プログラミングの本質的難しさと日本語／英語プログラミング言語の比較検討
—高大接続のプログラミング教育を考える—

綾　皓二郎[*1]
Email: aya.k2015h27@gmail.com
*1：みやぎインターカレッジコープ

©Key Words　　高校教科「情報」，教育用プログラミング言語，英語の発想

1. はじめに

高校の情報教育は 2022 年度からの新学習指導要領の下で「情報I」（必修）と「情報II」（選択）に再編され，両科目にプログラミングの単元が含まれ，高校生全員がプログラミングを学ぶことになる。本報告では，高校のプログラミング教育は，多数の日本語／英語プログラミング言語から何を基準にして選べば適切と言えるかを議論する。

この論文では第一に，プログラミングの特性を考察し，プログラミングの本質的難しさは言語にはよらないことを指摘する。第二に，日本語と英語プログラミング言語を比較してそれぞれの特徴や発想，語順，表記などの違いについて検討する。

結論として，高校のプログラミング教育においては英語プログラミング言語を採用し，大学のプログラミング教育と滑らかな接続を目指すことが望ましいことを述べる。

本報告でいうプログラミング教育とは，プログラマーの養成を目的とする教育ではない。アルゴリズムを考案・構築してプログラミムを作成し実行する過程のなかで，プログラミング的思考を育み，コンピューター科学とは何かを学ぶ教養教育である。

2. プログラミング言語の特性

2.1 自然言語によるプログラミング

自然言語のままではコードを一意的な機械語に変換することはきわめて難しい。その理由は，自然言語の表現の曖昧さにある。また，表記に冗長性がある自然言語のままでは，プログラミングにおいて論理的な思考と思考の節約を高めることも容易ではない。自然言語のままでは，プログラムは作成できても全体的に可読性が低いものになってしまう。数学の場合でも，数式を使って冗長さを避け，思考の節約と可読性を高めている。そこで，自然言語類似ではあるが，冗長さの少ない「プログラミング言語」を用いてプログラムを作成するようになった。プログラミング言語の中で COBOL は英語に近い記述でプログラムを作成できるといわれている。しかし，プログラムは冗長となることが多い。COBOL の冗長さは，英語を母語とする現代のプログラマーに歓迎されていない。このように自然言語に近いプログラミング言語が優れているとは必ずしも言えないことにまず注意すべきである。

2.2 プログラミング言語の文法

どの言語にも程度の差はあるが，次の特徴がある。

・文法は自然言語よりは，はるかにやさしい。
・文法は明確で曖昧さがなく，例外がない。
・構文の種類は少なく，各構文は短く冗長さはない。
・語彙は極めて少なく，単語の意味は一義的に決まっている。
・構文に数式と記号を使い，コードを略記する。

コードを略記法で記述することについては，初級の学習者には低い可読性であり思考の妨げとなるが，逆に中級以上の学習者には高い可読性であり思考の節約となる。この学習壁を乗り越えるプログラミング教育が高校情報教育の大きな課題である。

2.3 プログラミングの第一要件

アルゴリズムおよびプログラミングに第一に求められていることは「抽象化」である。プログラミングの発展は抽象化という方向に進んでいる。抽象化がプログラムの可読性を高め，プログラミングの効率化，再利用性，信頼性，保守性を高めることに大きく関わっている。

プログラミング教育の到達目標の一つは，どの言語であれ，個々のプログラムを作成する中で，この抽象化の概念を理解し身に付けることである。

2.4 プログラミングにおける本質的難しさ

プログラミングの本質的難しさはどのプログラミング言語を使っても共通に認められる（下記）。

・課題を緻密に分析して抽象化し，定式化して適切なアルゴリズムを構築する難しさがある。
・プログラミングにおける概念・用語を理解する難しさがある。たとえば，抽象化，構造化，変数，配列，手続き，関数，メソッド，オブジェクトなど。
・プログラムは，順次，条件分岐，繰り返しの基本構造から成り立つが，自然言語の文章に現れることのない多重の条件分岐や繰り返しのコードブロックが書けないと大きなプログラムは作れない。この多重性がプログラミングを難しくしている。
・言語の文法に従って事細かにコードを書いてコンピューターに指示を与えなければならない。
・表記と思考の節約のため，多くの記号を使うことによる難しさもある。
・実行効率の高いプログラムを作成するためには，冗長なコードをそぎ落とす難しさがある。

