第A章　さまざまなプログラミング言語

A.1　プログラミング言語の分類

プログラミング言語とは、プログラム（つまりコンピューターに与える指示）を書くための形式である。プログラミング言語は、まず**機械語**（および**アセンブリ言語**）と**高級言語**の 2 種類に大別することができる。

高級言語は、さらにパラダイムによって大まかに分類することができる。

（プログラミング）パラダイムとは —

＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿（空欄 *A.1.1*）のこと。

プログラミング言語には大きく次の 4 つのパラダイムに分けられる。

手続き型

＿＿＿＿＿＿＿＿＿＿＿＿（空欄 *A.1.2*）

命令型とも言う。

機械語に比較的近く、手続き（procedure）中心にプログラムを組み立てる。

代入文（変数の値の変更）を多用する。

関数型

＿＿＿＿＿＿＿＿＿＿＿＿＿＿（空欄 *A.1.3*）

関数（function）を、整数などのデータと同等に扱う。

リスト・木などの汎用的なデータ型を扱うのが得意である。

代入文を（ほとんど）使用しない。（値を変更するよりも、新しいデータをつくる。）

オブジェクト指向

＿＿＿＿＿＿＿＿＿＿＿＿＿＿（空欄 *A.1.4*）

オブジェクト（object）中心にプログラムを組み立てる。

GUI などのように、良く似た型が数多く使われる場合が得意である。

論理型

＿＿＿＿（空欄 *A.1.5*）, miniKanren, . . .

論理式（〜ならば〜）の集まりがプログラムになる。

このような分類は慣習的なもので、また排他的なものではない。一つの言語は 2 つ以上のパラダイムの影響を受けていることが普通である。

すべてのパラダイムをカバーし、どんな用途のプログラムでも記述できるような一つの言語があれば理想的だが、現実には、そのような誰もが納得するような

“**究極の**” 言語はこの世にまだ現れていない。そのため用途に応じて、いくつかのプログラミング言語を使い分ける必要がある。

また、型付け (typing) に関する考え方をもとに、プログラミング言語を分類する方法もある。一方の典型は、変数や関数の型宣言が必須で、コンパイル時に（静的に）型検査を行なってしまい、実行時には型エラーを出さないプログラミング言語であり、もう片方の典型は、変数や関数の型宣言が必要なく、実行前には型検査をせず、実行時に（動的に）型エラーを発行する言語である。前者は厳格（不自由）だが安全、後者は自由だが危険と表現することができる。**スクリプト言語**と呼ばれるプログラミング言語の多くは後者になる。この分類も、実はさまざまな中間的なアプローチが存在する。

また、強い型付け (strong typing) と弱い型付け (weak typing) という分類もあるが、人によって定義がいろいろあり、はっきりしない。強い型付けの定義の一つは、型の間違った操作は（静的なチェックまた動的なチェックのいずれにせよ）必ず防ぐことができる、というものである。弱い型付けの定義の一つは、型の不整合があったとき処理系が適当に型変換してしまう、データの解釈が完全にプログラマまかせである、というものである。（この定義によると、強い型付けと弱い型付けは相反する性質ではない。）

A.2　参考リンク

- TIOBE Index[1] さまざまな検索エンジンに基づく、プログラミング言語の人気ランキングである。
- GitHut[2] GitHub で使われているプログラミング言語のランキングである。
- RedMonk [3] GitHub と Stack Overflow の使用量に基づくプログラミング言語のランキングである。
- Programming Languages Influence Network[4] プログラミング言語が互いにどう影響を与えあっているかをグラフに表したものである。

A.3　主なプログラミング言語

A.3.1　Fortran

＿＿＿＿＿＿＿＿＿＿＿＿（空欄 A.3.1）の略とされる。1957 年、IBM の John Backus らによる。

- 最古の高級言語の一つである。手続き型言語に分類される。
- 現在でも ＿＿＿＿＿＿＿＿（空欄 A.3.2）に使われることが多い。

[1] https://www.tiobe.com/tiobe-index/
[2] http://githut.info/
[3] http://redmonk.com/
[4] http://exploring-data.com/vis/programming-languages-influence-network/

