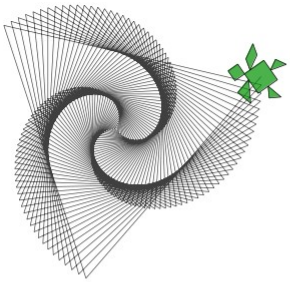
ITIS-LS "Francesco Giordani" Caserta

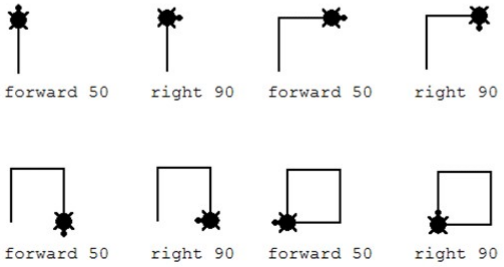
prof. Ennio Ranucci

a.s. 2019-2020

Semplici procedure LOGO

Esercitazioni in ambiente Logoit





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: *Logoit versione 6.4*
Obiettivo didattico:
Conoscere l'ambiente di programmazione
Obiettivo del programma:
disegnare una linea tratteggiata

Comandi logo: Avanti(a), indietro(i), destra(d), sinistra(s), puliscischermo(ps), puliscitesto(pt), tana, sulapenna(su), pennagiu(giu), nasconditartaruga(nt), mostratartaruga(mt)

ps d 90 a 50 su a 50 giu a 50 su a 50 giu a 50 d 90 nt

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: *Logoit versione 6.4*
Obiettivo didattico:
cicli e procedure. Divide et impera, top-down
Obiettivo del programma:
disegnare una casa

Disegno il quadrate: ps a 50 d 90 a 50 d 90 a 50 d 90 a 50 d 90 nt

I comandi a 50 d 90 vengono ripetuti 4 volte.
I cicli "comprimono la scrittura delle istruzioni", le istruzioni ripetute vengono scritte una sola volta.

Disegno il quadrate con una sola istruzione: ripeti 4[a 50 d 90]

Divide et impera: dividere un problema in parti più semplici(sottoproblemi) aiuta a trovare la soluzione dei sottoproblemi e poi del problema generale.
Top-down: dovendo dividere il problema in sottoproblemi si può decider di farlo seguendo un percorso che dal generale "passa" al particolare in modo sempre più dettagliato (per raffinamenti successivi).

Dovendo disegnare una casa possiamo "pensarla" come unione di quadrati, triangoli e alter figure geometriche, quindi, possiamo divedere il problema "disegna una casa" in sottoproblemi che disegnano le figure geometriche necessarie.

Per disegnare un quadrate possiamo scrivere la procedura "quadrato":
Scrivere sulla riga bianca della finestra comandi **EDITA "quadrato** e premere Esegui

per quadrato

a 50 d 90 a 50 d 90 a 50 d 90 a 50 d 90

Fine

Con un ciclo possiamo riscrivere la procedura quadtrato in questo modo:

per quadrato

ripeti 4[a 50 d 90]

Fine

Il quadrate potrebbe essere la facciata della casa.

Il tetto potrebbe essere un triangolo:

Per triangolo
ripeti 3[a 50 d 120]
fine

La porta un rettangolo

Per rettangolo
ripeti 2[a 30 d 90 a 20 d 90]
fine

Abbiamo risolto i sottoproblemi, possiamo scrivere la soluzione generale del problema:

```
Per casa
ps
giu
quadrato        ;parete
su a 50 giu      ;posizione per tetto
d 30 giu
triangolo       ;tetto
su s 30
su i 50 giu
su d 90 a 15 s 90
giu
rettangolo
nt
Fine
```

A questo punto si può disegnare un villaggio come insieme di case.

Per casa
pareti
tetto
Fine

Per pareti
ripeti 4 [a 40 d 90]
a 40
Fine

Per tetto
d 30 ripeti 3 [a 40 d 120]
Fine

```
Per villaggio
 giu casa su tana s 90 a 100 d 90  giu casa
su tana s 90 i 100 d 90 giu casa
Fine
```

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: *Logoit versione 6.4*
Obiettivo didattico:
procedure con parametri
Obiettivo del programma:
disegnare un quadrato con la lunghezza del lato dato in ingresso alla procedura

```
Per quadrato :lato
ps
ripeti 4[a :lato d 90]
nt
fine
```

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: *Logoit versione 6.4*
Obiettivo didattico:
procedure con parametri
Obiettivo del programma:
disegnare un triangolo equilatero con la lunghezza del lato dato in ingresso alla procedura

```
Per TriangoloEquilatero :LATO
Tana
ps
d 30
ripeti 3 [a :LATO d 120]
s 30
nt
fine
```

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: *Logoit versione 6.4*
Obiettivo didattico:
procedure con parametri
Obiettivo del programma:
disegnare un poligono regolare con la lunghezza del lato ed il numero di lati dati in ingresso alla procedura

```
Per poligono :numLati :lungLato
tana
puliscischermo
ripeti :numLati[a :lunglato s 360/:numLati]
nt
Fine
```

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: *Logoit versione 6.4*
Obiettivo didattico:
usiamo i colori
Obiettivo del programma:
disegnare un quadrato rosso

ASCOLPENNA [rosso verde blu] ***ASCP*** [rosso verde blu]

rosso= numero	*il numero, compreso fra 0 e 255, una tonalità di rosso*
Verde= numero	*il numero, compreso fra 0 e 255, una tonalità di verde*
blu= numero	*il numero, compreso fra 0 e 255, una tonalità di blu*

```
ascp [255 0 0]
ripeti 4[a 100 d 90
```

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: *Logoit versione 6.4*
Obiettivo didattico:
usiamo i colori
Obiettivo del programma:
disegnare un quadrato rosso

ASCOLRIEMPI [rosso verde blu] ***ASCR*** [rosso verde blu]

Imposta il colore del riempimento delle figure.

Il colore della superficie ha effetto con i comandi RIEMPI e BLOCCODIS, ma prima dobbiamo portare la tartaruga in un punto qualsiasi interno alla figura da riempire. Per evitare che la tartaruga entri lasciando traccia è necessario il comando PENNASU (SU). L'istruzione ***GIU*** *ripristina la traccia.*

Esempio: ps giu
ascp [0 0 0] assegna il colore nero alla penna
ascr [255 128 0] assegna il colore di riempimento
ripeti 4[a 100 d 90] disegna un quadrato
su d 45 a 5 la tartaruga entra nel quadrato senza lasciare traccia
riempi il quadrato viene riempito del colore arancione
giu la penna è abbassata

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: *Logoit versione 6.4*
Obiettivo didattico:
procedure con parametri
Obiettivo del programma:
disegnare "lineatratteggiata" con parametri (numLine, lungLine)

```
Per lineatratteggiata :numLine :lungLine
tana
ps
ripeti :numLine[a :lungLine su a :lungLine giu]
nt
fine
```

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: *Logoit versione 6.4*
Obiettivo didattico:
assegnazione e numeri casuali
Obiettivo del programma:
disegnare un quadrate di lato random

```
Per quadratoRandom
as "lato acaso 100
ripeti 4[a :lato d 90]
nt
Fine
```

La lunghezza massima del lato può essere parametrizzata:

```
Per quadratoRandom2 :maxLung
as "lato acaso :maxLung
ripeti 4[a :lato d 90]
nt
Fine
```

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: *Logoit versione 6.4*
Obiettivo didattico:
stampa e struttura di controllo del flusso delle istruzioni: Selezione (se-if)
Obiettivo del programma:
Stampare se il numero dato in ingresso alla procedura eè dispari o pari

```
per pariDispari :n

 se (resto  :n 2) = 0 [stampa [pari]]

 se (resto  :n 2) = 1 [stampa [dispari]]

fine
```

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: *Logoit versione 6.4*
Obiettivo didattico:
stampa e strutture di controllo del flusso delle istruzioni: Selezione (se-if) e Iterazione (ripeti)
Obiettivo del programma:
Stampare i divisori del numero dato in ingresso alla procedura

```
Per divisori :n
 ripeti :n [se (resto  :n iterazioni) = 0 [stampa iterazioni]]
fine
```

iterazioni assume I valori 1,2,3…n (è il contatore delle ripetizioni)

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: *Logoit versione 6.4*
Obiettivo didattico:
La ricorsione
Obiettivo del programma:
Stampare il fattoriale del numero dato in ingresso alla funzione

Il principio d'induzione asserisce che se P è una proprietà che vale per il numero 0, e se $P(n) \Rightarrow P(n+1)$ per ogni n, allora $P(n)$ vale per ogni n.

```
Per fattoriale :n
se :n = 1 [riporta :n]
riporta :n*(fattoriale :n-1)
fine
```

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: *Logoit versione 6.4*
Obiettivo didattico:
La ricorsione
Obiettivo del programma:
Stampare la Potenza naturale (la base e l'esponente sono dati in ingresso)

```
Per potenzaNaturale :b :e
se :e = 0 [riporta 1]
riporta :b*(potenzaNaturale :b :e-1)
fine
```

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: *Logoit versione 6.4*
Obiettivo didattico:
La Potenza natural senza ricorsione
Obiettivo del programma:
Stampare la Potenza naturale (la base e l'esponente sono dati in ingresso)

```
per potenzanaturale :x :n
se tuttiveri? :x=0 :n=0 [stampa [0 alla 0 non definito] stop]
se :n<0 [stampa [esponente negativo] stop]
assegna "p 1
ripeti :n [assegna "p :p * :x]
stampa :p
Fine
```

