日本語プログラミング言語の可能性について

中居　七瀬

目　次

——日本語プログラミング言語の可能性について——

1　はじめに

今回、私は日本語プログラミング言語の可能性について論じることにした。最初は、卒業論文のテーマとして、日本語プログラミング言語コンパイラを製作してみてはどうかと勧められていた。しかし、コンパイルを作る以前の問題として、私は日本語プログラミング言語というものについて、全くの無知であった。

少しずつ調べてみたところ、初心者向けの日本語プログラミング言語が多いことから、「日本語はプログラミング言語に向いているのだろうか？」という疑問が沸き上がってきた。

そこで、初心者向けの言語に焦点を当てながら、日本語プログラミング言語にどのような特徴があるのかを分析する。それから問題点を提起し、日本語をプログラミング言語に採用するならば、どのような方向性の利用が向いているかを考察する。

2　既存の日本語プログラミング言語について

まずは、日本語プログラミング言語には、どのようなものがあるかを述べる。多くのユーザを獲得し、実際に利用されている言語もあれば、言語の構成案だけ、文法の制定だけといった言語まで多数存在する。大まかに実用と教育用という二つの大きな目的の違いで、節を分けた。ただし、実際にコードが組めない構想段階の言語は省くことにする。

(1) 実用目的の日本語プログラミング言語

(i) 日本語 AFL

1983 年に松下電産グループが開発した。仮名漢字まじりの日本語でプログラミングが可能であった。

（ii）和漢

開発者は国際データ機器株式会社の鈴木孝則。日本語 AFL を改良して商品化したものである。

（iii）G-BASIC

開発はマイクロソフト。1982 年にトミー工業株式会社（現在の株式会社タカラトミー）から発売された「ぴゅう太」に搭載された言語である。「PRNT」を「カケ」と書くなど、Basic の命令をただ日本語に置き換えただけという単純なものであった。独自性があり、他のシリーズの BASIC と互換性が無かった(1)。

（iv）Mind

開発者はスクリプツ・ラボ社の片桐明。Forth のスタックという概念を利用した逆ポーランド記法を元にして、1985 年に開発された。Windows 版・UNIX 版・MS-DOS 版が販売されてきたが、最近になって MS-DOS 版のみフリーウェア化された。

株式会社ぐるなび、株式会社イー・マーケティングといった、企業の検索エンジンに採用された実績がある。最も歴史のある日本語プログラミング言語である(2)。

（v）なでしこ（ひまわり）

開発者はクジラ飛行机。インタプリタ方式のスクリプト型言語。対応機種は Windows のみ。親言語は Delphi であり、フリーウェア且つ、オープンソースである。2001 年に開発したひまわりを元に改良されて、なでしこが 2004 年に開発された。

開発の目的として、開発者が「自分の事務仕事を簡略・自動化したかったため」と述べている。そのため、Windows や Microsoft Office 関連の機能が充実している。「なでしこで誰でも簡単プログラマ」をモッ

トーに、初心者を中心に、上級者までカバーする幅広い機能を持つ[3][4]。

(vi) TTSneo(TTS)

開発者は馬場ゆうと。インタプリタ方式のスクリプト型言語。正式名称は Technology Terminal Script neo。対応機種は Windows のみ。親言語は Visual Basic であるために、Visual Basic6.0 ランタイムが必須である。フリーウェアであるが、ソースは公開されていない。1999 年に前身である TTS が開発され、2002 年に現在の名称に変更された。

Logo のタートルグラフィックスに似た機能を持ち、図形描画の機能が充実している。また、親言語の性質を受け継ぎ、GUI 部品関連が充実している[5]。

(vii) 分析

まず、上位４つの言語は、1980 年代に開発された言語である。Mind のみを除いて、現在は使われていない。詳しい情報は残っていないが、既存のアルファベット言語を直訳しただけのものであったことから、大して実用的な言語にならなかったのではないかと思われる。また、言語開発は企業主体で進められていたことが分かる。

