I.T.I.S. & L.S.
F. GIORDANI
TECLIM

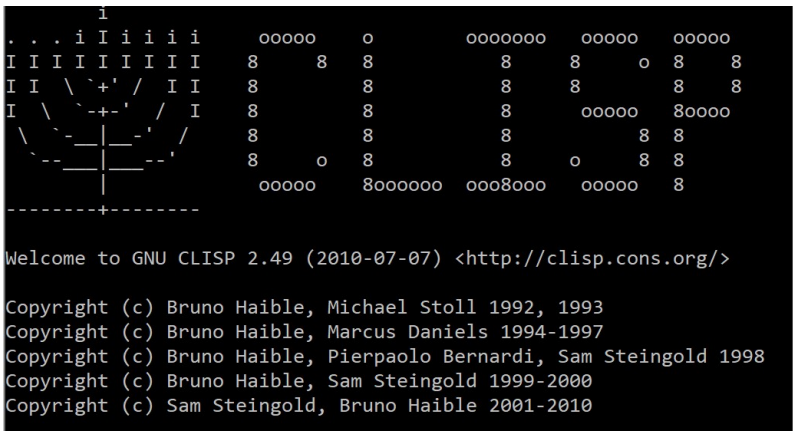
ITIS-LS "Francesco Giordani" Caserta

prof. Ennio Ranucci

a.s. 2019-2020

Semplici esercitazioni Lisp

Esercitazioni GNU CLisp 2.49



Il Lisp è uno dei linguaggi più antichi (John McCarthy,1960) utilizzato diffusamente ancora oggi. Il nome deriva da LISt Processor, infatti si basa su poche e chiare regole sintattiche, prima delle quali è il concetto di lista.

Programmazione funzionale: PROGRAMMA=FUNZIONE

La programmazione funzionale o programmazione orientata alle espressioni tratta espressioni, le funzioni vengono usate come oggetti,ovvero vengono trattate in un programma in modo strettamente analogo alle variabili. Un programma puramente funzionale viene strutturato definendo un'espressione che racchiude l'obiettivo del programma. Ogni termine dell'espressione è a sua volta un'affermazione relativa ad una caratteristica del problema(magari in capsulata dentro un'altra espressione) e la valutazione di tutte queste espressioni alla fine porta al risultato. Il flusso di esecuzione del programma assume la forma di una serie di valutazioni di funzioni matematiche.

L'esempio più vecchio di linguaggio funzionale è ilLisp(ListProcessor) spesso usato nei progetti di intelligenza artificiale,anche se né ilLisp originale né iLisp moderni,come il Common Lisp e le sue varianti (Logo,Scheme,DylaneClojure)sono puramente funzionali. Altri linguaggi funzionali sono l'Haskelle ilinguaggi della famigliaML(MLeOcaml,F# sviluppato da Microsoft all'interno delframework.NET). Altri linguaggi, come per esempio Ruby, Python, Perl e TCL, possono essere usati in stile funzionale.

Il Lisp si presenta con un Prompt di inserimento comandi esattamente come farebbe una shell a comandi. Inserendo una lista composta da elementi separati da spazi tra parentesi() otteniamo l'esecuzione, il processore Lisp valuta il comando inserito, sommando tutti termini che compongono la lista. Nel Lisp gli oggetti usati sono le S-expression(symbolic expressions) la cui sintassi è:

(comando liste liste liste ...) Il Lisp non tipizza il dato delle variabili. Il codice di un programma può essere scritto sottoforma di S-expression, utilizzando la notazione prefissa.

ITIS-LS "Francesco Giordani" Caserta
Anno scolastico: *2019/2020*
Classe 3^ sez.B spec. Informatica e telecomunicazioni
Data:
Numero progressivo dell'esercizio: *es0*
Versione: *1.0*
Programmatore/i:
Sistema Operativo: *Windows 10*
Compilatore/Interprete: *GNU CLisp 2.49*
Obiettivo didattico:
Conoscenza dell'ambiente di programmazione
Obiettivo del programma:
definire una funzione che stampa a video un saluto

```
;Saluto
(defun Ciao (N)
(concatenate 'string "Ciao " N)
)
 Oppure
(defun Ciao (N)
 (setq x (cons "Ciao" N))
)
; per caricare il file ciao.lsp (load "ciao.lsp)
; (load "C:/users/ennio/desktop/scuola/lisp/ciao.lsp")
;(ciao "Caserta")
```

Nota:

Assegnamento: (setq x 5) questa istruzione si limita ad inserire nella variabile di nome x il numero intero 5

Scrivere messaggi per l'utente:(print"ciao")

ITIS-LS "Francesco Giordani" Caserta
Anno scolastico: *2019/2020*
Classe 3^ sez.B spec. Informatica e telecomunicazioni
Data:
Numero progressivo dell'esercizio: *es1*
Versione: *1.0*
Programmatore/i:
Sistema Operativo: *Windows 10*
Compilatore/Interprete: *GNU CLisp 2.49*
Obiettivo didattico:
Definire una funzione
Obiettivo del programma:
definire una funzione che, dato un numero restituisca il suo cubo

```
(defun CUBO (N)

(* N N N)

)

;Esempio di una possibile chiamata della funzione:

; ? (cubo 5)

; 125

; per caricare il file cubo.lsp (load "cubo.lsp)
; (load "C:/users/ennio/desktop/scuola/lisp/cubo.lsp")
```

Le liste sono rappresentabili da una sequenza di coppie di celle dette CONS. Ogni cons contiene due puntatori: CAR e CDR, che indirizzano il contenuto ed il cons successivo. Nel caso di liste annidate il car può puntare ad un ulteriore cons.
Il cdr dell'ultimo elemento contiene NIL (è vuoto).
Il lisp ammette anche una rappresentazione più esplicita (e basica) dell'oggetto CONS, come coppia puntata:

```
(car . cdr)
```

Ognuno dei mattoni fondamentali (CONS, car e cdr) ammette un'operatore corrispondente:

```
(cons 'a 'b) ; forma un cons (A . B)
>>(cons 1 (cons 2 (cons 3 nil))) ; forma una lista
(1 2 3)
>>(car '(1 2 3)) ; estrae il primo elemento dalla lista
1
>>(cdr '(1 2 3)) ; estrae la restante parte della lista
(2 3)
```

Da notare come la formazione di liste sia possibile semplicemente reiterando l'operatore cons.

```
>>(atom '(1 2))
NIL
```

che ritorna true se l'oggetto è un'atomo, cioè un singolo simbolo, NIL (cioè false) se è una lista.

```
>>(setq a '(1 2 3 4 5)) ; assegna ad a la lista (1 2 3 4 5)
>>(cons 0 a) ; aggiunge alla lista l'elemento 0 (0 1 2 3 4 5)ma non modifica il
valore di a (1 2 3 4 5)
>>(setq a (cons 0 a)) ; modifica a restituisce (0 1 2 3 4 5)
```

Per estrarre elementi (esistono anche first, second, third, ecc.):

```
>>(nth 1 '(a b c d)) restituisce  B
>>(subseq '(a b c d) 1 3)restituisce (B C)
```

Per concatenare, intersecare, unire, ecc:

```
>> (union '(a b c) '(b c d)) restituisce (a b c d)
>> (intersection '(a b c) '(b c d)) restituisce (b c)
```

```
>> (append '(a) '(b c d) '(e f)) restituisce (a b c d e f)
```
Per ottenere la dimensione di una lista:
```
>>(length '(a b c d))restituisce 4
```
Per rimuovere un elemento:
```
>>(remove c '(a b c d)) restituisce (a b d)
```
Per estrarre il primo elemento o i restanti (analoghe ai già visti car e cdr):
```
>>(first '(a b c d)) restituisce A
>>(rest '(a b c d))restituisce  (b c d)
```
Inoltre, essendo car e cdr molto utilizzati, esistono anche una serie di operatori combinati, ad esempio cddr equivale a (cdr (cdr ...)), cadr equivale a (car (cdr ...)), ecc.
Per estrarre elementi da una lista:
```
>>(member 'c '(a b c d))restituisce (c d)
>>(find 'a (a b c d)) restituisce A
```
Gli operatori push e pop permettono di lavorare su una lista con la logica dello stack:
```
>>(setq a nil)restituisce NIL
>>(push 4 a) restituisce (4)
>>(push 5 a)restituisce (5 4)
>>(pop a)restituisce 5
```
L'operatore sort permette di utilizzare come condizione di ordinamento una funzione booleana arbitraria:
```
>>(sort '(2 3 7 1) '>)restituisce (7 3 2 1)
```

