2020 年度 卒業論文

日本語プログラミング言語は日本人にとって可読性が高いが、なぜあまり普及していないのか

学籍番号： 17 M3247

氏　　名： 新村 裕太

要旨

プログラミング教育が 2020 年度から小学校、2021 年度から中学校、2022 年度から高校で必修化されることを受け、近年は様々なプログラミングツールが普及しつつある。そして、プログラミング言語の中でも特に有名で使用率が高いものには、「C/C++、Python、JavaScript、SQL、C#、Java、VBA、HTML/CSS」などがある。しかし、我々が使い慣れた母国語で記述できる「日本語プログラミング言語」は、まだまだ認知度は低く、使用率も低いため、とても普及しているとは言えない。日本語プログラミング言語の中では有名なものに、「なでしこ、プロデル、ドリトル」というものがある。なでしこは、元々、前身のひまわりという言語の頃から事務の自動化を目標に開発されている。そのため、ファイル処理、画像処理、ネットワーク、Word/Excel 連携、文字列処理など、日々の定型作業を自動化するための便利な命令を 1000 以上備えている。このように、作業の自動化をはじめ、趣味、教育、仕事など幅広く利用することができる。プロデルは、「読めば すぐ分かるプログラム、実用的な用途、速い、プラグイン」の 4 つの特徴をかかげている。また、日本語プログラミング言語の中では珍しく、教育分野での使用のためには作られておらず、汎用的なソフトウェア開発を目標として作られた。ドリトルは、なでしこやプロデルと比べ、より教育用途に特化し、教育現場を中心に普及が進んでいるのが特徴である。リファレンスやマニュアルも主には授業用テキストや教師向けの授業資料という形で整備され、配布されている。

本研究では、日本語プログラミング言語の特性を把握するとともに、これらの日本語プログラミング言語の普及を妨げている要因について、英語記述のプログラミング言語との比較や、アンケート調査、考察をし、普及方法や新たな活用方法の模索をすることを目的とする。

日本語プログラミング言語は、教育分野での普及、業務分野での普及ともに、利用するのに適した環境は用意できているように見えるが、学習する日本語プログラミング言語の構文が「ミスマッチ」しているという事考えられるのである。そして、そのような「ミスマッチ」を起こさず、日本語プログラミング言語を普及させていくには、初等教育では 2020 度より開始したプログラミング教育を受け、プログラミング的思考が育まれた状態である、中等教育以降で日本語プログラミング言語を活用することが重要であると考える。

目次

第1章　はじめに

「プログラミング教育が 2020 年度から小学校、2021 年度から中学校、2022 年度から高校で必修化されることを受け、近年は様々なプログラミングツールが普及しつつある。そして、プログラミング言語の中でも特に有名で使用率が高いものには、「C/C++、Python、JavaScript、SQL、C#、Java、VBA、HTML/CSS」などがあり、これらは主に英語で記述する形である。一方で、我々が使い慣れた母国語で記述できる「日本語プログラミング言語」も存在し、「なでしこ、プロデル、ドリトル」というものが比較的有名である。しかし、プログラミング言語全体からみればまだまだ認知度は低く、使用率も低いため、とても普及しているとは言えない。

本研究では、日本語プログラミング言語の特性を把握するとともに、これらの日本語プログラミング言語の普及を妨げている要因について、英語記述のプログラミング言語との比較や、アンケート調査、考察をし、普及方法や新たな活用方法の模索をすることを目的とする。

本論文の構成は次の通りである。まず第 1 章で、本論文の目的を説明する。第 2 章では、プログラミング言語の定義、言語処理系、プログラミング言語の種類について述べる。第 3 章では、日本語プログラミング言語について、なでしこ、プロデル、ドリトルの 3 つの紹介をし、日本語プログラミング言語の現状を述べる。第 4 章では、日本語プログラミング言語はなぜあまり普及していないのかを、可読性や必要性の観点から考え、プログラミング教育での活用についても論じる。第 5 章では、まとめと今後の課題について述べる。

第2章　プログラミング言語について

本章では、プログラミング言語について論じる。第 2.1 節でプログラミング言語について定義し、第 2.2 節で言語処理系について、第 2.3 節でプログラミング言語の種類について、という順に論じる。

2.1　プログラミング言語とは

まず初めに言語とは、一定のルールのもとで文字・音声を組み合わせて意味を表すもの、あるいはその体系のことである。そして、プログラミングとは、コンピュータへの指示書となるプログラムを作成することである。つまり、プログラミング言語とは、コンピュータに指示をするため、文字で意味を示すことやその規則といえる。プログラミング言語の存在意義は、人間が直接扱うには難しい機械語に代わって、より人間が扱いやすい形を提供することにある。機械語とは、コンピュータのマイクロプロセッサ（CPU/MPU）が直接解釈・実行できる命令コードの体系のことであり、0 と 1 を並べたビット列として表され、人間が直に読み書きしやすい形式ではない。そして、コンピュータが直接理解し実行することのできる言葉は、そのコンピュータの種類に固有の機械語だけである。したがって、最終的には機械語を使ってコンピュータが行うべき作業・計算を指示しなければならない。

機械語命令と一対一に対応するアセンブリ言語もプログラミング言語の一つではあるが、一般には、C 言語や Java などもっと人間に解り易いプログラミング言語が多く使われる。アセンブリ言語は低水準言語、C 言語や Java などは高水準言語（高級言語）と呼ばれ、機械寄りの言語が低水準言語、人間寄りの言語が高水準言語ということになる（図 2-1-1）。

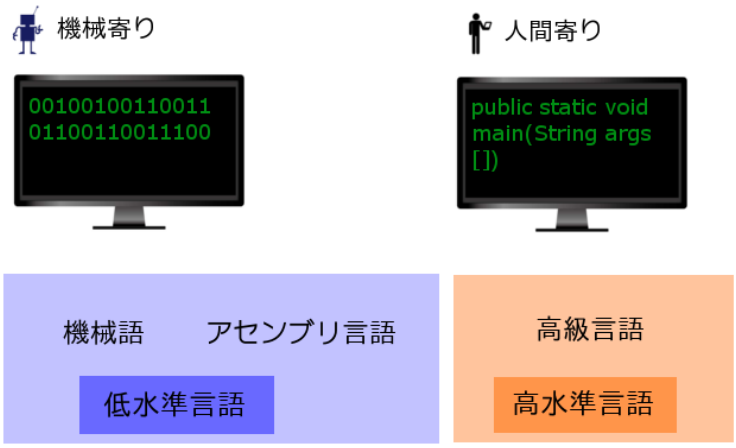

図 2-1-1 「低水準言語と高水準言語」
（出展：Mecha log「「プログラミング」 初心者でも理解できる！基礎中の基礎」
（https://mechalog.com/programming-beginner））

高水準言語で書かれたプログラムを機械語プログラムに変換するものをコンパイラと呼ぶ。また、機械語プログラムに変換するのではなく、高水準プログラミング言語で書かれたソースコード（プログラム）を逐次解釈実行するものをインタプリタと呼ぶ。プログラムの実行に主としてインタプリタを用いるプログラミング言語をインタプリタ言語（またはスクリプト言語）と呼び、Perl、JavaScript、VBScript などがよく知られている。そして、コンパイラを用いるプログラミング言語をコンパイラ言語と呼び、Java、C、C+、C#などがよく知られている（図 2-1-2）。

図 2-1-2 「コンパイラとインタプリタ」
（出展：じゃぱざむ「コンパイラとインタプリタの違いは？言語の違いを分かりやすく解説！」（https://jpazamu.com/interpreter-compiler/））

また、ハイパーテキストを記述するための言語である HTML（HyperText Markup Language）は、プログラミング言語だと思われがちだが、マークアップ言語の一種である。マークアップ言語とは、データ中に特定の記法を用いて何らかの情報を埋め込むためのものである。テキストデータ中に特定の記号で囲まれたタグと呼ばれる表記を用いて構造や見栄えなどを記述するものがよく知られるが、バイナリデータ中に埋め込むものなど、様々な種類がある。そして、このようなコンピュータ上で使用されている言語のことを、コンピュータ言語と呼ぶ。すなわち、コンピュータ言語の一種にプログラミング言語や、マークアップ言語などが存在しているということである（図 2-1-3）。

