場合はその結果を残しておく必要があるでしょう。そのために、⑤でCPUは実行結果をメモリーに書き込みます。

CPUとメモリーは、言い換えればコンピューターというのは、このような処理を延々と繰り返しているのです。アプリ開発では、このような細かな命令によって構成されたプログラムを書くことはめったにありません。このような細かな命令を複数組み合わせ、人間にも意味の理解しやすいような単位にまとめたものをプログラムすることになります。しかし、それでもプログラムの基本的な考え方は変わりません。プログラムの動作をこのようにメモリーとCPUのやり取りにまで遡って考えると、理解しやすくなる場合も少なくないでしょう。

check!

「プログラム」と「データ」はごちゃまぜ？

左の図を見て、プログラムとデータが同じメモリーに含まれているのを不自然に思った人もいるかもしれません。本当は別々のメモリーに記録されているものを、簡略化して1つのメモリーとして描いているのではないかと。

実はそうではありません。プログラムとデータは、実際に連続する1種類のメモリーの中に保存されています。もちろんそれぞ

れの場所は独立していて混同されることはありません。コンピューターの構造には、色々なタイプが考えられますが、現在のコンピューターは、パソコン、スマートフォンも含めて、プログラムとデータに関する限り、ほとんどがこのような方式になっています。この方式は、発案者(ジョン・フォン・ノイマン)の名前をとって「ノイマン式」と呼ばれるものです。



プログラムの種類

コンピュータのプログラムには、動作する場所、使われ方、構造などによって、色々なタイプがあります。パソコンにも、WindowsやOS XといったOSがあり、その上で動作するアプリケーションがあり、何か周辺機器をつなぐために必要なドライバーなどがあることはご存知でしょう。

そうしたプログラムの種類ごとに、作成するためのプログラミングの手法は異なります。このため、プログラミング言語にも多様な種類があり、プログラマーに要求されるスキルもさまざまに異なります。1つをマスターしたからといって、別のタイプのプログラミングがすぐにできるようになるとは限りません。もちろん共通する要素もあるので、どれか1つにでも習熟しておけば、他のタイプのプログラミングの習得も楽になるでしょう。

この講座では、もっぱらAndroidという「プラットフォーム」上で動作するアプリのプログラミングについて学習します。プログラミング言語にはJavaを使います。それはプログラミングという多様な行為の中の、ほんの一部であることは、頭の片隅にでも入れておいた方が良くかもしれません。

プラットフォームとは、簡単に言えば、ハードウェアと、その上で動作するOSを合わせて指す言葉。プログラムの「実行環境」のようなもの。

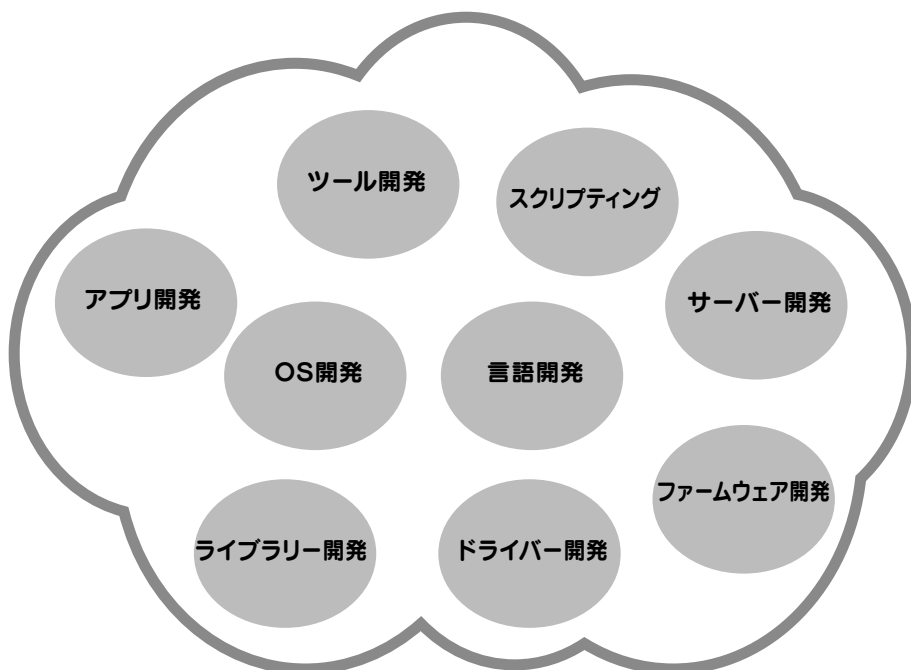


図2: プログラムにはさまざまな種類があり、それぞれに対応するプログラミングも性格の異なるものとなっている

プログラミングを職業として考えると、プログラムの種類によって、それぞれ別の職種と考えられるほど、性格が異なる場合もあります。

アプリのプログラミングとは

この講座全体がアプリのプログラミングを扱うものなので、アプリのプログラミングがどういうものなのかは、すべてを修了した時点で明らかになるとも言えるでしょう。それでも、ここで、アプリのプログラミングがどういう位置付けにあるものなのか、簡単なイメージを示しておきます。

