プログラム言語論

亀山幸義

筑波大学 情報科学類

No. 11

この授業

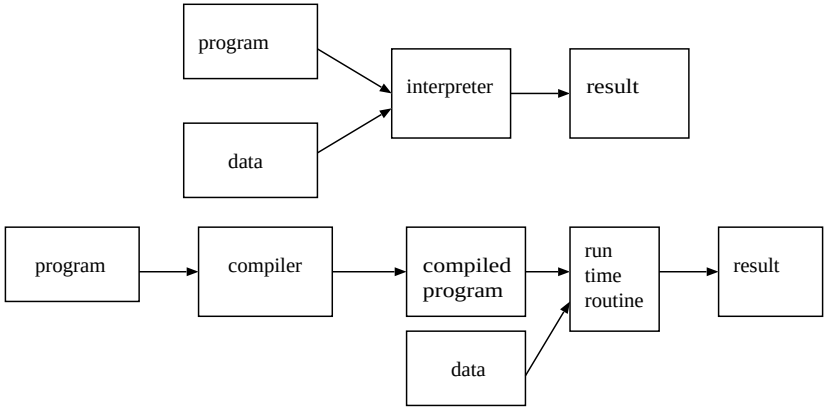
これまでの話題:

- インタープリタとコンパイラ
- プログラム言語の基礎
  - 構文; BNF など。
  - 意味論と処理の基本: ブロック構造、変数のスコープと束縛、スタックを用いた処理
  - 評価順序と制御構造; 値呼び、名前呼び、必要呼び
  - データ構造: ヒープ、ゴミ集め、型システム
  - 手続き型言語 (命令型言語) と関数型言語: 単一代入かどうか、副作用
- 発展
  - プログラムの抽象化: 抽象データ型、モジュール
  - オブジェクト指向の基本概念
  - 静的言語と動的言語、スクリプト言語

インタープリタとコンパイラ

1 つの見方:

- インタープリタ (interpreter) は解釈系 (実行系)
- コンパイラ (compiler) は翻訳系 (変換系)



コンパイル時と実行時

- Compile Time (コンパイル時) vs Run Time (実行時)
- Compile Time の情報
  - プログラムを実行する前にわかる事，静的な情報
- Run Time
  - プログラムを実行している時にわかる事，動的な情報

静的束縛 vs 動的束縛、静的な型システム (型検査、型推論) vs 動的な型システム (型検査)、静的解析 vs 実行時解析 etc.

John Mitchell 先生の (過去の) 期末試験から引用

`http://theory.stanford.edu/~jcm/books/cpl-teaching.html`
以下の性質は，コンパイル時に決定できる情報 (C)，実行時にならないと決定できない情報 (R)，どちらでも決定できない情報のどれか (N)？

- プログラム中の全ての変数が，宣言された時に初期値を与えられているか？C.
- プログラムの実行が終了するか？N.
- (C 言語) 配列の要素への参照は，宣言された配列の範囲 (下限以上，上限未満) におさまっているか？R.
- (C++言語) プログラムは型が整合しているか？C.
- すべての宣言された変数は，式の中に現れるか？C. なお,「実際に使われるか？」という質問なら，答は「R」
- システムコールの返り値は，そのシステムコールの呼び出し元でチェックされているか？C.
- 2 つの変数名がメモリ上の同じ番地を指しているか？R.

いろいろなプログラム言語

John Mitchell, Concepts in Programming Languages, 2003 から抜粋。

言語	式	関数	ヒープ	例外機構	module	object	thread
Lisp	x	x	x				
C	x	x	x				
Algol60	x	x					
Modula-3	x	x	x	x	x	x	
ML	x	x	x	x	x		
Simula	x	x	x			x	x
Smalltalk	x	x	x	x		x	x
C++	x	x	x	x	x	x	
Java	x	x	x	x	x	x	x

いろいろなプログラム言語

言語	変数束縛	呼び出し	closure	object	型付け	副作用
Lisp	静的	CBV	有り		動的	有り
Scheme	静的	CBV	有り		動的	有り
C	静的	CBV			静的	有り
C++	静的	CBV		有り	静的	有り
Java	静的	CBV		有り	静的	有り
OCaml	静的	CBV	有り	有り	静的	有り
Haskell	静的	必要呼び	有り		静的	無し
Ruby	静的	CBV	有り	有り	動的	有り
Prolog	-	-			動的	有り

プログラム言語はなぜたくさんあるか？

表現力が非常に高く、高速処理が可能なプログラム言語がたった 1 つあれば、良いか？

- 言語の表現力: A 言語が B 言語より表現力が高いとは、B のプログラムと同等なプログラム全てを A 言語で書けるとき。

答: おそらく NO。

- (答その 1) 巨大過ぎたり、複雑すぎるプログラム言語は使えない。
  - 言語の処理系を書く人やプログラムを保守 (検証、再利用 etc.) する人にとっては、言語が大き過ぎると大変。
  - Ada 言語の失敗。
- (答その 2) 解くべき問題に応じた、適切な抽象度 (right level of abstraction) の言語を使うべき。
  - 数式処理のアルゴリズムを書く人は、ゴミ集めアルゴリズムの詳細は知らなくてよい。
  - 非常に高速のネットワーク・スイッチの内部コードを書く人は、(どう効率的に実装されるかわからない) オブジェクトを使っていられない。

プログラム言語: 汎用 vs 特化

- この授業で取り上げてきた言語はほとんどすべて汎用 (general purpose) プログラム言語
- 領域特化言語 (Domain Specific Language)：
  特定の領域 (数値計算、数式処理、推論、グラフィクス etc.) に特化した言語。

汎用言語の役割と DSL の役割は相補的 (どちらか一方だけでは、すべてのニーズに対応できない)。

お勧めの読みもの: "Purpose-Built Languages", ACM Queue, Mike Shapiro (インターネット上で無料閲覧可能)。
`http://queue.acm.org/detail.cfm?id=1508217`

プログラム言語の理解

プログラム言語を理解する 3 つの方法:

- その言語の「概念」を説明している文献を読む。
- その言語で書かれた、質の良いコードを理解しようとする。
- その言語のインタープリタを、その言語自身で書いてみる。

期末試験

- 6 月 27 日 (月) 2 限プラスアルファ(10:30-12:00)
- 遅刻厳禁!
- 資料等の持ち込みは不可です。(留学生が辞書を持ち込む場合は OK)
- 暗記物ではなく、理解度を問います。たとえば、ML 言語の詳細な構文を覚えていないと解けない、というものは出しません。いろいろなプログラム言語の幅広い知識ではなく、基本的な言葉を、その意味内容とともに理解してください。
- 答案返却期間を、期末試験終了後 1-2 週間に設定します。(授業 web page に日程等を掲載する予定。)