RA-003

日本語らしい日本語プログラムを書くために
Searching for the Advantages of Japanese Programming Languages

馬場 祐人† 筧 捷彦‡
BAMBA Yuto KAKEHI Katsuhiko

1. はじめに

日本語プログラミング言語は，変数や関数を日本語で名付け，また日本語の表記および語順に近い文法でプログラムを書くプログラミング言語である。日本語でプログラムを書くことの利点として，日本語で書かれた設計を英単語へ置き換えることなく直にプログラムで表現できる点が挙げられる。

現在の日本語プログラミング言語の多くは，プログラム文を自然な日本語の表記および語順に近づけて書けるようになることを目指して設計されている。プログラムを母国語で書くことができることから，日本語プログラミング言語は，プログラミング入門や事務処理における単純なバッチ処理を行うことを目的としたものがほとんどである。

筆者らは日本語プログラミング言語のプログラム構文だけではなく，識別子の名付け方(命名規約)，関数定義およびクラス定義を含めたプログラム設計全体で，日本語らしい日本語プログラム(以下，単に日本語プログラムという)の書き方を検討していく必要があると考える。その理由は，既存の日本語プログラミング言語は，プログラム構文レベルの工夫で自然に書けることに重きがあり，日本語で書くことを踏まえたプログラム設計について深く考えられたものがないからである。今後は日本語での設計をそのまま日本語で書くことができる利点を活かすことができる日本語プログラム設計を考える必要がある。

本研究の目的は，日本語プログラミングの利点や価値を求めて，日本語らしい日本語プログラムを書くための方法を示すことである。筆者らは現時点での日本語プログラミング言語の構文を使って実用的に使用されるプログラムを作成し，その過程で日本語らしいプログラム設計を模索している。

筆者らは，実際にいくつかのソフトウェアを実装し，実例を通じて日本語らしい日本語プログラムを書くための方法を模索している。その過程で日本語プログラミング言語を使ってミニコンパイラを作成した。本稿では，日本語らしいプログラム設計方法を示し，その方法を適用した日本語プログラムの一例を報告する。

† 早稲田大学 理工学術院 基幹理工学研究科
Graduate School of Fundamental Science and Engineering, Waseda University
‡ 早稲田大学 基幹理工学部
Faculty of Science and Engineering, Waseda University

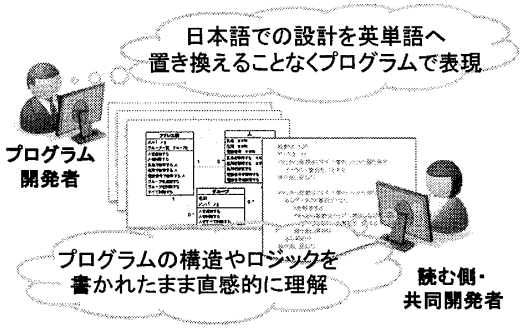
2. 日本語プログラミング言語

2.1. 日本語でプログラムを書く利点

日本語で考えた設計を，語彙を英単語へ変えずにそのままの形でプログラムに落とし込むことできることが日本語でプログラムを書くことの利点である。

ソフトウェア開発において分析段階では仕様書に日本語で機能名やデータ構造を表記する。プログラム実装段階では，識別子(クラス名，関数名，変数名)を仕様書の日本語表記から英単語へ置き換えてプログラムを書く。これでは仕様書に書かれた日本語と，識別子との間にギャップが起こる。プログラムを日本語で一貫して書くことができれば，このギャップを解消することができる。

またプログラム作成者とそのプログラムを読む共同作成者との意思疎通において利点がある。プログラム作成者は，日本語で書かれた設計書の言葉を直にプログラムで表現できる。共同作成者は，プログラムの構造やロジックを書かれたまま直感的に理解できる。日本語を使ってプログラム書くことによる効果についての研究[1]では，『仕様書とプログラムの対応の良さはプログラムの可読性や保守に寄与』すると結論づけられている。



最近のプログラミング言語処理系は日本語の識別子を使うことができる。ただ予約語や API 群には英単語が使われていることから，ソフトウェア開発者が実装するクラスやメソッド，変数だけを日本語で名付けても，英単語と日本語が混在したプログラムとなり可読性に欠ける。日本語プログラミング言語では，プログラム構文の段階で日本語らしく書けることを目標としており，一貫して日本語でプログラムを書くことができる。