Le espressioni condizionali

Esistono diversi operatori di confronto:
```
>>(eq 'a 'A) ; confronto tra simboli restituisce T
>>(= 3 4) ; confronto tra numeri restituisce NIL
>>(equal '(1 2 3) '(1 2 3)) ; confronto tra liste restituisce T
```
Inoltre l'operatore eql si comporta come = per i numeri e come eq per i simboli.
L'operatore if necessita sempre di entrambe le condizioni (true e false):
```
(if test val-true val-false)
```
dove val-true/false possono ovviamente essere ogni simbolo o lista, quindi anche altro codice.
Se invece non serve una delle due condizioni si usano gli operatori when e unless:
```
(when test val-true)
(unless test val-false)
```
A differenza di if gli operatori when e unless ammettono un numero arbitrario di istruzioni:
```
>>(when t (setq a 5) (+ a 6)) restituisce 11
```
Se invece le condizioni sono molte esiste l'operatore cond:
```
(cond ((test1 val1)
(test1 val2)
...)))
```
Per un confronto tra molti valori è disponibile l'operatore case:
```
 (case var (val1 ret1)(val2 ret2)...)
```
che valuta il ret corrispondente al val corrente di var.
```
(case a
(0 "zero")
(1 "uno")
(2 "due")
)
```
Gli operatori and e or, molto utilizzati nel controllo di flusso, accettano un numero arbitrario di argomenti, ed effettuano il confronto in modo sequenziale tra di essi.

ITIS-LS "Francesco Giordani" Caserta
Anno scolastico: *2019/2020*
Classe 3^ sez.B spec. Informatica e telecomunicazioni
Data:
Numero progressivo dell'esercizio: *es2*
Versione: *1.0*
Programmatore/i:
Sistema Operativo: *Windows 10*
Compilatore/Interprete: *GNU CLisp 2.49*
Obiettivo didattico:
Struttura che controlla il flusso delle istruzioni: La selezione(If)
Obiettivo del programma:
Restituire il valore assoluto

```
;*****************************************************************

;*  valass.lsp                                                   *

;*  20/01/2003       GNU CLisp 2.49                              *

;*  valore assoluto                                              *

;*****************************************************************

;(valass -5 )  -> 5
(defun valass (X)
  (if (< X 0) (* -1 X) X)
)
```

ITIS-LS "Francesco Giordani" Caserta
Anno scolastico: *2019/2020*
Classe 3^ sez.B spec. Informatica e telecomunicazioni
Data:
Numero progressivo dell'esercizio: *es3*
Versione: *1.0*
Programmatore/i:
Sistema Operativo: *Windows 10*
Compilatore/Interprete: *GNU CLisp 2.49*
Obiettivo didattico:
Struttura che controlla il flusso delle istruzioni: La selezione(If).
Tecnica della contatore ricorsivo
Obiettivo del programma:
definire una funzione che restituisca il Massimo tra 3 valori

```
;(max3 15 13 9)  -> 15

(defun max3 (A B C)

  (if (> A B)

      (if (> A C) A C)

    (if (> B C) B C)

  )

)
```

ITIS-LS "Francesco Giordani" Caserta
Anno scolastico: *2019/2020*
Classe 3^ sez.B spec. Informatica e telecomunicazioni
Data:
Numero progressivo dell'esercizio: *es4*
Versione: *1.0*
Programmatore/i:
Sistema Operativo: *Windows 10*
Compilatore/Interprete: *GNU CLisp 2.49*
Obiettivo didattico:
Struttura che controlla il flusso delle istruzioni: La selezione(If).
Tecnica della contatore ricorsivo
Obiettivo del programma:
definire una funzione che restituisca la lunghezza di una lista

```
; ? (LUNG '(R S U))
;3
(defun LUNG (L)

 (cond

   ((null L) 0)

   (t (+ (LUNG (cdr L)) 1))

 )
)
```

Trace
lung(r s u)->+ lung(s u) 1 -> + (+ lung(u) 1) 1 ->+ + (+ lung() 1) 1 1->+ + (+ 0 1) 1 1->+ +1 1 1-> + (+1 1) 1->+ 2 1-> 3

ITIS-LS "Francesco Giordani" Caserta
Anno scolastico: *2019/2020*
Classe 3^ sez.B spec. Informatica e telecomunicazioni
Data:
Numero progressivo dell'esercizio: *es5*
Versione: *1.0*
Programmatore/i:
Sistema Operativo: *Windows 10*
Compilatore/Interprete: *GNU CLisp 2.49*
Obiettivo didattico:
Struttura che controlla il flusso delle istruzioni: La selezione(If).
Tecnica della contatore ricorsivo
Obiettivo del programma:
definire una funzione che restituisca quanti sono I numeri positivi di una lista

```
? (posit '(1 -1 9))
2
(defun posit (L)

  (if (null L) 0

    (if (> (car L) 0)  (+ (posit (cdr L)) 1)

       (posit (cdr L))

    )

   )

)
```

