2023年度 情報科学発展演習

言語指向プログラミング言語RACKETでつくるプログラミング言語

> たとえそれが史上最もくだらない言語のコンパイラであっても，ゼロからコンパイラを書くことはとても満足のいく取り組みだ…宇宙の支配者になったような気分になれるんだ． — Jay McCarthy

担当教員

馬谷（umatani@kanagawa-u.ac.jp）

定員

10〜12名程度

曜日・時限

履修者が確定後，全員の都合の良い時間帯を調整

場所

20-415（馬谷研学生研究室）

概要

コンパイル方式の簡単なプログラミング言語処理系の作成を通じ，プログラミング言語の基本概念や処理系実装手法について学ぶ．Racket言語が備えている**強力な言語指向機能**を用いることで，伝統的なコンパイラ実装技術（オートマトン，構文解析等）を知らずとも，少ない労力で**新しいプログラミング言語をつくり出す手段**を身につける．プログラミング言語を自らつくり出せることの有用性や実用性，そして何よりその楽しさを体験してもらう．

授業計画（全7回）

1. Racketの言語指向機能の学習（1〜4回）： 資料の輪講
2. 途中休憩（2〜3週）：自作言語の構想を練る期間（必要に応じて個別相談も可）
3. 自作言語の作成（5〜7回）：グループワーク（2〜4名で1グループ）

評価方法

出席（必ず毎回出席すること）と最終成果による評価を行う．最終成果としてソースコードの提出および自作した言語のプレゼンテーションを行ってもらう．

教材サンプル

Stacker

stacker.rkt

```
#lang br/quicklang

(define (read-syntax path port)
  (define src-lines (port->lines port))
  (define src-datums (format-datums '(handle ~a) src-lines))
  (define module-datum `(module stacker-mod "stacker.rkt"
                          ,@src-datums))
  (datum->syntax #f module-datum))
(provide read-syntax)

(define-macro (stacker-module-begin HANDLE-EXPR ...)
  #'(#%module-begin
     HANDLE-EXPR ...
     (display (first stack))))
(provide (rename-out [stacker-module-begin #%module-begin]))

(define stack empty)

(define (pop-stack!)
  (define arg (first stack))
  (set! stack (rest stack))
  arg)

(define (push-stack! arg)
  (set! stack (cons arg stack)))

(define (handle [arg #f])
  (cond
    [(number? arg) (push-stack! arg)]
    [(or (equal? * arg) (equal? + arg))
     (define op-result (arg (pop-stack!) (pop-stack!)))
     (push-stack! op-result)]))
(provide handle)

(provide + *)
```