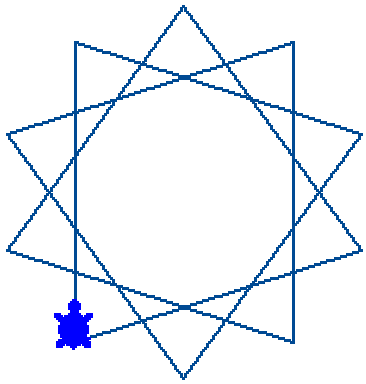


Introduzione a MSW-Logo

Corso di istruzione

di Guido Gay

Milano, gennaio 2002

Questo documento è stato redatto utilizzando come base di partenza un manuale introduttivo a MSW-Logo predisposto da Paolo Lazzarini, che ringraziamo sentitamente, disponibile al seguente indirizzo: http://users.iol.it/prof.lazzarini/mswlogo/

Prima edizione: gennaio 2002

Sommario

Premessa

Logo è un linguaggio di programmazione ideato con finalità didattiche dal matematico e informatico americano Seymour Papert. E' un linguaggio ormai diffuso nelle scuole di tutto il mondo. Papert, derivando alcune idee dalla teoria dell'apprendimento di Piaget, propone un ambiente di sperimentazione geometrica che coinvolge l'allievo, lo rende diretto costruttore di strutture, gli consente di apprendere operando.

Una caratteristica importante di Logo è quella di favorire non solo l'apprendimento di una corretta tecnica di programmazione ma anche l'acquisizione di nozioni e concetti matematici profondi (in particolare geometrici, ma non solo: si pensi ad esempio al concetto di variabile).

Operare in ambiente Logo significa programmare una piccola tartaruga che si muove sullo schermo del computer in risposta a dei nostri comandi. La tartaruga, come entità geometrica, è caratterizzata dalla posizione nel piano e dall'orientamento (ad esempio la tartaruga può trovarsi in un dato punto P ed essere orientata verso Nord).

I comandi fondamentali per muovere la tartaruga sono il comando `avanti`, `indietro` e i comandi `destra` e `sinistra`. Il comando `avanti` fa avanzare la tartaruga di un numero di "passi" che possiamo stabilire a nostro piacere; l'avanzamento avviene nella direzione determinata dall'attuale orientamento della tartaruga. `Indietro` fa tornare indietro la tartaruga. I comandi `destra` e `sinistra` ci consentono di modificare l'orientamento della tartaruga facendola ruotare su se stessa in senso orario o antiorario (l'ampiezza della rotazione è a nostra scelta). La tartaruga muovendosi lascia una traccia sullo schermo, potremo così disegnare qualsiasi figura geometrica se sapremo descriverne proceduralmente la costruzione.

MSW-Logo è una delle migliori implementazioni del linguaggio Logo sotto MS-Windows (Win95, Win98, …) ed è stato sviluppato George Mills a partire da UCBLOGO di Brian Harvey. Paolo Passaro ha recentemente curato la traduzione in italiano di MSW-Logo 6.4. MSW-Logo è un software open source e potete utilizzarlo liberamente.

Qui di seguito trovate un'introduzione sintetica all'uso di MSW-Logo, versione italiana.

Primi passi con MSW-Logo

Installazione

Se MSWLogo è già installato sul vostro calcolatore potete passare ad “Avvio”.

In caso contrario, si possono seguire i seguenti tre semplici passaggi:

Scaricare il file LogoIt.exe da questo sito web: http://www.racine.ra.it/curba/links.htm .

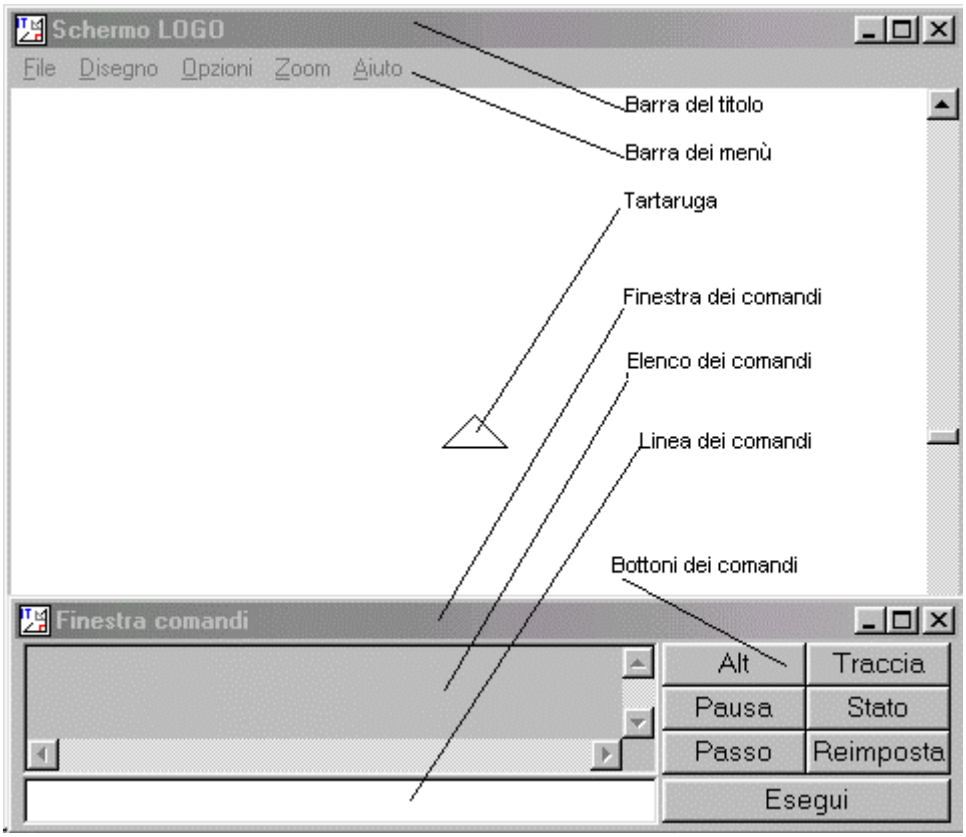
Copiare il file LogoIt.exe alla radice del disco fisso del calcolatore (probabilmente C:\>).

Scompattare il file (un modo è questo: dal prompt C:\> battere LogoIt.exe seguito da un *invio).*

Se tutto è andato bene Logo è ora installato sul vostro calcolatore.

Avvio

Fare un *doppio click* – cioè premere due volte velocemente il tasto sinistro del mouse – sull'icona di Logo sul desktop. Dovrebbe comparire una figura simile a quella riportata qui di seguito.



Parti dello schermo

In alto c'è la Barra del titolo, che dà il nome alla finestra aperta ("Schermo LOGO"). La Barra dei menù è appena sotto la barra del titolo. Contiene il nome dei menù che descriveremo tra poco.

Al centro di un'area bianca c'è un cursore triangolare (la tartaruga). Sotto c'è la Finestra dei comandi, in cui vengono digitati i comandi sulla Linea dei comandi . La finestra dei comandi è separata dalla finestra denominata "Schermo LOGO". Ciò significa che può essere spostata, rimpicciolita, ingrandita o trasformata in una icona.

Sopra la linea dei comandi c'è uno spazio denominato Elenco dei comandi, su cui scrivono comandi come `mostra` o `stampa`, in cui Logo scrive i messaggi di sistema in caso di errore ed in cui vengono riscritti i comandi digitati nella linea dei comandi.

Alla destra c'è un gruppo di bottoni descritti di seguito.

Bottoni dei comandi

Alt: ferma ogni azione di Logo.

Traccia: attiva il comando traccia. Ricliccando si attiva il comando opposto notraccia.

Pausa: Logo si blocca temporaneamente e attende il comando `continua`.

Stato: apre una finestra che mostra alcune informazioni sulle azioni eseguite dal Logo in quel momento.

Passo: indica a Logo di non permettere l'utilizzo di altri programmi mentre Logo sta funzionando.

Reimposta: funziona come il comando `puliscischermo(ps)`, ripristina e pulisce lo schermo.

Esegui: è come schiacciare il tasto *Invio*, dice alla tartaruga di eseguire quanto contenuto sulla riga dei comandi.

I menù di logo

Ci sono cinque menù nella barra dei menù.

File: gestione delle procedure, uscita da Logo.

Nuovo: Serve per cancellare tutte le procedure caricate in memoria.

Apri: appare un nuovo menù in cui selezionare un file che contiene le procedure da caricare**.**

Salva: salva le procedure create.

Salva con nome: appare un nuovo menù in cui scrivere il nome di un file che conterrà le procedure salvate.

Modifica: appare un nuovo menù in cui selezionare una procedura da modificare.

Cancella: appare un nuovo menù in cui selezionare una procedura da cancellare.

Esci: uscita da Logo.

Disegno: comandi per aprire, salvare e stampare le immagini create con i comandi grafici (`avanti, sinistra`, …)

Nuovo: Cancella lo schermo.

Apri: appare un nuovo menù in cui selezionare un file che contiene l'immagine salvata in precedenza**.**

Salva: salva l'immagine creata.

Salva con nome: appare un nuovo menù in cui scrivere il nome di un file che conterrà l'immagine salvata.

Stampa: stampa l'immagine creata.

Imposta stampante: seleziona la stampante e le caratteristiche della stampa.

Area attiva: permette di selezionare l'area di lavoro da stampare o da salvare.

Opzioni: questo menù permette di cambiare le dimensioni ed i colori della penna, quelli dello schermo, le dimensioni ed il tipo dei caratteri utilizzati da Logo.

Zoom: permette di ingrandire o rimpicciolire i disegni .

Aiuto: se si ha bisogno di chiedere istruzioni su come fare qualcosa in Logo, il menù d'aiuto ti offre alcune scelte:

Indice: lista dei capitoli in cui è suddiviso il volume degli aiuti collegati a Logo**.**

MCI: serve a chi programma strumenti multimediali

Uso dell'aiuto: aiuto su come utilizzare l'aiuto

Tutorial: riporta all'indice…

Dimostrazione: illustra alcune delle possibilità di utilizzo di Logo

Primi passi

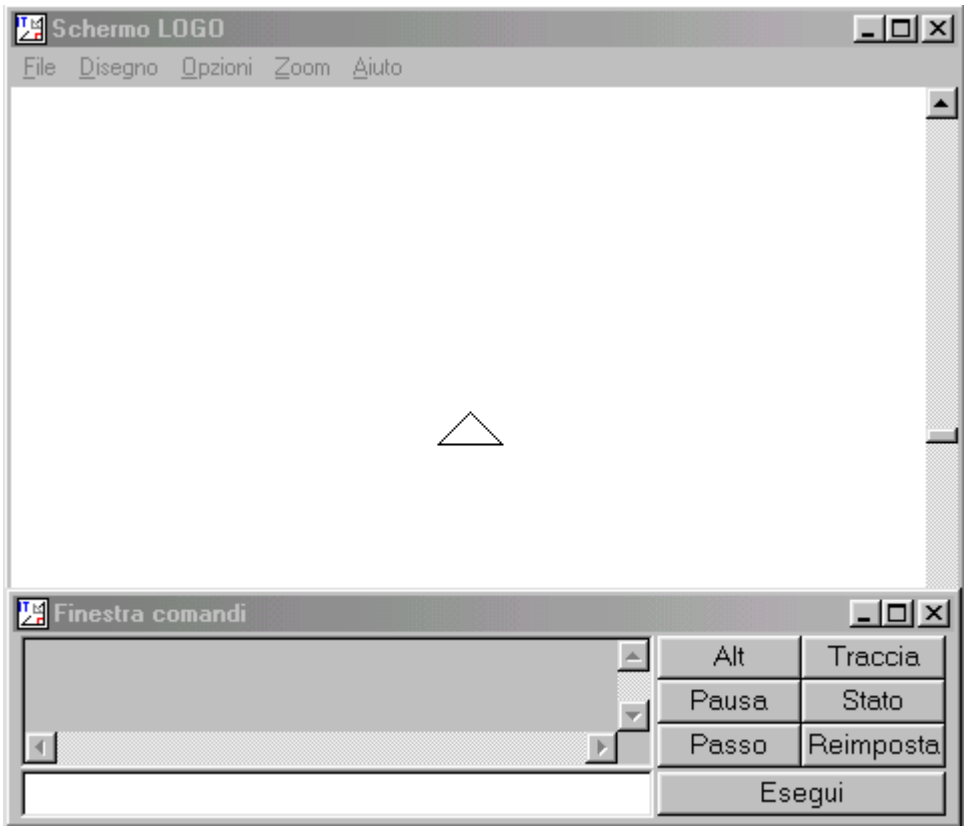
L'area bianca è l'ambiente all'interno del quale potremo muovere la tartaruga, è il mondo della tartaruga; al centro dello schermo vedete un triangolino isoscele che rappresenta proprio la tartaruga. Nella situazione iniziale la tartaruga punta verso l'alto (verso nord); questo stato iniziale della tartaruga (posizione al centro, orientamento verso nord) viene denominato tana (a cui corrisponde il comando `tana` che riporta la tartaruga in quello stato).

Ora provate a digitare sulla linea dei comandi l'istruzione

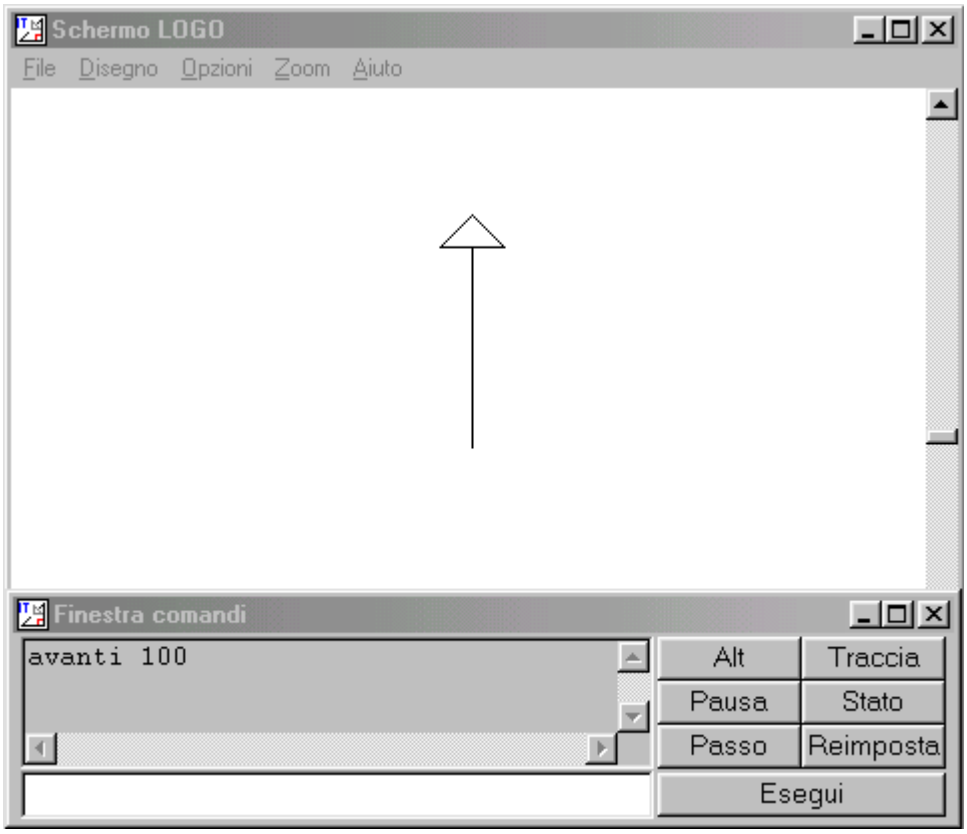
```
avanti 100
```

che serve a far avanzare la tartaruga di 100 passi nella direzione determinata dal suo orientamento attuale (che è verso nord). Per mettere in esecuzione l'istruzione dovete premere sul tasto Invio oppure cliccare sul bottone Esegui. Notate inoltre che Logo non distingue tra lettere maiuscole e minuscole, pertanto `AVANTI 100`, `avanti 100` oppure `AvanTi 100` sono comandi equivalenti. In questo manuale usiamo per i comandi sempre le lettere minuscole, in un carattere diverso da quello del testo.

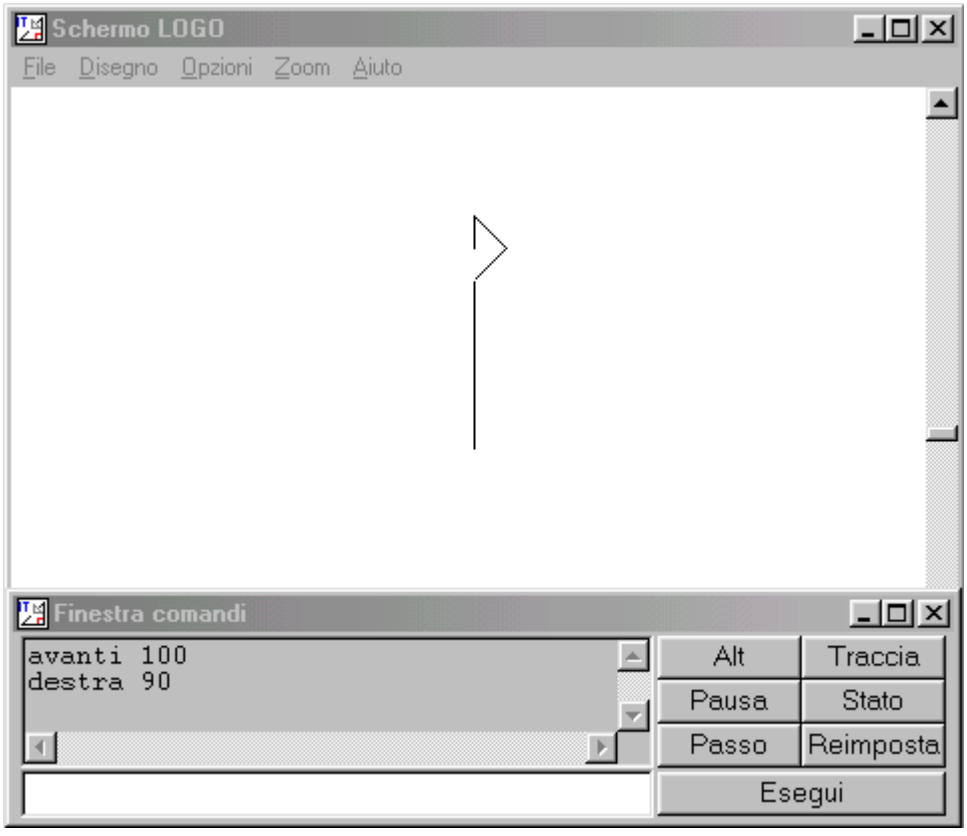
La situazione iniziale è la seguente.