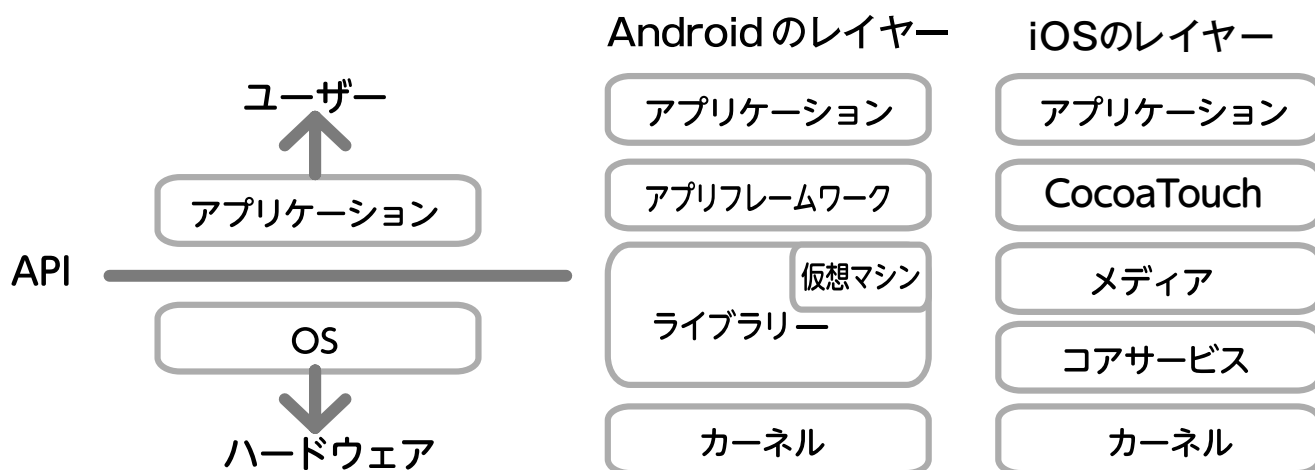


図3: アプリのプログラミングは、OSが提供するさまざまな機能を組み合わせて実現する

まず、パソコンにしるスマホにしる、その他多くのプラットフォームにしる、アプリはOSの上で動作します。スマホなら、ほとんどの機種がAndroidか、iOSというOSを搭載しています。AndroidとiOSのアプリには、通常は互換性がありません。つまり、Android用のアプリをiOSに持っていったっても動かないのです。その逆も同じです。なかには、Android用とiOS用に同じアプリが揃っているものもありますが、それは同じ機能と操作を実現し、同じように見える別のアプリです。

互換性がないのは、OSによって「API(Application Programming Interface)」が異なるからです。アプリは、OSが提供するAPIの機能を利用して動いているのです。さらに言えば、OSが動作しているハードウェアも異なり、それによってCPUも異なるので、そもそもプログラム自体に互換性がないとも考えられますが、APIの違いに比べれば、それはむしろ小さな問題と言えるでしょう。

AndroidのAPIについては、これからこの講座でじっくりと学んで行くことになるので、ここでは詳しく触れません。OSの中にはさまざまな機能が階層状に積み重なっていて、レベルの異なる多くの機能が、その上で動作するアプリから利用できるようになっています。従って、APIはスパッと一直線に切れるような単純なものではないと考えられます。

一方、OSが提供する機能を図3のような階層で考えてみると、例えばAndroidとiOSを比べてみても、だいたい同じような機能を、同じような階層で提供していることが分かります。とはいえ、それぞれアプリからの利用のしかたは異なっているので、両者の互換性を維持するのはたやすいことではありません。



アプリ開発のサイクル

アプリ開発をはじめとするプログラミングという作業は、最初から最後まで既定の手順に従って一通りやれば、それで終わりという単純なものではありません。もしそれで済むなら、アプリ開発は現在とは比べものにならないほど効率的なものとなり、自動化も促進されて、熟練した人間のアプリ開発者など必要なくなってしまうでしょう。

