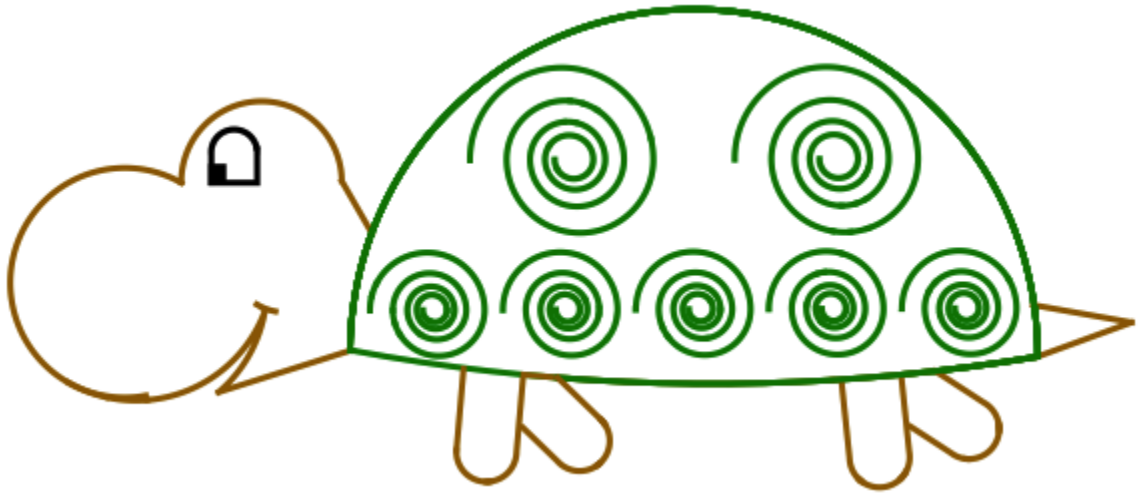
ABZ AUSBILDUNGS- UND BERATUNGSZENTRUM
FÜR INFORMATIKUNTERRICHT

Heidi Gebauer Juraj Hromkovič Lucia Keller
Ivana Kosírová Giovanni Serafini Björn Steffen

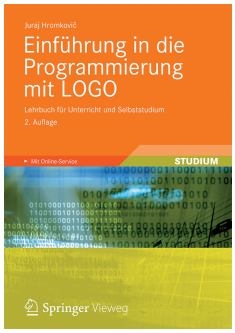
Programmare con LOGO



ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Programmare con LOGO

Questa guida è una versione abbreviata delle lezioni 1 fino a 7 del libro di testo in tedesco *Einführung in die Programmierung mit LOGO*. Il libro contiene numerosi esercizi e spiegazioni supplementari, oltre a fornire delle indicazioni per i docenti. In tutto, il libro comprende 15 lezioni.



Juraj Hromkovič. *Einführung in die Programmierung mit LOGO: Lehrbuch für Unterricht und Selbststudium*. 2. Aufl., Springer Vieweg 2012. ISBN: 978-3-8348-1852-2.

Versione 3.1, 7 aprile 2014, SVN-Rev: 14199

Traduzione in italiano a cura di Nicolas Kick e Giovanni Serafini
Revisione del testo: Giovanni Serafini

Ambiente di sviluppo

La guida è stato realizzata per l'ambiente di sviluppo XLogo. XLogo è disponibile gratuitamente sul sito internet `xlogo.tuxfamily.org`.

Per eseguire i programmi in Logo presenti nella guida è necessario aver installato XLogo in inglese.

Diritti d'uso

ABZ

L'ABZ (Ausbildungs- und Beratungszentrum für Informatikunterricht der ETH Zürich) è il centro di formazione e consulenza per l'insegnamento dell'informatica del Politecnico Federale di Zurigo. Grazie ad un'ampia paletta di offerte, l'ABZ sostiene scuole e docenti che desiderano iniziare o ampliare l'insegnamento dell'informatica. L'offerta comprende tra l'altro consulenze individuali, insegnamento da parte di professori del Politecnico e del team dell'ABZ direttamente nelle scuole, corsi di formazione e aggiornamento per docenti come pure la preparazione di materiale didattico.

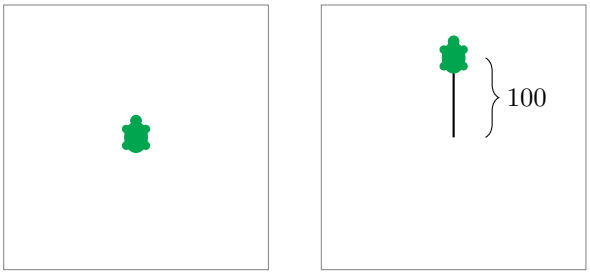
`www.abz.inf.ethz.ch`

1 Istruzioni di base

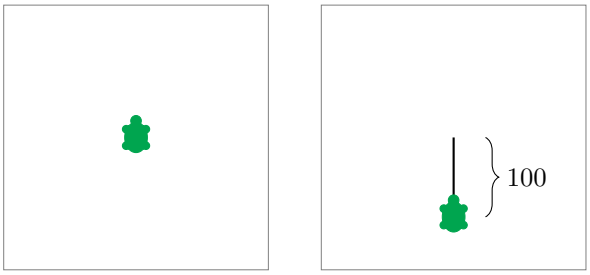
Un'**istruzione** è un comando che il computer è in grado di capire e di eseguire. In realtà, il computer capisce soltanto un numero ridotto di istruzioni. Se vogliamo fargli eseguire dei compiti complessi, dobbiamo istruire il computer in modo adatto. Lo facciamo scrivendo una dettagliata sequenza di istruzioni. Una tale sequenza di istruzioni è chiamata **programma**. Scrivere un programma non è sempre facile. Ci sono programmi costituiti da milioni di istruzioni. Per non perdere la visione d'insieme è necessario procedere in modo adeguato e sistematico, come impareremo durante il nostro corso di programmazione.

Tracciare una linea retta

Con l'istruzione `forward 100` oppure `fd 100` ordini alla tartaruga di muoversi di 100 passi in avanti:



Con l'istruzione `back 100` oppure `bk 100` la tartaruga indietreggia di 100 passi:



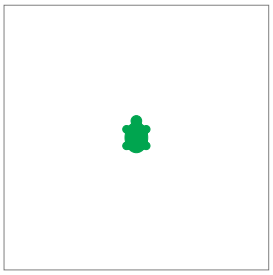
Cancellare e ricominciare da capo

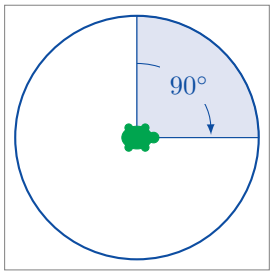
L'istruzione **`cs`** cancella tutto ciò che hai disegnato finora. La tartaruga torna alla sua posizione iniziale.

Girare

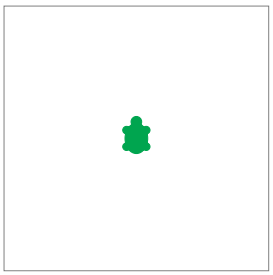
La tartaruga si muove sempre nella direzione in cui sta guardando.

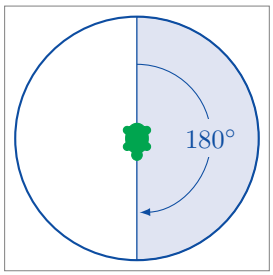
Con l'istruzione **`right`** `90` oppure **`rt`** `90` la tartaruga si gira di 90° verso destra. Ciò corrisponde a un quarto di cerchio:



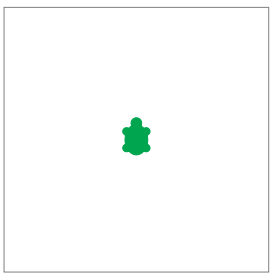


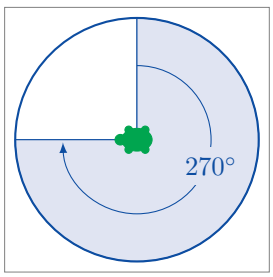
L'istruzione **`right`** `180` oppure **`rt`** `180` fa girare la tartaruga di 180° verso destra. Ciò corrisponde a un semicerchio:



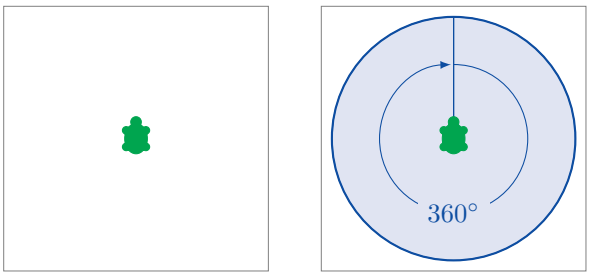


`right` `270` oppure **`rt`** `270` fa girare la tartaruga di 270° verso destra:

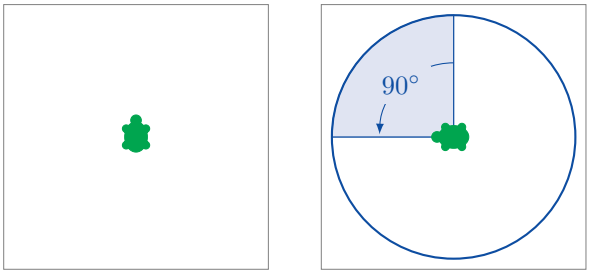




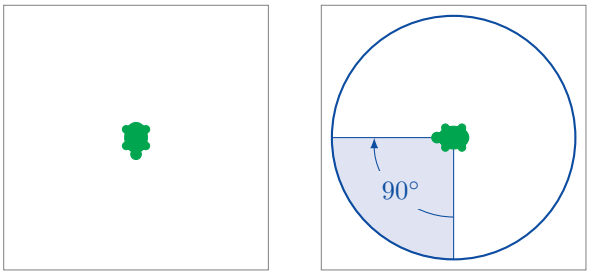
Le istruzioni `right 360` e `rt 360` fanno girare la tartaruga di 360° verso destra. La tartaruga fa un giro completo su se stessa.