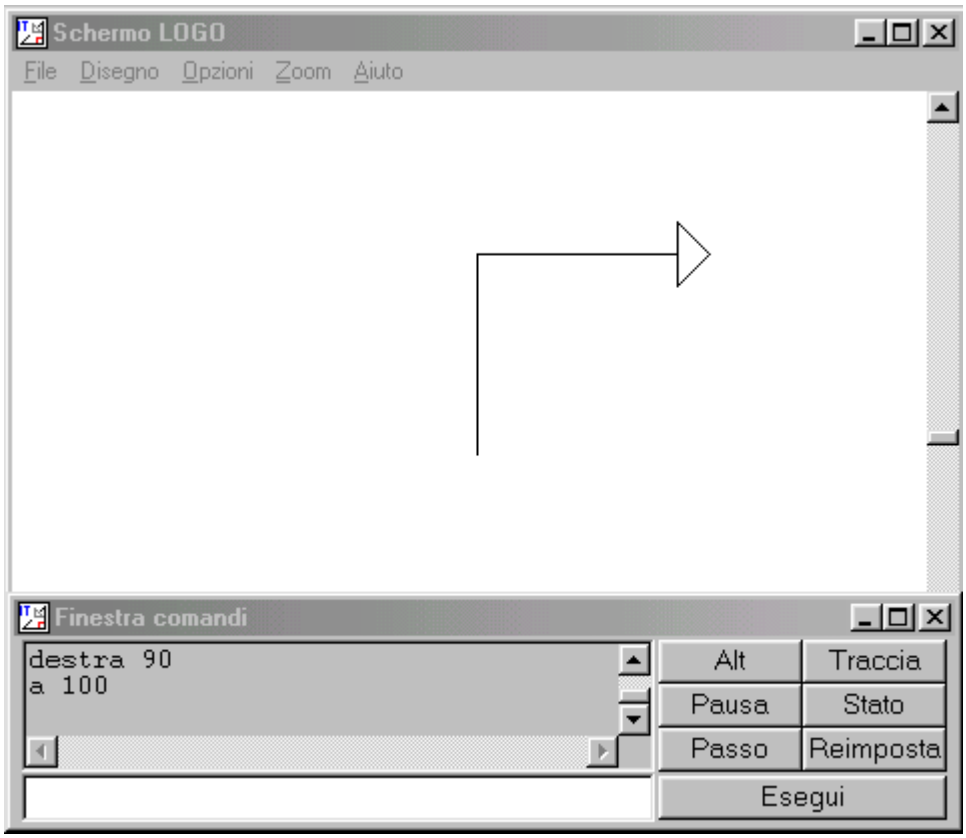
Qui sotto invece vedete l'effetto del comando impartito dopo aver digitato Invio. La tartaruga, muovendosi, ha lasciato una traccia: abbiamo quindi ottenuto un segmento verticale lungo 100 passi di tartaruga (100 pixel). Osservate che nella lista delle istruzioni al di sopra del campo di input appare ora l'istruzione che abbiamo appena digitato e che potremo rieseguire senza doverla riscrivere (basta cliccarci sopra due volte in rapida successione).



Ed ecco come si presenta la situazione dopo aver eseguito l'istruzione `destra 90` che fa ruotare la tartaruga di 90° in senso orario (cambia solo l'orientamento, non la posizione).



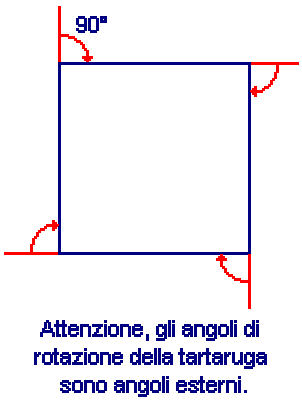
E' chiaro che se ora daremo di nuovo l'istruzione `avanti 100` la tartaruga, tenendo conto del nuovo orientamento, avanzerà ancora di 100 passi ma questa volta verso est.



Esaminiamo ora tutte le istruzioni necessarie per tracciare un quadrato:

```
avanti 100
destra 90

avanti 100
destra 90

avanti 100
destra 90

avanti 100
destra 90
```

Vi rendete conto che questo blocco di istruzioni rappresenta una descrizione operativa (procedurale) del quadrato. Per poter disegnare un quadrato con il Logo, l'allievo deve sapere quali proprietà caratterizzano questa figura. Deve rendersi conto che gli angoli sono tutti retti (la rotazione della tartaruga è sempre di 90°) e i lati tutti uguali (l'avanzamento è sempre di 100 passi). Inoltre la regolarità della figura appare nella regolarità che il blocco di istruzioni presenta (ci sono due istruzioni che si ripetono quattro volte).



Ora vi chiederete se il Logo sia tutto qui. Certamente no. Finora abbiamo operato in modo diretto dando un'istruzione per volta. Il passo successivo consiste nel definire delle procedure. E' solo costruendo delle procedure che potremo cominciare a renderci conto delle potenzialità logiche e geometriche del Logo. Ce ne occuperemo nella sezione successiva.

Un'ultima cosa: come si fa ad uscire dall'ambiente MSW-Logo? Avete due possibilità: selezionare l'opzione "Esci" del menu "File" della finestra principale oppure dare in modo diretto, sulla riga dei comandi della finestra comandi, l'istruzione `ciao`.

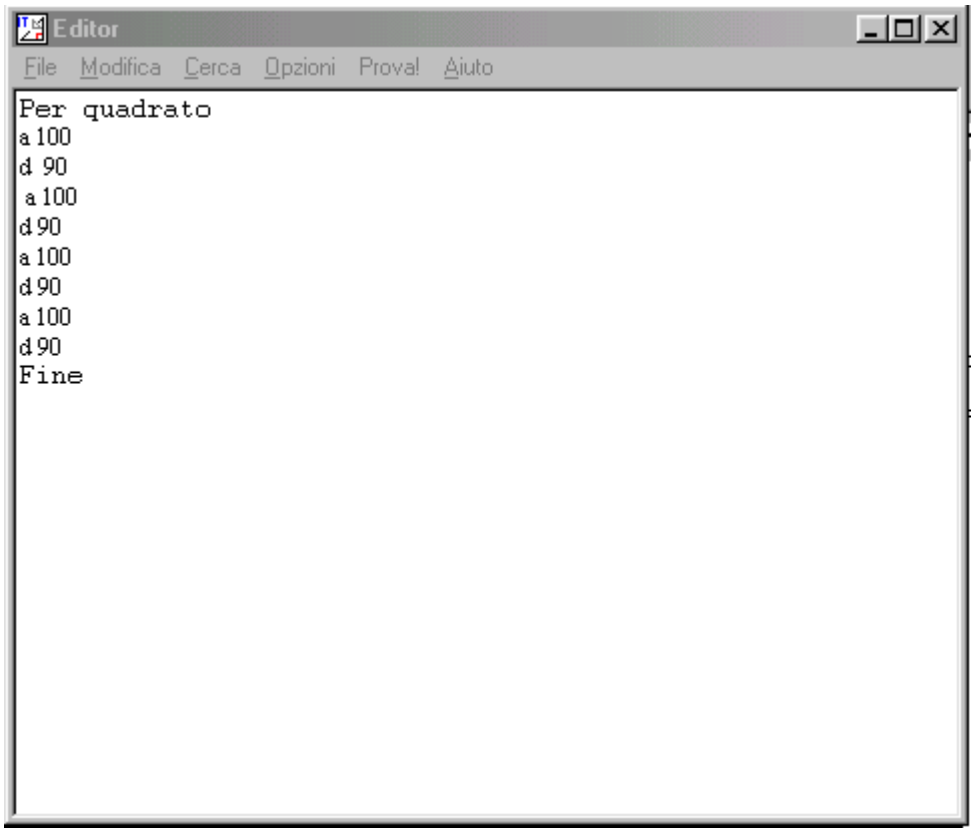
Scrivere ed eseguire una procedura

Per introdurre la nozione di procedura sarà opportuno utilizzare con i ragazzi una metafora. La tartaruga è in grado di comprendere un certo numero di comandi senza che le si debba spiegare il loro significato. Questi comandi sono chiamati istruzioni primitive; sono istruzioni primitive, ad esempio, le istruzioni avanti e destra che abbiamo utilizzato per costruire il quadrato della sezione precedente. Ma la tartaruga è anche in grado di apprendere nuovi comandi se saremo in grado di esprimerli mediante istruzioni primitive. Se ad esempio vogliamo insegnare alla tartaruga come si fa a tracciare un quadrato dovremo definire la procedura seguente:

```
per quadrato
a 100
d  90
a 100
d 90        (“a” è una abbreviazione di avanti, “d” di destra)
a 100
d 90
a 100
d 90
fine
```

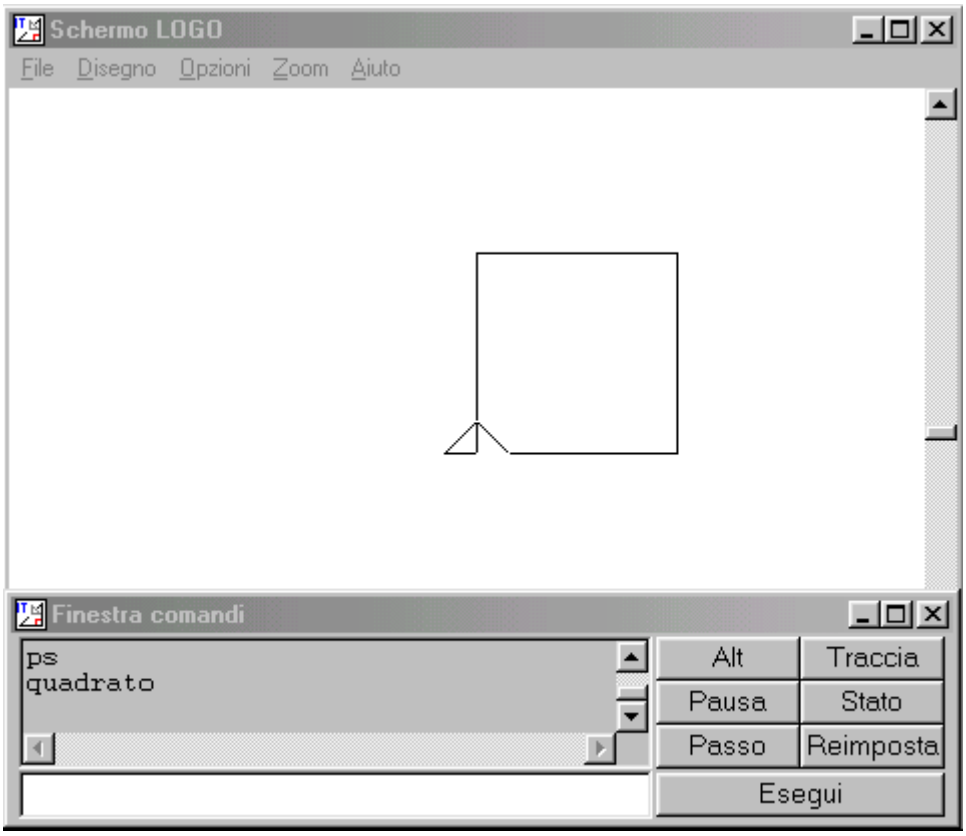
Come vedete si tratta proprio delle stesse istruzioni che abbiamo dato in modo diretto per ottenere un quadrato di lato 100. Attenzione però, all'inizio del blocco di istruzioni troviamo la dichiarazione “per quadrato” ("per tracciare un quadrato") e alla fine la parola “fine” (che indica la fine della procedura). Ecco fatto, in questo modo abbiamo spiegato alla tartaruga il significato del nuovo comando quadrato. La procedura quadrato si comporta ora come una delle istruzioni primitive: è come se avessimo ampliato il numero di vocaboli "compresi" dalla tartaruga.

Vediamo ora praticamente come si procede. Scriveremo il testo della procedura in un nuovo ambiente chiamato editor. Per accedere all'editor possiamo selezionare dal menu "File" della finestra principale la voce "Modifica". Procedendo in questo modo, ci si presenterà un finestra, denominata Modifica procedura, che contiene l'elenco di tutte le procedure definite (sinora nessuna). Facciamo click sul bottono "ok", entrando così nell'editor. La vediamo nella figura precedente (in cui abbiamo già dato il nome "quadrato"alla procedura che vogliamo definire).
Digitiamo il testo della procedura. Nella finestra dell'editor abbiamo a disposizione tutti i servizi che normalmente sono disponibili in un word processor (ad esempio taglia, copia, incolla, cerca). Il risultato è rappresentato nella figura seguente.



Dopo aver digitato la procedura usciremo dall'editor mediante l'opzione "Salva ed esci" che troviamo nel menu "File".

Non rimane che mettere in esecuzione la nostra procedura. Prima però è opportuno pulire lo schermo cancellando i disegni tracciati precedentemente: per far questo utilizzeremo in modo diretto il comando `ps` (`puliscischermo`) che ha anche la funzione di riportare la tartaruga nella tana (stato iniziale). Bene, digitiamo sulla riga dei comandi la parola `quadrato` seguita da Invio. Abbiamo ottenuto, in un sol colpo, il nostro quadrato. Nella figura seguente vedete il risultato.



Tenete infine presente che per tornare nell'editor possiamo selezionare dal menu "File" della finestra principale la voce "Modifica". Procedendo in questo secondo modo, ci si presenterà un finestra, denominata Modifica procedura, che contiene l'elenco di tutte le procedure definite. Selezioneremo "quadrato", l'unica procedura che per il momento abbiamo definito, e cliccheremo sul bottone "ok".

L'istruzione ripeti

Riconsideriamo la procedura per definire un quadrato di lato 100

Come vedete, abbiamo dovuto ripetere uno stesso blocco di istruzioni per quattro volte. Il Logo ci consente di fare di meglio utilizzando l'istruzione `ripeti`. Ecco come si trasforma la nostra procedura

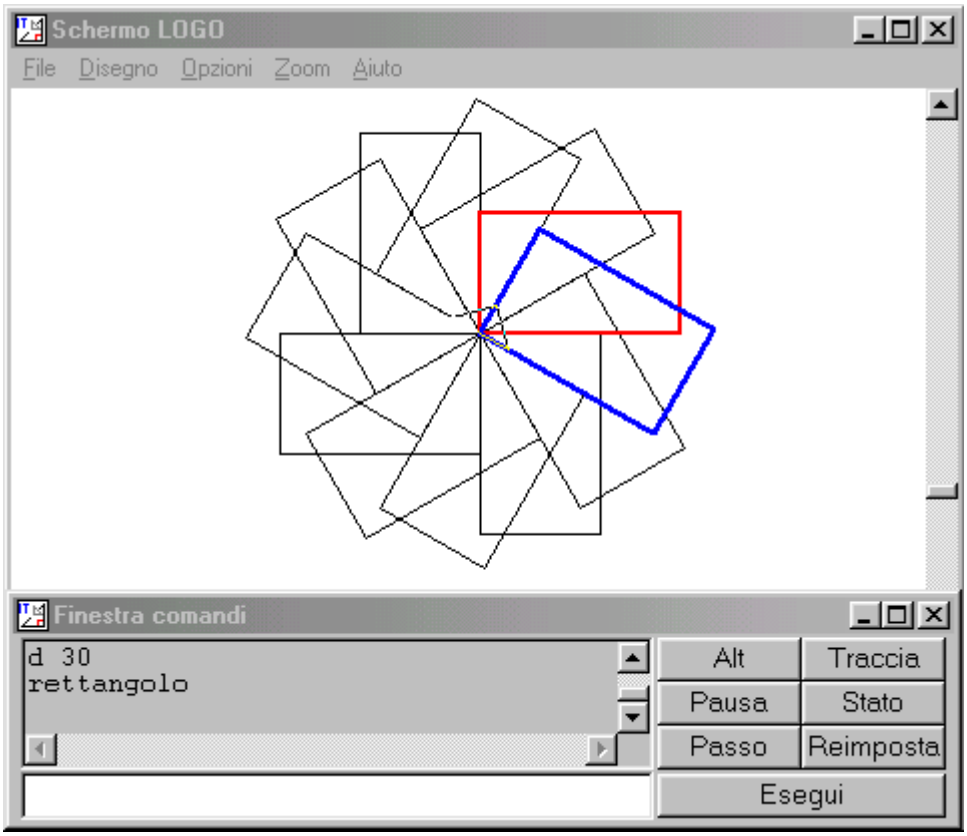
```
per quadrato
      ripeti 4 [a 100 d 90]
fine
```

Tenete presente che il blocco di istruzioni che si ripete, nel nostro caso di due istruzioni, va messo tra parentesi quadre.

Considerate ora queste due procedure

```
per rettangolo
      ripeti 2 [a  60 d  90 a 100 d 90]
fine

per RuotaRettangolo
      ripeti 12 [rettangolo d 30]
fine
```

Mettendo in esecuzione la procedura `RuotaRettangolo` otteniamo la schermata seguente.



E' il caso di fare alcune osservazioni.

La procedura `rettangolo` traccia un rettangolo di altezza 60 e di base 100.

La procedura `RuotaRettangolo` usa al suo interno la procedura rettangolo, da noi precedentemente definita, come una qualsiasi altra istruzione.

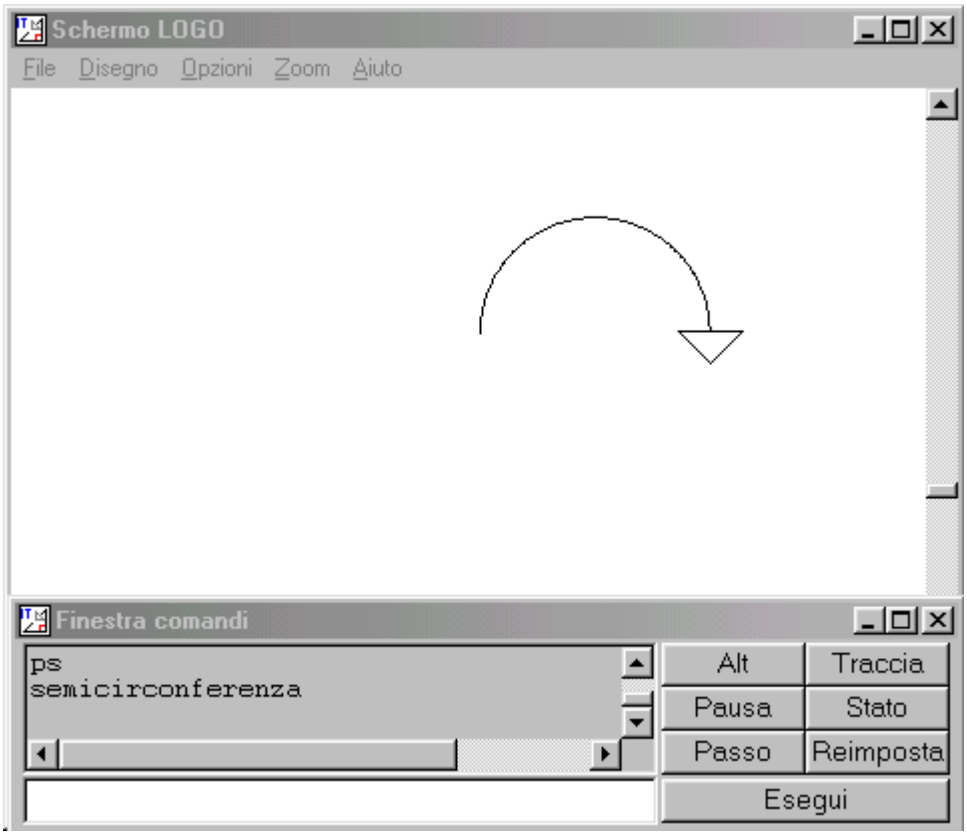
La procedura `RuotaRettangolo` ripete 12 volte le due istruzioni `rettangolo` e `d 30`: viene tracciato un primo rettangolo (evidenziato in rosso), poi la tartaruga ruota di 30° e quindi traccia il secondo rettangolo partendo in una nuova direzione (evidenziato in blu), di nuovo ruota di 30° e traccia il terzo rettangolo e così via fino a completare il giro (vengono eseguite 12 rotazioni di 30° per un totale di 360°).

Uso creativo di ripeti

Con i soli comandi `avanti, destra e ripeti` si possono creare delle composizioni molto interessanti.

