2018年 後期 Lisp班　活動報告書

西見元希　松本幸大　岡本陽太　堀越俊行　服部瑠斗　浜田直弥　北村優奈　石井創　大柴寿浩

目次

1　活動の概要

文責：西見元希

この班は、プログラミング言語Common Lispの基本的文法や環境構築法、関数型プログラミングの考え方について学ぶことを目的として活動した。活動日は水曜5限と金曜4限の2回であり、各自どちらかに参加した。Common Lisp入門に適した本は備品に無かったため、プロジェクトリーダーが外部サービスScrapboxを用いて学習用の資料を作成し、それを確認しながら進めていくという勉強会形式を取った。

2　Lisp について

文責：西見元希

1. Lispの概要
    Lispは現在広く利用されている高水準プログラミング言語言語の中でもFORTRANに次いで2番めに古い言語であり、前置記法などを特徴とする。様々な方言を持ち、その中でもCommon LispとSchemeはよく知られている。このプロジェクトではCommon Lispを用いた。
2. Common Lispの概要
    Common Lispは増え続ける膨大なLisp方言を統一し、標準化しようという目的のもと作られた。Lispと聞いて一般に想像されるような関数型プログラミング言語という側面だけではなく、手続き処理やオブジェクト指向プログラミングの機能も標準として備えている。また、強力なメタプログラミングの機能を備えていることもその特性の一つである。今回Common Lispを選んだ理由としては、clispという比較的多くの環境で利用可能な処理系が存在していることや、班員に経験者が数人いたことなどが挙げられる。しかし今回学習したことの多くはSchemeやその他Lisp方言に対しても応用が可能であるだろう。

3　学習内容

文責：西見元希

全5週、計10回行ったプロジェクト活動で学んだ内容を以下に簡潔に示す。
また、作成した資料は次のページで閲覧が可能。-> https://scrapbox.io/ritscc/Lisp%E7%8F%AD2018

3.1 第1回

文責：松本幸大

1. SBCL または CLISP のインストール
2. Lisp の基本を理解するための、`print` 関数や四則演算、変数宣言、代入の関数を用いたサンプルコードの確認

```
(print "Hello, world")
```

```
CL-USER> (+ 1 2)
3
CL-USER> (- 4 6)
10
CL-USER> (* 3 4)
12
CL-USER> (/ 5 10) ; 有理数表示
1/2
CL-USER> (float (/ 5 10)) ;  小数表示
0.5
CL-USER> (sqrt -1) ; 複素数
#C(0.0 1.0)
CL-USER> (defparameter *i* (sqrt -1)) ; グローバル変数の宣言/初期化
*I*
CL-USER> (defparameter *z* *i*)
*Z*
CL-USER> (setf *z* (+ 1 *z*)) ; 代入
#C(1.0 1.0)
CL-USER> (abs *z*) ; 絶対値を求める abs 関数
1.4142135
```

3. 評価と破壊的変更

```
CL-USER> (1+ *a*) ; 破壊的でない例 (*a* の値は変化しない)
2
CL-USER> (incf *a*) ; 破壊的な例 (*a* の値が、1 足されたものに更新される)
2
```

4. `quote` の挙動

シングルクォートまたは `quote` を使うと、指定されたシンボルやＳ式は評価されずに返る

```
CL-USER> *a*
2
CL-USER> '*a*
*A*
CL-USER> (+ 2 3)
5
CL-USER> '(+ 2 3)
(+ 2 3)
```

5. リストの構造と `cons`, `car`, `cdr` 関数

- リストの中にはコンスセルという単位がある
- 2つのポインタを持つことのできる構造体のようなもの
- 片方に値のアドレス、もう片方に別のコンスセルのアドレスを持つ
- 「リスト」=「コンスセルの連なり」

```
CL-USER> (defvar *list* (cons 2 (cons 3 nil)))
*LIST*
CL-USER> *list*
(2 3)
CL-USER> (car *list*)
2
CL-USER> (cdr *list*)
(3)
CL-USER> (cadr *list*)
3
CL-USER> (cddr *list*)
nil
```

6. atom と pair について

- pair: 評価すると Lisp のリストになるものすべて
- atom: pair でないもの

7. Lisp の S 式について

- S 式はリスト (car 部が「オペレータ」、cdr 部が引数として扱われる)
- オペレータは 3 種類
    - 関数: すべて引数を評価する
    - マクロ: 引数の一部を評価しないものもある
    - 特殊形式: `if`, `quote`, `block` など、関数でもマクロでもないもの