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: *Logoit versione 6.4*
Obiettivo didattico:
La ricorsione
Obiettivo del programma:
Stampare la somma delle cifre di un numero dato in ingresso

Si può ottenere il primo o 1' ultimo elemento di una lista, con le funzioni
PRI primo
ULT ultimo
Da una lista se ne può ottenere un'altra ricavata dalla prima eliminando il primo o 1' ultimo elemento, con i comandi
MENPRI mp
MENULT mu

```
per sommacifre :numero :somma
se vuoto? :numero [stampa :somma stop]
sommacifre mp :numero :somma + primo :numero
fine
```

oppure:

```
Per sommacifre :numero
 se vuoto? :numero [riporta  0]
 riporta  (sommacifre mp :numero)+ primo :numero
fine
```

traccia funzionale:
365->sommacifre 65 +3->sommacifre 5 9 + -> sommacifre vuoto +14 ->14

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: *Logoit versione 6.4*
Obiettivo didattico:
La ricorsione
Obiettivo del programma:
Disegnare una spirale e una corniceSpirale

```
Per spirale :lato
a :lato d 90
assegna "lato :lato+ 2
se :lato >200 [stop]
spirale :lato
Fine

Per cornicespirale1 :passo
a :passo d 90
se :passo > 100 [stop]
cornicespirale1 :passo + 2
fine

per cornicespirale2 :passo :incremento
a :passo d 90
se :passo > 100 [stop]
cornicespirale2 :passo + :incremento :incremento
fine

per cornicespirale3 :passo :incremento :gradi
a :passo d :gradi
se :passo > 100 [stop]
cornicespirale3 :passo + :incremento :incremento :gradi
fine
```

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: *Logoit versione 6.4*
Obiettivo didattico:
Cicli annidati
Obiettivo del programma:
Stampare la tabellina pitagorica

```
Per tabellina
ripeti 10 [
  assegna "r iterazioni
  ripeti 10 [scrivi formato :r*iterazioni 4 0 ]
stampa " ]
fine
```

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: *Logoit versione 6.4*
Obiettivo didattico:
Suoni
Obiettivo del programma:
Riprodurre un file .wav

```
Per suono1
suonawave "musica1.wav 1      ; suono di avvio
fine
nascondi "suono1
```

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: *Logoit versione 6.4*
Obiettivo didattico:
Coordinate cartesiane e polari
Obiettivo del programma:
Disegnare una linea dalla tana fino al punto di coordinate cartesiane 80 60

```
ps tana VAXY 80 60 nt
```

Le coordinate polari sono un sistema di coordinate del piano determinato da un punto O, detto polo, una semiretta x avente origine in O, detta asse polare, e un segmento unitario u.

I comandi LOGO del sistema polare sono: ASDIR <numero gradi>e DIREZIONE
Con ASDIR la tartaruga ruota verso destra del numero di gradi indicate dall'argomento, se questo è un numero positivo, mentre ruota verso sinistra se il numero è negativo.
Con il comando DIR il LOGO risponde indicando la direzione angolare sulla quale la tartaruga sta in quel momento disposta.

ps tana asdir 45 a 300

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: *Logoit versione 6.4*
Obiettivo didattico:
All'interno di una procedura richiamare altre procedure
Obiettivo del programma:
Disegnare un fiore

per fiore
ascolpenna [200 200 000]
ripeti 12 [petalo d 30]
fine
Per petalo
ripeti 2 [ripeti 60 [a 1 d 1] d 120]
fine

Per pianta
ps
ascolpenna [000 128 000]
a 30 petalo
s 60 petalo d 60
a 120
fiore
fine

LOGO

Funzioni primitive

operazioni aritmetiche

SUM a b, DIFFERENCE a b, PRODUCT a b, QUOTIENT a b, REMAINDER a b, RANDOM n, ROUND n, SQRT n , POWER n a, EXP a

operazioni di selezione

FIRST ***PRI***	*FIRST SI USA NELLA VERSIONE INGLESE DEL LOGO* *VERSIONE ITALIANA*
sintassi	*FIRST oggetto*
uscita	*restituiesce come valore il primo termine del suo argomento*
esempi Logo	*?PRINT FIRST [A B C]* *A*

BUTFIRSTM MENPRI	*VERSIONE INGLESE DEL LOGO- ABBR.* ***BF*** *VERSIONE ITALIANA*
sintassi	*BF oggetto*
uscita	*restituiesce come valore il suo argomento senza il primo termine*
esempi Logo	*?PRINT BUTFIRST [A B C]* *B C*

operazioni di costruzione

FPUT ***INPRI***	*VERSIONE INGLESE DEL LOGO* *VERSIONE ITALIANA*
sintassi	*FPUT oggetto lista*
uscita	*restituiesce come valore una lista il cui primo termine è il primo argomento ed i seguenti sono quelli del secondo argomento*
esempi	*?PR FPUT "A [B C]*

Logo	*A B C*

LPUT ***INULT***	*VERSIONE INGLESE DEL LOGO* *VERSIONE ITALIANA*
sintassi	*LPUT oggetto lista*
uscita	*inserisce oggetto nella lista collocandolo all'ultimo posto*
esempi Logo	*?PR LPUT "C [A B]* *A B C*

LAST ***ULT***	*VERSIONE INGLESE DEL LOGO* *VERSIONE ITALIANA*
sintassi	*LAST oggetto*
uscita	*restituiesce l'ultimo elemento di oggetto*
esempi Logo	*? PRINT LAST [A B C]* *C*

ITEM	*VERSIONE INGLESE DEL LOGO* *VERSIONE ITALIANA*
sintassi	*ITEM n oggetto*
uscita	*estrae l'elemento n-esimo di oggetto*
esempi Logo	*?PR ITEM 3[IO TU EGLI ELLA ESSO]* *EGLI*

SENTENCE ***FRASE***	*VERSIONE INGLESE DEL LOGO - ABBR.* ***SE*** *VERSIONE ITALIANA*
sintassi	*SE oggetto1 oggetto2*
uscita	*restituisce la lista che contiene oggetto1 e oggetto2. Se sono già liste, viene*

	formata una nuova lista di tutti gli oggetti contenuti in ognuna di esse
esempi Logo	*?PR SENTENCE [A B] [C D]* *A B C D*

predicati

WORDP ***PAROLA?***	*VERSIONE INGLESE DEL LOGO* *VERSIONE ITALIANA*
sintassi	*WORDP oggetto*
uscita	*restituisce true se oggetto è una parola, altrimenti false*
esempi Logo	*? PR NOT WORDP [A] LISTP [SO]* *FALSE* *?PR OR WORDP [A] LISTP[SO]* *TRUE*

NUMBERP ***NUMERO?***	*VERSIONE INGLESE DEL LOGO* *VERSIONE ITALIANA*
sintassi	*NUMBERP oggetto*
uscita	*restituiesce come valore True se e solo se il suo argomento è un numero, in caso contrario restituisce False*
esempi Logo	*?PR AND NUMBERP "123 LISTP [A B C]* *TRUE*

LISTP ***LISTA?***	*VERSIONE INGLESE DEL LOGO* *VERSIONE ITALIANA*
sintassi	*LISTP oggetto*
uscita	*restituiesce come valore True se e solo se il suo argomento è una lista, in caso contrario restituisce False*

esempi Logo	*TO SECONDO :LISTA* *IF LISTP :LISTA [PR FIRST BF :LISTA] [PR [LISTAVUOTA]]* *END*

EQUALP ***UGUALE***	*VERSIONE INGLESE DEL LOGO* *VERSIONE ITALIANA*
sintassi	*EQUAL oggetto1 oggetto2*
uscita	*restituiesce come valore True se e solo se i suoi due argomenti sono uguali, in caso contrario restituisce False*

EMPTYP ***VUOTO?***	*VERSIONE INGLESE DEL LOGO* *VERSIONE ITALIANA*
sintassi	*EMPTYP oggetto*
uscita	*restituiesce come valore True se e solo se il suo argomento è la lista vuota, in caso contrario restituisce Flase*
esempi Logo	*?PR EMPTYP "CUORE* *FALSE* *?PR EMPTYP [A D]* *FALSE* *?PR EMPTYP BF BF BF "MAR* *TRUE* *?PR EMPTYP []* *TRUE*

operazioni di ricerca

MEMBERP	*VERSIONE INGLESE DEL LOGO* *VERSIONE ITALIANA*
sintassi	*MEMBERP oggetto lista-di-oggetti)*

uscita	*prova la persenza o l'assenza di oggetto all'interno dalla lista-di-oggetti*
esempi Logo	*? PRINT NOT MEMBERP "A [A B C]* *FALSE*

GRAFICA DI TARTARUGA

FORWARD ***AVANTI***	*VERSIONE INGLESE DEL LOGO - ABBR.* ***FD*** *VERSIONE ITALIANA*
sintassi	*FD distanza*
uscita	*muove la tartaruga* distanza *passi in avanti*
esempi Logo	*FD 40*

BACK ***INDIETRO***	*VERSIONE INGLESE DEL LOGO - ABBR.* ***BK*** *VERSIONE ITALIANA*
sintassi	*BK distanza*
uscita	*muove la tartaruga* distanza *passi indietro*
esempi Logo	*BK 60*