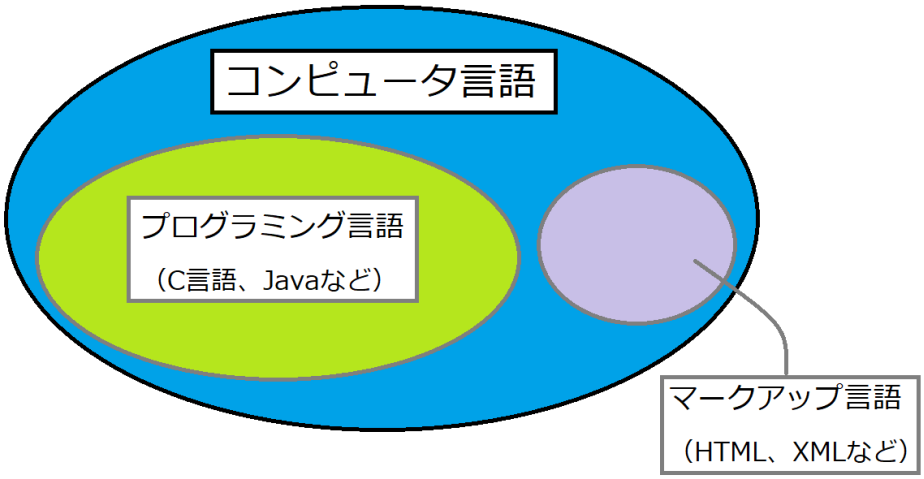

図 2-1-3 「コンピュータ言語」
（出展：マイナビクリエイター「マークアップ言語とは」（https://mynavi-creator.jp/blog/article/what-is-mark-up-language）を参考に作成）

我々が普段話す日本語のような日常的に使われている言語は、自然言語といわれているが、プログラミング言語はそれとは別で、人の手によって作られた特殊な規則にもとづいた言語である、形式言語というものである。AI などを用いてその自然言語を機械で処理することを自然言語処理というが、開発には未だに難しい点も多く、中でも自然言語には言葉の「あいまいさ」「意味の重複」が含まれていることから、感情や文脈などの解釈違いにより、勘違いをする場面が多々ある。

例としては、「い　ぬ　が　ね　こ　を　う　ん　だ　よ」というものがある。解釈の仕方については、①のように「いぬ　が　ねこ　を　うんだよ」すなわち、「犬が猫を産んだよ」という仕方と、②のように「いぬ　がね　こ　を　うんだよ」すなわち、「犬がね　子を産んだよ」という仕方ができる。人間は犬が猫を生むわけがないので②だとわかるのに比べ、コンピュータの自然言語処理である予測変換などでは①が出てしまう。このような「あいまいさ」や「意味の重複」があってはコンピュータには理解がしにくいため、そういったものをすべて取り除き、プログラミング言語のような形式言語は、厳格な文法規則にもとづいている。そのため、自然言語とは違い、コンピュータでの処理がしやすい（図 2-1-4）。

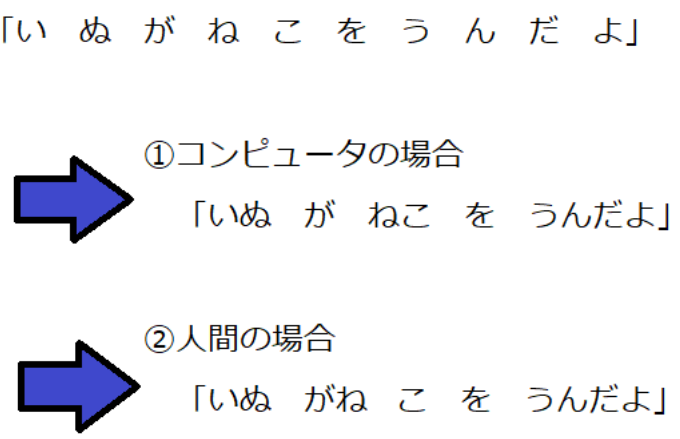

図 2-1-4 「いぬがねこをうんだよ」
（出展：AI:人工知能特化型メディア「自然言語処理（NLP）とは | 仕組み・活用例・今後の課題」（https://ledge.ai/nlp/）を参考に作成）

2.2 言語処理系について

言語処理系とは、プログラミング言語で記述されたプログラムを計算機上で実行するためのソフトウェアである。そのための構成として、インタプリタとコンパイラという 2 つの構成方法がある。第 2.1 節でも解説したが、もう少し分かりやすく言うと、インタプリタとは、言語を意味解析しながら、その意味する動作を実行するものである（図 2-2-1）。

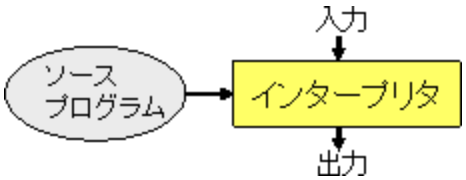

図 2-2-1 「インタプリタ（インタープリタ）」
（出展：HPCS Lab - 筑波大学システム情報工学研究科
「言語処理系とは」（http://www.hpcs.cs.tsukuba.ac.jp/~msato/lecture-note/comp-lecture/note1.html））

コンパイラとは、言語を他の言語（機械語や中間言語）に変換し、その言語のプログラムを計算機上で実行させるものである（図 2-2-2）。

元のプログラムをソースプログラム、翻訳の結果と得られるプログラムをオブジェクトプログラムと呼ぶ。機械語で直接、計算機上で実行できるプログラムを実行プログラムと呼ぶ。オブジェクトプログラムがアセンブリプログラムの場合には、アセンブラにより機械語に翻訳されて、実行プログラムを得る。他の言語の場合には、オブジェクトプログラムの言語のコンパイラでコンパイルすることにより、実行プログラムが得られる。仮想マシンコードの場合には、オブジェクトコードはその仮想マシンにより、インタプリトされて実行される。

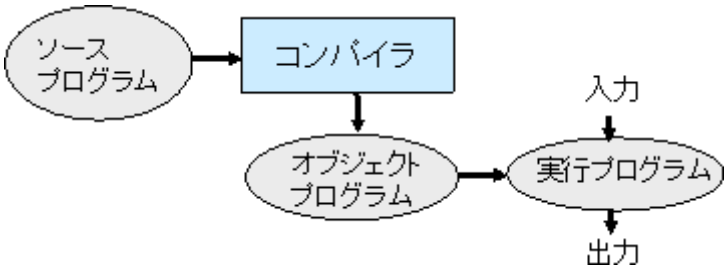

図 2-2-2 「コンパイラ」
（出展：HPCS Lab - 筑波大学システム情報工学研究科
「言語処理系とは」（http://www.hpcs.cs.tsukuba.ac.jp/~msato/lecture-note/comp-lecture/note1.html））

言語処理系の基本構成（インタプリタとコンパイラの基本構成）を分けると 5 つあり、インタプリタとコンパイラには多くの共通部分がある（図 2-2-3）。

1. 字句解析：文字列を言語の要素（トークン）の列に分解する。
2. 構文解析：トークン列を意味反映した構造（木構造）に変換する。
3. 意味解析：木構文の意味を解析する。インタプリタでは、ここで意味を解析し、それに対応した動作を行う。コンパイラでは、この段階で中間コード（内部的なコード）に変換する。
4. 最適化：中間コードを変形して、効率のよいプログラムに変換する。
5. コード生成：内部的なコードをオブジェクトプログラムの言語に変換し、出力する。例えば、ここで、中間コードよりターゲットの計算機のアセンブリ言語に変換する。

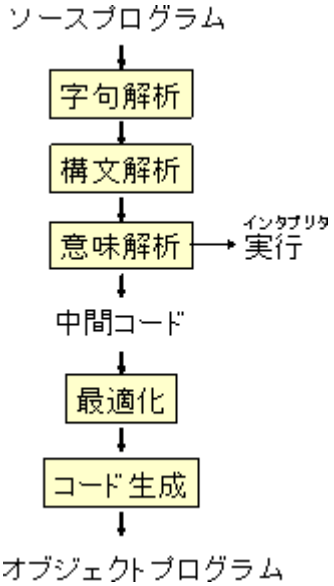

図 2-2-3 「言語処理系の基本構成」

（出展：HPCS Lab - 筑波大学システム情報工学研究科

「言語処理系とは」（http://www.hpcs.cs.tsukuba.ac.jp/~msato/lecture-note/comp-lecture/note1.html））