Costruiamo per prima cosa una procedura che disegna una semicirconferenza.

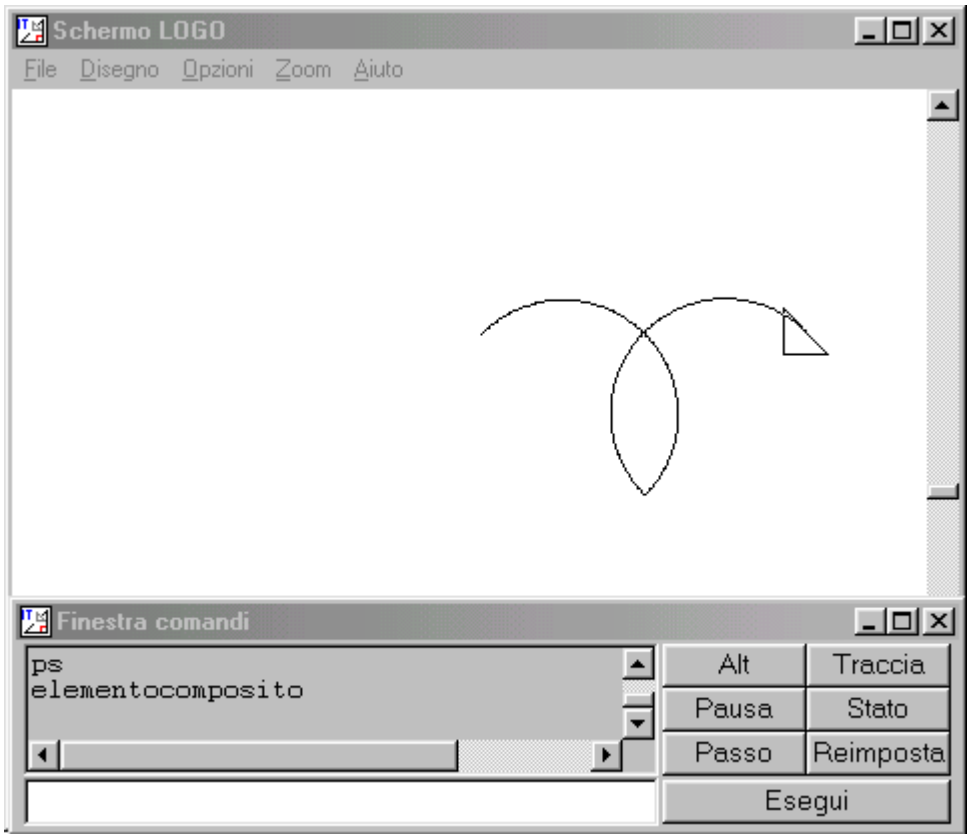
```
per semicirconferenza
          ripeti 180 [a  1 d  1]
fine
```



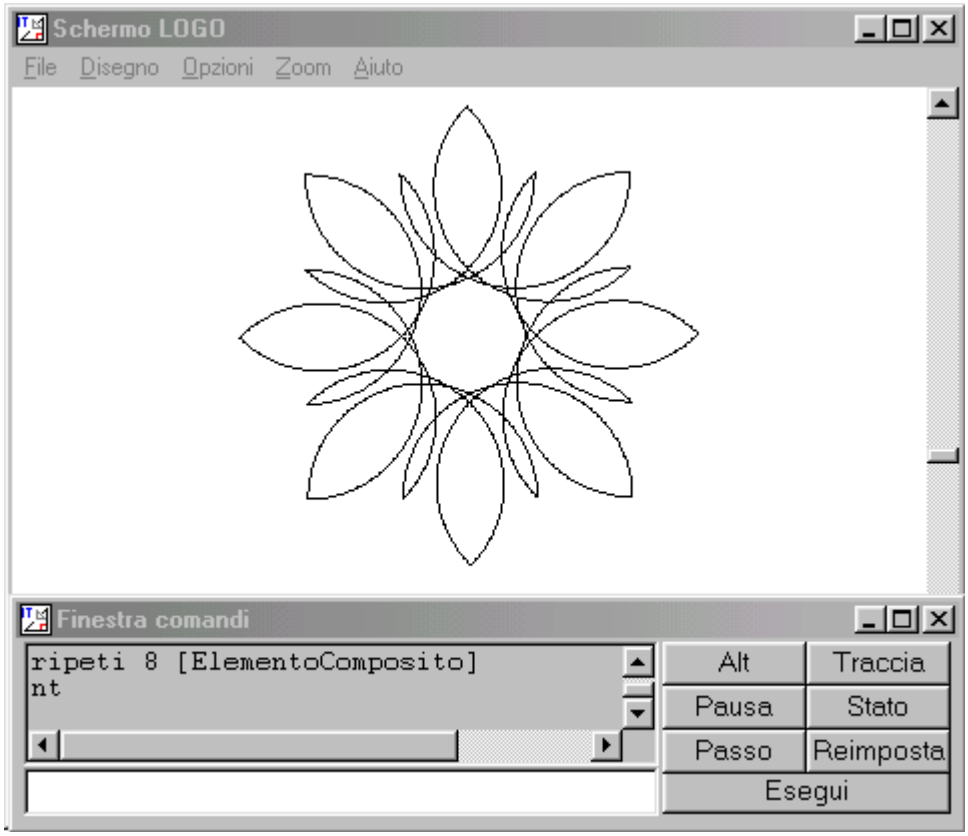
Digitiamo `semicirconferenza` sulla linea dei comandi e vediamo il risultato.

Definiamo poi la procedura `ElementoComposito`:

```
per ElementoComposito
          d 45 ripeti 2 [semicirconferenza d 90]
fine
```



Infine, ripetiamo otto volte la procedura ElementoComposito.



Se amate le sequenze complicate di istruzioni avreste potuto digitare direttamente

```
ripeti 8 [d 45 ripeti 2 [ripeti 180 [a 1 d 1] d 90]]
```

ma in questo modo è più difficile capire come si componga la figura.

Procedure con variabili

L'uso di variabili apre nuovi orizzonti alla programmazione Logo.

Come ricorderete la procedura per tracciare un quadrato di lato 100 era questa

```
per quadrato
      ripeti 4 [a 100 d 90]
fine
```

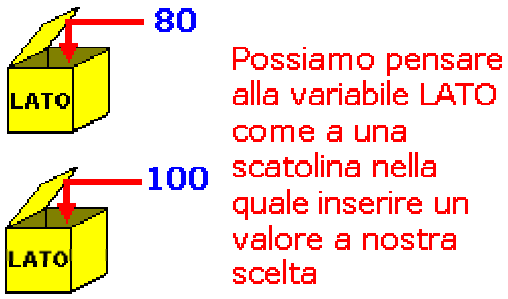
E se volessimo tracciare un quadrato di lato 80? Allo stato delle nostre conoscenze dovremmo scrivere una nuova procedura mettendo al posto del numero 100 il numero 80.

```
per quadrato
      ripeti 4 [a 80 d 90]
fine
```

Ma capite che questo modo di operare è decisamente poco pratico. Sarebbe bello disporre di una procedura per tracciare tutti i quadrati. Il Logo ci consente di farlo utilizzando una variabile. Ecco come si trasforma la nostra procedura

```
per quadrato :LATO
      ripeti 4 [a :LATO d 90]
fine
```

Come vedete questa volta entra in gioco un nuovo elemento: la variabile LATO. Quando noi scriviamo la procedura non specifichiamo quale sia l'avanzamento della tartaruga, cioè quale sia la lunghezza del lato, ma al posto di un numero mettiamo il simbolo :LATO. Probabilmente vi chiederete quale sia la funzione dei due punti che vanno messi davanti al nome della variabile senza lasciare uno spazio. E' molto semplice: LATO è il nome della variabile, :LATO rappresenta il valore che le sarà assegnato. Tutto sarà chiaro dopo aver esaminato qualche esempio.



Supponiamo allora di aver scritto nell'editor la procedura precedente e di averla salvata. Cosa digiteremo nel campo di input della finestra dei comandi per metterla in esecuzione? E' questo il momento per assegnare un valore numerico alla variabile. Se vogliamo un quadrato di lato 50 digiteremo

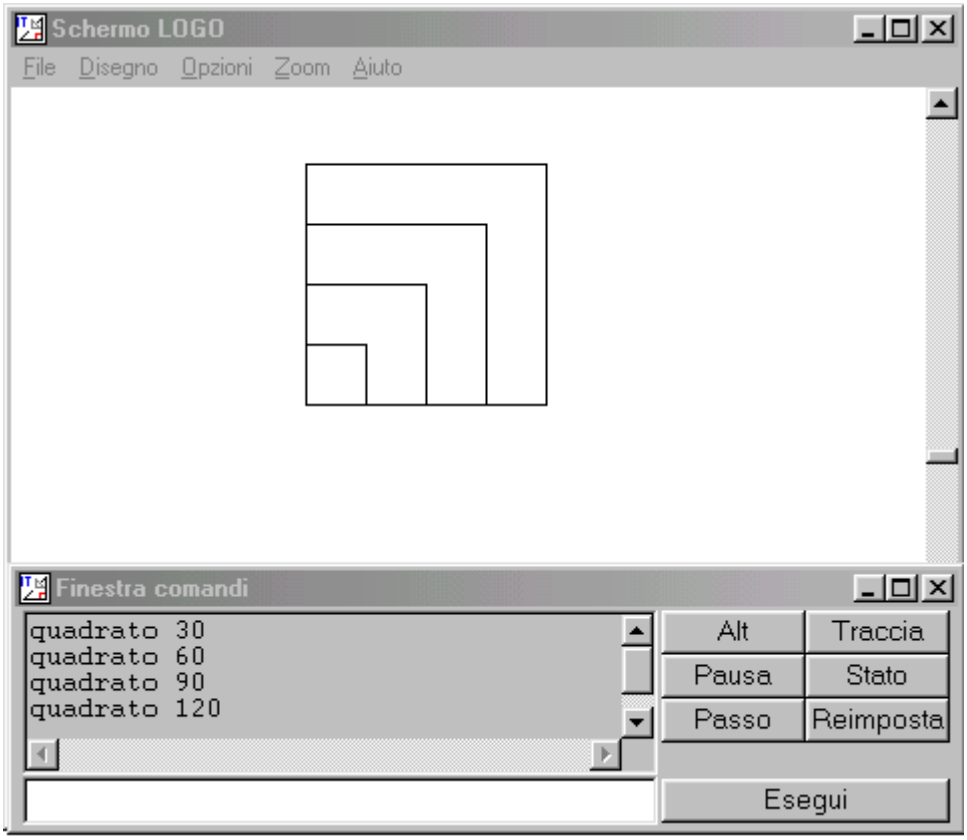
```
  quadrato 50
```

Se vogliamo un quadrato di lato 100 digiteremo

```
  quadrato 100
```

e così via.

La schermata seguente mostra come un'unica procedura quadrato :LATO possa generare quadrati di lato diverso.



Vi siete resi conto che la nuova procedura quadrato richiede la specificazione di un parametro (di una variabile) esattamente come accade per molte istruzioni primitive del Logo, ad esempio per l'istruzione `avanti`, che richiede di specificare di quanti passi la tartaruga deve avanzare. Nel caso della procedura quadrato dobbiamo specificare la misura del lato.

Sul piano sintattico tenete presente che la variabile LATO che viene utilizzata all'interno della procedura deve essere dichiarata subito dopo il nome della procedura

```
  per quadrato :LATO
```

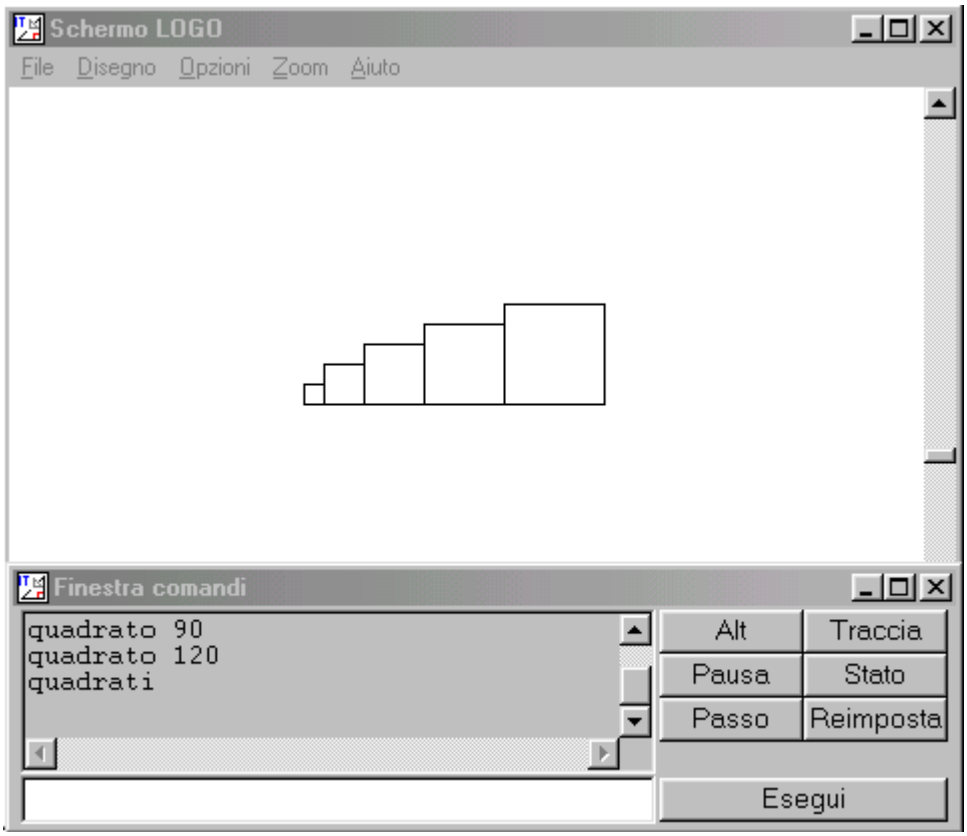
E' poi evidente che il nome della variabile, anziché LATO, può essere uno qualsiasi, ad esempio L o X.

Esaminate ora la procedura seguente al cui interno viene utilizzata più volte la procedura quadrato con diversi valori del parametro.

```
per quadrati
      ps
      nt
      quadrato 10
      d 90 a 10 s 90

      quadrato 20
      d 90 a 20 s 90

      quadrato 30
      d 90 a 30 s 90

      quadrato 40
```

```
      d 90 a 40 s 90

      quadrato 50
fine
```

Mettendola in esecuzione otterrete la schermata della figura seguente. Tenete presente che l'istruzione `nt` (nascondi tartaruga) ha l'effetto di non visualizzare il triangolino che rappresenta la tartaruga. Per poterlo rendere di nuovo visibile bisognerà dare l'istruzione `mt` (mostra tartaruga).



Ancora ripeti

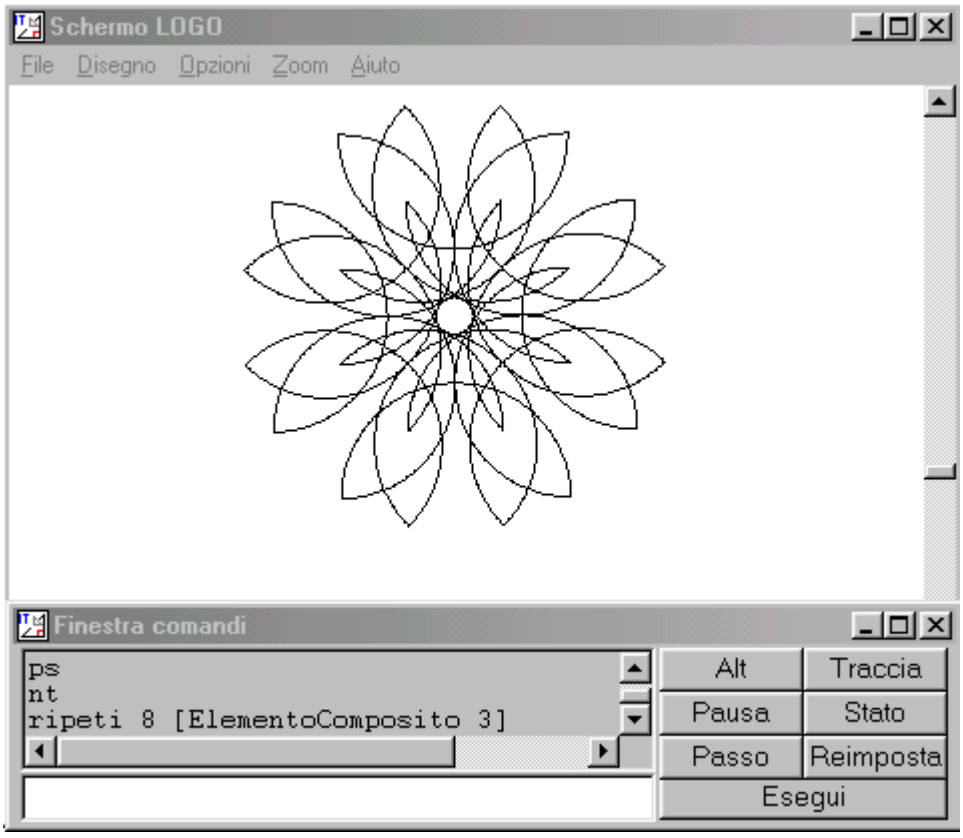
Conosciamo già le procedure `arco` ed `ElementoComposito` ed abbiamo visto come utilizzarle per disegnare una figura (`ripeti 8 [ElementoComposito]`).

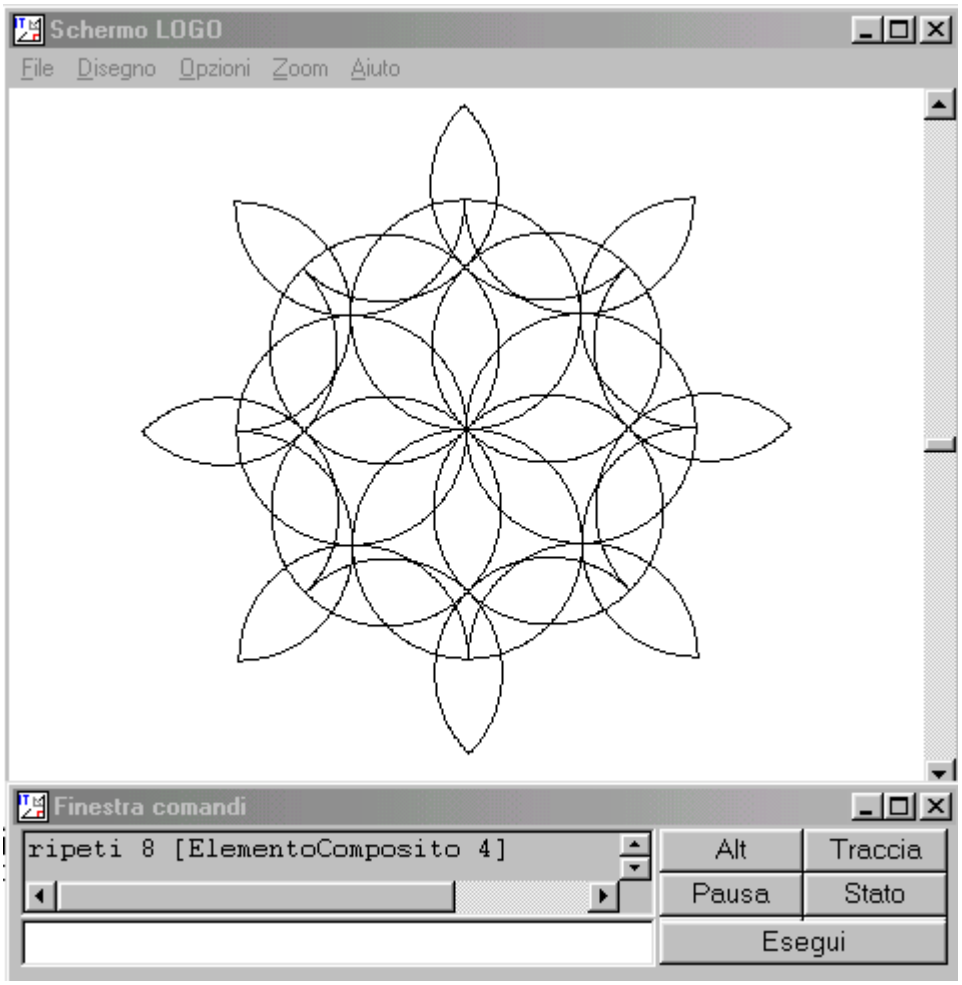
```
per semicirconferenza
          ripeti 180 [a  1 d  1]
fine

per ElementoComposito
          d 45 ripeti 2 [semicirconferenza d 90]
fine
```

Consideriamo questa modifica di `ElementoComposito`, in cui la nuova procedura individua una famiglia di elementi determinati dal parametro TIPO:

```
per ElementoComposito :TIPO
          d 45 ripeti :TIPO [arco d 90]
fine
```

Digitando `ripeti 8 [ElementoComposito 2]` otterremo la figura già vista. Proviamo con `ripeti 8 [ElementoComposito 3]` oppure `ripeti 8 [ElementoComposito 4]`.