一通りプログラミングを完了させて実際に動かしてみたとき、思った通りに動作することはむしろ稀です。思ったような結果が得られないだけでなく、エラーが発生して、プログラムの動作が途中で止まってしまうこともあります。そこで、そのような場合にどこに問題があったのか調べる作業が必要になります。そして、問題点が分かったら、それを修正して再び動作させてみることになります。場合によっては、計画していた機能そのものを修正する必要性が生じたり、根本的な設計の誤りが発覚して、ゼロから考え直さなければならない部分が出てくることもあるでしょう。

いずれにしても、プログラミングの作業は直線的なものではなく、同様の手順を何度も繰り返しながら、よりよいものを開発していくための「サイクル」(円環状の手順)と考えることができます。

アプリ開発なら、企画の段階で計画や設計という手順が必要です。狭い意味のプログラミングには、ソースコードを記述し、編集する「コーディング」と呼ばれる手順や、アプリで使用する画像や音声などのリソースを作成、調整する作業もあるでしょう。そうして作成したアプリは、実際のデバイスや、パソコンの仮想環境上で動かしてみて、機能をテストしたり、動作速度を計測して評価したりすることが必要です。その結果分かった問題点を解決する「デバッグ」作業も不可欠です。そうした一連の作業が、円環状につながっているのです。

アプリ開発もPDCA

このような作業を並べてみると、アプリ開発は大きく4段階に分類できると考えることもできます。世の中には、何らかの製品開発や、その他の作業を、やはり4つのステップ「Plan」、「Do」、「Check」、「Action」に分けて、「PDCAのサイクル」を回して改善しようという考え方があります。細かな点は別にして、アプリ開発の作業にも、基本的には同じような考え方が適用できるものと考えられます。

「ソースコード」とは、プログラミング言語によって記述するプログラムそのものの。

なお、実際のアプリ開発では、例えばコーディング作業の途中で、プログラムを部分的に取り出してテストしたりすることもあり、かならずしもアプリ開発全体に関わる大きなサイクルで動いているわけではありません。しかし、その場合でも、局部的にはPDCAの小さなサイクルが回っていると考えることができます。

こうした途中での部分的なテストを「ユニットテスト」と呼ぶ。

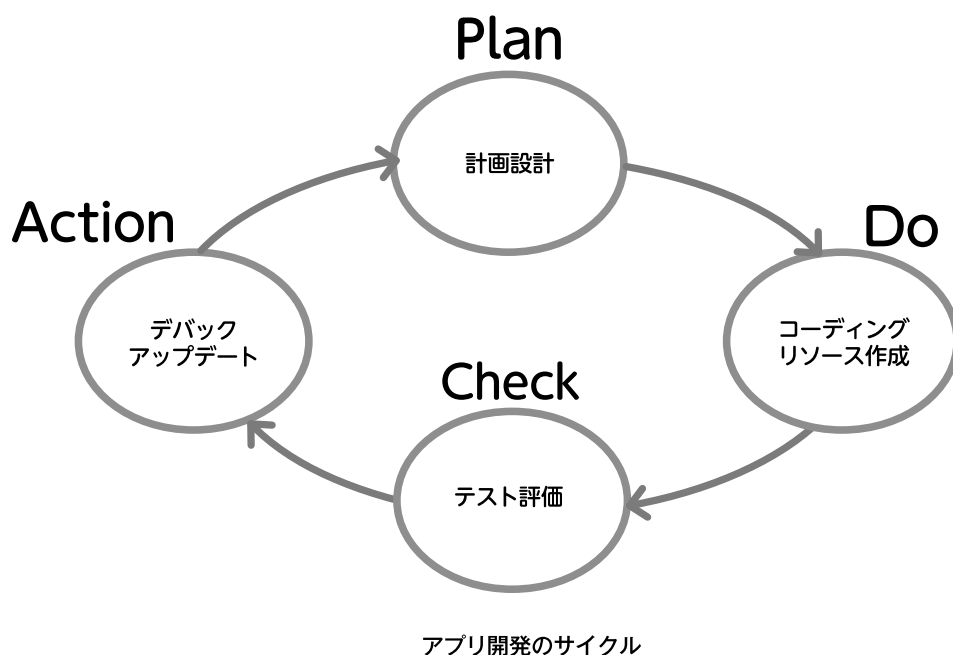


図4: アプリ開発の手順は、直線的なものではなく、サイクル状のものだと考えられる



1-1-5 プログラミングに使う道具

開発環境とは

現在では、「開発環境」と呼ばれる非常に多機能なツールを使ってアプリや、その他のプログラムを開発するのが普通になっています。プログラムの開発環境は、「IDE(Integrated Development Environment)」とも呼ばれます。IDEを直訳すれば、「統合開発環境」となります。これは複数の機能をひとまとめにした開発ツールという意味です。