Con l'istruzione `left 90` oppure `lt 90` la tartaruga si gira di 90° verso sinistra:



Stai attento: la tartaruga si gira sempre alla **sua** destra e alla **sua** sinistra, come illustrato nell'esempio seguente, con il comando `rt 90`:



Programmare

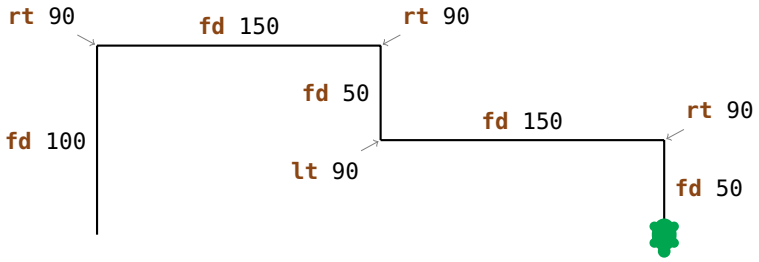
Programmare significa scrivere una sequenza di istruzioni.

Esercizio 1

Copia ed esegui il programma seguente:

```
fd 100
rt 90
fd 150
rt 90
fd 50
lt 90
fd 150
rt 90
fd 50
```

Hai ottenuto quest'immagine?



Esercizio 2

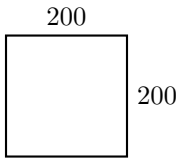
Scrivi ed esegui il programma seguente:

```
fd 100
rt 90
fd 200
rt 90
fd 80
rt 90
fd 100
rt 90
fd 50
```

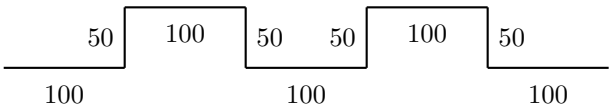
Disegna la figura che ottieni nella griglia soprastante e descrivi (come nell'Esercizio 1) quello che fa ogni singola istruzione.

Esercizio 3

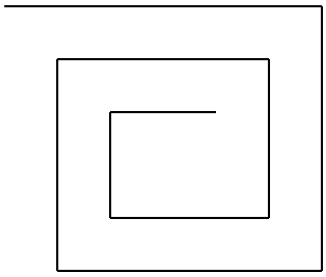
Scrivi dei programmi per disegnare le immagini seguenti. Per ogni immagine sei libero di scegliere il punto di partenza della tartaruga.



(a)

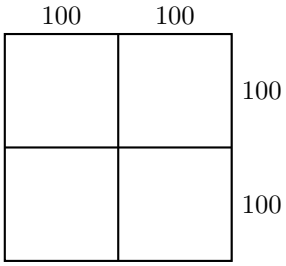


(b)



Puoi scegliere la grandezza come vuoi.

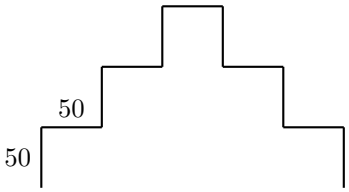
(c)



(d)

Esercizio 4

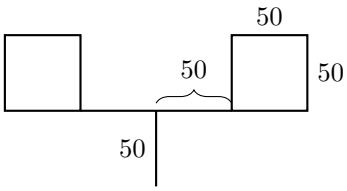
Scrivi un programma che disegna l'immagine seguente:



Sei in grado di modificare il tuo programma in modo da usare soltanto le istruzioni `fd 50` e `rt 90`?

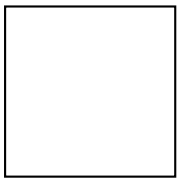
Esercizio 5

Anna vuole disegnare l'immagine seguente. Riesci ad aiutarla?



2 L'istruzione `repeat`

Per disegnare un quadrato di lato 100



possiamo utilizzare il programma seguente:

```
fd 100
rt 90
fd 100
rt 90
fd 100
rt 90
fd 100
rt 90
```

Constatiamo che le due istruzioni

```
fd 100
rt 90
```

sono ripetute quattro volte. Non sarebbe più semplice dire al computer di ripetere lui stesso quattro volte queste due istruzioni, invece di scriverle quattro volte di fila?

Il programma seguente fa proprio questo:

`repeat`	`4`	`[fd 100 rt 90]`
Istruzione per ripetere un programma	Numero di ripetizioni	Sequenza di istruzioni da ripetere

Esercizio 6

Copia ed esegui il programma seguente:

```
fd 75 lt 90
fd 75 lt 90
fd 75 lt 90
fd 75 lt 90
```

Che cosa disegna il programma? Riesci a utilizzare l'istruzione **repeat** per abbreviare il tuo programma?

Esercizio 7

Scrivi il programma seguente per vedere che cosa disegna:

```
fd 50 rt 60
fd 50 rt 60
fd 50 rt 60
fd 50 rt 60
fd 50 rt 60
fd 50 rt 60
```

Accorcialo usando l'istruzione **repeat**.

Esercizio 8

Utilizza l'istruzione **repeat** per disegnare un quadrato di lato 200.

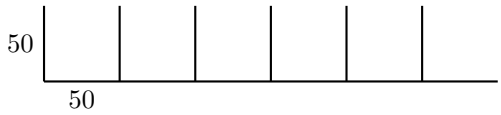
Esercizio 9

Scrivi ed esegui il programma seguente:

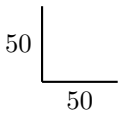
```
fd 100 rt 120
fd 100 rt 120
fd 100 rt 120
```

Che cosa disegna il programma? Riesci a utilizzare l'istruzione **repeat** per abbreviare il tuo programma?

Ora vogliamo disegnare l'immagine seguente usando l'istruzione **repeat**:



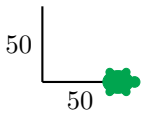
Prima di cominciare a disegnare, dobbiamo capire qual è il motivo che si ripete. Per esempio possiamo prendere il motivo seguente:



Se cominciamo in basso a sinistra, possiamo disegnare il motivo col programma seguente:

```
fd 50 bk 50 rt 90 fd 50
```

Dopo aver eseguito il programma, la tartaruga è posizionata come nell'immagine seguente e guarda verso destra:



Ora la tartaruga è già in posizione per disegnare il prossimo motivo. Dobbiamo solamente girarla con l'istruzione **lt** `90`.

Eseguiamo il programma per controllare se funziona correttamente:

```
fd 50 bk 50 rt 90 fd 50
lt 90
```

Otteniamo proprio ciò che volevamo:

Se eseguiamo di nuovo lo stesso programma, otteniamo:

Vediamo quindi che la nostra idea funziona e che possiamo ripetere il nostro programma 6 volte:

```
repeat 6 [fd 50 bk 50 rt 90 fd 50   lt 90   ]
```

Motivo — Riposizionamento

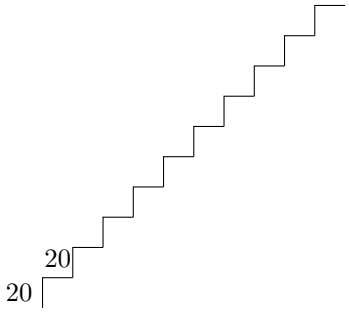
Possiamo risolvere molti esercizi utilizzando proprio quest'ultimo metodo. Per prima cosa devi sempre trovare il motivo che si ripete. In seguito devi scrivere due programmi: un programma per il *motivo* e un altro per il *riposizionamento* della tartaruga in modo che sia pronta a disegnare il prossimo motivo. Il tuo programma avrà l'aspetto seguente:

repeat *numero* [*motivo riposizionamento*]

Esercizio 10

Disegnare una scala.

(a) Disegna una scala con 10 gradini di grandezza 20:



- Trova dapprima il motivo che si ripete e scrivi un programma per disegnarlo.
- Pensa poi a come potresti scrivere un programma per riposizionare la tartaruga, in modo che guardi nella direzione giusta per la prossima ripetizione del motivo.
- Infine unisci in modo adeguato i due programmi per risolvere l'esercizio.

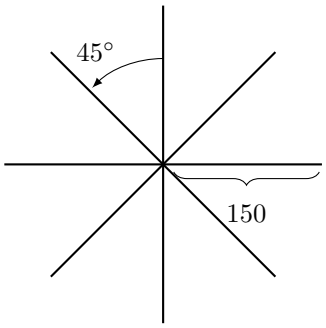
(b) Disegna una scala con 5 gradini di grandezza 50.

(c) Disegna una scala con 20 gradini di grandezza 10.

Esercizio 11

Adesso vogliamo disegnare delle stelle.

(a) Disegna la stella seguente:

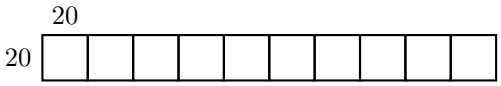


(b) La stella ha otto raggi di lunghezza 150. Riesci anche a disegnare una stella con 16 raggi di lunghezza 100?