2.2. 既存の日本語プログラミング言語

プログラムを日本語で書くことができる処理系は，以前から存在する。“まほろば[2]”は，『通常我々が読み書きしているように分かち書きをしない表現

方法を採用し日本語として不自然さがない言語仕様』となっている。実証として実際にまほろばを用いてコンパイラを実装している。ところがオブジェクト指向プログラミングには対応していないなど，現在のソフトウェア開発には適用できない点がある。また，教育用プログラミング言語“ドリトル[3]”や“PEN[4]”も，日本語表記の言語仕様を採用している。特にドリトルはプロトタイプ型のオブジェクト指向プログラミング言語である。これらは，教育教材としての比較的短いプログラムに限れており，実用用途で比較的規模の大きいプログラムが作られた例は見当たらない。

2.3. 現在の日本語プログラミング言語の問題点

日本語プログラミング言語の構文は，C 言語や Java などの構文や関数名を単純に日本語に置き換えるだけでは，利点や魅力は数が限られる。例えば，リスト 1に示すプログラムは，Java のプログラムと，それを単純に日本語に置き換えたプログラムである。

リスト 1 日本語らしさを活かしていないプログラム

Java のプログラム

```
System.out.print("こんにちは");
```

日本語に置き換えただけのプログラム

```
システム.出力.印刷("こんにちは");
```

このように，変数やメソッドを単純に日本語に置き換えることは，プログラミング初心者のプログラムに対する抵抗を和らげることに有効である。しかし，プログラミングの習熟者には，利点が少ない。構文や関数名を覚えることは英単語であっても変わらない，日本語入力の手間が掛かるからである。

日本語プログラミング言語は，構文を日本語らしく書けるように工夫することだけでは，日本語で書くことの利点は活かされない。日本語でプログラムを書くことの利点を活かすためには，日本語らしい命名規約やプログラム設計などを日本語らしく書くための方法を検討していく必要がある。それらを踏まえて日本語プログラミング言語仕様も洗練していくべきである。

3. 日本語らしく書くための検討方針

3.1. 日本人であれば理解できる

“日本語らしさ”には，主観があるため厳密に定義することは難しい。筆者らは，次の点を考慮すれば，日本語らしく書くことができると考える。

違和感のない日本語　プログラム文は，日本語文と同じ語順に書き，普段使う日本語として違和感のない表現で書く。適宜，日本語文法に関する文献[6]を参考にして検討する

動作や役割に的確な名前　関数名や変数名は，プログラムの機能や役割を明確に表す名前を付ける。ソフトウェアの仕様書で使用する表現をそのままプログラムで用いる

日本語プログラムは，これらの点を踏まえて書くことができれば，日本語らしくプログラムを書くことができると考える。なお，日本語らしくプログラムを書くことを考慮すると，プログラム文は冗長になる。しかし，開発環境の入力補完などを用意することでその問題は軽減されると考え，本稿では議論しない。

3.2. 使用する日本語プログラミング言語

筆者らは，日本語プログラミング言語“プロデル[5]”を開発した。日本語として違和感のない表現でプログラムを書くために，適宜構文を改良しながら，プログラム設計を検討していく。ここで，プロデルのプログラム例を挙げる。リスト 2は，プロデルにおける関数呼出し文のプログラム例と，それに相当するプログラムを Java で書いたものである。なお，以下プロデルのプログラムでは，関数名を**太字**で表し，キーワードを下線で表す。

リスト 2 Java とプロデルで書かれたプログラム

```
// Javaの場合
System.out.println(gcd(1071,1029));

public int gcd (int value1, int value2) {
        if( value2==0 ) return value1;
        int newValue= value1 % value2;
        return gcd(value2, newValue);
}
```

```
／／プロデルの場合
1071と1029の最大公約数を表示する

【値1】と、【値2】の、最大公約数を求める手順
   もし値2が0なら、値1を返す
   新しい値は、値1を値2で割った余り
   値2と新しい値の最大公約数を返す
終わり
```

リスト 2の Java のプログラムで示すように，一般的に関数呼出し文は，関数名および実引数(必要な場合)を書く。プロデルの関数呼出しも同様に関数名と実引数を書くが，Java と異なる点として，実引数と仮引数を結びつけるために“助詞”を書く点がある。実引数の直後に添えられた助詞によって，実引数と仮引数が対応づけられる。これによって，補語(実引数+助詞)の順番に関係なく，関数へ正しく引数を渡すことができる。リスト 2を見るように，プロデルの文は，複雑な構造ではなく，補語(名詞+助詞)と動詞で構成されている単純な文である。