RIGHT ***DESTRA***	*VERSIONE INGLESE DEL LOGO - ABBR.* ***RT*** *VERSIONE ITALIANA*
sintassi	*RT gradi*
uscita	*gira la tartaruga di* gradi *a destra*
esempi Logo	*RT 90*

LEFT ***SINISTRA***	*VERSIONE INGLESE DEL LOGO - ABBR.* ***LT*** *VERSIONE ITALIANA*
sintassi	*LT gradi*
uscita	*gira la tartaruga di* gradi *a sinistra*

esempi Logo	*LT 60*

PENUP ***PENNASU***	*VERSIONE INGLESE DEL LOGO - ABBR.* ***PU*** *VERSIONE ITALIANA*
sintassi	*PU*
uscita	*solleva dalla superficie del disegno la punta scrivente della tartaruga*
esempi Logo	

PENDOWNPENNAGIU	*VERSIONE INGLESE DEL LOGO - ABBR.* ***PD*** *VERSIONE ITALIANA*
sintassi	*PD*
uscita	*pone la penna della tartaruga a contatto con la superficie del disegno*
esempi Logo	

HOME: muove la tartaruga nella posizione 0,0 e pone la direzione pari a 0;

HIDETURTLE, HT: rende invisibile la tartaruga;

CLEAN: cancella lo schermo grafico;

CLEARSCREEN, CS : cancella lo schermo, muove la tartaruga nella posizione 0,0 e pone la direzione pari a 0;

SETPC numerocolore: assegna alla penna il colore;

VARIABILI

MAKE nome oggetto: assegna il nome *all'*oggetto;

STRUTTURE DI CONTROLLO

IF predicato lista1 (lista2): se il predicato fornisce la risposta true viene eseguita la lista1, altrimenti la lista2;

OUTPUT, OP oggetto: restituisce il controllo dell'esecuzione insieme al valore oggetto alla routini di chiamata;

REPEAT n listaistruzioni: esegue un numero di volte pari ad n la listaistruzioni;

COMANDI DI SISTEMA

.DOS: uscita al sistema operativo;

TYPE "oggetto: stampa l'oggetto senza ritorno a capo;

esempio

?type [A B]

A B?

PRINT, PR "oggetto: stampa l'oggetto seguito dal ritorno a capo;

esempio

?print [A B]

A B

?

SHOW "oggetto: stampa l'oggetto con le parentesi seguito dal ritorno a capo;

esempio

?show[A B]

?[A B]

?

POTS: stampa i nomi delle funzioni definite in memoria centrale;

POPS: stampa il corpo delle funzioni definite in memoria centrale;

PO "nome: stampa il corpo della funzione nome;

PONS : stampa nomi e valori delle variabili;

LOAD "nomearchivio: carica in memoria centrale nomearchivio;

SAVE "nomearchivio: registra il contenuto della memoria centrale in un file nomearchivio;

EDIT, ED "nome: attiva l'editor per modificare la procedura nome.

TO nome: inizia la definizione di una procedura nome;

END: termina la definizione della procedura iniziata da TO.

ESEMPI DI PROCEDURE LOGO

definizione procedura	***esecuzione***
TO PRI *PRINT FIRST [A B C]* *END*	*? PRI* *A*

definizione procedura	***esecuzione***
TO MENPRI *PRINT BUTFIRST [A B C]* *END*	*? MENPRI* *B C*

definizione procedura	***esecuzione***
TO ESTREMI :LISTA *PRINT FIRST :LISTA* *PRINT LAST :LISTA* *END*	*? ESTREMI [A B C]* *A* *B*

definizione procedura	***esecuzione***
TO TOTALE :NUM1 :NUM2 *PRINT (SUM :NUM1 :NUM2)* *END*	*? TOTALE 2 3* *5*

definizione procedura	***esecuzione***
TO SECONDO :LISTA *OP FIRST BUTFIRST :LISTA* *END*	*?PR SECONDO [A B C]* *B*

Problema: definire una funzione che, dato un numero restituisca il suo cubo.

```
to CUBO :N
  stampa  :N * :N * :N
end
```

Codifica logo in ambiente: ***MSWLogo Version 6.5b for Micro-Soft Windows 95/98/NT/2000/XP***

Problema: Ddefinire una funzione che restituisca il secondo elemento di una lista

Problema: Descrivere il risultato del seguente programma LOGO:

```
repeat 36[FD 80 bk 80 RT 10]
```

Problema: Descrivere il risultato del seguente programma LOGO:

```
repeat 100[FD Random 40 RT 90]
```

Problema: Definire una funzione che restituisce l'ennesimo elemento di una lista.

```
TO N_ESIMO_ELEM :QUALE  :LISTA
 OP ITEM :QUALE :LISTA
END
```

Problema: Disegnare un triangolo con la tartaruga

```
TO TRIANGOLO
CS                    (cancella lo schermo grafico, CT cancella il testo)
REPEAT 3 [FD 50 RT 120]
END
```

Problema: Dato il lato disegnare il triangolo con la tartaruga

```
TO TRIANGOLO :LATO
CS
REPEAT 3 [FD :LATO RT 120]
END
```

Problema: Disegnare una spirale quadrata con la tartaruga.

```
TO SPIRALEQUADRATA :LATO
FD :LATO
RT 90
SPIRALEQUADRATA :LATO + 3
END
```

Problema: Dato il lato e il numero di lati, disegnare una spirale quadrata con la tartaruga.

```
TO SPIQUA :LATO :RIC
IF :RIC = 0 [STOP]
```

FD :LATO

RT 90

SPIQUA :LATO + 2 :RIC - 1

END

Problema: Disegnare la curva di Vonkoch con la tartaruga.

TO VONKOCH :X :LIV

IF :LIV = 0 [FD :X]

IF :LIV > 0 [VONKOCH INT QUOTIENT :X 3 :LIV - 1]

IF :LIV > 0 [LT 60]

IF :LIV > 0 [VONKOCH INT QUOTIENT :X 3 :LIV - 1]

IF :LIV > 0 [RT 120]

IF :LIV > 0 [VONKOCH INT QUOTIENT :X 3 :LIV - 1]

IF :LIV > 0 [LT 60]

IF :LIV > 0 [VONKOCH INT QUOTIENT :X 3 :LIV - 1]

END

Problema: *Tre missionari e tre cannibali cercano di attraversare un fiume dalla riva sinistra alla riva destra. C'è una barca che ha due posti e che può navigare con una qualsiasi combinazione di missionari e cannibali di una o due persone. In qualunque momento, se i missionari su una riva del fiume sono in minoranza rispetto ai cannibali, questi ultimi cederanno alle loro tendenze antropofaghe. Quando la barca è ormeggiata la si considera appartenente alla riva.*
Trovare la più semplice successione di traversate che consenta a tutti i missionari e i cannibali di attraversare il fiume sani e salvi.

- *descriviamo in modo simbolico gli stati delle rive del fiume in questo modo:*
  per es. M M C C / BARCA M C significa che sulla riva sx ci sono 2 missionari e due cannibali e sulla riva dx la barca un missionario e un cannibale.

- *rappresentiamo gli stati e gli oggetti come liste e i movimenti da una riva all'altra (mosse) con procedure di modifica di tali liste.*
  es. STATO M M C C / BARCA M C con

  rivasx [M M C C]

  rivadx [BARCA M C]

  es. MOSSA con

  una lista di oggetti da trasferire

  [M C BARCA]

- *i tentativi di soluzione del problema possono essere rappresentati da un ramo di un albero in cui i nodi sono etichettati da stati delle rive e gli archi da mosse. In tale albero una soluzione è rappresentata da un ramo che va dallo stato iniziale allo stato finale cioè:*
  [M M M C C C BARCA] / [ ] → [ ] / [M M M C C C BARCA]

- *la soluzione può essere trovata esplorando uno per uno tutti i rami dell'albero.*
- *se proviamo a risolvere il problema tentando una dopo l'altra sequenze di traversate il nostro comportamento può essere descritto come una ricerca in "profondità" lungo l'albero.*

*La **mossa**, quindi, rappresenta una lista di "oggetti" che denominiamo MOVLIST da trasferire dalla lista RSIN alla RDES o viceversa, una procedura "muovi-da-sinistra-a-destra" denominata MSD, una procedura "muovi-da-destra-a-sinistra" denominata MDS.*

procedura MSD

assegna alla lista RSIN la lista RSIN senza gli elementi di MOVLIST

assegna alla lista RDES la lista RDES con gli elementi di MOVLIST

fine

La barca è sempre inclusa nella MOVLIST.