Salvare e recuperare procedure

Dopo aver scritto una o più procedure nell'editor e dopo averle eseguite verrà il momento che dovremo uscire da MSW-Logo. Tenete presente che le procedure sono per il momento salvate in memoria (workspace) e non su disco, quindi in modo non permanente. Uscendo dall'ambiente le perderemmo irrimediabilmente. E' allora necessario salvare su disco il nostro lavoro. A questo scopo utilizzeremo in modo diretto il comando `salva`. Fate attenzione, con il comando `salva` salvate tutto il contenuto dell'editor, quindi scegliete un nome opportuno per il file che sarà creato su disco. Se ad esempio abbiamo definito nell'editor le due procedure

```
per arco
          ripeti 180 [a  1 d  1]
fine

per ElementoComposito :TIPO
          d 45 ripeti :TIPO [arco d 90]
fine
```

potremmo chiamare il file in cui salveremo il contenuto dell'editor "figure.lgi" (l'estensione .lgi identifica tutti i file del MSW-Logo). Ecco il comando da dare nel campo di input

```
salva "figure.lgi
```

(attenzione ad aprire ma non chiudere le virgolette). Se volessimo specificare un percorso per salvare il file in una ben precisa cartella (directory) utilizzeremo un comando di questo tipo

```
salva "c:/logoit/corso/figure.lgi
```

In questo caso il file sarà salvato nella cartella "corso" contenuta nella cartella "logoit" che si trova sotto la radice del disco c: . Fate attenzione ad usare la barra in avanti (/) oppure, per ragioni che vedremo più avanti, ad usare due barre indietro (\\).

Tenete inoltre presente che il comando `salva` scrive su disco anche i valori assegnati a tutte le variabili create con l'istruzione `assegna` che esamineremo più avanti.

Naturalmente, all'inizio di una nuova sessione di lavoro, dovremo poter gestire l'operazione inversa, cioè riportare nell'editor le procedure salvate su file. A tal fine utilizzeremo il comando `carica`. Riferendoci ai due esempi precedenti dovremo dare, nel primo caso, il comando

```
carica "figure.lgi
```

e, nel secondo caso, il comando

```
carica "c:/logoit/corso/figure.lgi
```

e ritroveremo nell'editor le nostre due procedure. Attenzione, nell'aiuto di MSW-Logo in italiano il comando `carica` viene riportato erroneamente come `apri`.

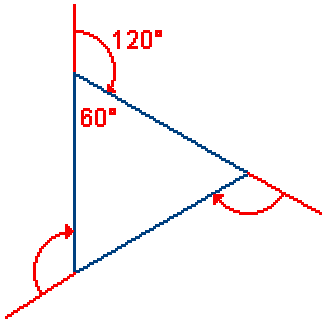
Fate attenzione, se avete definito nell'editor delle procedure e caricate un file di procedure mediante il comando `carica`, saranno sovrascritte le procedure che hanno lo stesso nome. Se ad esempio nell'editor avete definito le due procedure triangolo e quadrato e caricate un file che contiene le procedure rettangolo e quadrato, nell'editor troverete le tre procedure triangolo, quadrato e rettangolo ma la procedura quadrato sarà quella che avete salvato su file e non quella che avete digitato nell'editor.

Tenete infine presente che per salvare o recuperare procedure potrete anche utilizzare le voci "Salva", "Salva con nome" e "Apri" del menu "File" della finestra principale. Procedendo in questo modo sarà più facile individuare la cartella nella quale volete operare perché potrete selezionarla come fate normalmente nel caso delle finestre di Windows che ci consentono di accedere a risorse su disco.

Costruire poligoni regolari

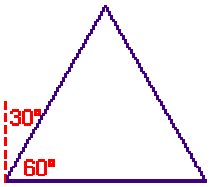
Cominciamo a tracciare con Logo un triangolo equilatero. La figura ci fa capire che la tartaruga deve compiere tre rotazioni di 120° (non dimenticando che gli angoli di rotazione della tartaruga sono angoli esterni).
Ecco allora la procedura:

```
per TriangoloEquilatero :LATO
      ripeti 3 [a :LATO d 120]
fine
```



Se volessimo ottenere un triangolo con un lato orizzontale dovremmo modificare la procedura nel modo seguente:

```
per TriangoloEquilatero :LATO
      d 30
      ripeti 3 [a :LATO d 120]
      s 30
fine
```



Ci rendiamo conto della necessità della prima istruzione `d 30`, che fa ruotare la tartaruga di 30° verso destra; alla fine per riorientare la tartaruga verso nord è opportuno dare l'istruzione `s 30` che la fa ruotare a sinistra di 30° (`s` è l'abbreviazione del comando `sinistra`).

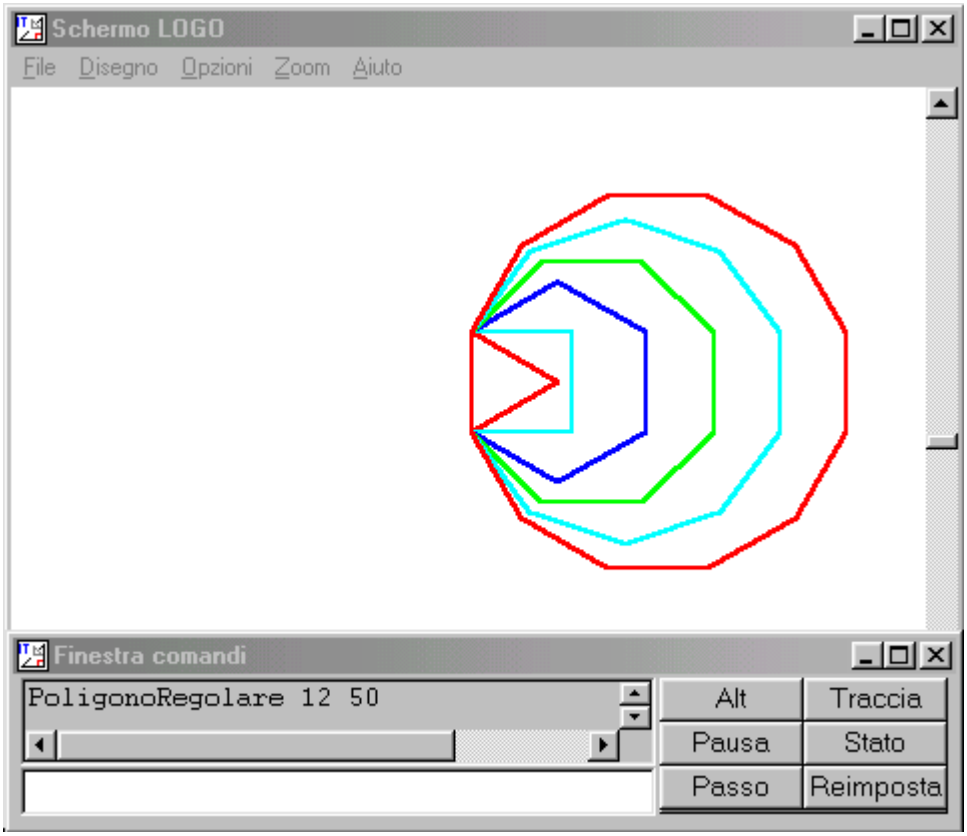
Sappiamo costruire, per il momento, due tipi di poligoni regolari: i quadrati e i triangoli equilateri. Ma se riflettiamo sulle relative procedure ci rendiamo conto di essere a un passo da una procedura più generale che ci consenta di generare qualsiasi poligono regolare. Infatti per tracciare un poligono regolare di n lati la tartaruga dovrà:

- Eseguire n avanzamenti tutti uguali (cioè di uno stesso numero di passi).
- Eseguire n rotazioni tutte uguali.

Poiché alla fine della costruzione la tartaruga deve ritornare con l'orientamento iniziale (all'inizio, se la tartaruga è orientata verso nord, alla fine del suo cammino è di nuovo orientata verso nord), la rotazione complessiva eseguita deve essere di un giro completo, cioè di 360°. Ma le rotazioni, si è detto, sono tutte uguali quindi ciascuna di esse è pari a 360/n gradi. E' allora facile scrivere la procedura, indicando con N il numero dei lati e con LATO la misura del lato del poligono regolare:

```
per PoligonoRegolare :N  :LATO
      ripeti :N [a :LATO  d 360/:N]
fine
```

Nella schermata seguente vedete alcuni poligoni regolari tracciati con questa procedura (dal triangolo equilatero al dodecagono regolare).

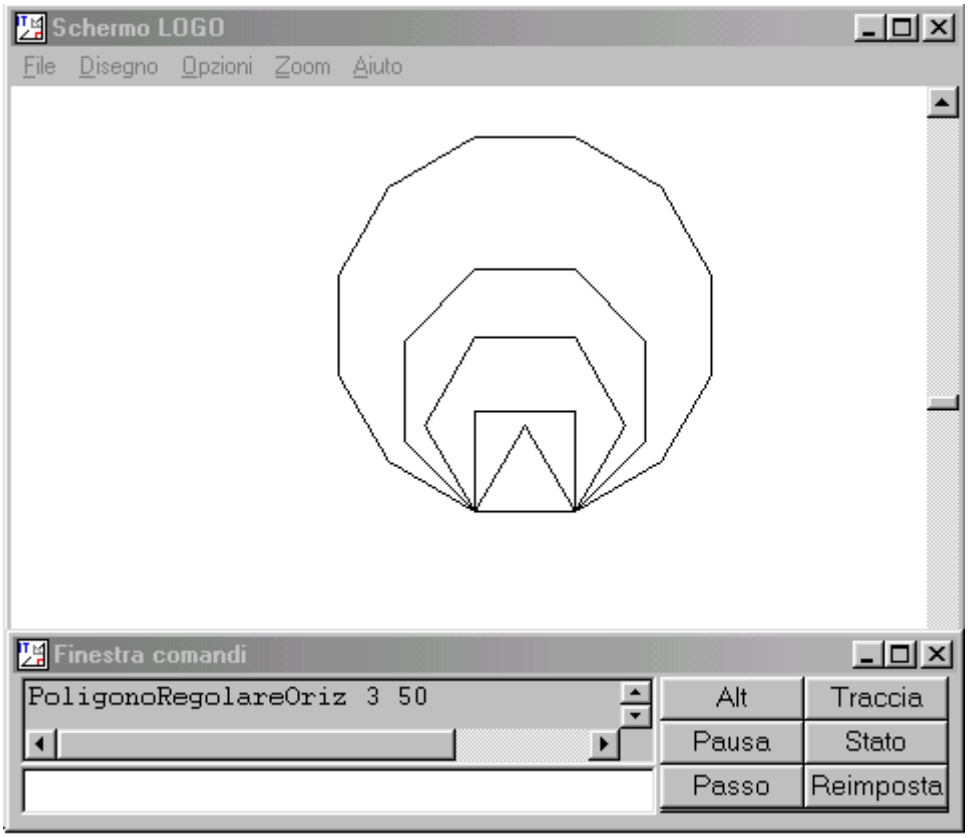


Come vedete i poligoni sono di colori diversi; per ottenere questi effetti cromatici non dovete far altro che utilizzare l'opzione "Colore Penna" che trovate nel menu "Opzioni" della finestra principale. Questa opzione vi consente di scegliere tra 8 colori di base oppure di definire un colore personalizzato su oltre16 milioni di sfumature possibili. Il colore selezionato verrà assunto per tutte le figure tracciate successivamente. Naturalmente potremo impostare un colore anche mediante un'istruzione ma di questo ci occuperemo tra non molto. Tenete presente che mediante le opzioni "Colore Schermo" e "Colore Riempimento" potete cambiare anche il colore dello sfondo del mondo della tartaruga (che per il momento è bianco) e il colore del "riempimento" di una figura (ce ne occuperemo più avanti).

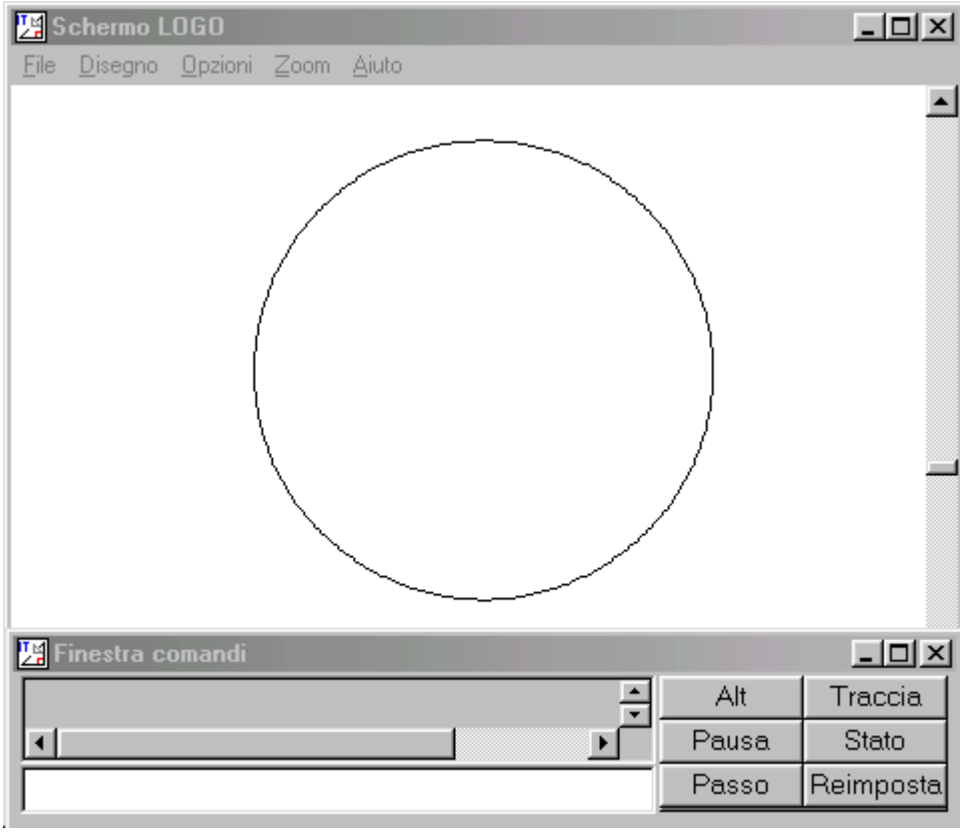
E' anche molto facile scrivere la procedura per ottenere poligoni regolari con la base orizzontale. Nel caso del triangolo equilatero dovevamo operare una rotazione iniziale di 90-a gradi dove a è l'angolo interno del triangolo. La stessa cosa dovremo fare per un generico poligono regolare, tenendo presente che in questo caso a=180-360/n. Ecco la procedura:

```
per PoligonoRegolareOriz :N  :LATO
      d 90-(180-360/:n)
      ripeti :N [a :LATO  d 360/:N]
      s 90-(180-360/:n)
fine
```

Nella schermata seguente vedete alcuni poligoni generati con questa nuova procedura.



Quest'ultima schermata mostra un poligono regolare di 360 lati: come vedete è indistinguibile da una circonferenza.



La procedura PoligonoRegolare :N :LATO ci ha consentito di toccare e di mettere a fuoco tre questioni importanti.

a) Abbiamo potuto riflettere sulle proprietà angolari dei poligoni regolari. L'idea di fondo è molto semplice: per tracciare un cammino chiuso la tartaruga deve ritornare nel punto di partenza con l'orientamento iniziale e ciò avviene quando la rotazione complessiva, cioè la somma di tutte le rotazioni eseguite, è di 360°. Nel caso dei poligoni regolari ciò implica che gli angoli esterni, che sono uguali, siano di 360/n gradi. Ne segue che gli angoli interni sono di 180-360/n gradi.

b) Ci siamo resi conto che la costruzione di un poligono regolare dipende da due soli parametri (variabili): il numero dei lati e la lunghezza del lato. Se fisso il numero dei lati e la misura del lato, posso costruire un unico poligono regolare (non tenendo conto, ovviamente, della sua disposizione nel piano). Se fisso il numero dei lati e faccio variare la misura del lato, ottengo poligoni più o meno grandi ma tutti della stessa forma (simili).

c) Ci siamo resi conto che un poligono regolare con un gran numero di lati (e non troppo esteso) appare sullo schermo indistinguibile da una circonferenza. Ciò suggerisce l'idea che la circonferenza possa essere approssimata mediante poligoni regolari e avvicina i ragazzi, in modo intuitivo, alla nozione di limite.

Colori ed animazioni

Abbiamo già visto come si possa cambiare il colore della traccia lasciata dalla tartaruga mediante un'opzione del menù "Opzioni" della finestra principale. Ma in molte occasioni dovremo poter gestire i colori mediante istruzioni poste all'interno delle nostre procedure. Ce ne occuperemo adesso, vedendo come realizzare delle semplici animazioni.

L'MSW-Logo è in grado di gestire un numero di colori pari a 16.777.216. Ogni colore viene individuato mediante una lista di tre numeri interi: il primo numero della lista è la componente rossa, il secondo la componente verde, il terzo la componente blu. Ciascuna componente può variare tra 0 e 255. La procedura seguente, ad esempio, traccia un quadrato in rosso.

