Conditionnels

```
(cond ((> x 3) (setf y (g ...)) (+ x y))
      (finished (+ x 15))
      (t 0))

(case (f ...)
  ((apr jun sept nov) 30)
  (feb (if (leap-year) 29 28))
  (t 31))

(typecase ...)
```

Itération

```
(do ((i 0 (1+ i))
     (j n (1- i)))
    ((<= j 0) (+ x y))
  (format t "~a ~a~%" i j)
  (f i j))

(do* ...)

(dolist (elem (mapcar ...) 'done)
  (format t "~a~%" elem))

(dotimes (i 10 (+ x y))
  (format t "~a~%" i))

(mapc ...)
```

Itération

```
(loop for i from 1 to (compute-top-value)
      while (not (unacceptable i))
      collect (square i)
      do (format t "Working on ~D now~%" i)
      when (evenp i)
        do (format t "~D is a non-odd number~%" i)
      finally (format t "About to exit!"))
```

Valeurs multiples

```
* (multiple-value-bind (x y) (floor 75 4)
    (list x y))

(18 3)
* (values (cons 1 2) 'hello (+ 3 4))

(1 . 2)
HELLO
7
* (multiple-value-call #'+ (floor 75 4))

21
* (multiple-value-list (floor 75 4))

(18 3)
*
```