HyperSpec ひとめぐり

こんなことまで決まってますよ、Common Lisp では

NANRI Masaoki (南里 征興)

いちおう自己紹介

- NANRI
- Common Lisp勉強中
- テキストエディタxyzzy使い
- xyzzy lispのリファレンスの世話とかしてました

HyperSpecとは

- ANSI Common Lisp の仕様
- Lispworksのウェブサイトで全内容が読めます
- 1300ページを越えるボリューム

http://www.lispworks.com/documentation/HyperSpec/Front/

CLtL2とは

- Common Lisp the Language 2nd Edition
- Common Lispの仕様を決める際の途中経過を出版したもの
- 日本語訳が出版されています

全内容に触れるのは無理なので

- 感心したところ
- 他の言語仕様に無さそうなところ

をお話します

数の体系

Schemeの数の体系も大体同じなので、Schemerの方々も復習として聞いてください

数の種類

- 整数
- 有理数
- 実数
  - 浮動小数点数
- 複素数

正確数と非正確数の区別

有理数の正準化

分数の分母が1のとき

```
CL-USER> 2/1
2
CL-USER> 10/1
10
```

有理数ではなく、整数になる

複素数の正準化

複素数の虚部が0のとき

```
CL-USER> #C(1 0)
1
CL-USER> #C(1/2 0)
1/2
```

複素数ではなく、整数や有理数になる

複素数の正準化

複素数の虚部が0.0のとき

```
CL-USER> #C(1.0 0.0)
#C(1.0 0.0)
CL-USER> #C(0.5 0.0)
#C(0.5 0.0)
```

非正確数だから正準化は起こらない

Common Lispには
コンパイラが含まれます

「3.2 Compilation」
http://www.lispworks.com/documentation/HyperSpec/Body/03_b.htm

コンパイルする関数 compile

Syntax:
compile *name &optional definition*
=> *function, warnings-p, failure-p*

例

```
(compile 'foo)
(compile nil '(lambda () "replaced"))
```

http://www.lispworks.com/documentation/HyperSpec/Body/f_cmp.htm

コンパイラがあって嬉しいこと

ソースコードを実行する前にコンパイラのチェックを受けられる

- テストを考える
- 時間のかかるテストの実行

を後回しにできる

他の言語では

- java
  - 標準ライブラリ java.lang.Compiler
- Haskell
  - 「11. Compiler Pragmas」

逆アセンブルする関数disassemble

Syntax:
disassemble *fn*
=> ***nil***
例
(defun f (a) (1+ a))
(disassemble 'f)

http://www.lispworks.com/documentation/HyperSpec/Body/f_disass.htm

Common Lispには
デバッガが含まれます

デバッガを起動する関数
invoke-debugger

Syntax:
invoke-debugger *condition*
=>|

http://www.lispworks.com/documentation/HyperSpec/Body/f_invoke.htm

デバッガがあって嬉しいこと

あって当たり前なので、特に思いつきませんでした
Common Lispではエラーが起こるとデバッガに移行します

Slimeを使っているとデバッガの処理系依存の部分に触れないので、機能とかあまり知らなかったり……

Common Lispには
エディタが含まれます

エディタを起動する関数 ed

Syntax:
ed *&optional x*
=> *implementation-dependent*

xが関数名のときは、その関数定義を開く

http://www.lispworks.com/documentation/HyperSpec/Body/f_ed.htm

まとめっぽいもの

- HyperSpecはよく考えられている
- プログラムを書く環境まで仕様の中に入っている

Common Lispの思想を少しでも感じ取っていただけましたでしょうか

デモ