情報実験II
Lisp処理系の開発

担当: 鈴木 徹也

2017年10月4日

1 課題

- Lisp 言語 (Scheme のサブセット) のインタプリタを開発する.
- 開発したインタプリタ上で動作する応用プログラムを開発する.

2 要求

2.1 Lisp言語

- 開発する Lisp 言語の動作は別資料「情報実験 II のための Scheme 入門」[1] で説明する Lisp 言語 (Scheme) に概ね準拠するものとする.
  - 細かい仕様は自分たちで決める.
- 別資料「情報実験 II のための Scheme 入門」の第 15 節「応用 1」に書かれているプログラム (もしくはそれに相当するプログラム) が記述できる.
- タートルグラフィックスのプログラムを記述できる.
- 少なくとも次のデータを扱える.
  - 整数値, 真理値, 記号, 未定義値, 組み込み手続き, 特殊形式, クロージャ, ドット対
- 少なくとも次の特殊形式を持つ.
  - `if`, `quote`, `lambda`, `define`, `set!`
- 特殊形式 `define` については少なくとも次の書式を扱える.

[1] http://www.sic.shibaura-it.ac.jp/~tetsuya/ の「情報実験 II」のページから PDF をダウンロードする.

  - (`define` *symbol expression*)

- 少なくとも次の組み込み手続きをあらかじめ扱える.
  - `+`, `-`, `*`, `/`, `cons`, `car`, `cdr`, `list`, `append`, `=`, `eq?`, `write`, `newline`, `map`, `exit`

2.2 Lisp インタプリタ

- 実装には Java 言語を用いる.
- Linux のシェル上から実行する.
- コマンド名は lisp.
- コマンドライン引数には Java プログラムの実行に必要なクラスパスを書かなくてよい.
- コマンドライン引数に Lisp プログラムのファイル名が指定されなかったら, 対話型で Lisp プログラムを実行する.
- コマンドライン引数に Lisp プログラムのファイル名が指定されたら, その Lisp プログラムをバッチ処理する.

“対話型” と “バッチ処理” の違いについては, 別資料「情報実験 II のための Scheme 入門」を参照すること.

3　スケジュール

9 月 27 日 (1) ガイダンス. Lisp の最初歩.

10 月 4 日 (2) 班分け. 課題詳細説明. Lisp の理解.

10 月 11 日 (3) GitHub リポジトリのクローン. 言語処理系の基礎

10 月 18 日 (4) Lisp 言語仕様の決定, 応用プログラムの決定, 言語処理系の外部設計

10 月 25 日 (5) Lisp 言語仕様の決定, 応用プログラムの決定, 言語処理系の外部設計

11 月 1 日 休講

11 月 8 日 (6) Lisp 言語仕様の決定, 応用プログラムの決定, 言語処理系の外部設計, 第 1 回レポート提出

11 月 15 日 (7) GitHub 概要. 言語処理系の内部設計

11 月 22 日 (8) 言語処理系の内部設計

11 月 29 日 (9) 言語処理系の内部設計

12 月 6 日 (10) 言語処理系の内部設計, 第 2 回レポート提出

12 月 13 日 (11) 実装とテスト

12 月 20 日 (12) 実装とテスト

12 月 27 日 休業期間

1 月 3 日 休業期間

1 月 10 日 (13) 実装とテスト

1 月 17 日 (14) 発表会

1 月 19 日 第 3 回レポート提出

注意

- 各レポート毎に, 担当のリーダーと連絡係を決める.
- リーダーは, メンバーのレポートへの貢献度を 5 段階で評価して, レポート提出の際に教員に報告する.
- 連絡係は教員に実験日に活動報告をする.
  - メールの件名は「情報実験 II Lisp 第 X 班 月/日 活動報告」とする.
  - 連絡先メールアドレスは `tetsuya-eie2-staff@sic.shibaura-it.ac.jp`

4 第1回レポート

- Lisp 言語仕様
  - 扱うデータ型
    - 数値, 真理値, 記号, 未定義値, 組み込み手続き, 特殊形式, クロージャ, ドット対, その他（自分たちで導入することにしたデータ型)
  - 構文
    - 字句の構文図式
    - S 式の構文図式
    - その他 (コメントの書き方など)
  - 組み込み手続き
    - 引数, 返り値, 副作用の説明, 発生しうるエラー
  - 特殊形式
    - 引数, 返り値, 副作用の説明, 発生しうるエラー
  - 式の評価方法
    - 数値, 真理値, 記号, 未定義値, 組み込み手続き, 特殊形式, クロージャ, ドット対, その他（自分たちで導入することにしたデータ型, 組み込み手続き）
  - 全エラーの種類と意味
- インタプリタの仕様
  - 動作環境
    - オペレーティングシステム, Java のバージョンなど
  - 実行方法
    - コマンド名, コマンドライン引数など
  - 対話型実行の画面例
  - バッチ処理の画面例
- 応用プログラム
  - 別資料「情報実験 II のための Scheme 入門」の第 15 節「応用 1」に書かれているプログラム (もしくはそれに相当するプログラム) のソースコード（テキストファイル）
  - タートルグラフィックスプログラムのソースコード（テキストファイル）
- 役割分担
  - 班の中での役割分担 (リーダー, 連絡係, その他)

