第3回 プログラミング入門

目次

イントロダクション

今回のトピック

- ソフトウェアとは
  - ソフトウェアの構成要素
  - ソフトウェアの分類
- プログラミング言語と実行方式
  - プログラミング言語とは
  - プログラミングパラダイム
  - プログラムの実行方式

本日の目標

- ソフトウェアについての理解を深める。
  - ソフトウェアとは
    - ソフトウェアの構成要素にはなにがあるか
    - ソフトウェアの分類はどのようなものか
  - プログラミング言語と実行方式
    - なぜプログラミング言語を使うのか
    - プログラムの実行方式にはどのようなものがあるか

ソフトウェアとは

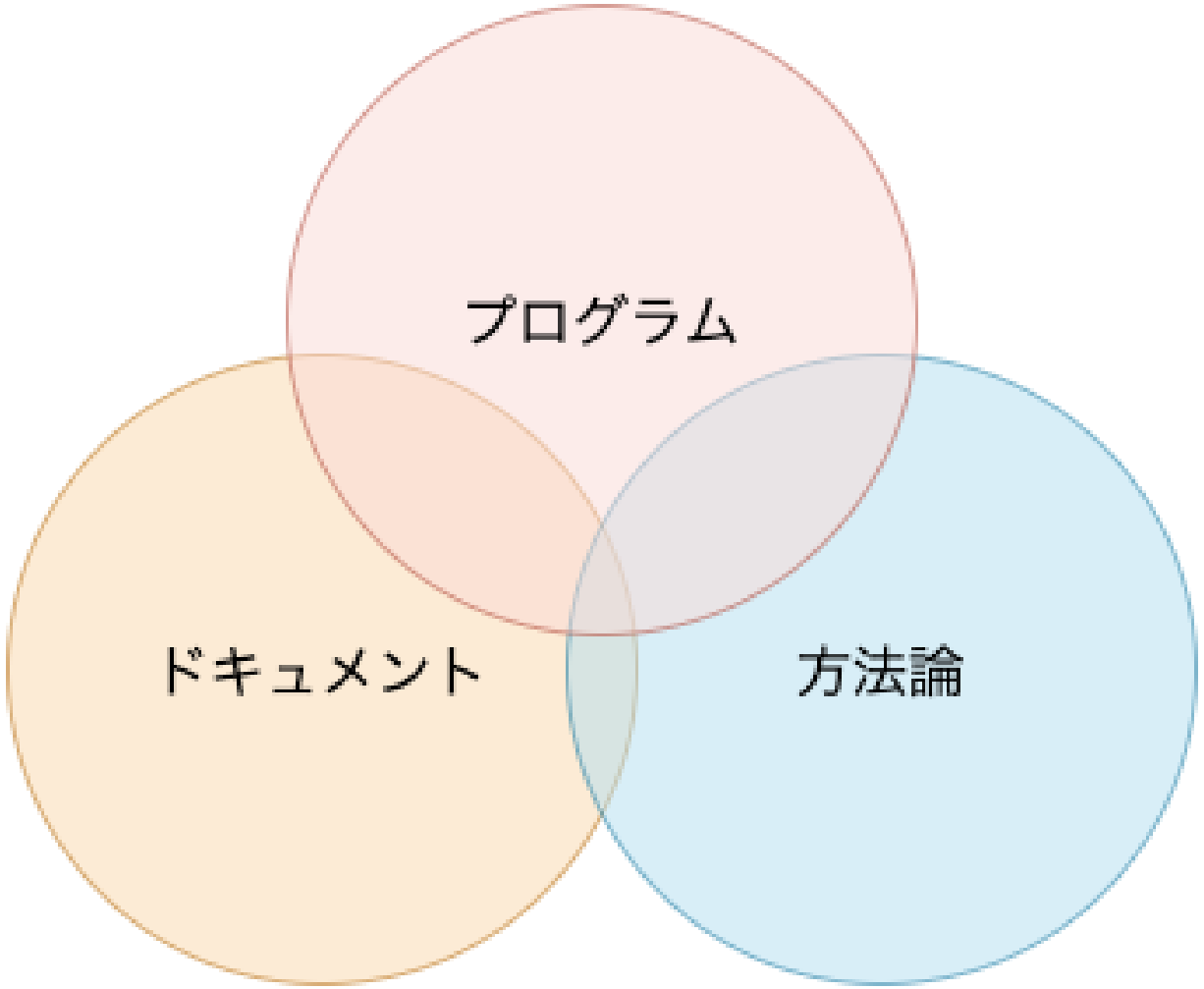
ソフトウェアとは

- ハードウェアに対比される用語
- ハードウェアとソフトウェアの特性の対比

ハードウェア	ソフトウェア
固いモノ	柔らかいモノ
金物	利用技術
物質的	抽象的
有形	無形
定量的	定性的

ソフトウェアの構成要素（1）

- プログラム
- ドキュメント
- 方法論



ソフトウェアの構成要素（2）

- 狭義のソフトウェア
  - プログラムのこと
  - プログラムとは、ハードウェア上で処理を実行するための命令を記述したもの。
- 実行環境とプログラム
  - 単体で実行可能なプログラム
  - スクリプト（ソースコード）など単体では動かないプログラムを動かすための環境
  - プログラムを実行するために必要なライブラリ・パッケージ

ソフトウェアの構成要素（３）

- ソフトウェアの開発および保守作業に必要とされる価値のある文書
  - 要件定義書
  - 各種仕様書
  - 運用マニュアル
  - 保守手順書

ソフトウェアの構成要素（4）

- ソフトウェアを開発するために必要とされる考え方や概念
  - 要求定義・システム設計・プログラム設計・テストなどの設計手法
  - アジャイル開発などのプロジェクトマネジメントの知識
- ソフトウェア開発における工学的なアプローチを **ソフトウェア工学** と呼びます。
  - 他の工学分野と性質が大きく異なり、定性的・統計的なアプローチ
    - ソフトウェアの定量化が難しいため
  - もちろん定量的なプローチもある