次に、下位２つの言語は、1990 年代以降に開発された新しい言語である。企業主体ではなく、専門家でもない個人の開発者が、フリーウェアで開発している点が目新しい。また、どちらも初心者向けを謳っている。日本語がプログラミングに向いている印象を受け取ることができる、Windows95 の発売以降、高性能な家庭用 PC が普及し、個人が趣味でプログラムを組める時代になったと言える。

(2) 教育目的の日本語プログラミング言語

(i) 言霊

開発者は、慶応義塾大学の大岩研究室に所属する岡田健。親言語はJava。Java 実行環境さえあれば、OS は問われない。2002 年に開発された教育用言語である。

初心者がコンパイルエラーなどに戸惑わないよう、アルゴリズムに集中できるように、自然な日本語の文章を用いてプログラミングできる言語を目指して作られた。ただし、2007 年現在、公式サイトが消失している(6)。

(ii) ことだま on squeak

開発者は、慶応義塾大学大岩研究室。2005 年に開発され、Windows 版がある。フリーウェアである。

アラン・ケイが開発したビジュアルプログラミング環境 squeak を元に、日本語化した教育用言語である。条件分岐タイルや、命令タイルを並べて、視覚的にプログラムを組むことが出来る。自分で書いた絵をオブジェクトにし、制御することが可能である。全国の小学校・中学校でプログラミング入門用の言語として採用されている実績がある(7)。

(iii) ドリトル

開発者は一橋大学の助教授である兼宗進。Windows 版と Macintosh 版が存在する。フリーウェアである。オブジェクト指向が学べる作りになっている。

Logo のタートルグラフィックスを、日本人向けに改良したものである。GUI オブジェクトや音楽オブジェクトがあり、カメを操作して視覚的にプログラムを組むことができる。ことだま on squeak と同様に、全国の小学校・中学校の技術家庭や情報の授業に、教材として採用され

ている[8]。

(iv) DNCL

大学入試センターの入試科目「情報関係基礎」で用いられている手順記述言語。これだけで実行できるものではなく、いわゆる疑似コードである。公平性を期すために特定の言語の出題を避ける目的で、入試試験問題に採用されている。受験生に対する特別な説明もなく使われていることから、構造の分かりやすさに優れている[9]。

また、この言語を元に機能を拡張した xDNCL を実行できる、初学者向けプログラミング学習環境 PEN が存在する。正式名称は Programming Environment for Novices である。2003 年に、大阪学院大学の西田研究室と大阪市立大学大学院の松浦研究室の共同で開発された。ソースの入力支援機能があり、容易にプログラムを入力できる。また、プログラムを実行している最中の変数の移り変わりを観察できる仕組みがある[10]。

(v) 分析

開発時期が判明しなかった言語もあるが、2000 年代に入ってから教育用日本語プログラミング言語の開発が進んでいる。開発元は大学関連機関が占めている。身の回りにネットワークや、プログラムで制御された機械が増えた中で、学生に情報処理を教える必要が出てきたためであろう。

教育目的とはいえ、ユーザ対象によって大きく 2 つに分けることが出来るのではないだろうか。一つ目は低年齢を対象としたプログラミングを体験する言語、二つ目はプログラミングを習得したい人向けのプログラミングの入門言語。前者は、ことだま on squeak とドリトルであり、後者は DNCL と言霊である。

3　各言語の特徴を分析する

(1) 日本語化の方針についての分析

前章において、様々な目的の日本語プログラミング言語が存在していることを述べた。この節では、それぞれの言語がなぜ日本語で開発したのか、狙いを分析する。

(i) なでしこ

開発者は、「これからプログラミングをはじめたい人。パソコンでやる仕事を自動化したい人。ちょっとしたツールを作りたい人。Excel のマクロや VB に挫折した人。日本語とプログラミングが好きな人」に薦めている。

つまり、プログラミング初心者、プログラマではない職業の人々を、主なユーザ対象にしている。プログラミングに抱かれている敷居の高さを下げる目的に、日本語が採用されている。

「それで、その人の書いたプログラムをちょっと改造する必要が生じたとき、日本語でプログラムが書いてあれば、どこを直したらよいのか、大体検討がつくものです。可読性の良さが日本語プログラミング言語のメリットです。」とのように、日本語の可読性の良さも述べられている。事務用に、使い捨てプログラムに向いていると公言している。