コンパイラの性能とは、如何に効率のよいオブジェクトコードを出力できるかであり、最適化でどのような変換ができるかによる。インタプリタでは、プログラムを実行するたびに、字句解析、構文解析を行うため、実行速度はコンパイラの方が高速である。もちろん、機械語に翻訳するコンパイラの場合には直接機械語で実行されるために高速であるが、コンパイラでは中間コードでやるべき操作の全体を解析することができるため、高速化が可能である。また、中間言語として、都合のよい中間コードを用いると、いろいろな言語から中間言語への変換プログラムを作ることで、それぞれの言語に対応したコンパイラを作ることができる（図 2-2-4）。

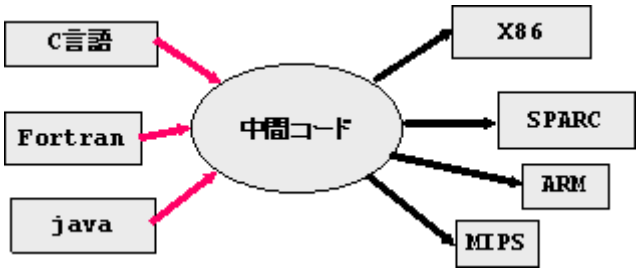

図 2-2-4 「中間コード」

（出展：HPCS Lab - 筑波大学システム情報工学研究科

「言語処理系とは」（http://www.hpcs.cs.tsukuba.ac.jp/~msato/lecture-note/comp-lecture/note1.html））

2.3　プログラミング言語の種類

現在、プログラミング言語は Java や PHP、C++、JavaScript などのメジャーなものから、マイナーなものまで合わせると 200 種類以上あるといわれている。その中には企業が作成したものから個人が作成したものまで数多く存在し、文字を書くものだけではなく、視覚的に表現されたブロックを組み合わせるものや画像データを用いる言語まである。このように複数のプログラミング言語が存在する理由としては大きく分けて 3 つの要因があると考える。

第一に、プログラミング言語ごとに得意分野が異なるという点があげられる。Web 開発であれば、PHP や JavaScript、Ruby、Python などのスクリプト言語がまず候補にあがる。しかし、PHP などだけ覚えておけば、Web 開発以外でもすべてのシーンでも使えるかというとそうではなく、Web 開発に特化した PHP では、例えば iPhone や iPad で動くアプリを作ることはできない。iPhone や iPad 用のアプリを作るには、Objective-C か Swift が主要となっている。また、Android 用のアプリを作るなら、Java や Java との互換性が高い Kotlin であり、同じスマホのアプリでも、スマホゲーム開発では、C#を用いて Unity、C++を用いて UnrealEngine や Cocos2d-x などが主要となっている。そして、一般的なプログラミング言語とは少し違うが、Web サイトを表示するには、HTML という専用のマークアップ言語を使用してページ記述をし、データベースを操作する場合には、SQL というデータベースを操作する専用の言語を使用する。さらに、コンピュータの種類によっても使用する言語が異なってくる場合もある。このように、プログラミング言語ごとに、専門分野や得意分野が異なり、何をどのような環境で作るかに応じて、プログラミング言語を変える必要がある。一応、上記であげたプログラミング言語を含め、多くのプログラミング言語では、複数の用途に使えるようにはなってはいるが、得意分野と苦手分野が明確にある。そのため、複数の言語を習得し、使い分ける必要があるため、複数のプログラミング言語が生まれたと考える。

第二に、プログラミング言語の進化という点があげられる。IT の世界は常に進化していっているイメージが強いが、それはプログラミング言語においてもそうだと考えられる。プログラミング言語が使われていく中で、その言語では解決できない、もしくはしにくい課題が見つかっていき、それに合わせて様々な新しいプログラミング言語が開発されていき、その言語が主流となっていく場合もある。例えば、かつて Web サーバでは Perl が標準的なプログラミング言語であったが、Web サーバに特化しており、性能や書きやすさで優れた PHP が人気になり、一気に Web サーバ分野での使用率を抜かしていった。また、最近では、Java と Kotlin の間でも同様のことが起ころうとしている。Kotlin が Java と比較して優れているといわれる点は、コンパクトなコードによる安定性とセキュリティの向上、安全性を重視したコンパイラ、豊富な IDE の選択肢、生産性向上の可能性などがある。しかし、それ以上に重大な点は、Kotlin と Java の互換性が 100%なことである。つまり、Java から Kotlin

に移行するのに余分な作業が必要なく、Java のフレームワークも利用できるため、移行リスクを最小限にして、より利便性の高い言語を使用できるのである。また、2017 年には、Kotlin が Android 公式言語となり、2019 年には、Android 上で Kotlin ファーストの推進を発表し、Java 以上に Android 上で優先されるプログラミング言語となった。このような事が要因となり、Kotlin は近年急速にユーザーを伸ばし、Java に取って代わろうとしている。このように新しいプログラミング言語を作ることによって、これまで時間のかかっていた処理を簡単に実現できるようになるなど、効率性、利便性の向上を見込めることもあるため、プログラミング言語の種類が増加していくのではないかと考える。

第三に、プログラミング言語の多様性という点があげられる。プログラミング言語は誰でも自由に作れ、その新しく作られる言語は、第二でも述べたように何かしらの課題解決のために生まれている。そのため、同様の用途のプログラミング言語でどうしであっても、完全に近い進化であるかどうかに関わらず、劣っているところもあれば、優れているところもあるといった状態で並立して生き残っている。それぞれ独自の事情やセールスポイントがあるからこそ、並立していても何かしらの使い分けができ、共存が可能になっているという事である。さらに、IT の分野では一般的に独占が起きづらいといわれており、常に多様な選択肢が存在する。例えば、スマートフォン一つとっても、Apple の iOS や、Google の Android など複数の OS が存在し、どの OS にも良い点があり、一概にどちらが優れているとは言えない。そして、それぞれの OS に合わせた言語も発表していき、プログラミング言語の多様性も生まれているのである。その他にも、用途的に並立した言語どうしでも個人の書き方やプラットフォームなどに関する好みによって分かれてくると考える。例えば、Java と C#はどちらもデスクトップアプリを開発したり、Web アプリを開発したりすることが可能である。さらに、静的型付けを行うオブジェクト指向のプログラミング言語という点や、一度は中間形式に変換される点など、細かい点でも似ているところがある。しかし、その他の書き方やプラットフォームなどでは異なっているところも多く、Java と C#のそれぞれに数多くの支持者がおり、現在も開発継続されている。このように、用途的に並立しても消えずに存在することが多いという、プログラミング言語の多様性も、複数のプログラミング言語が存在する理由として考える。

上記であげたプログラミング言語以外のものとして、本研究の題材でもある、「日本語プログラミング言語」という種類がある。第 1 章でも述べたように、「日本語プログラミング言語」は、まだまだ認知度は低く、使用率も低いため、とても普及しているとは言えない。日本語プログラミング言語の中で有名なものに、「なでしこ、プロデル、ドリトル」というものがある。この 3 つのような日本語プログラミング言語どうしでも、様々な用途や特徴があるが、その中でもほとんどの日本語プログラミング言語に共通していて、最も使用率の高い用途が、プログラミング初心者や、学生向けのプログラミング教育での使用である。日本人にとっての最大の利点は、日本語で読み書きできるという点であり、それ故に、プログラミングの入門用の言語として期待できるのである。また、日本語プログラミング言語の先

駆けとなった Mind や、熱心に開発が続く、なでしこ、プロデル、ドリトルはいずれもトイ言語などではなく、十分に本格的な言語であり、教育分野以外でも、多くの業務に使用可能となっている。詳しくは第 3 章で論じる。

第3章　日本語プログラミング言語について

本章では、第 3.1 節で日本語プログラミング言語について、第 3.1.1 項でなでしこ、第 3.1.2 項でプロデル、第 3.1.3 項でドリトル、第 3.2 節で日本語プログラミング言語の現状、という順に論じる。

3.1　日本語プログラミング言語について

日本語プログラミング言語とは、日本語風のプログラミング言語のことである。ひらがなやカタカナ、漢字などを使って、日本語でソースコードを書き、アプリケーション開発を行うことができる。

