4 章 プログラム言語 30 年のあゆみ

和田英一
東京大学 工学部 計数工学科

1 はじめに

昔のことを思い出してみると, 東大で PC-1 ができたのが 1958 年, TAC が 1959 年; パリの IFIP で日立と日電の計算機を展示したのも 1959 年; 第 1 回プログラミングシンポジウムの開催と情報処理学会の設立が 1960 年であった [1]. PC-1 や TAC ではイニシャルオーダーでプログラムを書いていたが, 一方 ACM の「プログラム言語の歴史」コンファレンスの報告書 [2] を見ると Fortran は 1957 年, Lisp は 1960 年頃にはでき上っていたらしいから, 情報処理学会 30 年のあゆみはそのままプログラム言語 30 年のあゆみということになる. 筆者は第一世代の計算機の頃からずっとプログラム言語につき合ってきたし, 今でもまだ少しはつき合っているつもりなので, 機会を与えられたのを幸い, プログラム言語の 30 年を思いつくままに振り返ってみることにする.

この 30 年間におびただしい数のプログラム言語が提案, 試作, 使用されてきたが, そのうちほんの一握りのものだけが生きのび, 実用になった. あとはすべて人類にわずかな貴重な経験を残しただけで消滅した. どういうプログラム言語が生きのび, あるいは消滅したかという分析は真面目にやれば面白かろうが, ここでは筆者の無責任なコメントを付すにとどめる. 具合の悪いことが書いてあってもどうか気にしないでほしい.

ここで取りあげる言語についてもあまり確固とした基準はない. 筆者がそれとなく関心をもっている言語というのが正しいと思う. それでは出発.

2 1960 年代のプログラム言語

2.1 Fortran

彼の講義を聴講する条件は「Fortran を知らないこと」と噂のあるダイクストラはチューリング賞記念講演で「Fortran などというものが存在していたことを我々が忘れ去る日は早ければ早いほど良いのです. なぜなら思考のための道具として Fortran はもはや適当でないからです [3]」と言っている. もう何度も死んだと言われたりその亡びるのが期待されたりした Fortran だが, ジョン・バッカスの Fortran が計算機業界に果たした役割りは絶大である. 広辞苑 (第三版) に見出し語にあるプログラム言語はフォートランとコボルだけである (第一版の最後の頃の広辞苑にはアイ・ビー・エムがあり「 . . . などの作業を電気仕掛けで処理する自動計算機」となっていたが, 今はない.) Fortran の功罪は木村君が充分に論じた [4] から繰返さないが, とにかく IBM が全力をあげて普及と保守につとめ, 他のメーカーもこの処理系だけはと IBM に合わせて用意したので規格の制定と相俟って利用環境の整備は進んだ. また森口先生が解説書を出され NHK で講義をされたので学習環境も他の言語に比べ格段によかった. 特に JIS 規格とその用語に忠実な解説書のおかげで, 計算機科学者, ソフトウェア工学者の間で専門用語が安定し, 研究・教育の効率が改善されたことは特筆に値する.

Fortran がいろいろ批判されながらも “still going strong” であった裏には幾度かに亘る改訂作業と, 優れた最適化処理系の作成, さらにはスーパーコンピュータの登場などの理由がある.

大学の計算機センターで見る限り, 大部分の利用者は数値計算が目的なのでプログラム言語は Fortran で充分であり, Unix などより例のジョブコンをつけてバッチで流すという生活様式である．特に最近小規模な計算はワークステーションに移って行き, 大口だけが残り, 益々 Fortran 一色になっている．

スーパーコンピュータでは Fortran しか使えないということで, (東大の) 金田君の π の計算は数値計算だから Fortran で書いて当然だが, 論理型プログラムをスーパーコンピュータで走らせる (日立の) 金田君 [5], ニルソン君 [6] の仕事では処理系を Fortran で書いている．

だが Fortran 全盛も教育用計算機センターで見る限り様子が変わりつつある．パソコンで Pascal が動き, Pascal を教えられるスタッフが増えてきたとか, いろいろな要因で他の言語も使われだした．その栄光はながく語りつがれていくにしろ, Fortran 万才, ヤッホー Fortran の時代は果していつまで続くだろうか.

2.2 Algol

フレッド・フォートランの食料品店を出てバベル通りを 60 番地までくると, そこにはアルゴル夫人がチョコレート店を経営している．鏡の中に鏡が再帰的に見える鏡やジェンセン器械のおいてある店にはいまも夫人が端正に座っている [7].