(ii) TTSneo

公式サイトには「英単語を覚える必要がないので、“ プログラミング ” という言葉に抵抗があった方でも気軽にソフトやツールを作ることができます」「ソフトを作ったことがなくても、例文を参考にしていくことですぐにソフトが作れます。」とのように書かれている。なでしこ同様に、プログラミング初心者をユーザ対象にしている。日本語の採用は、敷居を下げることが目的である。

(iii) ことだま on squeak

「プログラミングによるものづくり・プロジェクトを通して、意欲ある中高生が IT 職人を目指すきっかけを提供することが本プロジェクトの最終目的である」「本プロジェクトでは日本人の思考の流れを妨げないよう日本語の語順でプログラミングできる環境を提供する」

対象は IT 職人を目指すような意欲ある中学生や高校生である。日本人はプログラミングを考える際に、先ず母国語で手順を考え、その後に英単語に置き換えている。この思考プロセスを慣れた人は簡単にできるが、初心者には難しいのではないか、ということから日本語を採用している。

(iv) ドリトル

「学校教育 (小中高) を中心に、大学生や社会人にも適した入門用のプログラミング言語です。BASIC や LOGO といった、30 年以上前に設計された言語に代り、最新のオブジェクト指向の考え方でプログラムを学ぶことができるのが特徴です。」「日本語文字の採用により、親しみやすく分かりやすい言語になっています。」

対象は小学生から社会人までのプログラミング初心者である。これもまた、日本語の採用は敷居を下げること。そして、ローマ字を習得していない小学生を対象に含めているためである。

(v) 分析

日本語化の目的には、3 つの目的がある。まず 1 つに、どの言語も日本語で書くことで、プログラミングは難しいものであるという敷居を下げようとしている。確かに、低学年の学生にはアルファベット入力からして敷居となるだろう。　2 つ目の理由として、可読性を良くすることを目的としている。一般的に漢字で書かれた熟語は、英語よりも意味が

汲み取りやすい。　3つ目の理由として、日本語の語順でプログラムを組むことで、日本人の思考を生かしている。

(2) 実際のユーザ層についての分析

前節において、殆どの日本語プログラミング言語は初心者に向けて作られていることが判明した。次に、実際のユーザ層について分析する。

(i) なでしこ

なでしこの前身であるひまわりのユーザ６８２人を対象にしたアンケートが、2004年に行われた[11]。ここからは、50％以上が中高生であり、15～20％がサラリーマンであることが読み取れる。専門職であるプログラマ・SEも5％前後存在する。ひまわりとなでしこの対象ユーザ層は同じであるため、なでしこでも同様のユーザ層を持つと推測できる。

(ii) TTSneo

具体的な数値は公表されていない。ただ、掲示板への書き込み規約に「掲示板への参加は原則として14歳以上とします」とあるため、低年齢のユーザ層を獲得していると推測できる。

(iii) ドリトル・ことだま on squeak

どちらも教材であるため、1次ユーザは教員が占めており、2次ユーザは小中学生がほとんどだと思われる。

(iv) 分析

分析対象が少なく断定はできないが、なでしこを見る限り、プログラミングを本職としていないユーザがほとんどを占めているようだ。特に低年齢層のユーザの多さが目立つ。また、割合としては少ないが、高年齢者層にも支持されている。最もPC利用の機会が多い社会人層には、

あまり普及していない。

低年齢層ということは、他言語を学習した経験が無い者が殆どであると思われる。実用と教育用の日本語プログラミング言語が、人生で初めて学習する言語になる可能性は非常に高い。

(3) 初心者の支援体制についての分析

たとえ、どんなに優れた言語であっても、マニュアルが無ければ、ユーザが自力で学習することは叶わない。また、言語が発展し、質を良くして行くためにはコミュニティの発展が欠かせない[12]。以上のことから、この節では、言語自体ではなく、初心者を支援する体制について分析する。

(i) なでしこ

コミュニティとして、公式サイトに誰でも閲覧できる掲示板が、交流・質問・初心者・プログラムの 4 種類の用途に分けて用意されている。質問用と初心者用の掲示板では、投稿者が疑問を解消できたときに、解決済みアイコンをつけることが出来る。初心者でも参加しやすいように、掲示板のチュートリアルもあり、過去ログの検索も可能である。