Sviluppiamo un programma logo che permetta di controllare le potenziali soluzioni del problema dei missionari

"a mano" , usando il computer solo per tenere nota dello stato in cui siamo.

```
(procedura che restituisce il carattere spazio)

TO SPAZIO

OP CHAR 32

END


(procedura che elimina un solo elemento da una lista)

TO SENZA1 :DACANCELLARE :ORIGINALE

IF EMPTYP :ORIGINALE [PRINT (SENTENCE [NON POSSO TOGLIERE] FPUT :DACANCELLARE [DALLA LISTA DATA]) STOP]

IF :DACANCELLARE = FIRST :ORIGINALE [OP BF :ORIGINALE]

OP FPUT FIRST :ORIGINALE SENZA1 :DACANCELLARE BF :ORIGINALE

END

(procedura che elimina una sottolista da una lista)

TO SENZA :DACANCELLARE :ORIGINALE
```

```
IF EMPTYP :DACANCELLARE [OUTPUT :ORIGINALE]
MAKE "ORIGINALE SENZA1 FIRST :DACANCELLARE :ORIGINALE
OUTPUT SENZA BF :DACANCELLARE :ORIGINALE
END
TO MDS :MOVLIST
MAKE "RDES SENZA :MOVLIST :RDES
MAKE "RSIN SENTENCE :RSIN :MOVLIST
END
TO MSD :MOVLIST
MAKE "RSIN SENZA :MOVLIST :RSIN
MAKE "RDES SENTENCE :RDES :MOVLIST
END
(procedura “INnizializza Missionari e Cannibali”  che assegna a RSIN e RDES i valori iniziali)
TO INMC
MAKE "RSIN [M M M C C C BARCA]
MAKE "RDES []
END
(procedura che scrive a video lo stato di RDES)
TO STAMPARDES
TYPE "RDES
TYPE SPAZIO
TYPE "E'
TYPE SPAZIO
PRINT LIST :RDES
END
(procedura che scrive a video lo stato di RSIN)
TO STAMPARSIN
TYPE "RSIN
TYPE SPAZIO
```

TYPE "E'

TYPE SPAZIO

PRINT LIST :RSIN

END

(procedura che scrive a video lo stato corrente)

TO STAMPASTATO

STAMPARSIN

STAMPARDES

END

Utilizzando queste procedure si può risolvere il problema "a mano":

?INMC

?STAMPASTATO

RSIN E' [M M M C C C BARCA]

RDES E' [ ]

?MSD [M C BARCA]

?STAMPASTATO

RSIN E' [M M C C]

RDES E' [M C BARCA]

?MDS [M BARCA]

?STAMPASTATO

RSIN E' [M M C C M BARCA]

RDES E' [C]

?MSD [M C BARCA]

?STAMPASTATO

RSIN E' [M M C]

RDES E' [C M C BARCA]

MISSIONARIO MORTO!

RICOMINCIAMO

?INMC

?...

Possiamo migliorare il programma aggiungendo la procedura "APPLICAMOSSA" che sceglie tra MSD e MDS tenendo conto della posizione della barca.

TO APPLMOSSA :MOVLIST

IF MEMBERP "BARCA :RSIN [MSD :MOVLIST STOP]

IF MEMBERP "BARCA :RDES [MDS :MOVLIST STOP]

PRINT [ERRORE IN APPLICA MOSSE] STOP

END

esempio

?INMC

?STAMPASTATO

RSIN E' [M M M C C C BARCA]

RDES E' [ ]

?APPLICAMOSSA [C BARCA]

?STAMPASTATO

RSIN E' [M M M C C]

RDES E' [C BARCA]

?...

Possiamo migliorare ulteriormente il programma con le seguenti procedure:

TO CHIEDIMOSSA

STAMPASTATO

PR [BATTI UN MOVLIST]

MAKE "RISPOSTA RL

IF MEMBERP "BARCA :RISPOSTA [OP :RISPOSTA]

PRINT [TI SEI SCORDATO LA BARCA RIPROVA]

CHIEDIMOSSA

END

TO FAIMOSSE

APPLICAMOSSA CHIEDIMOSSA

FAIMOSSE

END

TO MEC

INMC

FAIMOSSE

END

esempio

?MEC

RSIN E' [M M M C C C BARCA]

RDES E' [ ]

[BATTI UN MOVLIST]

C C BARCA

RSIN E' [M M M C]

RDES E' [C C BARCA]

[BATTI UN MOVLIST]

...

Se siamo in un vicolo cieco e vogliamo un "back-up" cioè cambiare l'ultima mossa?

TO PROVAMOSSE

MAKE "MESSAGGIO CHIEDIMOSSA

IF :MESSAGGIO = [BACKUP] [STOP] [ESPLORASTATO :RSIN :RDES :MESSAGGIO]

PROVAMOSSE

END

TO ESPLORASTATO :RSIN :RDES :MOVLIST

APPLICAMOSSA :MOVLIST

PROVAMOSSE

END

TO CHIEDIMOSSA

STAMPASTATO

PR [BATTI UN MOVLIST O BACKUP]

MAKE "RISPOSTA RL

IF OR MEMBERP "BARCA :RISPOSTA :RISPOSTA = [BACKUP] [OP :RISPOSTA]

PRINT [TI SEI SCORDATO LA BARCA RIPROVA]

CHIEDIMOSSA

END

TO MEC

INMC

PROVAMOSSE

END

esempio

?MEC

RSIN E' [M M M C C C BARCA]

RDES E' [ ]

[BATTI UN MOVLIST O BACKUP]

C C BARCA

RSIN E' [M M M C]

RDES E' [C C BARCA]

[BATTI UN MOVLIST O BACKUP]

C BARCA

RSIN E' [M M M C C BARCA]

RDES E' [C]

[BATTI UN MOVLIST O BACKUP]

M C BARCA

RSIN E' [M M C]

RDES E' [C M C BARCA]

MISSIONARIO MORTO!

[BATTI UN MOVLIST O BACKUP]

BACKUP

RSIN E' [M M M C C BARCA]

RDES E' [C]

ADESSO SI PUO' PROVARE UNA MOSSA DIVERSA

[BATTI UN MOVLIST O BACKUP]

M BARCA

RSIN E' [M M C C]

RDES E' [C M BARCA]

Possiamo migliorare ancora il programma verificando automaticamente la soluzione del problema:

(la procedura conta il numero di cannibali o missionari presenti su una sponda, cioè un predicato con due argomenti, un oggetto e una lista, e che restituisce il numro di volte che l'oggetto occorre nella lista.)

TO NUMERODI :ELEMENTO :LISTA

IF EMPTYP :LISTA [OP 0]

IF :ELEMENTO = FIRST :LISTA [OP 1 + NUMERODI :ELEMENTO BF :LISTA]

OP NUMERODI :ELEMENTO BF :LISTA

END

(la procedura crea il predicato che verifica se la condizione per il cannibalismo è stata violata)

TO MMANGIATI? :RIVA

OP AND (NUMERODI "C :RIVA) > (NUMERODI "M :RIVA) (NUMERODI "M :RIVA) > 0

END

(la procedura crea un predicato che riporta vero quando il problema è risolto)

TO SUCCESSO?

OP EMPTYP :RSIN

END

TO MISSIONARIMANGIATI?

OP OR MMANGIATI? :RSIN MMANGIATI? :RDES

END

TO ESPLORASTATO :RSIN :RDES :MOVLIST

APPLMOSSA :MOVLIST

IF MISSIONARIMANGIATI? [PR [MISSIONARI MANGIATI MOSSA ANNULLATA] STOP]

IF SUCCESSO? [PRINT "SUCCESSO FINE]

PROVAMOSSE

END

A questo punto il programma sta eseguendo tutti i compiti tranne la scelta delle mosse.

Su quale base segliere le mosse?

Un modo può essere quello di provare tutte le possibili mosse (che sono cinque).

Costruiamo, quindi, una lista delle mosse possibili.

TO INMC

MAKE "RSIN [M M M C C C BARCA]

MAKE "RDES []

MAKE "STATIVISTI [[1 3 3]]

MAKE "MOSSEPOSSIBILI [[C C BARCA] [C BARCA] [M C BARCA] [M M BARCA] [M BARCA]]

END

TO MEC

INMC

PROVATUTTE :MOSSEPOSSIBILI

PR [NON HO TROVATO ALCUNA SOLUZIONE]

END

TO PROVATUTTE :INSIEMEDIMOSSE

IF EMPTYP :INSIEMEDIMOSSE [STOP]

ESPLORASTATO :RSIN :RDES FIRST :INSIEMEDIMOSSE

PROVATUTTE BF :INSIEMEDIMOSSE

END

TO ESPLORASTATO :RSIN :RDES :MOVLIST

IF NOT APPLICABILE? :MOVLIST [STOP]

APPLMOSSA :MOVLIST

PRINT [MOSSA APPLICATA :] PR :MOVLIST

IF MISSIONARIMANGIATI? [PR [MISSIONARI MANGIATI MOSSA ANNULLATA] STOP]

IF STATOVISTOPRIMA? [PR [STATO GIA' VISTO MOSSA ANNULLATA] STOP]

PR "STATONUOVO REPEAT 2 [PR SPAZIO] STAMPASTATO

IF SUCCESSO? [PRINT "SUCCESSO FINE]

REGISTRASTATONUOVO

PAUSE

PROVATUTTE :MOSSEPOSSIBILI

PR "BACKUP

END

(la procedura controlla se è possibile effettuare la mossa, cioè vi devono essere tanti missionari quanti indicati nella mossa e analogamente per i cannibali)

TO APPLICABILE? :MOSSA

OP AND NOT (NUMERODI "M :MOSSA) > (NUMERODI "M LATODA) NOT (NUMERODI "C :MOSSA) > (NUMERODI "C LATODA)

END

(la funzione restituisce la riva dalla quale la barca si muoverà)

TO LATODA

IF MEMBERP "BARCA :RSIN [OP :RSIN]

IF MEMBERP "BARCA :RDES [OP :RDES]

PRINT [ERRORE IN LATODA] STOP

END

(per evitare che l'esecutore entri in loop di stati ripetuti, bisogna registrare quali stati sono già stati visitati per evitare di esplorarli di nuovo. E' sufficiente ricordare lo stato di una riva, perchè lo stato delll'altra riva lo si può ricavare di conseguenza. Quindi possiamo registrare lo stato di una riva come un gruppo di tre numeri: 1 o 0 barca presente o assente, numero di missionari sulla riva sx, numero di cannibali sulla riva sx)

TO STATOVISTOPRIMA?