4.1 第2回レポート

- レポート1の改訂版
- 各クラスの役割とクラス同士の関係 (astah のファイル)
  - クラス図 (必須), シーケンス図など
- 関連するファイルのディレクトリ構成
  - クラスファイル, シェルスクリプトなどの配置
- 組み込み手続きの実装方法
  - Java で記述する?
  - インタプリタ実行開始時に基本的な Lisp の手続きの組み合わせて定義する?
- 応用プログラムの設計
- テスト項目とテスト方法

  対象 Java プログラム, Lisp の各組み込み手続きと各特殊形式

  方法 手作業?, JUnit[2]のようなフレームワーク?
- 役割分担 (リーダー, 連絡係, その他)

4.2 第3回レポート

- 第1回レポートの改訂版
- 第2回レポートの改訂版
- 応用プログラムのテキストファイル, 応用プログラムの実行結果
- 役割分担 (リーダー, 連絡係, その他)

5 発表会

- 全員で発表する。
- 課題の内容は事前に TA が説明する。

[2]http://junit.org/

6 ヒント

6.1 応用プログラムの例

タートルグラフィックス (turtle graphics) を実装するのに必要な Lisp 言語の機能を考える. タートルグラフィックスとはコンピュータ画面上の仮想的な亀 (turtle) に次の指示を与えて描く図形である.

- 右方向もしくは左方向へ, 指定された角度だけ進行方向を変える.
- 亀が持っているペンを下ろす, もしくは上げる.
- 指定された距離だけ直進する. ペンが下ろされている時は, 移動の軌跡 (線分) を画面に描画する.

インターネットで検索をすれば, このタートルグラフィックスによる図形の描画方法や描画例がみつかる.

このタートルグラフィックスを実装するためには少なくとも次の機能が必要になるだろう.

- 小数点以下の値を扱える数値 (例えば 浮動小数点形式)
- 画面に線分を描画する関数

さらに次のことを決めなければならない.

- 数値の字句定義
- 角度の表現を度 (0-360) にするのかラジアンにするのか
- 線分を描画する関数の名前と引数, そして画面への描画方法
- 定義済み手続きでの浮動小数点形式の数値の扱い
  - 例えば, 整数と小数点以下の値を持つ数値との加算, 減算, 乗算, 除算, 比較についてどのように計算するかを決めなければならない.

6.2 実装順序の例

以下に言語処理系の基本的な仕組みを実装する順序の例を示す. この手順の後, 残りのデータ型, 組み込み手続き, 特殊形式を実装すると良いだろう.

1. S 式を構文解析できるようにする.
   - quote の省略記法 (') は後回しにしても良い.
2. 整数を評価できるように評価器を変更する.

```
lisp> 1
1
```

3. 環境を実装する.
4. 記号を評価できるように評価器を変更する.
5. 特殊形式 quote を実装する.
6. Lisp 処理系起動時に大域的なフレームにおいて, 記号 quote を特殊形式 quote に束縛する.
7. 特殊形式 quote を使った S 式を評価できるように, 評価器を変更する.

```
lisp> (quote (1 2 3))
(1 2 3)
```

8. 組み込み手続き car を実装する.
9. Lisp 処理系起動時に大域的なフレームにおいて, 記号 car を組み込み手続き car に束縛する.
10. 組み込み手続き car を使った S 式を評価できるように, 評価器を変更する.

```
lisp> (car (quote (1 2 3)))
1
```

11. 組み込み手続き cdr を実装する.
12. Lisp 処理系起動時に大域的なフレームにおいて, 記号 cdr を組み込み手続き cdr に束縛する.

13. 組み込み手続き cdr を使った S 式を評価できるように, 評価器を変更する.

```
lisp> (cdr (quote (1 2 3)))
(2 3)
```

14. 特殊形式 define を実装する.

    - (define 記号 S 式) の形式に対応する.

15. Lisp 処理系起動時に大域的なフレームにおいて, 記号 define を特殊形式 define に束縛する.

```
lisp> (define x 1)
x
lisp> x
1
```

16. 特殊形式 lambda を実装する.

17. Lisp 処理系起動時に大域的なフレームにおいて, 記号 lambda を特殊形式 lambda に束縛する.

```
lisp> (define add1 (lambda (x y) (+ x y)))
add1
lisp> (add1 2 3)
5
```

18. 特殊形式 define を実装する.

    - (define (関数名 引数リスト) 関数本体) の形式に対応する.

```
lisp> (define (add2 x y) (+ x y))
add2
lisp> (add2 2 3)
5
```