1958 年の予備報告に続き, 1960 年に Algol 60 の報告書がでた．アメリカの Fortran に対しヨーロッパの Algol 60 は実用的なプログラム言語としては決して成功したとはいえないが, ホア [8] のみならず世界の計算機屋にプログラム言語とはどういうものであるかを教えたのであった．プログラム言語は構文と意味に分けて記述する; プログラムが扱うデータには型がある; 再帰呼出しの意味の記述はかくかくしかじか; 実数の計算については “in the sense of numerical analysis” と書いた.

プログラム言語として成功しなかった理由は理想を追い過ぎて金物表現をインプリメンタにまかせたこと, own や switch など問題と思われる機能を入れたこと, その他である．我が国でも Algolip などいくつかの処理系が作られたが, 予約語の表現が面倒であったという印象は否めない．だが goto 文なしにプログラムが書ける楽しさを味わうことができた．もちろんこれが goto 文有害論 [9] に発展するとは予想しなかったが.

あの文法書 [10] はひとつの定冠詞にまで注意を払って書いてあるという．プログラム言語としては成功しなくても, 文法書は新人に意味を深く味わいつつ読んで貰いたいものである．幸い我が国には米田さんと野下君の解説書 [11] がある.

バベル通り 68 番地にアルゴル夫人の親戚の店があるといわれるのは Algol 68 のことである．1981 年 5 月, 筆者はホア, ヴィルト, 米田さん等とアムステルダムの宿舎のバーで飲みながら「あの日」の話をしていた．筆者は行かなかったが, 1968 年 12 月 (何日か?) IFIP TC2 WG2.1 はミュンヘンで次期 Algol を決めるべく会合し, ワインハルデンの言語を Algol 68 として選んだ. 日本は大学紛争と 3 億円事件のさなか, 米田さん, 島内さん, 筧君等と用意した Algol N を, 一方ホアとヴィルトは Algol W を候補言語として持ち込んだが, どちらも採用にならなかった．「鐘を鳴らした人は酒代を負担すべし」の貼り紙を見ながらホアが鐘を鳴らし全員にビールを注文して思い出話は深更に及んだ.

Algol 68. これも所詮委員会言語であった. Algol 60 はバッカスの考えた BNF で記述してある (BNF は今でも多くの言語の構文記述に使われている.) これに対して Algol 68 はワインハルデン文法という 2 レベルの構文記述法を使った．つまり Algol 68 の BNF を作り出すための構文規則である．まずメタルールがメタノーションからプロトノーションを山のように作りだす．一方 BNF のひな形ハイパールールがあり, この中のメタノーションをそれから作ったプロトノーションで一斉に置き換えると BNF の構文規則が出来る仕掛けであった．プロトノーションが無限に作られるから構文規則を無限にでき, 袋小路になるような場合もある．いずれにしろ Algol 68 はこの構文記述が驚きであり, これで例えば「宣言していない変数は使えない」など通常なら構文規則以外で記述するようなことも構文規則で表現していた．そのかわり難解で「涙なしの Algol 68」という本を誰かが書いたように思う.

演算は減算しか定義してなく, あとはそれを使って構成するという手法であった．もう亡くなったがワインハルデンは年をとってもなかなか鋭く, 面白いことをいったり皆と飲みに行ったり, Algol グルー

プの善き指導者であった.

アルゴル夫人が自分には子孫が多いというのは Algol 68 でなく Algol 60 の方である. BNF, 型宣言, ブロック構造等 Algol 60 の考えを継承した言語は各地でずい分作られた. Pascal はその代表である.

2.3 Lisp

数年前, プログラム言語の講義で一学期間に Pascal と Lisp の両方を説明したことがあった. 期末試験で Pascal と Lisp で対照的な点を書いて貰ったところ「Lisp には熱烈なハッカーがいるが Pascal にはそういう人はいない」というのがあって, いたく感動した. 左様, Lisp はハッカーのプログラム言語であったのだ. 「これまでどれだけ多くの若者がベートーベンの音楽で勇気づけられたか」とはロマン・ロランの言葉である. 同様に Lisp も多くの計算機ハッカーの心の琴の糸をかき鳴らし, 彼等はまた処理系の自作に若い血潮をたぎらせた. 近山君の小さな Lisp[12], 猪飼君の Basic で書いた Lisp[13] からは熱い想いがフツフツと伝わってくる. 精魂こめた学位論文を「Lisp に捧げ」たのは誰であったろうか.