OP MEMBERP STATOTRIPLA :STATIVISTI

END

TO REGISTRASTATONUOVO

MAKE "STATIVISTI FPUT STATOTRIPLA :STATIVISTI

END

(la procedura costruisce la tripla)

TO STATOTRIPLA

```
OP (SE NUMERODI "BARCA :RSIN NUMERODI "M :RSIN NUMERODI "C :RSIN)

END

Possiamo contare gli stati attreaversati

TO CONTASTATI

MAKE "INCR 1

MAKE "STATIATTRAVERSATI (SUM :STATIATTRAVERSATI :INCR)

PR :STATIATTRAVERSATI

END

PROGRAMMA MEC COMPLETO

TO SENZA1 :DACANCELLARE :ORIGINALE

IF EMPTYP :ORIGINALE [PRINT (SENTENCE [NON POSSO TOGLIERE] FPUT :DACANCELLARE [DALLA LISTA DATA]) STOP]

IF :DACANCELLARE = FIRST :ORIGINALE [OP BF :ORIGINALE]

OP FPUT FIRST :ORIGINALE SENZA1 :DACANCELLARE BF :ORIGINALE

END

TO STATOTRIPLA

OP (SE NUMERODI "BARCA :RSIN NUMERODI "M :RSIN NUMERODI "C :RSIN)

END

TO MMANGIATI? :RIVA

OP AND (NUMERODI "C :RIVA) > (NUMERODI "M :RIVA) (NUMERODI "M :RIVA) > 0

END

TO LATODA

IF MEMBERP "BARCA :RSIN [OP :RSIN]

IF MEMBERP "BARCA :RDES [OP :RDES]

PRINT [ERRORE IN LATODA] STOP

END

TO NUMERODI :ELEMENTO :LISTA

IF EMPTYP :LISTA [OP 0]

IF :ELEMENTO = FIRST :LISTA [OP 1 + NUMERODI :ELEMENTO BF :LISTA]

OP NUMERODI :ELEMENTO BF :LISTA
```

```
END

TO SENZA :DACANCELLARE :ORIGINALE

IF EMPTYP :DACANCELLARE [OUTPUT :ORIGINALE]

MAKE "ORIGINALE SENZA1 FIRST :DACANCELLARE :ORIGINALE

OUTPUT SENZA BF :DACANCELLARE :ORIGINALE

END

TO REGISTRASTATONUOVO

MAKE "STATIVISTI FPUT STATOTRIPLA :STATIVISTI

END

TO SUCCESSO?

OP EMPTYP :RSIN

END

TO STATOVISTOPRIMA?

OP MEMBERP STATOTRIPLA :STATIVISTI

END

TO MISSIONARIMANGIATI?

OP OR MMANGIATI? :RSIN MMANGIATI? :RDES

END

TO APPLICABILE? :MOSSA

OP AND NOT (NUMERODI "M :MOSSA) > (NUMERODI "M LATODA) NOT (NUMERODI "C :MOSSA) > (NUMERODI "C
LATODA)

END

TO MDS :MOVLIST

MAKE "RDES SENZA :MOVLIST :RDES

MAKE "RSIN SENTENCE :RSIN :MOVLIST

END

TO MSD :MOVLIST

MAKE "RSIN SENZA :MOVLIST :RSIN

MAKE "RDES SENTENCE :RDES :MOVLIST
```

```
END

TO APPLMOSSA :MOVLIST

IF MEMBERP "BARCA :RSIN [MSD :MOVLIST STOP]

IF MEMBERP "BARCA :RDES [MDS :MOVLIST STOP]

PRINT [ERRORE IN APPLICA MOSSE] STOP

END

TO STAMPASTATO

STAMPARSIN

STAMPARDES

END

TO ESPLORASTATO :RSIN :RDES :MOVLIST

PR [SONO IN ESPLORASTATO]

PAUSE

IF NOT APPLICABILE? :MOVLIST [STOP]

APPLMOSSA :MOVLIST

PRINT [MOSSA APPLICATA :] PR :MOVLIST

IF MISSIONARIMANGIATI? [PR [MISSIONARI MANGIATI MOSSA ANNULLATA] STOP]

IF STATOVISTOPRIMA? [PR [STATO GIA' VISTO MOSSA ANNULLATA] STOP]

PR "STATONUOVO REPEAT 2 [PR SPAZIO] STAMPASTATO

IF SUCCESSO? [PRINT "SUCCESSO STOP]

REGISTRASTATONUOVO

CONTASTATI

PAUSE

PROVATUTTE :MOSSEPOSSIBILI

PR [BACKUP - - - - - - - -]

END

TO STAMPARSIN

TYPE "RSIN

TYPE SPAZIO
```

```
TYPE "E'
TYPE SPAZIO
PRINT LIST :RSIN
END
TO SPAZIO
OP CHAR 32
END
TO STAMPARDES
TYPE "RDES
TYPE SPAZIO
TYPE "E'
TYPE SPAZIO
PRINT LIST :RDES
END
TO PROVATUTTE :INSIEMEDIMOSSE
PR [SONO IN PROVATUTTE]
IF EMPTYP :INSIEMEDIMOSSE [STOP]
ESPLORASTATO :RSIN :RDES FIRST :INSIEMEDIMOSSE
PROVATUTTE BF :INSIEMEDIMOSSE
END
TO INMC
MAKE "RSIN [M M M C C C BARCA]
MAKE "RDES []
MAKE "STATIVISTI [[1 3 3]]
MAKE "MOSSEPOSSIBILI [[C BARCA] [C C BARCA] [M C BARCA] [M M BARCA] [M BARCA]]
MAKE "STATIATTRAVERSATI 0
END

TO MEC
```

INMC

PROVATUTTE :MOSSEPOSSIBILI

PR [NON HO TROVATO ALCUNA SOLUZIONE]

END

TO CONTASTATI

MAKE "INCR 1

MAKE "STATIATTRAVERSATI (SUM :STATIATTRAVERSATI :INCR)

PR :STATIATTRAVERSATI

END

Problema:Un contadino deve attraversare un fiume portando con sè un lupo, una capra e un cavolo. Egli dispone di una barca che può trasportare uno solo dei tre "passeggeri" oltre se stesso. Trovare una sequenza di azioni per portare il contadino sull'altra sponda con le sue cose tenendo presente che, in assenza del contadino, il lupo mangerebbe la capra e la capra mangerebbe il cavolo.

LCC1 (prima versione)

TO CHIEDIMOSSA

PR [BATTI MOSSA]

MAKE "MOSSA RL

OP :MOSSA

END

TO RIPORTA_INDIETRO

MAKE "ELEMENTO [CAPRA]

MAKE "RSIN SE :RSIN :ELEMENTO

MAKE "RDES SENZA :ELEMENTO :RDES

END

(la procedura crea un predicato che verifica se la situazione sulla riva dx è legale, cioè il lupo non mangia la capra e la capra non mangia il cavolo)

TO SIT_LEGALE_DX? :RIVA

IF AND (AND (MEMBERP "LUPO :RIVA) (MEMBERP "CAPRA :RIVA)) (MEMBERP "CAVOLO :RIVA) [OP LISTP :RIVA]

IF AND (MEMBERP "LUPO :RIVA) (MEMBERP "CAVOLO :RIVA) [OP MEMBERP "LUPO :RIVA]

IF AND (MEMBERP "CAVOLO :RIVA) (MEMBERP "CAPRA :RIVA) [OP NOT LISTP :RIVA]

IF AND (MEMBERP "CAPRA :RIVA) (MEMBERP "LUPO :RIVA) [OP NOT LISTP :RIVA]

OP LISTP :RIVA

END

(la procedura crea un predicato che verifica se la situazione sulla riva sx è legale, cioè il lupo non mangia la capra e la capra non mangia il cavolo)

TO SIT_LEGALE_SX? :RIVA

IF AND (MEMBERP "LUPO :RIVA) (MEMBERP "CAPRA :RIVA) [OP NOT MEMBERP "LUPO :RIVA]

IF AND (MEMBERP "LUPO :RIVA) (MEMBERP "CAVOLO :RIVA) [OP MEMBERP "LUPO :RIVA]

IF AND (MEMBERP "CAPRA :RIVA) (MEMBERP "CAVOLO :RIVA) [OP NOT MEMBERP "CAPRA :RIVA]

OP LISTP :RIVA

END

TO MUOVI :MOSSA

MAKE "RSIN SENZA :MOSSA :RSIN

MAKE "RDES SE :RDES :MOSSA

IF EMPTYP :RSIN [PRINT [SUCCESSO] STAMPASTATO FINE]

IF NOT SIT_LEGALE_DX? :RDES [RIPORTA_INDIETRO]