ソフトウェアの分類

- オペレーティングシステム
- ミドルウェア
- アプリケーションソフト

ソフトウェアの分類 - オペレーティングシステム

- コンピュータ全体の管理をするためのソフトウェア
  - 基本ソフトとも呼ばれる
  - WindowsやmacOS、Linuxなど
- ハードウェアを **抽象化** してユーザが扱いやすいように提供する
  - **抽象化とは類似したモノ・コトに対して共通の要素を捉える** こと
- OSの主な役割
  - ハードウェアの管理
  - ソフトウェアの管理
  - ファイルの管理

ソフトウェアの分類 - ミドルウェア

- アプリケーションソフトとOSの中間に位置するソフトウェア
  - アプリケーションに **共通する機能を取り出し、他のアプリケーションでも流用できる** ようにしたもの
- ミドルウェアの例
  - データベース管理システム
  - ウェブサーバー
  - メッセージ通信

ソフトウェアの分類 - アプリケーションソフト

- 我々が普段一般的に使うソフトウェア
- アナログでの作業をコンピュータ上に再現したソフトが多い
  - 計算・グラフ・チャート作成 → 表計算ソフト（Excel・Numbers）
  - 執筆 → 文書ソフト(Word・Pages)
  - OHP→プレゼンテーションソフト（PowerPoint・Keynote）
  - 郵便→メール(Outlook)

まとめ（1）

- ソフトウェアは、プログラム・方法論・ドキュメントから構成されている。
- ソフトウェアには、OS・ミドルウェア・アプリケーションソフトがある。

プログラミング言語と実行方式

プログラミング言語とは（1）

- プログラミングとは、**プログラムを作成する作業**のことを指します。
  - プログラムとは、コンピュータで処理をするための命令を記述したもの。
- コンピュータに処理をさせるには、CPUが理解できる形式（機械語）である必要があります。

プログラミング言語とは（2）

- プログラミングとは、**プログラムを作成する作業**のことを指します。
  - プログラムとは、コンピュータで処理をするための命令を記述したもの。
- コンピュータに処理をさせるには、CPUが理解できる形式（機械語）である必要があります。