すなわち，プログラミング学習の難易度は，言語が何かということよりも，アルゴリズムの構築とプログラミングの特性に大きく依存する。

2.5 プログラミング言語の大きな違い

プログラミング言語には，元となる自然言語の発想と語順，表記に大きな違いが認められる。英語プログラミングが難しいのは，日本語で考えたアルゴリズムを，英語を用いてプログラムとして具現化しなければならないことにあると考えられている。英語のプログラムを解釈する際には，日本語の語順に変換する必要があることが指摘されているが[1]，英語は語順どおりに直読直解すればよいとすれば，その必要はないことになる。

2.6 実用言語と教育用言語

英語実用言語を使って初心者がプログラミングを学ぶ場合には，英語を基本としているところに，
① 新しい言語を理解すること
② それを使って仕組を作ること
という初めての作業を同時に行なわなければならない。従来の実用言語を用いたプログラミング教育では初心者のほとんどは ①の段階で挫折してしまう。実用言語のプロ仕様の文法を理解することが難しいからである。それ故，英語実用言語を学ぶことは勧められない。代わって，日本語プログラミング言語を用いれば，①の負荷を大きく軽減でき，②に専念できるはずだという考えのもとで，日本語教育用言語の開発と授業実践が行われている[1],[2]。

他方，①の負荷は，英語実用言語の仕様を教育的立場から制限しても同様に軽減することができる。同時に英語の負荷を軽減するために，英語の発想を理解し英語プログラムを直読直解する学習を習慣とする。ただし，補助として翻訳ツールなどを使うことを学習の途中段階では妨げない。

2.7 教育用プログラミング言語の要件

① 学びやすく取り扱いやすい言語であること。プログラムを書けば直ぐに実行できて結果が得られるインタープリター型のスクリプト言語がよい。
② 実行環境は，インストールが不要で Web ブラウザ上で使える環境であることが望ましい。
③ 学習支援環境と教材が Web ブラウザ上に数多く用意されていること。
④ 大学の一般情報教育に円滑に接続できるばかりでなく，専門教育に進んでも躓くことのない発展性・将来性があること。
⑤ オープンソースのソフトウェアで，非営利の組織による開発・保守・支援の体制が確立できていることが望ましい。

3. 日本語／英語プログラミング言語の比較

3.1 日本語プログラミング言語を用いる長所

日本人は日本語プログラミング言語で学べという主張がある[1],[2]：「日本語が母語であれば，ものごとは日本語で考える。アルゴリズムも日本語で考える。ならばプログラミングも日本語でやるのが自然である」。小・中学校の教育では日本語を用いるほうがおそらく分りやすいが，高校教育以上にも当てはまるか，検討が必要と思われる。日本語プログラミング言語の長所として次のことが挙げられている。
① アルゴリズムを自然な日本語に近い形でプログラムとして表現できることが多い。
② 日本語の語順「何をどうする」（O+V 型）はコンピューターに対する命令に適し，効率がよい。
③ 日本語さえ分かればコードおよびエラーメッセージが読め，プログラムの可読性が高い。

3.2 日本語プログラミング言語を用いる長所の評価は適切か

① 日本語で記述したアルゴリズムを英語でプログラムとして表現することは，パラグラフや論文を書くことではないから，それほど難しいことではない。
② 数式は通常 O+V 型で表現されていない。O+V 型の命令に変換することは処理系に任せればよいことであって個々のプログラミングでやることではない。
③ 英語は語順変換して訳読しないと理解できないというのは，伝統的な英語教育による「刷り込み」であって，英語は英語の語順で理解すればよい[3]。
④ プログラムの可読性はコードブロック，プログラ

ム全体の可読性を問題としなければならない。英語であっても，必要ならば翻訳ツールを用いれば，コードやエラー原因の大体の見当はつく（後述）。

上記の日本語プログラミング言語の評価[1],[2] に関しては，以下の疑問もある。

・脳の負荷が小さいとされる O+V 型のプログラミング言語が英語ネイティブに受け入れられず，脳の負荷が大きいとされる V+O 型のプログラミング言語が主流を占めていることをどう説明するか。