ダイクストラが “the most intelligent way to misuse a computer” と絶賛した Lisp と筆者との出会いは 1958 年であった. 研究室の藤村先輩から渡されたジョン・マッカーシーの論文である. 全容がわかったのは Lisp 1.5 のマニュアルを入手した時で, 冬の軽井沢でそれに読みふけった. 1964 年頃になると日本橋の IBM 計算センターで Lisp が使えるようになり東山君が Lisp を使って卒業論文の研究をした. 1965 年, 東大の大型計算機センターが開設され, Hitac 5020 が入ってきたので, 助手の飛沢さん (現姓 遠藤) と 5020 用の Lisp を作った. 当時はリンケージエディタが遅く, デバッグに随分手間がかかったが, 中西君の本 [14] によるとこれが我が国の最初の Lisp 処理系のようである.

その後筆者と Lisp の関係は一時疎縁になるけれども, 1971 年頃ファンデルプールから PDP-8 用 Lisp 処理系を貰い, これが学科の PDP-12 で対話的に動くため再び Lisp を楽しむことができた. 1973 年から 74 年にかけ MIT のプロジェクト MAC に滞在したが, ここは夢に見た Lisp 誕生の地. 当時ディビッド・ムーンはまだ学部学生だったが, 彼の作った MacLisp を堪能することができた.

第一回の Lisp コンテストが行われたのは 1974 年である. 我々の研究室 (森口研) はこれに参加するような Lisp 処理系は持っていなかった. Algol N 以来 Algol 系言語マイクロ Plan や Pascal, Solo の方に関心があったためである. だがいろいろな言語とつき合うほど, Lisp の簡潔な良さが目立ってくる [15].

筆者のまわり (和田研) に再び Lisp の環境ができるようになったのは研究室にいた近山君による. 1975 年に進学してきた近山君は 3 年夏の実習で東芝へ行き, そこで Lisp の洗礼を受け, 卒業論文, 修士論文, 博士論文のすべてで Lisp に関係した研究をした. 彼は 1977 年にインテルのマイコンに小さな Lisp[12] を作り, 翌年情報工学の Facom 230/38 用の Lisp/P を Pascal で書き [16], 1980 年から大型計算機センターの Hitac M200H のための UtiLisp を開発した. しばらくは UtiLisp の話題になるが許してほしい.

UtiLisp は一応 MacLisp 系だが InterLisp 流のエディタも持ち, 独自に工夫したところもあり非常にバランスのよい仕様だ. とにかく高速実行を身上としており, 当時理研に来ていたノーマンが感心していた. UtiLisp はその後 Facom や IBM のメインフレームへも移植した.

研究室はその後ワークステーション時代になり, UtiLisp をまず Sun1 (MC68000) へ移植した. これはアドレスが 24 ビット, レジスタ数 16 なのでメインフレームのポインタタグの手法がそのまま使えた. ただ GC (ゴミ集め) はメインフレームの式年遷宮方式からポインタ逆転とスライドコンパクション方式に変わった. この UtiLisp68 は 512K の Macintosh でも動き仲々快適である.

一方メインフレームの UtiLisp はアーキテクチャの異なる日電の ACOS シリーズへも移植された. これは日電の上田さんによる.

ワークステーションが MC68010, 68020 になると, アドレス幅が増えポインタタグが使えず, Sun1 の UtiLisp68 は Sun2, Sun3 へは移せなかった. だがなんとしても新しい Sun にも UtiLisp が欲しいという研究室の熱意で, アドレス, オフジェクトタグ併用の UtiLisp32 ができた. 湯浅, 高橋, 金子, 石井君等の努力による. これにはこれまでの UtLisp になかった多重精度整数 (bignum) がついた. 特にその 2

進 10 進変換は高橋君の苦心の作である．UtiLisp32 は VAX にも移植した．

日電では，68020 系のワークステーションにも UtiLisp をのせるべく工夫し，我々とは違ったアドレスタグ方式のものを実現した．EWS4800 の UtiLisp がそれで，橋本さん等による．

80x86 の CPU へは金子君が UtiLisp32 を多少簡単化し cUtiLisp を作った，日電やエプソンの PC，ラップトップなどで動いている．

1988 年夏，研究室に Sun4 が入ると，この SPARC チップの CPU のため，今回は田中君によって C で書いた UtiLisp/C が完成した．　コンパイラは村松君の労作である．

UtiLisp は上述の処理系以外にも我が国の Lisp マシンである富士通の Facom α，NTT の Elis でも動くことを付け加えたい．

以上は Lisp ハックのほんの一例に過ぎない．ここには書き切れないくらい国内，国外到るところに Lisp の現役，予備軍が満ちている．Lisp は誰にでも作れ，誰にでも使える．Lisp しか使えない Lisp マシンを作ろうとした滝君その他の気持ちもよくわかるではないか．

