プログラミング言語論

プログラミング言語 発展の歴史

水野嘉明

内容

1. はじめに
2. プログラム以前のプログラム
3. プログラムの黎明期
4. 古典的高級言語（'70以前）
5. その後の高級言語（'70以後）

2

1. はじめに

- 歴史を学ぶ意味
  - ① どんな言語やパラダイムがあるかを、俯瞰する
  - ② 各パラダイムや言語が出現した背景を理解する

3

1. はじめに

- 年号は、覚えなくてよい
  - 各言語の特徴や背後の考え方
  - 何が何に影響を与えたか
- 系統、影響については、異存もありえる

4

2. プログラム以前のプログラム

- ジャカード織機（1801）
  - パンチカードを利用した、自動織機
  - カードを入れ替えることで操作パターンを簡単に変えられることから、その後計算機にも応用された

5

2. プログラム以前のプログラム

- エイダ・ラブレス（1815～1852）
  『史上初のプログラマ』
  - 詩人ジョージ・バイロンの娘
  - 解析機関についての著作中に、解析機関用のプログラムコードが記述されている

6

2. プログラム以前のプログラム

- 解析機関

チャールズ・バベッジが考案した機械式の汎用コンピュータ
蒸気機関を動力とし、入力（プログラムとデータ）は、パンチカードで供給される

(写真は、1992年に復元したもの)

7

2. プログラム以前のプログラム

- 初期の電子計算機のプログラム
  - パッチボード上の配線やスイッチ類によって プログラムを作成
  - 問題に応じて、配線を変更

8

3. プログラムの黎明期

- フォン・ノイマン（1903～1957）
  - プログラム記憶式のコンピュータ（いわゆるノイマン方式計算機）を提案（1946）
- 磁気ドラムなどの主記憶装置の実用化（1950頃～）

⇩

プログラムの出現

9

3. プログラムの黎明期

- 最初は、プログラミングは*機械語*で行われた
  - コンピュータが解釈・実行できる命令語。バイナリで記述される

10

3. プログラムの黎明期

- 機械語では、プログラムを作成するのが困難
- 機械語に1対1に対応するように、記号(mnemonic)を使って表記したのが*アセンブリ言語*である

例） 1000 1001 1110 0101
↓
mov %esp, %ebp

11

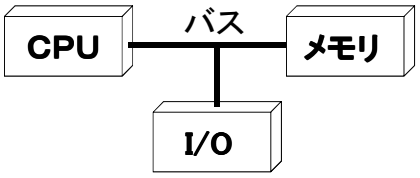
3. プログラムの黎明期

- 機械語／アセンブリ言語は、CPUの種類ごとに異なる
- 今日では、一般的なプログラム開発では*高級言語*を使用するのが普通
  - 移植性
  - 人間のわかりやすさ

12

【補足】ノイマン方式計算機

- プロセッサ（CPU)、主記憶（メモリ）、入出力装置（I/O)と、それらを結ぶバス により構成される



13

【補足】ノイマン方式計算機

- プログラム内蔵
  - 実行されるプログラムは、主記憶内にデータとして格納される。
  - プログラムとデータの区別はない。データの意味は、それをどう取り扱うかによって決まる

14

【補足】ノイマン方式計算機

- 逐次実行
  - 明示的に又は暗黙的に指定された順序により、命令は逐次実行される
- 線形メモリ
  - 主記憶には、順番に整数のアドレスがつけられている。このアドレスにより、アクセスする場所を指定する

15

【補足】ノイマン方式計算機

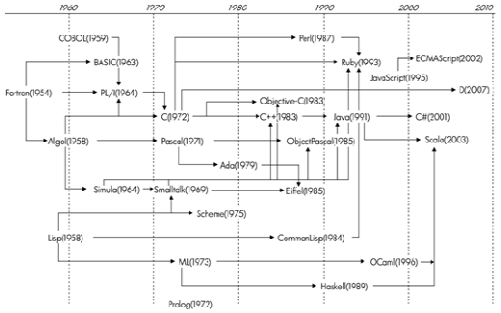
- 2進数演算
  - 計算機内部では、データも命令も すべて2進数(binary)で表現される