マニュアルとして、公式サイト上にプログラミングの基礎から学べる講座が用意されている。キャラクタ同士の会話をベースに、サンプルソースや実行結果を載せている。

ただし、一部の命令は使用例や解説が不足している[13]。また、ツール作成といった、発展した内容の講座が完成していない[14]。

そのような内容を取り扱った、なでしこ公式ガイドブックが 2005 年 9 月に出版されている。しかし、絶版となってしまったため、中古本が高価で取引されており、入手が困難である[15]。

(ii) TTSneo

コミュニティとして、公式サイトに誰でも閲覧可能な掲示板が、初心者・質問・不具合報告・雑談の 4 種類の用途に分けて用意されている。FAQ や、メールマガジンもある。また、メールアドレスを登録した会員のみが閲覧できる、プログラム掲示板やサンプル集が存在している。他にも、2005 年に TTS ミニコンテストが開催されたなど、ユーザが参加できる場が用意されている。

なでしこと同様に、オンラインマニュアルが充実している。かなり細やかにサンプルソースや解説が用意されている。なるべく、専門用語を使わない配慮があり、初心者に優しい作りになっている。全てのマニュアルから単語検索が可能なため、膨大なマニュアルの中から必要な情報を取り出しやすくなっている。

(iii) ことだま on squeak

公式サイトが存在するものの、マニュアルや言語本体をダウンロード出来るのみである。あまりコミュニティは発展していない。

マニュアルとしては、サンプルプログラム、練習問題、命令一覧が存在する。文法の説明だけではなく、プログラム作品の製作のプロセスも説明されている。まずは企画から始まり、実装、そしてプレゼンテーションから、報告書の作成までを、実例とともに書かれている。

(iv) ドリトル

公式の Wiki が存在する。これは会員のみが編集できるものである。議論と情報交換用にメーリングリストがあり、定期的に研究会が開かれている。

マニュアルとしては、サンプルプログラム、命令一覧が存在する。また、具体的なサンプルを利用した授業例が投稿されており、ダウンロー

ドが可能になっている。これは、実際に授業に用いられたもので、ドリトルの文法と使用例が流れにそって、関連した練習問題が書かれている。そのため、情報の授業に不慣れな教諭でも、授業でドリトルを導入しやすくなっている。

(v) 分析

どの言語にも、オンライン上のマニュアルが、充実の程度や使い勝手の差があるものの存在している。そして、どの言語にもユーザが自由に参加できる Wiki や掲示板が設けられている。ユーザ数が多くないために、あまり活発とは言えないが、ユーザ間だけでなく開発者との交流や、作品の発表、疑問を解消するコミュニティを形成している。言語の改良や、不具合の修正、マニュアルの作成、掲示板への対応など、プログラミング言語の開発は多岐に渡る。開発者だけでは到底手が回らないところを、こうしたコミュニティがカバーすることができる。

また、どの言語も書籍の類いは不足している。指導者の存在無しに、自力でプログラミングを学ぶものにとって、質のいい参考書や解説本は必需品である。ただ、日本語プログラミング言語であるがゆえに、出版しづらい状況があるのではないかと思われる。ユーザ数が少ないために大量の購買が見込めないことが理由だろう。また前述のとおり、開発者の仕事が多く、執筆だけに集中できないことも理由ではないだろうか。

教育用日本語プログラミング言語は、機能が少なくオンラインマニュアルで事足りているようだ。また、学生のためではなく、指導者寄りのサポートになっている。

4　各言語の文法上の違いを比較する

(1) 基本的な文法の比較

まずは、構造化の基本である「順次・選択・繰り返し・サブルーチン」の表現方法について比較する。C 言語を基本形として、なでしこ・TTSneo・ドリトル・xDNCL を比較対照として取り上げる。

list1 から **list33** に列挙したとおり、どの言語も、全ての要素を備えている。インデントによって動作範囲が決定されていることが共通している。

C 言語 for 文

```
for(初期条件; 終了条件; 繰り返しごとに行う処理){
  処理;
}
```

C 言語 while 文

```
while(条件式){
    処理;
}
```

なでしこと TTSneo は、繰り返し文において、条件文と初期値を省略できる特徴がある。なでしこは、繰り返し処理の終わりを記述しなくても動作するようになっている。最低で 1 行を書くだけで、繰り返し文を記述することが可能だ。このような書き方は、日本語として自然で入力しやすい文章だが、構文としてはやや崩れている。