日本語プログラミング言語の歴史は意外と古く、1982 年に発表されたぴゅう太には、日本語 BASIC (G-BASIC)が搭載されていた。1983 年には、ワープロ感覚で日本語をもちいてプログラミングができればという考えで開発された和漢が開発された。1985 年には、Forth の語順が日本語に近いことに注目した Mind が開発された。そして、2000 年には、TTSneo、プロデル、ひまわり、ドリトルなど、インタプリタ型の日本語プログラミング言語が次々とフリーソフトとして発表された。

日本語プログラミング言語の最大の利点としては、我々が使い慣れた母国語である日本語で記述できるという事にある。しかし、日本語プログラミング言語は、まだまだ認知度は低く、使用率も低いため、とても普及しているとは言えない。そのため、「日本語プログラミング言語は日本人にとって可読性が高いが、なぜあまり普及していないのか」について考える必要がある。そのためにまず、日本語プログラミング言語の中でも有名である、「なでしこ、プロデル、ドリトル」について、特徴や使用用途を述べていく。

3.1.1　なでしこ

ホームページでの紹介には、「「なでしこ 3」は、日本語をベースに開発されたプログラミング言語です。主に Web ブラウザで動かすことを目的に開発されています。ブログや Web サイトに組み込んで使えます。また、OS や環境を問わず、PC/スマホ/タブレットとさまざまな環境で動きます。作業の自動化をはじめ、趣味、教育、仕事などに利用できます。」などとある[14]。

このようになでしこは、日本語で記述をするため、日本人のプログラミング初心者や、学生、英語の苦手な人などに向けて作られている。しかし、元々、前身のひまわりという言語の頃から事務の自動化を目標に開発されている。そのため、ファイル処理、画像処理、ネットワーク、Word/Excel 連携、文字列処理など、日々の定型作業を自動化するための便利な命令を 1000 以上備えている。そして、なでしこは 2004 年に公開されて以降、2017 年には Web ブラウザでも動かせる v3（CoffeeScript や TypeScript のように、JavaScript に変換

されて実行される広義の altJS) が公開されるなど、その後も活発に開発が続けられている。

3.1.2　プロデル

ホームページでの紹介には、「プロデルは、日本語でプログラムが書けるプログラミング言語です。次のような特徴を持っています。プロデルの文法は日本語文のように書けるように設計されています。プログラム例文を見ればすぐに何をしているか理解できます。バッチ処理や画像描画、GUI 部品の機能が用意されています。Excel, CSV 形式操作、ODBC 接続もできます。ちょっとしたツールやゲーム作りも可能です。TTSneo よりも高速に動くようになりました。大量のデータ処理も短時間で実行できます。また MSIL コンパイラも搭載していますので、コンパイルで更なる高速化も可能です。プラグインで FeliCa や Twitter との連携機能を利用できます。また C#を使って独自のプラグインを作ることもできます。」などとある[15]。

このようにプロデルは、「読めば すぐ分かるプログラム、実用的な用途、速い、プラグイン」の 4 つを特徴としてあげられる。そのため、日本語プログラミング言語の中では珍しく、教育分野での使用のためには作られておらず、汎用的なソフトウェア開発を目標としている。また、プロデルにはプロデル Web サーバという、プロデルで開発された Web アプリケーションを動作させるための専用 Web サーバが用意されている。プロデル Web サーバを使うことで、プロデルで Web アプリケーション開発を実施や、サービスを公開することが可能である。プロデルは同じく日本語プログラミング言語の TTSneo を前身とし、仕様を一新、アップグレードする形で公開された言語。TTSneo の公開は 2002 年、さらに、TTSneo の前身である TTS が公開されたのは 2000 年であるため、TTS の公開から数えると、プロデルは 20 年以上もの歴史を持つ。

3.1.3　ドリトル

ホームページでの紹介には、「ドリトルは教育用に設計されたプログラミング言語です。中学校・高校の教科書や副教材などに採用されています。小学校（総合・算数・理科・音楽など)、中学校（技術科の計測制御・双方向コンテンツ)、高等学校（情報の科学・社会と情報、情報 I・情報 II)、大学（プログラミング入門、IoT）などでご利用ください。」などとある[16]。

このようにドリトルは、なでしこやプロデルと比べ、より教育用途に特化し、教育現場を中心に普及が進んでいるのが特徴である。リファレンスやマニュアルも主には授業用テキストや教師向けの授業資料という形で整備され、配布されている。

3.2　日本語プログラミング言語の現状

日本語プログラミング言語の現状について、「日本語で記述できるプログラミング言語の

存在を知っていますか？」という内容の認知度アンケート調査をウェブアンケートで行った（図 3-2-1）。アンケート調査を行った対象は、いずれかのプログラミング言語を授業もしくは、職業関連などで一度は触れたことがある方のみ、100 名である。アンケート対象の年齢には特に制限は設けず、年代ごとに認知度が大きく変わるという事もなかったため、年齢は表記しない。

アンケート結果は、「はい」と回答した方が 12 名、「いいえ」と回答した方が 88 名となった。

（アンケート対象 100 名の中では、いずれかのプログラミング言語に職業関連などで触れたことがある方が 78 名、授業で触れたことがあるだけの方が 22 名。職業関連の方が比べて多いため、「はい」が多少多い可能性も考えられる。）

いずれかのプログラミング言語に一度は触れたことがある方のみに絞っても、認知度がかなり低く、どのプログラミング言語にも一度も触れたことがない方を含めてしまえば、更に認知度が低くなる結果が出ると考える。

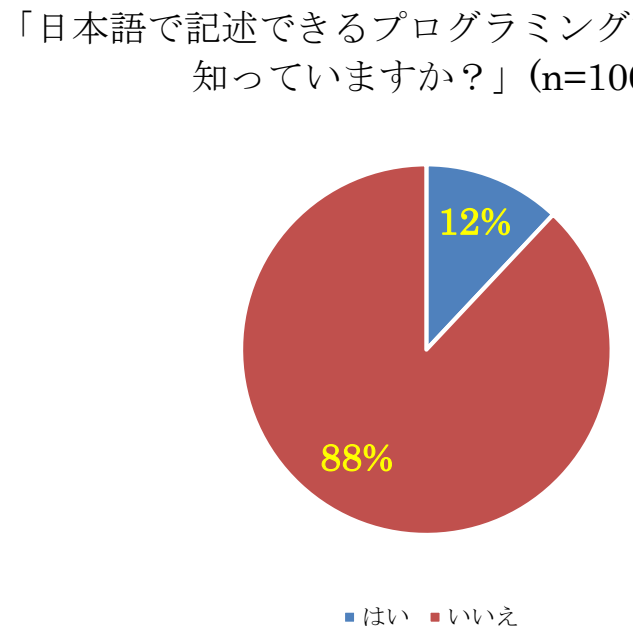

図 3-2-1 「日本語プログラミング言語の認知度　アンケート調査」

このように、我々の母国語である日本語で記述できるプログラミング言語であるにも関わらず、日本語プログラミング言語は、まだまだ認知度は低く、使用率も低いため、とても普及しているとは言えない。そのため、「日本語プログラミング言語は日本人にとって可読性が高いが、なぜあまり普及していないのか」について考察していく必要がある。そして、普及を妨げている要因について調査・考察するにあたって、日本語プログラミング言語と英語記述のプログラミング言語との可読性の比較や、日本語プログラミング言語どうしでの

可読性の比較も行っていく必要がある。その上で、普及している言語との、活用目的、環境、構文などの違いを見つけ出し、そこから考察を行わなければならない。

これらの現状を踏まえた上で、第 4 章では普及を妨げている要因について調査・考察に加え、日本語プログラミング言語を普及させるにはどのようにすればよいのかを論じる。

第4章　日本語プログラミング言語を普及させるには

本章では、第 4.1 節で英語記述のプログラミング言語との可読性の比較、第 4.2 節で日本語プログラミング言語どうしでの可読性の比較、第 4.3 節で日本語プログラミング言語の必要性、第 4.4 節でなぜあまり普及していないのか、第 4.5 節でプログラミング教育での活用、第 4.6 節で初等教育からのプログラミング教育の必要性、第 4.7 節でプログラミング教育の実態、という順に論じる。