8. 関数定義/呼び出し

9. ローカル変数とローカル関数

- `let` でローカル変数を、`labels` でローカル関数を作ることができる
- どちらも、指定した S 式外では使うことが出来ない

10. クロージャ

- 環境として保存されるローカル変数を持った関数 (状態を持った関数とも言える)

演習内容

- 行列の足し算・掛け算を関数として定義する
  - 行列は二重リストで☒表現する
  - `list` 関数: 引数すべてをリストにして返す
- `atom` 関数を用いて、様々なオブジェクトが atom なのか pair なのかを判定する
- クロージャの内部状態を変更できる関数 `make-adder` を定義する
  - この関数は引数を 1 (または 2) 個取る
  - 1 個の場合は、クロージャに保存されている値をその数に足して返す
  - 2 個の場合は、クロージャに保存されている値を 2 個目の引数の値に変更してから 2 数を足して返す
  - `if` 関数で 2 つ以上のS式を実行したいときは `progn` 特殊形式を使う
    例: `(progn exp1 exp2)`
  - 引数の個数を変更する場合、`&optional` キーワードを用いる

`&optional` を用いた例

```
(defun foo (x &optional y)
  (if y
    (+ x y)
    x))
```

3.2 第2回

文責：松本幸大

- 比較関数 (比較できる対象が少ないほど動作が高速なため、使い分けが重要)
  - `eq` : シンボルの比較のみ
  - `eql` : シンボルに加えて数と文字に対応
  - `equal` : シンボルの時は `eq`, 数や文字☒の時は `eql` の挙動をして、文字列やリストの時は各要素を比較する
  - `equalp` : 構造体やハッシュテーブルなどのデータ構造の比較が可能
  - `=` : 数の比較のみ
  - `string=` : 文字列の比較のみ
- 条件分岐
  - `if`: 特殊形式
  - `cond` / `when` / `unless` / `case` マクロ
- lambda (無名関数)
  - 関数定義をせずにその場だけで呼び出したい関数オブジェクトを生成する

- 高階関数
    - `mapcar` 関数: リストの要素すべてに同じ関数を適用する
    - `apply` 関数: リストの要素すべてを引数としてオペレータに与える
    - `reduce` 関数: リストの畳込み

演習内容

- 恒等関数 `f_i` を定義する
- `x`, `y` を引数に取って必ず `x` を返す関数 `f_k`
- `x` (2 引数関数), `y` (1引数関数), `z` を引数にとって、`x` を `z` に適用したものに、`y` を `z` に適用したものを適用させる関数 `f_s` (定義のみ可能で実際には動作しない)
- リストの長さを求める関数 `my-length`
- リストの要素の最大値を求める関数 `my-max` (要素はすべて 0 以上とする)
- 階乗を計算して返す 1 引数関数 `factorial`
- リストの要素をすべて 2 乗して返す関数 `double-list`

`funcall` を用いた例

```
(defun one_apply (f x) (funcall f x))
```

3.3　第3回

文責：堀越俊之

第3回の活動では、第2回以前で勉強した、関数、lambda 関数、高階関数やマクロなどを使って Lisp の演習を行った。 今回は再帰の演習を行うことに重点を置き、特段新しいものは学ばず、演習するにとどめた。演習は指示された関数を実装するという形式で行った。

演習内容

(1) 関数定義と抽象化の基本的な手順に関する問題

1. 底辺と高さを受け取って平行四辺形の面積を返す関数
2. 同様に三角形の面積を返す関数
3. 三角形の面積と高さを受け取って三角錘の体積を返す関数

(2) 変数のスコープとリスト操作に関する問題

1. 左から順に要素として `1`, `2`, `3` をもつリスト、`a` をグローバルに宣言
2. `a` の最初の要素を `10` に破壊的に変更
3. `a` の最初の要素を `1` に戻す
4. リストと値を受け取ってそのリストの最初の要素をその値に(破壊的に)変更する関数
5. 同様にリストの `2` 番目、`3` 番目の要素を変更する関数
6. リストと値と `n` を受け取ってリストの `n` 番目の要素をその値に変更する関数

