C 4. 数式解析のプログラムをFORTRANで書いた話

戸川隼人（航技研）
照井勝利（HSE）
西村剛（成蹊大）
戸川保子（航技研）

1. フィロソフィー

近頃では数式解析のプログラムと言えばリスト処理用言語（または少なくともリスト処理向きの言語）で書くのが当りまえのことになつてしまつた．そういう言語を用いれば確かに便利には違いない．しかしFORTRAN で書いても結構うまいこと書けるものだという例を一つお話ししようと思う．

近頃はいろんな言語がはやる．リスト処理用言語，シミュレーション用言語，コンパイラー作成用言語，ナントカ解析用言語，等々．それにはそれなりの存在理由があり利益もある訳だけれども，結構づくめの話ではない．第一，これらの言語は，その最も得意とする分野に限つてみても「特に覚え易い」とは言えない．時には「特に使い易く」もないことがある．計算機の使用効率について言えば，概してあまり良くない．それに最も困ることは言語の限界が応用分野を限定してしまうことである．全部が全部そうだとは言わないが，新興言語の多くのものは，ショーウィンドーで見るといかにも便利そうだが買つてくると飾りもの程度の役にしか立たない大根の自動皮むき機とか，ジグザグミシンとか家庭用電気大工とか，どうもそういうもののような気がしてならない．専用言語の唯一のメリットは，その専門分野の概念と用語だけを使つて記述できるということ，いうなれば「計算機くさくない」プログラムを書ける，ということであろう．

一方，これをFORTRANとかALGOLなどの汎用言語で書くことにすると，たいてい誰でも知つているからプログラマー・コストが安い，ディバックの時に普通のモニターランに簡単に挿入できるので回転が速い，既製のサブルーチンを必要に応じて利用できる，他の機構や他のメーカーの計算機にもそのまま（または少しの修正で）使用できる，等の利益がある．

ただし最後の点については「常に」という訳には行かない．実は今回のプログラムも，設計時にいろいろ検討した結果，他機種とのコンパティビリティーは考えない，という方針にふみきつた．それは磁気ドラムを使用すること（そのために，5020 FORTRAN 個有のステートメントを使用することになる）およびun-normalized floating-point numberを効果的に利用するということの二点である．磁気ドラムのかわりに磁気テープを使用すればコンパティビリティーが保証されるのであるが，その代償として計算時間が

莫大になる．un-normalize の方もコンパティビリティーを持たせる方法はいくらもあるが，ワーキングスペースの効果的利用と処理の簡素化のため，後に述べるような方式をあえて採用することにしたのである．

リスト処理用言語を用いない場合，普通は数式解析のプログラムをアセンブラー言語で書く．特に文字処理機能の完備した最近の大型計算機では，それらの機能を活用してアセンブラーで書けば，非常に性能のよい数式プロセッサーを作ることができる．しかし不幸なことに近頃はアセンブラーを使える人が少なくなつてしまつた．ことに大型計算機の諸機能をフルに使いこなせる人は本当に少ない．したがつて協同作業で短期間に仕上げようとしても人手が集まらないので困る．またコーディングやディバッグに要する時間がFORTRANの場合に比較して格段に長い．例えば原因不明のシラミさがしに一週間かかるという事は，アセンブラーの方では珍らしいことではないがFORTRAN ならばそういう事態はまず起らない．また他人が作つたプログラムにちよつと手を入れて自分用に改造して使うこともFORTRAN ならば容易であるがアセンブラーではむつかしい．

FORTRANは語単位の処理を得意とし，字単位というのは全く苦手である．ところで普通，数式解析というと字単位の仕事のように考えられているが，これは一概にそうとも言えないように思う．数式解析の中にも数値計算が少しではあるが含まれている．その部分に関しては語単位の方が便利である．変数名や関数名も，何文字かで構成される名前が一つの変数とか関数を表わすのであるから，1語の中に名前をつめこんで語単位に処理すれば，変数名単位に処理できることになり，これも便利である．その反対に，演算子やカッコなどに関しては，語単位で処理すると，せいぜい6ビットぐらいで表現できるものを何十ビットも使つて表現するのであるから，場所を食つて損だということになる．小容量の計算機の場合にはこの損失が重大であるが，65K以上のコアメモリーを持ち，更に大容量のドラムを使用できる場合には，この点はさほど大きな問題点とはならない．

つぎに式の表現については，ポーランド記法または逆ポーランド記法を用いることが定石のようになつている．しかしこれも一応反省してみる必要があるように思う．数値計算用コンパイラーの算術式の処理においては逆ポーランド記法が明らかに非常に有利であるが，これは計算の中間結果が常に1個の数値の形でコンパクトにまとまつてしまうから良いのであつて，数式解析のように長つたらしい式がいつまでもくつついて廻るような時には，あまり効果を期待できないものである．展開と微分だけが目的ならばともかく，もつと複雑な問題にまで拡張した場合，どこまでポーランド記法が有利になるか甚だ疑問である．それに，新しいプログラマーにポーランド記法を教え，その処理のフローチャートを書けるようにするには相当の期間がかかる．数式解析のプログラムは（それに限らずコンパイラーでもそうであるが）特定の例題だけをスカッとさわやかに処理することが目的ではなく，どんな問題が来ても誤りなく処理できることが最も重要であり，そのためにはコ

ーディングの際にスペシャルケースの見落しがあつてはならない．ところが慣れない記法だと，えてして見落しやすい．その点，日常使い慣れている普通の記法をそのまま使うと，誰でも比較的誤りなくロジックを構成することができる．著者らも後者の方式を採用した．