4.1　英語記述のプログラミング言語との可読性の比較

日本語プログラミング言語と英語記述のプログラミング言語との可読性の比較をするにあたって、まずは、①Java と②なでしこについてアンケート調査を行った。両者ともいわゆる「FizzBuzz 問題」についてのソースコードである。①Java は、少しでも分かりやすくするため、簡潔に記述しているが、②なでしこは定義から呼び出しまでしっかりと記述しているため、両者の間に多少のプログラム上の違いはあるものとする。

① Java

```
public class FizzBuzz {
  public static void main(String[] args) {
    for (int i = 1; i <= 100; i++) {
      if (i % 3 == 0 && i % 5 == 0) {
        System.out.println(“FizzBuzz”);
      } else if (i % 3 == 0) {
        System.out.println(“Fizz”);
      } else if (i % 5 == 0) {
        System.out.println(“Buzz”);
      } else {
        System.out.println(“i”)
      }
    }
  }
}
```

② なでしこ

●（N で）フィズバズ取得とは

FIZZ は、N を 3 で割った余りが 0 と等しい。

BUZZ は、N を 5 で割った余りが 0 と等しい。

もし、FIZZ、かつ、BUZZ ならば、それは「FizzBuzz」。

違えば、もし、FIZZ ならば、それは「Fizz」。

違えば、もし、BUZZ ならば、それは「Buzz」。

違えば、それは N

ここまで

N を 1 から 100 まで繰り返す

N でフィズバズ取得して表示。

ここまで。

以上の前提のもと、「プログラム①と②ではどちらの方がわかりやすいですか？」という内容のアンケート調査をウェブアンケートで行った（図 4-1-1）。また、そのように回答した理由も同時に集計した。アンケート調査を行った対象は、プログラミング言語に触れた経験がない方も含めた、100 名。アンケート対象の年齢には特に制限は設けず、年代ごとに認知度が大きく変わるという事もなかったため、年齢は表記しない。

アンケート結果は、「①」と回答した方が 4 名、「②」と回答した方が 96 名となった。ほとんどの方が Java よりなでしこの方が分かりやすい、すなわち英語記述のプログラミング言語より日本語プログラミング言語の方が、可読性が高いといえる結果となった。

「①」と答えた理由としては、「Java のような英語記述のプログラミング言語は使い慣れているため」、「普段から Java を使うことが多いため」、「日本語プログラミング言語は初めて見たので少し違和感があるため」、「知らないプログラミング言語よりも知っているプログラミング言語の方が分かりやすいため」という 4 つがあがった。このことから、「①」と回答した 4 名ともに共通する要因は、Java となでしこの認知度や普及度の差であると考えられる。日本語プログラミングは英語記述のプログラミング言語と比べ、はるかに認知度や普及度で劣っているため、当然といえる意見である。そのため、これらの意見は、日本語プログラミング言語がより普及した際には解決へ向かうと考える。

「②」と答えた理由としては、簡潔に言うとすべて、「日本語だから」というものであった。やはり、我々が使い慣れた母国語である日本語で記述できるという事は最大の利点になり得るということである。さらに、なでしこは、自然言語の日本語に近づけようと作られているため、プログラミング言語をあまり知らない方にとっても可読性が高かったということも考えられる。しかし、同じ日本語プログラミング言語であるドリトルのような、英語記述のプログラミング言語に近い性質を持っている言語との比較ではどうなるかは、更に考察していく必要がある。

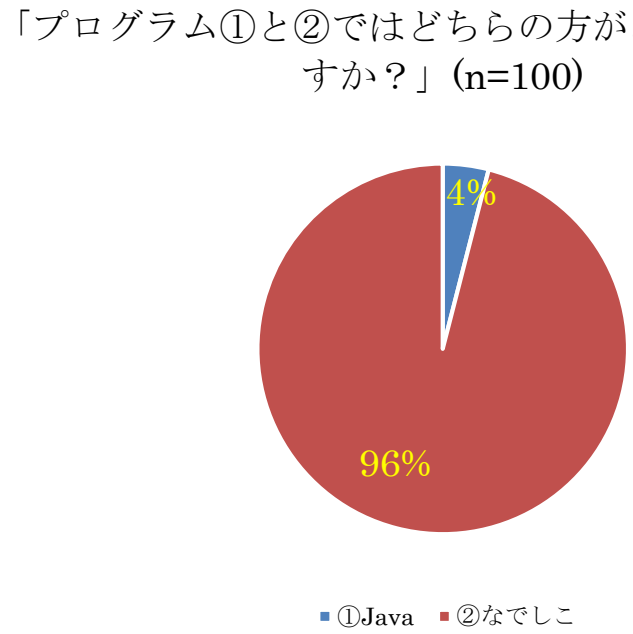

図 4-1-1 「英語記述のプログラミング言語との可読性の比較　アンケート調査」

4.2　日本語プログラミング言語どうしでの可読性の比較

日本語プログラミング言語の中でも有名な、「なでしこ、プロデル、ドリトル」の 3 つの中では、なでしことプロデルは自然言語の日本語に近く、ドリトルはより英語記述のプログラミング言語に近い性質を持っている。第 4.1 節、日本語プログラミング言語と英語記述のプログラミングとの可読性の比較では、Java となでしこをもちいてアンケートを行ったが、母国語である日本語で記述されているからという理由で、日本語プログラミング言語の方が英語記述のプログラミング言語より分かりやすいと回答した方がほとんどであった。しかし、母国語である日本語プログラミング言語の中でも、構文が自然言語の日本語に近づけて作られている言語か、英語記述のプログラミング言語に近づけて作られている言語と分かれている。そのため、この節では、違う国の言葉どうしでの比較ではなく、そのような同じ日本語で記述するプログラミング言語どうしでの可読性の比較をしていく。

第 4.1 節と同様に、①なでしこと②ドリトルについてアンケート調査を行った。両者ともいわゆる「FizzBuzz 問題」についてのソースコードである。両者の間に多少のプログラム上の違いはあるものとする。

① なでしこ

```
●（N で）フィズバズ取得とは
  FIZZ は、N を 3 で割った余りが 0 と等しい。
  BUZZ は、N を 5 で割った余りが 0 と等しい。
  もし、FIZZ、かつ、BUZZ ならば、それは「FizzBuzz」。
  違えば、もし、FIZZ ならば、それは「Fizz」。
  違えば、もし、BUZZ ならば、それは「Buzz」。
  違えば、それは N
ここまで

N を 1 から 100 まで繰り返す
  N でフィズバズ取得して表示。
ここまで。
```

② ドリトル

```
数 = 1。
「結果 = 「数 % 15 == 0」！なら「"FizzBuzz"」
    そうでなければ「数 % 3 == 0」なら「"Fizz"」
    そうでなければ「数 % 5 == 0」なら「"Buzz"」
    そうでなければ「数」実行。
  数 = 数 + 1。
  ラベル！（結果）作る。」！100　繰り返す。
```

以上の前提のもと、「プログラム①と②ではどちらの方がわかりやすいですか？」という内容のアンケート調査をウェブアンケートで行った（図 4-2-1）。アンケート調査を行った対象は、プログラミング言語に触れた経験がない方も含めた、100 名。アンケート対象の年齢には特に制限は設けず、年代ごとに認知度が大きく変わるという事もなかったため、年齢は表記しない。

アンケート結果は、「①」と回答した方が 45 名、「②」と回答した方が 55 名となった。一見、ほぼ同じようにも見えるが、「①」と回答した方は、ほとんどがプログラミング言語に触れた経験がない、もしくはプログラミング言語を授業で触れたことがある程度であった。それに対し、「②」と回答した方は、プログラミング言語に職業関連などで触れたこと

がある、すなわちかなり英語記述のプログラミング言語を使い慣れているということがほとんどであった。すなわち、なでしこのように、構文が自然言語の日本語に近づけて作られている言語は、プログラミング未経験の初心者や、プログラミング言語の学習途中の学生などに受け入れされやすい傾向があると考えられる。そして、ドリトルのように、構文が英語記述のプログラミング言語に近づけて作られている言語は、いずれかのプログラミング言語を習得している、もしくは、プログラミング言語の学習過程にあるが、すでにある程度十分な知識を持っている方に受け入れされやすい傾向があると考えられる。そのため、教育分野での普及では、構文を自然言語の日本語に近づけて作られた、学習用の日本語プログラミング言語がカギとなる可能性があり、仕事で実際に使用するなどの本格的な業務用途での普及では、構文を英語記述のプログラミング言語に近づけて作られた、ソフトウェア開発環境の整った日本語プログラミング言語がカギとなる可能性がある。