この広い世界のどこかには「我は Lisp のために生まれ，Lisp のために生き，Lisp のために死なん」と誓っている若者がきっといるに違いないと思うのである．

2.4 Pascal

本物のプログラマから「女・子供の使うもの」と決めつけられた [17]Pascal はプログラム言語 30 年の歴史の中でやはり無視できない存在である．Pascal の成立過程について筆者は 10 年程前の雑誌の連載 [18] の最初に詳しく述べたが，簡単に繰り返しておく．前の章にあるように IFIP TC2 WG2.1 が 1968 年 12 月に Algol 60 の後継言語を選定したが，その時の候補の Algol W がその後さらにリファインされ Pascal になったといってよい．今でも Pascal の規格書の前書きに書いてあるが，ヴィルトが Pascal を設計した目的は

- 言語にはっきり現れている概念に基づき，系統的なプログラムの作り方を教えるのに適した言語を作る;
- 信頼でき，速いコンパイラが実現できる言語を作る;

ことであった．

最初のコンパイラは Fortran で書き，次のコンパイラは Pascal で記述した．この Pascal の文法は発刊されたばかりの Acta Informatica に載った．その後 Pascal の文法は多少手直しされたけれども，あまり普及する前だったので，その影響はあまりない．

Pascal が出たのが 1970 年として，30 年の歴史のうち，もう初めから 1/3 位の処になるけれども，その後の Pascal の普及には二つのステップがあると見られる．その一つは 1970 年代前半で，東大でもその頃使えるようになったのだが，その拡まり方は今考えてみると:

1. Pascal の考え方に共鳴した人から要求があり，ヴィルトはそれに快く応じてテープを送った．
2. 構造的プログラミングの論文がいくつも発表されたが，それらの論文の例題は Pascal(風の言語) で書いたものが多い．
3. 比較的早い時期に手引き書を出版した．

この 3 条件が揃っていたために第一ステップの Pascal の普及があったと筆者は思うがいかがであろうか．

もちろん Pascal 自身が良い設計であったというのが前提条件であるが，これは主観的な判断かもしれない．現に Pascal にはハーバーマンの批判という文書があった．Pascal にはこういう機能も欲しいとか，こんな風にしか書けないのは困るとか，いろいろ書いてあったように思うけれども，ヴィルトがそれ

を一切黙殺したのは良かった．Pascal は Algol のような委員会言語に対してニクラウス・ヴィルトの個人設計言語なのだから，設計者個人が何でも独断で決めればよい．委員会になると民主的に何でも取り入れて，言語は主体性がなくなってしまう．

第二ステップはパソコンの登場である．1970 年代半ばに登場した 4K バイトのパソコンの Basic のインタプリタを読んだ時は，15 年程前の 0.5K 語パラメトロン計算機のコーディングテクニックを思い出し，ついハラハラと涙を流したりしたものだが，やがて Apple にケン・ボールスが Pascal を乗せるという日が来る．これを契機にパソコンとともに Pascal が普及した．Apple Macintosh はシステムプログラムが Pascal で書いてある．パソコン用 Pascal がその後どの位あるか，咄嗟には Turbo Pascal 位しか思い出せないが，ずい分あるのだろう．

少し時代はバックするが，Pascal は移植キットがよく出来ていた．それも普及に役立ったに違いない．これも書き出すと長くなるが，ひとつだけ面白い話は Pascal のコンパイラが Pascal で書いてあり，それに 10 進 2 進変換の定数が 0.1 と書いてあって問題を引き起こしたことだ．2 進法の計算機では 10 進法の小数 0.1 は正確には表現できないが，この定数を使って次の版のコンパイラの 0.1 を計算するのだから 0.1 は前より不正確になる．それを使って次の版の 0.1 を作るとさらに不正確になる．という具合で 0.1 の値を見ればヴァージョンがわかるというジョークもあったけれども，これは筆者がチューリッヒに行ったとき報告をしたので既に修正されている．

Pascal が普及するのはわかったけれども，その結果いろいろな処理系ができ，すると今後は規格を作りたいなどと変なことをいう人が出てくる．これも規格の前書きに書いてあることだが，Pascal は最初の目的を超えて広く使われるようになり，システムプログラム，応用プログラムも沢山書かれるようになった．ポータビリティを保証するために規格を作ったことになっている．