なでしこ　条件無しの繰り返し文

```
(回数)回　処理
```

なでしこ　条件無しの繰り返し文

（回数）回

　　処理１

　　処理２

TTSneo　**１行で繰り返し**

（回数）回、「（処理）」を繰り返せ

TTSneo　**複数行で命令する場合**

（回数）回、繰り返せ

　　処理１

　　処理２

繰り返し終わり

　C 言語の記述方法と、構文の形がもっとも近いのは DNCL であるといえる。処理の終わりを必ず明記し、条件を省略することもない。構文の形を崩さずに、なおかつ自然に日本語化している。

DNCL　**前条件判定** while-do **文**

《条件式》の間、

　　<処理>

を繰り返す

DNCL　**範囲指定の加算型** for **文**

《変数》を《数値１》から《数値２》まで《増加値》ずつ増やしながら、

　　<処理>

を、繰り返す

ドリトルは、選択・繰り返し・サブルーチンの全てにおいて、柔軟性の無い固い文法となっている。さらに、サブルーチンの中で繰り返し命令が使えない、繰り返し命令の中で繰り返し命令が使えないなど、文法の制限が多い[16]。低学年向け言語に特化するということは、命令が単純化し、実現できる動作の幅が狭まるということになる。

(2) 親言語との比較

プログラミング言語とは、別のプログラミング言語を用いて開発されるものである。どの日本語プログラミング言語も例外なく、親言語が存在する。そこで、親言語との違いを分析する。

(i) なでしこと Delphi

なでしこで記述した **list34** は、親言語である Delphi を用いて記述した **list35** よりも、簡単に GUI 部品のボタンを作成できる。**list36** と **list37** はともにウィンドウのキャプチャを撮ることができるソースである。Delphi で、ボタンを作成する場合、まず procedure という宣言が必要である。この procedure（手続きの意）は値を返さない関数用の宣言であるが、具体的に何を行っている部分なのか分かりづらい。なでしこでは、その謎の宣言は記述することなく、ボタンが作成できる。

それから、なでしこでは、条件と処理を一続きに記述出来る。そのために、具体的で分かりやすい記述が可能である。「ボタンをクリックしたときに、こんにちはと言う」と、日本人の日常的な思考の順で記述できている。さらに、処理の始まりと終わりを書かなくて良いために、文字数が少なくて済む。

こうした特徴を生かした実例として、MYCOM ジャーナルのサイト

で「日本語で 10 行プログラミング」というコラムがある[17]。これは、題名の通り、なでしこ開発者のクジラ飛行机が、なでしこを使って 10 行のサンプルプログラムを組んでいる。10 行という制限の中で、Excel と連携したカレンダーや、バックアップツール、チャットソフトといった便利なソフトが作れることを証明している。

(ii) TTSneo と Vusual Basic

TTSneo で記述した **list38** も、親言語の関数を省略して記述することが可能である。**list38list39** は、ともにウィンドウズの種類を表示するだけの関数である。Visual Basic では 40〜50 行に渡る命令を、たったの 1 文の命令で実行できてしまう。初心者がとまどいそうな、API 関数を呼び出す必要がないためである。このように、Windows のバージョンを知ったところであまり意味は無いかもしれない。しかし、他の関数もこのように省略されているため、言語の利便性が向上している。

(iii) 分析

利便性は向上しているが、関数の内容を、子言語である日本語プログラミング言語で記述できない。そのため関数の処理が、ブラックボックスと化している。その上、マニュアルの説明が不足していることもあり、初心者は仕組みを理解せずに記述するしかない。　とはいえ、現在の実用向け日本語プログラミング言語は、実用性に欠けていた 1980 年代の日本語プログラミング言語を反面教師としている。命令をただ置き換えるだけでなく、日本語らしさを追求している。