ITIS-LS "Francesco Giordani" Caserta
Anno scolastico: *2019/2020*
Classe 3^ sez.B spec. Informatica e telecomunicazioni
Data:
Numero progressivo dell'esercizio: *es6*
Versione: *1.0*
Programmatore/i:
Sistema Operativo: *Windows 10*
Compilatore/Interprete: *GNU CLisp 2.49*
Obiettivo didattico:
Struttura che controlla il flusso delle istruzioni: La selezione(cond)
Tecnica della somma ricorsiva
Obiettivo del programma:
Calcolo della somma degli elementi di una lista q

```
;(lsomma '(2 3 4)) ->9

(defun lsomma (q)
     (cond

        ((null q) q)

        ((null (cdr q)) (car q))

        (t (+ (car q) (lsomma (cdr q))))
     )
)
```

ITIS-LS "Francesco Giordani" Caserta
Anno scolastico: *2019/2020*
Classe 3^ sez.B spec. Informatica e telecomunicazioni
Data:
Numero progressivo dell'esercizio: *es7*
Versione: *1.0*
Programmatore/i:
Sistema Operativo: *Windows 10*
Compilatore/Interprete: *GNU CLisp 2.49*
Obiettivo didattico:
Struttura che controlla il flusso delle istruzioni: La selezione(cond)
Obiettivo del programma:
definire una funzione che restituisca il secondo elemento di una lista

```
;Esempio:
; (SECONDO '(R U S))
;U
```
La funzione riceve come valore attuale la lista (R U S),viene esaminato il primo predicato (null L) che risulta falso perchè la lista non è vuota, viene esaminato il secondo che risulta falso perchè la coda della lista non è vuota (è la lista (U S)), viene, quindi, valutata l'ultima espressione che richiede la testa della coda della lista che, in questo caso, vale U.

```
(defun SECONDO (L)
 (cond
   ((null L) NIL)
   ((null (cdr L)) NIL)
   (T (car (cdr L)))
 )
)
```

Oppure volendo segnalare il caso della lista contenente un solo elemento:

```
(defun SECONDO (L)
 (cond
   ((null L) 'lista_vuota)
   ((null (cdr L)) 'lista_composta_da_un_solo_elemento)
    (t (car (cdr L)))
 )
)
```

ITIS-LS "Francesco Giordani" Caserta
Anno scolastico: *2019/2020*
Classe 3^ sez.B spec. Informatica e telecomunicazioni
Data:
Numero progressivo dell'esercizio: *es8*
Versione: *1.0*
Programmatore/i:
Sistema Operativo: *Windows 10*
Compilatore/Interprete: *GNU CLisp 2.49*
Obiettivo didattico:
Struttura che controlla il flusso delle istruzioni: La selezione(cond)
Obiettivo del programma:
definire una funzione che scambia i primi due elementi di una lista

```
;*************************************************************
;*  scambio.lsp                                              *
;*  19/01/2003          GNU CLisp 2.49                       *
;*  restituisce la lista ottenuta scambiando i primi due elementi *
;*************************************************************
(defun scambio (L)
 (cond
   ((null L) NIL)
   (T (cons (car (cdr L)) (cons (car L) (cdr (cdr L)))
     )
   )
 )
)
```

ITIS-LS "Francesco Giordani" Caserta
Anno scolastico: *2019/2020*
Classe 3^ sez.B spec. Informatica e telecomunicazioni
Data:
Numero progressivo dell'esercizio: *es9*
Versione: *1.0*
Programmatore/i:
Sistema Operativo: *Windows 10*
Compilatore/Interprete: *GNU CLisp 2.49*
Obiettivo didattico:
If (cond)
Obiettivo del programma:
definire una funzione che restituisce i primi due elementi di una lista

```
(defun primidue (L)

 (cond

   ((null L) NIL)

   ((null (cdr L)) NIL)

   (T (cons (car L) (cons (car (cdr L)) NIL)

     )

   )

 )

)
```

Nota: restiuisce la lista che è formata dal primo elemento della lista(car) e dal primo elemento della restante parte della lista (cdr)