規格を作るという話を読んだ時，筆者はこれは有難くないと思った．Fortran や Cobol は知らず，Pascal ならヴィルトの文法書で充分である．そもそも Pascal のコンパイラは別として真面目に処理系を作るならせめて PL/360 のようにレジスタに手の届く言語を使うべきではないか．ヴィルトの考えを聞くと，彼も規格化には反対で規格化の作業には加わらないという．彼のいい分はプログラム言語の研究のために Pascal の処理系に自由に手を加えていろいろな実験をやってみて貰いたいからというのであった．

規格を作る作業は難航した．すでに各地で独自に作られた処理系があって，簡単には統一的な記述ができなかった．また委員会形式で規格文案を作ったので，次々と意見がでては精密化しなければならないという事態が続き，その結果，今見るような難解な規格書ができ上った．規格書ができてもことは収まらなかった．第一に規格書にまだ曖昧な点が見付かり，そのため 10 名位のインタプリテーショングループを作り，そこへ次々と質問状を送り，意見を聞くことになった．実は筆者もそのメンバーにされたので，初めのうちは返事をしていたが，そのうち馬鹿らしくなって交替して貰った．第二に，待っていましたとばかり拡張 Pascal の規格を作ろうという提案が出たのである．もうこうなると PL/I と同じで，中毒症状になり，次々と拡張しないと気が済まないということになる．拡張 Pascal の規格も米国を中心に進んでいたが，筆者はもうただただ呆れてフォローするのを止めている．

プログラム言語の規格は ISO/IEC JTC1/SC22 でやることになっているのだけれども，Pascal の失敗に懲りずに次々といろいろな言語について規格化の提案が後を断たず，まったく世も末だと思うこの頃である．

Pascal に関しては Pascal 8000 の事を書き留めておきたい．これは疋田君，石畑君によって東大大型計算機センターの Hitac 8800 上に作られた．その後，前野君等がチューニングし，かなり高性能の国産 Pascal であった．最近はどの位使われているであろうか．

3　1970 年代のプログラム言語

さしてメジャーでもないのに，独自の哲学を持っているということで書き留めておきたい言語に BCPL[19] と Pop-2[20] がある．BCPL はケンブリッジのマーチン・リチャーズの，Pop-2 はエディンバラのロビン・ポップルストーンの，ともに個人設計言語である．

3.1 BCPL

BCPL は CPL を簡単化した言語ということになっている．CPL で書いたプログラムは見たことがあるが, 自慢できる処理系が存在したかどうかは不明であるる．リチャーズはしばらく MIT のプロジェクト MAC に滞在し, その時 BCPL を作ったといわれる．筆者の昔持っていた BCPL のマニュアルの一つは MIT 版, もう一つはケンブリッジ版である．結論からいえばこれは CPL → BCPL → C となって, 今や C が全盛の時代である．だが, 筆者は C より BCPL の方にずっと愛着がある．

BCPL のデータには型がない．すべてビット列と気持ちがよい．式には右の値 (rv) と左の値 (lv) があり, 最初はまごつくけれども, Algol 68 で独りでに deref がかかるのと違い, 構文上で右か左の値をとるのが決まっている他は陽に明示しなければならないが, それはそれで良いことだと思う．

Unix の本拠のベル研は C へ移行したが, Smalltalk のゼロックスには BCPL の伝統がしばらく残っていたようである．Alto は BCPL マシンであったらしい．

3.2 Pop-2

BCPL は移植性に工夫を凝らした処理系を持つコンパイラ言語であるが, Pop-2 は対話性を前面に出した解釈実行型の言語であった． 当時アメリカには Joss という対話型の言語だかシステムがあったし, PDP で動く Lisp も対話型だったようだが, Pop-2 ははるかにいろいろなことができる言語であった．

まず, いまなら電卓でやるような卓上計算機の機能があった．変数を宣言して値を入れることもできる．関数や手続きも宣言できる．ラムダ式もある．数値計算だけでなく, 記号処理もできた．

BCPL と同様に Pop-2 も計算メカニズムの本質が言語に現れているような気がした．なにしろ計算に使うスタックを前面に押し出している．1,2 → y → x で 1 をスタックに push; 2 をスタックに push; スタックを pop して y へ; スタックを pop して x へ; となり x に 1 が, y に 2 が入る仕掛けであった．面白いのは BCPL の rv と lv に似ているが selector と updater で, リスト l の先頭の要素をリスト k の先頭の要素とするのに hd(l) → hd(k) と書く訳だが, 要素を取り出す hd は selector, 要素に代入する hd は updater といい, hd はふたつの関数からなるダブレットという．hd の updater を作るには < 定義 > → updater(hd), selector を作るには < 定義 > → setector(hd) とするが, updater, selector もダブレットで updater (hd) → . . . と書くと updater の selector が使われることになる．