5　日本語らしさとその影響を分析する

この章では、もっとも日本語らしい記述が可能な、なでしこを取り上げる。そして、日本語らしい記述と、その影響を分析する。

（iv）関数名について

基本的には、全ての変数名・命令・関数を日本語で書くことが出来る。一部、計算関数（INT、RGB、SIN、SQRT）、論理演算（NOT、OR）、描画関数（LINE、POLY）など、アルファベットでしか記述できないものがある。これらは普遍化されている命令のため、日本語で記述する必要が無いと思われる。

逆に、無理矢理に日本語化した命令も存在する。以下の例は、初心者でなくても、何を指しているのか、どういう動きをするのか分かりにくい。

母艦

…　メインフォームのこと

S で|S を　ナデシコする

…　文字列 S の内容をなでしこのプログラムとして実行する

A の|A に|A を　エクセルシート注目

…　A 番目（1～n）または名前 A のエクセルシートをアクティブにする

一般的には命令規則に沿って、変数名には名詞を、関数名には動詞を名付けるものである。しかし、なでしこは、動詞と名詞の関数が混在している。そのため、変数名と関数名と命令名の区別がつきにくい。

動詞の関数

A を B で正規表現区切る

…　文字列 A をパターン B で区切って結果を返す

A で B をファイルストリーム開く

…　ファイル名 A をモード B(作|読|書|排他）でストリームを開き

ハンドルを返す。

名詞の関数

A の配列要素数

…　配列 A の要素数を返す

S から CNT 文字右部分

…　文字列 S の右から CNT 文字分を抜き出して返す

A と B の日数差

…　日付 A と B の差を日数で求めて返す

(v) 助詞について

なでしこには、28 種類もの助詞が用意されている。(とは、は、について、ならば、なら、でなければ、から、まで、までを、までの、で、を、の、が、に、へ、と、して、だけ、くらい、なのか、として、より、ほど、など、って、では、て、)これを使うことで、変数と命令、変数と関数が日本語らしく繋げられるようになっている。

「りんご」と表示する。

ファイルから拡張子抽出する。

つまり、命令や関数に引数を与える際は、助詞を指定する必要性がある。例のように引数の種類さえ合っていれば、順序は問われない。かなり自由な記述が可能となっている。

変数 A に 5 を足す。

５を変数 A に足す。

このとき、助詞の種類が多過ぎて、何の命令に何の助詞が使えるのか判断しにくい。例のように、日本語として正しい助詞が使えなかったり、不自然に助詞が使われているものもある。

○パス A をファイル B にパックファイル作成
×パス A をファイル B へパックファイル作成
…「ファイルパス=パック名=暗号化（0or1）」のリストを使ってファイル B へ保存する

○ S に母艦タイトル設定
× S として母艦タイトル設定
× S で母艦タイトル設定
…メインフォームのタイトルバーのテキストを変更する

なでしこは、これらの助詞を区切りと見なし、トークンに分割している。そのため、変数名に助詞を含む単語は使えないという制約が生じている。上の例だと、「に」が助詞であると判断されて、エラーとなる。下の例だと、「の」が助詞であると判断されてエラーとなる。解決策としては、変数を「」で括る、もしくは、わざとカタカナを用いて表記するしかない。

かにみそ とは 変数
りんごの値段 は 200 円

（vi）変数「それ」について

なでしこでは、引数を省略したとき、変数「それ」に直前の処理の値が代入される。さらに、「それ」をも省略することが出来る。例のプログラムの実行結果は、共に「トマト」と表示される。

「とまと」をカタカナ変換。
それを表示

「とまと」をカタカナ変換。
表示

なでしこの日本語らしさは、対象と処理方法を助詞で結び、具体的に記述できることで実現されている。これと、前節で述べた、おまじないのような宣言を省略することと合わせて、アルファベットのプログラミング言語が持っていた、記号性を無くすことに成功している。意味が読み取りやすく、ある程度まで、日本人の思考順のままに記述することができる。