Esercizio 12

Disegna le figure seguenti:

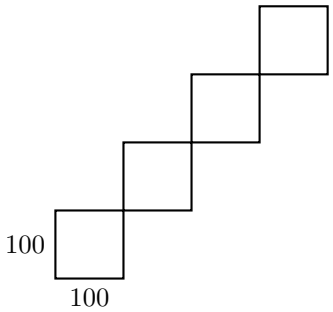
(a)



(b)

Esercizio 13

Scrivi un programma per disegnare l'immagine seguente:



Esercizio 14

Copia ed esegui il programma seguente:

```
repeat 4 [fd 100 rt 90]
rt 90
repeat 4 [fd 100 rt 90]
rt 90
repeat 4 [fd 100 rt 90]
rt 90
repeat 4 [fd 100 rt 90]
rt 90
```

Che cosa disegna il programma? Riesci a scriverne una versione più breve?

Modalità di spostamento

Normalmente la tartaruga si trova in **modalità di disegno**. Ciò significa che ha una matita in mano e che, quando si sposta, disegna.

In **modalità di spostamento** la tartaruga si muove sullo schermo senza disegnare. Puoi passare alla modalità di spostamento con l'istruzione

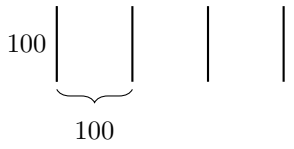
> **penup** oppure semplicemente **pu**.

Puoi di nuovo passare dalla modalità di spostamento alla modalità di disegno con l'istruzione

> **pendown** oppure semplicemente **pd**.

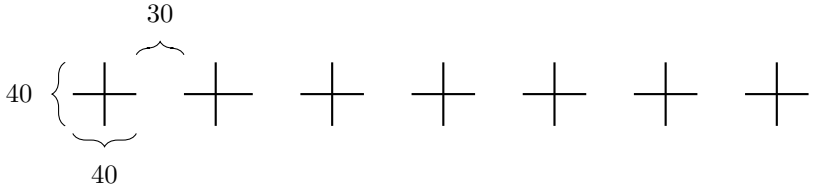
Esercizio 15

Scrivi un programma per disegnare la figura seguente:



Esercizio 16

Scrivi un programma per disegnare la figura seguente:



3 Dare un nome e richiamare dei programmi

Possiamo dare un nome a ogni programma che scriviamo. Quando, più in là, scriviamo il nome del programma nella riga di comando, il suo contenuto verrà eseguito.

Il programma per disegnare un quadrato di lato 100 è:

```
repeat 4 [fd 100 rt 90]
```

Possiamo dare il nome `QUAD100` a questo programma:

```
to QUAD100
repeat 4 [fd 100 rt 90]
end
```

Abbiamo quindi scritto lo stesso programma due volte: la prima senza un nome e la seconda con un nome.

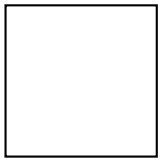
I programmi con un nome devono essere scritti nell'**editor**. In questa guida, tali programmi sono contrassegnati con un rettangolo su sfondo grigio. Non appena abbiamo finito di scrivere un programma, premiamo il pulsante con la tartaruga per chiudere l'editor.

Ciascuno può scegliere liberamente che nome dare a un programma; noi abbiamo scelto `QUAD100` per sottolineare che il programma disegna un quadrato di lato 100. Le sole regole per il nome di un programma sono che esso sia composto soltanto di lettere e numeri e che non contenga spazi.

Sullo schermo non è ancora stato disegnato niente, perché finora ci siamo limitati a dare un nome al programma, senza però eseguirlo. Se ora scriviamo il nome

```
QUAD100
```

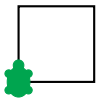
nella riga di comando, verrà eseguito il programma `repeat 4 [fd 100 rt 90]`. Sullo schermo comparirà l'immagine seguente:



Osserviamo di nuovo l'Esercizio 12(a). Possiamo risolvere questo esercizio più facilmente se dapprima scriviamo un programma per il motivo da ripetere, ovvero il quadrato di lato 20, e se a questo programma diamo un nome:

```
to QUAD20
repeat 4 [fd 20 rt 90]
end
```

Dopo aver disegnato `QUAD20` la tartaruga è posizionata nell'angolo del quadrato in basso a sinistra:



Per disegnare il prossimo quadrato, la tartaruga deve trovarsi nell'angolo in basso a destra. Perciò scriviamo il programma

```
rt 90 fd 20 lt 90
```

Anche in questo caso diamo un nome al programma:

```
to RIPOSIZIONARE20
rt 90 fd 20 lt 90
end
```

Con l'aiuto di questi due programmi, possiamo scrivere un programma per l'Esercizio 12(a) nel modo seguente:

```
repeat 10 [QUAD20 RIPOSIZIONARE20]
```

Possiamo dare un nome anche a questo nuovo programma. Per esempio:

```
to FILA10
repeat 10 [QUAD20 RIPOSIZIONARE20]
end
```

In questo caso diciamo che i programmi `QUAD20` e `RIPOSIZIONARE20` sono **sottoprogrammi** del programma `FILA10`.

Esercizio 17

Scrivi un programma per risolvere l' Esercizio 12(b), usando un programma che disegna dei quadrati di lato 30. Il programma assomiglierà a questo:

```
repeat 4 [QUAD30 RIPOSIZIONARE30]
```

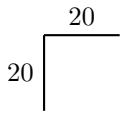
Devi quindi scrivere i sottoprogrammi `QUAD30` e `RIPOSIZIONARE30` adeguati.

Esercizio 18

Utilizza il programma `QUAD100` come sottoprogramma per disegnare la figura dell'Esercizio 13.

Esercizio 19

Scrivi un programma per disegnare uno scalino



e utilizzalo come sottoprogramma per risolvere l'Esercizio 10(a).

Esercizio 20

Risolvi di nuovo l'Esercizio 11(a), questa volta usando il sottoprogramma seguente:

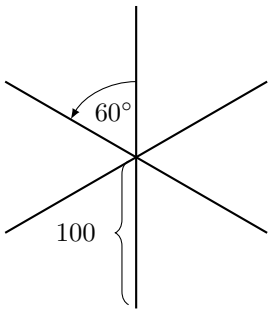
```
to LINEA
fd 150 bk 150
end
```

Esercizio 21

Scrivi ed esegui il seguente programma `RAGGIO`:

```
to RAGGIO
fd 100 bk 200 fd 100
end
```

Utilizza il programma `RAGGIO` come sottoprogramma per il programma `STELLA6`, che disegna l'immagine seguente:



Esercizio 22

Risolvi di nuovo l'Esercizio 15 e l'Esercizio 16 con l'aiuto di sottoprogrammi.

Esercizio 23

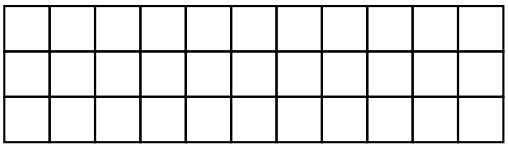
Precedentemente abbiamo creato il programma `FILA10`. Che cosa fa il programma seguente?

```
FILA10 fd 20 lt 90 fd 200 rt 90
```

Verifica la tua idea al computer.

Esercizio 24

Scrivi un programma che disegna l'immagine seguente:



Esercizio 25

Disegnare quadrati di grandezza diversa.

(a) Scrivi un programma che disegna un quadrato di lato 50 e dagli il nome `QUAD50`. Eseguilo per verificarne il funzionamento.

(b) Scrivi un programma che disegna un quadrato di lato 75.

(c) Esegui il programma

```
QUAD50
QUAD75
QUAD100
```

Che cosa compare sullo schermo?

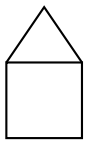
(d) Come modificheresti il programma per aggiungere altri tre quadrati, più grandi dei precedenti?

Costruire delle case

Come prossimo passo vogliamo aiutare un architetto a costruire un quartiere. Per facilitare il compito, l'architetto ha deciso di costruire case identiche. Gli facciamo la proposta seguente:

```
to CASA
rt 90
repeat 4 [fd 50 rt 90]
lt 60 fd 50 rt 120 fd 50 lt 150
end
```

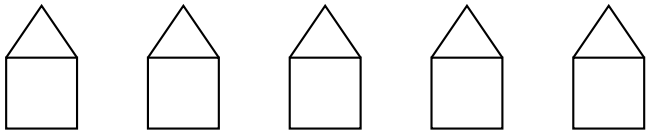
Il programma disegna la casa sottostante:



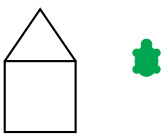
Esercizio 26

Da dove parte la tartaruga per disegnare la casa? Pensa a quale potrebbe essere la strada che la tartaruga percorre quando il programma `CASA` viene eseguito. Dove si trova la tartaruga quando ha finito di disegnare la casa? Disegna l'immagine corrispondente e descrivi, come nell'Esercizio 1, cosa fa ogni singola istruzione.