オペレータも利用者がその優先順位とともに自由に定義できる (Prolog や Algol N と同様.) エディンバラでバーストールが Pop-2 のデモを見せてくれ, いきなりバックトラックの関数を作ったのにいささか驚いたが, 結構何でもできる言語である．

「データがそのままプログラムになる言語が Lisp の他にあるか?」IFIP WG2.1 の会合で誰かが発言した．ファンデルプールが即座に「あるとも！ Pop-2 ならできる」と答えた．popval という関数が文字列を Pop-2 のプログラムにしてしまうのである．

Pop-2 はエディンバラの人工知能研究所で使っていた． Popeye という視覚の実験をしたり Poplar という Prolog のようなものを作ったりしていたが, 最近は他の大学でパソコンのための Pop-11 を開発したようである．

4 1980 年代のプログラム言語

アメリカで作られた沢山の言語からここでは Smalltalk[21] と PostScript[22] を取り挙げたい．Smalltalk はゼロックスの, PostScript はアドービの, ともに企業設計言語というべきか．

4.1 Smalltalk

Smalltalk は言語というよりシステムだが, オブジェクト指向の独特の言語を内蔵している．ダイナブックの記事を読んだのはいつ頃だか忘れた．1979 年の春だと思うが当時まだ鈴木君のいたゼロックスの研究所を訪れたところ, Alto にダイナブックさながらの画面が映っていて感激した．それが Smalltalk

であったかどうかは知らない. Smalltalk について詳しく知ったのは Byte 誌の Smalltalk 特集号によってであった. だがプログラムをどう入力するのかさっぱりわからず, その後東大にきた鈴木君に聞くと, ウィンドウで編集して作るだけだというすげない返事. それがどういう意味かは鈴木君が Sun1 に移植した Smalltalk を直接見たり使ったりした時である.

Smalltalk はしかし大変なカルチャーショックであった. 環境としては今となっては小さいかも知れないが, あれ程のユーティリティ, 基本的メソッド, 環境の集合が用意してあるものは Unix をおいてないのではないかと思った. The Language and its Implementation を院生諸君と 8 割りかた読み, MVC の何たるかもわかり, Macintosh や Sun でも使うと, よくぞこんなものを作ったと恐れ入った. ただ筆者の悲観的結論は, ああいうものが自由に使えるのはやはり 35 才までらしいということである.

Smalltalk については梅村君の意見 [23] を引用したい.

> Smalltalk と Lisp を比べてみると, Smalltalk の方が純粋培養の名花で, Lisp のほうは雨風を生きぬいてきた野の花のような気がします. Smalltalk の方は, それぞれの機能がうまく融合して, 一つの統合したシステムを作っているのに対し, Lisp の方は, それぞれが微妙な不協和音を出しながらも, なんとか協力し合っているような感じです...

4.2 PostScript

Smalltalk に比べれば一見して大したことはないと思うのが PostScript である. 共通点はともに Xerox が発祥の地ということでろうか. 半年ほど前, 大学の自室に Macintosh , Sun3 と並べて Apple レーザライタを設置したのでいろいろな図を書いたり, 文字端末のハードコピィシミュレータを作ったりで PostScript で直接プログラムを書くことも多くなった. ずい分悠長な言語とも思うが Pop-2, Forth なみのスタックベースの命令体系で, 感心するほど何でもできる. 今のところ, latex の中から使うのに多少手古摺っているが, PostScript の世界でこれはできなかったという記憶はない.

Macintosh で書いた絵をレーザライタに送る時も PostScript になるし, ビットマップも PostScript のインタープリトできるものがあるとか, PostScript はこの業界の標準になりつつある. 下手に規格屋が規格を作ろうとして泥沼に陥るより, センスのよい設計者が Smalltalk や PostScript のような哲学のしっかりしたものを早目に作るほうがユーザにとってはるかに有難いことだと思う.

5 日本のプログラム言語

我が国のソフトウェア界はインプリメントは勝れているが, 独自に考え出したプログラム言語はどの位あったろうか.

Algol N の N は「日本」のことで, これを Algo 68 の候補言語として設計したことを前に述べた. これはインプリメントもしたけれど全然使われなかった.