A.3.2 BASIC

_____（空欄 A.3.3）の略とされる。1964 年、ダートマス大 (Dartmouth College) の John G. Kemeny と Thomas E. Kurtz による。

- 系統的には Fortran から派生した手続き型言語である。
- Microsoft の商品がヒットしたため、パソコン上の主力言語の一つになった。（ただし、現在の Visual Basic は、発表当初の Basic とはまったく異なった言語になっている。）

A.3.3 COBOL

_____（空欄 A.3.4）の略とされる。1960 年頃、Grace Hopper らによる。

- _____（空欄 A.3.5）の手続き型言語である。

A.3.4 ALGOL

1958 年頃、**ALGO**rithmic **L**anguage の略とされる。

- Pascal や C など多くの言語に影響を与えた。これらの言語を総称して ALGOL 系と呼ぶことがある。
- 静的スコープ (lexical scope) や入れ子の関数 (nested function) などを初めて提供した。

A.3.5 Pascal

1968 年頃、スイスのチューリッヒ工科大学 (ETH Zürich) の Niklaus Wirth による。名前はフランスの哲学者 Blaise Pascal にちなむ。

- _____（空欄 A.3.6）に適した ALGOL 系の手続き型言語である。（`goto` 文を使わずに、プログラムを書くことができる。）
- 主に _____（空欄 A.3.7）に使われた。
- Delphi 言語は Pascal を拡張したものである。

A.3.6 C

1973 年頃、AT&T のベル (Bell) 研究所の Dennis Ritchie により、_____（空欄 A.3.8）というオペレーティングシステムを記述するために開発された。

- ALGOL 系の手続き型言語である。
- Pascal とは従兄弟のような関係にあたる。
- マシンレベルに近い記述が可能だが、Pascal と同様、構造化プログラミングに向く。

A.3.7　Lisp

1960 年頃、MIT の John McCarthy による。＿＿＿＿＿＿＿＿（空欄 A.3.9）の略といわれる。
格言 — “Lisp を知らない人は Lisp を再発明する。”

- ＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿（空欄 A.3.10）である。
- **関数型言語**に分類されるが手続き的な特徴も多く持つ。
- 対話的なプログラミング環境を持つ（通常、インタプリターにより実行される）
- メモリ管理を自動化している。（**ゴミ集め**）

A.3.8　Scheme

1970 年代前半、MIT の Guy L. Steele Jr. と Gerald Jay Sussman らによる。

- Lisp の方言の一つである。
- 単純で、しかも強力である。
- アプリケーションソフトの拡張用言語として、よく使用される。（GIMP の Script-Fu など）

A.3.9　ML

1970 年代、スコットランドのエディンバラ大学 (University of Edinburgh) の Robin Milner による。
＿＿＿＿＿＿＿＿（空欄 A.3.11）の略とされる。

- **関数型言語**に分類される。
- パターンマッチング・**多相型**・**型推論**などを特徴とする。
- OCaml, F#なども ML の方言になる。

A.3.10　Haskell

1987 年以降、開発された。名前は、数学者 Haskell B. Curry (1900–1982) にちなむ。Paul Hudak, John Hughes, Simon Peyton Jones, Philip Wadler など多くの研究者が設計・開発にかかわっている。

- **関数型言語** — 純粋な関数型言語で副作用を完全に排している。
- **モナド**による入出力を行なう。
- ＿＿＿＿＿＿（空欄 A.3.12）を採用し、**並列プログラミング**との親和性も期待されている。

A.3.11 Scala

2003 年 スイス連邦工科大学（ローザンヌ）(École Polytechnique Fédérale de Lausanne, EPFL) の Martin Odersky による。**Sca**lable **la**nguage の略とされる。

- JVM 上で動作するマルチパラダイムのプログラミング言語であるが、特に関数型の影響を受けている。
- Java との連携が容易である。
- 型推論があるため、多くの場所で型宣言を省略できるが、コンパイル時の型検査で型エラーを発見できる。