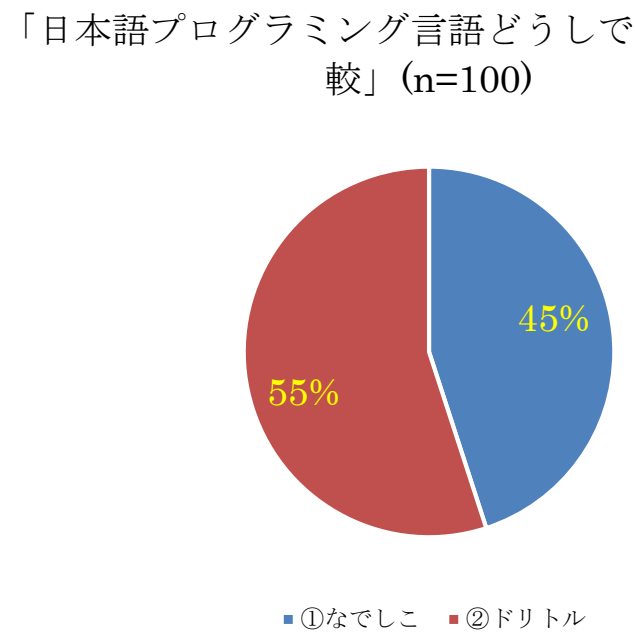

図 4-2-1 「日本語プログラミング言語どうしでの可読性の比較　アンケート調査」

4.3　日本語プログラミング言語の必要性

日本語プログラミング言語の必要性として、日本人のプログラミング初心者や、学生、英語の苦手な人などにとって学習コストが低いという事ももちろんあるだろうが、必要になってくる要因の中でも最も大きいものとして、言語間距離があると考える。

インド・ヨーロッパ語族の言語として代表的なものには英語がある。しかし、図 4-3-1 を見れば分かるように、日本語は英語からかけ離れた言語である。そのため、プログラミング言語においては、他の国に比べても、英語記述のプログラミング言語ではなく、母国語であ

る日本語の言語の必要性が高くなってくる。また、言語間距離の原因としては、日本語との単語や文法、発音、文化、風習などの違いが考えられる。その中でも、プログラミング言語について考えているため、文法に着目すると、日本語は SOV 型であるが、インド・ヨーロッパ語族の言語（ドイツ語、オランダ語を除く）は SVO 型である。このように、文法の違いは、可読性の高さについて考えていく際にも重要な要素になり得ると考える。

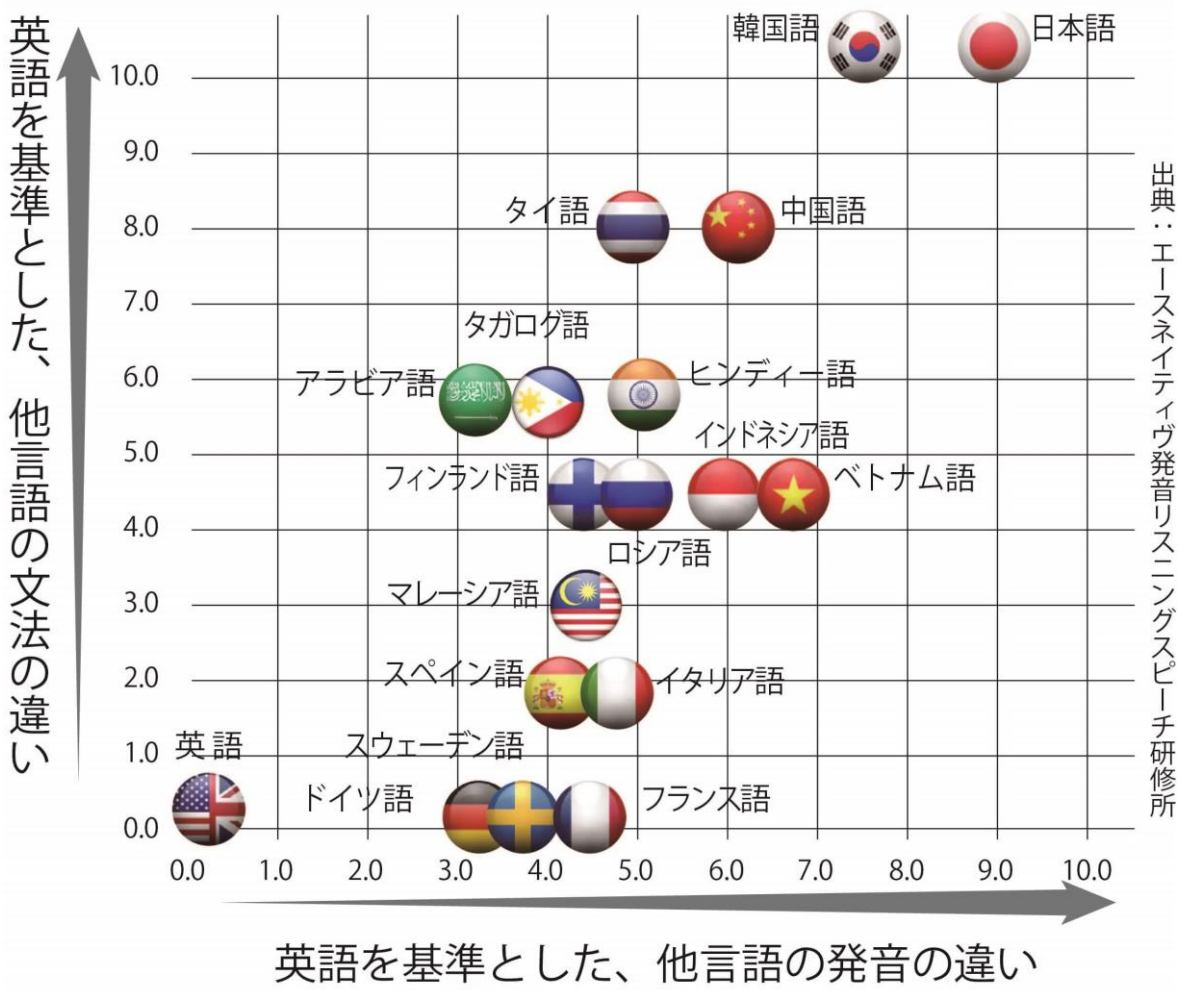

図 4-3-1 「言語間距離」
（出展：ACE PRO
「エースプロについて」（http://www.ace-schools.co.jp/about_ace.html））

4.4 なぜあまり普及していないのか

今までにあげたことを踏まえた上で、日本語プログラミング言語の普及を妨げる要因は、日本語プログラミング言語を取り巻く環境で起きている、「ミスマッチ」にあると考える。第 2.3 節、プログラミング言語の種類でも述べたように、プログラミング言語ごとに専門分

野や得意分野が異なる。さらに、コンピュータの種類によっても使用する言語が異なってくる場合もあるため、何をどのような環境で作るかに応じて、プログラミング言語を変える必要がある。また、第 4.2 節、日本語プログラミング言語どうしでの可読性の比較でのアンケート結果から、教育分野での普及では、構文を自然言語の日本語に近づけて作られた、学習用の日本語プログラミング言語がカギとなる可能性があり、仕事で実際に使用するなど本格的な業務用途での普及では、構文を英語記述のプログラミング言語に近づけて作られた、ソフトウェア開発環境の整った日本語プログラミング言語がカギとなる可能性があるといえる。

はじめに、教育分野での普及を考えて行くと、日本語プログラミング言語の中では最も進んでいるように見えるドリトルは、なでしこやプロデルと比べ、より教育用途に特化し、教育現場を中心に普及が進んでいるのが特徴である。リファレンスやマニュアルも主には授業用テキストや教師向けの授業資料という形で整備され、配布されている。このような事から、ドリトルを教育分野で利用するのに適した環境は整っているように見える。しかし、ドリトルは、構文を英語記述のプログラミング言語に近づけて作られた言語であり、それを利用すると考えられる層は、アンケート結果より、いずれかのプログラミング言語を習得している、もしくは、プログラミング言語の学習過程にあるが、すでにある程度十分な知識を持っている層である。