水谷さんの小朱唇 [24]. Red Lip (reduced list processor) の小型版という意味らしい. これも PostScript のようにオペレータが最後に来る. というよりはつまり日本語の命令形の記法に従っているのだ. 文を逆ポーリッシュにするのなら同じ構文解析手法を使うことにして, 式も PostScript 風にした方がよかったのではないかと思われる. 逆ポーリッシュ記法がオブジェクト指向によく似合うとは竹内君の説だが小朱唇はそこまでは徹底していない.

ICOT に拡張 Prolog オブジェクト指向ともいうべき ESP がある. PSI マシンの操作システムなどはこれで書いた. 筆者は話を何度か聞いた程度であまり詳しくは知らないが, この言語のお陰で SIMPOMS が早く書けたそうだ [25]. とはいえ, あれは難しい言語なのではなかろうか. ICOT だからこそ使えたのではないかというのはいい過ぎだろうか.

オブジェクト指向も C++ とかいろいろ言語があるが, 並列オブジェクト指向言語 ABCL/1[26] ももうかなり年期の入ったプロジェクトである. 少し使わせて貰ったが筆者なりには「あと少し若ければもっと使ったかも」という印象で, 学生諸君の間にもう少し物好きがいないかと思っている.

ICOT の GHC[27] は並列論理型言語としては, これは計算のモデルではないかと思う程単純なセマンティクスで気持ちがよい. 最近のプログラム言語の講義では Pascal と Lisp ではなくて GHC と uc を説明している.

その uc[28] は完全遅延評価の関数型プログラム言語で, ターナーの Miranda の親戚みたいな言語である. 現在のシステムはシングルプロセスでコンパイルゴー方式だけれども, 並列プロセス対話式のシステムの早い登場が期待されるところである.

これらの新しい言語は, 時には面白いと思って使ってみ, これらの言語の設計者の力量に敬服するが, これは Pure Lisp と同じで, 本当になんでも書こうとすると心ならずも他のパラダイムも入れなければならないのではないかという気がしている.

他にも国産言語があるだろうがここもすぐ思い出せたものだけにした.

6 おわりに

30 年間に現れ, 前の章に書けなかったプログラム言語を思い出し, その名前だけ並べてみる. (参考文献は省略する)

Ada, Alphard, Apl, Awk, B, Basic, C, C++, Chill, Clu, Cobol, Comit, CommonLisp, D32, Edison, Euler, Forth, GPSS, icon, IPL-V, Joss, L6, Maple, Mary, Mesa, Miranda, ML, Modula, Mumps, Oc, PL/I, Plits, PL/360, Prolog, Sail, Scheme, Setl, Simula, Snobol, Solo, Tao, Tartan, Trac, Turing, Watfor, ...

この他, 名前だけ存在したもの
Abstracto, Utopia 84, ...

一年に平均いくつのプログラム言語が出来たか計算してほしい.

ハードウェアが 30 年間になしとげた長足の進歩に比べたら, 30 年 1 日のごとき Fortran をかかえているプログラム言語界はいったい何をやってきたのだろうかと思われるに違いない. だがプログラム言語は人間の言語である. 人間の自然言語ももちろん万古不易ではない. だから言語年代学があるわけだ. それに比べれば次々と新プログラム言語の生まれてくるのは驚くばかりである. 生れてきたものは必ず滅びるというが問題はその半減期のオーダーで, 1 カ月のものから数十年のものまである. これは複雑な社会現象のなせるわざであろうか. 考えられるファクターとしては

- 設計者のセンス
- そのセンスを見抜く利用者の眼
- 不和雷同
- 一度使った利用者の思い入れ
- メーカーの熱意
- 先生の学生に対する押しつけ
- ISO などのお墨つき

こんな問題は解くわけにいかない. 筆者としては長い間沢山の言語を見たり聞いたり使ったりしてきたが, いつでも使いたい言語がきちんと保守され快適に使える環境の中で生きてこられたのは望外の幸せと思うほかはない.

CommonLisp について

30 年のあゆみの編集委員会から CommonLisp についてもなにかコメントするように求められたが, 筆者はそこに軍事研究の臭いがする; 商業政策が Lisp を汚しているとしか思えない; という理由で CommonLisp には関心がないためあまりコメントできない. 中西君は

> 筆者の本音としては, CommonLisp は大嫌いであり, Lisp 本来の純粋性やシンプル性を壊すものとして, やがては自分自身の寿命を短くするものとして, 極言すれば AIDS のような存在であるとすら思っている [29]. ...