L'architetto ha ora fatto costruire una casa e constata che tutto ha funzionato perbene. Decide perciò di usare questo programma come motivo per costruire una strada di case identiche. Alla fine la strada dovrà apparire così:



Visto che le case sono tutte uguali, l'architetto può utilizzare il programma `CASA` cinque volte senza dover pensare ogni volta a come costruire una casa nuova. Egli dice alla tartaruga di disegnare la prima casa, sulla sinistra, e poi di andare al punto di partenza della prossima casa:



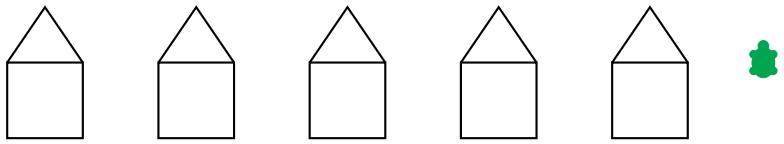
Per fare ciò, l'architetto usa il programma seguente:

```
CASA rt 90 pu fd 50 lt 90 pd
```

Ora la tartaruga può disegnare di nuovo la stessa casa e quindi andare al punto di partenza della prossima casa. La tartaruga ripete questo procedimento finché ha disegnato tutte e cinque le case. Per ottenere una fila di 5 case identiche, dobbiamo ripetere il programma `CASA` per 5 volte. Al programma corrispondente diamo il nome `FILACASE`:

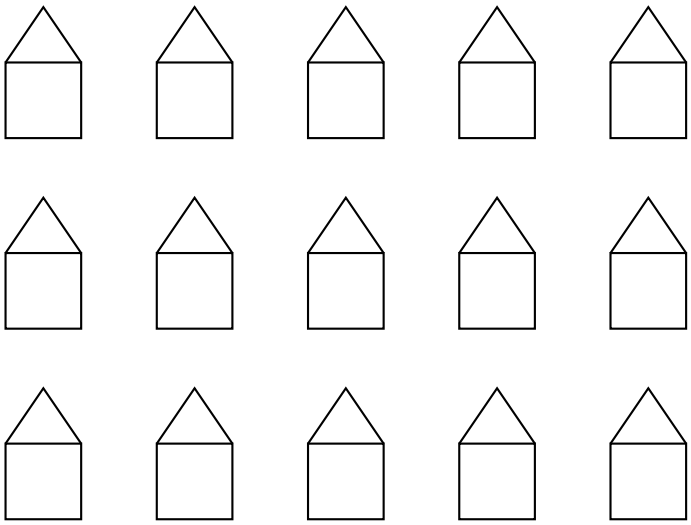
```
to FILACASE
repeat 5 [CASA rt 90 pu fd 50 lt 90 pd]
end
```

Alla fine la tartaruga si trova sulla destra, lì dove verrebbe disegnata la prossima casa:



Esercizio 27

Ora vogliamo aggiungere un paio di strade al quartiere di case. Utilizza il programma `FILACASE` come motivo per disegnare l'immagine seguente:



Suggerimento: Dopo ogni fila, la tartaruga deve di nuovo essere posizionata nel punto giusto per disegnare la prossima fila.

Linee spesse e quadrati neri

Esercizio 28

Disegnare delle linee spesse col programma `LINEASPESSA`.
Dai (nell'editor) il nome `LINEASPESSA` al programma

```
fd 100
rt 90
fd 1
rt 90
fd 100
rt 180
```

e scrivi quindi

```
LINEASPESSA
```

nella riga di comando. Che cosa disegna la tartaruga? Disegna con la matita su un foglio la figura che hai ottenuto.

Esercizio 29

Ripeti il programma `LINEASPESSA` 100 volte col programma

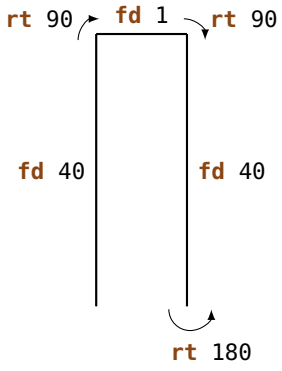
```
repeat 100 [LINEASPESSA]
```

Che cosa compare sullo schermo?

Esercizio 30

In questo esercizio disegneremo delle linee spesse. Nell'Esercizio 28 abbiamo visto che una linea spessa può essere disegnata nel modo seguente:

```
to LINEASPESSA40
fd 40
rt 90
fd 1
rt 90
fd 40
rt 180
end
```



La linea spessa viene creata disegnando due linee una vicino all'altra, di modo che si vede soltanto una linea spessa.

Scrivi il programma `LINEASPESSA40` ed eseguilo.

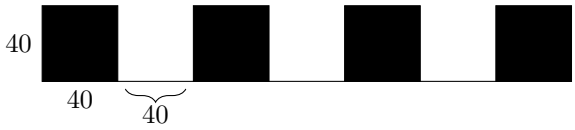
Esercizio 31

Una linea spessa di lunghezza 40 può essere interpretata come un rettangolo di larghezza 1 e altezza 40. Dopo aver disegnato `LINEASPESSA40`, la tartaruga si trova ai piedi della seconda linea e guarda in alto. Se ripetiamo il programma `LINEASPESSA40`, la tartaruga passerà di nuovo su questa seconda linea. Otteniamo quindi un rettangolo di larghezza 2 e altezza 40. Ogni ripetizione aggiunge una nuova linea. Se eseguiamo `LINEASPESSA40` 40 volte, otteniamo il quadrato nero di lato 40. Esegui `LINEASPESSA40` 40 volte per verificare che l'idea funziona.

Scrivi un programma di nome `NERO40` che disegna un quadrato nero di lato 40.

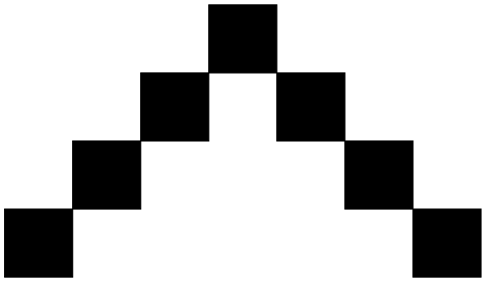
Esercizio 32

Utilizza il programma `NERO40` per disegnare l'immagine seguente:



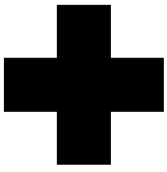
Esercizio 33

Utilizza il programma `NERO40` per disegnare l'immagine seguente:



Esercizio 34

Disegna l'immagine seguente:



Esercizio 35

Scrivi un programma per disegnare l'immagine seguente:

Esercizio 36

L'architetto decide di ordinare il tetto della casa da un altro fornitore. Ottiene due componenti: una componente `TETTO` e una componente `COMPONENTEINFERIORE`. Scrivi per l'architetto due programmi che disegnano queste due componenti ed uniscili nel nuovo programma `CASA1` per disegnare una casa.

Esercizio 37

Le case dell'Esercizio 27 sono molto semplici da costruire. Sii creativo e progetta una nuova casa per costruire un nuovo quartiere.

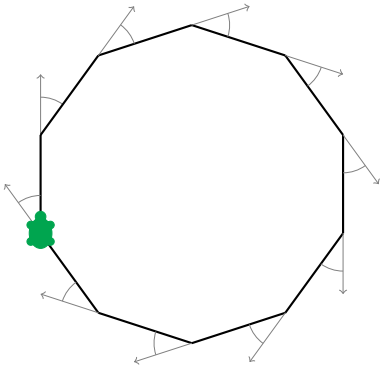
4 Poligoni regolari e cerchi

Poligoni regolari

Un poligono regolare ha n angoli e n lati della stessa lunghezza. Per esempio, per disegnare un decagono (un poligono regolare con 10 lati) a matita, devi disegnare 10 linee cambiando dopo ogni linea leggermente la direzione.

Di quanto bisogna cambiare direzione?

Per disegnare un poligono regolare dobbiamo girare la tartaruga più volte, ma alla fine essa si troverà alla posizione iniziale e guarderà nella stessa direzione come all'inizio.



Ciò significa che la tartaruga si è girata in tutto di ben 360°. Per disegnare un decagono bisogna quindi girarsi dello stesso angolo 10 volte. Questo angolo misura:

$$\frac{360°}{10} = 36°$$

Perciò bisogna girarsi sempre di 36°: `rt 36`. Proviamoci, scrivendo il programma seguente:

```
repeat 10 [fd 50      rt 36   ]
          ^^^^^       ^^^^^
          Lato        Rotazione di 36°
```

Esercizio 38

Disegna i poligoni regolari seguenti:

(a) un pentagono (5 lati) di lato 180,

(b) un dodecagono (12 lati) di lato 50,

(c) un quadrato di lato 200,

(d) un esagono (6 lati) di lato 100,

(e) un triangolo di lato 200 e

(f) un ottadecagono (18 lati) di lato 20.

Se vogliamo disegnare un ettagono (poligono regolare con 7 lati), abbiamo un problema: 360 non è divisibile senza resto per 7. In questo caso lasciamo che sia il computer a calcolare il risultato esatto scrivendo

```
360/7
```

('`/`' per il computer significa 'dividere'). Il computer troverà il risultato esatto. Perciò possiamo disegnare un ettagono di lato 100 procedendo nel modo seguente:

```
repeat 7 [fd 100 rt 360/7]
```

Verifica che il programma funziona.

Disegnare dei cerchi

Con i comandi **fd** e **rt** non è possibile disegnare dei cerchi esatti. Ma come avrai sicuramente già notato, i poligoni regolari con molti angoli assomigliano parecchio a dei cerchi. Possiamo quindi ottenere dei cerchi usando molti angoli e lati corti.

Esercizio 39

Prova il programma seguente:

```
repeat 360 [fd 1 rt 1]
repeat 180 [fd 3 rt 2]
repeat 360 [fd 2 rt 1]
repeat 360 [fd 3.5 rt 1]
```

3.5 significa 3 passi e mezzo.

Esercizio 40

(a) Come faresti a disegnare un cerchio piccolissimo? Scrivi un programma per farlo.

(b) Come faresti a disegnare un cerchio grande? Scrivi un programma per farlo.