プログラミング言語とは（3）

- コンピュータに処理をさせるには、CPUが理解できる必要があります。
  - しかし、人間がコンピュータに理解できる形で書き下すのは、至難の業。
- そこで、**プログラミング言語** と呼ばれる言語を使います。
  - コンピュータに命令をするための人工言語です。
  - 英語の文法に近い形で記述します。
- プログラミング言語で書かれたプログラムの原型を **ソースコード** と呼びます。

プログラミングパラダイム

- プログラミング言語がサポートしている考え方や概念を **プログラミングパラダイム** と呼びます。
- 代表的なプログラミングパラダイム
  - 手続き型言語
  - オブジェクト指向型
  - 関数型
  - 宣言型
- 現在主流の言語は複数のプログラミングパラダイムをサポートしています。

プログラムの実行方式

- プログラミング言語は人間が理解しやすい形式ではありますが、コンピュータはこのままでは理解できません。
- 実行するときに、そのまま理解させるのではなく、コンピュータが理解できる形式に変換する必要があります。

プログラムの実行方式の種類

- 代表的なプログラムの実行方式
  - コンパイル方式
  - インタプリタ方式
  - VM方式

コンパイル方式

- コンパイル型言語は、プログラム全体をまとめて翻訳する（実行ファイルへ変換する）。
    - この作業を **コンパイル** と呼びます。

コンパイル方式の特徴

- コンパイル型言語は、ソースコードをまとめて翻訳する（バイナリファイルへ変換する）。
    - 実行前に、文法エラーなどのエラー検出が可能
    - 実行するときは実行ファイルを起動するだけでよく、高速に実行できる。
    - コンパイルが必要なため、手間がかかる。
- コンパイル型言語には
    - C・C++・Rustなどがあります。

インタプリタ方式

- インタプリタ型言語は、ソースコードを一部ずつ読み込んで逐一翻訳します。
    - コンパイル型言語で必要だった **コンパイル** が不要です。

インタプリタ方式の特徴

- インタプリタ方式は、ソースコードを一部ずつ読み込んで逐一翻訳するため
  - すぐにソースコードを実行できる。
  - 逐一翻訳しているしているため、コンパイル型に比べると実行速度が遅い。
- インタプリタ型言語には
  - Python・PHP・JavaScript・Ruby・Haskell・Schemeなどがあります。

VM方式

- VM方式は、プログラムを実行するための仮想的な環境（仮想マシン）上で動かす方式です。
    - VM上で動作させるため、WindowsやmacOSなどOSの違いがあっても動かすことができる。
- コンパイル型と同様にコンパイルをして、**中間コード**を生成します。

VM方式の特徴

- VM方式はコンパイル方式とインタプリタ方式をあわせたように見えますが、コンパイル方式やインタプリタ方式それぞれの利点がありません。
    - インタプリタ方式と違い、 **コンパイルが必要**
    - コンパイル方式と違い、単体では動かせないため、**VMのインストールも必要**
- それでも大きなメリットがあり、広く利用されています。
    - **移植性の向上・開発の効率化**
- VM型言語には
    - C#・JAVA・Erlangなどがあります。

プログラムを作成する

- エディタでソースコードを編集する。
  - 5回目以降で実際にコードを書いていきます。
- アルゴリズムと呼ばれる **計算の手順** をプログラミング言語で書く。
  - 料理で言うところの **レシピ**
  - 一般的な問題（並べ替えや検索など）たくさんのアルゴリズムが考案されている。
  - 現実には自分で考えて書くことも多い。

まとめ（2）

- プログラミングとは、プログラムを作成する作業のこと。
  - プログラミングで使うための言語をプログラミング言語と呼ぶ。
- プログラムの実行方式には、**インタプリタ方式** と **コンパイル方式** と **VM方式** がある
  - インタプリタ方式は、ソースコードを逐一翻訳してながら実行する。
  - コンパイル型は、ソースコードを一コンパイルしてすぐに実行可能な状態に翻訳してから実行する。
  - VM方式は、ソースコードを中間コードに変換し、仮想マシン上で実行する。

次回

- ネットワークの仕組みについて理解を深めます。