3.3 日本語プログラミング言語を用いる短所

① 日本語プログラミング言語は，日本という小さな閉じた領域でしか通用しない。

② 大学で英語プログラミングを学ぶことに円滑に接続されない。

③ 日本語の発想は前置きが長く，話の核心がなかなか出てこない「後方／下方重心型」と言われる[4]。そのため，日本語プログラムの可読性は高いとは一概にいえない。

④ 日本語プログラミング言語には，表記・表現に問題があり，思考の節約が難しい。

・厳密な正書法がないからコードの表記を一義的に決められず，表記の曖昧さやゆれが許されている。
・表記・表現が冗長になり，プログラムが大きくなると本質的な部分を読み取ることが煩わしくなる。
・述語のないコードは不自然な日本語となる。
・プログラムの大部分で「かな漢字変換」を必要とし，文字の入力が面倒で時間がかかる。
・コードにローマ字とかな漢字を併用することは，エラーの原因となりがちである。

3.4 日本語プログラミング言語の現状

おおむね次の 3 つのタイプに分かれる。

① 小・中学校教育用言語 :「言霊」「ドリトル」など

② 高校教育用言語： x DNCL[5],「どんぐり」[6]など

③ 一般向け実用言語：Mind,「なでしこ」など

このうち，日本語の語順など，日本語の特徴にこだわって開発されているのは,「言霊」, Mind,「なでしこ」である。②は，大学入試センター試験『情報基礎』に使われているプログラミング言語 DNCL に準拠したものである。DNCL は入試用の疑似言語であるが，xDNCL には PEN という，「どんぐり」には同名の実行環境が用意されている。これらは実用英語プログラミング言語の仕様に制限を加えて日本語化したもので，日本語の特徴をプログラミングに生かそうとする考えは認められない。「どんぐり」では，英語表示と編集が可能となっている。ここまでやるのであれば，最初から実用英語プログラミング言語を制限して使ってプログラミングを教えればよいという考えが浮かばないのであろうか。

3.5 日本語プログラミング言語「どんぐり」

「どんくり」は，DNCL の仕様に加えて独自の拡張機能を用意している。教育用途を考えた場合には，変数名や命令語だけでなく記号や空白文字などを含めたプログラム全体で，学習者の母語に対応することが有効であるとしている。そこで，プログラムの中で「数字，英字，記号，空白」は，全角半角を区別せずに使える。たとえば, 掛け算には *, ＊, × を，割り算には /, ／ を使う。しかし，このような表記は入試の実行を伴わない疑似プログラミングでは許されても，実行環境において表記の大きなゆれを認めることには賛成できない。高校教育で日本語プログラミング言語のこのような書き方に慣れてしまうと，大学での英語プログラミング教育に好ましくない影響を与えることを恐れる。プログラミング言語は厳密な意味での正書法を用いることが望ましい。

具体例として,「倍数判定（）」プログラムがあるので，それを紹介する。

```
1: 倍数判定（）は
2:      x を 1 から 10 まで 1 ずつ増やしながら、
3:          x を改行なしで表示する
4:          もし x%3=0 ならば
5:              「<- 3 の倍数！」を表示する
6:          を実行し、そうでなければ
7:              改行を表示する
8:          を実行する
9:      を繰り返す
10: を実行する
11: 倍数判定（）
```

文字数は 166 である。さらに変数の型宣言も必要である。このプログラムを Python で記述すると

```
1: def is_multiple():
2:     for i in range(1, 11):
3:         if i % 3 == 0:    print(i, ' 3 の倍数', '¥n')
4:         else:    print(i, '¥n')
5: is_multiple()
```

文字数は 121 である。Python では変数の宣言を必要としない。繰り返しの for 文と条件分岐の if 文は，完全な文（sentence）になっていないが，コードとしてはこれで十全である。このプログラムを Google 翻訳させると（下記），プログラミングを学び始めの高校生でも何をしているか，大体の予想はつく。

```
1：def is_multiple（）:
2：範囲（1，11）の i の場合
3：i％3 の場合== 0：print（i，　'3 の倍数'，　'¥ n'）
```

4：それ以外の場合：print（i,　'¥ n'）
5：is_multiple（）

プログラムの確かな読解にはプログラム全体が簡潔であることが求められる。Python が各コードを含めてシンプルで可読性の高いことは自明であろう。

ここで，繰り返しに使われる for 文について説明する。英和辞書では for の意味として「時間／距離の範囲を表す」が載せられている。日常生活では，たとえば This train is bound for Tokyo. のように使われる。ここで for は，出発地から到着地までのすべての距離を含んでおり，これは for 文の表す意味と同じである。さらに Python では for が range()という範囲を示す関数といっしょに用いられている。この range も日常生活で用いられるものと意味に違いはない。以上は，英語の授業で学ぶことが英語プログラミングの学習に繋がっていることを明確に例示している。