Esercizio 41

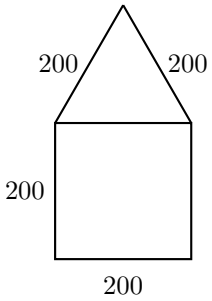
Prova a disegnare i seguenti semicerchi. Puoi scegliere tu la loro grandezza:

(a)

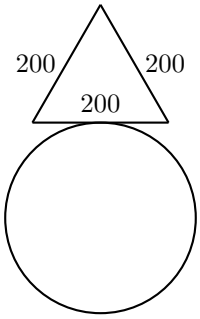
(b)

Esercizio 42

Utilizza le tue nuove conoscenze per disegnare le immagini seguenti. Puoi scegliere tu la grandezza del cerchio:



(a)



(b)

Diamo spazio alla fantasia

Disegna un ettagono (poligono regolare con 7 lati):

```
repeat 7 [fd 100 rt 360/7]
```

In seguito gira la tartaruga di 10° con l'istruzione

```
rt 10
```

Ripeti un paio di volte questi due programmi e osserva l'immagine ottenuta. Dopo ogni ettagono la tartaruga si gira di 10° con `rt 10`. Se vogliamo che essa torni alla posizione

iniziale dobbiamo ripetere il procedimento

$$\frac{360°}{10°} = 36$$

volte. Osserviamo quindi che cosa disegna il programma seguente:

```
repeat 36 [repeat 7 [fd 100 rt 360/7] rt 10]
```

Esercizio 43

Disegna un dodecagono (poligono regolare con 12 lati) di lato 70 e giralo 18 volte per tornare alla posizione iniziale.

Suggerimento: Puoi scrivere dapprima un programma per un dodecagono di lunghezza 12 e dargli per esempio il nome `DODECAGONO`. In seguito dovrai soltanto completare il programma

```
repeat 18 [DODECAGONO rt ... ]
```

Esercizio 44

Inventa un esercizio simile all'Esercizio 43 e scrivi un programma per risolverlo.

Colori

Se diamo sfogo alla nostra fantasia, perché non fare capo anche dei colori? La tartaruga può disegnare non soltanto in nero, ma con altri colori di tua scelta. A ogni colore è associato un numero. Di seguito trovi una lista dei colori disponibili:

0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16

Con l'istruzione

`setpencolor`	`X`
Istruzione per cambiare colore	Numero del colore desiderato

la tartaruga passerà ad usare il colore numero `X`. Possiamo anche abbreviare il comando con `setpc`.

In questo modo possiamo disegnare forme divertenti, come per esempio quella creata dal programma seguente. Dapprima diamo un nome a due programmi che disegnano due cerchi di grandezza diversa:

```
to CERCHIO3
repeat 360 [fd 3 rt 1]
end

to CERCHIO1
repeat 360 [fd 1 rt 1]
end
```

Ora utilizziamo questi cerchi per disegnare delle forme simili alle precedenti:

```
to FORMA3
repeat 36 [CERCHIO3 rt 10]
end

to FORMA1
repeat 18 [CERCHIO1 rt 20]
end
```

Ora proviamo ad aggiungere un po' di colore:

```
setpc 2
FORMA3 rt 2
setpc 3
FORMA3 rt 2

setpc 4
FORMA3 rt 2
setpc 5
FORMA3 rt 2

setpc 6
FORMA1 rt 2
setpc 15
FORMA1 rt 2

setpc 8
FORMA1 rt 2
setpc 9
FORMA1 rt 2
```

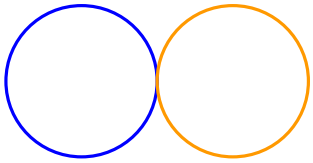
Se vuoi, puoi continuare questo lavoro e disegnare ancor più. Oppure disegna una forma di tua fantasia.

Esercizio 45

Disegna `FORMA3` in arancio. In seguito utilizza l'istruzione **`setpc 7`** per passare al colore bianco. Che cosa succede se esegui di nuovo `FORMA3`?

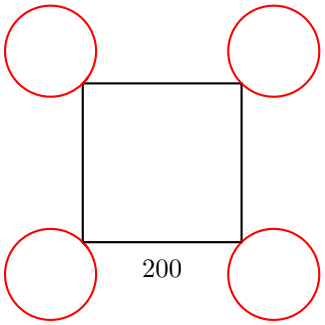
Esercizio 46

Disegna l'immagine seguente. All'inizio la tartaruga si trova nel punto d'intersezione dei due cerchi (cioè dove essi si toccano).



Esercizio 47

Scrivi un programma per disegnare l'immagine seguente. Puoi scegliere tu la grandezza dei cerchi.



5 Programmi con parametri

Nella Lezione 3 abbiamo imparato a dare un nome ai programmi e a chiamarli col loro nome per far disegnare l'immagine desiderata al computer. Nella Lezione 4 abbiamo imparato a disegnare dei poligoni regolari. Tuttavia, è molto ripetitivo dover scrivere un nuovo programma per ogni poligono regolare a dipendenza del numero degli angoli.

Osserviamo per esempio i tre programmi seguenti:

```
repeat 7 [fd 50 rt 360/7]
repeat 12 [fd 50 rt 360/12]
repeat 18 [fd 50 rt 360/18]
```

I programmi si assomigliano molto e differiscono soltanto per i numeri evidenziati in giallo `7`, `12` e `18`. Questi numeri determinano il numero di angoli. Ora vorremmo scrivere un programma col quale disegnare un qualsiasi poligono regolare:

```
to POLIGONO :ANGOLI
repeat :ANGOLI [fd 50 rt 360/:ANGOLI]
end
```

Che cosa abbiamo fatto? In ogni parte del programma abbiamo sostituito il numero di angoli con un nome, in questo caso `:ANGOLI`. Affinché il computer sappia fin dall'inizio che vogliamo scegliere il numero di angoli più tardi, dopo il nome del programma dobbiamo scrivere anche `ANGOLI` preceduto da `:`.

Se ora scriviamo l'istruzione `POLIGONO 12` nella riga di comando, il computer sostituirà `:ANGOLI` col numero `12` dappertutto nel programma

```
repeat :ANGOLI [fd 50 rt 360/:ANGOLI]
       12                    12
```

disegnando così un dodecagono.

Prova ad eseguire

```
POLIGONO 3
POLIGONO 4
POLIGONO 5
POLIGONO 6
```

Diciamo che `:ANGOLI` è un **parametro**. Nell'esempio sopra, 3, 4, 5 e 6 sono i **valori del parametro** `:ANGOLI`. Il computer riconosce un parametro grazie a `:`. Per questa ragione bisogna scrivere `:` prima del nome di ogni parametro.

Esercizio 48

I programmi seguenti disegnano dei quadrati con lati di lunghezza diversa tra loro:

```
repeat 4 [fd 100 rt 90]
repeat 4 [fd 50 rt 90]
repeat 4 [fd 200 rt 90]
```

Possiamo considerare i numeri evidenziati in giallo, 100, 50 e 200, come valori di un parametro che descrive la lunghezza dei lati del quadrato. Scrivi un programma con un parametro `:LATO` per disegnare un quadrato di grandezza qualsiasi:

```
to QUADRATO :LATO
...
end
```

Esercizio 49

I programmi seguenti disegnano cerchi di grandezza diversa tra loro:

```
repeat 360 [fd 1 rt 1]
repeat 360 [fd 12 rt 1]
repeat 360 [fd 3 rt 1]
```

Scrivi un programma con un parametro per disegnare dei cerchi di grandezza qualsiasi. Prova ad eseguirlo usando 1, 2, 3, 4 e 5 come valori del parametro. Puoi scegliere tu il nome del programma e del parametro. Fai attenzione a scrivere sempre i due punti prima del parametro.

Esercizio 50

Ti ricordi ancora come si disegnano delle linee spesse (Esercizio 28)? Scrivi un programma con un parametro per disegnare delle linee spesse di lunghezza qualsiasi.

Suggerimento: Scrivi dapprima un programma per disegnare una linea di lunghezza 100 e un programma per disegnare una linea di lunghezza 50 per capire dove inserire il parametro.

Esercizio 51

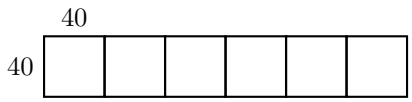
Scrivi un programma con un parametro per disegnare un triangolo di grandezza qualsiasi. In seguito utilizza questo programma per disegnare uno dopo l'altro dei triangoli di grandezza

20, 40, 60, 80, 100, 120, 140, 160 e 180.

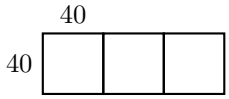
Che cosa compare sullo schermo?

Esercizio 52

Ora vogliamo disegnare dei quadrati di lato 40 uno di fianco all'altro. Scrivi un programma `QUADRATI` con un parametro `:NUM`. Il parametro `:NUM` determina il numero di quadrati da disegnare. Se quindi eseguiamo `QUADRATI` 6, la tartaruga deve disegnare l'immagine seguente:

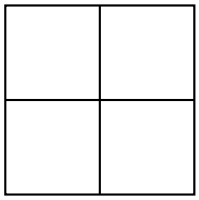


Ecco cosa compare se eseguiamo `QUADRATI` 3:



Esercizio 53

Scrivi un programma per disegnare l'immagine seguente costituita da 4 quadrati. Utilizza un parametro per determinare la lunghezza del lato del quadrato.

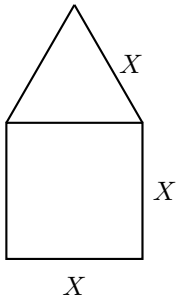


Esercizio 54

Scrivi un programma con un parametro che disegna degli esagoni (poligoni regolari con 6 lati) di lato qualsiasi. Prova ad eseguire questo programma per disegnare degli esagoni di lato 40, 60 e 80.