ITIS-LS "Francesco Giordani" Caserta
Anno scolastico: *2019/2020*
Classe 3^ sez.B spec. Informatica e telecomunicazioni
Data:
Numero progressivo dell'esercizio: *es10*
Versione: *1.0*
Programmatore/i:
Sistema Operativo: *Windows 10*
Compilatore/Interprete: *GNU CLisp 2.49*
Obiettivo didattico:
Append
Obiettivo del programma:
definire una funzione che inserisce un elemento in coda ad una lista.

```
;(inserisci_in_coda 'z '(a b c))

(defun inserisci_in_coda (X L)

  (append L (list X))

)
```

ITIS-LS "Francesco Giordani" Caserta
Anno scolastico: *2019/2020*
Classe 3^ sez.B spec. Informatica e telecomunicazioni
Data:
Numero progressivo dell'esercizio: *es11*
Versione: *1.0*
Programmatore/i:
Sistema Operativo: *Windows 10*
Compilatore/Interprete: *GNU CLisp 2.49*
Obiettivo didattico:
If (cond) Append
Obiettivo del programma:
restituisce la lista ottenuta leggendo la lista da dx a sx

```
(defun rovesc (L)
 (cond
   ((null L) NIL)
   (T (append (rovesc (cdr L))  (cons (car L) NIL)      )

   )

 )

)
```

ITIS-LS "Francesco Giordani" Caserta
Anno scolastico: *2019/2020*
Classe 3^ sez.B spec. Informatica e telecomunicazioni
Data:
Numero progressivo dell'esercizio: *es12*
Versione: *1.0*
Programmatore/i:
Sistema Operativo: *Windows 10*
Compilatore/Interprete: *GNU CLisp 2.49*
Obiettivo didattico:
If (cond)
Obiettivo del programma:
definire una funzione che inserisce un elemento al secondo posto di una lista.

```
(defun inserisci_al_sec (X L)

  (cond

    ((null L) nil)

    (t (cons (car L) (cons X (cdr L))))

 )

)
```

ITIS-LS "Francesco Giordani" Caserta
Anno scolastico: *2019/2020*
Classe 3^ sez.B spec. Informatica e telecomunicazioni
Data:
Numero progressivo dell'esercizio: *es13*
Versione: *1.0*
Programmatore/i:
Sistema Operativo: *Windows 10*
Compilatore/Interprete: *GNU CLisp 2.49*
Obiettivo didattico:
Equal rem
Obiettivo del programma:
definire una funzione che controlla la parità di un numero

```
(defun PARI (N)

  (equal (rem N 2) 0)

)
```

ITIS-LS "Francesco Giordani" Caserta
Anno scolastico: *2019/2020*
Classe 3^ sez.B spec. Informatica e telecomunicazioni
Data:
Numero progressivo dell'esercizio: *es14*
Versione: *1.0*
Programmatore/i:
Sistema Operativo: *Windows 10*
Compilatore/Interprete: *GNU CLisp 2.49*
Obiettivo didattico:
If (cond) equal
Obiettivo del programma:
definire una funzione che ricerca un elemento dato in una lista.

```
(defun ricercaX (X L)

 (cond

   ((null L) nil)

   ((equal (car L) X) T)

   (t (ricercaX X (cdr L)))

  )

)

;esempio

;? (ricercaX 2 '(a 2 b))

;T

;?(ricercaX 'A '(S A W))

;T

;?(ricercaX 3 '(2 5 A))

;NIL
```

ITIS-LS "Francesco Giordani" Caserta
Anno scolastico: *2019/2020*
Classe 3^ sez.B spec. Informatica e telecomunicazioni
Data:
Numero progressivo dell'esercizio: *es15*
Versione: *1.0*
Programmatore/i:
Sistema Operativo: *Windows 10*
Compilatore/Interprete: *GNU CLisp 2.49*
Obiettivo didattico:
If (cond)
tecnica del Massimo ricorsivo
Obiettivo del programma:
Ricerca in una lista dell' elemento di valore maggiore

```
(defun massimo (x)
        (cond
            ((null x)                   nil)
            ((null (cdr x))             (car x))
            ((> (car x) (massimo (cdr x))) (car x))
            (t              (massimo (cdr x)))
        )
)
```