3.3. 動作や役割に的確な名前

日本語らしい日本語プログラムを書くためには，構文設計の他に，次に挙げるような点を日本語らしく書く方法を考える必要がある。C 言語や Java などの英単語記述を基本とするプログラミング言語に慣れているプログラマは，それらの言語の名付け方や関数定義を，単に置き換えて日本語プログラムとして書きがちである。日本語プログラムを書くにあたっては，ライブラリ関数や変数名の単純な翻訳ではなく，関数が意味する機能の役割を日本語で名付けることを考慮して日本語プログラムを書くことが望ましい。

1) 変数の名付け方

変数は，名詞で命名することが望ましい．局所的な変数は，“番号”，“値”などの抽象的な名称で構わない。広域変数やメソッドの仮引数には，それが持つ値の役割がわかる明確な名称を付けることが望ましい。

2) 関数の名付け方

関数は，動詞で命名する。オブジェクト指向において関数(メソッド)は，オブジェクトの振る舞いであるから，メソッド名は自然と動詞になる。プログラムを読む際にも，動詞で書かれた部分があれば，それが関数呼出しであることは違和感なく理解できる。

4. 日本語らしいプログラム設計

4.1. プログラム設計の提示

前節で述べた検討方針をもとにいくつかの日本語プログラム開発を実践した。この節では，日本語プログラムを設計するにあたり考慮すべき点を提示する。具体的には，変数や関数の命名規約，関数定義およびクラス定義である。

4.2. 命名規約

Java の仕様書では推奨する命名規約が書かれている。命名規約は言語仕様として拘束されるものではないが，API 群はこの命名規約に従っている。英単語では Camel 表記や Pascal 表記といった文字の大小で区別できるが日本語では別の方法で区別しなければならない。そのために日本語プログラムにおいても次のような命名規約を決めた。

配列型の変数名　名前の末尾に“一覧”を付ける
インスタンス名　名前の末尾に“クラス名”を付ける

4.3. 関数呼出し文

日本語は文脈上で明らかな主語や目的語を省略することから，同じ変数を引数として渡す関数が複数あるプログラムは，日本語として違和感がある。例えば，リスト 3に示すプログラムは，Java とプロデルで書いた同じ変数が何度も使われる関数呼出しのプログラム例である。“リスト”変数が何度も表れるから，日本語として違和感がある。

この違和感を解消するために同じ変数が何度も使われる関数呼出しは，その変数を広域変数として宣言し，その上でラッパ関数を用意する方法が考えられる。リスト 4は，リスト 3のプロデルのプログラムから“リスト”変数を省略したプログラムである。

リスト 3 同じ変数が何度も使われる関数呼出し

```
//Java
list.clear();
list.add("Hello");
list.add("World");
list.removeAt(1);
```

```
//プロデル
リストを【すべて消す】
リストへ「Hello」を【加える】
リストへ「World」を【加える】
リストから 1 番目を【消す】
```

リスト 4 ラッパ関数

```
【すべて消す】
「Hello」を【加える】
「World」を【加える】
1 番目を【消す】
```

リスト 5 ラッパ関数の定義

```
【すべて消す】手順
    リストを【すべて消す】
終わり
【【内容】を、【加える】】手順
    リストへ【内容】を【加える】
終わり
【【番号】番目を、【消す】】手順
    リストから【番号】番目を【消す】
終わり
```

4.4. 関数の引数

引数を 3 つ以上持つ関数を，日本語らしく書くことは困難になることがある。引数が多いと，一文で表現することができないからである。なお，プロデルは，関数呼出しを一文で書く。“○○して□□する”という書き方は複文となり書くことができない。日本語らしく書くには，関数に指定する引数を必要最低限に減らす工夫が必要である。

例えば，Java の Graphics クラスの drawLine メソッドは，次に示すようにメソッド呼出し文を書く。

```
g.drawLine(x1, y1, x2, y2);
```

これを日本語プログラムに直訳すると，

```
x1 と y1 から x2 と y2 まで【線を描く】
```

と書ける。しかし，日本語として違和感がある。この違和感を解消するために，関数の引数を減らすこ

とを考える。このメソッドの場合は，始点座標と終点座標を指定する代わりに，Java の java.awt.Point クラスに相当する，“点”クラスを用意し，引数次のように書くことで日本語らしいプログラムになる。