次に、業務分野での普及を考えて行くと、日本語プログラミング言語の中では最も進んでいるように見えるプロデルは、日本語プログラミング言語の中では珍しく、教育分野での使用のためには作られておらず、汎用的なソフトウェア開発を目標としているのが特徴である。バッチ処理や画像描画、GUI 部品の機能が用意されている、Excel, CSV 形式操作、ODBC 接続ができる、ツールやゲーム作りが可能、大量のデータ処理を短時間で実行できる、MSIL コンパイラを搭載しているので、コンパイルで更なる高速化が可能、プラグインで FeliCa や Twitter との連携機能を利用できる、C#を使って独自のプラグインを作ることができる、など実用的である。このような事から、プロデルを業務用途で利用するのに適した環境は整っているように見える。しかし、プロデルは、構文を自然言語の日本語に近づけて作られた言語であり、それを利用すると考えられる層は、アンケート結果より、プログラミング未経験の初心者や、プログラミング言語の学習途中の学生などの層である。

つまり、教育分野での普及、業務分野での普及ともに、利用するのに適した環境は用意できているように見えるが、学習する日本語プログラミング言語の構文が「ミスマッチ」しているという事考えられるのである。そして、そのような「ミスマッチ」を起こさず、日本語プログラミング言語を普及させていくには、初等教育では 2020 度より開始したプログラミング教育を受け、プログラミング的思考が育まれた状態である、中等教育以降で日本語プログラミング言語を活用することが重要であると考える。

4.5　プログラミング教育での活用

はじめに、プログラミング教育の目的はプログラミング的思考を養うことにあり、プログラミング的思考とは、「自分が意図する一連の活動を実現するために、どのような動きの組合せが必要であり、一つ一つの動きに対応した記号を、どのように組み合わせたらいいのか、記号の組合せをどのように改善していけば、より意図した活動に近づくのか、といったことを論理的に考えていく力」と文部科学省によって定義されている。また、「児童がおのずとプログラミング言語を覚えたり、プログラミングの技能を習得したりするといったことは考えられますが、それ自体をねらいとしているのではない」とも述べられている。つまり、プログラミング教育の目的は、技能の習得ではなくプログラミング的（論理的）思考力を身に着けることである。

そして、そのような目的であるプログラミング教育で、日本語プログラミング言語を活用していくには、「動きに対応した記号」が日本語で表現されている日本語プログラミング言語で書かれたコードは、日本語を理解できれば手順書のように読めるが、曖昧さを含まないように一定のルールに従って記述されている必要がある。この特性を活かして、読解力や分かりやすく記述する力を養うような活用ができるのではないかと考える。

4.6　初等教育からのプログラミング教育の必要性

プログラミング教育の必要性として最も重要な要素に、「論理的思考力を鍛えること」がある。そして、その論理的思考力を早期段階である初等教育の頃から鍛えることが重要であると考える。また、文部科学省の文書には、「今後の社会の在り方について、とりわけ最近では、「第 4 次産業革命」ともいわれる、進化した人工知能が様々な判断を行ったり、身近な物の働きがインターネット経由で最適化されたりする時代の到来が、社会の在り方を大きく変えていくとの予測がなされているところである。教育界には、そのような社会的変化の中でも、子供たちが自信を持って自分の人生を切り拓き、よりよい社会を創り出していくことができるそうした資質･能力として、読解力、論理的思考力、創造性、問題解決能力などは、時代を超えて常にその重要性が指摘されてきており、これからの時代においてもその重要性が変わることはない。これらに加えて、情報や情報技術を問題の発見･解決に活用していく力（情報活用能力）の重要性も高まっている。学校教育は、こうした資質･能力の育成に向けて充実を図らなければならない。」というような記述もあり、これからの時代、より一層必要になってくるといえる。

また、第 4.4 節、なぜあまり普及していないのかでも述べたように、日本語プログラミング言語を普及させていくには、初等教育では 2020 度より開始したプログラミング教育を受け、プログラミング的思考が育まれた状態である、中等教育以降で日本語プログラミング言語を活用することが重要であると考える。また、それが日本人にとっては最も学習コストが

低く、効率の良い学び方だと考える。そのため、中等教育以降での学習成果をより上げるためにも、初等教育でのプログラミング教育による下積みというものも必要性としてあげられる。

4.7　プログラミング教育の実態

コロナウイルスの影響による大きな遅延はあったものの、2020 年度から初等教育でのプログラミング教育が実際に開始された。そこで、プログラミング教育の実態調査を行い、現場の教師はどのような実感や考えであるか、小学校教師 1 名に 5 項目の質問に答えて頂いた。

まず初めに、プログラミング教育で向上するといわれている、読解力や記述力の現状について把握するため、「児童の読解力・記述力は 5 年ほど前と比較してどう変わったか」についてである。児童の読解力・記述力については、低下の傾向にあるそうである。それは、日頃から自分の頭で考えることが減ってきていることと関係しているように感じ、答えが分からなかったらすぐに大人に聞いたり、調べたりする習慣がつき、考える力を養う機会を逃しているように思えるとのことである。その結果、文章から様々なことを読み取ったり、想像したりすることが苦手な子たちが増え、また、記述力に関しても、「何を書いたらいいかわからない」と考えることをやめてしまう子や、頭の中で自分の思考を整理して文章に表すことが苦手な子が増えてきている現状であるそうだ。低下の一途をたどっているからこそ、このタイミングで初等教育でのプログラミング教育が開始されたのは正解であったといえる。

次に、プログラミング教育で能力を育んでいく際に、重要な部分となっていく、算数の力について把握するため、「児童の算数の回答力は 5 年ほど前と比較してどう変わったか」についてである。児童の算数の回答力基本的な計算の力は大きな変化はないように感じるそうである。しかし、子どもが自分の考えを発表する場面で、友達に分かりやすく、順序立てて説明することを苦手とする子は増えてきているようである。プログラミング的思考とは、「自分が意図する一連の活動を実現するために、どのような動きの組合せが必要であり、一つ一つの動きに対応した記号を、どのように組み合わせたらいいのか、記号の組合せをどのように改善していけば、より意図した活動に近づくのか、といったことを論理的に考えていく力」と文部科学省によって定義されているが、まさにこのプログラミング的思考をする力が低下してきているということである。

次に、「小中学校でのプログラミング教育を必要と感じるか否か」についてである。論理的思考力を鍛えることは今の子たちにとって大切なことだと思い、従来の国語や算数などの科目の中でも、教材の扱い方や教師側の指導の仕方によって論理的思考力を育てることができるのではないかとのことである。これは、2020 年度から開始された初等教育でのプログラミング教育の手法である、既存の科目にプログラミングの要素を取り込むという方

法でやっていけるのではないかということである。

次に、2020 年度で始まったばかりの初等教育でのプログラミング教育なため、なにかしらトラブルや解決しなければならない事柄があるだろうという事で、「現在のプログラミング教育について困っていること」についてである。プログラミング教育が算数や理科などの教科書の中で、一部扱われるようになってきたものの、国や県、市の研修などが進んでおらず、何をどのように教えていけばよいのか、個人に任されている部分が多いため、はっきりと分かっていないのが現状のようである。これに関しては、開始 1 年目だからという事も考えられるが、最も大きな要因としては、コロナウイルスの影響で、2020 年度の授業や研修が大幅に遅延したことであると考える。しかし、小学校教師でプログラミング言語に本格的に触れてきた方は少数であると考えられるため、プログラミング教育を行うことによって教師への負担が大きくなってしまう可能性が高い。実際にこの方は現在、スクラッチジュニアという感覚的にプログラミングが学べるアプリを使って、プログラミングに触れる学習を行っているそうだ。近年、初等教育の高学年科目は、専門の教師を置く方針になったのと同様に、プログラミング教育を本格的に教育に取り入れるならば、新たに教える科目への理解を深めなければいけないという、教師への負担を少しでも軽減するため、専門の教師を用意すべきだと考える。

最後に、「日本語で記述できるプログラミング言語を初等教育に利用してみたいか否か」についてである。日本語で記述できるプログラミング教材は小学校の高学年、中学生くらいであれば活用できるのかも知れないが、まず、初等教育では国語や算数などの教科学習の中で、論理的思考力を育てていくことが大切であると考えているとのことである。やはり、初等教育の間は、現状のプログラミング教育のやり方で、プログラミング的（論理的）思考力を育んでいくことが大切であり、日本語プログラミング言語などを導入するとするならば、中等教育以降で行っていくのが適切であると考える。