2. スペシフィケーション

使用計算機　HITAC 5020

使用言語　5020 FORTRAN

入力記法　FORTRAN コンパティブル（ただし変数名，関数名等は頭から3文字が有効）

出力記法　同　上

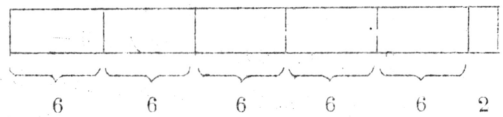
内部記法

要　素　常数，変数，関数，演算子，カッコ

演算子　＋，↑のみ．－は－1の乗算で，／は－1乗で代用する．乗算記号はすべて省略する（数学と同じ）．

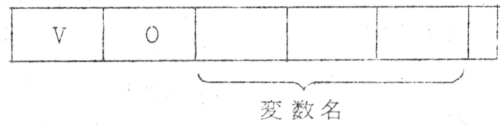
カッコ　普通の意味のカッコ（ ）のほか，関数には<>を使用する．将来は添字に〔 〕を使用する予定．

語構成

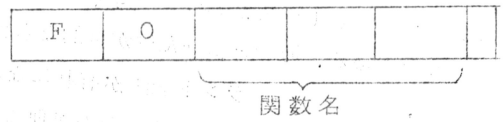


計32ビットに5文字

変　数

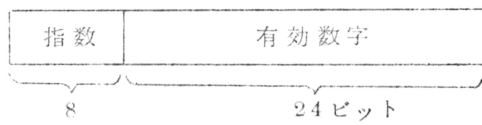


関　数



演算子とカッコは第1文字に置き，第2字は0；第3～5字はブランクとする．

常　数　正規化された浮動小数点語で表わす．変数，関数，演算子，カッコは，いずれも第2字が0なので浮動小数点語として見れば非正規形である．浮動小数点語の構成は



第1ビットが指数部の符号，第9ビットが有効数字の符号，負数の表現は2

の補数，正規化されたゼロは，0×2^{-128}である．

このようにして要素はすべて各1語で表現されるので，判断は，例えば

IF(JI.EQ.2H+0)

というような形で書く．変数，関数の判別は，左2文字だけをとり出すファンクションLEFTを定義しておいて

IF(LEFT(JI).EQ.2HV0)

等で行なう．数字であるか否かの判定は，非正規の数に0を浮動小数点加算すると正規化されて，ビットパターンが変化することを利用して行ない，これをロジカル・ファンクションの形に組んで利用している．

3. 出来ぐあい

現方式にする以前に，やはりFORTRANを用いてパイロットモデルを作り，入出力，展開，単純化をほぼ完成した．この時は1要素の表現に2語を使用し，1語で常数，関数，変数，演算子等の区別，もう1語で変数名，関数名，数値等を表現し，判別等の処理には単長語で，また転送時には倍長語で取扱つた．しかしこの方法はあまり成功しなかつた．また積の記号＊を使用していたが，かえつてじやまになることが多いことを知つた．それに，プログラムの品質管理をしなかつたので，よく脱線し，原因解明と復旧に苦労した．そこで新たに更に効率のよい新方式に切り換え，現在作業中である．今回は特にフェイル・セイフ・デザインを心掛け，中間のチェックを強化し，コーディングの規格化を行ない，最も下位の小道具から作りはじめて，それを積みあげて中位のサブルーチンを作り，その上に上位のサブルーチンを組み立てて行くというヒマラヤ・アタック・システムにより作業を進めている．1月までに，計算コストのデータなどを集めてシンポジウムに持参しようと思つている．

本 PDF ファイルは 1968 年発行の「第 9 回プログラミング―シンポジウム報告集」をスキャンし、項目ごとに整理して、情報処理学会電子図書館「情報学広場」に掲載するものです。

この出版物は情報処理学会への著作権譲渡がなされていませんが、情報処理学会公式 Web サイトの https://www.ipsj.or.jp/topics/Past_reports.html に下記「過去のプログラミング・シンポジウム報告集の利用許諾について」を掲載して、権利者の捜索をおこないました。そのうえで同意をいただいたもの、お申し出のなかったものを掲載しています。

過去のプログラミング・シンポジウム報告集の利用許諾について

情報処理学会発行の出版物著作権は平成 12 年から情報処理学会著作権規程に従い、学会に帰属することになっています。

プログラミング・シンポジウムの報告集は、情報処理学会と設立の事情が異なるため、この改訂がシンポジウム内部で徹底しておらず、情報処理学会の他の出版物が情報学広場 (=情報処理学会電子図書館) で公開されているにも拘らず、古い報告集には公開されていないものが少からずありました。

プログラミング・シンポジウムは昭和 59 年に情報処理学会の一部門になりましたが、それ以前の報告集も含め、この度学会の他の出版物と同様の扱いにしたいと考えます。過去のすべての報告集の論文について、著作権者（論文を執筆された故人の相続人）を探し出して利用許諾に関する同意を頂くことは困難ですので、一定期間の権利者捜索の努力をしたうえで、著作権者が見つからない場合も論文を情報学広場に掲載させていただきたいと思います。その後、著作権者が発見され、情報学広場への掲載の継続に同意が得られなかった場合には、当該論文については、掲載を停止致します。

この措置にご意見のある方は、プログラミング・シンポジウムの辻尚史運営委員長 (tsuji@math.s.chiba-u.ac.jp) までお申し出ください。

加えて、著作権者について情報をお持ちの方は事務局まで情報をお寄せくださいますようお願い申し上げます。

期間：2020 年 12 月 18 日 ～ 2021 年 3 月 19 日
掲載日：2020 年 12 月 18 日

プログラミング・シンポジウム委員会

情報処理学会著作権規程
https://www.ipsj.or.jp/copyright/ronbun/copyright.html