Esercizio 55

Scrivi un programma con un parametro `:X` per disegnare delle case di grandezza qualsiasi, come nell'immagine seguente.



Programmi con più parametri

Un programma può avere più di un parametro. Se vogliamo disegnare dei poligoni, possiamo definire un parametro `:ANGOLI` per il numero di angoli e un parametro `:LATO` per la lunghezza dei lati.

Nei programmi seguenti il parametro `:ANGOLI` è evidenziato in giallo e il parametro `:LATO` in verde:

```
repeat 13 [fd 100 rt 360/13]
repeat 3 [fd 300 rt 360/3]
repeat 17 [fd 10 rt 360/17]
repeat 60 [fd 3 rt 360/60]
```

Il prossimo programma ci permette di disegnare tutti i poligoni che vogliamo:

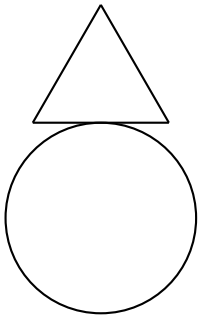
```
to POLIGONI :ANGOLI :LATO
repeat :ANGOLI [fd :LATO rt 360/:ANGOLI]
end
```

Testa il programma `POLIGONI` con le istruzioni seguenti:

```
POLIGONI 12 60
POLIGONI 12 45
POLIGONI 8 30
POLIGONI 9 30
POLIGONI 7 31
POLIGONI 11 50
```

Esercizio 56

Scrivi un programma con due parametri per disegnare l'immagine seguente. La grandezza del cerchio e quella del triangolo devono poter essere scelte liberamente.



Esercizio 57

Il programma

```
fd 100 rt 90 fd 200 rt 90 fd 100 rt 90 fd 200
```

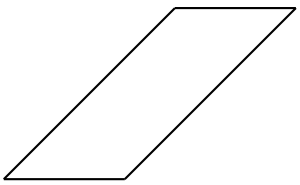
disegna un rettangolo di altezza 100 e larghezza 200. Verifica che funziona e scrivi un programma con due parametri per disegnare un rettangolo di altezza e larghezza qualsiasi.

Esercizio 58

Il programma seguente

```
repeat 2 [rt 45 fd 200 rt 45 fd 100 rt 90]
```

disegna un parallelogramma:



Scrivi un programma con due parametri per disegnare un tale parallelogramma con i lati di lunghezza qualsiasi.

Esercizio 59

Per disegnare un fiore, disegna dapprima un cerchio con

```
POLIGONI 360 2
```

poi gira un po' la tartaruga con

```
rt 20
```

e in seguito disegna di nuovo un cerchio con

```
POLIGONI 360 2
```

e continua così con `rt 20 POLIGONI 360 2 rt 20 POLIGONI 360 2 ...`

Quando hai finito di disegnare il fiore, la tartaruga si trova di nuovo nella posizione iniziale. La tartaruga ha disegnato quindi 18 cerchi, girandosi dopo ciascuno di 20°, per una rotazione totale di $18 \times 20° = 360°$.

Riassumendo, abbiamo ottenuto il programma seguente:

```
repeat 18 [POLIGONI 360 2 rt 20]
```

Prova ad eseguirlo.

(a) È anche possibile disegnare dei fiori con 10 oppure 20 petali (cerchi). Come faresti? Scrivi un programma e prova ad eseguirlo.

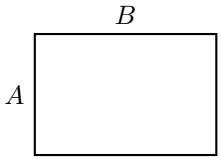
(b) Riesci a scrivere un programma con un parametro per disegnare dei fiori con un numero qualsiasi di petali (cerchi)?

(c) Riesci a scrivere un programma coi parametri seguenti:

- il numero di petali (cerchi) e
- la grandezza dei cerchi?

Esercizio 60

Scrivi un programma per disegnare un rettangolo qualsiasi con un colore qualsiasi:



Ciò significa che devi poter scegliere i lati A e B e anche il colore.

6 Disegnare dei fiori e passare dei parametri a un sottoprogramma

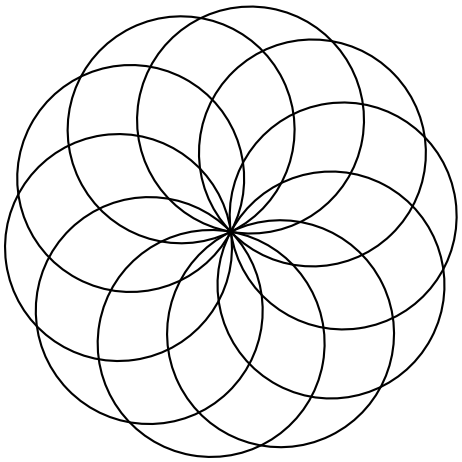
In questa lezione impariamo a disegnare dei fiori. Con l'aiuto dei parametri, cambieremo la forma ed i colori dei fiori a nostro piacimento. La tartaruga disegnerà figure belle, colorate ed estremamente fantasiose.

Diamo un'occhiata al programma seguente:

```
to CERCHI :GR
repeat 360 [fd :GR rt 1]
end
```

Copiamo il programma nell'editor. Per disegnare un fiore con 10 petali possiamo ora utilizzare il programma:

```
repeat 10 [CERCHI 1 rt 36]
```



Esercizio 61

Paolo vuol disegnare un fiore con 24 petali. Come deve cambiare il programma che abbiamo scritto qui sopra?

Esercizio 62

Disegna un fiore con 12 petali. La grandezza dei petali deve essere il doppio rispetto a quelli disegnati nell' esercizio precedente.

Adesso desideriamo scrivere un programma nell'editor, che ci permetta di disegnare dei fiori e di scegliere a nostro piacimento la grandezza dei petali. Utilizziamo il sottoprogramma `CERCHI :GR`, scegliendo liberamente il valore da assegnare a `:GR`. Il programma che disegna il fiore deve perciò contenere il parametro che determina la grandezza dei petali.

Scrivi nell editor:

```
to FIORE :DI
repeat 10 [CERCHI :DI rt 36]
end
```

Esegui `FIORE 1`, `FIORE 2`, `FIORE 3` e osserva attentamente la figura che vien disegnata. Cosa è successo? Quando abbiamo eseguito `FIORE 1`, il valore 1 è stato assegnato al parametro `:GR`. Il sottoprogramma `CERCHI :GR` è stato quindi eseguito come `CERCHI 1`.

Esercizio 63

Descrivi cosa succede quando esegui il programma `FIORE 2`.

Esercizio 64

Cosa fa il programma seguente? Rifletti bene, prima di eseguirlo.

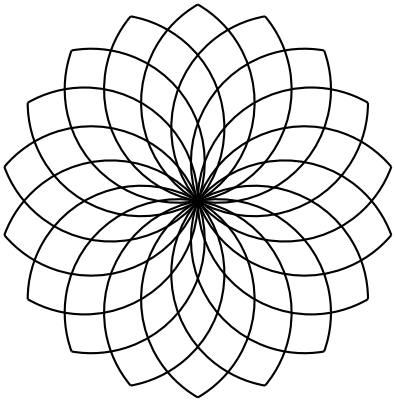
```
to FIORI :GR1 GR2
setpc 3 FIORE :GR1
setpc 4 FIORE :GR2
end
```

Esercizio 65

Vogliamo cambiare il programma `FIORE` in `FIORE1` in modo tale che si possano scegliere liberamente non solo la grandezza, ma anche il numero di petali. Come si fa?

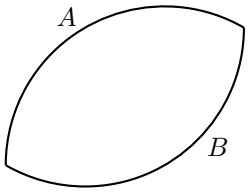
Un fiore con petali aguzzi

Vuoi imparare a disegnare un fiore con dei petali belli aguzzi? Ti piace, per esempio, questo fiore?



Per disegnare un fiore simile, dobbiamo prima di tutto capire, come disegnare un singolo petalo.

Per ottenere un petalo



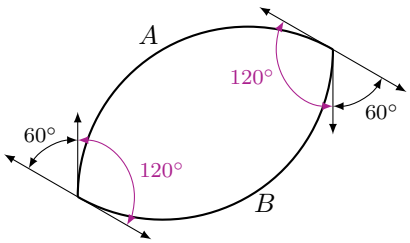
incolliamo due archi di cerchio A e B. Possiamo disegnare un arco di cerchio, per esempio, con il programma che segue:

```
repeat 120 [fd 2 rt 1]
```

Provalo.

Ci accorgiamo che questo programma assomiglia molto a quello per disegnare un cerchio. Invece di 360 piccoli movimenti seguiti da rotazioni di 1°, eseguiamo soltanto 120 volte `[fd 2 rt 1]` disegnando così solo un terzo del cerchio (120°).