第5章　まとめ

本論文では、日本語プログラミング言語はなぜあまり普及していないのか、そして、普及方法や新たな活用方法の模索を行ってきた。

教育分野での普及を考えて行くと、日本語プログラミング言語の中では最も進んでいるように見えるドリトルは、なでしこやプロデルと比べ、より教育用途に特化し、教育現場を中心に普及が進んでいるのが特徴である。リファレンスやマニュアルも主には授業用テキストや教師向けの授業資料という形で整備され、配布されている。このような事から、ドリトルを教育分野で利用するのに適した環境は整っているように見える。しかし、ドリトルは、構文を英語記述のプログラミング言語に近づけて作られた言語であり、それを利用すると考えられる層は、アンケート結果より、いずれかのプログラミング言語を習得している、もしくは、プログラミング言語の学習過程にあるが、すでにある程度十分な知識を持っている層である。

業務分野での普及を考えて行くと、日本語プログラミング言語の中では最も進んでいるように見えるプロデルは、日本語プログラミング言語の中では珍しく、教育分野での使用のためには作られておらず、汎用的なソフトウェア開発を目標としているのが特徴である。バッチ処理や画像描画、GUI 部品の機能が用意されている、Excel, CSV 形式操作、ODBC 接続ができる、ツールやゲーム作りが可能、大量のデータ処理を短時間で実行できる、MSIL コンパイラを搭載しているので、コンパイルで更なる高速化が可能、プラグインで FeliCa や Twitter との連携機能を利用できる、C#を使って独自のプラグインを作ることができる、など実用的である。このような事から、プロデルを業務用途で利用するのに適した環境は整っているように見える。しかし、プロデルは、構文を自然言語の日本語に近づけて作られた言語であり、それを利用すると考えられる層は、アンケート結果より、プログラミング未経験の初心者や、プログラミング言語の学習途中の学生などの層である。

このように、日本語プログラミング言語は、教育分野での普及、業務分野での普及ともに、利用するのに適した環境は用意できているように見えるが、学習する日本語プログラミング言語の構文が「ミスマッチ」しているという事考えられるのである。そして、そのような「ミスマッチ」を起こさず、日本語プログラミング言語を普及させていくには、初等教育では 2020 度より開始したプログラミング教育を受け、プログラミング的思考が育まれた状態である、中等教育以降で日本語プログラミング言語を活用することが重要であると考える。

また、今後は、2021 年度は中等教育で、2022 年度は高等教育でプログラミング教育が開始されていくため、その動向にも注目していく。更に、2020 年度はコロナウイルスにより、授業や教師の研修が不十分であることや、複数人の教師への調査を行えなかったことなどがあるため、改めて 1 年後に調査し直す必要があると考える。

謝辞

本研究を進めるにあたり、お忙しい時期でも研究の進め方や論文の書き方などご指導いただきました卒業論文指導教員の毛利元昭准教授に心より感謝を申し上げます。

同じく、様々な助言や参考文献の紹介などをしていただきました岩田員典教授、小学校へのアンケートにご協力いただきました前原裕樹准教授に心より感謝を申し上げます。

最後に、多くのご支援をいただきました毛利ゼミの皆様には心より感謝を申し上げます。

参考文献

[1] クジラ飛行机（2020）『プログラミング言語大全』株式会社技術評論社

[2] 文部科学省「小学校段階における論理的思考力や創造性、問題解決能力等の育成とプログラミング教育に関する有識者会議」
＜https://www.mext.go.jp/b_menu/shingi/chousa/shotou/122/index.htm＞（最終閲覧日 2020-8-20）

[3] 文部科学省「小学校段階における論理的思考力や創造性、問題解決能力等の育成とプログラミング教育に関する有識者会議について」
＜https://www.mext.go.jp/b_menu/shingi/chousa/shotou/122/attach/1370404.htm＞（最終閲覧日 2020-8-20）

[4] 文部科学省「小学校段階におけるプログラミング教育の在り方について（議論の取りまとめ）」
＜https://www.mext.go.jp/b_menu/shingi/chousa/shotou/122/attach/1372525.htm＞（最終閲覧日 2020-8-20）

[5] 文部科学省「小学校段階における論理的思考力や創造性、問題解決能力等の育成とプログラミング教育に関する有識者会議　議事要旨・議事録・配付資料」
＜https://www.mext.go.jp/b_menu/shingi/chousa/shotou/122/giji_list/index.htm＞（最終閲覧日 2020-8-20）

[6] 文部科学省「小学校段階における論理的思考力や創造性、問題解決能力等の育成とプログラミング教育に関する有識者会議（第 3 回）　議事録」
＜https://www.mext.go.jp/b_menu/shingi/chousa/shotou/122/gijiroku/1382219.htm＞（最終閲覧日 2020-8-21）

[7] 文部科学省「小学校段階における論理的思考力や創造性、問題解決能力等の育成とプログラミング教育に関する有識者会議（第 3 回）　配付資料」
＜https://www.mext.go.jp/b_menu/shingi/chousa/shotou/122/shiryo/1371637.htm＞（最終閲覧日 2020-8-21）

[8] AI:人工知能特化型メディア「自然言語処理（NLP）とは | 仕組み・活用例・今後の課題」
＜https://ledge.ai/nlp/＞（最終閲覧日 2020-8-23）

[9] Hatada's Home Page「プログラミング言語の基本概念」
＜http://home.a00.itscom.net/hatada/lp2012/chap01/language.html＞（最終閲覧日 2020-8-23）

[10] IT 用語辞典「機械語」
＜https://e-words.jp/w/%E6%A9%9F%E6%A2%B0%E8%AA%9E.html＞（最終閲覧日 2020-8-23）

[11] HPCS Lab - 筑波大学システム情報工学研究科 ハイパフォーマンス・コンピューティング・システム研究室「言語処理系とは」
＜http://www.hpcs.cs.tsukuba.ac.jp/~msato/lecture-note/＞（最終閲覧日 2020-10-3）

[12] TECH のススメ「プログラミング言語とは？特徴や言語の種類について解説」
＜https://haa.athuman.com/media/it/programming/2120/＞（最終閲覧日 2020-10-3）

[13] TechTarget「Java から「Kotlin」に乗り換えたくなる 5 つの理由」
＜https://techtarget.itmedia.co.jp/tt/news/2002/17/news06.html＞（最終閲覧日 2020-10-18）
[14] なでしこ 3「日本語プログラミング言語「なでしこ 3」について」
＜https://nadesi.com/doc3/index.php?＞（最終閲覧日 2020-11-12）
[15] 日本語プログラミング言語「プロデル」「プロデル　もうひとつの日本語プログラミング言語」
＜https://rdr.utopiat.net/＞（最終閲覧日 2020-11-12）
[16] プログラミング言語「ドリトル」「教育用プログラミング言語「ドリトル」情報ページ」
＜https://dolittle.eplang.jp/＞（最終閲覧日 2020-11-12）
[17] TECH CAMP「「なでしこ」「プロデル」「ドリトル」三大日本語プログラミング言語を解説」
＜https://tech-camp.in/note/technology/41396/#i-9＞（最終閲覧日 2020-11-12）
[18] ACE PRO「エースプロについて」
＜http://www.ace-schools.co.jp/about_ace.html＞（最終閲覧日 2020-12-20）
[19] IT 用語辞典「マークアップ言語」
＜https://ewords.jp/w/%E3%83%9E%E3%83%BC%E3%82%AF%E3%82%A2%E3%83%83%E3%83%97%E8%A8%80%E8%AA%9E.html＞（最終閲覧日 2021-2-4）
[20] マイナビクリエイター「マークアップ言語とは」
＜https://mynavi-creator.jp/blog/article/what-is-mark-up-language＞（最終閲覧日 2021-2-4）
[21] Mecha log「「プログラミング」 初心者でも理解できる！基礎中の基礎」
＜https://mechalog.com/programming-beginner＞（最終閲覧日 2021-2-5）
[22] じゃぱざむ「コンパイラとインタプリタの違いは？言語の違いを分かりやすく解説！」
＜https://jpazamu.com/interpreter-compiler/＞（最終閲覧日 2021-2-6）