(3) 素数判定を行うプログラムを関数プログラミングスタイルで実装する問題

1. 自然数 x を受け取って、 x が偶数のとき T、そうでないとき NIL を返す関数
2. 同様に 3 の倍数、4 の倍数の場合の関数
3. 自然数 x, y を受け取って x が y で割り切れるとき T、そうでないとき NIL を返す関数 foo
4. 自然数 n を取って n の平方根以下の最大の整数を返す関数 my-root
5. 自然数 n を受け取って、2からnまでの数を要素に持つリストを作る関数mklist(再帰)
6. 自然数 n を受け取って、(foo n x) を (mklist (my-root n)) によってできたリストにマップする関数 mapfoo
7. 自然数 n を受け取り、(mapfoo n) で生成されるリストの中にTがあれば NIL、そうでなければ T を返す関数

演習内容

再帰の練習問題

1. 自然数 n を受け取って、1 から n までの和を返す関数 my-sum
2. 自然数 n を受け取って、1 から n までの積を返す関数 my-multi
3. 自然数 n を受け取って、 n の階乗を返す関数 fact
4. リストを受け取ってリストの長さ(length)を返す関数 my-length
5. リストを受け取って要素の順を反転させたリストを返す関数 my-reverse
6. リストを受け取って全く同じコピーを返す関数 list-copy
7. 積算 * を + 関数で再帰的に計算する関数 multi
8. 引数 x, y を受け取って x の y 乗を返す関数
9. 2 つのリストを破壊的に連結する関数 my-nconc
10. リストと値を受け取って、その値がリストの要素に含まれていれば、それ以降を返す関数 my-member

3.4 第4回

文責：服部瑠斗

第 4 回の活動では主に以下の 2 つの内容に取り組んだ。

1. 前回の演習問題 前回の演習問題は、主に再帰について取り扱った。 演習問題は10問あり受け取った自然数の総和を求める関数からリストなどを絡めた関数まで再帰のみの演習ではなく、 再帰 + 以前までの範囲といった幅広い範囲での演習だった。
2. 組み込みマクロについて 今回取り扱った組み込みマクロは主に 2 つある。ひとつ目は dotimes だ。 dotimes とは、(dotimes (var limit result) S式 ...) という構文で用いることができる。このマクロの効果は、limit の回数 S 式を評価するというものだ。 ふたつ目は loop / return だ。 loop / return は loop 内の S 式を無限に繰り返し、return を評価すると loop を終了するというものだ。

3.5 第5回

文責：服部瑠斗

第 5 回の活動では主に以下の 2 つの内容に取り組んだ。

1. マクロに必用な文法の復習 マクロに必用な文法の復習は主にバッククォートについて取り組んだ。 バッククォートがつくかつかないかで評価結果が大幅に変わることを確認した。

2. マクロについて マクロについては、理論について学習した。 マクロを使うことによって括弧の数を減らすことが可能になったり、 メタプログラミングとマクロの関係性について学習した。 また Lisp の

式評価の順番や、C言語のマクロとの相違点についても学習した。

4　活動で得られたもの

- 手続き型言語との違いを体感することによって、今までより洗練されたコードを書くことができるようになった。
- 自分の今まで書いたことがある環境と全然違う書き方でコードを書き違う世界を経験した。
- 考え方として、最後から処理したり、すぐに一つの関数をローカルに定義したりと、Lispとは関係のないプログラミング言語でも通用する考えを獲得した。
- Common Lispをさわり、オブジェクト指向としてのプログラミング経験を得た。

5　問題点

文責：西見元希

問題点として、活動が週一回であったために、演習量が少なく知識が身につきにくかったというものが挙げられた。宿題として問題を用意した回も合ったが班員の余裕がなく、毎回の活動の最初に復習の時間を設けていた。対策としては、活動の最後に行う演習の量を増やしたり、具体的な制作活動を行うなどが挙げられる。 また、プロジェクトリーダーが不在のときに活動を進めることができなかったという問題もあった。これに関しては資料を前もって準備しているので班員に行う内容を指示することで復習時間に当てるなどの対策がとれたのではないかと考えている。

6　展望

文責：西見元希

今回の活動ではいわゆる関数型プログラミングと呼ばれるプログラミング手法やマクロを用いたメタプログラミングについて、班員全員に知ってもらうことができた。今後の展望としてはこのメタプログラミングに注目し、Lispによらず様々な言語でのメタプログラミング手法を学習・研究する活動を行いたいと考えている。