点 1 から点 2 まで**線を描く**

“点 1”と“点 2”は，Java の java.awt.Point クラスに相当する，“点”型の変数である。このように関数に渡す引数を減らすだけで日本語らしいプログラムになる。

このように日本語プログラムを書くにはクラス設計を含めて考えなければならない。

5. 日本語プログラムによるコンパイラ作成

5.1. コンパイラの概要

筆者らは，日本語らしいプログラム設計を模索するために，プロデルの現時点での構文や4で述べたプログラム設計を使って，C 言語風のコンパイラを作成した。それを通して，日本語らしいプログラムの一例を報告する。

C 言語風のコンパイラを，WL コンパイラと呼ぶ。WL コンパイラは，早稲田大学情報理工学科の言語処理系に関する講義の教材で使われている C 言語風のコンパイラである。WL コンパイラの構文解析は，再帰下降構文解析で実装している。この言語処理系を選んだ理由として，再帰下降構文解析が人間の目で処理を追いやすい，講義資料の中に日本語で書かれた言語仕様書が含まれているからである。なおプログラムコード行数は，1100 行程度である。

5.2. 変数や関数の名付け方

変数名や関数名は，仕様書を元に名付けた。リスト 6は，WL コンパイラの構文解析部の一部を C 言語で書いたものとそれに相当するプログラムをプロデルで書いたものである。関数名について注目する。リスト 6の C 言語のプログラムで宣言されている関数は，arithmetic と名付けられている。この関数に相当するプログラムをプロデルの関数として書こうとすると，日本語に直訳して，“算術”と名付けがちである。

2)で述べたように，日本語らしいプログラムを書くには，関数を動詞で命名するべきである。そこで，講義資料にある言語仕様書(図 1)を参考にして“算術式を評価する”と名付けて定義した(リスト 6)。

算術式 (arithmetic expression) **の評価は，算術式が算術項単独の形であるなら，その算術項を評価し，その結果を評価の結果とする。そうでなければ，すべての算術項を左から順に右辺値解釈し，加減演算子** (additive operator) **が示す加減演算を左から順に行って評価の結果とする。**

図 1 講義資料にある言語仕様書の一部

リスト 6 WL コンパイラのプログラムの一部

```
//C 言語の場合
void arithmetic() {
  arith_term();
  while( token==Plus || token==Minus ) {
    int t; t= token;
    get_token();
    arith_term();
    if( t==Plus )
      gen_op("+");
    else if( t==Minus )
      gen_op("-");
  }
}
```

```
//プロデルの場合
算術式を、評価する手順
  算術項を評価する
  (「+」と合致する)または(「-」と合致する)の間、繰り返す
    【演算子字句】は、現在字句
    次字句へ進める
    算術項を評価する
    もし[演算子字句が「+」と合致する]なら
      加算命令を書き込む
    その他でもし[演算子字句が「-」と合致する]なら
      減算命令を書き込む
    もし終わり
  繰り返し終わり
終わり
```

リスト 7 仮想機械語を出力するプログラム例

```
C#      : Label label = il.DefineLabel();
プロデル : ilへlabelというラベルを書き込む

C#      : il.Emit(OpCodes.Add);
プロデル : ilへ加算命令を書き込む

C#      : il.Emit(OpCodes.Ldc_I4, value);
プロデル : ilへvalueという整数定数を書き込む
```

5.3. 仮想機械語生成ライブラリ

プロデルで作成した WL コンパイラは，入力プログラムを解析後，仮想機械語を出力する。仮想機械語は，MSIL(Microsoft Intermediate Language)を用いる。そのために，仮想機械語を生成するためのプロデルの外部ライブラリを用意した(図 2)。このライブラリは，.NET Framework に含まれる System.Reflection.Emit 名前空間に含まれるクラス群のラッパクラスである。このライブラリの設計にあたり4で述べたプログラム設計を適用した。

System.Reflection.Emit 名前空間に含まれる ILGenerator クラスは，仮想機械語を出力するクラスである。このクラスをプロデル向けにラップした“仮想機械メソッド”クラスを用意した。日本語らしいプログラムの模索に当たり必要に応じて，この“仮想機械メソッド”クラスを中心に改良を行った。リスト 7は，仮想機械語を出力するプログラムを C#とプロデルで書いた場合のプログラム例である。