（vii）日本語らしいプログラム

list40 は、なでしこのプログラム投稿掲示板から引用してきたプログラムである。ソフト名は「色表示ゲーム」、作者は「浩子」である。

これは、例えば「赤」という文字を青色で表示し、何色で表示しているかをユーザに当てさせるゲームである。このソースの問題点は、ただ上から下へ実行したい順に書いていること、そして記述が重複しており、構造化できていないことである。

各色の表示ごとに、質問して、答えを入力させて、正誤判断し、正誤に応じて結果を表示するという、一連の流れは共通である。この流れを抽象化し、どんな元データ（この場合は、色）を設定した場合でも使えるように、サブルーチンにするべきである。文字書体と文字サイズは、1 度記述するだけで、以降の「～と表示する」という命令全てに適用される。何度も書く必要は無い。

日本語の文章としては、何をしているのかが具体的で分かりやすく、

自然な文章となっている。しかし、文章としての自然さを優先させた結果、構造化には至らなかったと考えられる。

6　現在の日本語プログラミング言語の問題点

ここまで、日本語プログラミング言語に対する比較と分析を行ってきた。それらを踏まえた上で、日本語をプログラミングに利用することの問題点を述べる。

(1) 指導者層とサンプル・資料の不足

どの言語も新しいということと、実際のユーザ層から、上級者が少なく初心者ばかりという、不安定なユーザ層になっていると言える。プログラム投稿掲示板などのコミュニティにおいて、初心者を誘導する立場の者が少ない。

また、サンプルや資料が不足している。特に、なでしこは事務作業を簡略する用途を想定している。そして便利な使い捨てのプログラムばかりが作成され、プログラム掲示板に投稿される。それらは、初心者がプログラミングを学ぶための良い手本ではない。

(2) 日本語らしさの追求による独自性

省略や、独自の記述方法をもって日本語らしさを実現している言語は、簡単で便利である。しかし、日本語らしさを強調するあまり、他の言語を学ぶ際に習得の妨げになりかねない。その言語だけを使うのならば問題無いだろう。

(3) プログラミングの基本的な思考の軽視

構造化をするための要素は、日本語プログラミング言語にも揃っていることを第 3 章にて述べた。しかし、前出のプログラムでは動けば良いとばかりに、BASIC や Fortran に見られるような手続き式で作成されている。これは、日本語化することによって、個別の命令を具体的に記述できてしまうからではないだろうか。具体的に記述できるということは、抽象的に思考することを妨げる可能性がある。抽象化できないということは、構造化できないことに繋がる。プログラミングを組んで動かすことが出来ても、能力は向上しない。

つまり、日本語らしさを追求するプログラミング言語は、便利である代わりに、作業を抽象化しづらい言語であると言える。「省略可能」が可能な実用向け日本語プログラミング言語で学習すると、強引に作る癖を身につけてしまうのではないだろうか。プログラミング言語とは、決められた言語で手順を書くものである。それを日常的な言語に近い感覚で記述すると、プログラミング本来の書き方が出来なくなってしまう。

ただし、例外としてドリトルや言霊 on squeak のような低年齢者を対象とした教育用言語は、構造化について重視していないと考えられる。何故ならば、教育指導要領に基づいて、言語の習得よりも実行させることを第一の目標にしているためである。つまり、短時間の授業の中で、複数の学生が簡単にプログラムを組んで、動かせる言語でなくてはならない。こうした小学生や中学生といった年齢層を対象にした言語は、文法的制限が多く、プログラミングの本質からは程遠い。

(4) 構造化について

何事も、初心者は基本の手法を学ぶことから始める。プログラミングを習得するということは、ただ実行できるものを作れるようになることではなく、より良いプログラムを組めるということだ。そのために、構造化プログラミングを学ぶべきである。元々この手法は、大きなプログラム作成のために考えだされた。今日では、小さなプログラムを組む際にも考慮すべき基本的作法となっている。