IDEでは、プログラム開発に必要なさまざまな機能が、少なくとも見かけ上は1つにまとまっています。実際には多くのツールの寄せ集めであっても、多くの機能は1カ所から操作できるようになっていたり、少なくとも複数のツールが連携して動作するようになっています。それによって、アプリ開発という複雑な作業を完遂することができるのです。

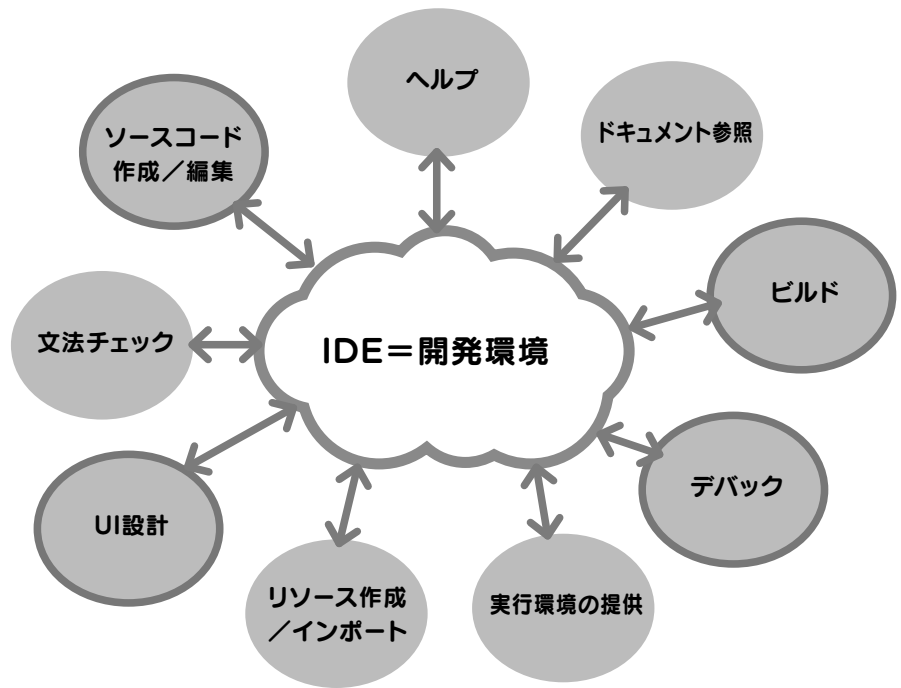
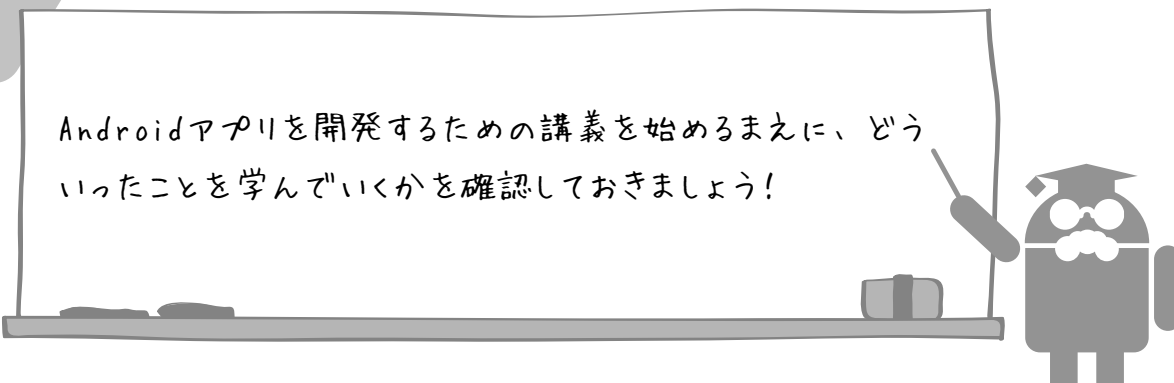