STAMPASTATO

APPLICA_MOSSA_LECITA CHIEDIMOSSA

END

TO STAMPARSIN

TYPE "RSIN

```
TYPE SPAZIO
TYPE "E'
TYPE SPAZIO
PRINT LIST :RSIN
END

TO SPAZIO
OP CHAR 32
END

TO STAMPARDES
TYPE "RDES
TYPE SPAZIO
TYPE "E'
TYPE SPAZIO
PRINT LIST :RDES
END

TO SENZA1 :DACANCELLARE :ORIGINALE
IF EMPTYP :ORIGINALE [PRINT (SENTENCE [NON POSSO TOGLIERE] FPUT :DACANCELLARE [DALLA LISTA DATA]) FINE]
IF :DACANCELLARE = FIRST :ORIGINALE [OP BF :ORIGINALE]
OP FPUT FIRST :ORIGINALE SENZA1 :DACANCELLARE BF :ORIGINALE
END

TO SENZA :DACANCELLARE :ORIGINALE
IF EMPTYP :DACANCELLARE [OUTPUT :ORIGINALE]
MAKE "ORIGINALE SENZA1 FIRST :DACANCELLARE :ORIGINALE
OUTPUT SENZA BF :DACANCELLARE :ORIGINALE
END
```

```
TO STAMPASTATO
STAMPARSIN
STAMPARDES
END

TO INLCC
MAKE "RSIN [LUPO CAPRA CAVOLO]
MAKE "RDES []
END

TO LCC
INLCC
STAMPASTATO
APPLICA_MOSSA_LECITA CHIEDIMOSSA
END

TO APPLICA_MOSSA_LECITA :MOSSA
MAKE "NUOVA_RIVA_SX SENZA :MOSSA :RSIN
IF SIT_LEGALE_SX? :NUOVA_RIVA_SX [MUOVI :MOSSA]
PR [MOSSA NON LECITA RIPROVA]
APPLICA_MOSSA_LECITA CHIEDIMOSSA
END
```

LCC2 (seconda versione)

```
TO RIPORTA_INDIETRO
MAKE "ELEMENTO [CAPRA]
MAKE "RSIN SE :RSIN :ELEMENTO
```

```
MAKE "RDES SENZA :ELEMENTO :RDES
END

TO SIT_LEGALE_DX? :RIVA
IF AND (AND (MEMBERP "LUPO :RIVA) (MEMBERP "CAPRA :RIVA)) (MEMBERP "CAVOLO :RIVA) [OP LISTP :RIVA]
IF AND (MEMBERP "LUPO :RIVA) (MEMBERP "CAVOLO :RIVA) [OP MEMBERP "LUPO :RIVA]
IF AND (MEMBERP "CAVOLO :RIVA) (MEMBERP "CAPRA :RIVA) [OP NOT LISTP :RIVA]
IF AND (MEMBERP "CAPRA :RIVA) (MEMBERP "LUPO :RIVA) [OP NOT LISTP :RIVA]
OP LISTP :RIVA
END

TO MUOVI :MOSSA
MAKE "RDES SE :RDES :MOSSA
IF EMPTYP :RSIN [PRINT [SUCCESSO] STAMPASTATO FINE]
IF NOT SIT_LEGALE_DX? :RDES [RIPORTA_INDIETRO]
STAMPASTATO
APPLICA_MOSSA_LECITA FIRST :RSIN
END

TO STAMPARSIN
TYPE "RSIN
TYPE SPAZIO
TYPE "E'
TYPE SPAZIO
PRINT LIST :RSIN
END

TO SPAZIO
OP CHAR 32
```

```
END

TO STAMPARDES
TYPE "RDES
TYPE SPAZIO
TYPE "E'
TYPE SPAZIO
PRINT LIST :RDES
END

TO SENZA1 :DACANCELLARE :ORIGINALE
IF EMPTYP :ORIGINALE [PRINT (SENTENCE [NON POSSO TOGLIERE] FPUT :DACANCELLARE [DALLA LISTA DATA]) FINE]
IF :DACANCELLARE = FIRST :ORIGINALE [OP BF :ORIGINALE]
OP FPUT FIRST :ORIGINALE SENZA1 :DACANCELLARE BF :ORIGINALE
END

TO SENZA :DACANCELLARE :ORIGINALE
IF EMPTYP :DACANCELLARE [OUTPUT :ORIGINALE]
MAKE "ORIGINALE SENZA1 FIRST :DACANCELLARE :ORIGINALE
OUTPUT SENZA BF :DACANCELLARE :ORIGINALE
END

TO SIT_LEGALE_SX? :RIVA
IF AND (MEMBERP "LUPO :RIVA) (MEMBERP "CAPRA :RIVA) [OP NOT MEMBERP "LUPO :RIVA]
IF AND (MEMBERP "LUPO :RIVA) (MEMBERP "CAVOLO :RIVA) [OP MEMBERP "LUPO :RIVA]
IF AND (MEMBERP "CAPRA :RIVA) (MEMBERP "CAVOLO :RIVA) [OP NOT MEMBERP "CAPRA :RIVA]
OP LISTP :RIVA
END
```

```
TO STAMPASTATO
STAMPARSIN
STAMPARDES
END

TO INLCC
MAKE "RSIN [LUPO CAPRA CAVOLO]
MAKE "RDES []
END

TO LCC
INLCC
STAMPASTATO
APPLICA_MOSSA_LECITA FIRST :RSIN
END

TO APPLICA_MOSSA_LECITA :MOSSA
MAKE "RSIN SENZA1 :MOSSA :RSIN
IF SIT_LEGALE_SX? :RSIN [MUOVI :MOSSA]
MAKE "RSIN LPUT :MOSSA :RSIN
PR [MOSSA NON LECITA RIPROVA]
STAMPASTATO
PR FIRST BF :RSIN
APPLICA_MOSSA_LECITA FIRST :RSIN
END
```

LCC3 (terza versione)

```
TO APPLICA_MOSSA :MOSSA
IF MEMBERP "CONTADINO :RSIN [MSD :MOSSA STOP]
IF MEMBERP "CONTADINO :RDES [MDS :MOSSA STOP]
PR [ERRORE IN APPLICA MOSSE] STOP
END

TO STATOTRIPLA
OP (SE NUMERODI "LUPO :RSIN NUMERODI "CAPRA :RSIN NUMERODI "CAVOLO :RSIN NUMERODI "CONTADINO :RSIN)
END

TO LATODA
IF MEMBERP "CONTADINO :RSIN [OP :RSIN]
IF MEMBERP "CONTADINO :RDES [OP :RDES]
PR [ERRORE IN LATODA] STOP
END

TO NUMERODI :ELEMENTO :LISTA
IF EMPTYP :LISTA [OP 0]
IF :ELEMENTO = FIRST :LISTA [OP 1 + NUMERODI :ELEMENTO BF :LISTA]
OP NUMERODI :ELEMENTO BF :LISTA
END

TO REGISTRASTATONUOVO
MAKE "STATIVISTI FPUT STATOTRIPLA :STATIVISTI
END

TO SUCCESSO?
OP EMPTYP :RSIN
END
```

```
TO STATOVISTOPRIMA?

OP MEMBERP STATOTRIPLA :STATIVISTI

END


TO APPLICABILE? :MOVLIST

OP MEMBERP :MOVLIST LATODA

END


TO PROVATUTTE :INSIEMEDIMOSSE

IF EMPTYP :INSIEMEDIMOSSE [STOP]

APPLICA_MOSSA_LECITA :RSIN :RDES FIRST :INSIEMEDIMOSSE

PROVATUTTE BF :INSIEMEDIMOSSE

END


TO RIPORTA_INDIETRO

MAKE "RSIN LPUT FIRST :MOSSA :RSIN

MAKE "RDES SENZA :MOSSA :RDES

END


TO SIT_LEGALE_DX? :RIVA

IF AND (AND (AND MEMBERP "LUPO :RIVA MEMBERP "CAPRA :RIVA) MEMBERP "CAVOLO :RIVA) MEMBERP
"CONTADINO :RIVA [OP LISTP :RIVA]

IF AND MEMBERP "LUPO :RIVA MEMBERP "CAVOLO :RIVA [OP MEMBERP "LUPO :RIVA]

IF AND (AND MEMBERP "CAVOLO :RIVA MEMBERP "CAPRA :RIVA) NOT MEMBERP "CONTADINO :RIVA [OP NOT LISTP
:RIVA]

IF AND (AND MEMBERP "CAPRA :RIVA MEMBERP "LUPO :RIVA) NOT MEMBERP "CONTADINO :RIVA [OP NOT LISTP
:RIVA]

OP LISTP :RIVA

END
```

```
TO STAMPARSIN
TYPE "RSIN
TYPE SPAZIO
TYPE "E'
TYPE SPAZIO
PRINT LIST :RSIN
END

TO SPAZIO
OP CHAR 32
END

TO STAMPARDES
TYPE "RDES
TYPE SPAZIO
TYPE "E'
TYPE SPAZIO
PRINT LIST :RDES
END

TO MSD :MOSSA
MAKE "RSIN SENZA :MOSSA :RSIN
MAKE "RDES SENTENCE :RDES :MOSSA
END

TO SENZA1 :DACANCELLARE :ORIGINALE
IF EMPTYP :ORIGINALE [PRINT (SENTENCE [NON POSSO TOGLIERE] FPUT :DACANCELLARE [DALLA LISTA DATA]) FINE]
IF :DACANCELLARE = FIRST :ORIGINALE [OP BF :ORIGINALE]
```

```
OP FPUT FIRST :ORIGINALE SENZA1 :DACANCELLARE BF :ORIGINALE

END


TO SENZA :DACANCELLARE :ORIGINALE

IF EMPTYP :DACANCELLARE [OUTPUT :ORIGINALE]

MAKE "ORIGINALE SENZA1 FIRST :DACANCELLARE :ORIGINALE

OUTPUT SENZA BF :DACANCELLARE :ORIGINALE

END


TO MDS :MOSSA

MAKE "RDES SENZA :MOSSA :RDES

MAKE "RSIN SE :RSIN :MOSSA

END


TO SIT_LEGALE_SX? :RIVA

IF AND (AND MEMBERP "LUPO :RIVA MEMBERP "CAPRA :RIVA) NOT MEMBERP "CONTADINO :RIVA [OP NOT
MEMBERP "LUPO :RIVA]

IF AND (MEMBERP "LUPO :RIVA) (MEMBERP "CAVOLO :RIVA) [OP MEMBERP "LUPO :RIVA]

IF AND (AND MEMBERP "CAPRA :RIVA MEMBERP "CAVOLO :RIVA) NOT MEMBERP "CONTADINO :RIVA [OP NOT
MEMBERP "CAPRA :RIVA]

OP LISTP :RIVA

END


TO STAMPASTATO

STAMPARSIN

STAMPARDES

END


TO INLCC

MAKE "RSIN [LUPO CAPRA CAVOLO CONTADINO]
```

```
MAKE "RDES []
MAKE "MOSSEPOSSIBILI [[CONTADINO] [CAPRA CONTADINO] [LUPO CONTADINO] [CAVOLO CONTADINO]]
MAKE "STATIVISTI [[1 1 1 1]]
END

TO LCC
INLCC
STAMPASTATO
PROVATUTTE :MOSSEPOSSIBILI
END

TO APPLICA_MOSSA_LECITA :RSIN :RDES :MOSSA
IF NOT APPLICABILE? FIRST :MOSSA [TYPE [NON APPLICABILE -] STOP]
APPLICA_MOSSA :MOSSA
IF NOT SIT_LEGALE_SX? :RSIN [TYPE [SIT_SX NON AMMESSA -] STOP]
IF NOT SIT_LEGALE_DX? :RDES [TYPE [SIT_DX NON AMMESSA -] STOP]
IF STATOVISTOPRIMA? [TYPE [MOSSAFATTA -] STOP]
PR [-]
STAMPASTATO
PR SPAZIO
IF SUCCESSO? [PR "SUCCESSSSO FINE]
REGISTRASTATONUOVO
PAUSE
PROVATUTTE :MOSSEPOSSIBILI
PR [BACKUP]
END
```

