PRINTER

Programování v Common Lispu

Printer

- tisk objektů tak, aby výsledek byl čitelný readerem
- zpětným přečtením by měl vzniknout "ekvivalentní" objekt
- funkce pro tisk: `write`, `print`, `prin1`, `princ`, `pprint`, `write-to-string`, `prin1-to-string`, `princ-to-string`, `format`
- proměnné ovlivňující tisk: `*print-base*`, `*print-radix*`, `*print-case*`, `*print-circle*`, `*print-escape*`, `*print-gensym*`, `*print-readably*`, `*print-array*`, `*print-length*`, `*print-level*`, `*print-lines*`, `*print-miser-width*`, `*print-pprint-dispatch*`, `*print-pretty*`, `*print-right-margin*`

Printer je programovatelný

```
;(lepší verze je přiložena: bicons.lisp)

(defclass bicons ()
  ((val :accessor val :initarg :val :initform nil)
   (prev :accessor prev :initarg :prev :initform nil)
   (next :accessor next :initarg :next :initform nil)))

(defmethod make-load-form ((obj bicons) &optional env)
  (make-load-form-saving-slots obj :environment env))

(defmethod biconsp (obj)
  nil)

(defmethod biconsp ((obj bicons))
  t)

(defun bicons (val prev next)
  (make-instance 'bicons :val val :prev prev :next next))

(defun bilist (&rest elements)
  (when elements
    (let ((result (bicons (car elements)
                          nil
                          (apply #'bilist (cdr elements)))))
      (when (biconsp (next result))
        (setf (prev (next result)) result))
      result)))
```

Printer je programovatelný

```
(defun left-brack-reader (stream char)
  (declare (ignore char))
  (apply #'bilist (read-delimited-list #\] stream t)))

(set-macro-character #\[ #'left-brack-reader)

(defun right-brack-reader (stream char)
  (declare (ignore stream char))
  (error "Non-balanced #\] encountered."))

(set-macro-character #\] #'right-brack-reader)
```

```
(defmethod print-object ((object bicons) stream)
  (princ #\[ stream)
  (loop for bc = object then (next bc)
        do (write (val bc) :stream stream)
        while (next bc)
        do (princ " " stream))
  (princ "] " stream))
```

```
> [1 2 3]
[1 2 3]

> (val *)
1

> (next **)
[2 3]
```

Tisk nečitelných objektů

```
(defclass my-obj ()
  ((val :reader val :initarg :val :initform 0)))

(defmethod print-object ((obj my-obj) stream)
  (if (or *print-escape* *print-readably*)
      (print-unreadable-object (obj stream :type t :identity t)
        (format stream "Val ~s" (val obj)))
    (format stream "Můj objekt ~s" (val obj))))
```

```
> (make-instance 'my-obj)
#<MY-OBJ Val: 0 200BB47F>

> (princ *)
Můj objekt 0
#<MY-OBJ Val: 0 200BB47F>

> (format t "Pro stroj: ~s, pro lidi: ~a" * *)
Pro stroj: #<MY-OBJ Val: 0 200BB47F>, pro lidi: Můj objekt 0
NIL
```