メソッドは，MSDN ライブラリの日本語訳のドキュメントを参考に名付けた。例えば，“il.Emit(OpCodes.Add);”に相当する文を，プロデルでは“il へ**加算命令を書き込む**”と書くようにした。ILGenerator クラスの Emit メソッドの説明では，“指定された命令を命令のストリームに書き込みます。[7]”と書かれている。そこで，ライブラリの設計時にはそれに従って動詞を“書き込む”とした。また4.4で述べたように，引数の数が多くなると，日本語として違和感がある。引数を減らすために，C# の Emit メソッドでは第一引数に，出力する仮想機械語命令の種類を書く。プロデルでは，出力する仮想機械語命令の種類も含めた“**加算命令を書き込む**”と名付けた。

5.4. 同じ変数を使う関数呼び出し

リスト 8に示したプログラムは，WL コンパイラの機械語を生成する一部である。プロデルのプログラムの一部を示す。このプログラムは，“メソッド”変数が何度も表れる。文脈上，書き込む対象は明らかに“メソッド”であり，4.3で述べた関数呼出しの工夫が適用した。適用したプログラムをリスト 9に示す。リスト 9は，リスト 8に比べて文が簡潔になり日本語として違和感がなく書くことができる。

6. まとめ

本稿では，日本語プログラミング言語を使った日本語らしいプログラム設計方法を示し，ミニコンパイラ作成を通して，その方法を適用した日本語プログラムの一例を報告した。日本語プログラミング言語は，プログラムの構文のレベルだけで日本語らしく書けることを目標にしていてはいけない。日本語らしい日本語プログラムを書くためには，命名規約や関数定義およびクラス定義を含めたプログラム設計全体を議論する必要がある。これらを踏まえて日本語プログラミング言語仕様も洗練していくべきである。筆者らは，コンパイラ作成に限らず，業務アプリケーションや Web アプリケーションなど実際に作られる，様々なケースにおいて日本語でプログラムを作り込んで，日本語らしい日本語プログラムの書き方を模索する必要があると考えている。

リスト 8 WL コンパイラの機械語生成部の一部

```
その他でもし「>=」と合致するなら
    次字句へ進める
    算術式を評価する
    メソッドへ小なり比較命令を書き込む
    メソッドへ 0 を整数定数として書き込む
    メソッドへ等価比較命令を書き込む
もし終わり
```

リスト 9 適用後のプログラム

```
その他でもし「>=」と合致するなら
    次字句へ進める
    算術式を評価する
    小なり比較命令を書き込む
    0 を整数定数として書き込む
    等価比較命令を書き込む
もし終わり
```

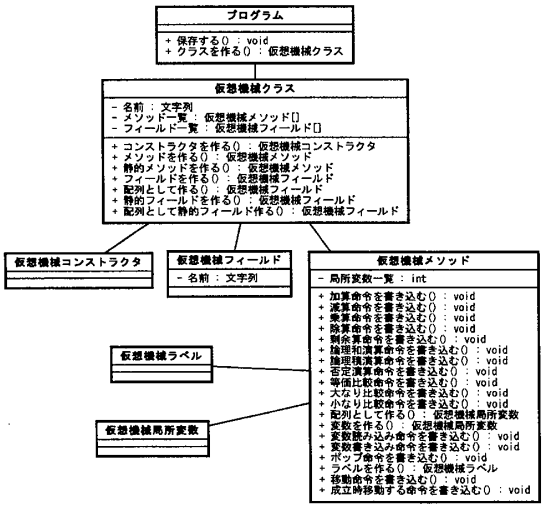


図 2 機械語生成ライブラリのクラス図

参考文献

[1] 中川正樹ら，“日本語プログラミングの実践とその効果”,情報処理学会論文誌 Vol.35,No.10(1994)

[2] 今城哲二ら，“日本語プログラム言語“まほろば”の言語仕様”，情報処理学会研究報告,2000-SE-130, pp.143-152,2001-3.

[3] 中谷多哉子, 兼宗進ら，“オブジェクトストーム：オブジェクト指向言語による初中等プログラミング教育の提案”. 情報処理学会論文誌 Vol.43 No.6 (2002).

[4] 西田知博ら，“初学者用プログラミング学習環境 PEN の実装と評価”. 情報処理学会論文誌 Vol.48 No.8 (2007).

[5] ゆうと，“日本語プログラミング言語「プロデル」”，http://rdr.utopiat.net/，2010 年時点.

[6] 三上章，“象は鼻が長い―日本文法入門”，くろしお出版 (1960).

[7] MSDN Library, “ILGenerator メンバー”, http://msdn.microsoft.com/ja-jp/library/system.reflection.emit.ilgenerator_members.aspx, 2010 年時点.