今日の計算機は、ほとんどが ノイマン方式である

16

4. 古典的高級言語 （'70以前）

- 主な言語の系統



17

4. 古典的高級言語 （'70以前）

4.1 Fortran
4.2 LISP
4.3 Algol
4.4 COBOL
4.5 BASIC
4.6 PL/I
4.7 Simula

18

4.1 Fortran （1954頃）

- FORmula TRANslator
- IBMのジョン・バッカス（John Backus）が中心になって開発
- 科学技術用
  - 数式の記述や複素数のデータ等が容易に取り扱えるような工夫
- 最初の高級言語

19

4.1 Fortran

- プログラム例（2次方程式の解 部分）

```
C *** 2-ZI-HOUTEISIKI ***
      READ(5,50) A,B,C
   50 FORMAT(3F12.5)
      DISCR=B**2-4.0*A*C
      IF (DISCR.LT.0.0) GO TO 100
      Y=-B/(2.0*A)
      Z=SQRT(DISCR)/(2.0*A)
      X1=Y-Z
      X2=Y+Z
      WRITE(6,60) X1,X2
   60 FORMAT(1H1///5H KON=,F15.5,1H,,F
```

20

4.1 Fortran

- Fortranの特徴
  - 一つの主(main)プログラムと複数の副(sub)プログラム
  - 1～6桁目には、行番号と継続行マーク。実行文は7桁目から
  - 空白文字が意味を持たない
  - 予約語が存在しない
  - 暗黙の型宣言

21

4.1 Fortran

- Fortranの規格
  - Fortran66 -- ASAにより制定
  - Fortran77 -- 長く使われた
  - Fortran90 -- ISO規格
  - Fortran95 -- 90の修正版
  - Fortran2003
  - Fortran2008

  言語仕様は、大幅に変化している

22

4.2 LISP （1958頃）

- LISt Processor
- MITのジョン・マッカーシ（J. McCarthy）が発明
- *ラムダ算法* の計算モデルを紙の上で表現するための記法として考案された関数型言語
- AIの研究に、広く使用されている
- 多くの方言がある

23

4.2 LISP

- プログラム例（命題の証明 部分）

```
DEFINE((
(PROVE (LAMBDA (f)
       (TEST (LIST (NORMALIZE F)) NIL NIL)))
(NORMALIZE (LAMBDA (F)
   (COND ((ATOM F)    F)
         ((EQ (CAR F)(QUOTE AND))
            (LIST (QUOTE AND)
            (NORMALIZE (CADR F))
            (NORMALIZE (CADDR F)) ))
         ((EQ (CAR F)(QUOTE OR))
            (LIST (QUOTE OR)
            (NORMALIZE (CADR F))  …
```

24

4.2 LISP

リスト＝要素の並びを括弧でくくったもの

- LISPの特徴
  - ➢リスト処理を主眼とする
    - ●括弧を多用する
  - ➢式と文の区別をしない
    - ●すべてのコードとデータは式として書き下される
    - ●LISPの関数は、それ自身がリストである

25

4.3 Algol （1958頃）

- ALGOrithmical Language
- ヨーロッパの研究者により、アルゴリズムの研究開発用に開発された
- 構造化プログラミングの考え方を最初に導入
- 多くの言語に影響を与えた（今は、あまり使われていない）

26

4.3 Algol

- プログラム例 （配列中の絶対値最大要素）

```
procedure Absmax(a) Size:(n, m) Result:(y) Subscripts:(i, k);
   value n, m; array a; integer n, m, i, k; real y;
begin
   integer p, q;
   y := 0; i := k := 1;
   for p:=1 step 1 until n do
      for q:=1 step 1 until m do
         if abs(a[p, q]) > y then
            begin
               y := abs(a[p, q]);
               i := p; k := q
            end
end Absmax
```

27

4.4 COBOL （1959頃）

- 米国国防総省のグレース・ホッパ（Grace Hopper）の指導により開発
- 事務処理用
  - ファイル処理、大きなデータの扱い、報告書の作成等に優れている
  - 計算誤差の発生しない2進化10進数による金額計算が可能