A.3.12 Swift

2014 年 アップル (Apple) 社の Chris Lattner らによる。swift はアマツバメ（雨燕）のことだが、「迅速」という意味もあるらしい。

- iOS や OS X 上のアプリケーションのための開発用言語として知られている。
- マルチパラダイムのプログラミング言語である。
- 型推論を持つ。
- LLVM コンパイラーフレームワーク (`http://llvm.org`) を使って実装されている。

A.3.13 Prolog

1970 年代はじめ、フランスのエクス=マルセイユ大学 (Aix-Marseille University) の Alain Colmerauer による。＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿（空欄 A.3.13）の略とされる。

- 一階述語論理に基づく**論理型言語**に分類される。
  （論理式の証明 = プログラムの実行）
- ユニフィケーション（単一化）、バックトラッキング（後戻り）などを特徴とする。

一時期、第五世代コンピュータープロジェクト（1982〜1992）の頃は論理型言語（Prolog に影響を受けた Guarded Horn Clauses や KL1 など）に大きな注目が集まった。今日、その熱狂は去ったが、再びもう少し注目を浴びても良い言語かもしれない。

A.3.14 miniKanren

2005 年頃、インディアナ大学 (Indiana University) の William E. Byrd らによる。名前は、日本語の “関連” から来ている。

- Prolog と同様に、論理型言語で入力と出力の引数を区別しない言語 (relational language) である。

- オリジナルの miniKanren は Scheme で実装されていて、Scheme をホストとする埋め込み型言語 (embedded language) である。つまり、独立した言語ではなく、Scheme の関数・マクロなどとして実装されている。
- 他に Haskell, Scala, Clujure, Ruby, OCaml などをホスト言語とする移植版も存在する。
- Clojure 版は **core.logic** と呼ばれている。Clojure 自体は JVM 上で動作する LISP 系の言語である。
- 機能を最小限に絞って、さらに他の言語に埋め込みやすくした、μKanren (microKanren) というヴァリエーションもある。μKanren はわずか 39 行の Scheme のコードで実装されている。

A.3.15　Smalltalk

1980 年、ゼロックス (Xerox) 社のパロ・アルト (Palo Alto) 研究所の Alan Kay による。英語の “small talk” は世間話、雑談、という意味があるが、プログラミング言語の名前としては Smalltalk で一語である。

- **オブジェクト指向言語**に分類される。
- 統合開発環境（IDE, Integrated Development Environment) のはしりとなった。

A.3.16　C++

1980 年代前半、AT&T のベル (Bell) 研究所の Bjarne Stroustrup による。

- C に _____（空欄 *A*.3.14）プログラミングのための仕組みを付け加えたものである。
- 仕様は巨大である。（電話帳並みと言われる。）

A.3.17　Java

_____（空欄 *A*.3.15）年、サン・マイクロシステムズ (Sun Microsystems) 社（現在はオラクル (Oracle) 社に吸収された）の James Gosling らによる。Java とはアメリカの口語で _____（空欄 *A*.3.16）のことである。

- C の文法に似せて設計された**オブジェクト指向言語**である。（ただし、C との互換性はない。）
- **JVM** と呼ばれる仮想機械上で動く（Pascal の P コードと同じ原理）ので、_____（空欄 *A*.3.17）
- WWW 上でプログラムをやりとりする仕組み（アプレットと呼ばれる）で普及した。
- WWW サーバー側でのプログラム（_____（空欄 *A*.3.18））としての利用も広まっている。

注: 下記の *JavaScript* とはまったく別の言語なので混同しないこと。

A.3.18 C#

2000 年頃、マイクロソフト (Microsoft) 社の Anders Hejlsberg らによる。

- 同社の .NET Framework の中心となる言語である。
- Java や Delphi の影響を強く受けている。
- C sharp と発音されるが、実際にはシャープ (♯) ではなくて、ナンバーサイン (#) を使う。