> 大きな計算処理を小さな仕事に分割し、最初ゼロから始める代わりに他人の作成したものを組み合わせてプログラムを作るには関数を使うのがよい。関数をうまくつくれば、知る必要のないプログラムの各部の操作の詳細を隠すことができるし、全体を明確化し、プログラム変更を容易にすることもできる
> (18)。
>
> プログラムに関しても、構造を考慮に入れず、テストによってわずかの疑いもなく正しさを保証するのは、全く望みがありません。いい換えれば、プログラムの正しさの保証のでき具合は、プログラムの外部仕様と動作だけでなく、その内部構造に決定的に依存しているのです
> (19)。

E.W. ダイクストラが述べたように、良いプログラムとは、正しく動作するプログラムのことである。そして、動作の正当性を証明しやすくするための、分かりやすい構造を持つ。

プログラムは順次、反復、分岐のいずれかの制御構造から成る。そしてプログラムは階層構造を持ち、プログラムの正当性は各階層において確認する。データと機能を分割し、サブルーチンに機能を抽象化する。

こうすることで、ミスを減少させ、生産性・能率・保守性を向上することができる。

実用性のある日本語プログラミング言語では、ソフトを作成できる。ソフトを実行させるならば、ソフトの正しい動作は必須である。よって、日本語プログラミングにおいても、構造化を心掛ける必要があるだろう。

7　結論

ここまで分析した結果、日本語らしさを追求した言語ほど、プログラミングを習得するのには向いていないと断言できる。やさしそうに見えて、便利であるが、けして初心者向けではない。

プログラミングの全てを日本語で記述することは、無理がある。日本語の分かりやすさを最も活かすためには、DNCL のような疑似コードがもっとも最良だと思われる。構造化のできるアルファベットのプログラミング言語に繋げることで、日本語で分かりやすく学びつつ、プログラミングの基本的な考え方が身につく。

とはいえ、けして、実用向け日本語プログラミング言語は無駄ではなく、面白い試みだと思う。特に、取り組みやすさは今までのアルファベット言語に無かった要素である。ぜひ教育用プログラミング言語にも取り入れて行くべきだ。

最後に、自分自身がプログラミング初心者に毛が生えたようなレベルなために、深い分析が出来なかったことが悔やまれる。

注

（1）G-BASIC　http://ja.wikipedia.org/wiki/BASIC

（2）Mind　http://www.scripts-lab.co.jp/mind/whatsmind.html

（3）なでしこ　http://nadesi.com/

（4）ひまわり　http://kujirahand.com/himawari/

（5）TTSneo　http://tts.s28.xrea.com/

（6）言霊　http://www.crew.sfc.keio.ac.jp/projects/kotodama

（7）ことだま on squeak　http://www.crew.sfc.keio.ac.jp/squeak/

（8）ドリトル　http://dolittle.eplang.jp/index.php

（9）DNCL http://www.dnc.ac.jp/old$_{d}ata$/$exam_{r}epo$/17/pdf/17$hyouka$38.pdf

（10）PEN　http://www.media.osaka-cu.ac.jp/PEN/

（11）http://journal.mycom.co.jp/news/2004/11/08/001.html

（12）2004 年　言霊コミュニティサイトの開発 xoops を使ってコミュニティサイトを作る　岡田健　有田直弘　橋山牧人

（13）なでしこ命令　http://nadesi.com/doc/cmd/doc.cgi?mode=cmdid=2303

（14）なでしこツール作成講座　http://nadesi.com/doc/kouza/gui/index.htm

（15）復刊ドットコム　日本語プログラミング言語「なでしこ」公式ガイドブック　http://www.fukkan.com/fk/VoteDetail?no=35913

（16）ドリトルで制御　http://www2.wbs.ne.jp/ kure/dl/dolittle.htm

（17）MYCOM ジャーナル　日本語で 10 行プログラミング
http://journal.mycom.co.jp/column/nihongoprog/

（18）B.W. カーニハン/D.M. リッチー　著石田晴久　訳『プログラミング言語 C』共立出版　1989 年 6 月 15 日　第 2 版 1 刷発行

（19）E.W. ダイクストラ/C.A.R. ホーア/O.-J. ダール　共著　野下浩平/川合慧/武市正人　共訳『構造化プログラミング』サイエンス

——日本語プログラミング言語の可能性について——

社　昭和 50 年 5 月 20 日　初版発行