図5: 開発環境 (IDE) は、アプリ開発に必要なさまざまな作業のための機能を統合的に提供する

1-2 アプリ開発者養成講座の概要

著：柴田文彦



Androidアプリを開発するための講義を始めるまえに、どう
いったことを学んでいくかを確認しておきましょう！

1-2-1 コースの概要

ここでは、今回の「アプリ開発者養成講座」の全体の概要を簡単に説明します。全65回の講座は大きく7つの「Chapter」に分かれています。各Chapterのタイトルは以下の通りです

- Chapter 1:ファーストステップ
- Chapter 2:Androidの仕組み
- Chapter 3:基礎編
- Chapter 4:ステップアップ編
- Chapter 5:応用編
- Chapter 6:実習編
- Chapter 7:公開とブラッシュアップ

この中で、最も多くの講義時間が費やされるのは、Chapter 5の応用編です。ここでは、Androidのアプリ開発に関して、さまざまな角度から実践的なトピックを取り上げ、色々なタイプのAndroidアプリ開発に応用できるような技術が身に付けられるように企画してあります。

なお、テキストとしては、連続する同じテーマの複数の講義を1つの章にまとめているため、全部で26章構成になっています。印刷版のテキストでは、それを全部で10冊 (Volume)に分けて収録しています。



1-2-2 Android 開発者サイトの概要

Androidのアプリ開発に関して、最も信頼できる第1次情報は、Androidアプリ開発者向けの「Android Developer」(<http://developer.android.com/index.html>) サイトに集約されています。必要に応じてこのサイトの内容を参照してください。

プログラミングそのものに関する情報は、サイトトップの「Develop」タブの下に整理されていますが、他の「Design」、「Distribute」タブ以下にも、アプリ開発を総合的に考える際には重要な内容が含まれています。

どこに何が書かれているのかを知り、このサイトが身近なものに感じられるようになるまで、探訪しておきましょう。



1-2-3 独自の情報収集

アプリ開発の過程で生じた疑問や問題に、この講座の講義内容やテキストがすべて応えられるとは限りません。講義やテキストでは、時間的な制約も多く、一般的な、言い換えればあまり問題が生じない範囲の内容を扱う場合が多いからです。

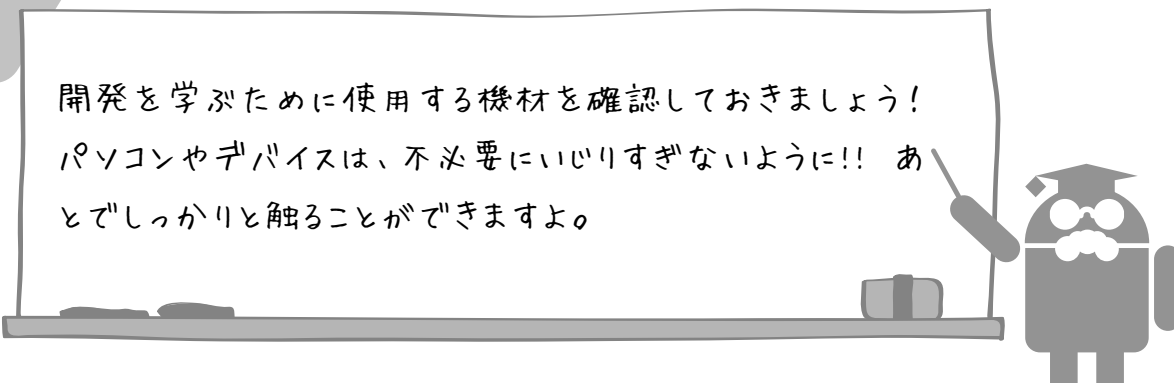
独自のプログラミング作業によって生じた疑問や問題は、なるべく自分で解決できるように試みてください。その際には、インターネット検索がかなりの助けになります。疑問や問題を詳しく英単語の羅列で表現して検索すれば、同じような問題、そしてその解決策が必ずと言って良いほど見つかります。

Android Developerサイトを読み込む場合もそうですが、インターネット上の情報を有効に活用するためにも、アプリ開発者にとって英語力は必須です。よく言われるように、日本の学校の英語教育が役に立たないということはまったくありません。学校英語で培った文章を読み書きする能力は、アプリ開発者にとって、聞く話す力よりもずっと有効に役立つ場面が多いはずです。会話能力もあった方が良いには違いありませんが、まずは英語の読解力を鍛えてください。そして、文法は間違っていないので、言いたいことを文章で表現する力を鍛えることも重要です。