といっているが, 筆者も大体同意見である. 先日 CommonLisp の作業委員会に呼ばれそこで数センチの厚さの規格書を見た. これは Pascal 標準化の二の舞いである. 数センチの規格書の一貫性など誰がどうやって調べるのであろうか. そのせいかどうかは知らないが, 規格化も予想通り遅れている.

しかし, 京大数解研の湯浅, 萩谷君がいちはやく KCl(Kyoto CommonLisp) の処理系を作り, 各地に送り出したのは, いろいろな意味で評価されてよいであろう. 最近は HCl(Hokkaido CommonLisp) という名前も耳にする.

参考文献

[1] 情報処理学会歴史特別委員会: "日本のコンピュータの歴史", オーム社 (1985 年).

[2] R. W. Wexelblat ed: "Preprints ACM SIGPLAN History of Programming Languages, Conference, Los Angeles, June 1978", SIGPLAN NOTICES, Vol.13, No.8 (Aug. 1978).

[3] E. W. ダイクストラ (木村泉 訳): "謙虚なプログラマ", bit, Vol.5, No.11 (1973 年 10 月).

[4] 木村 泉: "Fortran はいつ亡びるか", bit, Vol.9, Nos.3-6 (1977 年 3 - 6 月).

[5] 金田 泰: "スーパー・コンピュータによる Prolog の高速実行", 第 26 回プログラミングシンポジウム報告集 (1985 年).

[6] Martin Nilsson: "Parallel Logic Programming for SIMD Supercomputers and Massively Parallel Computers", 東京大学情報工学専攻博士論文 (1989 年 3 月).

[7] Peter Brown: "Taking a stroll down Babel Street"" CACM, Vol.27, No.4 (April 1984).

[8] C.A.R. ホア:"皇帝のふる着",bit,Vol.13,No.8(1981 年 7 月).

[9] 木村 泉:"GO TO 論争",bit, Vol.7, No. 5 (1975 年 5 月).

[10] P. Naur(ed): "Rivised Report on the Algoirthmic Language ALGOL 60", CACM, Vol. 6, No.1, pp. 1 - 17(Jan. 1963).

[11] 米田信夫, 野下浩平: "Algol 60 講義", 共立出版 (1979).

[12] 近山隆: "小さい Lisp", 情報処理学会プログラミングシンポジウム委員会, マイクロコンピュータソフトウェアシンポジウム報告集 (1977 年 7 月).

[13] 猪飼秀隆: "Basic で書いた Lisp", bit, Vol.18, Nos.9-10(1986 年 9 - 10 月).

[14] 中西正和: "Lisp 入門 - システムとプログラミング", 近代科学社.

[15] System5: "Lisp 礼讃", 科学セミナー, (1985 年 4 月).

[16] 近山隆: “Pascal による Lisp 処理系の作成”, 情報処理学会プログラミングシンポジウム委員会, 第20回プログラミングシンポジウム報告集 (1979年1月).

[17] “本物のプログラマは Pascal を使わない”, bit, Vol.17, No.4 (1985年4月).

[18] 和田英一: “プログラム言語 Pascal”, bit, Vol.10, Nos.1-12 (1978年1 - 12月).

[19] M. Richards, C. Whitby-Strevens, 和田訳:“BCPL – 言語とそのコンパイラ”, 共立出版 (1985).

[20] R. M. Burstall, J. S. Collins, R. J. Poppelstone: “Programming in Pop-2”, Edinburgh University Press(1971).

[21] A. Goldberg and D. Robson: “Smalltalk-80 –The Language and its Implementation”, Addison-Wesley(1983).

[22] Adobe Systems Inc.: “PostScript Language ”, Addison-Wesley (1985).

[23] 梅村恭司: “Smalltalk-80 入門”, サイエンス社 (1986).

[24] 水谷静夫: “小朱唇言語仕様”, 東京女子大学日本文学科 (1986年).

[25] 近山隆: “オブジェクト試行言語による OS の開発例”, 情報処理, Vol.29, No.4, pp.396-401 (1988年4月).

[26] 米澤明徳, 柴山悦哉 他: “オブジェクト指向に基づく並列情報処理モデル ABM/1 とその記述言語 ABCL/1”, コンピュータソフトウェア, Vol.3, No.3, pp.9-23 (1986).

[27] 上田和紀: “Guarded Horn Clauses”, 東京大学院情報工学博士論文 (1986).

[28] M. Takeichi: “Fully Lazy Evaluation of Functional Programs”, (1987).

[29] 中西正和: “パソコンと Lisp”, Computer Today, Vol.5, No.2 (1988年3月).