ITIS-LS "Francesco Giordani" Caserta
Anno scolastico: *2019/2020*
Classe 3^ sez.B spec. Informatica e telecomunicazioni
Data:
Numero progressivo dell'esercizio: *es16*
Versione: *1.0*
Programmatore/i:
Sistema Operativo: *Windows 10*
Compilatore/Interprete: *GNU CLisp 2.49*
Obiettivo didattico:
If
Obiettivo del programma:
Definire una funzione che, data la base e l'esponente, restituisce la potenza corrispondente

```
(defun pot (b e)
        (if (zerop e) 1
           (* b (pot b (- e 1)))
        )
)
```

ITIS-LS "Francesco Giordani" Caserta
Anno scolastico: *2019/2020*
Classe 3^ sez.B spec. Informatica e telecomunicazioni
Data:
Numero progressivo dell'esercizio: *es17*
Versione: *1.0*
Programmatore/i:
Sistema Operativo: *Windows 10*
Compilatore/Interprete: *GNU CLisp 2.49*
Obiettivo didattico:
If (cond)
Obiettivo del programma:
definire una funzione che determina il fattoriale di un numero

```
(defun fatt (n)

    (if (zerop n) 1

        (* n (fatt (1- n)))))
```

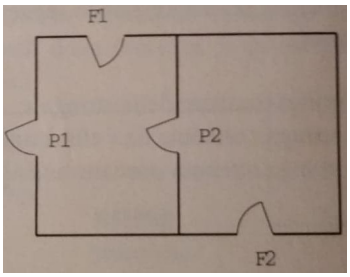
ITIS-LS "Francesco Giordani" Caserta
Anno scolastico: *2019/2020*
Classe 3^ sez.B spec. Informatica e telecomunicazioni
Data:
Numero progressivo dell'esercizio: *es18*
Versione: *1.0*
Programmatore/i:
Sistema Operativo: *Windows 10*
Compilatore/Interprete: *GNU CLisp 2.49*
Obiettivo didattico:
If
Obiettivo del programma:
definire una funzione che restituisce l'ennesimo numero di fibonacci.

```
(defun fib (x)
    (if (< x 2)
        x
        (+ (fib (1- x)) (fib (- x 2)))))
```

ITIS-LS "Francesco Giordani" Caserta
Anno scolastico: *2019/2020*
Classe 3^ sez.B spec. Informatica e telecomunicazioni
Data:
Numero progressivo dell'esercizio: *es19*
Versione: *1.0*
Programmatore/i:
Sistema Operativo: *Windows 10*
Compilatore/Interprete: *GNU CLisp 2.49*
Obiettivo didattico:
Notazione prefissa
Obiettivo del programma:
consumo carburante

```
(define consumo (CONTAATT CONTAPREC SPESA COSTOLITRO)
          (/ ( * (- CONTAATT CONTAPREC) COSTOLITRO) SPESA)
)
;valutare (consumo 7322 6520 45000 1750) ->31
```

ITIS-LS "Francesco Giordani" Caserta
Anno scolastico: *2019/2020*
Classe 3^ sez.B spec. Informatica e telecomunicazioni
Data:
Numero progressivo dell'esercizio: *es20*
Versione: *1.0*
Programmatore/i:
Sistema Operativo: *Windows 10*
Compilatore/Interprete: *GNU CLisp 2.49*
Obiettivo didattico:
Connettivi logici
Obiettivo del programma:
Data la piantina in figura dire se, conoscendo lo stato delle sue porte e finestre, c'è o meno corrente d'aria nella casa.



Detta Pi/Fi una generic porta o finestra, conveniamo di adottare il seguente modello per rappresentare lo stato del sistema:
Pi/Fi=false porta o finestra chiusa
Pi/Fi=true porta o finestra aperta
Dalla piantina possiamo dedurre che esiste corrente d'aria in tre casi:
1) P1 e F1 aperte;
2) P2, F1 e F2 aperte
3)P1, P2, e F2 aperte
quindi c'è corrente se è soddisfatta la seguente proposizione logica:

P1 and F1 or P2 and F1 and F2 or P1 and P2 and F2

```
(defun corrente (L)
 (cond
   ((null L) NIL)
   (T (or
        (and (car L) (car(cdr(cdr L)))
        )
        (and (car(cdr L)) (car(cdr(cdr L))) (car(cdr(cdr(cdr L))))
        )
        (and (car L) (car(cdr L)) (car(cdr(cdr(cdr L))))
        )
      )
   )
 )
)
;(corrente T NIL NIL NIL)  -> NIL
(defun corrent2 (P1 P2 F1 F2)
   (or
     (and P1 F1)
     (and P2 F1 F2)
     (and P1 P2 F2)
   )
)
```