A.3.19 JavaScript（ECMAScript）

1994 年ネットスケープ・コミュニケーションズ (Netscape Communications) 社（現在は AOL に吸収され、AOL は Verizon Communications の子会社になっている）の Brendan Eich による。（ただし最初の名前は LiveScript だった。）

- _____[(空欄 A.3.19)] に基づくオブジェクト指向言語である。
- HTML などに埋め込まれるスクリプト言語としてよく使用される。
- Java に（つまり C に）似た文法を持つ。（しかし Java とは全く別の言語である。）
- 関数型言語的なプログラミングも可能である。

マイクロソフト社の JScript も JavaScript に類似の言語である。複数の似た規格が乱立すると混乱が起きるため、ECMA という機関で言語規格の標準化が行われた。ここで決められた言語を ECMAScript という。つまり JavaScript も JScript も ECMAScript の実装の一つ（あるいは規格の拡張の一つ）という位置付けになる。

A.3.20 PHP

1995 年 Rasmus Lerdorf による。現在では **P**HP: **H**ypertext **P**reprocessor の略（再帰的頭字語, recursive acronym）とされるが、当初は **P**ersonal **H**ome **P**age から名付けられたらしい。

- Web サーバーで動的に HTML を生成するのに使われる**スクリプト言語**である。
- C 言語に似た文法を持っている。
- HTML に埋め込んだ形でプログラムを書くことができる。

A.3.21 Perl

1987 年 Larry Wall による。最初は Pearl（真珠）と名付けたかったが、同名の言語が既にあったため、綴りを一部変えた。

- CGI やテキスト処理などに広く使われる**スクリプト言語**である。
- 正規表現を強力にサポートする。
- モットー (?): “There’s more than one way to do it.”

A.3.22　Python

1990 年 Guido van Rossum による。名前はイギリスのコメディーグループ Monty Python に由来する。

- マルチパラダイムのスクリプト言語である。
- Web プログラミングで人気がある。
- モットー: “There should be one—and preferably only one–obvious way to do it.”

A.3.23　Ruby

1995 年 まつもとゆきひろ（松本行弘）による。名前は宝石のルビーから取られた。

- マルチパラダイムのスクリプト言語である。
- Web アプリケーションフレームワークの _____（空欄 *A*.3.20）でブレイクした。

A.3.24　Lua

1993 年、ブラジルの Pontifical Catholic University of Rio de Janeiro の Tecgraf グループによる。Lua はポルトガル語で月を意味する。

- 他のプログラムに組み込まれることを考えた、軽量のスクリプト言語である。
- ゲームプログラムのための拡張用言語として人気が出た。

A.3.25　Go

2009 年、グーグル (Google) の Robert Griesemer, Rob Pike, Ken Thompson らによる。

- Golang と呼ばれることもある。
- 大規模でネットワーク機能・並行性を持つアプリケーションなどに適している。
- 静的に型付けされた言語であるが、型推論を持つ。

A.3.26 Rust

2010 年、Graydon Hoare による。その後、モジラ (Mozilla) 社の支援により開発が続けられている。

- 多くのモダンなプログラミング言語と異なり、動的なメモリー管理であるゴミ集め (garbage collection, GC) を採用せず、コンパイル時にメモリーの安全性をチェックすることに重点をおいている。
- GC は実行時にメモリーを使い果たしたときに使われなくなったメモリーを解法する。一方、Rust はコンパイル時にメモリーを解放することができるかをチェックする。
- これまで C 言語が担っていた、システム記述の分野に適した言語を目指している。

A.3.27 Kotlin

2011 年、ジェットブレインズ (JetBrains) 社の Andrey Breslav, Dmitry Jemerov らによる。

- JVM で動作し、JavaScript にコンパイルすることも可能になっている。
- 型推論など現代的な特徴を取り入れ、“より良い” Java を目指して設計されている。

問 A.3.1 このプリントで紹介した以外のプログラミング言語で重要・あるいは将来有望と思う言語を 2 つ以上取り上げ、プログラミングパラダイム・型付けなどを含めて、その特徴を、自分が重要または有望と感じた理由とともにまとめよ。情報の出典を明記せよ。