3.6 英語プログラミング言語を採用する長所

① プログラミングに限らずコンピューターの世界では英語が共通言語となっていて，翻訳も含めて英語の情報が質，量ともに圧倒的である。英語プログラミングの英語／日本語による優れた解説が書籍や Web サイトに溢れている。これは日本語プログラミングと比べるとはるかに優れた学習支援環境である。
② 英語プログラミング言語では，プログラムが簡潔に効率的に書ける。プログラムの可読性も高い。
③ プログラミングのみならず文章の書き方では「英語の発想」が求められている。

日本人にとって英語プログラミングは，学び始めは難しくても，英語の語順で英語の発想を理解していけば，学習が進むにつれて，また大きなプログラムの作成になればまるほど，相対的に難しさは軽減していく。英語の発想は，前方／上方重心型で(4)，「先に概要，一般的／重要／旧情報を述べ，次に詳細，具体的／個別的／新情報を伝える」というものである。たとえば，オブジェクト指向言語では，データを結合子で処理と結んで「オブジェクト.メソッド」と表記する。ここにも英語の発想が認められる。

この「英語の発想」は，コードの構文だけでなく，プログラムの構造化など，プログラミング全体を貫いている。そこで，プログラムの全体を英語の発想に基づいて構成すれば，可読性の高いプログラムが作成できる。また Web サイトやマニュアル，論文の作成でも，英語の発想の下で構造化・階層化を行わないと，可読性の高いものとはならない。

3.7 教育用日本語プログラミング言語の問題点

教育用日本語プログラミング言語は，大学教育ではほとんど必要とされていない。大学の教育・研究で使用する言語は，間違いなく英語プログラミング言語である。専門教育で使うアプリケーションでも多くは英語プログラミングの知識を必要とする。

2024 年度からの大学入学共通テストではプログラミングの問題を含む『情報』科目を導入する方針と伺っている。試験に日本語プログラミング言語を採用するとなれば，プログラミング教育は xDNCL や「どんぐり」のような日本語プログラミング言語を使う教育となるのであろうか。そうなれば，大学教育との円滑な接続があまり期待できそうもないプログラミング教育となることはほぼ確実であろう。大学になって初めて英語プログラミングを学ぶのでは，二度手間で，貴重な限られた学習時間を無駄に使うことになることを強く危惧する。

現行の『情報基礎』と比べると，多数の多様な高校生が受験すると見込まれる共通テストでは，Python や Ruby などの英語プログラミング言語を選択できるように配慮する必要があると思われる。

4. おわりに

英語プログラミングを学ぶには，いままでは英語が高い障壁となっていた。しかし，Google 翻訳のサービスやオンライン辞書・事典などを利用すれば，この障壁を容易に乗り越えることできるようになった。高校生の英語力でも英語プログラミングができる時代となったのである。関係者はこの事実をきちんと認識する必要があると思われる。

プログラミングに英語を用いたい理由を，一つ付け加えるならば，高校生に英語の世界が英語科目の授業の外に大きく拡がっていることを実体験していただきたいという想いがある。

参考文献

(1) 岡田 健，中鉢欣秀，鈴木 弘，大岩 元：“プログラミング言語としての日本語”，KEIO SFC JOURNAL, Vol.2, No.1, pp.114-134（2003）
(2) 大岩 元，中鉢欣秀：“文法最小化を目指した日本語プログラミング言語「敷島」”，FIT2015（第 14 回情報科学技術フォーラム）第 3 分冊 469-472（2015）
(3) 酒井 邦秀：“さよなら英文法! 多読が育てる英語力”（ちくま学芸文庫），pp.94-135, 筑摩書房（2008）
(4) 外山滋比古：“英語の発想・日本語の発想”，pp.10-13, NHK 出版（1992）
(5) “初学者向けプログラミング学習環境 PEN” https://pen.moe.hm/#!manual/xdncl.md
(6) 大阪電通大 兼宗研究室:“DNCL 学習環境「どんぐり」” https://dolittle.eplang.jp/dncl