28

4.4 COBOL

- 事務処理用言語として、大いに使われた
  - 過去に作成されたプログラムやデータが膨大
  - 現在も使われ続けている

29

4.4 COBOL

- COBOLの特徴
  - 4つの部に分かれる
    - IDENTIFICATION DIVISION （見出し部）
    - ENVIRONMENT DIVISION （環境部）
    - DATA DIVISION （データ部）
    - PROCEDURE DIVISION （手続き部）
  - レコード型（構造体）が定義可能
  - 2進化10進数による固定小数点数

30

4.5 BASIC （1963頃）

- Beginner's All-purpose Symbolic Instruction Code
- 初心者向け汎用言語
- はじめは、大型機のタイムシェアリング用
  → 80年代、パソコン用として発展

31

4.6 PL/I （1964頃）

- Programming Language One
- IBMにより開発された、汎用言語
  - 商用計算と科学技術計算の両方に対応
- 言語仕様が膨大
  - ユーザの要求をまとめて仕様を作成した

32

4.6 PL/I

- PL/Iの特徴
  - FORTRANの記述形式
  - COBOLのレコード構造、入出力機能
  - ALGOLのアルゴリズム記述能力を備えている

33

4.7 Simula （1964頃）

- Simulation language
  → Simple universal language
- ノルウェー計算センターのクリステン・ニガード（K. Nygaard）とオルヨハン・ダール（O. J. Dahl）が作成
- 初め、シミュレーション用言語
  後に、汎用言語

34

4.7 Simula

- クラス、継承、動的束縛の機能を持つ
  ＝ *オブジェクト指向* の基本概念

  広く使われることは無かったが、後続の言語への影響は大きかった
  - Smalltalk、Ada、C++、Java ・・・

35

5. その後の高級言語 ('70以降)

- Fortran、COBOLは、直系の子孫以外の言語には、あまり大きな影響は与えていない
- Algolは、様々な言語に大きな影響を与えている
- Simulaは、初めてオブジェクト指向の基本的な概念を実現
- LISPは独特の文法を持つが、やはり大きな影響を様々な言語に与えた

36

5. その後の高級言語 ('70以降)

5.1 C言語
5.2 C++
5.3 Java
5.4 Pascal
5.5 Smalltalk
5.6 Ada
5.7 LISPの後継言語
5.8 ML
5.9 Haskell
5.10 Prolog
5.11 スクリプト言語

37

5.1 C言語 (1972頃)

- AT&Tベル研究所のデニス・リッチー (Dennis M. Ritchie) が主体となり開発
- Unixを書き換えるために開発された
- 表記法は、Algolに近い
  - 豊富な演算子やデータ型、制御構造を持ち、構造化プログラミングに適している

38

5.1 C言語

- OSを記述するために開発された
  - 余計なチェックをしないため高速
  - ハードウェアに密着した細かい操作が可能

プログラマは間違えない、という前提
「プログラマは神様」

39

5.1 C言語

- Cの歴史
  - 1972 デニス・リッチーが開発
  - 1973 UNIXをCで書き換え
  - 1978 カーニハン＆リッチー “The C Programming Language” （K&R本）
  - 1989 ANSI X4.159-1989 (C89)
  - 1999 ISO/IEC （C99）
  - 2011 ISO/IEC （C11）

40

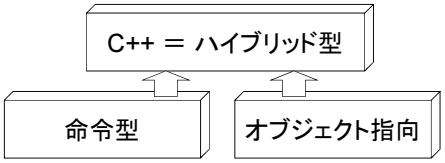
5.2 C++ （1983頃）

- ベル研のビャーネ・ストロヴストルップ（B.Stroustrup）が、C言語の拡張として開発
  - C言語と上位互換性がある
- 現在、Javaと並んで、最も広く使われている言語の一つ

41

5.2 C++

- C言語の「命令型」というパラダイムに、Simulaの「オブジェクト指向」というパラダイムを付け加えた、*ハイブリッド型言語*



42

5.3 Java （1991頃）

- Sun Microsystems Inc. が開発
- オブジェクト指向言語
- コンパイラが生成するバイトコードをJavaVMという実行環境（仮想マシン）が実行
- C++をベースに、その仕様の曖昧さ（式の評価順序など）を取り除き、型チェック等を厳密にしている