Problema: Descrivere la finalità delle procedure del seguente programma:

SAMPLES del LOGO-IBM

TO DEMO

INIT.MENU

SHOWMENU

RUN.IT

DEMO

END

TO SHOWMENU

CT TS ;CT cancella la parte di schermo occupata dal testo, TS predispone l'intero schermo per il testo

SETTC [0 7] ;SETTC Fissa i colori per il fondo in base ad una lista di due elementi

SETCURSOR [1 9] REPEAT 24 [TYPE "\ ] ;SETCURSOR posizione *manda il cursore alla* posizione *indicata*

SETCURSOR [2 9] PR [\ IBM LOGO DEMONSTRATION\ ]

SETCURSOR [3 9] REPEAT 24 [TYPE "\ ]

SETTC [14 0]

SETCURSOR [5 6] PR [TYPE A NUMBER FOR THE PROGRAM]

SETCURSOR [6 13] PR [OF YOUR CHOICE]

SETCURSOR [9 11] PR [0 EXIT TO LOGO]

SETCURSOR [10 11] PR [1 RANDOM RHYMES]

SETCURSOR [11 11] PR [2 LOGO PROGRAM TREE]

SETCURSOR [12 11] PR [3 HOME INVENTORY FILER]

*SETCURSOR [13 11] PR [4 TOWERS OF HANOI *]*

*SETCURSOR [14 11] PR [5 HANGMAN *]*

*SETCURSOR [15 11] PR [6 RECURSIVE CURVES *]*

*SETCURSOR [16 11] PR [7 MUSICAL MEMORY GAME *]*

SETCURSOR [19 0]

PR [IBM Color\/Graphics Adapter required]*

SETCURSOR [23 3]

SETTC [10 0]

PR [\ \ \ \ \ \(C\) Copyright LCSI 1983]

SETTC [7 0]

WAITKEY

END

TO WAITKEY

WAIT 4 ; WAIT n interrompe l'esecuzione del programma per un numero di volte pari ad n per 1/18 di
; secondo

IF KEYP [STOP] ;KEYP risponde true se un tasto è stato premuto ma non è ancora stato letto

WAITKEY

END

TO INIT.MENU

; PACKAGE pacchetto nome(i) raccoglie i programmi indicati ed eventualmente anche le vatriabili sotto la voce

; pacchetto

PACKAGE "PRINC [DEMO VALID SHOWMENU WAITKEY RUN.IT LOAD.IT INIT.MENU TITLE WRITEWORD]

;BURY pacchetto nasconde il contenuto del pacchetto indicato (programmi e variabili), quindi l'uso di POTS, PONS, ;POPS, ERNS, SAVE non esplica nessun effetto su pacchetto.

BURY "PRINC

ERPS ; cancella tutti i programmi

ERNS ; cancella tutte le variabili

SETPRECISION 5 ;SETPRECISION n fissa il numero di cifre decimali n che un numero può avere nel corso di un
;calcolo. PRECISION indica il numero di cifre significative di un numero.

END

TO LOAD.IT

CT TS

SETCURSOR [12 2] PR SE [LOADING FILE] :RUN

LOAD WORD "FILE :RUN

```
RECYCLE  ; forza l'eliminazione delle risorse (variabili, spazio di memoria,...) non più utili

START

END


TO RUN.IT

MAKE "RUN RC

IF NOT VALID [SETCURSOR [21 6] SETTC [14 8] PR [A number from 0 to 7, please.] SETTC [7 0] WAIT 30 RUN.IT]

IF :RUN = 0 [TS CT THROW "TOPLEVEL]    ;THROW parola rimette il controllo dell'esecuzione alla
                                        ;corrispondente parola indicata da CATCH

CT

IF :RUN = 2 [TEST .EXAMINE 61440 65534 = 253 IFTRUE [SETCURSOR [11 10] PR [LOGO PROGRAM TREE] SETCURSOR
[13 6] PR [REQUIRES ADDITIONAL MEMORY] WAIT 50 STOP]]

IF :RUN > 3 [CATCH "ERROR [CS HT LOAD.IT] PR [MAKE ANOTHER CHOICE] WAIT 50 STOP]

LOAD.IT

END


TO VALID

IF NOT NUMBERP :RUN [OP "FALSE]

IF NOT (AND :RUN > -1 :RUN < 8) [OP "FALSE]

OP "TRUE

END


TO WRITEWORD :WRD

IF EMPTYP :WRD [FD 10 STOP]

SETSHAPE ASCII FIRST :WRD   ;SETSHAPE  n indica alla tartaruga un'altra forma di rappresentazione ove ad n si
                            ;possono attribuire valori fra 0 e 255.

STAMP                       ; ordina la stampa del disegno della tartaruga

FD 12

WRITEWORD BF :WRD

END
```

```
TO TITLE :XC :YC :LISTE
(LOCAL "NUM)
MAKE "NUM 1
PU
SETPOS SE :XC :YC                ;SETPOS posizione invia la tartaruga nella sede indicata da posizione
SETH 90                  ; SETH gradi assegna alla tartaruga la direzione definita da gradi
ST                               ; rende visibile la tartaruga
REPEAT COUNT :LISTE [WRITEWORD ITEM :NUM :LISTE MAKE "NUM :NUM + 1]
SETSHAPE "TURTLE
HT                               ;rende invisibile la tartaruga
PD                               ;pone la penna della tartaruga a contatto con la superficie
SETH 0
END

MAKE "STARTUP [DEMO]
```

Problema: scrivere la funzione valore assoluto.

codifica logo TOOL-IBM

```
TO ABS :NUM
; [L' OUTPUT DI ABS E']
; [SEMPRE  POSITIVO]
OP IF :NUM < 0 [-:NUM] [:NUM]
END
```