La prossima domanda che ci poniamo è: di quanto bisogna girare la tartaruga prima di cominciare a disegnare l'arco di cerchio B per la parte inferiore del petalo? Guardiamo un pò l'immagine seguente:



Se alla fine vogliamo tornare alla posizione da cui siamo partiti, in tutto dobbiamo girare la tartaruga di 360°. Sia nella parte A, che nella parte B giriamo la tartaruga di 120°. Quindi rimangono ancora

$$360° - 120° - 120° = 120°$$

da dividere in parti uguali tra le due rotazioni sulle punte dei due petali:

$$\frac{120°}{2} = 60°.$$

Il risultato è il programma seguente:

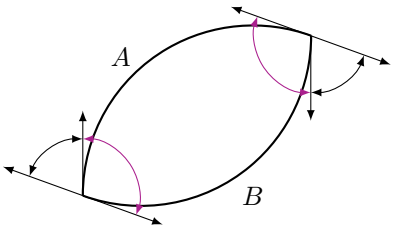
```
repeat 120 [fd 2 rt 1]
rt 60
repeat 120 [fd 2 rt 1]
rt 60
```

oppure più semplicemente:

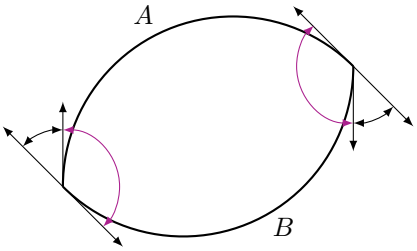
```
repeat 2 [repeat 120 [fd 2 rt 1] rt 60]
```

Provalo.

Se vogliamo, possiamo anche disegnare foglie più strette (le parti A e B sono più corte) oppure più larghe (le parti A e B sono più lunghe).



PARTE=120°



PARTE=135°

Anche in questo caso possiamo utilizzare un parametro, che possiamo per esempio chiamare :PARTE. Per calcolare la rotazione sulla punta del petalo procediamo come segue: prima di cominciare a disegnare la parte B del petalo, dobbiamo aver già eseguito metà della rotazione completa, cioè $\frac{360°}{2} = 180°$. Perciò la rotazione sulla punta del petalo è

$$180° - \texttt{:PARTE}.$$

Grazie a quest'ultima indicazione possiamo scrivere il programma seguente nell'editor:

```
to PETALO :PARTE
repeat 2 [repeat :PARTE [fd 2 rt 1] rt 180-:PARTE]
end
```

Prova il programma scrivendo le istruzioni seguenti nella riga di comando:

```
PETALO 20
PETALO 40
PETALO 60
PETALO 80
PETALO 100
```

Che cosa succede?

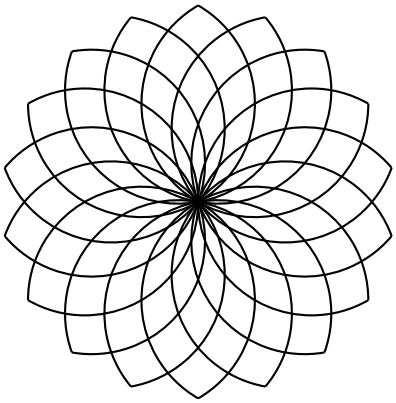
Un fiore è composto da tanti petali aguzzi

Arrivati a questo punto desideriamo utilizzare `PETALO` come sottoprogramma per disegnare fiori con petali aguzzi.

Esercizio 66

Disegna dapprima un fiore col programma seguente:

```
PETALO 100
rt 20
PETALO 100
rt 20
PETALO 100
....
```



Quante volte devi ripetere le istruzioni `PETALO` e `rt 20` per completare il fiore?

Scrivi il programma per disegnare il fiore in una riga sola utilizzando l'istruzione **`repeat`** adeguata. (Ricorda che la somma delle rotazioni **`rt`** tra i singoli petali deve dare in tutto 360°.)

Esercizio 67

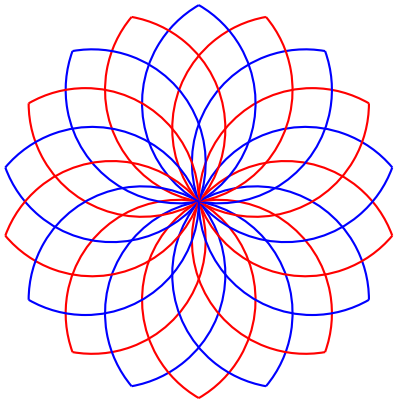
Copia il programma dell'Esercizio 66 nell'editore e dagli il nome `FIORE3`. Il programma deve avere il parametro `:PARTE`. Che cosa succede se scrivi `FIORE3 60`, `FIORE3 80` e `FIORE3 100` nella riga di comando?

Esercizio 68

(a) Scrivi un programma con un parametro che disegni il fiore dell'Esercizio 66 in un colore a scelta. Chiamalo `FIORE4`.

(b) Modifica il tuo programma (e chiamalo `FIORE5`) in modo tale che il numero di petali da disegnare venga scelto con il nuovo parametro `:NUM`. Ricorda che la somma delle rotazioni `rt` tra i singoli petali deve dare in tutto 360°.

(c) Modifica il programma `FIORE5` in modo tale che il fiore abbia due colori, da scegliersi a piacimento. Chiamalo `FIORE6`.



Esercizio 69

Nel programma `PETALO` il comando `fd 2` determina le dimensioni del cerchio dal quale ritagliamo un arco di lunghezza `:PARTE`. Il valore concreto 2 può essere sostituito da un parametro `:GR` (Grandezza). Scrivi il programma

```
PETALI :PARTE :GR
```

con i parametri `:PARTE` e `:GR`, che ci permettono di scegliere l'arco di cerchio e la grandezza dei petali. Prova il programma secondo le indicazioni seguenti:

```
PETALI 100 1
PETALI 100 1.5
rt 100
PETALI 80 2
PETALI 80 2.5
```

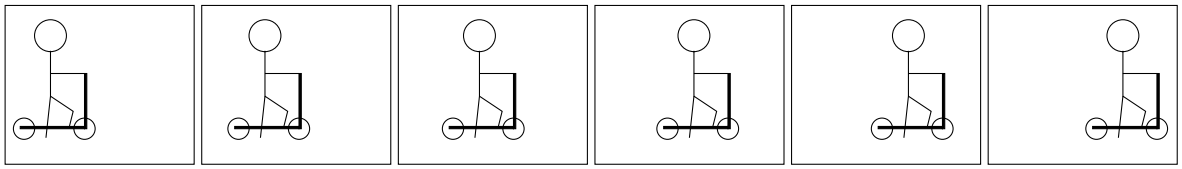
Gira poi la tartaruga verso destra di 80° e ripeti il programma soprastante.

Esercizio 70

Inventa altre figure fantasiose.

7 Programmare animazioni

Sai come viene prodotto un cartone animato? Funziona esattamente come per il folioscopio. Prima si disegnano un paio di immagini che differiscono soltanto un po' l'una dall'altra. Nella sequenza successiva, per esempio, il ragazzo sul monopattino si sposta sempre di un po' tra un'immagine e la successiva:

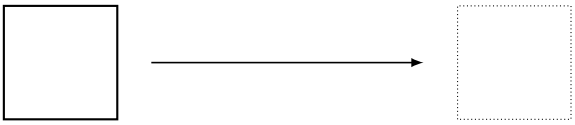


Quando mettiamo le immagini una sopra l'altra e le sfogliamo velocemente col pollice, abbiamo l'impressione che il ragazzo si sposta sul suo monopattino da sinistra a destra. Immagini in movimento vengono chiamate **animazioni**.

In questa lezione impariamo come programmare un'animazione con l'aiuto della tartaruga.

Un quadrato che lascia delle tracce

Per la nostra prima animazione scegliamo una figura che non è troppo difficile e che conosciamo già da molto: faremo spostare un quadrato da sinistra verso destra.



Conosciamo già il programma per disegnare il quadrato:

```
to QUAD100
repeat 4 [fd 100 rt 90]
end
```

Dopo aver disegnato il quadrato una volta, spostiamo la tartaruga un po' a destra e disegniamo il quadrato ancora una volta. Poi spostiamo la tartaruga di nuovo a destra e disegniamo nuovamente il quadrato. Ripetiamo questo procedimento un paio di volte.

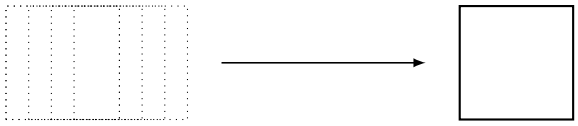
Nel programma seguente disegniamo 120 volte questo quadrato:

```
to QUADANIM
repeat 120 [QUAD100 rt 90 fd 4 lt 90]
end
```

Esercizio 71

Scrivi i programmi `QUAD100` e `QUADANIM` nell'editor ed esegui `QUADANIM`. Che cosa compare?

Puoi ben vedere che vengono disegnate le tracce di *tutti* i quadrati. In un'animazione però vogliamo vedere soltanto l'ultimo quadrato e cancellare le altre tracce.

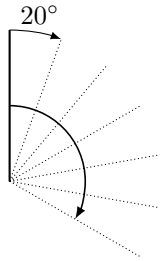


Esercizio 72

Fai muovere il quadrato dal basso verso l'altro invece che da sinistra a destra.

Esercizio 73

Scrivi un programma per disegnare una linea di lunghezza 20. Utilizza questo programma per ruotare una linea in senso orario attorno alla sua estremità inferiore:



Disegnare e cancellare un quadrato

Per far sparire le tracce, dobbiamo imparare a cancellare le figure dopo averle disegnate. A tal fine, la tartaruga deve utilizzare una gomma invece che una matita. Con l'istruzione **penerase** oppure più brevemente **pe**, la tartaruga impugna la gomma al posto della matita.

Esercizio 74

Rifletti su cosa fa il programma `QUAD100 pe QUAD100` senza però eseguirlo al computer.

E se ora vogliamo che la tartaruga riprenda a disegnare? Anche per questa situazione è disponibile un nuovo comando: **`penpaint`** oppure semplicemente **`ppt`**. Utilizziamo subito questo nuovo comando per il programma dell'Esercizio 74.