43

5.4 Pascal （1971頃）

- スイス チューリッヒ工科大学のニクラウス・ヴィルト（Niklaus Wirth）が開発
- ALGOLの影響を受けた簡素で厳密な構造化言語仕様を持つ
- 教育用に開発されたが、実用性もあり現在も使用されている
  - ➢ObjectPascal、Modula-2等に発展

44

5.5 Smalltalk （1969頃）

- XEROXのパロアルト研究所（PARC）にて、1970年代に約10年かけ3世代（Smalltalk-72、76、80）を経て整備された
- 当初は、AltoのOSという位置づけ → 統合プログラミング環境
- オブジェクト指向パラダイムの基礎を築いたと言われる

45

【付録】パロアルト研究所 (PARC)

- Alto （1973）
  - アラン・ケイのダイナブック構想
  - SmalltalkをOSとして使用
  - GUI ⇒ 現在のPCの原型
- イーサネット（1973）
- レーザプリンタ　　等々

46

【付録】パロアルト研究所 (PARC)

Xerox Alto

- ビットマップディスプレイによるウィンドウシステム
- マウスによるメニュー操作

47

5.6 Ada （1979頃）

- 米国国防総省が主導
- 当時としては先進的な概念を網羅的に取り入れており、仕様は複雑
- ALGOLやPascalに類似した文法と、高度な型の体系をもつ

48

5.7 LISPの後継言語

- Scheme （1975頃）
  - シンプルで一貫した思想に基づく
  - 言語機能を必要十分の最低限まで単純化
- Common Lisp （1984頃）
  - LISPには、多くの方言がある
  - 方言を統合しようとして設計された

49

5.8 ML （1973頃）

- Meta-Language
- LISPと同様、関数型言語
- 英国のエディンバラ大学にて、定理証明支援系の推論規則を記述するために開発された

※ Meta:「高次の」「超」
メタ言語とは、「言語を記述するための言語」の意

50

5.8 ML

- 特徴
  - シンプルな関数定義を積み重ねて強力なプログラムを記述できる
  - 型推論機能
- Standard-ML、OCamlなどの方言がある

51

5.9 Haskell （1989頃）

- 純粋関数型プログラミング言語
  - 名前は、ラムダ算法のHaskell B. Curryという論理学者に由来する
- 特徴
  - 遅延評価
    - 引数等を必要になるまで評価しない
  - 高階関数
    - 関数を引数や戻り値にできる
  - 静的多相型付け、パターン照合 等

52

5.10 Prolog （1972頃）

- Programming in Logic
- フランスのカルメラウアーとコワルスキーが考案
- 論理型言語
  - プログラムは一階述語論理に基づく
  - 事象の論理的な関係や記号処理を簡単に記述できる
- AI研究やエキスパートシステム

53

5.11 スクリプト言語

- スクリプト言語とは
  - 台本（script）のように動作を記述
  - 比較的単純なプログラムを記述するための、簡易的なプログラミング言語全般
  - 動的型付け言語をスクリプト言語と呼ぶ定義もある（厳密な定義は無い）

54

5.11 スクリプト言語

- インタプリタで動作するのが一般的
- 主なスクリプト言語
  - UNIXのシェルスクリプト
  - Perl
  - Ruby
  - JavaScript
  - ECMAScript　　など

55

演習2

- 次の説明に対し、最も適切な言語名をあげよ

(1) 事務処理用言語。ファイル処理、報告書の作成等に優れる。

(2) 最初の高級言語。科学技術計算用。

(3) 一階述語論理を基盤とする論理型言語。

56

演習2

(4) Algolの影響を受け、教育用に開発された命令型言語。

(5) ラムダ算法を基盤とする関数型言語。マッカーシ（J. McCarthy）が考案。

(6) 最初のオブジェクト指向言語。クラス、継承、動的束縛の概念を持つ。

57

演習2

(7) Unixを書き換えるために、デニス・リッチーが開発した命令型言語。Algol に近い表記法を持つ。

(8) Sun Microsystems が開発したオブジェクト指向言語。コンパイラが生成するバイトコードを、仮想マシンが実行する。

58

お疲れ様でした