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

L'istruzione `ascp` (assegna il colore della penna) imposta il colore della traccia lasciata dalla tartaruga; in questo caso il primo numero della lista in input è il valore massimo 255 (componente rossa), mentre gli altri due valori sono nulli (componente verde e blu). Ecco perchè la traccia sarà rossa (rosso puro, componente rossa al 100%). Se il primo valore fosse più basso, ad esempio 200, avremmo un rosso più scuro. L'istruzione `ascp` [0 255 0] imposterebbe invece il verde puro (solo componente verde al 100%) e l'istruzione `ascp` [0 0 255] il blu puro (solo componente blu al 100%). Naturalmente potremo mescolare le tre componenti in diverse percentuali ottenendo tutti i colori possibili; ecco ad esempio delle terne di valori con i relativi colori:

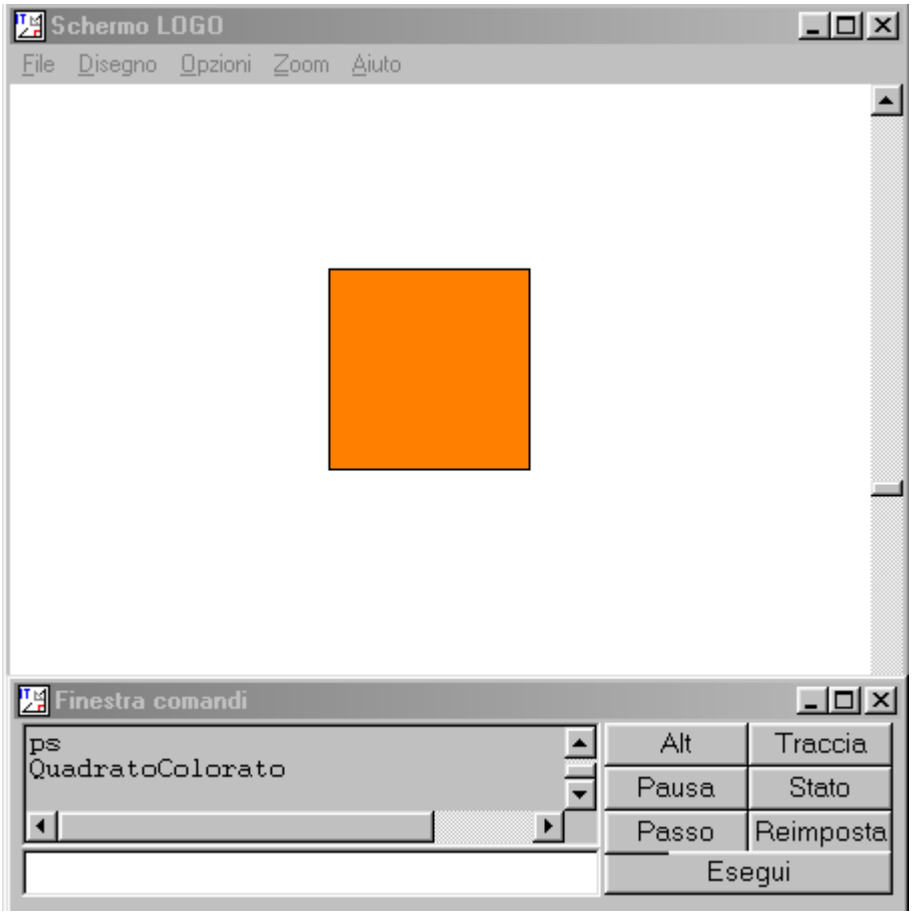
```
[0 0 0], nero
[255 255 255], bianco
[255 0 0], rosso
[0 255 0], verde
[0 0 255], blu
[0 0 140], blu scuro
[255 255 0], giallo
[0 255 255], azzurro
[255 0 255], viola
[162 123 91], marrone
[255 128 0], arancione
[128 128 128], grigio
```

Provate a definire qualche colore: avete a disposizione 256 scelte per ogni componente, per un totale di 256^3 colori !

Esaminate ora questa procedura che contiene alcune nuove istruzioni

```
per QuadratoColorato
     ps nt
     ascp [0 0 0]
     ascr [255 128 0]
     ripeti 4 [a 100 d 90]
     su
         d 45
         a 5
         riempi
     giu
fine
```

La schermata seguente ne mostra l'effetto.



Qui oltre all'istruzione `ascp` è presente anche l'istruzione `ascr` (assegna colore riempimento) che imposta il colore del riempimento delle figure. Nel nostro caso il colore della traccia è nero (contorno della figura) mentre il riempimento è arancione.

Ma come abbiamo fatto a "riempire" la figura con un colore? E' molto semplice: dobbiamo portare la tartaruga in un punto qualsiasi che sia interno alla figura e poi dare l'istruzione `riempi`. Attenzione però, la tartaruga deve arrivare all'interno della figura senza lasciare la traccia. A questo scopo abbiamo utilizzato l'istruzione `su` (penna su) che impedisce alla tartaruga di lasciare la traccia durante i suoi spostamenti. L'istruzione `giu` (penna giù) rimette poi le cose a posto, ripristinando la traccia. Allora il frammento della procedura precedente

Un'altra istruzione per la gestione dei colori che potrà farci comodo è `ascs` (assegna colore schermo) e serve a impostare il colore dello sfondo. `ascp` , `ascr e ascs` accettano come parametri una lista di tre numeri. Delle liste parleremo in seguito, per ora pensiamole come dei numeri (le tre componenti del colore) compresi tra parentesi quadre. Esaminiamo ora una procedura che ci consente di animare la rotazione di un quadrato

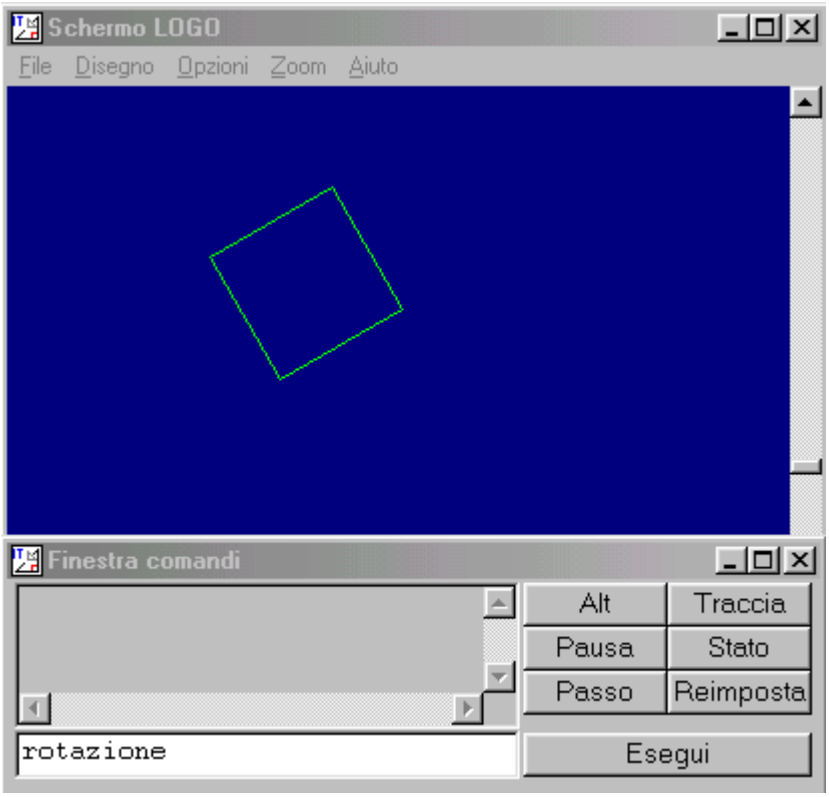
```
per quadrato
ripeti 4 [a 70 d 90]
fine

per rotazione
ps nt  ascs [0 0 130]
ripeti 36 [ ascp [0 255 0]
            quadrato
            aspetta 5
            ascp [0 0 130]
            quadrato
            d 10]
ascp [0 255 0]
mt
fine
```

Il meccanismo che sta alla base dell'animazione è molto semplice:

- si imposta il colore per la traccia (nel nostro caso verde)
- si traccia il quadrato
- si crea un breve pausa (istruzione `aspetta`)
- si imposta per la traccia lo stesso colore dello sfondo (nel nostro caso blu scuro)
- si ritraccia il quadrato (in realtà si cancella il quadrato precedente dato il colore impostato)
- si ruota di 10 gradi.
-

Si ripete questa stessa sequenza per 36 volte (o per più volte se non si vuole che la rotazione termini dopo un giro).

Ecco cosa vedete eseguento la procedura.



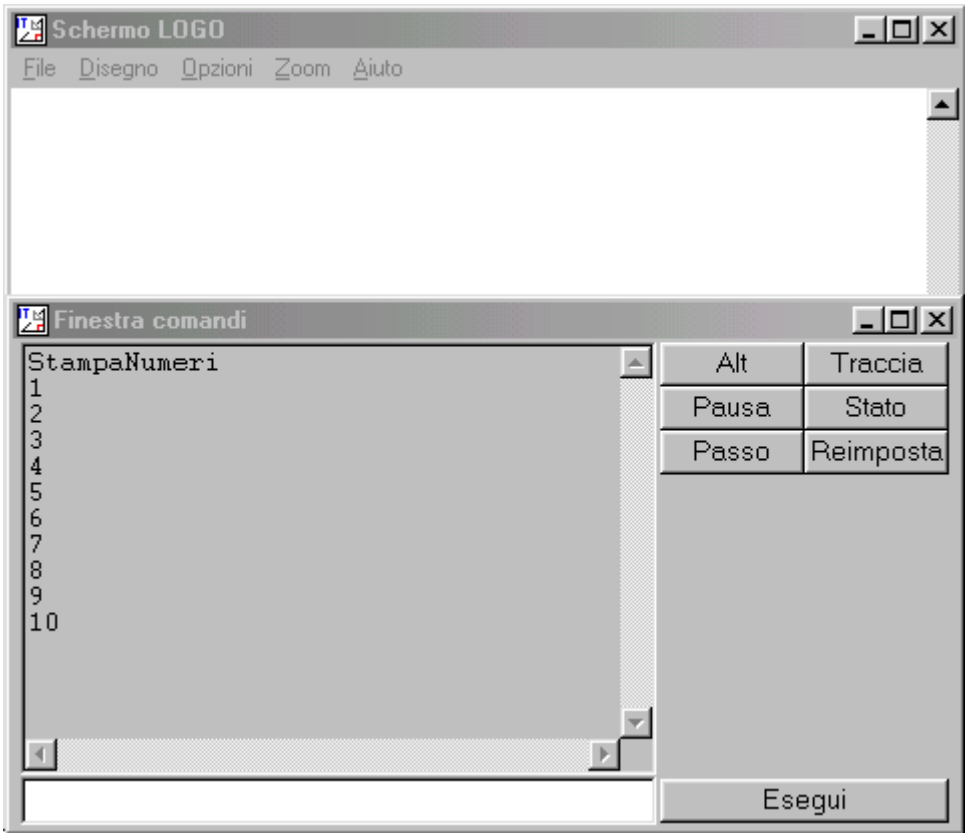
Procedure di tipo aritmetico

Sarebbe un errore pensare che il raggio d'azione del Logo sia limitato alla costruzione di figure. MSW-Logo è un linguaggio di programmazione completo che consente di realizzare procedure di qualsiasi tipo. In questo numero ne vedremo alcune di tipo aritmetico.

La prima che esamineremo ha lo scopo di visualizzare i numeri da 1 a 10:

```
per StampaNumeri
      ripeti 10 [stampa iterazioni]
fine
```

Prima di spiegarne il funzionamento, vediamone l'effetto. A tal fine ridimensioniamo (ingrandiamo) la finestra dei comandi trascinando in modo opportuno, col mouse, i suoi bordi (come facciamo per qualsiasi finestra di Windows). Diamo poi in modo diretto l'istruzione `pt` (abbreviazione dell'istruzione `puliscitesto`) che cancella tutto il testo che fosse presente nella finestra comandi. Digitiamo infine StampaNumeri. Ecco quello che vedremo.



Nella procedura troviamo due nuove istruzioni. L'istruzione `stampa` serve a visualizzare dati nella finestra comandi. Nel nostro caso servirà a visualizzare i numeri restituiti dall'istruzione `iterazioni`. Questa istruzione può essere usata solo all'interno di un comando `ripeti` e restituisce il numero di ripetizioni che sono state fatte, compresa quella corrente. Cioè, essa restituisce 1 la prima volta, 2 la seconda volta, e così via.

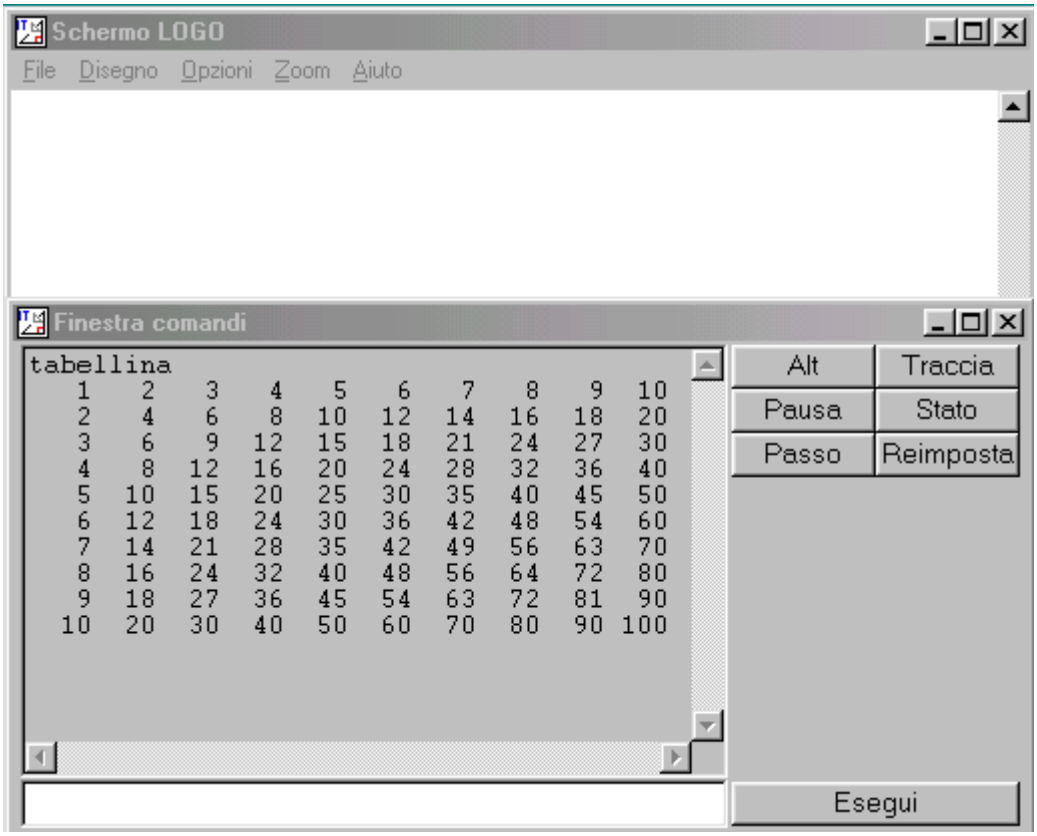
Pertanto `stampa` visualizzerà nella finestra comandi prima 1, poi 2, sino a 10.

Possiamo stampare, se lo desideriamo, quanti numeri vogliamo con una semplice modifica della procedura:

```
per StampaNumeri :N
      ripeti :N [stampa iterazioni]
fine
```

La procedura tabellina fa quello che ci aspettiamo che faccia: stampa una tabella pitagorica.

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



Nonostante l'obiettivo modesto, questa procedura ci permette di esplorare alcune nuove istruzioni di Logo.

Nella procedura trovate la nuova istruzione `assegna` che serve ad assegnare un valore ad una variabile. Inizialmente viene assegnato il valore 1 alla variabile r (alla variabile che ha per nome r); poi 2, 3, … sino a 10. Il nome della variabile deve essere qui preceduto dalle virgolette; attenzione, non confondete il nome di una variabile col valore di una variabile che deve essere preceduto dai due punti.

La procedura `scrivi` scrive il suo argomento nella finestra dei comandi e, a differenza di stampa, senza ritornare a capo; `stampa "` è necessario perciò per ritornare a capo alla fine di una riga.

I numeri sono stati allineati verticalmente con `formato :r*iterazioni 4 0. Formato` accetta tre numeri per argomenti: il primo è il valore che restituisce alla procedura `stampa` o `scrivi` (in questo caso è il prodotto di r per il numero restituito da `iterazioni`), il secondo la larghezza della colonna (4 caratteri in questo caso), il terzo il numero dei decimali (0).

Divisori e numeri primi

La procedura di cui ci occuperemo ora è più interessante e serve a determinare i divisori di un numero dato.

Prima di affrontarla è però opportuno esaminare un importante comando di tipo aritmetico, il comando `resto`, che ci farà comodo in molte situazioni. Il comando `resto` accetta in entrata due parametri, i numeri interi m e n, e restituisce il resto della divisione di m per n. Provate ad esempio a dare in modo diretto il comando

```
stampa resto 10 3
```

L'output visualizzato nella finestra comandi sarà 1.

Ed ecco la procedura per determinare i divisori del numero n

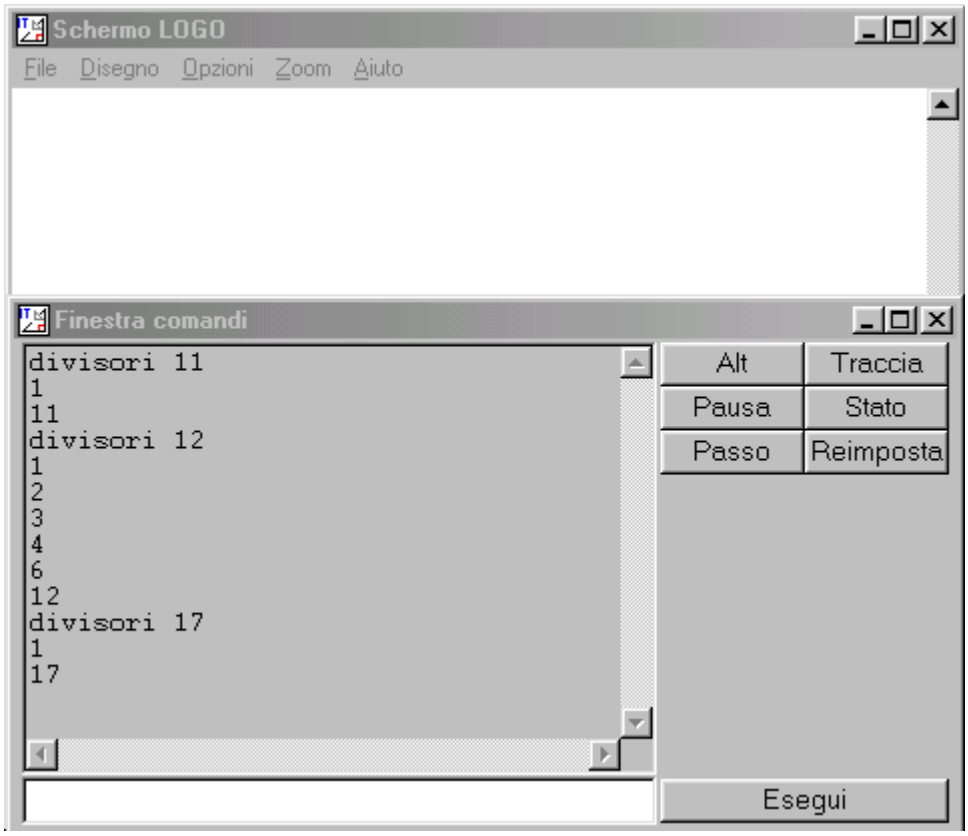
```
per divisori :n
       ripeti :n [se (resto  :n iterazioni) = 0 [stampa iterazioni]]
fine
```

Esaminiamo la procedura. Il numero di cui si vogliono trovare i divisori è n. Nell'istruzione `ripeti` l'istruzione `iterazioni` assume tutti i valori compresi tra 1 ed n. Il blocco da ripetere è costituito dall'unica istruzione

```
      se (resto  :n iterazioni) = 0 [stampa iterazioni]
```

Si tratta della nuova istruzione `se`, di fondamentale importanza. Traduciamone in parole il significato:
SE il resto della divisione di n per il numero restituito da `iterazioni` è 0 ALLORA visualizza il numero restituito da `iterazioni`.