Problema : decrivere l'obiettivo delle seguenti procedure:

```
TO ARCL :RADIUS :DEGREES
LOCAL "STEP LOCAL "REM
MAKE "STEP 2 * :RADIUS * 3.1416 / 36
MAKE "REM REMAINDER :DEGREES 10
REPEAT :DEGREES / 10 [LT 5 FD :STEP LT 5]
```

```
IF :REM > 0 [FD :STEP * :REM / 10 LT :REM]
END
TO ARCR :RADIUS :DEGREES
LOCAL "STEP LOCAL "REM
MAKE "STEP 2 * :RADIUS * 3.1416 / 36
MAKE "REM REMAINDER :DEGREES 10
REPEAT :DEGREES / 10 [RT 5 FD :STEP RT 5]
IF :REM > 0 [FD :STEP * :REM / 10 RT :REM]
END
TO CIRCLEL :RADIUS
LOCAL "STEP                    ;LOCAL nome/i trasforma in variabili locali le variabili dai nomi assegnati
MAKE "STEP 2 * :RADIUS * 3.1416 / 36
REPEAT 36 [LT 5 FD :STEP LT 5]
END
TO CIRCLER :RADIUS
LOCAL "STEP
MAKE "STEP 2 * :RADIUS * 3.1416 / 36
REPEAT 36 [RT 5 FD :STEP RT 5]
END
TO CONVERT :N :FRBASE :TOBASE
OP DEC.TO.ANYBASE ANYBASE.TO.DEC :N :FRBASE 1 :TOBASE
END
TO ANYBASE.TO.DEC :N :BASE :POWER
IF EMPTYP :N [OP 0]
OP ( :POWER * C.TO.N LAST :N ) + ANYBASE.TO.DEC BL :N :BASE :POWER * :BASE
END
TO DEC.TO.ANYBASE :N :BASE
IF :N < :BASE [OP N.TO.C :N]
OP WORD DEC.TO.ANYBASE INT QUOTIENT :N :BASE :BASE N.TO.C REMAINDER :N :BASE
```

```
END

TO C.TO.N :N

IF NUMBERP :N [OP :N]

OP (ASCII :N) - 55

END


TO N.TO.C :N

IF :N < 10 [OP :N]

OP CHAR 55 + :N

END


TO DECTOHEX :N

OP CONVERT :N 10 16

END


TO HEXTODEC :N

OP CONVERT :N 16 10

END


TO DIVISORP :A :B

OP 0 = REMAINDER :B :A

END


TO DRIVE

IF KEYP [LISTEN]

FD 1

DRIVE

END
```

```
TO LISTEN

MAKE "ANS RC            ;RC , READCHAR ricava il simbolo letto dalla periferica attuale (di norma la
                        ;tastiera). Se non vi sono simboli disponibili, avviene l'interruzione del programma
                        ;sino all'immisione di un carattere.

IF :ANS = "S [THROW "TOPLEVEL]

IF :ANS = "R [RT 10]

IF :ANS = "L [LT 10]

END


TO FOREVER :INSTRUCTIONLIST

RUN :INSTRUCTIONLIST        ;RUN lista esegue le istruzioni nella lista

FOREVER :INSTRUCTIONLIST

END


TO LABELPIC :LABEL :POS

CT                          ;cancella la parte dello shermo occupata dal testo

LOCAL "CURSORPOS

MAKE "CURSORPOS CURSOR

SETCURSOR :POS

TYPE :LABEL

SETCURSOR :CURSORPOS

END


TO MAP :CMD :LIST

IF EMPTYP :LIST [STOP]

RUN LIST :CMD WORD "" FIRST :LIST

MAP :CMD BF :LIST

END


TO PEL :POS
```

```
OUTPUT ITEM (BYTEPOS (ROUND FIRST :POS)) 4PELS :POS

END


TO BYTEPOS :XPEL

IF :XPEL > 0 [OUTPUT 1 + REMAINDER (:XPEL - 1) 4] [OUTPUT 4 + REMAINDER :XPEL 4]

END


TO 4PELS :POS

LOCAL "P

LOCAL "N

MAKE "P []

MAKE "N DISPBYTE :POS

REPEAT 4 [MAKE "P FPUT REMAINDER :N 4 :P MAKE "N INT QUOTIENT :N 4]

OUTPUT :P

END


TO DISPBYTE :POS

OUTPUT .EXAMINE -18432 OFFSET :POS

END


TO OFFSET :POS

LOCAL "SCAN

MAKE "SCAN 100 - ROUND ((LAST :POS) * .SCRUNCH)

IF OR OR FIRST :POS > 160 FIRST :POS < -159 OR :SCAN > 199 :SCAN < 0 [PR [PLEASE ENTER A POSITION WHICH IS] PR
[WITHIN THE VISIBLE LIMITS OF THE] PR [GRAPHICS SCREEN.] THROW "TOPLEVEL]

OUTPUT (8192 * REMAINDER :SCAN 2) + (80 * INT QUOTIENT :SCAN 2) + (INT QUOTIENT (159 + ROUND (FIRST :POS))
4)

END


TO GREEN
```

```
SETPAL 0                ;STPAL determina la tavolozza 0 o 1
OP 1
END

TO RED
SETPAL 0
OP 2
END

TO BROWN
SETPAL 0
OP 3
END

TO CYAN
SETPAL 1
OP 1
END

TO MAGENTA
SETPAL 1
OP 2
END

TO WHITE
SETPAL 1
OP 3
END
```

```
TO POLY :SIDE :ANGLE
FD :SIDE
RT :ANGLE
POLY :SIDE :ANGLE
END

TO PROMPT :MESSAGE
TYPE :MESSAGE
TYPE CHAR 32
END

TO SHOWARGS :ARGLIST
IF EMPTYP :ARGLIST [STOP]
MAKE "NEWDEF LPUT (LIST "PRINT "SENTENCE (LIST (FIRST :ARGLIST) "IS) (WORD ": FIRST :ARGLIST)) :NEWDEF
SHOWARGS BF :ARGLIST
END

TO SHOWLINES :INSTRUCTIONS
IF EMPTYP :INSTRUCTIONS [STOP]
MAKE "NEWDEF LPUT (LIST "TYPE FIRST :INSTRUCTIONS) :NEWDEF
MAKE "NEWDEF LPUT [STEPPER] :NEWDEF
MAKE "NEWDEF LPUT FIRST :INSTRUCTIONS :NEWDEF
SHOWLINES BF :INSTRUCTIONS
END

TO STEPPER
TYPE "??
IGNORE READLIST          ;READLIST, RL riporta la linea letta (lista) dell'archivio indicato (di norma la tastiera)
END
```

```
TO IGNORE :INPUT
END

TO UNSTEP :PROC
IF EMPTYP :PROC [STOP]
IF LISTP :PROC [UNSTEP FIRST :PROC UNSTEP BF :PROC STOP]
IF EMPTYP TEXT WORD ". :PROC [PR SE :PROC [NOT STEPPED.] STOP]
COPYDEF :PROC WORD ". :PROC
ERASE WORD ". :PROC
END

TO STEP :PROC
IF EMPTYP :PROC [STOP]
IF LISTP :PROC [STEP FIRST :PROC STEP BF :PROC STOP]
IF PRIMITIVEP :PROC [PR SE [CAN'T STEP PRIMITIVE] :PROC STOP]
IF EMPTYP TEXT :PROC [PR SE [NO PROCEDURE NAMED] :PROC STOP]
COPYDEF WORD ". :PROC :PROC
MAKE "OLDDEF TEXT :PROC
MAKE "NEWDEF (LIST FIRST :OLDDEF)
MAKE "NEWDEF LPUT (LIST "PRINT (LIST "ENTERING :PROC)) :NEWDEF
SHOWARGS FIRST :OLDDEF
SHOWLINES BF :OLDDEF
DEFINE :PROC :NEWDEF
END

TO TEACH
LOCAL "THISLINE
DEFINE "PROGRAM [[]]
```

```
CLEARSCREEN PENDOWN

ADDLINES

IF NOT EMPTYP BUTFIRST TEXT "PROGRAM [NAMEIT]

END


TO ADDLINES

TYPE [\>?]

MAKE "THISLINE READLIST

IF :THISLINE = [END] [STOP]

ADDIT :THISLINE

ADDLINES

END


TO ADDIT :THISLINE

IF EMPTYP :THISLINE [STOP]

TEST :THISLINE = [ERASE]

IFTRUE [WIPEOUT]

IFFALSE [RUNSTORE]

END


TO WIPEOUT

DEFINE "PROGRAM BUTLAST TEXT "PROGRAM

CLEARSCREEN

RUN [PROGRAM]

END


TO RUNSTORE

CATCH "ERROR [RUN :THISLINE]

LOCAL "GOOF
```

```
MAKE "GOOF ERROR
TEST EMPTYP :GOOF
IFTRUE [DEFINE "PROGRAM LPUT :THISLINE TEXT "PROGRAM]
IFFALSE [PRINT FIRST BUTFIRST :GOOF]
END

TO NAMEIT
LOCAL "NAME
PRINT [What should I call this?]
MAKE "NAME READLIST
TEST EMPTYP :NAME
IFTRUE [ERASE "PROGRAM STOP]
TEST DEFINEDP FIRST :NAME
IFTRUE [TRYAGAIN]
IFFALSE [COPY]
END

TO TRYAGAIN
PRINT SENTENCE FIRST :NAME [is already defined]
PRINT []
NAMEIT
END

TO COPY
COPYDEF FIRST :NAME "PROGRAM
PRINT SENTENCE FIRST :NAME [defined]
ERASE "PROGRAM
END
```

```
TO PROGRAM

END


TO BLACK.BLUE

OP [0 1]

END


TO BLACK.GREEN

OP [0 2]

END


TO BLACK.CYAN

OP [0 3]

END


TO BLACK.RED

OP [0 4]

END


TO BLACK.MAGENTA

OP [0 5]

END


TO BLACK.BROWN

OP [0 6]

END


TO BLACK.WHITE

OP [0 7]
```

```
END

TO WRITEWORD :WD
IF EMPTYP :WD [FD 5 STOP]
SETSHAPE ASCII FIRST :WD
STAMP FD 12
WRITEWORD BF :WD
END

TO TITLE :XC :YC :LST
MAKE "NUM 1
PU SETPOS SE :XC :YC
RT 90 ST
REPEAT COUNT :LST [WRITEWORD ITEM :NUM :LST MAKE "NUM :NUM + 1]
SETSHAPE "TURTLE
HT PD SETH 0
END

TO WHICH :MEMBER :LIST
IF NOT MEMBERP :MEMBER :LIST [OP 0]
IF :MEMBER = FIRST :LIST [OUTPUT 1]
OUTPUT 1 + WHICH :MEMBER BF :LIST
END
TO WHILE :CONDITION :INSTRUCTIONLIST
TEST RUN :CONDITION
IFFALSE [STOP]
RUN :INSTRUCTIONLIST
WHILE :CONDITION :INSTRUCTIONLIST
END
```