Il nostro programma ha ora questo aspetto:

```
QUAD100 pe QUAD100 ppt
```

Esercizio 75

Esegui il programma indicato sopra. Che cosa succede? Riesci a spiegartelo?

Il quadrato deve aspettare un po'

Come sicuramente avrai già notato nell'Esercizio 75, il quadrato scompare subito dopo averlo disegnato. Non ci accorgiamo nemmeno che il quadrato è stato disegnato. Prima di cancellare il quadrato, dobbiamo quindi ordinare al computer di fare una piccola pausa.

Possiamo fare ciò come segue:

`wait`	`4`
Istruzione per aspettare	Tempo di attesa

Esercizio 76

Prova il programma

```
QUAD100 wait 4 pe QUAD100 ppt
```

Un quadrato che si muove da sinistra a destra

Adesso siamo pronti a includere le istruzioni per cancellare il quadrato e per la pausa nel nostro programma QUADANIM:

```
to QUADANIM
repeat 120 [QUAD100 wait 4 pe QUAD100 rt 90 fd 4 lt 90 ppt]
end
```

Prova il programma. Se durante l'animazione la tartaruga ti dà fastidio inserisci l'istruzione **hideturtle** (oppure brevemente: **ht**) all'inizio del programma, per nasconderla. Ti accorgerai che l'animazione diventerà più veloce. Inserisci l'istruzione **showturtle** (oppure brevemente: **st**) alla fine del programma, prima di **end**. Così facendo la tartaruga comparirà di nuovo.

Esercizio 77

Fai spostare un quadrato di grandezza 50×50 verso l'altro.

Esercizio 78

Modifica il programma QUADANIM in modo tale che il quadrato si sposti verso destra con velocità doppia.

Esercizio 79

Riesci anche a modificare il programma QUADANIM in modo tale che il quadrato si sposti verso destra con velocità dimezzata?

Esercizio 80

Modifica il programma QUADANIM in modo tale che il quadrato si sposta da destra a sinistra invece che da sinistra a destra.

Esercizio 81

Rifletti sul programma seguente e in seguito eseguilo per verificare la tua idea:

```
to QUADANIM1
ht
repeat 50 [QUAD100 wait 5 pe QUAD100 fd 3 rt 90 fd 3 lt 90 ppt]
QUAD100
st
end
```

Esercizio 82

Rifletti sul programma seguente e in seguito eseguilo per verificare la tua idea:

```
to CERCHI
ht
repeat 360 [QUAD100 wait 4 pe QUAD100 fd 5 rt 1 ppt]
QUAD100
st
end
```

Esercizio 83

Modifica il programma `CERCHI` in modo tale che il quadrato si muova al quadruplo della velocità.

Esercizio 84

Che cosa fa il programma seguente?

```
repeat 6 [CERCHI]
```

Esercizio 85

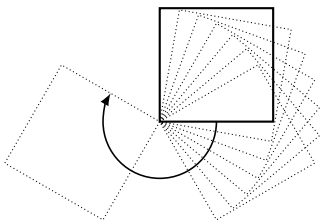
Prendi il programma seguente

```
to TERRA
repeat 45 [fd 16 rt 8]
end
```

e utilizzalo per creare un'animazione in cui la terra si muove lungo un cerchio attorno al sole. Utilizza la tua fantasia per rappresentare il sole in modo appropriato.

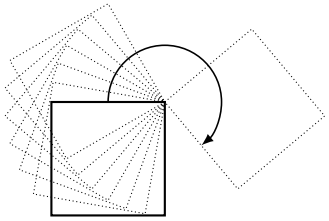
Esercizio 86

Fai ruotare un quadrato in senso orario attorno al suo vertice in basso a sinistra. Puoi scegliere tu la lunghezza del lato:



Esercizio 87

Ora fai ruotare il quadrato in senso orario attorno al suo vertice in alto a destra:



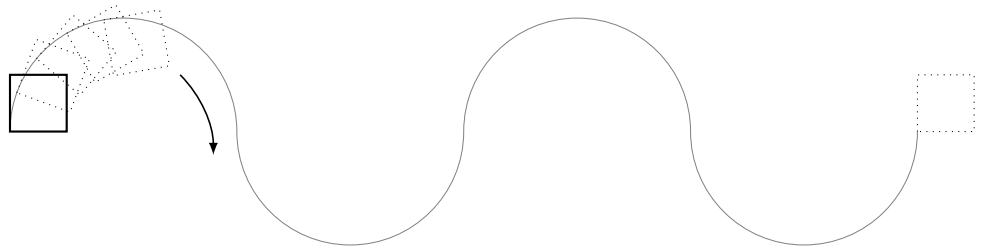
Se già sai cosa sono i parametri, puoi risolvere gli esercizi seguenti.

Esercizio 88

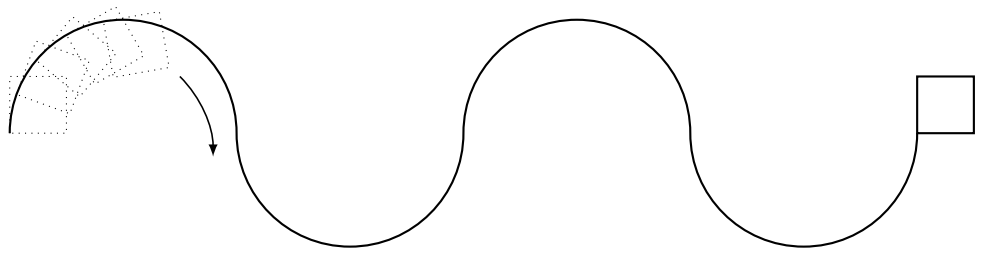
Scrivi un programma con *due parametri* per far muovere un quadrato da sinistra a destra. Il primo parametro deve determinare la lunghezza del lato, il secondo parametro la velocità con cui si muove il quadrato.

Esercizio 89

(a) Fai muovere un quadrato lungo la traiettoria qui sotto rappresentata e composta da 4 semicerchi. La lunghezza del lato del quadrato deve essere stabilita da un parametro.



(b) Ora vogliamo che anche la traiettoria venga disegnata come una traccia man mano che il quadrato si sposta.



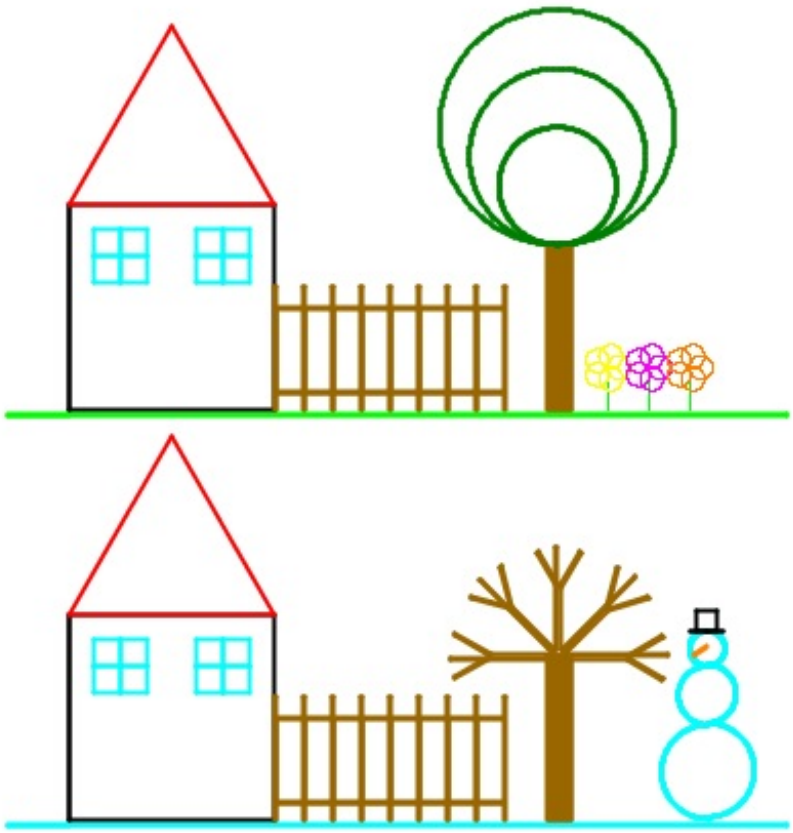
(c) Riesci a estendere il programma in (b) in modo tale che anche il numero dei semicerchi sia determinato da un parametro?

I miei appunti

I miei appunti

Lista delle istruzioni

`fd 100`	andare 100 passi in avanti
`bk 50`	andare 50 passi indietro
`cs`	cancellare tutto e cominciare da capo
`rt 90`	girare di 90 gradi a destra
`lt 90`	girare di 90 gradi a sinistra
`repeat 4 [...]`	il programma contenuto in `[...]` viene ripetuto 4 volte
`pu`	la tartaruga passa alla modalità di spostamento
`pd`	la tartaruga torna alla modalità di disegno
`setpc 3`	prendere il colore numero 3
`to NOME`	creare un programma con un nome
`to NOME :PARAMETRO`	creare un programma con un nome e un parametro
`end`	tutti i programmi con un nome terminano con quest'istruzione
`pe`	la tartaruga passa alla modalità gomma
`ppt`	la tartaruga torna alla modalità di disegno
`wait 5`	la tartaruga aspetta 5 unità di tempo



Programmare con LOGO

Informationstechnologie und Ausbildung
ETH Zürich, CAB F 15.1
Universitätstrasse 6
CH-8092 Zürich

www.ite.ethz.ch
www.abz.inf.ethz.ch