Ora attenzione, poiché l'istruzione `se` è interna a `ripeti`, saranno visualizzati i valori restituiti da `iterazioni`, tra 1 e n, che sono divisori di n. Proprio quel che volevamo. Deve esservi chiaro che l'istruzione `se` sarà eseguita n volte, con il numero restituito da iterazioni che varia da 1 a n. Mettiamo in esecuzione la nostra procedura. Digitiamo divisori 11, divisori 12 e poi divisori 17.

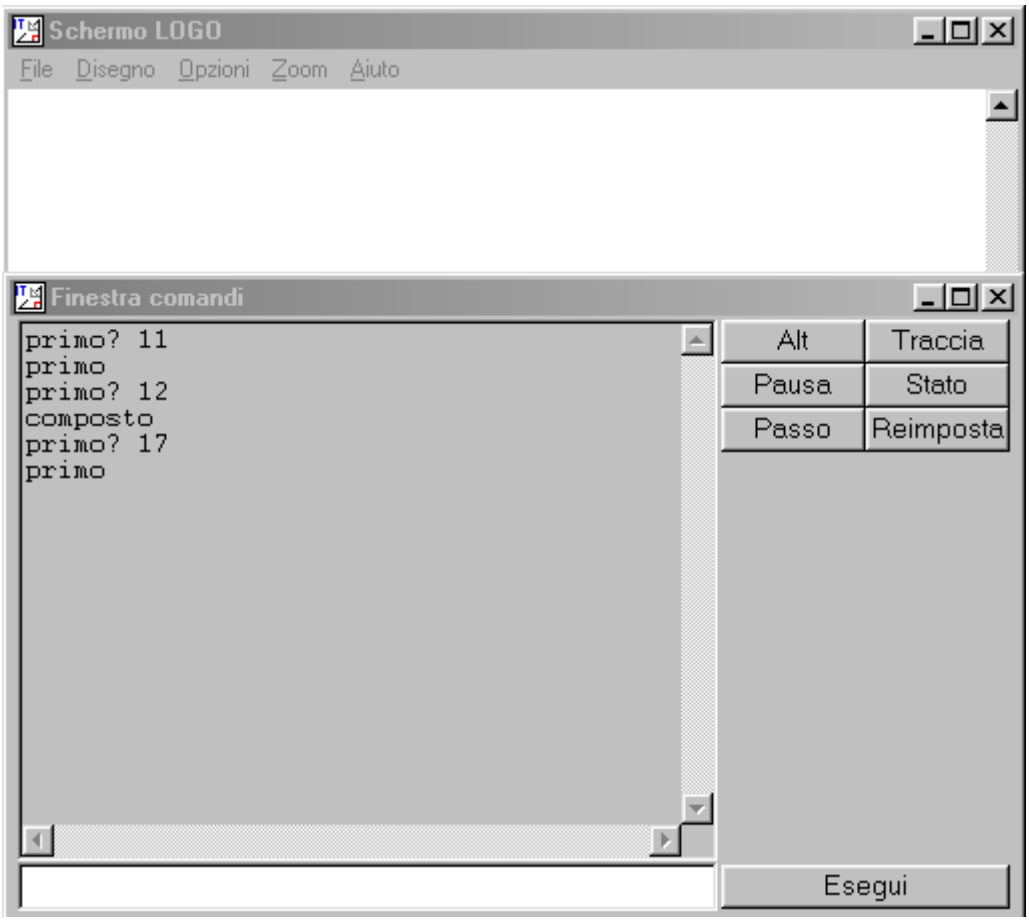


Ecco un'ultima procedura, per decidere se un numero n è primo, cioè se è divisibile solo per se stesso e per uno, oppure è composto.

```
per primo? :n
      se (:n = 1) [stampa [il numero deve essere maggiore di uno] stop]
      ripeti :n-2 [se (resto  :n iterazioni+1) = 0 [stampa [composto] stop]]
      stampa [primo]
fine
```

Tenete presente che l'istruzione `stop` ha lo scopo di interrompere l'esecuzione della procedura.
L'istruzione stop gioca qui un ruolo essenziale: provate a toglierla in uno dei due punti in cui compare e riflettete su quel che succede. Notate inoltre che l'istruzione `stampa` visualizza, in questo caso, il testo contenuto tra parentesi quadre.

Proviamo la procedura `primo?` con 11, 12 e 17, i tre numeri provati con la procedura `divisori`.

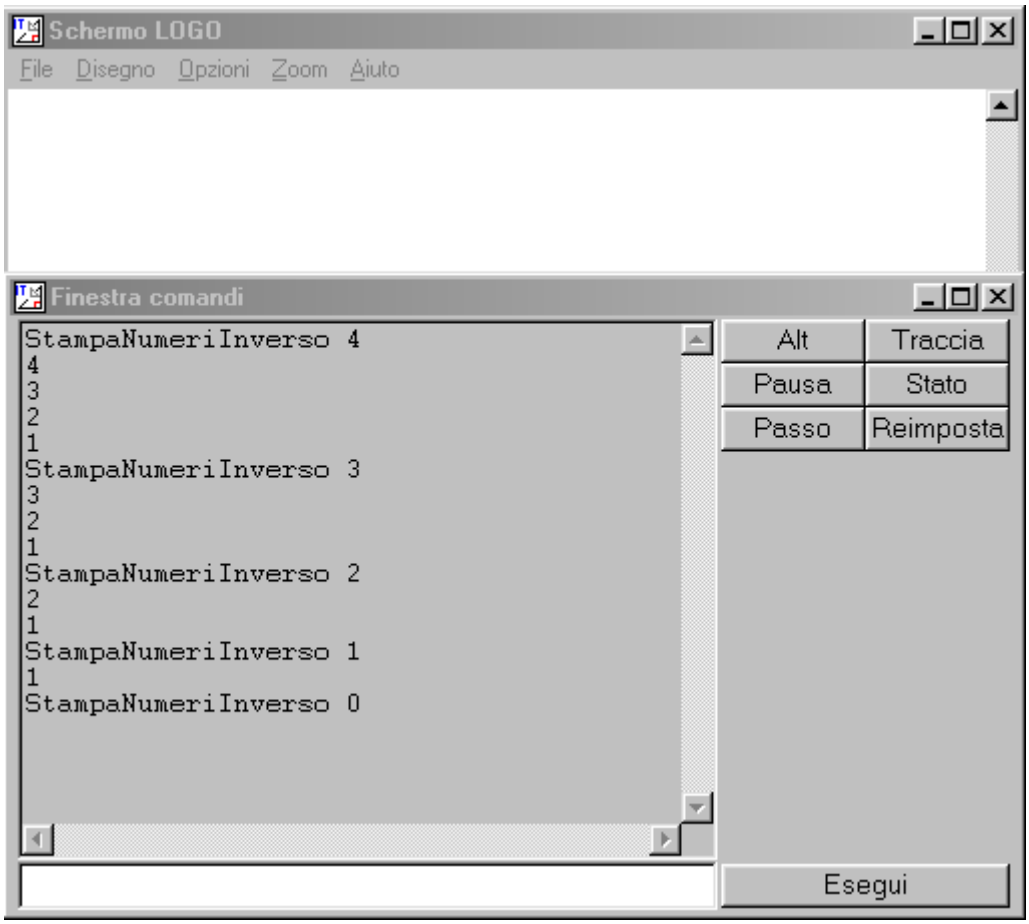


Ricorsione

Ci occuperemo ora di una versione modificata della procedura `StampaNumeri` esaminata in precedenza che ci consentirà di capire una tecnica importante della programmazione Logo. Ecco la nuova procedura (che produce l'inverso di quella già esaminata, visualizza cioè i numeri da 10 a 1) .

```
per StampaNumeriInverso :N
      se :N = 0 [stop]
      stampa :N
      StampaNumeriInverso :N-1
fine
```

Prima di spiegarne il funzionamento, vediamone il funzionamento con N uguale a 4, 3, 2, 1 e 0.



Consideriamo per semplicità il caso di `StampaNumeriInverso 2`. La procedura inizialmente verifica se N è zero. Poiché non è 0, stampa il valore 2. Dopo di che la procedura non fa altro che chiamare la procedura `StampaNumeriInverso :N`–1, cioè se stessa con l'argomento iniziale diminuito di uno. Chiama perciò `StampaNumeriInverso 1`. Anche in questo caso N è diverso da 0 e perciò la procedura stampa 1. Chiama infine `StampaNumeriInverso 0`. Adesso N è uguale a 0 e la procedura si interrompe.

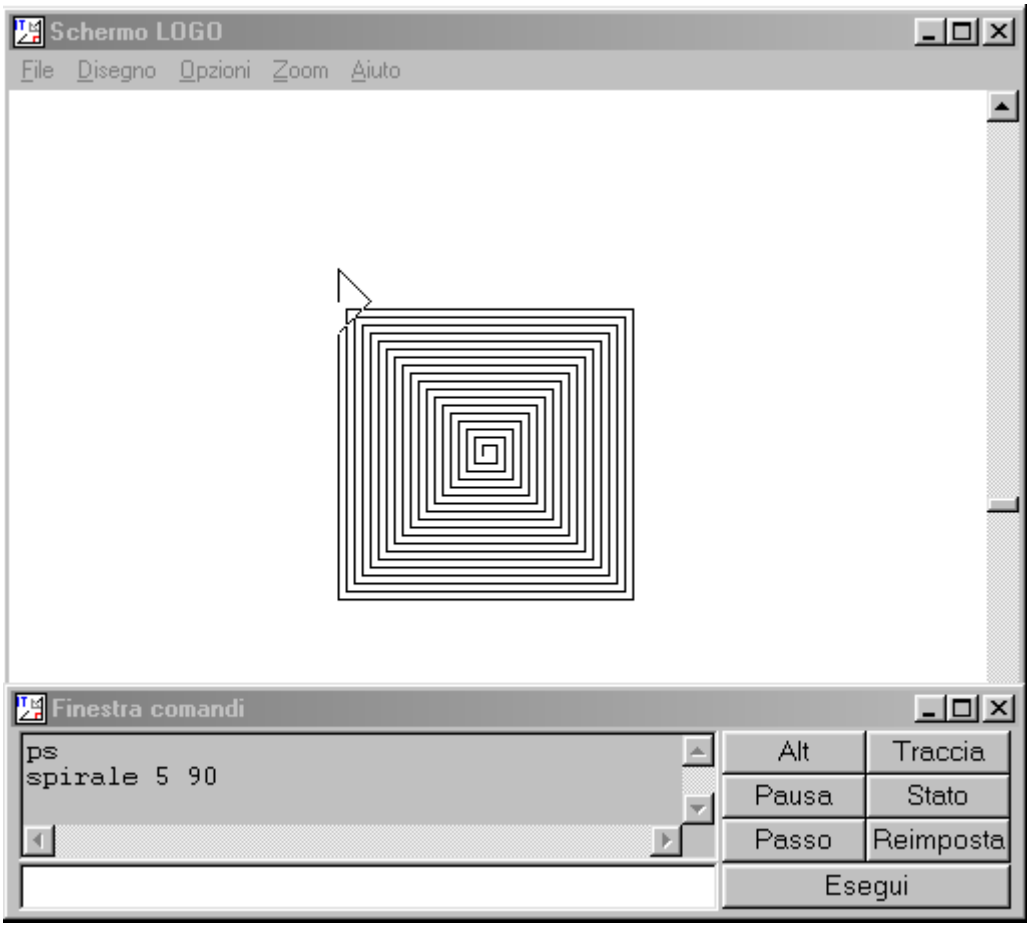
Sapete cosa succede se digitiamo `StampaNumeriInverso -2`?

Procedure geometriche

Esaminiamo ora un'elegante procedura ricorsiva di tipo geometrico in cui entra di nuovo in gioco il successivo incremento di una variabile:

```
per spirale :LATO :ANGOLO
      se :LATO > 150 [stop]
      a :LATO
      d :ANGOLO
      spirale :LATO+2 :ANGOLO
fine
```

Nella schermata seguente vediamo la spirale generata chiamando `spirale 5 90`.

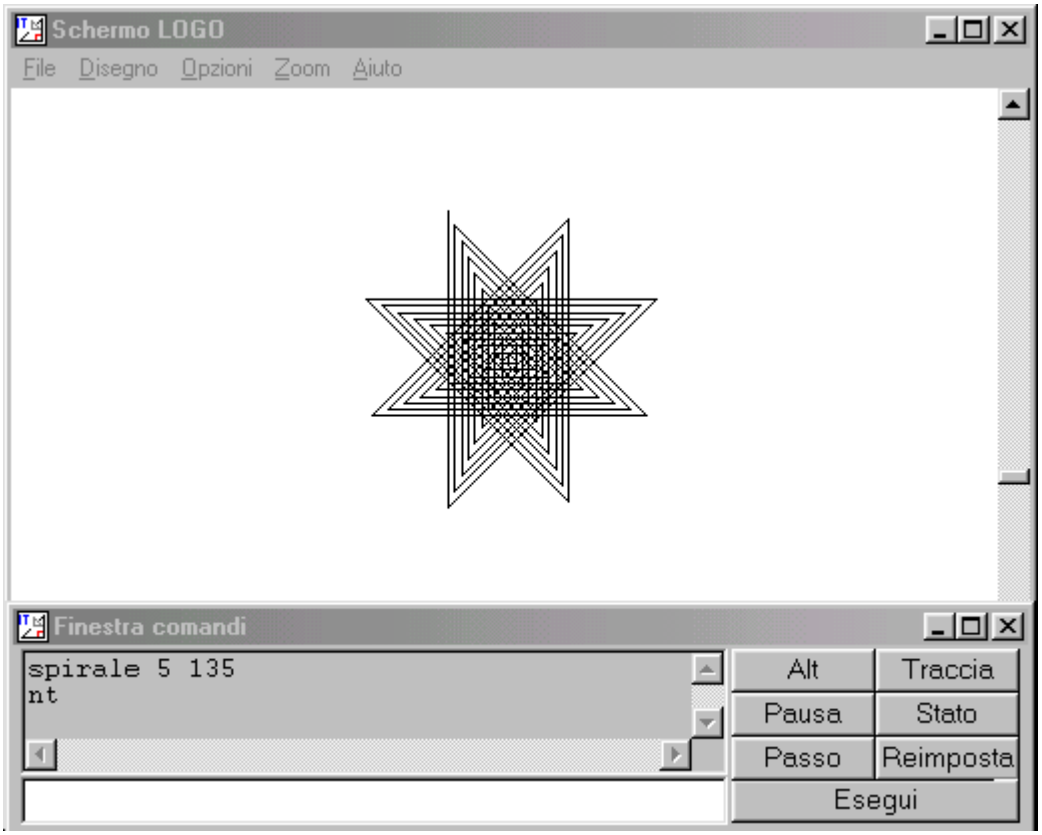


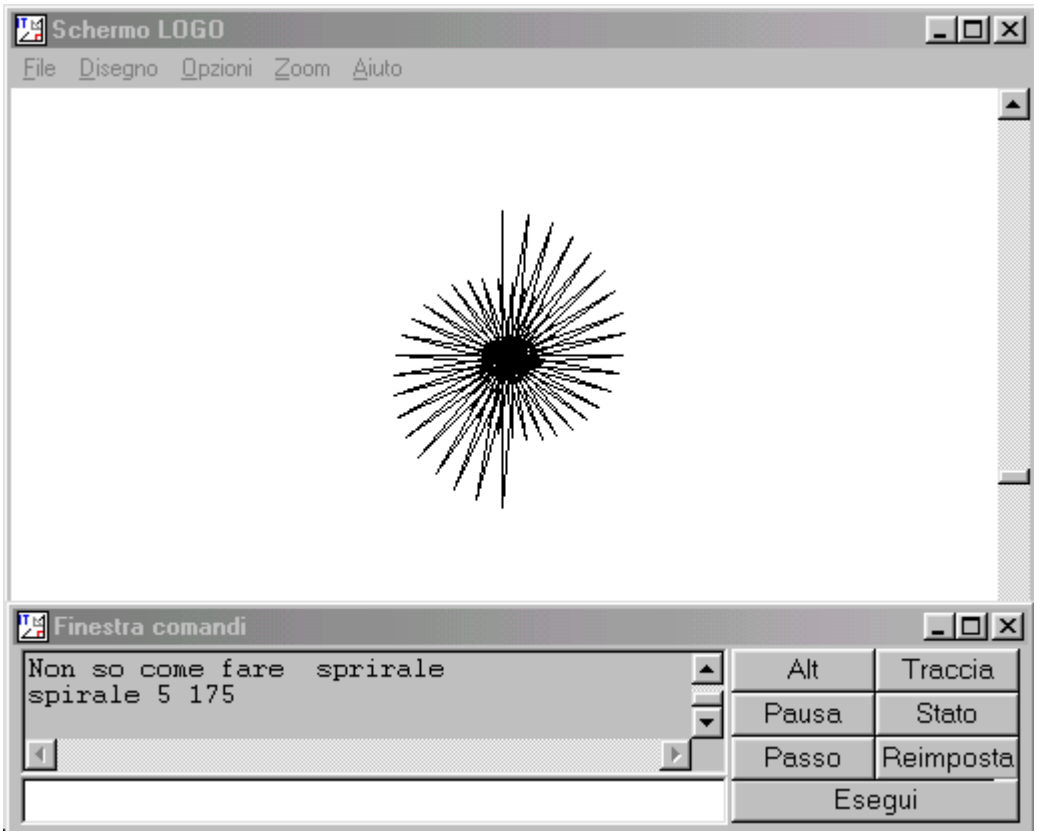
Cerchiamo di capirne il funzionamento. Le rotazioni eseguite sono tutte di 90 gradi, gli avanzamenti vanno invece via via aumentando. Si entra per la prima volta nella procedura con un valore di LATO uguale a 5, quindi il primo avanzamento è di 5 passi. Nella chiamata ricorsiva alla variabile LATO si assegna il valore attuale di LATO più 2

```
spirale :LATO+2 :ANGOLO
```

Quindi il secondo avanzamento sarà di 7 passi. E poi si capisce che il terzo avanzamento sarà di 7+2=9 passi e così via. Il processo termina quando il valore di LATO supera 150.

Di seguito sono riportate due combinazioni di valori, `spirale 5 135` e `spirale 5 175`; provate altre combinazioni di parametri, la varietà di forme che si possono creare vi stupirà.