もちろん、自分の力ではなかなか解決しない問題については当然ですが、簡単そうに見える問題についても、講義では気軽に講師陣に質問してください。しかし、いずれこの講座を卒業したら、自分だけが頼りという状態になることも多いでしょう。そのときに備え、問題解決能力を鍛えておくことは重要です。

1-3 実習機材の扱い

著：柴田文彦



開発を学ぶために使用する機材を確認しておきましょう！
パソコンやデバイスは、不必要にいじりすぎないように!! あ
とでしっかりと触ることができますよ。

1-3-1 実習環境の概要

今回の講座で使用するアプリ開発の実習機材は、Windows 8.1が搭載されたパソコンと、(現状では)Android 4.4.2が搭載されたスマートフォンです。

アプリ開発環境は、当然ながらWindows 8.1上で動作します。その開発環境には、Androidデバイスのエミュレーター機能もあり、実機のAndroidデバイスがなくても、パソコンの開発環境だけでアプリの動作が確認できるようになっています。ただし、実際のデバイス上の動作とは異なる部分があったり、実行速度が違ったりすることは避けられません。少なくとも他人に使ってもらうようなアプリを開発するには、実機上での動作確認、デバッグが不可欠です。その際には、USBケーブルでパソコンとデバイスを接続し、アプリをパソコンからデバイスに転送して動作させ、そのままデバッグもできるようになっています。

こうした開発環境や、実機での動作環境については第4章で詳しく解説します。それまでは、パソコンもデバイスも、あまりいじらずに待っててください。

1-3-2 パソコン環境の整備

すでに述べたように、パソコン上の開発環境の構築方法については、第4章で解説します。ただ、開発環境以前に、パソコンにインストールしておいていただきたいアプリがあります。それはウェブブラウザの「Google Chrome」です。Chromeは、「Chrome ブラウザ」サイト(<http://www.google.co.jp/intl/ja/chrome/browser/>)から無料でダウンロードできます。

Windows標準の「Internet Explorer」には、特にアプリ開発者にとっては、いろいろと使い難い面があるため、この講座ではChromeを「標準ブラウザ」とし

て推奨しています。

また、今後さまざまな場面で使用する機会が多いため、この講座の受講用として、普段使っているものとは別のGoogleアカウントを作成しておくことを強く推奨します。アカウント名の付け方については講座の中で説明します。

今後、講義中には、そのアカウントを設定しておくようにして、個人のアカウントは使用しない方が無難です。講義中には、講師陣や他の受講者が、パソコンやAndroidデバイスの画面を見る必要が生じることもあります。その際、お互いにプライバシーを気にしなくて済むようにするための提案です。



1-3-3 Android デバイスの設定

実習で使用するAndroidデバイスには、キャリアとしてドコモ用とau用があって、混在しています。基本的な機能は変わりませんが、キャリアによってホーム画面の機能やデザインが異なります。とくにドコモ版は、いろいろと付加機能が付いているため、受講者ごとに自由にホーム画面を設定すると、以後の操作が異ってくる場面がありそうです。

また、貸し出されたデバイスは、デバイスごとに来歴が異なるため、設定されている状態が異なる可能性が高くなっています。そこで、どちらのキャリア用のデバイスも、まず「工場出荷状態に初期化」を実行してください。この機能を使うには、「設定」の「ユーザーとバックアップ」グループにある「バックアップとリセット」画面を開き、「工場出荷状態に初期化」のボタンをタップします。

au版はとりあえずそれだけでOKです。

ドコモ版は、初期化してから立ち上がる際に、ホーム画面の種類を選択するようになっています。そこでは、「TouchWiz標準ホーム」を選択してください。これで、au版とほぼ同じような操作性が得られるようになります。なおドコモ版では、このホーム画面の選択は、「設定」の「個人設定」グループにある「ホーム切替」で、後から変更することもできます。

この設定に限らず、Androidデバイスは、自作のアプリをインストールする以外は、できるだけ標準設定で使ってください。あまり激しくカスタマイズすると、講義で作成するアプリの動作確認に支障をきたすことがないとも限りません。もしどうしてもカスタマイズする場合には、自己責任でお願いします。また、何をどうしたのかは、把握しておくようにして、もし問題が発生した場合には、関連が類推できるようにしておくといいでしょう。それは貴重なサンプルになるかもしれません。

また、AndroidデバイスにGoogleアカウントを設定する際には、プライベートなアカウントではなく、上で述べたような受講用のアカウントを設定することをお勧めします。