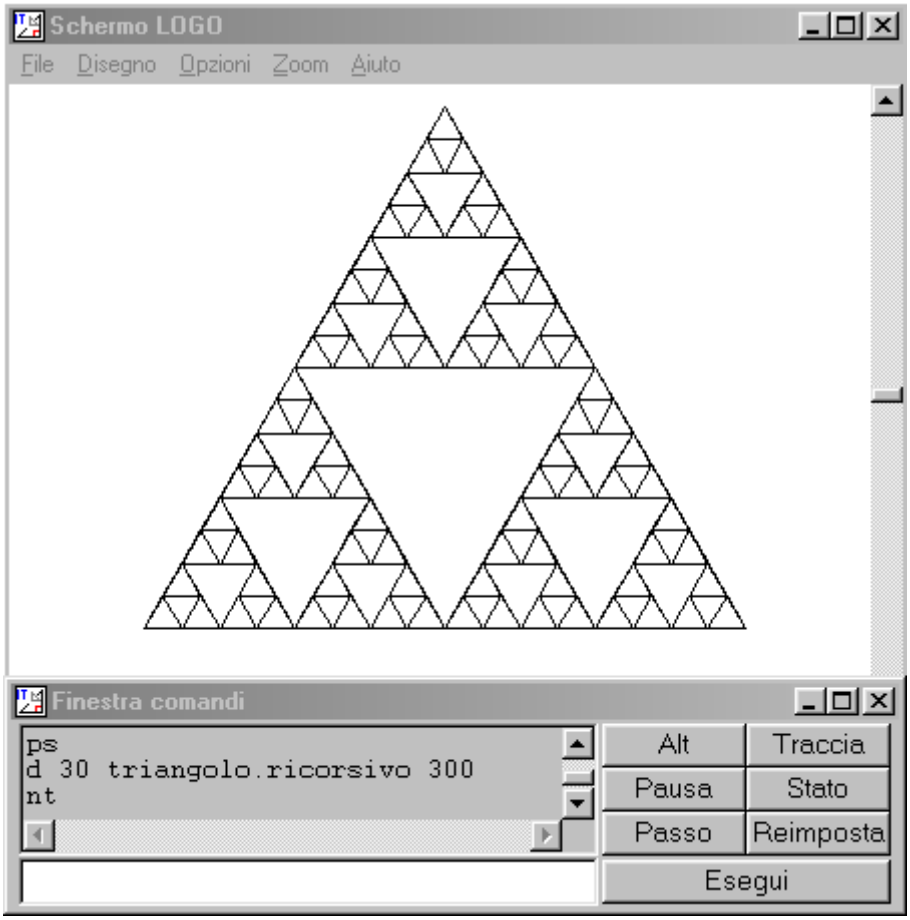

Schermo LOGO
File Disegno Opzioni Zoom Aiuto
Finestra comandi
spirale 5 135
nt
Alt
Traccia
Pausa
Stato
Passo
Reimposta
Esegui


Schermo LOGO
File Disegno Opzioni Zoom Aiuto
Finestra comandi
Non so come fare sprirale
spirale 5 175
Alt
Traccia
Pausa
Stato
Passo
Reimposta
Esegui

Di seguito riportiamo in'ulteriore procedura ricorsiva di tipo geometrico basata sulla composizione di triangoli.

```
per triangolo.ricorsivo :dim
      se :dim < 10 [stop]
      ripeti 3 [
      a :dim
      d 120
      triangolo.ricorsivo :dim/2
      ]
fine
```

Anche in questo caso la procedura traccia un disegno complesso e sorprendente.



Il capitolo delle procedure geometriche ricorsive è molto ampio. Nella sezione "risorse Logo" vengono riportate delle indicazioni utili per che voglia approfondire l'argomento.

Liste

La ricorsione, oltre che in applicazioni geometriche ed aritmetiche, viene utilizzata per la manipolazione delle liste, che sono degli insiemi ordinati di oggetti.

In Logo le liste sono delimitate da due parentesi quadre. Abbiamo già incontrato delle liste: è una lista, ad esempio, il blocco di istruzioni presente nell'istruzione `ripeti`.

Questa è una lista:

```
[Logo è  divertente]
```

e anche questa:

```
[Logo [non è] un giocattolo].
```

Questa è una lista vuota, cioè una lista che non contiene alcun elemento:

```
[]
```

Possiamo poi assegnare un nome ad una lista, di modo ché avremo una variabile che conterrà i valori di questa lista.

```
assegna  "l [Logo è  divertente]
```

Di una lista possiamo voler sapere il numero dei suoi elementi. Useremo la procedura primitiva di Logo `conta:`

```
mostra conta :l
```

ottenendo in risposta il valore 3. Quanti elementi ha la lista `[Logo [non è] un giocattolo]` ?

Per stampare il primo elemento della lista `:l` digito: `mostra primo :l` e ottengo in risposta `Logo`. Per vedere la lista eccetto il suo primo elemento, digito `mostra menpri :l`, ottenendo `[è  divertente].`

Vogliamo ora stampare tutti gli elementi di :l, uno per riga. A questo fine scrivo la seguente procedura:

```
per StampaLista :l
      se :l = [] [stop]
      stampa primo :l
      StampaLista menpri :l
fine
```

Per capirne il funzionamento consideriamo questo esempio: `StampaLista [1 2].`

Alla prima invocazione di `StampaLista` la lista non è vuota: perciò la procedura ne stampa il primo elemento (1) e poi invoca `StampaLista [2].` La lista non è vuota, ne viene perciò stampato il primo elemento (2) e poi viene invocata `StampaLista [].` La lista è ora vuota e perciò viene eseguito il comando `stop` che termina l'esecuzione della procedura.

Coordinate cartesiane

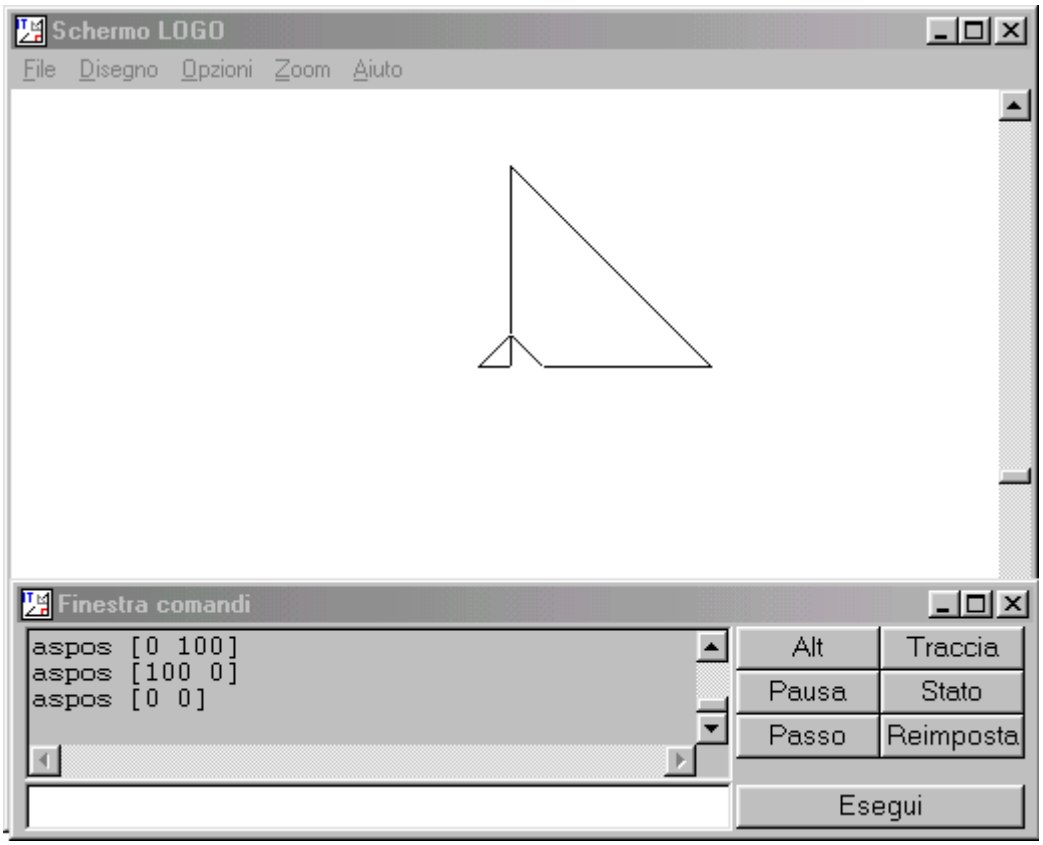
Sino ad ora abbiamo utilizzato i comandi grafici locali di Logo, tutti comandi che muovono la tartaruga o ne modificano la direzione di marci a partire da un dato punto.

Logo permette di connettere direttamente punti del piano tramite il comando `aspos` (assegna posizione), che richiede una lista di due elementi come argomento. Il primo elemento della lista è la coordinata delle ordinate, la seconda è quella delle ascisse.

Il centro dello schermo, dove si posiziona la tartaruga quando battiamo `ps`, ha coordinate [0 0].

Provate questi comandi:

```
ps
aspos [0 100]
aspos [100 0]
aspos [0 0]
```



Utilizzando il comando `su` possiamo muovere la tartaruga senza che lasci una traccia sullo schermo (`giu` ha l'effetto inverso).

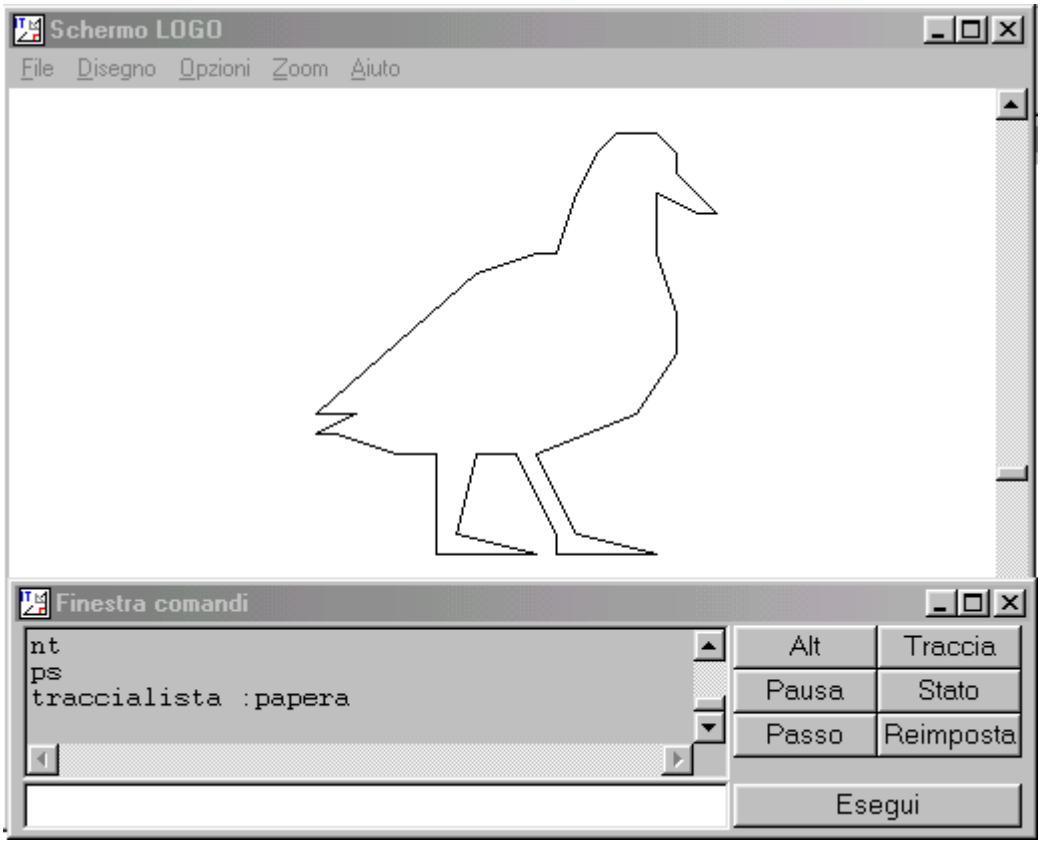
Definiamo ora una lista che contenga diverse coordinate. Ogni elemento di questa lista è perciò una lista composta da due elementi.

```
assegna "papera [[-100 -20] [-20 50] [10 60] [20 60] [30 90] [40 110] [50 120]
[70 120] [80 110] [80 100] [100 80] [90 80] [70 90] [70 60] [80 30] [80 10]
[60 -20] [10 -40] [30 -80] [70 -90] [20 -90]
[20 -80] [0 -40] [-20 -40] [-30 -80] [10 -90] [-40 -90] [-40 -40]
[-60 -40] [-90 -30] [-100 -30] [-80 -20]]
```

La procedura principale `TracciaLista` posiziona il cursore grafico (la tartaruga) alla coordinata rappresentata dal primo elemento della lista, chiama poi la procedura `TracciaElementi` che unisce i punti rappresentati dagli altri elementi della lista e termina tracciando l'ultimo segmento.

`TracciaElementi` è una procedura ricorsiva che esegue ripetutamente l'istruzione `aspos`.

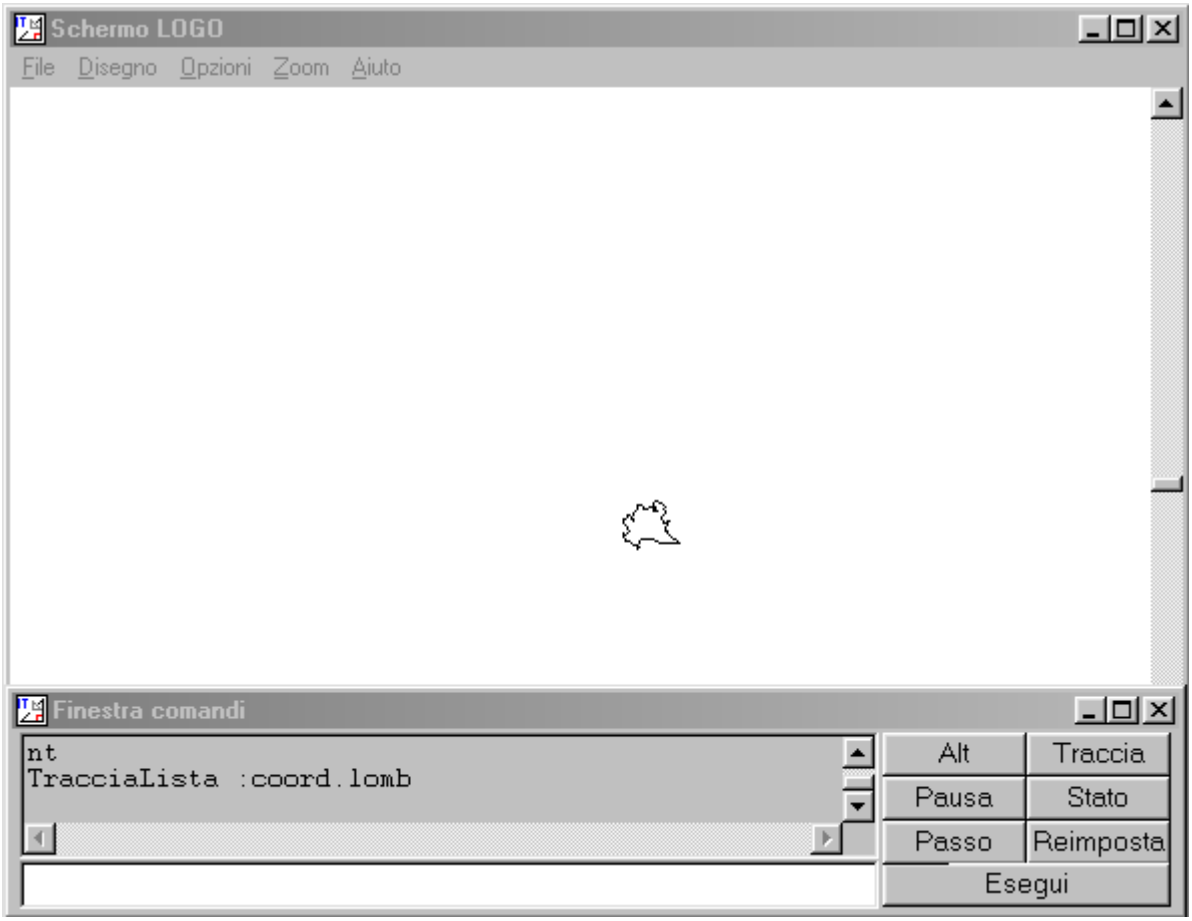
```
per TracciaLista :l
      su
      aspos primo :l
      giu
      TracciaElementi menpri :l
      aspos primo :l
fine

per TracciaElementi :l
      se :l = [] [stop]
      aspos  primo :l
      TracciaElementi menpri :l
fine
```



Possiamo usare questa procedura per tracciare i confini di un’area chiusa, come una regione. Proviamo con le coordinate della Regione Lombardia.

```
assegna "coord.lomb [[22.5 -14.5] [23.5 -14.5] [25.5 -18.25] [26 -17] [25 -
16.25] [28.25 -11.75] [27.75 -10.5] [28.25 -9.5] [30.25 -9.5] [30.75 -12] [35.5
-11] [36 -13] [37.5 -12.5] [36.5 -11.5] [37.25 -10.5] [36 -10.5] [36 -9.25]
[36.75 -8] [37.75 -8] [40.25 -9.25] [42.25 -10.75] [41 -11.5] [41.5 -14.25]
[40.25 -15.5] [41 -18.5] [44 -18] [42.25 -21] [42.25 -22.75] [43 -23] [43 -24]
[49.25 -28.75] [40.25 -29.25] [34.75 -26.5] [30 -27] [28 -29.75] [29 -30.75] [28
-32.25] [27.5 -31.25] [24 -27.75] [21.75 -27.75] [20.75 -26.25] [20.75 -24.25]
[22.25 -24.5] [23.5 -23] [20.75 -18.5] [21 -17.25] [22.5 -15.5]]
```

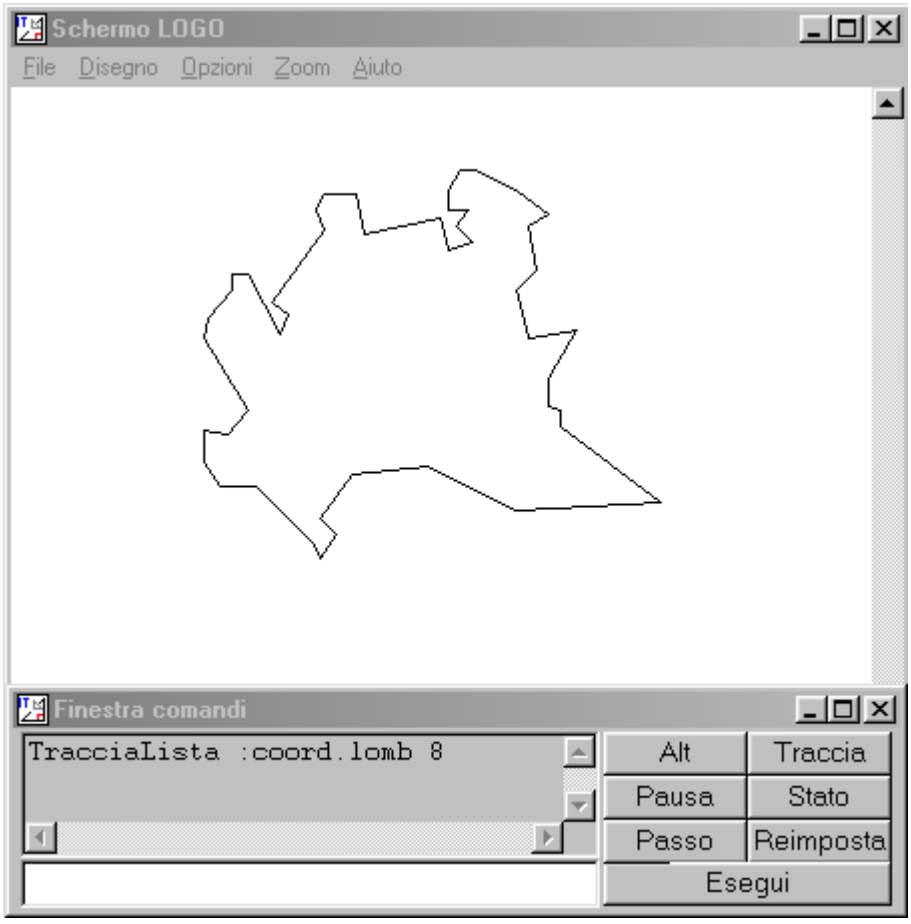
Come si vede il risultato non è molto soddisfacente, perché il disegno è troppo piccolo. Avremmo bisogno di una procedura che permetta di ingrandire (o rimpicciolire) i disegni.



Per ingrandire o restringere un'immagine dobbiamo applicare a ogni coordinata lo stesso fattore di scala. La procedura seguente contiene un nuovo parametro k che viene moltiplicato per ogni elemento della lista. Se k è maggiore di 1 l'immagine viene ingrandita, se minore di uno rimpicciolita.

```
per TracciaLista :l :k
      su
      aspos scala primo :l :k
      giu
      TracciaElementi menpri :l :k
      aspos scala primo :l :k
fine

per TracciaElementi :l :k
      se :l = [] [stop]
      aspos scala  primo :l :k
      TracciaElementi menpri :l :k
fine

per scala :l :k
      riporta frase prodotto primo :l :k prodotto ultimo :l :k
fine
```

Proviamo ora ad ingrandire il disegno di un fattore 8. E meglio, vero?



La media aritmetica

Per concludere questa sezione sulle liste, vogliamo calcolare la media aritmetica di una lista di numeri.

Le liste numeriche sono un caso particolarmente semplice di liste, che si prestano bene a rappresentare i risultati di una indagine statistica. Ipotizziamo di aver misurato l'altezza in centimetri di un gruppo di bambini di quarta elementare. I risultati della misurazione sono contenuti nella seguente lista, a cui abbiamo assegnato il nome `altezza`:

```
assegna "altezza  [150 140 142 138 152 132 139 139 132 135 137 134 140 121 141
142 139 142 140 136 138 139 146 144 140 147 151 150 147 152 137 147 147 135 135
142 137 139 136 136 139 152 156 143 143 142 145 139 138 142 137 129 133 140 143
132 130 136 124 141 145 142 141 140 147 135 156 136 140 136 136 131 126 138 140
143 137 126 135 132 135 139 145 130 138 142 139 143 144 140 142 156 141 137 144
138 144 142 134 138 137 134 135 146 137 144 132 139 142 141 132 131]
```

Per prima cosa vogliamo calcolare la somma di questi 112 valori (potete verificare che sono 112 con il comando `conta`).

La procedura `TotaleLista :altezza` riporta il valore di 15623.

```
per TotaleLista :l
      se :l = [] [riporta 0]
      riporta somma primo :l TotaleLista menpri :l
fine
```

La struttura di `TotaleLista` è simile a quella di `StampaLista` vista prima, con però alcune significative differenze.

Due comandi non ancora visti sono `somma` e `riporta`. Il loro significato è chiaro nella seguente procedura, che somma due numeri, restituendo (riportando) il risultato della somma ad un'altra procedura.

```
Per SommaNumeri :N :V
      riporta somma :N :V
fine
```

Provate a digitare `mostra SommaNumeri 3 5`. In primo luogo Logo esegue `SommaNumeri 3 5`; il risultato della somma, pari a 8, viene passato alla procedura `mostra` che lo stampa a video.

Torniamo a `TotaleLista`. Come al solito, consideriamo un esempio semplice, con una lista di soli due numeri : `[1 2]`.

`mostra TotaleLista [1 2]` inizialmente verifica se la lista `[1 2]` sia vuota. Non è vuota e perciò la procedura va alla sua seconda linea. Lì deve sommare il primo elemento della lista `[1 2]`, che è pari a 1, con il risultato di `TotaleLista [2]`.

```
      somma 1 TotaleLista [2]
```

Non può ancora farlo perché `somma` richiede come argomenti due numeri. Esegue perciò `TotaleLista [2]`. `TotaleLista [2]` verifica se la lista `[2]` sia vuota. Non è vuota e perciò la procedura va alla sua seconda linea. Lì deve sommare il primo elemento della lista `[2]`, che è pari a 2, con il risultato di `TotaleLista []`.

```
      somma 2 TotaleLista []
```

Non può ancora farlo perché `somma` richiede come argomenti due numeri. Esegue perciò `TotaleLista []`. `TotaleLista []` restituisce il valore 0, perché il test della prima riga è vero.

A questo punto la procedura precedente è in grado di fare la somma richiesta:

```
      somma 2 0
```

Il risultato di questa somma, pari a 2, viene fornito alla precedente procedura, che esegue:

```
somma 1 2
```

Il risultato finale di `TotaleLista`, pari a 3, viene restituito alla procedura `mostra` incaricata di stamparlo.

Possiamo a questo punto concludere calcolo della media aritmetica di `altezza`. Il suo valore è pari a 139,49, risultato della divisione, che in Logo può essere effettuata con l'istruzione `quoziente`, 15623 diviso 112.

La procedura `MediaAritmeticaLista` mette insieme tutti i calcoli fatti.

```
per MediaAritmeticaLista :l
      se :l = [] [riporta 0]
      riporta quoziente TotaleLista :l conta :l
fine
```

Istruzioni principali

Lista in ordine alfabetico dei principali comandi di MSWLogo in italiano.

acaso

Fornisce un numero intero casuale maggiore o uguale a 0 e minore del valore in input.

Sintassi: `acaso <valore>`

Esempio:

```
ripeti 5 [stampa acaso 6]
```

```
1
5
1
0
2
```

ascr

Imposta il colore per il riempimento delle figure. In input una lista di tre numeri interi compresi tra 0 e 255. I tre numeri della lista rappresentano nell'ordine la componente rossa, la componente verde e la componente blu del colore.

Sintassi: `ascr <lista componenti colore>`

Esempio:

```
per quadrato
ps nastarta
ripeti 4 [a 100 d 90]
ascr [255 0 0]
su
d 45
a 10
riempi
giu
fine
```

ascp

Imposta il colore per la traccia della tartaruga (il colore della penna). In input una lista di tre numeri interi compresi tra 0 e 255. I tre numeri della lista rappresentano nell'ordine la componente rossa, la componente verde e la componente blu del colore.

Sintassi: `ascp <lista componenti colore>`

Esempio:

```
ps
ascp [255 0 0]
a 100
```

ascs

Imposta il colore per lo sfondo del mondo della tartaruga. In input una lista di tre numeri interi compresi tra 0 e 255. I tre numeri della lista rappresentano nell'ordine la componente rossa, la componente verde e la componente blu del colore.

Sintassi: `ascs <lista componenti colore>`

Esempio:

```
ascs [0 0 255]
ps
```

aspetta

Crea una pausa prima dell'esecuzione dell'istruzione successiva di un numero di sessantesimi di secondo pari al valore in input.

Sintassi: `aspetta <valore>`

Esempio:

```
ps
a 50
aspetta 60
a 50
```

assegna

Assegna a una variabile un determinato valore.

Sintassi: `assegna "<nome variabile> <valore>`

Esempio:

```
assegna "n 10
stampa :n
```

10

Esempio:

```
assegna "nome [Pippo]
stampa :nome
```

Pippo

Esempio:

```
assegna "m 6
assegna "n :m
stampa :n
```

6

Esempio:

```
assegna "n 10
assegna "n :n + 1
```

stampa :n

11

Esempio:

assegna "n 10
stampa :n + 1
stampa :n

11
10

avanti

Fa avanzare la tartaruga di un numero di passi pari al valore in input, tenendo conto della posizione e dell'orientamento attuale. Abbreviazione: `a.`

Sintassi: `avanti <valore>`

Esempio:

```
a 100
```

Esempio:

```
assegna "n 100
a :n
```

cicloper

Esegue ripetutamente un blocco di istruzioni con la variabile di controllo che varia tra il primo valore in input e il secondo valore in input (con incremento o decremento di una unità). Se presente un terzo valore in input, l'incremento (o decremento) è uguale a tale valore.

Sintassi:

```
cicloper [<nome variabile di controllo> <valore> <valore>] [<blocco istruzioni>]
```

oppure

```
cicloper [<nome variabile di controllo> <valore> <valore> <valore> ] [<blocco istruzioni>]
```

Esempio:

```
cicloper [k 1 5] [stampa :k]
```

```
1
2
3
4
5
```

Esempio:

```
cicloper [k 5 1] [stampa :k]
```

```
5
4
3
2
1
```

Esempio:

```
cicloper [k 1 5 1.5] [stampa :k]
```

```
1
2.5
4
```

Esempio:

```
cicloper [k 5 1 -1.5] [stampa :k]
```

```
5
3.5
2
```

ciao

Uscita dal Logo.

destra

Fa ruotare la tartaruga su se stessa, alla sua destra, di un angolo pari al valore in input, tenendo conto dell'orientamento attuale. Abbreviazione: `d.`

Sintassi: `destra <valore>`

Esempio:

```
ps
a 50
d 90
```

Esempio:

```
ps
assegna "angolo 45
d :angolo
```

Esempio:

Per capire che la tartaruga ruota alla sua destra e non alla vostra destra date in modo diretto l'istruzione d 180 e poi l'istruzione d 90.

giu

Vedi su

indietro

Fa indietreggiare la tartaruga di un numero di passi pari al valore in input, tenendo conto della posizione e dell'orientamento attuale. Abbreviazione: `i`.

Sintassi: `i <valore>`

Esempio:

```
indietro 100
```

Esempio:

```
assegna "n 100
i :n
```

lista

Genera una lista i cui elementi sono i valori in input.

Sintassi:

```
lista <valore> <valore>
```

oppure

```
(lista <valore> <valore> ... <valore>)
```

Esempio:

```
per quadrato :rosso :verde :blu
ascp (lista :rosso :verde :blu)
ripeti 4 [a 100 d 90]
fine
```

mt

Vedi `nt`

nt

Nasconde la tartaruga . Per rendere di nuovo visibile la tartaruga bisogna dare l'istruzione `mt` .

Esempio:

```
ps
ripeti 5 [nt aspetta 60 mt aspetta 60]
```

non

Negazione logica di una condizione (NON).

Sintassi: `non <condizione>`

Esempio:

```
assegna "n acaso 10
```

```
stampa :n
se non :n>5 [ stampa [Il numero è minore o uguale a 5] ]
```

pulisci

Come `ps`, ma lascia la tartaruga nello stato in cui si trova.

Esempio:

```
a 100
pulisci
```

puliscischermo

Pulisce lo schermo eliminando tutto ciò che fosse stato tracciato. Viene inoltre riportata la tartaruga nello stato iniziale: al centro dello schermo e con orientamento verso nord (la tana). Abbreviazione: `ps.`

Esempio:

```
a 50
d 90
ps
```

puliscitesto

Cancella tutto il testo presente nella finestra comandi. Abbreviazione: `pt.`

Esempio:

```
stampa "Questo è un testo"
pt
```

resto

Fornisce il resto della divisione del primo valore in input per il secondo valore in input. I due valori devono essere numeri interi.

Sintassi: `resto <valore> <valore>`

Esempio:

```
stampa resto 7 2
```

1

Esempio:

```
stampa resto 6 2
```

0

riempi

Riempie di colore una figura chiusa. Il colore di riempimento è quello impostato con l'istruzione `ascr`. Per poter effettuare il riempimento la tartaruga deve essere portata all'interno della figura (senza lasciare la traccia).

Esempio:

```
per quadrato
ps nastarta
ripeti 4 [a 100 d 90]
ascr [255 0 0]
su
d 45
a 10
riempi
giu
fine
```

ripeti

Ripete le istruzioni comprese tra le parentesi quadre (blocco di istruzioni) per un numero di volte pari al valore in input.

Sintassi: `ripeti <valore> [<blocco istruzioni>]`

Esempio:

```
ps ripeti 5 [a 100 d 144]
ps ripeti 10 [a 100 d 108]
```

se

Istruzione condizionale (SE... ALLORA...). Vengono eseguite le istruzioni comprese tra parentesi quadre (blocco di istruzioni) solo se la condizione è verificata.

Sintassi: `se <condizione> [<blocco istruzioni>]`

Esempio:

```
assegna "n acaso 10
stampa :n
se :N > 5 [ stampa [ Il numero è maggiore di 5] stop ]
stampa [Il numero è minore o uguale a 5]
```

sealtrimenti

Istruzione condizionale (SE... ALLORA... ALTRIMENTI...). Se la condizione è verificata viene eseguito il primo blocco di istruzioni altrimenti viene eseguito il secondo blocco.

Sintassi:

```
sealtrimenti <condizione> [< primo blocco>] [< secondo blocco>]
```

Esempio:

```
assegna "n acaso 10
stampa :n
sealtrimenti :n > 5 [ stampa [ Il numero è maggiore di 5] ] [ stampa [ Il numero
è minore o uguale a 5] ]
```

sinistra

Fa ruotare la tartaruga su se stessa, alla sua sinistra, di un angolo pari al valore in input, tenendo conto dell'orientamento attuale. Abbreviazione: `s`.

Sintassi: `sinistra <valore>`

Esempio:

```
ps
a 50
s 90
```

Esempio:

```
ps
assegna "angolo 45
s :angolo
```

Esempio:

Per capire che la tartaruga ruota alla sua sinistra e non alla vostra sinistra date in modo diretto l'istruzione s 180 e poi l'istruzione s 90.

stampa

Visualizza nella finestra comandi il valore in input. Ogni istruzione stampa termina con un invio a capo.

Sintassi: `stampa <valore>`

Esempio:

```
pt
stampa 10
```

10

Esempio:

```
pt
stampa [Salve, come stai?]
```

Salve, come stai?

Esempio:

```
pt
assegna "N 5
stampa :N
```

5

Esempio:

```
pt
assegna "T [Salve, come stai?]
stampa :T
```

Salve, come stai?

Esempio:

```
stampa [Salve,]
stampa [come stai?]
```

Salve,
come stai?

stop

Interrompe l'esecuzione della procedura in cui si trova. Il controllo ritorna al contesto in cui la procedura ove è presente stop è stata chiamata.

Esempio:

```
per conta
assegna "k 0
stampa [inizio]
esegui
stampa [fine]
fine

per esegui
assegna "k :k+1
stampa :k
se :k=5 [stop]
esegui
fine
```

inizio
1
2
3
4
5
fine

su

Alza la penna. Dopo aver dato l'istruzione `su`, la tartaruga, spostandosi, non lascia la traccia sullo schermo. Per fare in modo che la tartaruga riprenda a lasciare la traccia bisogna dare l'istruzione `giu`.

Esempio:

```
ps
ripeti 5 [su a 10 giu a 10]
```

tana

Riporta la tartaruga nello stato iniziale (la tana) senza cancellare lo schermo.

Esempio:

```
ps
a 50
d 90
tana
```

tuttiveri?

Congiunzione logica di due condizioni (AND).

Sintassi: `tuttiveri? <condizione> <condizione>`

Esempio:

```
assegna "n acaso 10
stampa :n
se tuttiveri? :n>2 :n< 6 [ stampa [Il numero è compreso tra 2 e 6] ]
```

unovero?

Disgiunzione logica di due condizioni (OPPURE).

Sintassi: `unovero? <condizione> <condizione>`

Esempio:

```
assegna "n acaso 10
stampa :n
se unovero? :n>8 :n<4 [ stampa [Il numero è maggiore di 8 o minore di 4] ]
```

Risorse Logo

Di seguito indichiamo alcune risorse utili per approfondire la conoscenza di Logo e dei suoi utilizzi.

Aiuto in linea

La prima risorsa da esplorare è l'aiuto in linea di MSW-Logo. Nell'aiuto sono riportati tutti i comandi di MSW-Logo, e molti esempi che ne illustrano le pontenzialità.

Esempi

Nella cartella "c:\LogoIt\Esempi" sono contenute le seguenti sottocartelle:

1. Windows – programmi che costruiscono un'interfaccia grafica, provate "calc.lgo"
2. Misc – programmi vari
3. Ucblogo – programmi che accompagnano "Computer Science Logo Style" di Brian Harvey
4. Rete – uso del protocollo TCP/IP
5. 3d – grafica tridimensionale
6. Multimedia – Suoni, MIDI

Ognuna delle sottocartelle contiene numerosi programmi Logo – alcuni però piuttosto complessi - che possono essere eseguiti e studiati.

Internet

Su Internet sono disponibili molte risorse Logo. Di seguito ne riportiamo alcune per il loro interesse intrinseco e perché possono servire come punto di ingresso per ulteriori ricerche.

Versioni di Logo per Windows, Mac e Unix

http://www.softronix.com/
Versione aggiornata di MSWLogo (versione inglese) – link a molti siti web su Logo

http://www.racine.ra.it/curba/links.htm
Liceo Scientifico Statale "G. Ricci Curbastro"
MSWLogo (versione italiana), manuale , presentazione in power point

http://www.irre.veneto.it/informatica/logo/logo.asp
MSWLogo (versione italiana)

http://www.cs.berkeley.edu/~bh/
UCB Logo per Windows, Mac e Unix; "Computer Science Logo Style"

http://www.ugcs.caltech.edu/~dazuma/turtle/
Turtle Tracks, implementazione di Logo in Java, per Windows, Mac e Unix

Forum di discussione

http://groups.yahoo.com/group/dw-logo/
la didattica del logo nella scuola primaria italiana

http://groups.yahoo.com/group/LogoForum/
LogoForum · A forum for sharing and discussion among Logo users and Logo using educators

Siti web in italiano

http://wwwa024.infonegocio.com/381/media/logo/index.htm
BENVENUTI ALLA PAGINA DI LOGO DELLA SCUOLA MEDIA ITALIANA DI MADRID
Matematica, geometria, trangram

http://users.iol.it/prof.lazzarini/mswlogo/
Guida a MSWlogo (versione inglese)

http://www.racine.ra.it/orione39/sfida-logo.htm
Si tratta di una sfida a creare delle belle procedure per fare i disegni rappresentati

http://linda.dei.unipd.it/smac/logo/libro.htm
"Il libro della scoperta della tartaruga" di Jim Muller, traduzione italiana, MSWLogo (versione inglese)

http://web.tiscali.it/cdidatticomacomer/
Circolo Didattico di Macomer
Esperienze di utilizzo di Logo nella scuola elementare

http://digilander.iol.it/dirpin1/index.htm
SCUOLA ELEMENTARE - Pinerolo

http://members.xoom.it/rabbone/logo.htm
Pagina personale di Alessandro Rabbone, IRRE Piemonte

http://web.tiscali.it/mswlogo/
Manuali e programmi per MSWLogo, versione italiana

Siti web in inglese

http://el.www.media.mit.edu/groups/logo-foundation/
Logo Foundation
Punto di ingresso per le risorse Logo nel mondo (link, saggi, esperienze d'uso, rivista online, software)

http://www.eurologo.org/
Link ai convegni internazionali di studio su Logo tenutisi a Linz nel 2001, a Sofia nel 1999 e a Budapest nel 1997 – disponibili alcuni dei contributi ivi presentati

http://www.geocities.com/CollegePark/Lab/2276/
Recursive Graphic Designs – tutto sulle procedure grafiche ricorsive

http://www.educa.fmf.uni-lj.si/izodel/default.htm
Logo all'Università di Lubiana

http://www.mathcats.com/gallery/15wordcontest.html
"Write a Logo one-liner using 15 or fewer words, not counting square brackets and parentheses, to produce the most beautiful picture" – vincitore Paolo Passaro con "`repeat 1800 [fd 10 rt repcount + .1]`"

http://www.yukoncollege.yk.ca/%7Esrudge/comp052/outline.html
Corso universitario di programmazione con Logo allo Yukon College

http://mtl.math.uiuc.edu/modules/logo/
corso di e-learning: "Logo Programming for the Mathematics Classroom"